# On Genetic Algorithm Methods for Fast and Efficient Multi-Layer Network Planning under Quality-of-Service Constraints

Von der Fakultät für Informatik, Elektrotechnik und Informationstechnik
der Universität Stuttgart zur Erlangung der Würde
eines Doktor-Ingenieurs (Dr.-Ing.) genehmigte Abhandlung

vorgelegt von

Uwe Bauknecht

geb. in Filderstadt

| | |
|---|---|
| Hauptberichter: | Prof. Dr.-Ing. Andreas Kirstädter |
| Mitberichter: | Prof. Dr.-Ing. Thomas Bauschert (TU Chemnitz) |

Tag der mündlichen Prüfung: 17. Juni 2024

Institut für Kommunikationsnetze und Rechnersysteme
der Universität Stuttgart

2025

# Abstract

Network Service Providers (NSPs) and especially Internet Service Providers (ISPs) will face disruptive changes to their established mode of operation due to several factors. New access technologies like Fiber-To-The-x (FTTx) and fifth generation cellular mobile communications (5G) will enable novel use cases, which will require not only increased bandwidths, but also more precise guarantees for Quality of Service (QoS). Many future applications will require highly reliable, ultra-low delay and arbitrarily flexible services fueled by virtualization, Internet of Things (IoT) and industry 4.0. Especially for cloud-based applications, orchestration of computing and storage will be inextricable from network operation.

This puts forward a considerable challenge for NSP networks since their design is traditionally centered around services with long holding times and therefore uses overdimensioning of capacity as a means to assure the necessary quality for common services while dedicated resources are used for the smaller number of services with high QoS requirements. While earlier stages of networks such as the access and aggregation areas can still benefit from adopting faster technologies, especially transport networks cannot be overdimensioned feasibly in a similar fashion since they are already utilizing devices at the pinnacle of achievable capacity. The only remaining option in these parts of the network is to employ methods in order to maximize the efficiency of network resources. As traditional scale-up approaches cease to be effective, NSPs are prompted to adopt a network-wide scale-out strategy combined with a random interconnection network operation paradigm enabling utmost flexibility and efficiency. Future services on this platform will not specify technological parameters, but simply the required QoS while the platform itself will provide the exact bandwidth, latency, and availability figures custom-tailored for each individual request, provisioned at a moment's notice at an optimal resource usage to the operator. The network itself will be highly automated, shifting the focus of the network operators' personnel from maintenance to service design.

Core networks based on optical transport technologies with electrical routing layers on top will develop towards a consolidated model of two layers able to provide a maximum of flexibility and performance. Until optical packet switching technology becomes available, a packet optical paradigm combining the bandwidth and low delay of optical technologies with the efficiency and flexibility of packet-switched networks will be imperative. Enabling such a dynamic service-oriented network operation requires solving highly complex combinatorial problems considering all aforementioned aspects. Traditional network planning relies on many disjoint algorithms often working on heterogeneous representations of the same network state. Each algorithm therefore solves its subproblem according to its own definition of optimality without being able to consider effects of co-dependence with other algorithms thereby leading to a sub-par solution considering the complete network information. An integrated approach on the other hand could consider the problem in a holistic way and minimize or even eliminate safety margins required to assemble a complete solution from different algorithms' results. The

drawback is the much increased problem complexity which is prohibitive to traditional solutions such as mathematical optimization.

In this thesis several novel solution approaches based on a well-established metaheuristic method called Genetic Algorithms will be introduced, which can not only be used in network planning, but also for dynamic network operation. Genetic Algorithms encode an abstract problem as a genetic code and apply the principle of natural evolution due to mutation and crossover of genes under selection pressure as a means to solve a given problem. While the basic algorithm has been used for network optimization in the past, the approach given here will adapt the principle and combine several encoding schemes with novel operators and acceleration techniques which are specifically designed to be highly efficient in solving even the complex, large-scale problems associated with multi-layer network optimization under QoS constraints. The approach leverages graph properties common to all such problems and builds on this knowledge to provide a highly efficient and scalable optimization platform considering capabilities of future networking devices.

The performance of these algorithms will be systematically evaluated based on several approaches. The baseline reference will be the disjoint solution approach commonly used today with operators while a second metaheuristic approach based on Simulated Annealing, which has been successfully used in research applications in the past, will serve as reference to determine the quality of the optimization results. The evaluation is carried out based on future network layouts and architectures considering different use cases. Network load figures are generated from publicly available load and population statistics and will allow to investigate realistic traffic profiles with diverse network services.

# Kurzfassung

Internetdienstanbieter (*engl.* Internet Service Provider, kurz ISP) werden sich in naher Zukunft mit disruptiven Änderungen an deren etablierten Geschäftsprozessen bedingt durch diverse Faktoren konfrontiert sehen. Neue Netzzugangstechnologien wie Fiber-To-The-x und Mobilfunkdienste der fünften Generation (5G) werden nicht nur deutlich gesteigerte Bandbreiten, sondern auch präzisere Garantien für Dienstgüte (*engl.* Quality of Service, kurz QoS) erfordern. Weiter werden künftige Anwendungen hochverfügbare, hochflexibel ausgestaltbare Dienste von geringster Latenz benötigen, was durch Cloudifizierung, das Internet der Dinge (*engl.* Internet of Things, kurz IoT) und Industrie 4.0 zusätzlich an Relevanz erlangt. Die Orchestrierung von Berechnung und Speicherung von Informationen wird nicht zuletzt aufgrund von Netzfunktionsvirtualisierung (*engl.* Network Function Virtualization, kurz NFV) untrennbar mit dem Netzbetrieb verwoben.

Dies resultiert in einer beträchtlichen Herausforderung für ISP Netze, da sich deren Design traditionell an Diensten langer Haltezeiten orientiert und aufgrund dessen Überdimensionierung von Kapazitäten als Methode eingesetzt werden kann, um die geforderte Güte für die verbreiteten Standarddienste zu erzielen. Gleichzeitig können seltener angefragte Dienste höherer QoS-Anforderungen auf dedizierte Ressourcen abgebildet werden. Während vorgelagerte Bereiche der Betreibernetze wie die Zugangs- und Aggregationsbereiche weiterhin vermittels schnellerer Übertragungstechnologien in ihrer Leistungsfähigkeit gesteigert werden können, ist dies oftmals kein mögliches Vorgehen für Transportnetze. Dort ist durch eine Überdimensionierung wirtschaftlich sinnvoll kein Mehrgewinn an Leistung zu erzielen, da hier typischerweise bereits alle marktverfügbaren technologischen Möglichkeiten zur Kapazitätssteigerung ausgereizt sind. Die einzig verbleibende Vorgehensweise in diesen Netzbereichen stellt die Anwendung von Methoden dar, welche die Nutzungseffizienz verfügbarer Ressourcen maximieren. Da also Ansätze, welche auf vertikaler Skalierung basieren, nicht mehr effektiv sein werden, erfolgt eine Verschiebung hin zu einer netzweiten horizontalen Skalierungsstrategie in Kombination mit einem dynamisch anpassbaren Netzbetriebsparadigma, welches so die größtmögliche Flexibilität und Effizienz ermöglicht. Künftige Dienste auf dieser Plattform werden nicht mehr technologiespezifische Parameter angeben, sondern sich explizit auf Angeben zur Dienstgüte beziehen. Dies erlaubt es der Plattform, selbstständig die exakt benötigten Bandbreiten-, Latenz- und Verfügbarkeitswerte spezifisch für jede individuelle Anforderung zu bestimmen und diese so in kurzer Zeit unter optimaler Ressourcennutzung durch den Betreiber zur Verfügung zu stellen. Das Netz selbst wird dabei hochautomatisiert sein und den Arbeitsschwerpunkt der Betriebsmannschaft des Internetproviders von der Instandhaltung der Systeme hin zur Definition und Ausgestaltung von Diensten verschieben.

Kernnetze, welche auf optischen Übertragungstechnologien in Kombination mit einer elektrischen Routing-Schicht basieren, werden sich hin zu einem konsolidierten Zweischichtenmodell entwickeln. Dies erlaubt ein Maximum an Flexibilität und Leistung. Solange rein optische

Paketvermittlung noch keine praktisch verfügbare Technologie ist, bleibt ein Paket-Optisches Designparadigma, welches die hohen Bandbreiten und niedrige Latenzzeiten optischer Technologien mit der Effizienz und Flexibilität eines paketvermittelten Netzes kombinieren, die einzige Alternative. Um einen derartig dynamischen und dienst-orientierten Netzbetrieb zu ermöglichen, müssen hochkomplexe kombinatorische Probleme unter Berücksichtigung der vorgenannten Aspekte gelöst werden können. Herkömmliche Netzplanung setzt dabei eine Vielzahl unabhängiger Algorithmen ein, welche teils mit sehr heterogenen Darstellungen desselben Netzzustandes arbeiten. Jeder Algorithmus löst dabei sein eigenes Teilproblem entsprechend seiner jeweiligen Definition eines optimalen Zustandes. Es ist in einem solchen Ansatz daher kaum möglich, die engen Zusammenhänge der Kopplung zwischen den Problemlösungen einzelner Algorithmen zu berücksichtigen, sodass die entstehende Gesamtlösung hinter ihren Möglichkeiten zurückbleibt. Ein integrierter Lösungsansatz hingegen könnte das Problem holistisch betrachten und so zur Reduktion oder gar Vermeidung von unnötigen Sicherheitsmargen beitragen, welche sonst nötig wären. Der Nachteil dieses Ansatzes ist eine stark gesteigerte kombinatorische Komplexität, welche die Fähigkeiten traditioneller Ansätze wie mathematischer Optimierung übersteigen dürfte.

In dieser Arbeit werden neuartige Lösungsansätze, basierend auf einer etablierten Metaheuristik in Form von genetischen Algorithmen, eingeführt, welche nicht nur eine statische Netzplanung, sondern auch eine dynamische Anpassung im Betrieb ermöglichen können. Genetische Algorithmen bilden dabei ein abstraktes Problem als genetischen Code ab und verwenden natürliche Evolutionsprozesse wie Mutation und Rekombination von Genen unter Selektionsdruck, um ein gegebenes Problem zu lösen. Während grundlegende Varianten genetischer Algorithmen bereits in der Vergangenheit zur Optimierung von Netzen angewandt wurden, werden die vorgestellten Ansätze diese Grundprinzipien anpassen und weiterentwickeln. Die Kombination verschiedener problemspezifischer Kodierungsansätze und neuartiger Operatoren, ermöglicht so einen Ansatz zur effizienten Lösung von komplexen Optimierungsproblemen im Kontext von Multi-Layer-Netzen unter Berücksichtigung von Dienstgüteanforderungen. Dieser Ansatz nutzt Eigenschaften von Graphen, wie sie in allen solchen Problemen zu finden sind, um eine hocheffiziente und skalierbare Optimierungsplattform zu schaffen, welche auch die Eigenschaften künftiger Netz-Hardware berücksichtigt.

Die Leistung der entwickelten Algorithmen wird systematisch aus verschiedenen Blickwinkeln ermittelt. Dabei dienen der etablierte Ansatz, mehrere lose gekoppelte Algorithmen zu verwenden, sowie eine zweite Metaheuristik, welche auf Simulated Annealing basiert, als Referenz. Diese zweite Heuristik wurde bereits in der Vergangenheit erfolgreich zu Forschungszwecken eingesetzt und bildet so eine sinnvolle Grundlage für die Bewertung der erreichbaren Lösungsqualität. Der Evaluation liegen dabei verschiedene Netzdesigns zugrunde, wobei auch solche berücksichtigt werden, welche voraussichtlich repräsentativ für künftige Netze sind. Die Verkehrslast wurde basierend auf Messungen und öffentlich verfügbaren Daten über Verkehrsaufkommen und Populationsverteilungen ermittelt und beinhaltet repräsentative Dienstgüteanforderungen.

# Contents

# List of Figures

# List of Tables

# List of Algorithms

# Abbreviations

| | | |
|---|---|---|
| 3R | Reamplification, Reshaping and Retiming | 13 |
| 5G | Fifth generation of cellular mobile communications | 2 |
| 5G PPP | 5G Infrastructure Public Private Partnership | 4 |
| ACO | Ant Colony Optimization | 42 |
| APTI | Augmented Physical Topology Initialization | 83 |
| AR | Augmented Reality | 2 |
| ASIC | Application-Specific Integrated Circuit | 10 |
| ASON | Automatically Switched Optical Network | 55 |
| ASTI | Augmented Spanning Tree Initialization | 83 |
| ATM | Asynchronous Transfer Mode | 4 |
| BGP | Border Gateway Protocol | 55 |
| BRKGA | Biased Random-Key Genetic Algorithm | 50 |
| BUXO | Biased Uniform Crossover Operator | 53 |
| BVT | Bandwidth-Variable Transponder | 13 |
| CAPEX | Capital expenditure | 1 |
| CATI | Combined Augmented Topology Initialization | 109 |
| CNCXO | Cut-and-Crossfill Crossover | 53 |
| CORD | Central Office Re-architected as a Datacenter | 4 |
| CPU | Central Processing Unit | 181 |
| CSP | Cloud Service Provider | 2 |
| CWDM | Course WDM | 170 |
| DC | Data Center | 2 |
| DSL | Digital Subscriber Line | 3 |
| DVFS | Dynamic Voltage and Frequency Scaling | 17 |
| DWDM | Dense WDM | 9 |
| E2E | End-To-End | 2 |
| EA | Evolutionary Algorithm | 44 |
| EBXO | Exponentially Biased Crossover | 105 |

| | | |
|---|---|---|
| FCC | Federal Communications Commission | 24 |
| FEC | Forward Error Correction | 22 |
| FPS | Fitness Proportional Selection | 49 |
| FTTH | Fiber-To-The-Home | 3 |
| FTTx | Fiber-To-The-x | 2 |
| GA | Genetic Algorithm | 45 |
| GMPLS | Generalized Multi-Protocol Label Switching | 170 |
| gNMI | GRPC Network Management Interface | 171 |
| GRASP | Greedy Randomized Adaptive Search Procedure | 62 |
| ICR | Independent Circuit Routing | 24 |
| ICT | Information and Communication Technology | 17 |
| IGP | Interior Gateway Protocol | 171 |
| ILP | Integer Linear Program | 38 |
| ILS | Iterated Local Search | 41 |
| IoT | Internet of Things | 2 |
| IP | Internet Protocol | 4 |
| IPoDWDM | IP over DWDM | 12 |
| IS-IS | Intermediate System to Intermediate System Protocol | 55 |
| ISP | Internet Service Provider | 3 |
| ITU-T | Telecommunication Standardization Sector of the ITU | 7 |
| LBXO | Link Block Crossover Operator | 79 |
| LCoS | Liquid Crystal on Silicon | 11 |
| LHM | Longhorn Mutation | 103 |
| LNB | Link and Node Bias | 63 |
| LP | Linear Program | 38 |
| LSP | Label-Switched Path | 170 |
| LTE | Long-Term Evolution | 3 |
| MCF | Multi-Commodity Flow | 56 |
| MEMS | Microelectromechanical System | 11 |
| MILP | Mixed Integer Linear Program | 38 |
| MPLS | Multiprotocol Label Switching | 4 |
| MPLS-TE | MPLS Traffic Engineering | 170 |
| MPLS-TP | MPLS-Transport Profile | 93 |
| MSLS | Multi-Start Local Search | 40 |
| MTBF | Mean Time Between Failures | 24 |

| | | |
|---|---|---|
| NE | Network element | 8 |
| NETCONF | Network Configuration Protocol | 171 |
| NFV | Network Function Virtualization | 4 |
| NMS | Network Management System | 170 |
| NPM | N-Point Mutation | 52 |
| NPXO | N-Point Crossover | 53 |
| NSP | Network Service Provider | 1 |
| O-E-O | Optical–Electrical–Optical | 11 |
| OAM/OA&M | Operations, administration, and maintenance | 170 |
| OLA | Optical Line Amplifier | 10 |
| OMS | Optical Multiplex Section | 11 |
| ONF | Open Networking Foundation | 4 |
| OPEX | Operating expenditure | 1 |
| OSPF | Open Shortest Path First | 55 |
| OSPF-TE | OSPF - Traffic Engineering | 55 |
| OTN | Optical Transport Network | 171 |
| PCE | Path Computation Element | 171 |
| PCEP | PCE Communication Protocol | 171 |
| PoP | Point of Presence | 8 |
| PSO | Particle Swarm Optimization | 42 |
| PTD | Physical Topology Design | 18 |
| PWE3 | Pseudo Wire Emulation Edge-to-Edge | 170 |
| QoS | Quality of Service | 1 |
| RDRI | Randomized Dual-Range Initialization | 98 |
| RIP | Routing Information Protocol | 55 |
| ROADM | Reconfigurable Optical Add-Drop Multiplexer | 11 |
| RRM | Random Reset Mutation | 52 |
| RSA | Routing and Spectrum Assignment | 24 |
| RSVP-TE | Resource Reservation Protocol - TE | 170 |
| RWA | Routing and Wavelength Assignment | 24 |
| RWS | Roulette Wheel Selection | 49 |
| SDH | Synchronous Digital Hierarchy | 170 |
| SDN | Software-Defined Networking | 4 |
| SLA | Service-Level Agreement | 1 |
| SLCM | Small/Large Creep Mutation | 103 |

| | | |
|---|---|---|
| SLGM | Small/Large Gaussian Mutation | 103 |
| SONET | Synchronous Optical NETworking | 69 |
| SUS | Stochastic Universal Sampling | 49 |
| T-SDN | Transport-SDN | 5 |
| TDM | Time-Division Multiplexing | 23 |
| TDR | Traffic Demand Routing | 22 |
| TE | Traffic Engineering | 170 |
| TXP | Transponder | 10 |
| UPS | Uninterruptible Power Supply | 17 |
| URM | Uniform Random Mutation | 51 |
| UXO | Uniform Crossover | 53 |
| VoD | Video on Demand | 9 |
| VR | Virtual Reality | 2 |
| VSM | VTCS-Specific Mutation | 90 |
| VSXO | VTCS-Specific Crossover | 90 |
| VTD | Virtual Topology Design | 19 |
| WA | Wavelength Assignment | 25 |
| WCS | WDM Circuit Switch | 11 |
| WDM | Wavelength-Division Multiplexing | 4 |
| WR | Wavelength-Routed | 69 |
| WSON | Wavelength Switched Optical Network | 10 |
| WSS | Wavelength-Selective Switch | 11 |
| YANG | Yet Another Next Generation | 171 |

# Symbols and Notation

## Symbols

| | | |
|---|---|---|
| $E_c$ | Set of candidate edges for a graph | 17 |
| $\mathcal{C}$ | Set of conditions | 15 |
| $a_f$ | Fiber availability per km | 144 |
| $m_{\text{LC}}$ | Maximum number of Line Cards in a Rack | 121 |
| $deg_{max}$ | Maximum nodal degree of a vertex in a graph | 192 |
| $\mathcal{N}_1(x)$ | Neighborhood of a point $x$, i. e., set of points with metric distance 1 | 39 |
| $n_{\text{AV}}$ | Number of demands violating their availability constraint | 121 |
| $n_{\text{LV}}$ | Number of demands violating their latency constraint | 121 |
| $n_{\text{LC},v}$ | Number of Line Cards active at node $v$ | 121 |
| $n_{g,\text{VTCS}}$ | Number of genes used in the genetic encoding based on Spanning Trees | 86 |
| $c_{\text{paths}}(n_v)$ | Number of loop-free paths between two nodes in a full graph of $n_v$ nodes | 186 |
| $n_{\text{UD}}$ | Number of unrouted demands | 121 |
| $F$ | Optimization objective function | 15 |
| | | |
| $\delta_f$ | Delay per km of optical fiber | 121 |
| $\delta_h$ | Delay per upper-layer hop | 121 |
| $\kappa_{\text{mod}}$ | Abstract SNDlib module cost value | 136 |
| $(\mu,\lambda)$ | Selection of $\mu$ survivors exclusively from $\lambda$ offspring | 51 |
| $(\mu+\lambda)$ | Selection of $\mu$ survivors from union of $\mu$ parents and $\lambda$ offspring | 51 |

# Notation

| | |
|---|---|
| $a, b, c$ | Minuscules represent individual variables |
| $A, B, C$ | Majuscules represent sets, where not explicitly noted otherwise |
| $A = \{a, \dots\}$ | Curvy brackets represent set enumerations |
| $\wp(A)$ | Power set of $A$, i. e., set of all subsets of $A$ |
| $\langle a, \dots \rangle$ | Angled brackets represent tuples, i. e., finite sequences |
| $\langle a_i \rangle$ | Tuple of $n$ elements $a_i$ where $i \in \{1..n\}$ and $n \in \mathbb{N}$ |
| $V$ | Finite set of vertices |
| $e = \langle s, d \rangle$ | Directed edge $e$ connecting $s$ to $d$ where $s, d \in V$ and $s \neq d$ |
| $E$ | Finite set of edges |
| $g = \langle V, E \rangle$ | Graph $g$ represented as a tuple of a vertex set $V$ and an edge set $E$ where $\forall \langle s, d \rangle \in E : s, d \in V$ |
| $g' = \langle V', E' \rangle$ | Subgraph $g'$ to $g$ where $V' \subseteq V$, $E' \subseteq E$ and $\forall \langle s, d \rangle \in E' : s, d \in V'$ |
| $G = \langle g_i \rangle$ | Multi-layer graph of $n$ layers where $i \in \{1..n\}$ and $g_i = \langle V, E_i \rangle$ |
| $p = \langle e_i \rangle$ | Path of edges $e_i$ such that $\forall e_x = \langle s_x, d_x \rangle : e_{x+1} = \langle d_x, d_{x+1} \rangle$ |
| $f_{\mathbf{Hops}}(g, s, t) = h$ | Function for the minimum number of hops $h$ from node $s$ to node $t$ in graph $g$ |
| $map[x] = y$ | Associative map access providing a value $y$ for a given value $x$ |
| $map[x][y] = z$ | Bivariate associative map access giving $z$ for a tuple $\langle x, y \rangle$ |
| $arr[x] = y$ | Associative map access where the domain is a finite subset of $\mathbb{N}_0$ |
| $\mathbf{RND}(x, y) = r$ | Random number generation returning $r \in \mathcal{U}(x, y)$ as a uniformly distributed number smaller than $y$, but at least as large as $x$ |
| $\mathbf{RNDI}(x, y) = r$ | Random number generation of an integer value $i$ between integers $x$ and $y$ |
| $\mathbf{RND}(X) = x$ | Randomized retrieval of an arbitrary element $x \in X$ from a finite set $X$ |

In geometry, the term vertex formally only applies to points where two or more lines meet. However, in this monograph we will consider the more general usage of the term from graph theory. Therein, the terms node and vertex are interchangeable, and they may or may not be connected by at least one edge.

Associative maps can generally be described by functions mapping from a finite domain to an arbitrary codomain. In computer programming, however, it is possible to iteratively define such a mapping, by explicitly amending a set of relations $M$ between individual elements of the domain to elements of the codomain. This is especially true for the definition of Hashmaps which typically do not require a complete mapping of the domain and therefore a full definition of such maps requires a default element in the codomain to form a well-defined function. This means that formally, an associative map realizes a function $m$ with

$$m : X \to Y \cup \{\bot\},\ x \mapsto y$$

where $X$ and $Y$ are finite sets and $\bot$ signifies the default element, such that

$$m(x) = \begin{cases} y & \text{if } \exists \langle x, y \rangle \in M \\ \bot & \text{otherwise} \end{cases}, \quad M \subseteq X \times Y$$

We also introduce a cardinality for associative maps. For a hypothetical map *mapXY* based on $m$, we define this cardinality, which is given by the number of the contained mappings, as follows.

$$|mapXY| = |M| = |\{m(x) | x \in X \wedge m(x) \neq \bot\}|$$

we typically expect constant time access for our maps, except where explicitly noted otherwise, due to more complex hash functions. Furthermore, we define our maps to provide a deterministic iteration order for loop access. We will consider the contained elements to follow the order of their insertion.

There are two special cases of associative maps to which we apply different assumptions. The first are arrays, which formally realize a function $a$ with

$$a : \{0..n\} \to Y \cup \{\bot\},\ x \mapsto y,\ n \in \mathbb{N}_0$$

where we expect access time complexity to always be in $\mathcal{O}(1)$. The second case are mappings where the codomain is a numeric type including the 0 such as $\mathbb{N}_0$ or $\mathbb{R}$. In this case we define $\bot \equiv 0$.

# 1 Introduction

This chapter will provide a brief introduction to the basics of how Network Service Providers (NSPs) operate their networks and the challenges they will face in the near future due to changing customer demands and advancing technological capabilities. It will highlight why these changes will result in an increase in complexity of several aspects of network design and operation, which motivates the need for a flexible new kind of efficient planning algorithms, such as the proposed genetic algorithm approach introduced in this monograph.

## 1.1 Network Service Providers and Service-Level Agreements

An NSP's business model depends on the revenue generated from sales of connection services and the cost incurred from creating, operating, and maintaining its network as well as contractual penalties for unfulfilled service obligations. Traditionally, the largest share of these services was based on an agreement regarding the data rate an NSP has to provide to the customer. Such contracts are typically long-term and inexpensive but contain almost no further guarantees on quality meaning that the service will be provided on a best-effort basis. On the other hand, a smaller number of customers, mostly from the business sector, held more differentiated contracts. These include additional Quality of Service (QoS) metrics as well as penalties for the case that a service should not be rendered to its full extent. The sum of such specifications is called Service-Level Agreement (SLA). To ensure that, even in the face of failing networking hardware, any such penalties could be avoided, NSPs employed more resources than strictly required for customers with more demanding SLAs, thus providing them with higher quality of service than was actually agreed upon. Since these customers were fewer in number, but generating more revenue per contract, this was a feasible model for NSPs.

Services with and without extended SLAs are commonly implemented on the same network infrastructure, which needs to provide sufficient resources to handle the expected traffic load generated by the customers. Initially creating such a network requires a large amount of capital expenditure (CAPEX) for public works, site and hardware procurement as well as installation and configuration; over the lifetime of the network there is also significant operating expenditure (OPEX) for network and service management, hardware and site maintenance, heating, ventilation and electricity. In order to reduce these costs, NSPs try to dimension their networks, i. e., determining the amount and location of network resources required, to be as efficient as possible.

Planning and provisioning an efficient network requires detailed information on the traffic flows the resources will have to transport. NSPs utilize present and historical data about where users connect, which SLAs they have and how much data they send and receive at which time to derive statistically expected peak data rates which are then used as the basis for dimensioning. In order to account for uncertainties of these statistical values and to provide a reserve for large-scale failures or short-term contract acquisitions, NSPs are known to design their networks with

capacities exceeding the expected maximum by a certain predetermined factor [22]. This practice is also known as overdimensioning. By prioritizing the few services for their SLA-customers over the many of best-effort-customers, NSPs were able to have a simple operation paradigm that maximized revenue.

## 1.2 Future Traffic and Service Requirements

Several technological advancements as well as novel applications are expected to have a significant impact on services and traffic characteristics in the near future. The current growth in Internet traffic of 24 % annually[1] [40], which is dominated by video traffic, will most likely continue, supported through more potent access technologies like Fiber-To-The-x (FTTx) and fifth generation cellular mobile communications (5G). Beyond the sheer volume, however, there will also be an increased focus on other aspects of service delivery. Novel technologies such as Virtual Reality (VR)- and Augmented Reality (AR)-devices give rise to the so-called *Tactile Internet* [249], which has applications in areas ranging from online gaming to industry 4.0 and telesurgery. In this context, "tactile" refers to a user experience of quasi-instantaneous feedback when interacting with a remote process through an Internet-connection. This requires End-To-End (E2E)-control loops, which are inherently sensitive to delay and delay variations. Especially in the context of non-failsafe processes such as surgery or remotely operated machines, availability of connections will be of utmost importance.

In addition to these factors, the traffic volatility, i. e., its dynamic fluctuations on short timescales, and its patterns of emergence might also be subject to change due to the Internet of Things (IoT). The IoT represents a paradigm wherein things from smart watches, health trackers and refrigerators to industrial systems and even cars are connected to the Internet and therefore drastically increase the number and location density of sources and destinations of data traffic. Coupled with the high capacity of 5G radios, this enables so-called massive machine-type communication scenarios in which vast numbers of independent devices exchange and react upon information of their own accord and far beyond the capabilities of systems with a human in the loop.

In a business context, such applications will lead to an increase in the creation, processing, and storage of large amounts of data. Many businesses have therefore outsourced operation and maintenance of computing hard- and software to the Data Centers (DCs) of Cloud Service Providers (CSPs) for reasons of scalability: Customers of CSPs can order and provision computing resources within minutes [155] and do not have to invest in and maintain a potentially vastly underutilized DC at their own expense. To enable business-critical tasks, CSPs offer their own SLAs with various performance guarantees. In order to make an E2E service between businesses and CSPs viable, the SLA of the connection between them has to match the SLA requirements of the cloud service. For NSPs, this necessitates a shift away from long-term and static services of few SLA-classes, which could be manually planned once and left in service; rather future businesses will need on-demand, flexible and QoS-diverse service compositions and therefore all operational and planning tasks of NSPs will need to be aware of the traffic flows' SLA requirements every step of the way. This not only significantly increases the complexity of planning tasks, but also requires a greater amount of flexibility in the networks operated by the NSPs.

[1] Compound annual growth rate 2016 – 2021 of worldwide IP-traffic

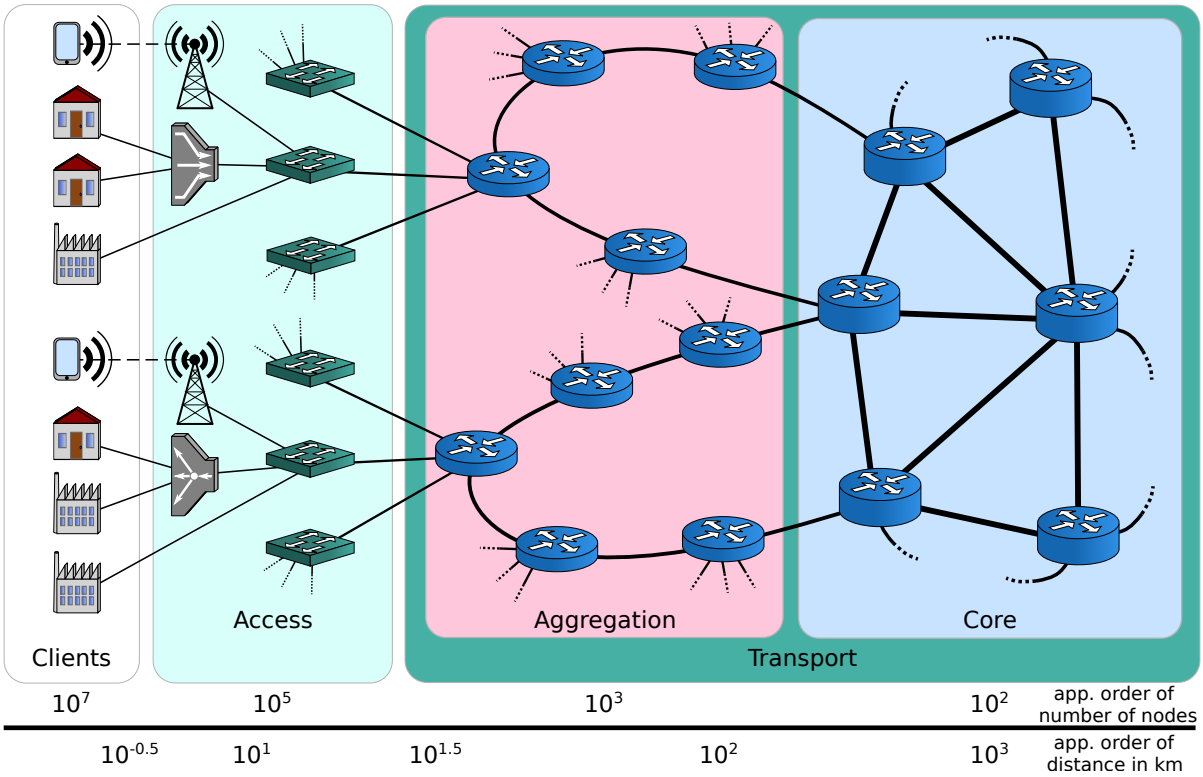



Figure 1.1: Common NSP/ISP network structure

## 1.3 NSP/ISP Network Structure

The networking infrastructure of larger NSPs and especially Internet Service Providers (ISPs) consists of many nodes, which represent an abstract view of physical sites with active networking hardware. These nodes differ in the functions they realize and are typically organized in three stages of networks. In the access stage, the users connect their user equipment such as home routers or cell phones to the access network. The networks in this stage often feature a variety of different technologies such as Digital Subscriber Line (DSL), Fiber-To-The-Home (FTTH) or Long-Term Evolution (LTE) cellular communication that may require very heterogeneous infrastructures on the side of the NSP. The access stage is therefore designed to handle technology-specific aspects and re-package the customers' traffic into a format that allows for a homogeneous transport in the next network stages which form the so-called transport network. The primary purpose of such networks is transmitting traffic from its source location to its destination location in the network [97]. For larger NSPs, the transport network is often further divided into an aggregation and at least one core stage. In the aggregation stage, which is often realized by redundant ring topologies, the traffic is collected from many diverse locations of the access stage and bundled on higher-rate interfaces toward the core for long-distance transport. The nodes of the core network commonly feature a much higher degree of interconnection than the previous stages forming a relatively dense mesh. The primary function in this stage is high-speed data transport over long distances between core nodes. On the egress side of the core the traffic is handed over, depending on where the destination is located, to another part of the aggregation network of the NSP itself, the transport network of a different NSP or the data center of a CSP. A schematic illustration of this structure along with rough estimates [239] regarding the order of the number of nodes and link distances is given in Figure 1.1.

Transport networks are designed as multi-layer networks, where the lowest layer is almost exclusively based on high-speed optical interconnection technologies such as Wavelength-Division Multiplexing (WDM), because they provide the highest data rates and reaches of commercially available interfaces. However, these technologies are based on circuit switching which impedes flexible traffic routing. To mitigate this, there is typically at least one layer on top which uses a packet-oriented transmission paradigm such as Multiprotocol Label Switching (MPLS) or Internet Protocol (IP). Many NSPs have several more layers, mostly due to hard-to-replace legacy systems like Frame Relay and Asynchronous Transfer Mode (ATM), but there is a tendency towards consolidating protocols towards a model of only two layers with a common control plane [116, 149]. This allows for a meaningful compromise between flexibility, performance, and management complexity.

## 1.4 Transport Network Design and Operation

In the past, transport networks could be designed and built layer after layer with the individual design teams including safety margins at layer interconnection points to avoid unforeseen shortfalls in this sequential process. After that, a network could remain largely static until the replacement by a next generation network. This had been a feasible mode of design and operation as any single trunk connection in the core could carry a very large number of client connections from different customers due to the large disparity in attainable data rates between the fiber-optic core and the copper-based access. A fact that also allowed transport networks to require only few core sites as they could readily collect the traffic of many aggregation sites.

But now, with the emergence of FTTx in the home segment and an increasing number of CSPs and other large businesses connecting directly to the aggregation stage with dedicated fibers, trunk connections can be saturated much more quickly by a smaller number of customers with varied needs in terms of bandwidth and low latency. Therefore, network design cannot rely on overprovisioning and ample safety margins for headroom anymore since trunk connections are costly. Further exacerbating the situation are new architecture concepts such as the 5G PPP's *network slicing* [85] and the Open Networking Foundation (ONF)'s *CORD*[1] [190] suggesting the integration of micro-data centers, edge cloud services and Network Function Virtualization (NFV). This leads to service creation points and data center functions moving from networks' cores closer to the access stage and becoming tightly integrated with network control functions. Consequently, nodes in the aggregation stage need to be enhanced in their functionality beyond simple traffic aggregation and become more similar to core nodes in connection density, function and technology [149, 239, 272]. Reacting to these changes, transport networks need to be planned with more sites, more functions and more hardware requiring precise dimensioning and safety margins. This induces additional complexity, not just to the network design and dimensioning problems, but also to the operation and service provisioning, especially when combined with diverse and short-term service requests.

To increase flexibility in network operation, NSPs have recently taken interest into adapting the Software-Defined Networking (SDN) paradigm for transport networks. SDN allows to monitor and configure network hardware by means of programmable interfaces, such that routing and connection properties can be dynamically adjusted to the present needs. While this has already been demonstrated to be viable in the packet-forwarding domain, optical technologies are also currently under development which allow a fine-granular control over various transmission and routing parameters.

[1] Central Office Re-architected as a Datacenter

A unified Transport-SDN (T-SDN) as envisioned by the ONF [275] would allow to jointly control devices on all layers and therefore enable a resource- and energy-optimized service-centric operation of the network where safety margins can be reduced and QoS flexibly provisioned as needed. The drawback, however, is the significantly increased complexity of finding an optimal network configuration on short timescales as more nodes, more technology-specific capabilities and limitations and more QoS parameters have to be taken into account simultaneously.

## 1.5 Problems and Algorithms for Future Transport Networks

NSPs need to design their network structure, dimension and configure their resources and provision connections according to their contracts and expected traffic. These tasks can be broken down into a small number of basic sub-problems such as versions of the topology design and routing problems. The most common sub-problems work on the substrate of graphs and can be formulated as combinatorial problems sharing certain constraints based on topological, technological, and traffic-related properties.

This allowed the development of many efficient solution methods, where the most common are either heuristics derived from graph-theoretical algorithms, or they are based on mathematical optimization. Heuristics are often readily extensible and scalable, but unable to find globally optimal solutions, whereas mathematical optimization allows to determine the global optimum, but is limited in scalability. As outlined in the introduction above, the mentioned problems will become significantly more difficult in the near future due to the following factors.

- Novel applications will require higher and more diverse QoS.
- Business cloud service adoption will emphasize SLA-awareness.
- Network density and device flexibility will increase complexity.
- Static overprovisioning will be replaced by dynamic allocation.

The increased complexity may well detract from the solution quality of common heuristics, but also render mathematical optimization infeasible. This motivates the need for novel algorithmic approaches which can solve these sub-problems by combining aspects from graph-theory and combinatorics while offering an improved scaling behavior through inherent parallelism and ease of modeling.

## 1.6 Contribution

This thesis provides two main contributions. First, it describes several different approaches to adapt a meta-heuristic optimization method based on genetic algorithms to solve the aforementioned problems in a scalable and readily extensible manner. Second, the performance of a select number of these algorithm variations is evaluated in comparison with another metaheuristic solution method, based on archetypal problem scenarios.

#### Genetic Algorithm Coding and Operator Design

Genetic Algorithms are a class of metaheuristics which can be used to efficiently solve highly complex, multimodal combinatorial problems. They express solutions as genetic codes and apply evolutionary operators to iteratively evolve better solutions. We present a number of approaches

combining various adaptions and acceleration techniques to create problem-specific genetic algorithm variants, which can be employed to solve complex network problems. We will explain each component and provide a theoretical analysis of their advantages and drawbacks.

**Evaluation in Multi-Layer Scenarios**

Several of the presented sub-problems will be combined in different ways to create typical problem combinations as they are commonly encountered in planning and operating multi-layer networks. We provide network and traffic examples to create scenarios in which we evaluate the most interesting combination of genetic algorithm approaches in terms of solution quality and solving time.

## 1.7 Outline

This thesis is structured as follows.

Chapter 2 gives a detailed description of the technological aspects of multi-layer networks. It explores the capabilities and limitations of the individual layers and their interplay, which define the central constraints to the problems under investigation. Based on these constraints, typical planning and routing problems are explained and categorized.

Chapter 3 provides background on different algorithms and methods which are commonly used in solving the problems from the previous chapter. It will show the basic working principles of common methods, both exact and heuristic in nature, and derive their applicability to different network optimization problems. Special attention will be given to the role of genetic algorithms. Finally, the chapter will summarize the current state of research on approaches to these problems.

Chapter 4 describes the design goals and approach for the genetic algorithms. It explains the network abstraction model utilized and how a network configuration can be represented as a chromosome by different genetic encodings and how corresponding genetic operators, both generic and problem-specific, interact with these encodings. These components are theoretically analyzed, both in isolation and as part of an integrated solution approach.

Chapter 5 specifies the scenarios used in the evaluation and explains the simulation environment and the solution methods used for comparison. The integrated solution approaches are evaluated in different scenarios with different sets of parameters and compared against the reference methods in order to demonstrate their efficacy in solving relevant multi-layer network problems.

Chapter 6 provides a summary of this monograph and ends on a conclusion on the algorithms and the results of the evaluation. Advantages and deficiencies are stated, and special emphasis is put on the identification of the remaining hurdles for application in a carrier-grade environment.

# 2 Planning and Routing for Multi-Layer Networks

This chapter is split into two sections. In the first section, a generalized description for multi-layer networks will be given, followed by the formulation of a dual-layer model architecture and its technical details, which reflect common properties of real-world multi-layer transport networks.

Building on these definitions, the second section presents typical tasks and problems, which arise when building or managing transport networks, such as traffic routing and network dimensioning. These problems are classified and discussed in the context of the dual-layer architecture. Finally, a selected set of possible goals for which operators typically design their networks will be given.

## 2.1 Multi-Layer Networks

A multi-layer network is a network of multiple layers, such that each layer may form its own, largely independent network, which is interconnected with the other layers at specific points. Typical transport networks and especially their cores are built as or have evolved into multi-layer networks for various reasons ranging from the need to maintain legacy services to combining advantages of different technologies. In this work, we will focus on a multi-layered, unified transport network structure following the same basic principles as suggested by the Telecommunication Standardization Sector of the ITU (ITU-T) in their recommendation G.800 [97]. This recommendation also contains a very comprehensive model for the proposed multi-layer network, capturing a plethora of different realizations, and dealing with transport and control aspects. However, this model is unnecessarily complex for the purpose of this monograph, such that we will consider a more generic and abstracted perspective, focusing on transport functions. This section will provide an overview on features, capabilities and limitations of such networks and commonly utilized hardware and provide the required details concerning the problems and algorithms in the following chapters.

### 2.1.1 Generic Structure

From an abstract perspective, a network consists of nodes and links, such that the links provide connectivity between nodes. A generic multi-layer network is an extension to this concept, such that it consists of different layers of individual networks. The layers are connected to each other by nodes in their respective networks that are associated to nodes in the other layers' networks. In communication networks, the layers follow a client–server principle such that lower layers provide connection services to higher layers. Higher layers then use these in order to realize their own links on which they provide connection services for their respective clients. A connection

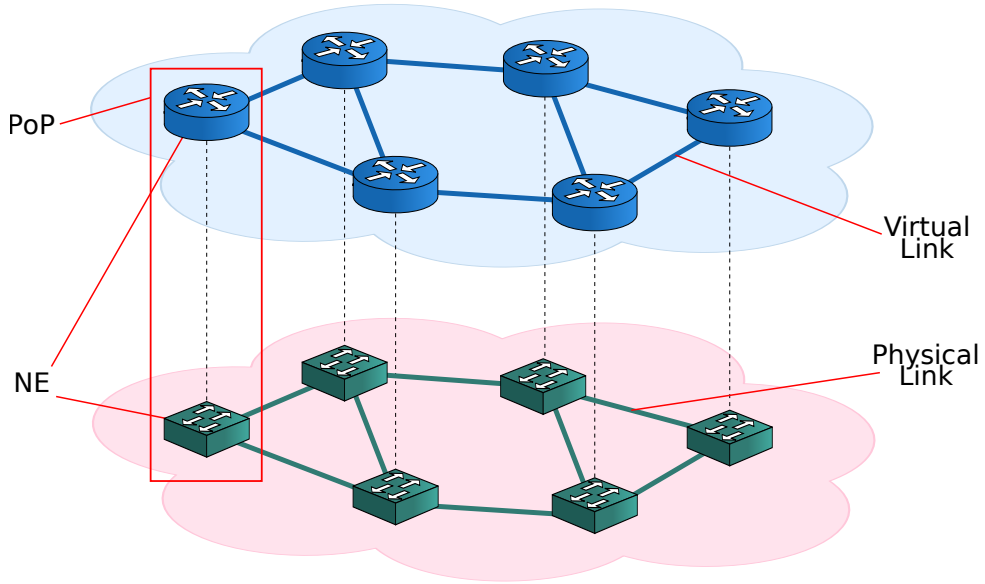



Figure 2.1: Multi-layer network with 6 PoPs where each virtual link has a physical link

service from a client perspective can therefore transparently be composed of several lower-layer connection services.

For the specific case of transport networks based on optical transmission equipment, only the lowest layer, formed by the optical fibers, physically interconnects nodes. All following layers recursively require connections of lower layers to establish their own links. Therefore, we will refer to the links of the fiber layer as *physical links* and to links on higher layers as *virtual links*. Since nodes with virtual links require lower-layer links to implement them, any node of a higher layer will consequently also have an associated node in the layers below. This monograph is focused on multi-layer networks, where the presence of a node in any layer means that there exist corresponding nodes in all other layers as well.

Considering the deployment of a network, this means that a node always corresponds to a site, where hardware resources of all layers are installed and connected. These physical hardware devices, called network elements (NEs) [153], are typically layer-specific and realize their layer's functionalities. Furthermore, we consider all of these sites to potentially feature connections to clients outside of the transport network. Such sites are also referred to as Point of Presence (PoP). Figure 2.1 shows an example with two layers and six PoPs, where each PoP consists of two NEs, a router on the blue upper layer and a switch on the red lower layer. The separation of layers is often motivated by a simplified management of heterogeneous systems, but there also other reasons, which are explored in Section A.1 of the appendix. Apart from the networking layers themselves, the customers' connection services between the nodes may also be considered as an individual layer above the actual transport network.

#### *2.1.1.1 Services and Demands*

From the perspective of a client, i. e., a service-requesting entity, outside of the network, a connection service in a transport network realizes a transparent, point-to-point connection between two PoPs. It allows the transmission of data from the source PoP on a given layer to another destination PoP located on the same layer. It has a required data rate and may or may not have associated QoS-constraints, e. g., regarding the maximum tolerable latency and minimum required

availability. This definition can apply to both, a client in terms of a request from another layer of the network itself, but also to the customer connecting to the transport network. In principle, NSPs can sell point-to-point connection services at every layer of their network, but the vast majority of the traffic is packetized traffic.

Some providers also offer special services such as routed or unrouted point-to-multipoint, encrypted or added value services such as Video on Demand (VoD) or converged telephony services. In this monograph, we will not focus on the peculiarities of such services in isolation, but rather treat them together with standard connection services. For the provider, these special services can be realized as a combination of individual point-to-point services and data center functions, and therefore, they can be modeled as connection services with specific constraints. Furthermore, we will consider so-called traffic demands as aggregates of connection services of identical source and destination nodes, as well as identical QoS-requirements summarized in QoS classes. Since all properties relevant to realize the individual services in the network are identical, their aggregate can be treated as an individual connection service.

### 2.1.2 Dual-Layer Architecture

Several multi-layer architectures for transport networks have been proposed and are in use with NSPs and ISPs. As stated in Section 2.1.1.1, most client traffic is packet-based, such that the top-most layer is implemented using a packet-based technology like IP or MPLS, while the point-to-point transport is handled by an optical layer providing high data rates and long-range connectivity, typically using Dense WDM (DWDM) [90]. More information on different multi-layer architectures and their protocols can be found in Section A.2 of the appendix. Abstracting from the different architecture variants, we will focus on a generic dual-layer architecture model, following the trend of network architecture consolidation [116, 149, 239, 243]. In the dual-layer architecture, every node consists of upper-layer and lower-layer NEs which are connected to each other. Since the principal interactions between these layers stay the same, no matter the exact technologies used, this generic dual-layer architecture is applicable to most common transport network architectures.

#### *2.1.2.1 Packet Layer*

All clients connect to the packet layer of this architecture through dedicated interfaces. Since the NSP knows these client interfaces and the associated connection services and SLAs, it is possible to assign arbitrary routes for every packet flow based on this information. The principle NE performing such actions on the packet layer is a router, which maintains both, client- and core-facing connections. Typical core routers (e. g., Cisco NCS 6000 [46], Nokia 7950 XRS [178], Juniper MX2000 [169], Huawei NE9000 [174]) follow a modular design that mainly consists of three different types of sub-components, such that each router has a number of slots for each type of module. Figure 2.2 shows an abstract model view of such a modular router.

Control plane functions and protocols are handled by the control modules. There are commonly two slots to allow for a primary and a backup module. The control modules can either use their own routing algorithms or receive explicit routes from external controllers from which they derive forwarding rules for the packet traffic on their own interfaces. The control modules compile these rules into forwarding tables and install them on the individual line cards, which are the second type of component.

Line cards house one or more transceiver ports through which packet traffic arrives and departs. Depending on the type, a transceiver may be a fixed part of the line card or come in the

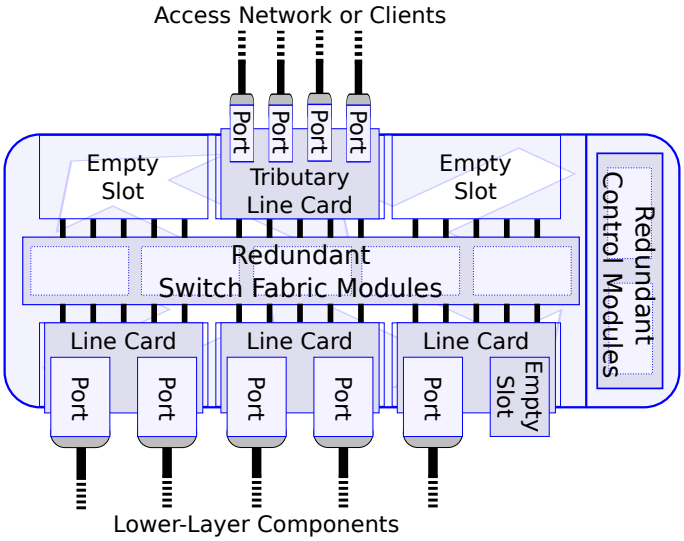



Figure 2.2: Upper-layer NE: A modular core router and its components

form of a pluggable module for slots on the line card, as indicated in the lower right of Figure 2.2. A single router chassis typically features between 2 and 20 line card slots. The total forwarding capacity of a router therefore depends on the number of installed line cards and ports.

They contain mostly buffer memory and complex Application-Specific Integrated Circuits (ASICs) optimized for high-speed packet classification and lookup in their respective forwarding tables. Depending on the rules, a packet is either forwarded to another port on the same line card or to a port on another line card. In the second case, the packets are relayed to the outgoing line card via an internal interconnection fabric which is formed by the switch fabric modules, the third sub-component. While the number of installed line cards may vary, the 4 to 8 slots for the switch fabrics modules are typically fully occupied for redundancy reasons. The client-facing line cards, which we will refer to as tributary line cards, exclusively feature ports connecting to client nodes directly or via the access network. The ports of core-facing line cards exclusively connect to other core nodes via the lower layer.

#### *2.1.2.2 Optical Layer*

The optical layer in this generic dual-layer architecture is a Wavelength Switched Optical Network (WSON) based on WDM. An optical connection in this architecture is realized by an optical circuit, also referred to as *lightpath*. We define an optical circuit to be a bi-directional point-to-point transmission between two NEs called Transponders (TXPs). Furthermore, each circuit has a defined *center wavelength* that occupies a specific part of the optical spectrum in a traversed optical fiber[1].

The physical length of fiber a circuit can traverse is limited due to different signal-degrading effects in the medium like attenuation and noise. While attenuation can be compensated by Optical Line Amplifiers (OLAs), which are installed at regular intervals of about 80 to 120 km [184, p. 106] along the fiber, the accumulation of noise through repeated amplification gradually reduces the signal quality with increasing distance [167, p. 79]. This requires circuits to be

[1] An optical fiber in this context physically consists of two individual strands of silica fiber to realize bidirectional transmission.

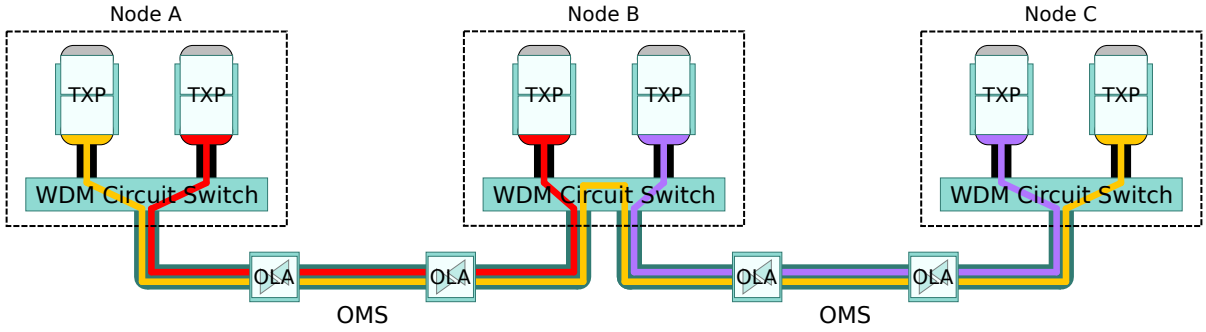



Figure 2.3: Three nodes connected by circuits of different wavelengths indicated by their color

terminated within their so-called *transparent reach*, which is the maximum fiber distance after which the data can still be recovered.

WDM transmission systems enable Optical Multiplex Sections (OMSes), where several circuits can be transmitted in parallel over the same fiber given that they use non-overlapping spectrum. To facilitate multiplexing, the available optical spectrum of such fibers is divided into a grid of a fixed number of standardized wavelength slots, e. g., as specified for DWDM in ITU-T recommendation G.694.1 [92]. A circuit may occupy one or more such slots, depending on the bandwidth of the exact signal. Two circuits in the same OMS cannot use the same wavelength slot. Figure 2.3 illustrates this by showing a topology with three circuits, each occupying a different part of the spectrum as indicated by their different colors. Note, that the circuit between A and B could also use the same wavelength slots as the circuit between B and C, since they never use the same OMS.

The defining property of a WSON is that switching can be performed on the level of individual circuits through special Wavelength-Selective Switches (WSSs). Since only WDM systems are considered within this monograph, we refer to these switches as WDM Circuit Switches (WCSs). In a WSON it is not necessary to terminate all circuits on the next node. Circuits with destinations other than the next node can be extracted from the OMS on the fiber and the WCS can directly switch them to be multiplexed into other fibers towards their destinations. This bypasses the upper-layer NEs and reduces the number of required TXPs. Such a bypass is illustrated in Figure 2.3, where the circuit between node A and node C passes through the switch at node B without being terminated at a TXP there.

The lower-layer NEs that require interaction with the control plane, i. e., mostly TXPs and WCS, are often integrated into larger chassis or shelves (e. g., Cisco NCS 2000 [44] or Adva FSP 3000 [88]) with common control modules as shown in Figure 2.4. Regarding the dual-layer architecture, the TXPs provide the link to the upper layer. A TXP can be considered to consist of two back-to-back transceivers linked by an electronic adaptation component. It performs Optical–Electrical–Optical (O-E-O)-conversion between a client-signal, which is typically a short-reach optical connection and a WDM circuit. The circuit is then switched through the WCS to an outgoing port, where it is multiplexed into the OMS and amplified for long-haul transport.

When it comes to optical switching NEs, there are various implementations combining different devices such as de- and multiplexers, splitters, couplers, fiber and WSSs based on Microelectromechanical System (MEMS) or Liquid Crystal on Silicon (LCoS) technology. In commercial systems many of these components are integrated in modules called Reconfigurable Optical Add-Drop Multiplexers (ROADMs), which can be combined to form larger, multi-

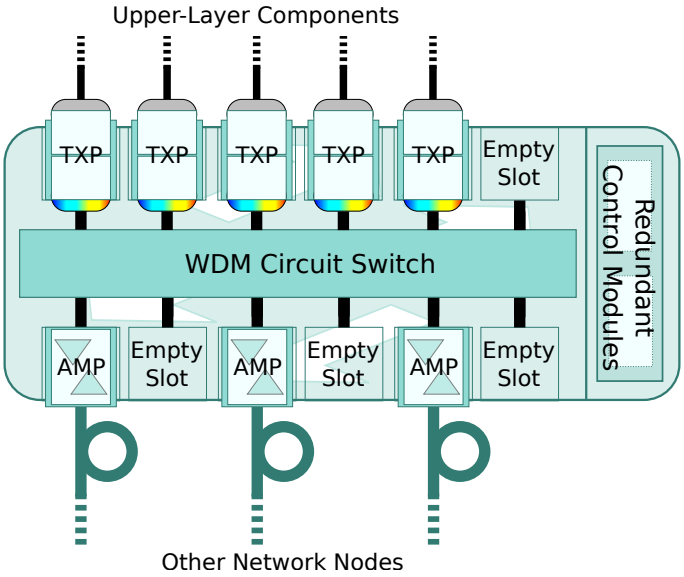



Figure 2.4: Lower-layer NEs and components: TXPs, WCS and optical amplifiers

degree ROADMs [50, 91, 195, 226]. In order to abstract from these implementation details and focus on the switching function, we define the WCS to be a generic, transparent and non-blocking optical switch for WDM-circuits with several fiber ports. It is able to switch any WDM-circuit from an OMS on any fiber port to an OMS on any other fiber port, thereby fulfilling the properties of being directionless, colorless and contentionless as defined in ITU-T recommendation G.672 [91]. All fibers leaving the lower-layer NEs, whether they connect to other nodes or to router ports via the TXPs, are attached to the WCS, such that only one is needed at each node. For more detailed information on the variations in composition and architectures of such switching devices, the interested reader is referred to the comprehensive work of Marom *et al.* [156].

#### *2.1.2.3 Layer Interactions*

Apart from the physical interaction between the layers, data rates, connectivity and failure mitigation have to be considered at their interface. The physical connection between upper layer and lower layer is typically formed by direct connections between router ports and TXPs or by having the TXPs integrated into the router port in case of an IP over DWDM (IPoDWDM) architecture[1]. For the proposed model, we consider this to be a non-reconfigurable 1:1 relationship, which is consistent with both variants. While special TXPs, called *muxponders* exist, which feature a larger number of ports to multiplex several low-rate router ports into few high-rate circuits, these make mostly sense when aggregating many client connections close to the customers. Since we do not consider the access stage of the network, the proposed model does not include muxponders. We further assume the line card ports and the TXP to always have matching data rates, such that the optical circuit can be used to the maximum of its capability without any adaption.

The circuits between two TXPs appear to the routers as direct connections and therefore represent a virtual link. If the fiber distance between the source and destination routers exceeds

[1]For more details on IPoDWDM see Section A.2 in the appendix.

the transparent reach of a circuit, then the circuit has to be terminated at an intermediate node and a second circuit needs to be established towards the destination.

This can be achieved in two ways. The first is via so-called 3R-regenerators (performing signal Reamplification, Reshaping and Retiming), which are essentially special TXPs with two back-to-back WDM-interfaces. These can be connected to the WCS and receive a degraded optical signal, recover the client data and create a new optical signal that is switched back onto the fiber towards the destination.

Alternatively, a circuit can be terminated at an intermediate node's router and the traffic can be switched onto a line card with an already existing circuit towards the destination. The advantage of the latter approach is that traffic of the original source node and traffic at the intermediate node, which is destined for the same target node, can be combined onto the same circuit. Additionally, traffic from the original source node to the intermediate node can also be combined onto one circuit. This prevents establishing additional circuits of potentially low utilization, thereby avoiding the undue use of associated hardware and spectral occupation. This harmonization of (data rate) demand to (data rate) capacity with the goal of improved efficiency is called *grooming* [296].

While grooming can help reduce the number of NEs required, it may come at a drawback regarding QoS figures. Packet processing in the router introduces additional latency. In the unlikely event that a packet interface becomes congested due to unmitigated traffic bursts, buffering may drastically increase this delay or even worse, lead to packet loss if the burst does not subside, before the buffer space is depleted. Furthermore, there is also a negative impact on availability since the traffic needs to traverse more NEs with several subcomponents which can all be subject to failure.

Dealing with failures is another important aspect of the coordination between upper and lower layer. The most likely failure scenario is a fiber cut. In this case, all of the circuits within an OMS on a cut fiber are interrupted and need to be rerouted in both layers and under the supervision of the control plane. Since the approaches under investigation in this monograph do not explicitly include failure occurrences and their recovery, we will omit specific failure recovery devices and procedures in the proposed architecture. We will include fiber cuts, as a static availability figure for each fiber, which we will consider for QoS-aware traffic.

### 2.1.3 Flexibility of Optic Transmission Systems

Traditionally, optical systems were mainly used to provide interconnection between neighboring nodes and therefore circuits were planned to be set up once and then simply remain active indefinitely to maintain connectivity [167, p. 29]. This enabled a reliable, but static network topology for the upper layers, which then provided the required flexibility in terms of routing. Nowadays, as packet router capacities approach and surpass the capacities of circuits and as circuit capacities approach the theoretical maximum data rate, flexibility is needed to manage resources efficiently. Legacy TXPs are hardwired to a single configuration supporting one predefined WDM slot and a fixed data rate given by a predefined modulation format and symbol rate. In contrast, present TXPs are not just *tunable*, meaning that they can be reconfigured in software to an arbitrary WDM slot, but also exhibit additional degrees of flexibility.

#### *2.1.3.1 Transmission Rates and Transparent Reach*

The new generation of *flexrate* TXPs, which are also referred to as Bandwidth-Variable Transponders (BVTs), can be reconfigured to adjust their modulation scheme [70, 280] and more advanced

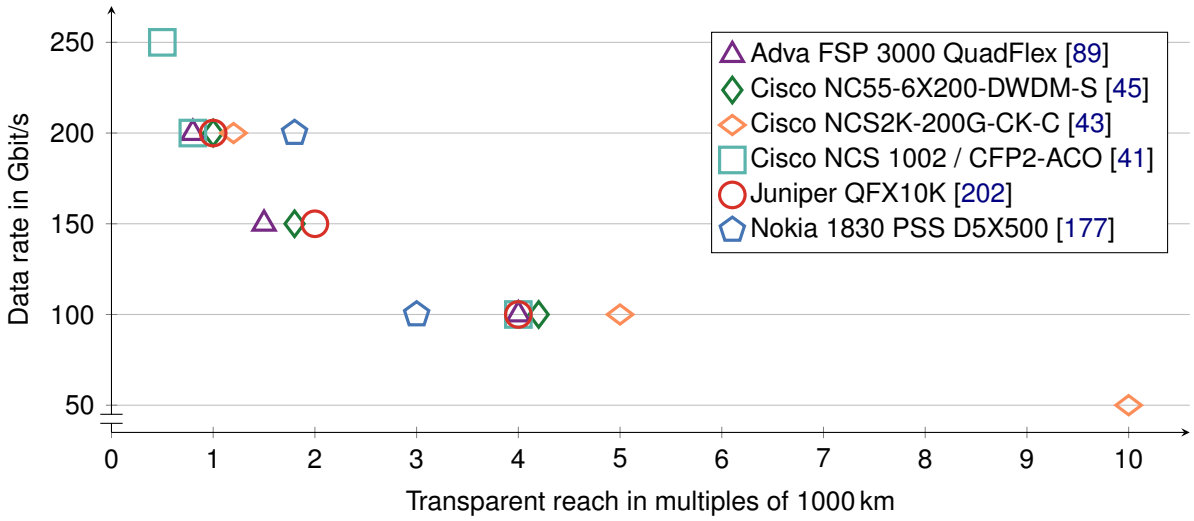



Figure 2.5: Illustration of different operation modes for commercial flexrate devices

models additionally their symbol rate [71, 177] in discrete steps. Since the net data rate is equal to the symbol rate times the number of bits per symbol as defined by the modulation, this can be used to realize circuits of different data rates. If lower data rates are sufficient, a more robust modulation format can be configured that exhibits a higher tolerance to degradation and therefore a much increased transparent reach. This allows bypassing more router nodes in the upper layer, potentially reducing latency and router hardware requirements. Figure 2.5 shows the reach values in different operation modes for a number of commercial flexrate devices. It should be noted that these values are given by the respective vendors and featured on promotional material. Therefore, they will, in all likelihood, only represent a rough, yet favorable estimate, motivated by business considerations.

### *2.1.3.2 Flexible Grid and Spectral Efficiency*

Increasing the symbol rate not just increases the data rate proportionally, but also the required optical spectrum. In legacy hardware, a circuit is limited by a fixed amount of bandwidth available in a WDM slot, which effectively creates an upper bound on the attainable symbol rate. E. g., a legacy DWDM slot can occupy at most 50 GHz of optical spectrum. Present-day optical NEs can allocate more than one slot to a circuit, but this capability introduces additional complexity. If slot size and bandwidth do not match well, this can lead to stranded spectrum which has a detrimental effect on the amount of traffic a fiber can transmit.

The *spectral efficiency* of a circuit, which is defined as the net data rate divided by the occupied spectrum, gives a measure of how well the spectrum is utilized. To avoid wasting optical spectrum by only partly using slots, the ITU-T specified a second, *flexible grid* [92], which reduces the DWDM slot size to multiples of 12.5 GHz. While this helps to increase the spectral efficiency for higher data rates, it conversely allows the same for lower data rates if a circuit does not require an entire 50 GHz slot.

## 2.2 Optimization in Planning, Operation and Routing

In the design and management of transport networks, NSPs most commonly have two overarching goals. One is to minimize costs, which is especially relevant during the design phase where hardware needs be procured. The second goal is to maximize utility for all resources that have already been installed. Many of these and related tasks can be formulated as optimization problems. In a broader sense, these are problems that aim to find a solution for which a predefined figure of merit reaches the best value possible. The function that determines this figure of merit for a solution, is called objective function and will be denoted $F$. Additionally, there are typically also a number of conditions or constraints that need to be fulfilled, which will be denoted as $\mathcal{C}$. This section will introduce common goals and conditions and present a classification of real-world problems in this simplified context. The following chapter will then give a more formal overview on the nature and solubility of optimization problems. For a brief summary of other networking optimization problems, see Section B.1 of the appendix.

### 2.2.1 Aspects of Planning and Operation

Optimization problems present themselves in many stages of a network's life cycle and accordingly, the objectives and conditions may differ at different points in time. The initial stages may offer greater freedom of choice, while later upgrades and extensions will face preconditions, e. g., because of the already installed network resources. While monetary aspects are always the primary driver, they may manifest themselves as different optimization goals. Especially during network operation, things like changing customer demands and unexpected resource reliability issues may shift the goal from hardware resources towards optimization of service provisioning.

#### *2.2.1.1 Time Frames and Resource Preconditions*

Network problems can mostly be divided into planning tasks and in-operation tasks. Network planning is traditionally defined by long-term tasks with a temporal horizon on the order of months up to years [194, p. 29]. They are solved offline, before a new network goes into operation (*green field design*) or undergoes large-scale redesign or expansion (*brown field design*). Typically, such network design problems consist of a given estimation of traffic demands along with some constraints on the resources with the objective to create an efficient network. In this context, the most efficient design typically needs the least amount of resources to handle the given traffic. When there is no initial constraint on the amount of resources, this is referred to as an *uncapacitated* problem [194, p. 106]. In colloquial terms, solving this type of problem means designing the network or part of it from scratch and determining where to install which combination of NEs.

Alternatively, the amount of available resources may be fixed from the beginning while the exact traffic demands are unknown. In this case, efficiency is often indicated by the largest amount of traffic a network can handle given these resources. This corresponds to a typical *capacitated* network design problem [194, p. 112], where network resources and a traffic generation procedure are given as inputs and the output defines the network. This procedure allows to generate scalable demands according to some model of expected traffic, such as those well-established in queueing theory. Less formally, solving this type of problems means finding a configuration for the installed or to-be installed NEs such that the resulting network provides the most capacity for future traffic.

Capacitated problems in green field or brown field design, which determine a configuration of the resources to be installed, are planning tasks, while dynamic networks, which can change

their configuration during operation, can re-interpret these problems to become operational tasks with traffic data based on actual measurements as input.

In network operation, the timescales are typically on the order of minutes to several days and the present network state with its configuration and traffic are reflected in the constraints. Such tasks can also be summarized as *network engineering* [194, p. 28]. The general problem can be formulated as follows. Given a set of measured or forecast traffic demands, determine the most efficient configuration of routes through a given base network topology constrained by the installed capacity and possibly required QoS. The definition of efficiency in the operational case traditionally translates to finding the shortest paths through the network, where a NSP can vary the meaning of "short" by assigning arbitrary link weights. With the advent of data plane programmability and optical flexibility, however, a myriad of other goals can be directly addressed. These range from providing lowest latency, highest availability to network-wide efficiency by adjusting the routing explicitly for each traffic flow.

These novel degrees of flexibility and the protocols and controllers of the SDN era, blur the lines between network planning and network engineering. Therefore, elements of traditional planning problems become relevant in operational tasks as well and may be subject to different optimization goals.

#### *2.2.1.2 Common Optimization Goals*

NSPs are typically companies that operate in a competitive market and therefore the primary goal of such enterprises is maximizing profit by increasing revenue and reducing cost. In terms of networking, this mostly translates to providing the largest number of connection services to paying customers with the least amount of expensive network resources while avoiding contractual penalties for violating SLAs. Although optimizing the composition and definition of connection service contracts is an interesting goal for an NSP in its own right, these aspects will not be considered here. Since the design of service contracts including pricing and penalties is subject to market effects and the NSP's specific customer structure and service portfolio, it is mostly driven by economic considerations outside the realm of networking and therefore such investigations are outside the scope of this monograph, which will focus on resources and SLA parameters.

##### *Resource Cost*

The cost of hardware resources is probably the most common subject of optimization, especially regarding CAPEX of fiber and NEs (e. g., [9, 14, 132, 273, 287]). A reduction in CAPEX can be achieved by optimizing the traffic demand routing, in order to groom traffic into fewer circuits and thereby help reduce the number of TXPs and line cards. Additionally, optimizing the virtual topology can enable bypassing upper-layer NEs, reducing their number, and may also be able to reduce the amount of optical fibers needed.

In the operational stage the network can be adapted for the traffic situation at hand. New connection services or extraordinary usage patterns may deviate from the originally planned capacity limits and cause local overloads or underutilization. Reconfiguring the network through adaption of virtual topology and routing can redistribute the load and return the network to a stable state if sufficient resources are available.

However, it is important to include QoS requirements as secondary objectives, as aggressive grooming may lead traffic onto longer paths increasing the delay and risk of being disrupted by a fiber cut.

***Availability***

The availability of a system is the probability that the system is functional at a given point in time. Since all NEs and fibers may be subject to degradation or failure, their availability naturally affects the availability of services. This has to be considered in both, planning and operation, especially when SLAs give explicit availability target figures.

A radical availability maximization will result in excessive amounts of hardware, such that an uncapacitated approach is not meaningful for this optimization goal, unless another factor indirectly limits the resources. This occurs naturally in brown field design, e. g., when the network is to be extended, where an optimization of availability can be constrained by a limited upgrade budget. For reconfigurable networks, this becomes a capacitated problem, where NSPs may optimize availability continuously, since critical traffic flows may vary over time. In both cases, traffic and circuit routing can be adjusted to move such traffic flows away from failure-prone hardware, while an optimized topology may offer more alternative routes for service restoration.

***Energy Efficiency***

Several studies have found that Information and Communication Technology (ICT) is a major consumer of electricity and accordingly contributes to greenhouse gas emissions [19, 21, 123, 128]. Therefore, more and more companies, including ISPs, CSPs, as well as equipment manufacturers and vendors, are striving for an eco-friendly profile due to market pressure and governmental regulations. Apart from external factors, energy consumption is not only part of OPEX, but also draws secondary costs in planning and operation with respect to power lines at PoPs, rack space, Uninterruptible Power Supply (UPS)-systems, power conversion losses, air conditioning and ventilation. Typical optimization goals in these works include maximizing the amount of traffic that can be offloaded to less energy-intensive systems, maximizing router bypasses or directly minimizing the overall power consumption of the NSP's network.

In the planning stage, a reduction in overprovisioning by consolidating resources and reducing margins can significantly improve energy efficiency. For the network operation stage, energy efficiency can be increased by switching off or reducing the capacity of devices through technologies well-established in consumer devices such as Dynamic Voltage and Frequency Scaling (DVFS) and extended sleep modes. The effectiveness of these approaches can be amplified by dynamically adapting routing and virtual topology to shift traffic away from lightly loaded devices, such that they may enter sleep or reduced capacity modes.

### 2.2.2 Topology Design Problems

The topology defines how the nodes of a network are connected to each other. For multi-layer networks, different topologies can exist at different layers. Common to all topology design problems is that they can be expressed as graph problems as follows. Given a set of nodes $V$ and a set of candidate edges $\langle s,t\rangle \in E_c$ with $s \neq t$ and $s,t \in V$ find a graph $g = \langle V,E\rangle$ with $E \subseteq E_c$ that fulfills a set of conditions $\mathcal{C}$ and is optimal regarding an objective function $F$.

Within $\mathcal{C}$ there are always two central components. There is a definition, how edges contribute capacity and cost to the network and there is a set of demands $D$ that specifies target capacities for pairs of nodes that have to be fulfilled. There exist capacitated and uncapacitated versions, where typically $F$ either minimizes cost under the constraint of providing a fixed target capacity sufficient for the demands or $F$ maximizes the capacity provided to the demands under the constraint of limited cost and/or resources.

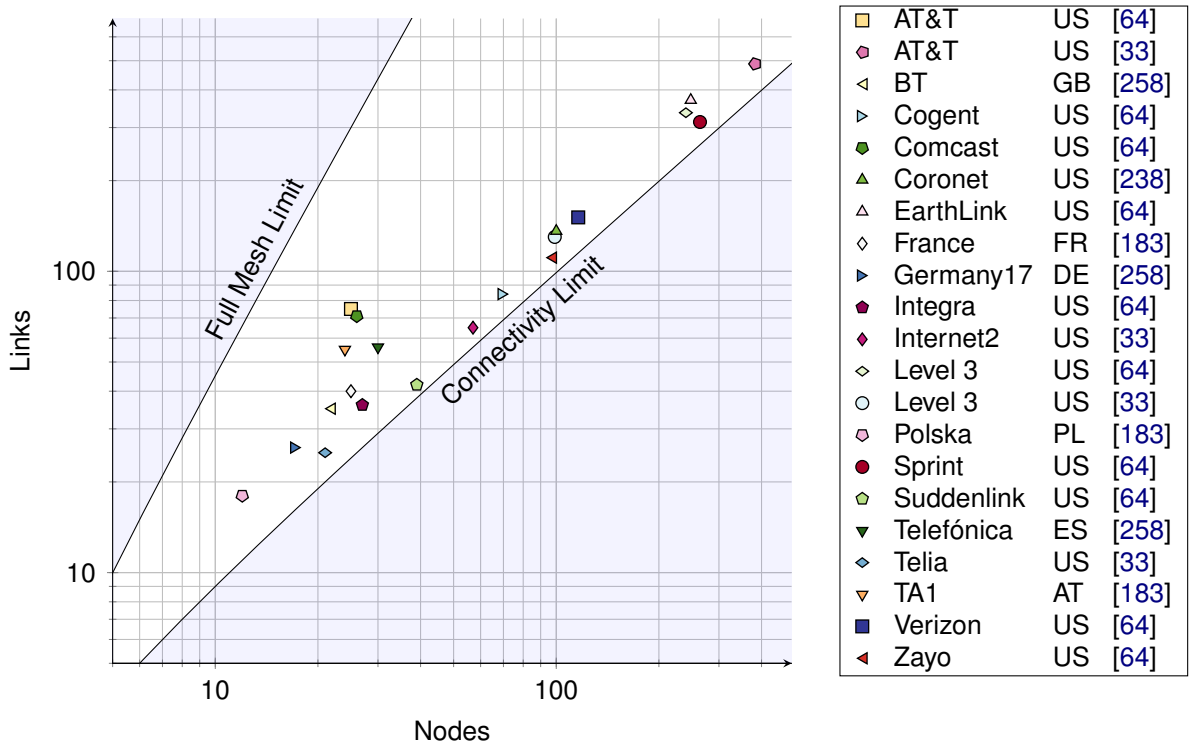



Figure 2.6: Layer-1 topologies of transport networks

#### *2.2.2.1 Physical Topology*

As explained in Section 2.1.1, the lowest layer is the physical layer, where interconnections are established by optical fibers. Therefore, the bidirectional candidate edges of the physical topology represent possible connections, where optical fibers can be deployed. The capacity fibers contribute to the topology graph can be defined by the number of fibers on a candidate edge or by the number of WDM slots, which can be carried by a set of parallel fibers. Physical Topology Design (PTD) problems therefore have a known set of node locations and a set of bidirectional candidate edges between them, and the goal is to find a topology such that $F$ is optimal and the resulting graph is connected. Commonly, there are also conditions to increase resiliency to outages, e. g., by requiring every node to be connected by at least two fibers on different edges. The cost of possible fibers between nodes may correspond to the price of deployment, purchase, or rent. Typical further edge parameters used in constraints may include expected availability, length, attenuation, or other fiber parameters.

The uncapacitated PTD is most commonly found in network planning tasks, where new fiber has to be added to the transport network in a cost-efficient way. Here, the possible interconnections may correspond to existing rights-of-way or similar easements, which legally restrict where new fiber can be deployed. The capacitated version commonly occurs in the planning tasks of NSPs who are not in possession of their own fiber infrastructure, but rather use a limited amount of leased dark fiber to create their networks, such that the possible interconnections are restricted by rental agreements with the fiber owners. For more details on planning and designing topologies with different motivations, constraints and goals, the interested reader is referred to Grover [115, ch. 9] or Pióro and Medhi [194, ch. 4 and 6].

The graphs of known physical topologies are often sparse or even planar graphs [27, 147, 183, 231, 258]. This fact can be illustrated by determining the averages of two standard measures

from graph theory for the fiber topology graphs of telecommunication networks: The node degree, i. e., the number of links connected to a node, and the normalized closeness centrality[1], which is a measure for node connectedness in graphs. For average closeness centrality, dense graphs with unit link weights result in values closer to 1, whereas sparse graphs result in values closer to 0. The number of links and nodes in a selected set of layer-1 topologies assembled in Çetinkaya *et al.* [33], Durairajan *et al.* [64], Rueda *et al.* [238], and others are shown in Figure 2.6. The trend in the ratio is much closer to the lower limit in graph density with average degrees between 2.3 and 2.72. Closeness centrality values range from 0.05 to 0.3. Many of these networks are structurally and sometimes even geographically very similar to road or railway networks [33, 64].

While it is argued that this sparseness is too common to be coincidental [27], it is unclear if it is an intrinsically intended result or a tangible result of cost-optimization or deployment preconditions. The physical topology of least fiber cost is by definition a minimum spanning tree, which is a property well-exploited in typical FTTx planning approaches [114]. Augmenting such minimum cost trees by resiliency constraints such as 2-connectedness yields graphs very similar to the presented networks. While the exact preconditions to fiber deployments are proprietary knowledge, NSPs are known to use preexisting fiber conduits for new deployments to reduce costs and some conduits are even shared between competing NSPs [64, 124]. In a similar fashion preexisting infrastructure and rights-of-way pertaining to operators of road, rail and gas pipeline networks are also used under contract by NSPs to deploy their networks [124], which inherently creates the structural analogy to these networks.

Due to such deployment constraints it is to be expected, that the cardinality of the candidate edge set $E_c$ is much smaller than that of a corresponding fully connected graph. Since the goal of topology design problems is choosing an optimal subset of $E_c$, the number of potential subsets is also smaller making the PTD problem comparatively more simple for larger problem instances. Furthermore, the graph using the entire candidate edge set $g' = \langle V, E_c \rangle$ may even be planar, especially when considering the capacitated problem versions. This property can contribute significantly to solving constrained topology design problems, as many efficient algorithms [154, 264] are known for planar networks.

#### *2.2.2.2 Virtual Topology*

The virtual topology is defined by the interconnections visible to the upper layer, which are in fact abstractions of the point-to-point capacity provided by the lower layer. Therefore, these bidirectional virtual links are the primary resource and the capacity they provide is the net data rate available to the upper-layer NEs. Virtual topology design problems, which we will denote as VTD, aim to find an optimal overlay topology based on a given physical topology and potential parameters restricting usage of lower-layer links. Since these links are realized by optical circuits for the given network architecture, this specific problem is also known as *lightpath topology design problem*.

The constraints on optical circuits typically contain an upper bound on the link length defined by the transparent reach, which directly relates to the capacity that can be provided as explained in Section 2.1.3.1. Therefore, a potential virtual link may in fact be infeasible if the shortest path in the physical topology exceeds the largest transparent reach available to a circuit. While this constraint is often addressed by a fixed mapping of lengths to data rates, there also exist more elaborate versions which incorporate the exact fiber parameters in determining the

[1] For further details on closeness centrality cf. section D.1 in the appendix.

transparent reach. Other relevant factors for virtual topology problems are latency and resiliency requirements of the traffic demands, which may also be expressed as constraints.

As mentioned before, a radical cost optimization without such constraints will result in a tree topology, which can in fact become a series of sequential interconnections where no alternative paths between two nodes exist. This situation is undesirable from a resiliency perspective since any failure of a node or link will immediately split the graph into unconnected islands. Such a topology is also unsuitable for low latency requirements since it creates unnecessarily long routes. E. g., traffic between leaf nodes will have to traverse all nodes and active links in the entire topology.

Similar to the PTD problem, there are capacitated and uncapacitated versions, but there are also common hybrid versions. For the uncapacitated variety, a virtual link may be realized by any number of circuits, which is a problem faced by NSPs when sites and fiber conduits are fixed, but there is no actual hardware installed or configured. In contrast to this, the networking hardware is already installed, but not yet configured in the capacitated case, such that circuits can only be created where sufficient hardware, e. g., in terms of TXPs and fibers, are available.

The hybrid versions are mostly motivated by the dependencies between NEs of different layers. E. g., the number of fibers in the physical topology may be fixed, but all other hardware still has to be installed. This can be considered as a specially constrained case of the uncapacitated version of the VTD problem. For more details on different aspects and subtypes of planning and design problems for virtual topologies, the interested reader is referred to Mukherjee [167, ch. 8] and again to Grover [115, ch. 9] as well as Pióro and Medhi [194, ch. 4 and 6] for more general considerations on topologies.

For the known layer-3 topologies given in the studies by Çetinkaya *et al.* [33] and Rueda *et al.* [238], we can see that the ratio of links to nodes can be significantly higher and more variable. Average nodal degrees range from 2.61 in the case of AT&T (with 2.55 for the corresponding layer-1 topology) up to 19.8 for Level 3 (compared to 2.63 for the underlying physical layer). Average closeness centrality varies between 0.3 for AT&T and 0.7 for Level 3. Furthermore, it can be observed that the number of nodes on the lower layers is always larger than the number of nodes on the higher layers as shown in Figure 2.7. This is caused by the fact that providing higher-layer services typically requires larger CAPEX investments since NEs at higher layers always require nodes on the layers below, whereas pure layer-1 services can be provided without hardware in additional layers. As outlined in Section 1.4, the difference between layer 1 and the layers above is likely to decrease in the near future as more and more layer-1 nodes will be extended towards higher layers to provide more and diverse service types.

While the above topologies are most likely based on a static optical layer, the increasing flexibility of optical devices as outlined in Section 2.1.3 has the potential to increase the density of the upper layer even further. Since transparent reaches exceeding 6000 km are feasible for state-of-the-art TXPs [26, 43], it is highly likely that any two points within the continental USA or Europe could theoretically be linked with a point-to-point connection. Moreover, port count, energy, and latency considerations provide a strong incentive to keep traffic in the lower layer and bypass any upper layer NEs when no grooming of meaningful magnitude can be performed.

Clearly, the VTD problem is much more complex than the physical topology design problem due to the potentially much larger size of the candidate edge set $E_c$. Coupled with the expected growth in the number of nodes and feasible links, this will severely impact the tractability of this problem in the near future. While the problem grows larger and more complex, there is also a growing interest in decreasing the solving times to enable more frequent network reconfigurations, further exacerbating the diminishing performance of established methods.

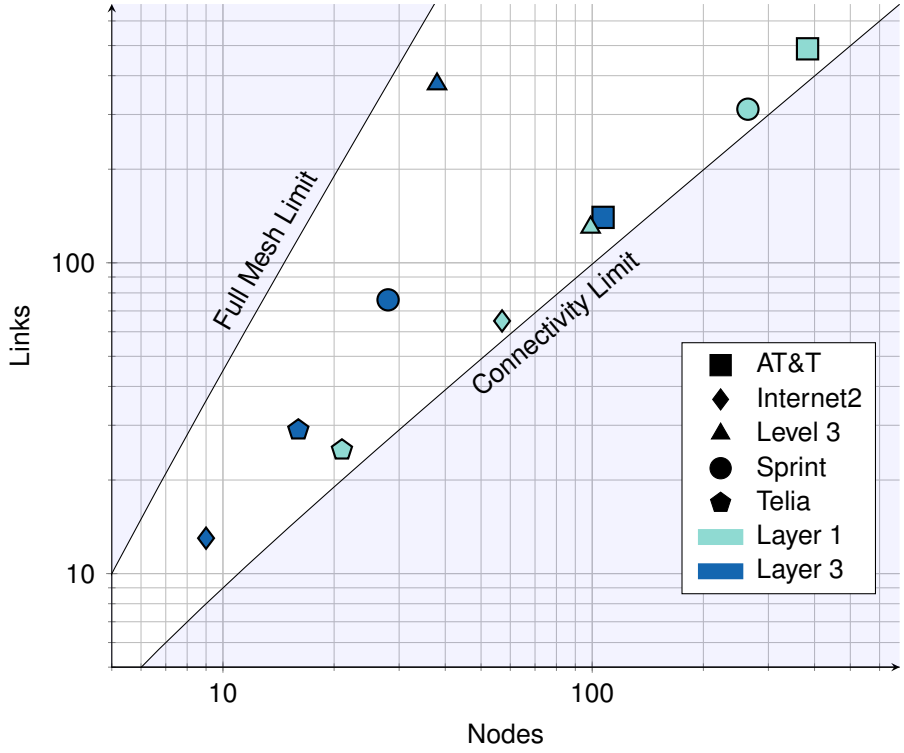



Figure 2.7: Layer-1 and layer-3 topologies of transport networks

### 2.2.3 Routing Problems

Routing can be defined as the act of determining a route or path through a network structure between a given source and destination. We will focus on single-source, single-destination routes since this is the most common type of routing problem. Real-world routing problems often depend on specific technology and service requirements, such that routes are often subject to further constraints.

Routing problems can inherently be expressed as graph problems as follows. Given a graph $g = \langle V, E \rangle$ with one or more pairs of a source node $s \in V$ and a destination node $t \in V$ with $s \neq t$, find a path $p = \langle e_i \rangle$ with $e_i \in E$ and $i \in \{1, \ldots, n\}$ for all node pairs required. For $p$ to be a path, it has to hold that $e_1 = \langle s, x \rangle$ and $e_n = \langle y, t \rangle$. Furthermore, when $n \geq 2$, it shall hold that for any $e_i = \langle a, b \rangle$ the subsequent element is $e_{i+1} = \langle b, c \rangle$. The most generic routing problem simply requires an arbitrary path for all possible pairs of nodes, but in addition to this definition[1] there may also exist a set of conditions $\mathcal{C}$, which have to be fulfilled. The most common is that there is a routing weight or routing cost function that defines how an edge contributes to the total cost of the path. When routing a single path for a single source-and-destination pair, the total path cost corresponding to the sum of the edge costs should be minimized, which corresponds to solving a shortest path problem. For the single source shortest path problem with non-negative edge cost, many efficient algorithms exist with the most well-known being Dijkstra's algorithm [59] (cf. Grover [115, pp. 189–192]).

While this generic routing problem requires routes between all node pairs without consideration of capacities, this may also be limited to a specific subset of node pairs, where demands are present. In this case, $\mathcal{C}$ contains a set of demands $D$, where a demand $d \in D$ is a tuple $d = \langle s, t, r, Q_\mathrm{d} \rangle$ of its source node $s \in V$, destination node $t \in V$, required capacity $r \in \mathbb{R}_+$ and a

[1] Note, that this includes the single-hop case with $s = y$ and $d = x$, such that $e_1 = e_n = \langle y, x \rangle = \langle s, d \rangle$

set of additional constraints $Q_d$, which can be empty. For routing problems in communication networks, the additional constraints may include limitations regarding length and resiliency.

For an entire network, routes for many or even all source and destination pairs have to be determined, with the goal of optimizing a global target function $F$ which depends on all routes. What makes optimizing $F$ much harder than solving the shortest path problem is that the constraints of any determined route may influence constraints for all other routes. E. g., when considering capacity constraints, deploying a route on a link may deplete the link capacity and thereby exclude it from use in subsequent routes. Therefore, optimizing all individual path costs separately, may not lead to an optimal value of $F$. Routing problems exist in capacitated and uncapacitated form, such that typically, $F$ either minimizes cost under the constraint of providing a target capacity sufficient for a fixed demand set or $F$ maximizes the number of serviceable demands by providing capacity under the constraint of limited cost.

#### *2.2.3.1 Traffic Demand Routing*

In the dual-layer architecture, packet routing is done on the upper layer, such that the edges available for a path are the links of the virtual topology and traffic demands can be routed along these paths. These virtual links provide a capacity in transmissible data rate which is used by traffic demands. The traffic demands in transport networks are typically aggregated packet flows, such that all properties influencing path selection are identical within a single traffic demand. Routing has to be performed for each demand, rather than each packet flow and therefore the goal of traffic routing is to find a feasible path for each demand. The feasibility of paths may be restricted not only by capacity constraints, but also by constraints derived from the QoS parameters of the underlying SLA. The most common constraints are based on loss, latency, and availability. We will denote these problems as traffic demand routing (TDR) problems.

Packet loss is mostly caused by nodes discarding packets due to overload or by transmission errors on links. However, in transport networks loss is a comparatively rare event due to careful network design. NSPs employ suitable Forward Error Correction (FEC) to compensate for link errors and apply rate shaping to client traffic early at the hand-over points, such that excessive traffic spikes are blocked in the access stages before they can cause congestion in the core. Since this leads to predictable traffic demand volumes, the problem of avoiding loss through congestion at nodes can be met by careful capacity planning.

Availability can be addressed by estimating the failure probabilities of NEs and fibers along the candidate path and choosing among those with sufficient availability or by including backup and restoration mechanisms.

Latency is the result of several delays occurring on both, nodes and links. On virtual links there is predominantly propagation delay which is determined by the length of fiber the circuit traverses. For nodes there are delays for processing, queueing, and transmission. Transmission delay is essentially the time the interface needs to put the data into the fiber. Therefore, this delay depends on the transmission rate and the packet size and is typically about 1 µs or less[1]. Processing delay is the time required to analyze and classify the packet header, look up the correct output interface and forward the packet to it, which typically remains well below 1 ms [47, 133, 200].

Queueing delays occur whenever the input rate to a node is higher than the output rate. This can be caused by traffic spikes and may amount to several 100 ms, but such cases should be prevented by prior rate shaping. The more common case is queueing delay due to output

[1] A typical packet of 1500 Bytes incurs 1.2 µs of transmission delay on a comparatively slow 10 Gbit/s interface.

contention, which arises when several input interfaces simultaneously receive traffic destined for the same output interface. If the sum of the input data rates remains below or equal to the output data rate, then a packet is queued for at most the number of contending input interfaces times the transmission delay on the congested output interface which is typically on the order of a few microseconds[1].

Another factor, which is sometimes encountered, is the requirement of having all traffic of a demand to be routed on a single path, which is also called *non-bifurcation constraint*. Being able to arbitrarily split an aggregated demand can help improve the utilization of network resources but may impact client traffic negatively. Splitting the traffic may lead to a re-ordering of packets due to different lengths of different paths which is not tolerable if the client relies on a predictable low-jitter environment, e. g., as is required for a Time-Division Multiplexing (TDM) emulation service.

Packet routing can be performed in two basic ways, a more traditional distributed and a fully centralized approach. For the first, a network management system or controller can determine a system of link costs, also known as link weights, and push these values to the NEs, such that distributed routing modules on each NE only need to compute shortest paths based on these costs. The alternative is to have a path computation engine determine explicit paths through the network and signal them directly to the forwarding engines of NEs. This allows arbitrary routing for all traffic and offers the greatest flexibility for traffic engineering, but it requires a tight control of NEs and the capability to quickly solve the global routing problem.

For the uncapacitated problems, arbitrary traffic volumes can be assigned to a path, which can be used to determine how much hardware, e. g., in terms of ports and line cards, need to be installed at the nodes. This is where many hybrid versions of this problem come into play since many combinations of pre-existing hardware and new hardware can be used. The number of TXPs may be fixed or the number of ports on the optical switch may be limited or there are upper limits on the number of line cards installable in NEs etc.

The capacitated problems occur often in network operation. When client traffic sufficiently deviates from its forecast, the routing may have to be adjusted to prevent resource bottlenecks at unplanned points in the network. While this is typically a reaction to long-term traffic changes, it may also be used for specific short-term goals such as restoring network operation after unplanned failures or moving traffic away from certain links or nodes when maintenance is required. More routing problems and versions thereof are given in Vasseur *et al.* [271, ch. 4], as well as Pióro and Medhi [194, ch. 7].

Traffic routing problems are among the most complex problems since they need to be solved on the dense topologies of the upper layer which leads to a very large space of possible paths. This is further exacerbated by the increasing focus on QoS, since demands of differing SLAs may require specific routes. In this case, the number of paths does not only depend on the number of source-destination pairs, but also on the number of SLA-classes. At the same time, however, NSPs like AT&T [266] and Verizon [20] show increasing interest in the optimization of routing in their SDN-enabled networks to support a flexible (re)configuration for NFV-based services.

[1] Considering an extreme case where a large router simultaneously receives 1500 Byte packets on 1000 different interfaces of a transmission rate of 1 Gbit/s which are all forwarded to a single 1 Tbit/s output interface, then the queueing delay amounts to about 12 µs.

#### *2.2.3.2 Circuit Routing*

Circuit routing is performed on the lower layer of the presented dual-layer architecture. The paths for the circuits consist of edges of the physical topology of the network, i. e., the available fiber links. The fiber links provide optical spectrum which is occupied by active circuits. As outlined in Section 2.1.2.2 and Section 2.1.3 there are many factors determining, whether an optical circuit is feasible. Two factors which always need to be addressed are the maximum transparent reach and which wavelength slots will be occupied. We will denote the circuit routing problem without integrated wavelength slot assignment as Independent Circuit Routing (ICR) problem.

Transparent reach is the most important constraint in determining path feasibility as explained in Section 2.1.2.3. It presents itself as an upper bound on the cumulative length of edges in a path. If a connection service exceeding this length is required, then there are two general and one hardware-specific approach. The routing of the lower layer may reject the request outright and delegate the problem to be solved in the routing of the upper layer. The second approach involves using sufficient 3R regenerators such that the resulting linear sequence of circuits can be treated as a single virtual link. Finally, if flexrate devices are considered, there is also the option of reducing the data rate to increase the transparent reach as explained in Section 2.1.3.1 and realize the service by bundling several circuits.

Apart from feasibility concerns related to the path length, a circuit also requires an appropriate amount of spectrum along the path as explained in Section 2.1.2.3. Considering the so-called *wavelength continuity constraint* [115, sect. 10.4.1], i. e., the constraint that a circuit traversing a sequence of fibers always has to occupy the same slots on all fibers, as part of the routing optimization gives rise to a number of complex subproblems. For fixed-grid systems this is known as the wavelength assignment problem, for flex-grid systems it is also referred to as spectrum assignment problem. These problems can either be solved independently or together with circuit routing resulting in the Routing and Wavelength Assignment (RWA) or Routing and Spectrum Assignment (RSA) problem, respectively. However, it is a common simplification to disregard this constraint [167, p. 442] [77, p. 18] and assume full wavelength conversion capabilities in every optical node, which could be realized using tunable 3R-regenerators. More details on RWA and related problems incorporating additional physical layer factors can be found in Mukherjee [167, ch. 7], Pióro and Medhi [194, pp. 446–451], Pachnike [184, ch. 5.3], or Farrel and Bryskin [76, pp. 248–253].

When considering QoS-based requirements, latency bounds may also lead to a length limitation of possible paths. Delays are incurred on fiber links as well as at nodes. Within nodes, TXPs add delay through internal processing (5 µs to 11 µs reported in Vjaceslavs *et al.* [276]) and especially through FEC calculation (about 4 to 50 µs [41, 42]). Amplifiers, switches and other purely optical components add further propagation delay due to internal fiber spools and external cabling. However, these fiber lengths are small compared to the fiber lengths between nodes. Light travels 1 km in typical optical fiber in about 4.9 µs, such that given the geographical layout of typical transport networks, the propagation delay on links becomes the dominating factor [284] [167, p. 442] which has to be considered when latency bounds are specified.

Availability and protection mechanisms are often included in SLAs. While active NEs are always a potential source of failures, their projected availability as given by the Mean Time Between Failures (MTBF) values stated by vendors, are typically much higher than that of the fibers. While underground conduits are regularly damaged during construction works, the situation is much worse for aerial fiber deployments which are often disrupted by traffic accidents, weather incidents or even vandalism according to a study [52] performed by the Federal

Communications Commission (FCC). Especially fiber topologies which feature both, aerial and underground deployments, should explicitly consider the likelihood of failures.

The capacitated version of the circuit routing problem has already deployed fibers, TXPs and WCSs. It is encountered during network operation and may be motivated by effects at different layers. E. g., SLA-based services are added, changed, or removed at the service layer, traffic behavior on the packet layer may differ too much from predictions or the establishment of new circuits on the optical layer is hindered by excessive spectral fragmentation. The goal is typically to either realize the requested circuits using the least amount of spectral resources, or to minimize blocking, i. e., the rejection of requests due to resource depletion. The uncapacitated version and its hybrids are used to determine the optical equipment required. This may mean that the packet routing NEs at nodes are already installed and the number of fibers to be deployed in conduits is subject to minimization, or, the more common case, fiber is already deployed, limiting the spectral resources and the goal is to minimize the number of TXPs or the ports on the switching systems to be installed at nodes.

Just like in packet routing, paths can be computed by routing modules locally at each node, which maintain and distribute adjacency and link cost information, or the paths can be computed by a centralized path computation engine and signaled explicitly to NEs. As routing occurs on the substrate of the fiber topology, which commonly is rather sparse as explained in Section 2.2.2.1, and the transparent reach restricts the path length, the number of possible paths is typically significantly reduced compared to the routing of demands on the upper layer. However, when wavelength continuity is considered, the complexity increases significantly when solving it as an integral part of the RWA/RSA problem. When separating WDM-slot allocation from routing, then referred to as Wavelength Assignment (WA), either a subgraph of available slots can be determined before routing, or a slot may be selected after routing and checked for feasibility. The latter option is the more common case with allocation strategies often relying on simple heuristics such as “first fit” or “least used”. With the increasing flexibility of the optical layer and the development of T-SDN there is an increasing interest in the optimization of routing on short timescales [175].

### 2.2.4 Dimensioning and Reconfiguration

*Network dimensioning* refers to the problem of designing a network such that it features the least amount of resources, while providing a given target capacity under given constraints. This target capacity is typically based on forecasts regarding future traffic demands and often includes ample overdimensioning to ensure that the network has sufficient capacity until the next upgrade or replacement cycle. Network dimensioning is the central task of network planning, which needs to solve a combined version of some or all of the previously mentioned sub-problems. Accordingly, the complexity of network dimensioning emerges from combining the complexities of its constituent sub-problems, such that dimensioning is very difficult to solve for large networks. While dimensioning is typically an uncapacitated problem, there is also a similar capacitated problem, which we will refer to as *dynamic resource reconfiguration* or reconfiguration for short.

Resource reconfiguration is a task closer to network operation, where the network is to be configured such that its resources fulfill the required traffic demands subject to a specific goal. Such a goal may be to find the minimum amount of active hardware for a given set of traffic demands with the intention of reducing OPEX or maximizing throughput given changing traffic patterns. In contrast to the large time frames common in network planning, a dynamic resource

reconfiguration may be required repeatedly to adjust to dynamically changing demands and therefore may have stricter timing constraints.

While it is also possible to maximize the routable traffic in a dimensioning problem given a fixed number of fibers, this would generally result in a maximization of NEs and by extension CAPEX, which is not typically desirable in the network planning phase where limiting cost is the main driver of optimization.

Inputs to the uncapacitated problems are the required point-to-point traffic demands and the basic network infrastructure of node locations and potential physical links between them. For the capacitated problem, there is typically a resource limit that cannot be exceeded when minimizing the number of active NEs. If the goal of the capacitated problem is to maximize the traffic throughput, then there is an additional input regarding the expected traffic growth, such that a maximum can be determined, e. g., in the form of a scaling factor.

#### *2.2.4.1 Subproblem Dependencies*

Figure 2.8 shows a multi-layer network based on the dual-layer architecture and illustrates, how the previously mentioned routing and topology design problems depend on each other and the input values. The physical topology marks a subset of the link candidates of the infrastructure and optical circuits can only follow the physical links. The virtual links correspond to the optical circuits, such that they have the same source and destination nodes. Since circuits may use more than one physical link, the virtual links can form a different topology than the physical links, which is illustrated between the nodes A, B and D.

Furthermore, the virtual topology is agnostic regarding the exact circuit routing. Therefore, a link in the virtual layer may actually not be routed on its corresponding physical link. While D and F have a physical link candidate, it is not used in the physical topology and the virtual link between them is realized by a circuit which connects D to F via E. This also allows for a bypass circuit to add a virtual link between two nodes that do not have a link in the physical topology such as between A and D.

The traffic routing is performed on the virtual links of the upper layer. The demands, which are presented on the very top of the figure are just point-to-point capacity requests, such that they are agnostic to both, the upper[1] and lower layers. In order to utilize resources efficiently, several traffic flows can be groomed into the same circuit when sufficient capacity is available. However, this may lead to excessively long routes, which can be observed for the demand between A and C. If this demand were to require a strict QoS limit on latency, then it would probably have to be served on its own circuit.

Therefore, most approaches for network dimensioning and resource reconfiguration focus on the traffic demands to mark the target quantity and use a top-down approach. The two solution methods presented in the following chapters also fall in this category. One is based on an uncapacitated upper-layer topology, the other uses an uncapacitated upper-layer routing as starting points to determine where and how many circuits are needed. This information is subsequently used to determine the amount and location of all NEs needed. The number of fibers may be a fixed input-parameter or, in the case of an uncapacitated physical layer, may also be determined as part of the network dimensioning.

[1]The exception being IPoDWDM without client traffic encapsulation, where intermediate IP nodes may be visible to the client. For more information on IPoDWDM see Section A.2 in the appendix.

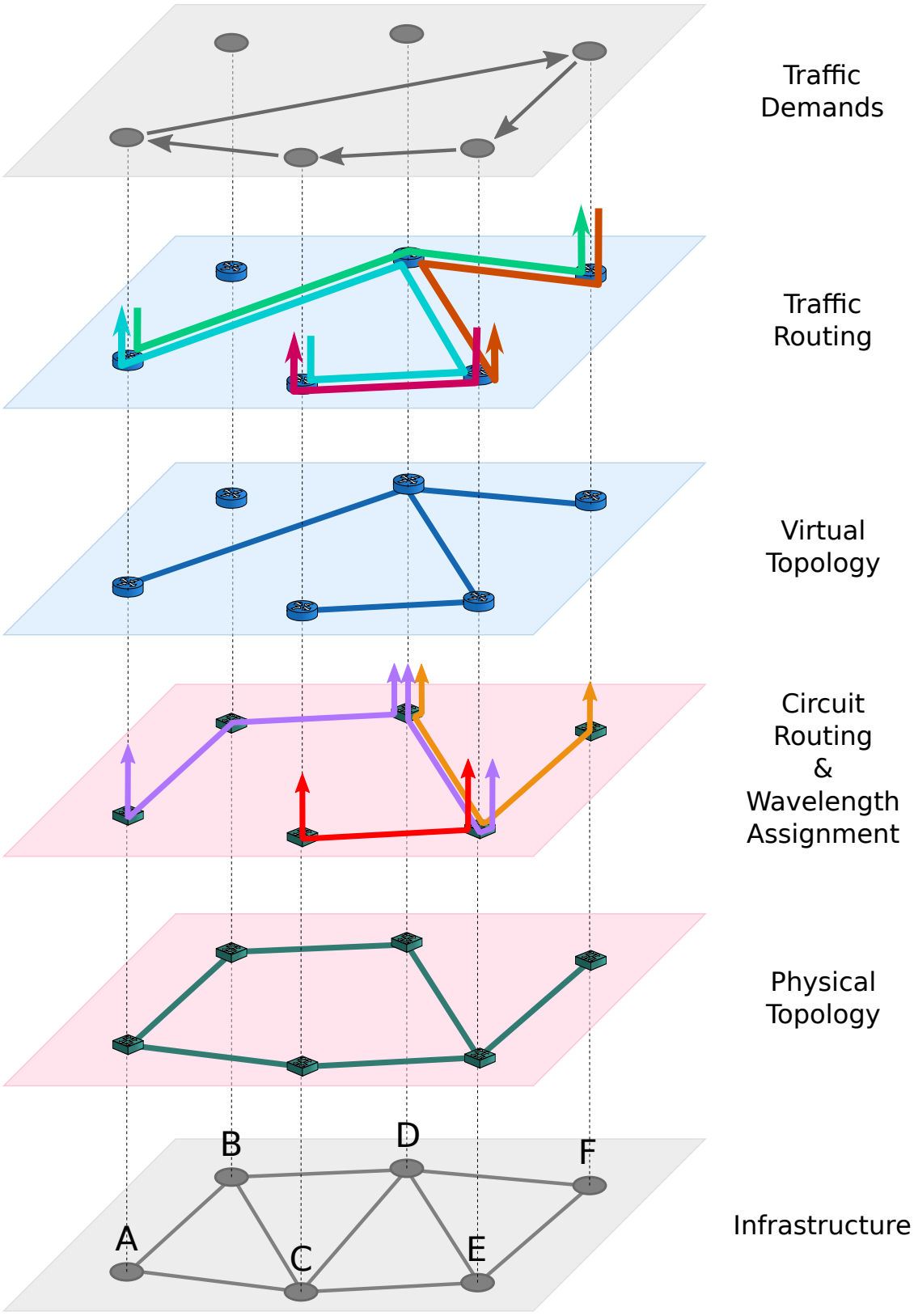



Figure 2.8: Aspects of multi-layer network dimensioning and reconfiguration

#### *2.2.4.2 Upper-Layer Resources*

As explained in Section 2.1.2.1, client traffic arrives at tributary line cards. Their number only depends on static information from traffic demands, since the number of connecting clients and the sum of incoming and outgoing traffic are known a priori. Consequently, there is no potential for alternative configurations exempting these line cards from optimization. The traffic demand information also puts a lower bound on the number of required core-facing line cards and their ports. Depending on traffic routing, there may be also be *transit traffic*, i. e., traffic which neither originates nor departs at that node, but is simply switched from one core-facing port to another. Since each line card may have several ports and traffic demands often require either more or less capacity than is provided by one port, this opens potential for traffic grooming.

Furthermore, in case of flexrate-capable ports, where the line rate and reach on the lower layer can be adjusted, there is potential for additional savings whenever smaller, but far-reaching traffic demands cannot be groomed into existing connections. The number of line cards and traffic flows could then be used to determine the number of required switching modules. Each switching module and each line card require a slot in the modular chassis of the router. Therefore, when TDR is solved, the information on the traffic volumes and their paths can be used to dimension the entire upper layer resources.

#### *2.2.4.3 Lower-Layer Resources*

The lower-layer NEs and by extension the number of their shelves, either depends on the fiber degree in the physical topology or directly relates to the upper-layer resources as established in Section 2.1.2.3. For every port in a line card, a corresponding TXP is required and in turn every TXP requires a client port in the WCS. Beyond that, the WCS needs a line port for each fiber of the physical topology, which can be assumed to be at least one for each physical link. The internal design of the WCS itself, while out of scope for this monograph, can also be derived from the number of client and line ports required. The last major component of the lower layer, the optical amplifiers, are installed on each line fiber to boost outgoing and incoming signals, while the number of OLAs depends on the physical length of the fiber.

Each TXP can establish a circuit to another TXP which requires spectral resources on a fiber. Having solved VTD and TDR provides information regarding which capacity is needed between which locations. This data can then be used as input to RWA/RSA or ICR, which determine the routes and spectral resources required by the circuits to deliver the requested capacity. The circuit routing may either be constrained by a capacitated physical topology or the number of required fibers and conduits can be determined as well.

## Chapter Summary

Within this chapter, a definition for multi-layer networks and their components has been given. Based on this, a generic dual-layer architecture utilizing an abstract packet routing layer and a circuit-switched lower WDM layer has been defined along with their respective Network Elements (NEs) such as modular routers and transponders (TXPs). Furthermore, their respective capabilities and limitations have been derived from commercial systems.

Several concepts relevant to network planning and dimensioning like bypass circuits, flexible optics and grooming have been introduced and their potential merit regarding different optimization goals has been motivated. In the second part, several problems arising in planning and operation of multi-layer networks have been explained. Especially topology design, both virtual (VTD) and physical (PTD), and routing for both, traffic demands (TDR) and circuits (ICR and RWA), have been introduced and discussed based on the structure of known transport network topologies and the properties of typically used NEs.

Finally, a number of common optimization goals pursued by NSPs and ISPs, such as resource utilization and energy efficiency, are presented and their implications regarding requirements and capabilities of NEs explored in the context of different optimization time frames. It has been derived, that a SDN-controlled network can be dynamically reconfigured in order to create opportunities that facilitate grooming and resource sharing. Thereby it is possible to increase the utilization of NEs to improve throughput or reduce cost. However, multi-layer awareness is key to maintain QoS guarantees, since excessive grooming can lead to larger delays.

Optimizing a network dimensioning or dynamic resource reallocation thus requires high-performance algorithms which readily scale to support the increased complexity of future transport networks.

# 3 Network Optimization

The previous chapter has shown that a large number of complex optimization problems exist in the context of transport networks. This chapter will provide a concise introduction to the formal aspects and the methods commonly used to find optimal solutions for them. It will outline the advantages and drawbacks of these methods with respect to common problem versions. Special emphasis will be placed on Genetic Algorithms as they represent the primary focus of this monograph, while also giving details on Simulated Annealing, which will later serve as a reference method. The last part of this chapter will provide an overview on current research, exploring where and how these methods are often integrated into larger solution approaches.

## 3.1 Introduction to Optimization

While the previous chapter has already introduced a more colloquial definition of optimization, this section will provide a more formal approach.

### 3.1.1 Basic Definitions

In mathematics there exist several problem categories which can be related to the nature of their solution. We will focus on decision, search and optimization problems. For a non-empty set of all possible inputs $S_{In}$, there is a relation $R(i,o)$ which associates an input $i \in S_{In}$ to its possible solution state $o \in S_{Out}(i)$, which may be empty. We will denote the set $S_{In}$ as *input space* and $S_{Out}$, i. e., the union of all $S_{Out}(i)$, as output or solution space. The subset $S_C \subseteq S_{In}$, where for all elements $i \in S_C$ the corresponding solution set $S_{Out}(i)$ is non-empty, is called *search space* and the corresponding elements are called *feasible*.

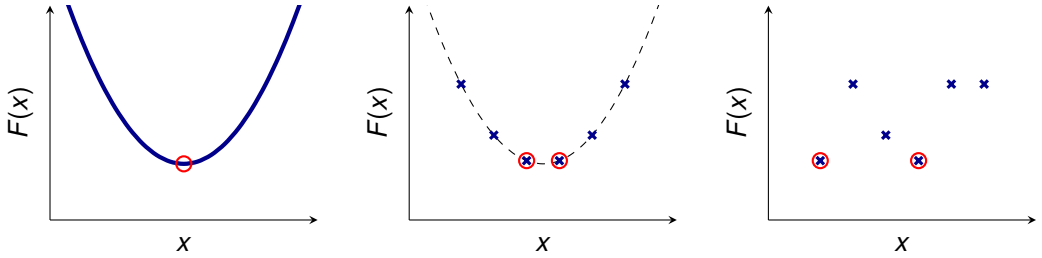
A *decision problem* asks for a function with a binary solution. In the case of the definition above, an exemplary decision problem is stated by a function $f_d$ which maps elements $x \in S_{In}$ to 1 iff the element has a non-empty solution set $S_{Out}(x)$ and otherwise to 0. More intuitively, an algorithm realizing $f_d$ answers the question, whether a given input state has a corresponding solution.

A *search problem* requires the identification of a solution for a given input. Considering the definitions above, a search problem is posed by a function $f_s$.

$$f_s : S_{In} \to S_{Out} \cup \{\varepsilon\},\ x \mapsto \begin{cases} y \in S_{Out}(x) & \text{if } S_{Out}(x) \neq \emptyset \\ \varepsilon & \text{otherwise} \end{cases}$$

Therefore, an algorithm solving this search problem will provide a random solution for a given input if such a solution exists. Obviously, it is related to the given decision problem since all elements which $f_s$ maps to $\varepsilon$ are exactly the ones mapped to 0 by $f_d$.

An *optimization problem* is a special search problem which provides a solution to a given input, which is optimal w.r.t a given cost or objective function. Optimal can then either mean that




(a) Continuous (b) Discrete (c) Combinatorial

Figure 3.1: Illustration of different optimization problems

the objective value should be maximized, typically when the objective function is bounded from above, or minimized, typically when the objective function is bounded from below. Since maximizing $F$ is identical to minimizing $-F$, any maximization can be converted to a corresponding minimization and vice versa. Again, based on the definitions above and given an objective function $F : S_{Out} \to \mathbb{R}_+$, a minimizing optimization problem or minimization problem is posed by a function $f_o$.

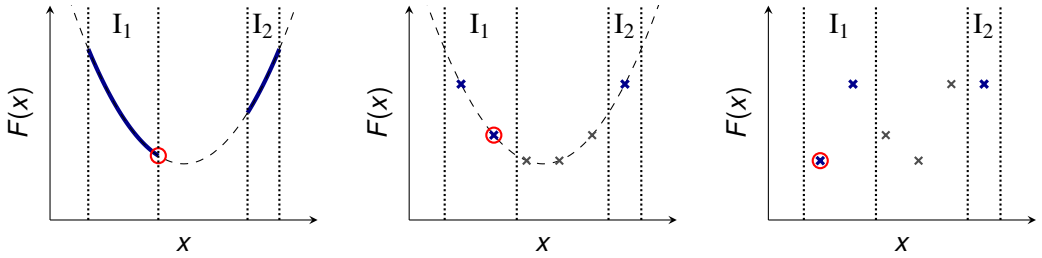
$$f_o : S_{In} \to S_{Out} \cup \{\varepsilon\},\ x \mapsto \begin{cases} \min_{y \in S_{Out}(x)}(F(y)) & \text{if } S_{Out}(x) \neq \emptyset \\ \varepsilon & \text{otherwise} \end{cases}$$

Note, that in the strict sense, solving the optimization problem $f_o$ implicitly solves $f_d$, but not necessarily $f_s$, since it only provides the objective value.

### 3.1.2 Classification of Optimization Problems

Depending on the nature of $S_{In}$ optimization problems can be further divided into discrete and continuous problems with great relevance to applicable solution methods. If $x \in S_{In}$ is a vector such that $x \in \mathbb{R}^n$, then $f_o$ is called *continuous* optimization problem, whereas if $x \in \mathbb{Z}^n$, then it is referred to as a *discrete* optimization problem. Figure 3.1a and Figure 3.1b illustrate this using a simple example of a convex objective function and $n = 1$, highlighting the respective optimal values. Generally speaking, discrete optimization problems are considered to be much harder to solve than continuous optimization problems [113, p. 7]. In the example shown in Figure 3.1, knowing $F$ to be convex and differentiable, one can apply simple methods from function analysis to determine the optimum for the continuous case. For the discrete case, however, it is neither immediately clear if one or two optimal values exist, nor how to directly find them with analytical means. A common approach for many discrete problems is therefore to convert them to a corresponding continuous problem if possible and solve this first. In the second step, the discrete points closest to the optima of the continuous problem are then analyzed for feasibility and objective value. This can in fact turn out to be a rather computationally intensive task since for $n$-dimensional input vectors, there can be up to $2^n$ candidate points in the search space for each of the optima.

A special case of discrete optimization problems are discrete *combinatorial* optimization problems, where the input space is defined by an underlying combinatorial choice, such as it often occurs in graph-related problems [16, 137, 142]. For such cases, the input often consists of a finite set of discrete structures [146, p. 1], which can be formally addressed by considering the domain of $x$ to be a finite index set to them. If $x$ is a mere index, however, then it may reflect




(a) Continuous (b) Discrete (c) Combinatorial

Figure 3.2: Illustration of different constrained optimization problems

no particular order of the structures identified by it, which can make it very hard to identify potential optima. This is exemplified in Figure 3.1c, where the function values do not exhibit any obvious pattern.

While optimization problems typically have a single objective function $F$, there exist problems where it is hard to formulate a comprehensive function due to the fact that the underlying real-world problems have several, potentially conflicting design goals. Such *multi-objective* optimizations can be considered to consist of several coupled individual optimization problems. In isolation, each could be solved to optimality, but in conjunction, their individual optima typically do not overlap. Therefore, the goal of such optimization is to identify the so-called Pareto frontier [176] between the individual goals in the solution space feasible to all of the problems.

Another classification feature apart from the objective and the input space is the presence of additional constraints, giving rise to the distinction between *constrained* and *unconstrained* optimization problems. Such constraints are often given as a set $\mathcal{C}$ of equalities and inequalities which directly or indirectly put restrictions on $x$ and thereby reduce the number of feasible solutions. Simple constraints may be no more than upper or lower bounds, but they can in fact be arbitrarily complex functions, such that constrained optimization problems can often be defined by sets of equations. Figure 3.2 shows the effects of adding 4 individual constraints for $x$ to the example problems from Figure 3.1, such that only the closed intervals $\mathrm{I}_1$ and $\mathrm{I}_2$ may contain feasible solutions. For the continuous case in Figure 3.2a this means that the intersections of any constraint and the objective function may introduce new optima, while in the discrete cases all potential solutions have to be checked against the constraints.

Finally, there exists the difference between *deterministic* and *stochastic* optimization problems. In the former case, all relevant parameters are known beforehand and the problem is fully deterministic. However, when modeling real-world problems some parameters may not be known exactly but can be described by incorporating a stochastic process into the optimization problems. The goal of such optimizations under defined uncertainty is to find solutions which are robust to errors.

## 3.2 Properties of Multi-Layer Optimization Problems

All of the optimization problems introduced in the last chapter require selecting a combination of discrete elements of graph structures with feasibility conditions to create solutions. For PTD and VTD, a subset of physical or virtual edges respectively have to be selected from the set of all feasible edges. For TDR, a number of virtual edges has to be combined into a sequence to form a feasible path for each demand, whereas for ICR a feasible series of physical edges have to be

found to create the required circuits. All of these problems are therefore inherently constrained combinatorial optimization problems.

### 3.2.1 Combinatorial Complexity

For PTD and VTD any possible edge available in the set of candidate edges $E_c$ may be part of the network to be designed. Therefore, the set of all solutions is initially the power set $\wp(E_c)$, such that the number of possible solutions is $2^{n_e}$ with $n_e := |E_c|$. The set of constraints $\mathcal{C}$ will likely include constraints which exclude some of these solutions, since it enforces that traffic demands have to be fulfilled. Since transport networks typically have traffic demands between all of their nodes, a solution will at least realize a spanning tree, such that the graph is connected, resulting in $n_v - 1$ edges with $n_v := |V|$.

This will immediately render all combinations with less edges infeasible, reducing the number of feasible solutions. The total number of these unconnected states $|S_\mathrm{u}|$ is the sum of the number of all combinations with less than $n_v - 1$ edges. For a given number of $i$ edges, the number of combinations can be determined using the binomial coefficient. Therefore, an upper bound on the number of feasible solutions can be determined by the following equation.

$$|\wp(E_c)| - |S_\mathrm{u}| = 2^{n_e} - 1 - \sum_{i=1}^{n_v-2} \binom{n_e}{i} \in \mathcal{O}\left(2^{{n_v}^2}\right) \tag{3.1}$$

While this reduction scales with the number of vertices, the increase scales with the number of edges, which typically grows much faster with increasing graph sizes. Further constraints such as 2-connectedness will further reduce the number of feasible solutions. However, this does not necessarily make the problem any easier, since a reduction in the feasibility impacts the output space, but may not automatically reduce the input space, such that the worst case remains in $\mathcal{O}(2^{n_e})$.

Routing problems such as TDR and ICR are much more complex, as they do not simply require one set of edges, but rather a sequence of edges for each route. We can consider the maximum number of paths between a pair of nodes in a full graph as an upper bound to the number of possible paths for a given route in any graph. The number of distinct loop-free paths is given by Equation (3.2) for a full graph of $n_v$ nodes[1].

$$c_\mathrm{paths}(n_v) = \sum_{i=0}^{n_v-2} \frac{(n_v-2)!}{i!} \tag{3.2}$$

Such a path is required for every demand $d \in D$. The number of demands can scale quadratically with the number of nodes and may be increased even further by the presence of QoS classes. Therefore, we can determine an upper bound on the number of solution candidates according to Equation (3.3), where $Q$ is the set of all QoS classes.

$$c_\mathrm{paths}(n_v)^{|D|} \leq \left(\sum_{i=0}^{n_v-2} \frac{(n_v-2)!}{i!}\right)^{n_v\cdot(n_v-1)\cdot|Q|} \in \mathcal{O}\left((n_v!)^{{n_v}^2}\right) \tag{3.3}$$

This level of complexity is largely intractable for exhaustive search approaches even for unrealistically small network graphs. However, in most practical applications the number of relevant

[1] Cf. Section D.2 in the appendix for our proof of this equivalence

paths is nowhere near this number, which may allow choosing a fixed number of path candidates, much smaller than the number of all possible paths.

Determining a multi-layer network configuration requires solving a joint version of many or even all of the problems above. Obviously, the resulting state space is excessively large and difficult to approach by a single method in all of its complexity, such that some kind of simplification is applied.

### 3.2.2 Modeling Accuracy and Tractability

A common approach to such multi-component problems is divide and conquer, meaning that the problem is solved by solving its constituent sub-problems and assembling the full solution from the individual sub-solutions. When a problem is *separable*, i. e., parts of it can be solved in isolation and their optimum is always part of the global optimum, this is a very efficient approach, especially when exact and fast algorithms are known for some or all of these sub-problems. While multi-layer problems do contain several sub-problems, as outlined in the previous section, they are typically neither separable, nor can they easily be solved.

Traffic routing can only be determined for a virtual topology, but any prior restriction to the virtual topology will therefore also impact the routing. Physical topology and connection routing in turn limit the virtual topology. The connection routing, however, determines the capacity, which needs to be sufficient for the traffic routing. Since these sub-problems all depend on each other, they cannot be solved individually such that the full multi-layer problem can be solved to optimality based on this. Furthermore, even for these sub-problems, there are no efficient and deterministic algorithms which could solve them. Exposing this full complexity to any optimizer will most likely result in problems so complex, that they cannot be solved with known computing resources in a time frame that is practically relevant. While such problems may theoretically be solvable given unlimited time, e. g., by exhaustive enumeration, they cannot be solved for any meaningful problem size. Problems of such complexity are called *intractable*.

In order to make large-scale problems tractable, they can be replaced by models of reduced complexity. This can be achieved by omitting or relaxing constraints, treating sub-problems as if they were separable, solving the problems using heuristics or even replacing them with more simple search problems. A common approach to multi-layer optimization problems is therefore, to replace the tight coupling of the layers by estimates and solve the individual problems using heuristics, as if they were fully separable. For example, VTD may be wholly omitted and the virtual topology statically assigned to be identical to the physical topology. For TDR, shortest paths can be considered a viable heuristic, such that demands may sequentially be routed on this topology. Rather than using all traffic demands individually, the total flow between neighboring nodes can be determined based on such a routing and used as input to ICR and WA.

While every simplification reduces model accuracy and will mostly likely have a negative impact on the solution quality for the overall multi-layer network, some aspects are more important than others. Routing traffic on shortest paths rather than using an optimized routing will most likely detract less from the optimality of the original problem, than replacing a topology optimization with a static assignment. However, traffic routing itself ultimately determines where capacity is needed and how efficiently it can be groomed, such that research suggests that an optimized traffic routing has more impact on the overall solution quality than an optimized topology [129].

The importance of sub-problems is also related to other scenario parameters. If traffic is very small compared to the circuit capacity, a regular hop-by-hop routing on shortest paths with an optimized topology may find the optimum faster than a complex routing optimization on

a dense, but fixed virtual topology. Similarly, a complex wavelength assignment may not be necessary if only few optical channels are used on any fiber. While it is typically true, that a reduced accuracy in modeling and solving will improve tractability, it is often difficult to assess, how big the negative impact is.

### 3.2.3 Aspects of Objective Functions and Constraints

Regarding the cost functions and demands this work is focused on deterministic single-objective optimization, noting that the fundamental approaches developed in this thesis are applicable to most other scenarios as well. Out of the most common goals outlined in Section 2.2.1.2 resource efficiency is the most universal aspect in network planning and will be used as the primary component of all objective functions discussed in this work. This is typically measured as the number of required network resources, especially in the form of NEs.

In terms of constraints, capacity is always considered, such that any demand shall receive sufficient capacity on all the resources it traverses. Additionally, some or all traffic demands may require specific QoS constraints, which will be given as a minimum amount of availability and latency.

Generally, the structure and properties of the input and search space determine which solution methods are applicable and how efficiently the problem can be solved. Especially interesting in this respect is whether meaningful gradient information can be obtained and if the number of optima can be bounded and how constraints are formulated.

For most combinatorial problems, including typical multi-layer problems, gradient information is difficult or even impossible to determine for the entire state space. One of the reasons for this is that when the combinatorial choice is represented as an index set, the index may be chosen arbitrarily and not reflect any property of the structure it represents. Consequently, the solutions of neighboring indices may have nothing in common and may feature vastly different objective function values making any gradient information beyond the immediate neighborhood purely coincidental.

Furthermore, a bad indexing scheme may also lead to a large number of local optima, i. e., points which are optimal for a subset of the input space forming a contiguous interval of $F$, also known as a *basin*. Apart from the index, the structure of some combinatorial problems itself may contribute to increasing the number of basins. Graph-related problems often contain many symmetries, e. g., due to paths of equal length, which may lead to many structurally different solutions of identical objective function values and therefore many optima.

For routing and topology design problems, changes to the chosen index value generally have non-trivial effects on the overall objective value of the resulting network configuration, such that a meaningful index with respect to the objective function is often hard to identify. Considering a VTD problem as an example, the removal of a single virtual link that leads to a network becoming disconnected, will potentially disrupt many traffic demands and make the solution infeasible. On the contrary, removing a virtual link in the network's periphery that is barely used may have little to no effect on the objective value. However, it is impossible to determine the importance of a virtual link without further assumptions on the scenario such as the exact traffic routing and its reaction to changes. For routing problems, the length of a path is often a very useful criterion in determining an index. A small-scale example visualizing the effects of constraints and index ordering can be found in Section B.2 of the appendix.

Finally, in constrained optimization problems the input space may feature intervals, where the objective function remains undefined, because a condition could not be met, rendering the solutions within these intervals infeasible. The corresponding constraints, which decide on the

feasibility, are called *hard* constraints. Depending on the exact nature of the constraints, these intervals may be localized to an area of the input space, but without further knowledge, neither their location, nor their number can easily be bounded.

In contrast to hard constraints, there are also so-called *soft* constraints. Not fulfilling a soft constraint does not make a solution infeasible, but dramatically increases the objective function value by a term, which typically dominates the original minimization goal. Since this term may grade the degree of constraint violation, it is also referred to as a penalty function.

In order to facilitate an efficient search through the input space, it is possible to relax hard constraints into soft constraints. Such a relaxation replaces undefined areas of the input space by areas of very high objective function values. Infeasible areas are hard to search, because infeasibility is a binary property and any two infeasible solutions are equally infeasible, such that there cannot be any gradient approximation between them. However, if the solution candidates are not infeasible, but rather result in high objective values due to penalties, these values can be compared. Therefore, a gradient approximation can be determined, which can help point the search towards areas of lower penalties.

For this monograph, we consider different constraints in different problems. However, all problems under investigation share a demand capacity constraint. This means that any demand is required to receive sufficient capacity on all the resources it traverses. Additionally, some or all traffic demands may require specific QoS constraints, which will be given as a minimum amount of availability and latency. Where not noted otherwise, all constraints are interpreted as soft constraints, which are typically set up as dominating penalty terms.

## 3.3 Overview on Optimization Methods

While there exist very efficient methods for very specific problems, e. g., Dijkstra's algorithm for finding the shortest paths in a graph, general discrete optimization problems are much more difficult to solve. If no special conditions can be exploited to formulate a domain-specific approach, often only an exhaustive enumeration of all feasible points is guaranteed to find the optima [113, p. 7]. Since combinatorial problems typically have very large solution spaces that grow rapidly with increasing input size [115, p. 222], this is only a feasible approach for very small problem instances. In order to use a more meaningful solution method, some additional information about the problem needs to be available. This can be in regard to $x$, $F$, or $\mathcal{C}$.

Ideally, $F$ and $\mathcal{C}$ can be formalized as a set of closed-form expressions which form a system of equations. While equation systems for general polynomial discrete problems are known to be incomputable by a single algorithm, even down to quadratic constraints [121], there exist different solution approaches for cases where the constraints remain convex or ideally even linear functions.

### 3.3.1 Linear Programming

If $F$ is a linear function of vector $x$ with $n$ components, it can be defined by its coefficient vector $p$ of $n$ components. If each of the $m$ conditions in $\mathcal{C}$ are such that they can be expressed as a linear inequality of the form ${a_m}^{\mathrm{T}} x \leq b_m$ with coefficient vector $a_m$ and value $b_m$, then the conditions can

be aggregated into a matrix $A = (a_{nm})$ and a vector $b = b_m$ such that the optimization problem can be written as a system of equations as follows.

$$\begin{array}{ll} \text{Maximize} & p^{\mathrm{T}}x \\ \text{subject to} & Ax \leq b \\ \text{and} & x \geq 0 \end{array}$$

Such systems of linear equations can be optimized by general solver algorithms and have been an active field of operations research for a long time. They can be distinguished by the domain of their variables as follows. For continuous, linear problems, abbreviated as LP, all variables are real-valued, while the variables of Integer Linear Programs (ILPs) are integer. The designation Mixed Integer Linear Program (MILP) is used to refer to a linear problem, where some variables are integer, while others are not. Depending on the structure of the system, different algorithms are applicable and many methods, ranging from Dantzig's simplex algorithm [170] to interior-point methods as well as more modern *branch-and-bound*, *cutting plane* and especially *branch-and-cut* algorithms have been developed and successfully employed to solve different complex optimization problems [194, pp. 160–168].

The great advantage of such solver algorithms is that they can provide exact solutions to the equation systems, such that if an optimization problem can be expressed accordingly, the optimal solution will eventually be found. The drawback is that the required runtimes may vary from seconds to hours or even days depending on the structure of the equations, which can be hard to determine a priori such that solvers are often run with time limits as termination conditions. While the original simplex algorithm has a worst-case time complexity of $\mathcal{O}(2^n)$ [112] for $n$ variables, more efficient algorithms have been developed for pure LPs that are able to achieve low-order polynomial time complexity [115, p. 258]. When variables are integer-valued, however, the problems become much harder. Solving ILPs is proven to be NP-complete [113, ch. 18.1], such that problem instances quickly become intractable for relevant problem sizes.

Solvers for ILPs and MILPs often try to identify separable sub-problems, transform them into corresponding LPs and try to solve these first and narrow down the possible integer solutions by introducing cuts, based on previous results. The efficacy of such approaches depends on the exact structure of the problem instance in rather non-trivial ways such that it is often difficult to tell how hard a problem instance is to solve [194, p. 110].

Finally, a very interesting aspect of linear problem solving is the duality principle[1]. While a solver is running, it may keep track of the so-called *primal* and *dual* solutions. From a simplified vantage point, the primal solution can be considered to be the best feasible solution currently known to the solver, while the dual solution is not a feasible solution to the original problem but represents a lower bound. Therefore, in a minimization problem the primal solution will improve over time by decreasing in objective value towards the dual solution, which in turn increases over the algorithm's runtime. The relative difference between the two values is known as the *duality gap*. Based on this, it is possible to quantify, how much better the actual optimum can at most be, although neither the exact optimum nor its value are known.

### 3.3.2 Heuristics and Numerical Approaches

When linear closed-form expressions for $F$ and $\mathcal{C}$ cannot be defined, either heuristics or numerical approaches need to be considered. These often exploit properties of the continuous

[1] Detailed information can be found in the books of Grover [115, pp. 231–235] and Schrijver [113, ch. 7.4].

equivalent of $F$, similar to the relaxation done by ILP solvers. We will refer to this relaxed objective function as $F'$.

#### 3.3.2.1 Approaches for Convex Problems

If the function $F'$ is known to be strictly convex and differentiable, such that only a single optimum exists, it is possible to exploit gradient information using *gradient descent methods* such as Newton's method to find the global optimum and check the closest integer points for feasibility and objective value. Newton's method generates a random starting solution $x_1$, determines the local value of Hessian matrix $H$ and generates a better solution $x_2$ by following the inverse gradient, according to the following function.

$$x_{n+1} = x_n - \left(H_{F'}(x_n)\right)^{-1} \cdot \nabla F'(x_n)$$

For $n \to \infty$ this approach is guaranteed to find the optimum of $F'$ and recursive probing of its neighborhood according to $F$ will eventually yield the integer optimum.

If a continuous relaxation of $F$ is either not differentiable or cannot be defined, neither the Hessian, nor the gradient can be computed and heuristics are required to improve the objective value. Heuristic approaches that are following the principle of iteratively generating subsequent points $x_{n+1}$ from a starting point $x_n$ can be categorized as *trajectory-based* methods. In this case, a trajectory-based heuristic may utilize an approximation of a local gradient, determined by a function $f_{\text{app}}$. For combinatorial problems, this often means that subsequent points are determined according to a local search in the neighborhood[1] $\mathcal{N}_d$ of $x_n$, such that the step function may become

$$x_{n+1} = x_n - \left(f_{\text{app}}(x_n)\right) = h_F(\mathcal{N}_d(x_n))$$

with local search heuristic $h_F$ being able to evaluate $F$ for the given points. There are various ways how $h_F$ can choose the candidate value $x_c$ from $\mathcal{N}_d(x_n)$.

*Hill climbing* chooses the first neighbor which improves the objective value and terminates if none improve it beyond $F(x_n)$. Obviously, this approach has a much lower rate of convergence since it lacks a global gradient and treats all dimensions separately. *Steepest-ascend Hill Climbing* tries to improve upon this by exhaustively testing all neighboring points to determine the one offering the largest improvement over $F(x_n)$, which can be computationally challenging for high-dimensional problems. *Stochastic Hill Climbing* offers a middle ground by sampling the neighborhood and accepting a candidate point $x_c$ with an acceptance probability $p$ based on the improvement in the objective value, e. g., such that

$$p = \max\left(\left(1 - \frac{F(x_c)}{F(x_n)}\right) \cdot \rho, 0\right)$$

with $x_c \in \mathcal{N}_d(x_n)$ and $\rho \in ]0,1[$ being a parameter which can be adjusted to match the problem at hand. This offers a flexible tradeoff between an accelerated rate of convergence and fine-granular search close to the optimum to avoid overshooting.

[1] More formally, we define the d-neighborhood $\mathcal{N}_d(x)$ of $x \in \mathbb{Z}^n$ as $\mathcal{N}_d(x) = \{p \in \mathbb{Z}^n \mid |(\|x\|_1 - \|p\|_1)| = d\}$.

#### *3.3.2.2 Approaches for Combinatorial Problems*

For typical combinatorial problems, the objective function is non-convex and therefore it may have any number of local optima. This is problematic for the basic methods above since they are only guaranteed to converge to the global optimum for convex functions. If $F$ has several local optima, then these methods will converge to the minima in whose basin the starting point $x_1$ is located. If the number of minima is known to be small, then Multi-Start Local Search (MSLS) approaches [159] may be employed. These approaches sample the search space for several starting points, such that several executions of the above methods may find different minima and the global minimum among them can simply be identified by searching through the local minima.

Often the exact number of minima cannot be determined a priori, such that there is no guarantee that all minima are found. For such problems, where $F$ is essentially a black box, only exhaustive enumeration is guaranteed to find a global optimum. However, the distribution of values in the search space is typically not random, since there is an underlying problem, which typically has some inherent, yet unknown or excessively complex structure. This leaves three possible groups of approaches to devise heuristics for these problems. The first is to approach the problem by a reduced-complexity proxy model which is built to be conducive to one of the approaches above. Especially devising models that can be optimized by ILP solvers is very common and as such will be discussed in more detail with respect to multi-layer optimization in Section 3.3.1.

The second group uses machine learning techniques such as Bayesian Optimization or Neural Networks in the hope that these can learn the properties of the objective function and minimize it accordingly. Especially Neural Networks can solve highly complex problems including path searching [111] very quickly once they have been properly designed and trained. This also highlights their drawbacks. For general combinatorial problems there is no single rule on which neural network structure should be employed. Furthermore, most approaches require a large amount of training data from which the targeted behavior can be learned. If such data is unavailable, approaches such as reinforcement learning can still be applied, but the training phases may become prohibitive in duration due to the large complexity of combinatorial problems.

Finally, general assumptions about the distribution and relation of values in the search space can be made and dynamically adjusted based on sampling the objective function to enable an efficient and effective analysis of the search space. This is the domain of a class of techniques, which offer generalized methods to approach black box objective functions, dealing with the absence of gradient information and knowledge about minima by defining general rules for sampling the search space on a global and on a local level, controlling the tradeoff between an exploratory search of the entire search space and a refining search of reasonably good points. Many such algorithms require only two problem-specific sub-algorithms. One for generating a starting point in relation to the search space and one for transforming a given point into another feasible point with some measure of distance in search space. These algorithms are known as *metaheuristic* algorithms.

### 3.3.3 Metaheuristic Optimization Approaches

Since these approaches do not describe heuristics for specific problems, but rather a general meta-approach to problem solving which needs further domain-specific adaptations, they are termed metaheuristics. Metaheuristics can be classified along several axes including their memory footprint and the amount of required adaptation, but the strongest distinction is between those

following a trajectory through search space and those examining multiple solution candidates simultaneously.

#### *3.3.3.1 Trajectory-based Metaheuristics*

Among the basic representatives of this category are multi-start approaches like MSLS, which utilize further heuristics to determine how the objective function should be sampled for meaningful starting points. More advanced metaheuristic approaches incorporate knowledge obtained from finding previous minima in both, the generation of starting points and guiding the refining search process. An example is Iterated Local Search (ILS) [150]. After starting from a random point, it finds the corresponding local minimum by iterative Hill Climbing. In the second step, it performs a so-called *perturbation* of this local minimum, i. e., it generates a new candidate point from that minimum according to a user-defined rule, which should ideally place this candidate outside of the current basin. If this candidate point matches a user-defined criterion, e. g., it covers a new search area or is a better solution altogether, then it is used as the new starting point to repeat this process. In each iteration the local optimum is stored if it is better than the best solution known up to this point. The algorithm terminates according to a user-defined criterion, e. g., after a given number of iterations or a target cost has been reached.

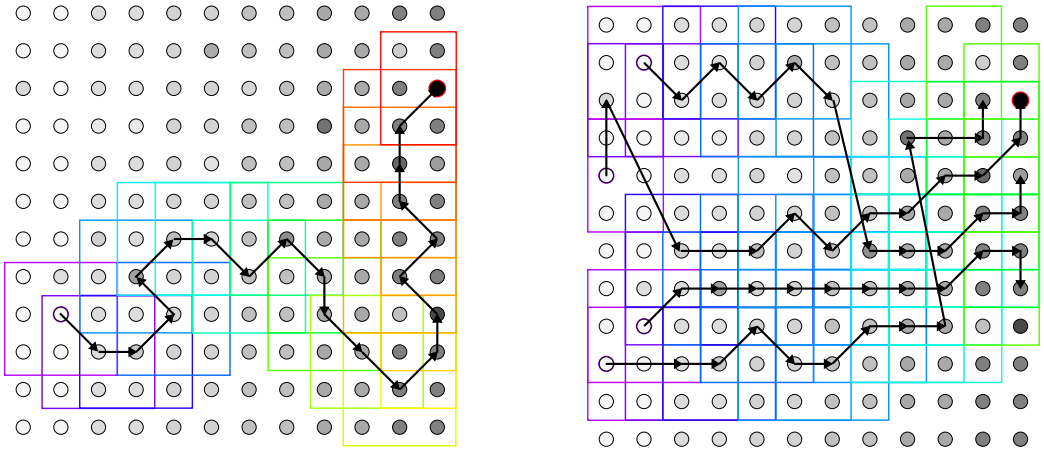
Most trajectory-based metaheuristics follow this general approach but apply more specific rules to perturbation and the selection criterion, since these are often difficult to define. For example, *Tabu Search* methods exclude the neighborhoods of recently found minima in the perturbation, such that if the excluded neighborhood is large enough, subsequent candidates will be located in different basins. Another example are *Simulated Annealing* methods, which will be presented in greater detail in Section 3.3.4. They allow the selection of intermediate candidates that initially can have worse objective values than the previous ones and therefore move "uphill" to escape the basins of local minima.

Figure 3.3a illustrates a very simple trajectory-based approach. The dots represent different solution candidates, where darker dots have better objective function values than lighter ones. The rectangles represent the inspected neighborhood from which the perturbation procedure will select the next solution candidate. Colors roughly indicate the progression of perturbations starting with the violet rectangle in the lower left. The approach always selects the best candidate from its current neighborhood, which has not been visited before, and breaks ties among equal solutions by choosing the one geometrically closer to the best-known dot so far. After 20 iteration steps with one perturbation in each it has reached the global optimum.

#### *3.3.3.2 Population-based Metaheuristics*

A second class of metaheuristic approaches are the *population-based* methods. While the trajectory-based methods move iteratively from one solution to the next, the population-based methods keep a set of solutions, the so-called population, which they operate on. The advantage of a population is that in determining new solution points, information from all known candidates can be used. This is especially beneficial when the search space exhibits a macro-structure, where several groups of basins exist in large distance to each other.

Figure 3.3b shows an approach that modifies the previous trajectory-based method to use a population of four searches, which are run in parallel. After every iteration, the individuals in locally bad neighborhoods are moved directly to the best-known positions among the set of all neighborhoods of the candidates within the current population. This way, promising regions can be explored quicker. This approach requires only 11 iteration steps to reach the



(a) Single trajectory search (b) Population-based search

Figure 3.3: Exploration of a two-dimensional search space by different approaches

global optimum. However, since every iteration step now includes four individual perturbations, their overall number has drastically increased to a total of 44. The number of perturbations per iteration step therefore has an impact on the performance and has to be chosen carefully. Most population-based metaheuristics therefore use very elaborate schemes to maximize the utility of the information gained through the population.

Most prominent among these are evolutionary, Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) approaches, which all adapt bio-inspired modes of problem solving. Evolutionary approaches are inspired by natural evolution such that they treat solution candidates as individuals which compete for survival in a common ecosystem. They will be elaborated upon in Section 3.4.

PSO [66] simulates the dynamic behavior of a cloud of particles, each representing a candidate point in search space. All particles have an initially large velocity in the solution space which is subject to change over time. The best-known position of each particle and the best-known position of all particles act as attracting forces which change the velocity vector of a particle, such that ideally particles are directed towards optima. This approach is very powerful since it requires very little adaption to the problem to be solved and the variable velocity naturally creates pseudo-gradients between different minima such that search spaces with macro-structures can be explored efficiently. The performance, however, depends on a good initialization of velocities, their update rates and starting points. In the worst case, bad choices for these parameters can prevent convergence altogether [25]. Furthermore, if a large number of similar local minima exist, velocity information updates may not be effective in guiding the algorithm.

ACO [49] mimics the foraging behavior of ants that create pheromone trails during their search to inform other ants about possible food sources. ACO typically works on graph representations of the optimization problem. A population of independent agents, corresponding to artificial ants, is created and each such agent follows the edges of the graph towards generating a full solution. They choose edges with a probability proportional to the intensity of existing trails on that edge, if possible augmented by a heuristic guidance. When their sequence of chosen edges forms a complete solution, this sequence is updated with a pheromone intensity correlating with the quality of the solution. A decay or *evaporation* factor constantly reduces the intensity on all edges by a small amount such that the intensity on less beneficial routes is reduced over

time and more beneficial routes are reinforced by the trails of increasing numbers of agents using them. Obviously, this type of algorithm is immediately applicable to problems that can easily be represented as paths in a graph. It works especially well when there is a straightforward guiding heuristic like the Euclidean distance on a plane, which is given for the Traveling Salesman Problem and its derivatives. For general combinatorial problems it's harder to formulate choices as a graph without that graph becoming excessively large.

#### *3.3.3.3 Properties and Performance*

Most metaheuristics share a number of traits. A termination criterion needs to be defined which may depend on the method (e. g., number of iterations) or the problem (e. g., known acceptable target cost). They all have their own set of parameters controlling their exact behavior (e. g., velocities for PSO or neighborhood size for Tabu Search). These metaheuristic-inherent parameters can be referred to as *hyperparameters* to distinguish them from problem-specific parameters. Furthermore, most metaheuristics require certain domain-specific adaptions or reformulations of the original problem such that it becomes compatible with their method of operation. This adaption can be simple as in the case of PSO, but also very complex as in the case of ILS, which is more of a framework than an immediately usable procedure. Finally, all of the outlined approaches begin from one or more random points in the search space. The selection of these points can have a significant impact on the rate of convergence, especially if the search space exhibits many local minima.

All of the metaheuristics mentioned above iteratively improve initial start solutions. This is generally a good fit for combinatorial optimization problems, because solving the relaxed search problem of identifying an arbitrary feasible solution is often relatively easy [115, p. 222]. Furthermore, metaheuristics can often easily be adapted and hybridized with other algorithms and domain-specific heuristics. Rather than randomly generating a starting solution, one can employ a basic heuristic to create an above-average quality solution. For example, considering the Traveling Salesman Problem one can generate an initial solution by repeatedly choosing the closest unvisited cities rather than having a random sequence of cities with arbitrarily long distances between them. Similarly, perturbation may be designed such that subpar solutions are avoided early on. The more information is known about the optimization problem, the better a metaheuristic can be adjusted, such that the resulting approach may be much more efficient than a generic one [137].

### 3.3.4 Simulated Annealing

Simulated Annealing [136] belongs to the trajectory-based metaheuristics and is related to Stochastic Hill Climbing and ILS. Its core functionality is based on an adaption of the Metropolis-Hastings algorithm [120, 161] which is a Markov Chain Monte Carlo method used to create a sequence of samples which approximate an unknown distribution. Rather than randomly sampling solution candidates, the algorithm follows a specific pattern inspired by the so-called *annealing* process from metallurgy. It accepts a solution candidate according to a certain acceptance probability $p$ according to the following function.

$$p = \min\left(1, \exp\left(-\frac{F(x_c) - F(x_i)}{T_i}\right)\right)$$

In this equation $x_i$ is the last solution in the trajectory and $x_c$ is a solution candidate. Depending on the value of the virtual temperature $T_i$, $x_c$ may be selected even if its cost is higher than that

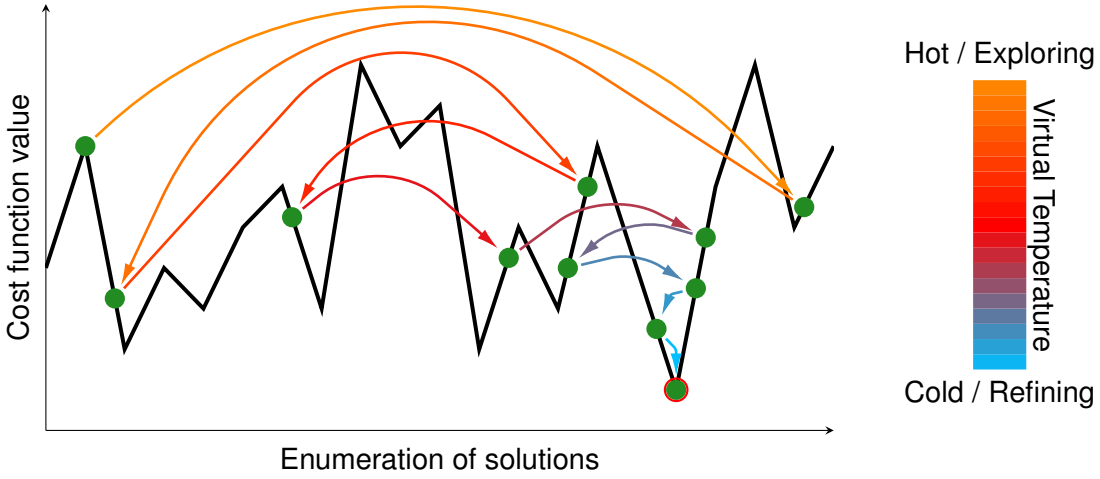



Figure 3.4: Simulated Annealing trajectory in search space

of $x_i$. This allows the algorithm to flexibly explore the search space without becoming stuck in local minima.

The value of $T_i$ is regularly updated during the runtime, such that following values are monotonically deceasing, and $T_i$ converges to 0. At this point, a candidate solution is only accepted, if it provides an actual improvement in the objective function value. The sequence of $T_i$ is known as the *cooling schedule* and is one of the most important parameters for this approach. Initially high values allow for a broad exploratory phase, while decreasing smaller values result in a gradual refinement. Figure 3.4 shows an idealized trajectory through search space, where the color of the arrows indicates the current virtual temperature.

New candidate solutions are created from the previous solution in the trajectory using the so-called *perturbation* function, which is a problem-specific function that has to be defined when adapting Simulated Annealing to a given optimization problem. The basic Simulated Annealing procedure is shown as Algorithm 1. Simulated Annealing is typically very memory-efficient, since it only needs to keep track of the best, the last and the next candidate solution. The only potentially expensive operations in terms of computation are the cost evaluation and the perturbation itself, but since both depend on the specific problem, no general statement can be made.

Further information including a more detailed analysis of the properties and capabilities of Simulated Annealing can be found in Section D.4 of the appendix.

## 3.4 Evolutionary and Genetic Algorithms

Evolutionary Algorithms (EAs) belong to the population-based metaheuristics. They are based on the observation that natural evolution resembles an optimization process, where individuals adapt to their surrounding ecosystem. More adapted individuals are more fit to survive in their surroundings, such that the individuals of higher fitness have a higher chance to mate and create offspring. The offspring inherit a mix of their parents' traits and therefore have a chance to exhibit an even higher fitness by combining different traits. Through natural selection, the population will evolve over time to optimally adapt to its ecosystem.

EAs therefore try to mimic this behavior by interpreting individuals as solution candidates and their fitness as their objective value. The population of solution candidates is then evolved

**Algorithm 1** Basic Simulated Annealing Algorithm

**Require:** Objective Function to minimize $F$
**Require:** Monotonically decreasing sequence of temperature values $T_n \in \mathbb{R}^0_+$
**Require:** Termination condition $c_t \to \{$ **true** , **false** $\}$
**Require:** Initial solution $x_s$
**procedure** SIMULATEDANNEALING($F$, $T_n$, $c_t$, $x_s$)
  **let** $i \leftarrow 0$ ▷ Number of iterations
  **let** $x_i \leftarrow x_s$ ▷ Current solution in iteration $i$
  **let** $x_{\text{best}} \leftarrow x_s$ ▷ Best solution currently known
  **while not** $c_t()$ **do**
    **let** $x_c \leftarrow$ PERTURBATION$(x_i)$ ▷ Problem-specific procedure, cf. Section D.4.3
    **let** $r \leftarrow$ RND$(0,1)$
    **if** $r \leq \min\left(1, \exp\left(-\frac{F(x_c)-F(x_i)}{T_i}\right)\right)$ **then**
      $x_i \leftarrow x_c$
      **if** $F(x_c) < F(x_{\text{best}})$ **then**
        $x_{\text{best}} \leftarrow x_c$
      **end if**
    **end if**
    $i \leftarrow i+1$
  **end while**
  **return** $x_{\text{best}}$
**end procedure**

by iteratively applying evolutionary operators to create new solution candidates based on the previous ones. High-quality solutions are fostered by exerting an artificial selection pressure depending on their objective value.

The details of how offspring candidates are created by these operators and how the optimization problem is represented vary significantly between different classes of these algorithms. For example, in Differential Evolution [197] and Evolution Strategies [17], the individuals are typically real-valued vectors, which may represent solution candidates for difficult equation systems. In Genetic [196], Evolutionary [68], and Gene Expression Programming [80], individuals are computer programs intended to solve a given problem and the operators evolve their program code or the parameters used in predefined code.

Genetic Algorithms (GAs) are closer to natural evolution. Their defining characteristic is that their operators do not immediately work on the candidate solutions, but rather that all candidates are represented by their genes to which the operators are applied. The genes are then used to construct the actual solution candidates and their objective value can be determined. Thus, an individual can be of arbitrarily complex structure, while the evolutionary operators can be kept simple, since the complexity of the genes is far lower than that of the entire individual. This monograph will focus on GAs and their extensions.

### 3.4.1 Definitions and Background

Any biological individual is largely defined by a sequence of genes which are carried in the individual's *chromosomes*. The position of a gene within this sequence is called *locus*. The gene sequence consists of nucleotides that exhibit one of four discrete states for most biological

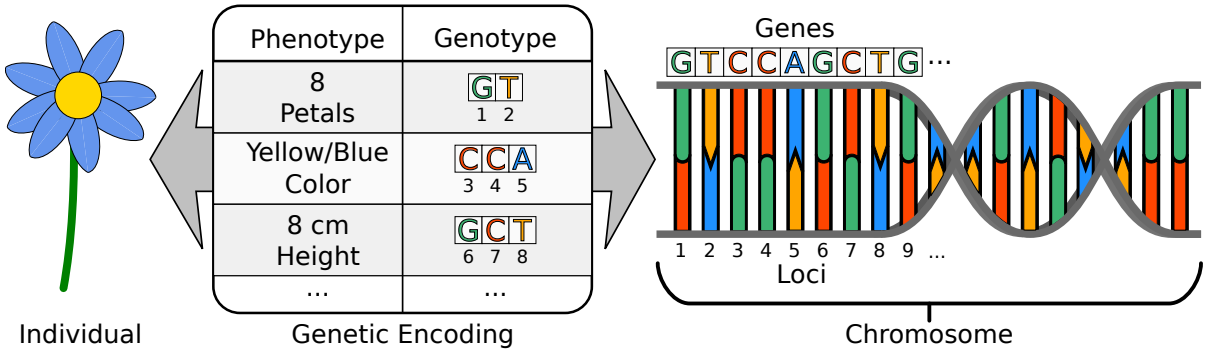



Figure 3.5: Idealized example of a chromosome encoding traits of a flower

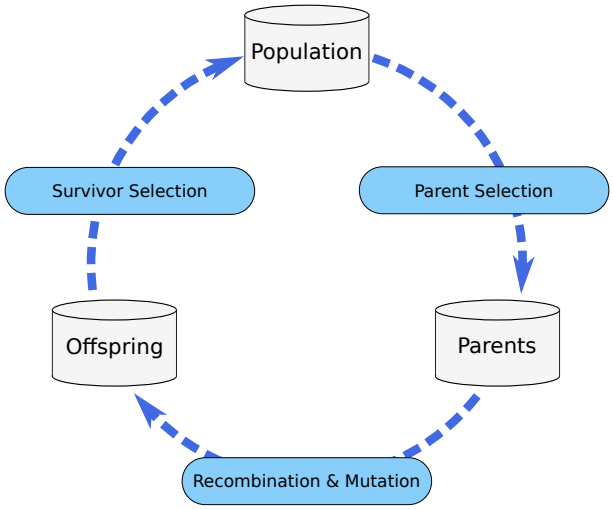



Figure 3.6: Evolutionary cycle

creatures. Subsets of genes typically encode different traits. The genetic code of a trait is called *genotype*, and the resulting trait is called *phenotype*. Figure 3.5 exemplifies these terms by showing traits of a flower encoded in a chromosome structure.

The gene sequence of an offspring consists of contiguous subsequences of its parents' genes and therefore it can inherit phenotypes if the entire subsequences representing the corresponding genotypes are present. The process of assembling an offspring's chromosome from its parents' chromosomes is called *recombination* or *crossover*. Furthermore, environmental effects lead to changes in the states of individual genes, which can result in entirely new features which may further improve or reduce the fitness. These random changes are referred to as *mutations*.

The set of individuals that may become parents is the population. While less fit individuals are gradually removed from the population, they are replaced by more fit individuals from their offspring. Which individuals become parents, which are removed, and which offspring enter the population is subject to a *selection* process depending on their fitness. Therefore, the population evolves from one generation to the next as shown in Figure 3.6.

### 3.4.2 Algorithmic Approach

The concepts for the original GA approach have been developed by John Holland and summarized in his 1975 book [125] on the topic. It is one of the more literal adaptions of natural evolution and does not interact directly with the solution candidates, but rather with a virtual

genetic encoding that defines a solution by its virtual genes in a virtual chromosome. Initially, it builds a starting population which is typically comprised of a number of randomly generated chromosomes and evaluates these for their objective values. After that it iteratively executes the evolutionary cycle as shown in Figure 3.6 using the basic genetic operators of parent selection, recombination, mutation, and survivor selection. In each cycle it performs a predefined number of such perturbations generating offspring solution candidates from two parent solutions. The basic approach is shown in Algorithm 2.

**Algorithm 2** Basic Genetic Algorithm

**Require:** Objective Function $F$
**Require:** Termination condition $c_t \rightarrow \{$ **true** , **false** $\}$
**Require:** Set of initial chromosomes $C$
**Require:** Number of evolutions per generation $n_o \in \mathbb{N}$
  **procedure** GENETICALGORITHM($F$, $c_t$, $C$, $n_o$)
    **let** $O \leftarrow \emptyset$ ▷ Offspring as set of chromosomes
    **let** $P \leftarrow C$ ▷ Population as set of chromosomes
    **let** $c_{\text{best}} \leftarrow \arg\min_c(F(\text{DECODE}(c)))$ ▷ Best solution known among $c \in C$
    **while not** $c_t()$ **do** ▷ Termination condition
      **for** $i = 0$ **to** $i = n_o$ **do**
        $\langle p_1, p_2 \rangle \leftarrow$ PARENTSELECT($P$)
        $\langle o_1, o_2 \rangle \leftarrow$ RECOMBINE($\langle p_1, p_2 \rangle$)
        $O \leftarrow O \cup \{\text{MUTATE}(o_1)\} \cup \{\text{MUTATE}(o_2)\}$
      **end for**
      **for all** $c \in O$ **do**
        **if** $F(\text{DECODE}(c))) < F(\text{DECODE}(c_{\text{best}}))$ **then**
          $c_{\text{best}} \leftarrow c$
        **end if**
      **end for**
      $P \leftarrow$ SURVIVORSELECT($P$, $O$)
      $O \leftarrow \emptyset$
    **end while**
    **return** DECODE($c_{\text{best}}$)
  **end procedure**

Originally, these virtual genes were simple bit strings, which made the design of genetic operators simple and efficient to implement in computer systems. For example, the recombination could be realized as *1-point crossover*, randomly determining a locus and setting all of the offspring's chromosome bits up until that locus identical to the bits of the first parent and all subsequent bits identical to those of the second parent. The mutation operation could be realized as a simple bit flip in a randomized locus. Since its beginnings, genetic encodings have been extended to include real-valued and integer-valued vectors as well as abstract structures, each with a wide variety of accompanying genetic operators. Especially interesting are so-called hybrid and *memetic* genetic algorithms which combine the basic evolutionary approach with additional heuristics and problem-specific knowledge.

### 3.4.3 Theoretical Behavior

While it is hardly contested that evolution per se "works", there are only few solid and general findings as to how exactly operators and encodings need to be designed such that convergence is probable, much less can be guaranteed. Holland had suggested the notation of Schemata [125, pp. 66–74] which Goldberg summarized in his 1989 book on Genetic Algorithms [107, pp. 41–45] as the *Building Block Hypothesis*. Simplified, it states that small substructures of a chromosome contribute differently to the emergent overall solution fitness, and it is suggested that recombination is an efficient operation for the proliferation and exploration of combinations of such building blocks. Although numerous works with empirical evidence of genetic algorithm performance exist, it has also been shown that the standard mutation operators typically render the algorithms incapable of any proof of convergence [237].

Regardless of how exactly the operators can be formalized and parameterized, the population itself may already provide a substantial benefit over purely trajectory-based approaches as well. It allows for a broad sampling of the solution space and is more resistant to becoming stuck in local minima since typically not all solution candidates will converge to the same local minimum. Some algorithm variants, especially when using smaller populations due to external constraints, employ diversity-control methods to restrict recombination of individuals with the goal of maintaining a broader coverage of the search space. This can be done by *fitness sharing*, such that only a limited number of solutions with identical objective values are admissible, by *crowding*, where the number of genetically similar candidates is restricted, or by running several entirely separate populations which only exchange chromosomes after a larger number of generations.

### 3.4.4 Problem Adaption

Since the genetic operators are defined on chromosomes, it is required to find a meaningful encoding of the problem and its solutions. Whenever problems can be described by numerical parameters, it is a straightforward way to encode solutions as vectors containing these parameters. Depending on the domain of the parameters and therefore the resulting vector, different genetic operators can be used. For combinatorial problems, a choice of discrete structures may be represented by a bit string where each locus corresponds to a structure and its value indicates if it has been chosen. When there are multiple discrete realizations per structure, an integer vector where gene values represent an index set are a possible choice.

Whenever the original problem is *separable*, this property should be exploited and may be reflected directly in the coding and its processing. While this is not typically the case for combinatorial problems, there may exist parts of the problem that may even be linked by common parameters, such that they can be solved by other, possibly fully deterministic algorithms. In the biological analogy this would correspond to knowing a priori which subsequences of a chromosome result in specific traits, such that only these traits are relevant, not their exact genetic composition. This is the domain of memetic algorithms, where the genetic algorithm only operates on these parameters, thereby significantly simplifying the original problem. Such encodings, if they exist for specific problems, are typically highly domain-specific and often difficult to find and inherently hard to generalize.

The difficulty in designing an encoding stems from that fact that one may easily under- or overrepresent the actual solutions. A smaller search space is easier to explore, but when the encoding is missing parts of the solution space, it might miss the actual optimum as well. On the other hand, an overrepresentation may negatively impact performance, especially in

combinatorial problems where a few redundant chromosome states easily lead to an exponential growth of the search space. The sequence of genes in the chromosome should ideally also represent some structural aspect conducive to the solution of the problem. If such an aspect can be identified, its inclusion into the design of the encoding has a chance to significantly improve the performance since recombination operates on sequences of genes.

### 3.4.5 Hyperparameters

Genetic Algorithms require a comparatively large number of hyperparameters, especially since there are various genetic operators, many of which also require their own specific hyperparameters. According to the mechanism they are attached to, they can be divided into the termination condition, hyperparameters for the population, parent selection, survivor selection, mutation, and recombination. Carefully controlling the effects of the genetic operators is imperative to avoid divergent behavior.

#### *3.4.5.1 Population and Evolution Rate*

The size of the population has to be large enough to allow a sufficient number of differing gene sequences to exist, such that a meaningful level of diversity is possible. If the number of individuals in the population is too low, then only a small number of different traits can exist at any point in time and improvements will result less from recombination but need to be evolved by possibly many mutations. Furthermore, recombination may counteract the effects of mutation, since the few existing good gene subsequences will enter all chromosomes quickly, which can result in premature convergence. Larger and more complex chromosomes will therefore also require larger populations to sufficiently explore the search space. A drawback of larger populations is the increased memory footprint.

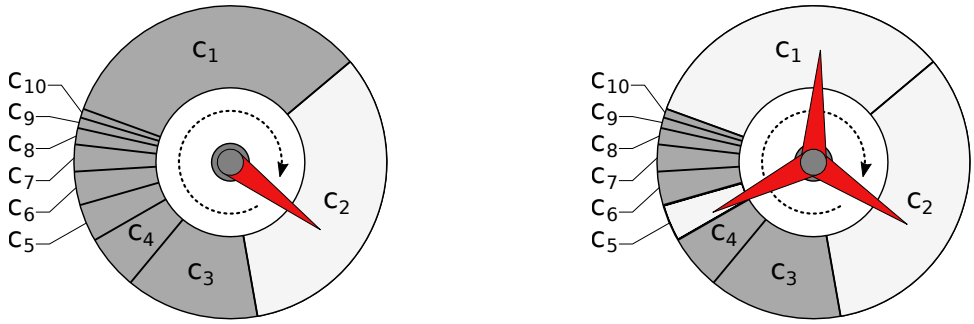
The population size also needs to be considered together with the number of offspring created in each cycle, also known as the *evolution rate*. A large population and a very small evolution rate mean that only few individuals are selected as parents and therefore large parts of the population contribute little to the evolution process.

The evolution rate is the primary driver for change in the population and allows for several different modes of operation in conjunction with the selection schemes. There exists the so-called *generational model* in which all parents are automatically removed from the population and replaced by an equal number of their offspring. In the *steady state model* evolution rates are much lower and only a part of the original population is replaced by their offspring, which may even need to compete with their parents for selection into the next generation of the population.

#### *3.4.5.2 Parent Selection*

The objective of the parent selection is to identify potentially good and bad individuals, selecting the good ones as parents to perpetuate beneficial traits. Most selection processes in some way utilize the objective value of individuals to determine the probability of selection. They may determine the probability proportional to the rank, i. e., the index in a list of individuals sorted by objective value, the difference in absolute or normalized objective values.

In parent selection, the most popular versions are Fitness Proportional Selection (FPS), also known as Roulette Wheel Selection (RWS), Stochastic Universal Sampling (SUS) [8], and *tournament selection*. FPS works by normalizing the objective values such that their sum corresponds to the $[0, 1[$ interval and all individuals receive a proportional share. It then proceeds by generating a random number and adding the normalized objective values in arbitrary sequence




(a) Roulette Wheel Selection (b) Stochastic Universal Sampling

Figure 3.7: Comparison of selection operators for a population of 10 candidates

until the sum is larger than this number, picking the individual that was added last. This is analogous to spinning a roulette wheel and assigning circle segments to individuals such that the length of their arc is proportional to their objective value, thus giving better individuals a higher chance of being selected. This is illustrated in Figure 3.7a, where the individual $c_2$, which has the best objective value, has been selected.

SUS is closely related to FPS, but rather than selecting $n$ individuals using $n$ random numbers, SUS draws a single random number $r$, selects the corresponding individual and then selects the remaining $n-1$ individuals by taking evenly spaced steps from $r$, such that $n$ multiplied by the step size is 1. This is analogous to a roulette wheel with $n$ arms as shown in Figure 3.7b, where a single spin with $n=3$ points to $c_1$, $c_2$ and $c_5$. In fact, for this example it is guaranteed, that $c_1$, $c_2$ and any one of $c_3$ to $c_{10}$ will be selected, whereas RWS has roughly a 30 % chance of missing all smaller segments in 3 consecutive spins. Therefore, SUS increases diversity by having less bias towards the objective value, which can be helpful in scenarios where a dominating individual represents only a local optimum.

Tournament selection has an additional hyperparameter specifying the number of contenders. These are drawn at random from the population and "compete" pair-wise such that the better objective value signifies the winner. There are also stochastic versions, where the winner is picked proportional to the number of competitions it has won. By combining random draws and fitness-based decisions, this approach can strike a good balance between diversity and rate of convergence but requires more computation time.

Other GA variants also intentionally introduce additional bias to enhance the rate of conversion. E. g., the Biased Random-Key Genetic Algorithm (BRKGA) divides the population into an "elite" and a regular subpopulation, where the elite population contains the best individuals. Here, parents are selected such that one is uniformly chosen among the elite and the other among the regular population.

#### *3.4.5.3 Survivor Selection*

Just like in parent selection, the selection operators try to determine the quality of individuals, but here the objective is to evict the bad ones from the population. They may use the same methods of selecting them, i. e., based on a ranking system, absolute or normalized objective values as before, sometimes with more aggressive parameters or cutoffs. *Sigma Truncation*, e. g., calculates the selection probability relative to the average objective value in the population, but sets the survivor probability to 0 for individuals that are worse than a predefined number of

standard deviations below the average. Additionally, the age of an individual, measured in the number of generations it has maintained in the population, may also play a role for steady-state approaches.

The most important strategies used are round-robin and stochastic tournament, replace worst, $(\mu + \lambda)$ and $(\mu, \lambda)$ selection. The tournament selection approaches are similar to their counterpart in parent selection, but they work on the union of the parent population and their offspring. For round-robin tournaments, all individuals compete with a fixed number of randomly drawn individuals and the ones with most wins become the new population. For stochastic tournaments, the number of contenders and tournaments is fixed and all contenders are drawn at random, such that not all individuals may compete, which signifies the stochastic element.

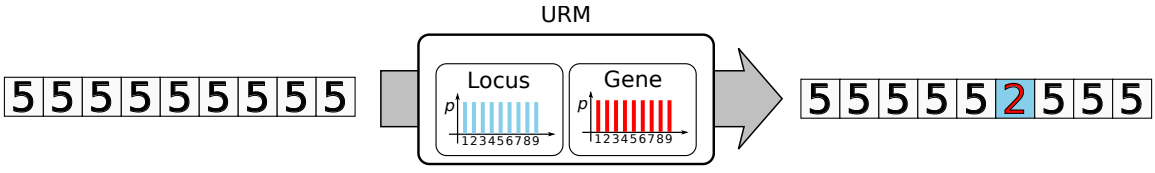
Replace worst requires the population to be larger or at most equal to the number of offspring. It first evicts a number of the worst elements of the population which is equal to the number of offspring. Afterwards it adds all offspring to the remaining population. Obviously, if both are of equal size, then the entire population is replaced by their offspring unconditionally, such that this approach is heavily influenced by an individual's age, rather than its fitness. $(\mu + \lambda)$ selection is similar to this but works by merging and sorting the current population and the offspring, before evicting the worst individuals. This means that high-quality individuals of the last generation can be transferred to the next generation if they are sufficiently good, which is not possible for the simple replace worst-approach. $(\mu, \lambda)$ selection can be considered a middle ground between the two. It requires the number of offspring to exceed the number of individuals in the present population. All elements of the last generation are removed and only the best among the children are selected for the following generation.

Finally, there is a special strategy known as *Elitism*. Generally, any individual, including the one with the best objective value, may be evicted from the population with a certain probability. Since this elite individual may be closest to the actual optimum, it is most likely detrimental to remove it from the population entirely, since this may significantly lengthen the runtime. The original version of Elitism therefore suggests that the best known individual should be able to bypass survivor selection and remain in the population, possibly replacing the worst or a random individual. However, when the elite individual is much better than the others, the population may quickly evolve to reflect only small changes from this individual, although the actual optimum may be vastly different. To prevent this premature convergence, a more advanced version of Elitism considers not one, but several elite individuals and may even require them to exhibit a certain genetic diversity to be eligible.

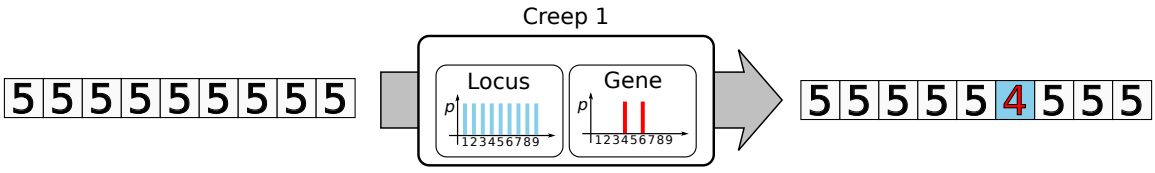
#### *3.4.5.4 Mutation*

The sheer number of mutation operators suggested in literature is well beyond the scope of this work such that only the most well-known versions shall be introduced. The exact mode of operation of the mutation depends on the genetic encoding. There exist versions for Boolean, real-valued, integer-valued, and abstract permutation encodings. Common to all of them is that mutation introduces a random change, which often consists of choosing one or more loci and modifying the current gene value. As such mutation is closely related to the simple perturbation operators of other algorithms.

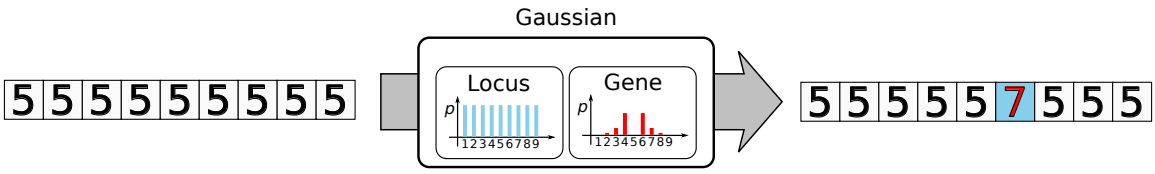
The most basic mutation operator, which we refer to as Uniform Random Mutation (URM) chooses a single locus and randomly changes its gene value to any other possible value with equal probability. For a binary encoding this corresponds to a single bit flip in the chromosome. More advanced versions offer additional hyperparameters, often regarding the number of affected loci or, for real- and integer-valued genes, a restriction on the target gene value. For example,



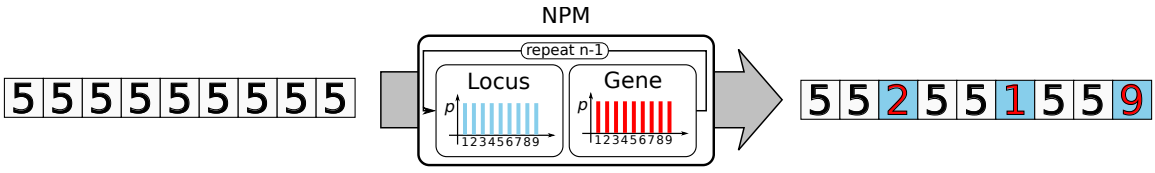

(a) Uniform Random Mutation



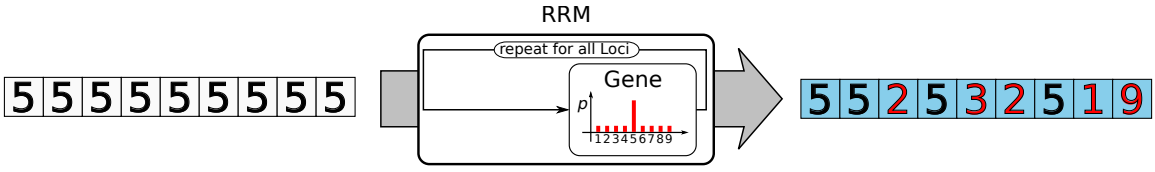

(b) Creep Mutation with step size 1




(c) Gaussian Mutation




(d) N-Point Mutation with $n = 3$




(e) Random Reset Mutation

Figure 3.8: Comparison of mutation operators for a chromosome with gene values 1 to 9

when the genes are in the integer domain, the so-called *creep mutation* either increments or decrements the previous gene value by a small value, typically with equal probability. The value itself is randomized and bounded by the maximum step size, which is the hyperparameter for this operator. Figure 3.8b shows this for a maximum step size of 1. In contrast to this, the *Gaussian mutation* draws the new value of the gene from a Gaussian distribution centered around the current gene value and takes its standard deviation as an additional hyperparameter.

Regarding the number of affected loci, the most common approaches are N-Point Mutation (NPM) and Random Reset Mutation (RRM). NPM essentially applies URM to $n$ uniformly selected loci in the chromosome, while RRM applies URM to each gene with a uniform probability, which marks another hyperparameter. This is equivalent to a distribution where the previous gene value is more likely than the others as depicted in Figure 3.8e.

Figure 3.8 illustrates all of the aforementioned operators for a chromosome of 9 genes, where each gene can take a value between 1 and 9. All mutations can be broken down to a choice of affected loci (marked blue in the offspring) and a choice regarding the change in value


(a) Cut and Crossfill


(b) N-Point Crossover with $n = 3$


(c) Uniform Crossover with $p = 0.5$

Figure 3.9: Comparison of recombination operators for an integer-valued chromosome

(marked red in the offspring). The graphs representing these choices show exemplary probability mass functions. Mutation can be a powerful driver in developing new traits, but the number of changes and possibly the step size need to be chosen with care. There even exist versions of genetic algorithms relying more or even solely on mutation (sometimes traditionally referred to as evolutionary algorithms). Aggressive mutation can create new phenotypes more quickly but may also disrupt existing ones more quickly as well.

#### *3.4.5.5 Recombination*

Many recombination or crossover operators exist in literature. The most common ones are uniform, *cut and crossfill* and *N-point crossover*. For the uniform or uniform random recombination operator, which we shall refer to as UXO, there is a 50 % chance that any given gene comes from parent 1 or parent 2. While this is a simple and efficient operation, it is not very conducive to the concept of the building block hypothesis, since traits with large genotypes are likely to be disrupted during random recombination. A more recent approach to UXO, called *biased* UXO (BUXO), explores the possibility of selecting a parent gene using a different probability value then $p = 0.5$. In this scheme, the genes are not split equally between the parents, but rather the selection probability is increased for the more fit parent.

Cut and crossfill, also known as single-point crossover, is a widely used approach in which a locus is randomly selected and all genes up to this locus are taken from parent 1 and all genes beyond this locus from parent 2. This approach helps to pass on intact genotypes but may be inefficient for very long chromosomes. The straightforward extension is the N-point crossover where the hyperparameter $n$ gives the number of loci to randomly select and the genes in-between are alternately copied from parents 1 and 2. We shall abbreviate the former as Cut-and-Crossfill Crossover (CNCXO) and the latter as N-Point Crossover (NPXO).

Figure 3.9 shows illustrations of the three major recombination operators. While all of these show only positional choices, it should be noted that further operators exist, which also consider gene values in recombination. E. g., when the values represent arbitrary integer variables, rather

than an index to a finite set of categories, it can make sense to blend gene values by calculating average values. However, these are out of scope for this monograph, which is primarily concerned with combinatorial choices.

Whenever there are clear structures emerging from subsequences, recombination is expected to be most effective in combining these and evolving better solutions. When the order of genes plays only a minor role and many discrete states exist, more aggressive mutation may be more efficient in exploring the search space.

#### *3.4.5.6 Termination*

Genetic Algorithms can use the typical termination conditions relating to the solution quality, but there are also a number of specific values. The number of evolutions of offspring or the number of generations can be bounded. Furthermore, the genetic algorithm can only evolve new solutions efficiently as long as there is sufficient genetic diversity in the chromosomes of the population, such that the genetic diversity can be used as a termination criterion as well. However, it should be noted that this may not be practical for very large populations if diversity has to be determined by comparing all elements of the population.

### 3.4.6 Aspects of Implementation

Genetic Algorithms are inherently parallelizable since most of their operations occur in isolation. Selection, Mutation, Crossover can be run in parallel. Only the survivor selection requires a global view with exclusive access when interacting with the population. This is a highly beneficial trait in times where computer architecture tends to scale out, rather than scale up.

The memory footprint of a genetic algorithm primarily depends on the size of the population and its individuals since none of the common operators need much memory or processing. In fact, depending on the underlying problem, executing the decoding function, or evaluating a decoded individual for its objective value may be the most computationally intensive tasks. In this case, it makes sense to store the objective values with the individuals in the population, rather than repeatedly execute those procedures on the same chromosomes.

The largest problem in using genetic algorithms is finding a good genetic code for the problem to be solved, such that a good compromise between performance and solution quality is possible.

## 3.5 Solution Methods for Multi-Layer Problems

The three most common approaches to solving multi-layer problems are the following: The classic method is to treat them as search problems and solve their constituent sub-problems independently using exact and fast algorithms. While the algorithms are well-understood and highly efficient, the solutions will most likely be of sub-par quality compared to a proper optimization. The second approach is to simplify and linearize the original problem in order to find a corresponding ILP. This requires a high degree of knowledge on the problem as well as on the method. When both are given, high-quality solutions can be obtained, but the runtime may be large. The third approach are metaheuristics, which are more simple to employ for large problems than linear programs and may find solutions better than the classic approaches at shorter runtimes than an ILP-solver can. While other approaches using problem-specific heuristics do exist, they typically are specific to certain scenarios or conditions.

### 3.5.1 Classic Methods

This approach of sequential network design can be found in commercial network planning software [38, 67] for offline use but is also at the core of control plane protocols for online use. Typically, the control planes do not typically consider changes to the entire network for reasons of performance and predictability but rather focus on solving local problems quickly. A new demand will therefore be provided with a locally optimal solution given the already established configuration. Offline planning tools may also sequentially route traffic in order to stay compliant with control plane behavior, but may also determine static elements such that the resulting configuration will be as efficient as possible. This includes defining weights for routing algorithms, but also determining a fixed virtual topology facilitating an efficient routing.

#### *3.5.1.1 Routing Algorithms*

For traffic and connection routing alike, demands are often routed individually, mostly making use of Dijkstra's algorithm (e. g., in IS-IS [205], OSPF [207] and OSPF-TE [225]) or the Bellman–Ford algorithm (e. g., in RIP [206] and BGP [53]), possibly enhanced to exclude paths that are not feasible, e. g., due to known resource limitations or not being QoS-compliant, by limiting the search graph to its feasible subset.

Furthermore, when disjoint paths are required between the same pair of nodes, there may be the additional requirement of being link- or even node-disjoint paths, i. e., that the paths do not include the same edges or nodes. While node-disjoint paths are rarely required in transport networks, pairs of link-disjoint paths are often required to avoid single points of failure. There exist specific algorithms for this task which are more efficient than the general approach of running subsequent iterations of regular shortest path algorithms. Most notable among these are Bhandari's [18] and Suurballe's algorithms [259]. The first can be considered to extend the Bellman–Ford algorithm, while the second extends Dijkstra's algorithm.

The advantage of the Bellman–Ford algorithm and by extension Bhandari's is that they support graphs of negative edge weights, which Dijkstra's and its derivatives do not. The advantage of the latter algorithms, however, is an asymptotically smaller time complexity, which makes these algorithms preferable when simple edge weights are sufficient. While all algorithms solve their respective problems in polynomial time, Dijkstra's and Suurballe's offer a worst-case time complexity of $\mathcal{O}(|V| \cdot \log |V| + |E|)$ and the other two algorithms of $\mathcal{O}(|V| \cdot |E|)$.

#### *3.5.1.2 Approaches for Topology Design*

Depending on the operating paradigm the virtual topology may either be statically pre-planned or reactive. In the reactive model network resources are capacitated and a virtual topology is mostly driven by the arrival or departure of connection requests from the upper layer, such that the state of the topology itself is a product of the sequence of demands and the network state at the time of arrival. For example, in an Automatically Switched Optical Network (ASON), the upper layer may request bandwidth for a new traffic demand from the lower layer which in turn triggers its own routing algorithm with these parameters to create a new circuit. The lower layer may explicitly advertise the remaining unused capacity, such that the upper layer can partition it into several traffic routes usable for future traffic demands. The drawbacks are that this may lead to resource fragmentation over time and that network operators have a hard time to predict where bottlenecks will emerge.

For transport networks, there is always traffic to be expected between any node pair and the required bandwidth may even be forecast with a certain degree of accuracy such that the

virtual topology can be preplanned. The maximum possible traffic flow between a node pair in capacitated networks can be determined deterministically by polynomial-time algorithms such as Ford–Fulkerson [84] and Goldberg–Tarjan [106], the latter achieving a time complexity of $\mathcal{O}(|V|^2 \cdot |E|)$. However, this is only possible for a single node pair. If there is simultaneous traffic between more than one pair, the overall optimization problem becomes much harder and either linear programming or approximation schemes have to be used.

However, another more basic requirement can indeed be optimized. When the topology design can be reduced to a connectivity problem, then the task is equivalent to identifying a spanning tree. There are several well-known algorithms applicable, which yield a spanning tree of minimal overall edge cost. Most notable among these are Prim's [198] and Kruskal's algorithms [143] with a time complexity of $\mathcal{O}(|V| \log |V| + |E|)$ and $\mathcal{O}(\log |V| \cdot |E|)$ respectively.

For virtual and physical topologies alike, there is no single algorithm known at the time of writing which is guaranteed to solve the general design problem to optimality in polynomial time and consequently, no such algorithm is known for general multi-layer design. A classic approach may therefore use a greedy shortest path routing, e. g., routing more important traffic first, on a minimum cost spanning tree, which is iteratively augmented by heuristics such as adding least cost links first or adding new links to areas of emerging bottlenecks. Such a composite approach is highly unlikely to find an actual optimum but is very efficient and may work reasonably well for small problems. If high quality solutions for larger networks are required, other algorithmic approaches are more viable.

### 3.5.2 Linear Programming

As explained in Section 3.3.1, there are very powerful solver tools for problems that can be expressed as linear programs, which yield exact solutions even for very complex problems. In fact, there exists a plethora of works solving multi-layer problems by formulating them as ILP or MILP and using standard solvers to find solutions. While hybrid methods exist, where only the more complicated sub-problems are solved by linear programming, the method allows a complete multi-layer formulation integrating all sub-problems. While there are many benefits to such approaches, such as exact solutions and information about the duality gap, there are also a number of drawbacks: For example, the linear formulation needs to represent the multi-layer problem with sufficient accuracy, and it needs to be small enough to remain tractable.

#### *3.5.2.1 Problem Formulation*

A problem commonly solved by using linear programming is the so-called Multi-Commodity Flow (MCF) problem [11, pp. 2354–2361]. It consists of a connected graph, where each edge has a finite capacity and there are several demands, called commodities, which have a source node, a target node and require a specific capacity. The objective of this search problem is to find paths between source and target nodes for each commodity, such that there is sufficient capacity on each link traversed.

In the basic version of this problem, the link capacity is fixed, and the commodities can be split arbitrarily and can therefore be routed on several paths simultaneously, explaining the notion of a flow between source and target. Two common variations are enforcing a singular path routing, such that demands cannot be split, and using modular capacities for links, such that capacity is not fixed, but added in integer multiples of fixed quantities. For these search problems there exist common optimization problems. Most relevant to dimensioning in multi-layer networks is the so-called minimum-cost MCF problem, where each fraction of a flow

incurs a cost for using an edge and the objective is to find the solution of minimal cost. For modular capacities, the number of capacity modules typically contributes to the cost as well.

Such MCF formulations already capture the essence of a routing problem of traffic demands in a network very well and the installed capacity modules can be used to determine if candidate edges are actually used in the resulting topology or not. They can easily be extended by additional constraints and cost-values, such that extended MCF formulations [167, p. 358] are commonly used approaches to optimizing solutions to the RWA problem. This works well, since many important criteria in networking have linear dependencies, such as transmission delay being a linear function of fiber length.

However, there are limitations for the applicability of linear programming to optimization in networking. Not all constraints of common multi-layer problems are easy to include in linear programs and same may even be impossible to adapt [194, ch. 9.3.2]. For example, whenever the original problem requires non-linear functions, they have to be relaxed into a linear approximation. This may occur on the physical layer, e. g., when determining the maximum data rate for a given transparent reach of a symbol-rate variable transponder, but also on the upper layer, e. g., when considering statistics for packet delay variations in routing.

Another example are problems, for which the number of integer variables becomes too large, such that they may be relaxed into continuous variables or, in case of combinatorial choices, reduced to a much smaller number of preprocessed choices. The latter is a common approach in complex routing problems, where candidate paths may be precomputed and enumerated by an index. Such relaxations can result in situations, where the linear program may be solved to optimality, but its optimal solution might not be the optimal solution to the original problem anymore [194, p. 404].

To illustrate this point, we can examine the "SNDlib" library [183]. It consists of a variety of common, real-world network problem instances including combinations of TDR, ICR, VTD and PTD along with cost functions for network resources. The accompanying website[1] lists optimal solutions different researchers found using various formulations and solvers. Out of the 39 problems that are listed as solved without any optimality gap there are 11 for which several distinct optimal solutions exist. The objective values of solutions to the same problem instances differ by up to 11.8 %[2], although there can only exist one true optimal objective value.

#### *3.5.2.2 Complexity and Scalability*

Linear programming formulations can vary significantly in their tractability. As explained in Section 3.3.1, pure LPs are much easier to solve than ILPs. This is also true for MCF formulations, where purely linear versions can be solved in polynomial time, but if capacities are integer the MCF is NP-complete [75]. The more integer variables an extended MCF or generally a MILP formulation contains, the more demanding it can be to solve.

While this is not generally true for any problem instance, a correlation with this tendency is visible in the SNDlib data as well. Out of the 335 problems for which objective values are present, 64 have solutions with gaps smaller than 5 %. Among those 64 with small gap there are 47 solutions using continuous variables to assign traffic to links, 11 using integer fractions of the traffic and only 6 using undivided, single routes for each traffic demand. For all problems that are identical except for the treatment of traffic routing, the solutions with single routes per demand always show larger or at least equally large gaps than the other two options. Furthermore,

[1] `http://sndlib.zib.de`, last accessed 2019-04-30.

[2] cf. problem "pdh–U-U-E-N-C-A-N-N"

problems of modular capacity are also known to be quite hard to solve [194, p. 65] and within SNDlib not a single problem using modular link capacities and single routes has been solved to optimality.

A common way to reduce the complexity of the routing sub-problems is to employ a precomputed set of paths, such that the ILP or MILP formulations simply select among these. While this can significantly boost performance and even make solving previously intractable problems possible [167, p. 359], it also presents a restriction to the original problem and if the number of paths is too small, it may even exclude the optimum [194, p. 404]. Another approach to tackle large-scale problems using linear programming is to employ a divide-and-conquer approach, where sub-problems are separated and individual ILPs and MILPs are formulated, such that the solution to one problem provides the input to the next [232]. While such approaches allow for very efficient problem-solving, they may also significantly detract from the quality of the attainable solution. Furthermore, when considering multi-layer networks, often additional constraints need to be taken into account, such that the problems may become even less scalable and full featured models may only be tractable for smaller networks.

#### *3.5.2.3 Application Examples*

There is a very large variety of problems in multi-layer networks, that have been addressed with the help of linear programming. The book of Mukherjee [167] and the book of Pióro and Medhi [194] list a wide variety of example formulations for many networking problems, including multi-layer problems and sub-problems. Feller lists more than 20 different works [77, tab. 3.1] which address resource-efficiency, energy-efficiency, and reconfiguration in multi-layer problems among others. Rožić *et al.* [236] list 13 works which specifically address multi-layer problems by integer linear programming or combined heuristic and linear programming-based approaches. Ergün provides a large number of different examples and discusses in detail how different formulations can be specifically designed for multi-layer problems [289]. Due to the vast amount of works in the area, only a small number of exemplary works based on linear programming formulations will be provided in the following along with the problem size investigated.

Risso [229] combined routing and protection switching optimization for a multi-layer network in an ILP formulation. Their approach required a solving time on the order of seconds for networks of 5 nodes or less. For larger networks, depending on the exact scenario, they needed between a few seconds and several days of solver runtime and problems beyond 8 nodes turned out to be impractical to solve.

Sousa *et al.* [57] developed methods to determine a network configuration and routing which optimize the cost of the required bandwidth-variable transponders. They used a complex transponder model with associated reach and abstract cost values and considered traffic scenarios with on average several sub-rate demands per node-pair. They found that, while the basic formulations where unable to achieve optimality gaps smaller than 5 % in 2 hours runtime for a 12-node network, their extended approach including various preprocessing steps improved the scalability and allowed solving the problem for a network of 17 nodes and 26 links with gaps below 1 %. This is the largest network that has been solved to optimality for simple traffic scenarios.

Feller [77] and Zefreh [291] both used MILP formulations to optimize the resource efficiency of rather intricate multi-layer network models, where Zefreh used equipment cost and Feller energy consumption as the measure of efficiency. To make the problems tractable, Feller used path-preprocessing and Zefreh solved the entire routing problem with deterministic algorithms

before passing these results to the MILP to perform the hardware allocation. Zefreh's largest network has 17 nodes and 52 links, while Feller's approaches were viable even for the 22-node Géant-network.

While linear programming is a very powerful approach, large networks with many constraints are difficult to tackle and either require relaxations of the original problem [77, 291] or complex and highly problem-specific adaptions [57].

### 3.5.3 Heuristic Approaches

Problem-specific heuristics and metaheuristics are used in multi-layer networking, when the problem size becomes too large for linear problems, but solution quality is expected to surpass the common approaches. Especially Simulated Annealing and Genetic Algorithms are potentially a good fit, when modular capacities and singular flow routing are required [194, p. 515], but others may be applicable as well.

#### *3.5.3.1 Problem-Specific Heuristics*

There is no single problem-specific heuristic for a holistic approach to multi-layer problems, but there are several heuristics solving integral parts of multi-layer problems or act as part of multi-layer frameworks. Some examples of such approaches are presented in the following works.

Zhu *et al.* [295] have designed an auxiliary graph-based heuristic to groom traffic efficiently into circuits. Kleekamp [138] uses an algorithm to define the virtual topology by increasing its density according to given parameters and subsequently routing traffic and circuits. Martínes *et al.* [157] use a constrained shortest path first algorithm with different metrics, depending on the QoS requirements of traffic demands. Gkamas *et al.* [105] solve a very complex multi-layer problem including rate- and spectrum flexible transponders using a very advanced path search algorithm where virtual, physical, and inter-layer links are modeled separately and a combined metric assigns cost values based on the connecting NEs among other values.

While a general statement about the limits of such heuristics is impossible, it can be argued that they will typically produce good results on short time scales, but their result may be far from a global optimum. In situations, where the difference in solution quality relative to the global optimum is less important than the difference relative to the classic approaches, problem-specific heuristics may yield acceptable results faster than metaheuristic approaches. However, the drawback is that such heuristics need to be crafted for the specific problems at hand, since they are often not readily generalizable to other problems.

#### *3.5.3.2 Particle Swarm Optimizer*

While PSO approaches are primarily associated with real-valued vectors, there are a sizable number of scientific works successfully applying this approach to combinatorial problems in multi-layer networks as well, as the following examples show. Tao *et al.* [262] use a PSO to place regenerators in WDM topologies with the goal of minimizing the blocking of new traffic demands. Their particle vectors are bit strings of possible candidate positions for regenerators. Hassan *et al.* [119] use their approach to minimize blocking when solving a dynamic RWA for individual connection requests. Their PSO encodes next-hop selection priorities as particle vectors. Türk *et al.* [269] solve a complicated multi-period problem in which a multi-layer network is evolved over several years with the goal of minimizing the total cost of ownership. Their particles encode which number of NE at which layer are installed at each node for each

step in time. Although this results in a large search space, they were able to obtain significant improvements for a 17-node network using several different PSO versions.

While PSO is a very powerful tool, the adaptation to multi-layer problems with mostly integer variables or even binary choices is not trivial. This is especially relevant when considering the large effects of small-scale input changes in highly multimodal search spaces. If there is no macro-structure to the search space, areas of good optima may be skipped too quickly when the initial particle velocity is still very high. Furthermore, when very distinct solutions of similar objective value exist, the global attractor may be shifted quite often, providing ambiguous directions. Therefore, if the problem representation does not match this algorithm well, its greatest assets in terms of guiding heuristics may contribute little to solving the problem beyond a number of individual random walks.

#### *3.5.3.3 Ant Colony Optimizer*

Since ACOs work on graph representations of optimization problems, they are often a good fit for problems with inherent graph structures. A number of such approaches have been developed to address different flavors of single-layer RWA problems, such as presented, e. g., in the works of Triay *et al.* [265] and Varela *et al.* [270].

However, while multi-layer networks are also built on graph-structures, the large combinatorial design space makes a holistic representation difficult. Since representing all choices as different edges will lead to very large state graphs, which are hard to explore with limited time, most works in this area require more complex representations or update procedures. Zu *et al.* [297] seem to approach this issue by introducing a hierarchical routing structure and reflecting network link cost and available wavelength channels in the pheromone changes. Their experiments on a 15-node network show increased resource usage at slightly decreased blocking ratios compared to a regular shortest path algorithm.

Before Türk *et al.* had developed the PSO mentioned above, they had also been experimenting with solving their multi-period networking planning with ACO approaches [268]. They encode the resources installed in the lower or upper layer at each node as states in the search graph, which they prune by eliminating impossible state transitions. Furthermore, their search procedure between subsequent states is not random, but uses predefined probabilities which make drastic changes less probable. They found that their approach leads to reasonably good results compared to a problem-specific heuristic in small networks, and drastically outperforms the reference approach for a large, 67-node network.

When a meaningful and scalable representation can be found, ACO-based methods can solve complex problems very efficiently since the virtual ants lend themselves well to a parallel implementation. Furthermore, solutions can improve rapidly as all subsequent ants can immediately profit from beneficial choices of their predecessors. The drawback is that this can also lead to premature convergence which has to be considered in the choice of parameters.

#### *3.5.3.4 Simulated Annealing*

A number of works have used Simulated Annealing to tackle multi-layer problems, both concerning sub-problems as well as holistic formulations, which are especially easy to maintain due to the comparatively small amount of adaption required. Pióro and Medhi use Simulated Annealing for a number of networking problems throughout their book [194], including PTD with candidates for different node types and links [194, pp. 241–244]. Closest to multi-layer design problems, they solve an RWA with and without backup circuits [194, pp. 449–452] and

compare the results to their ILP formulation. While the ILP achieves about 25 % to 45 % lower objective values for a simple version of the problem, they found that when increasing problem complexity by adding constraints and increasing the number of choices, the gap between the results of both methods decreases and eventually the problems become intractable for the ILP approach, while Simulated Annealing still obtains meaningful results.

In a similar comparison, Mukherjee [167, p. 295] solved a single-hop traffic grooming problem using these methods and found that his Simulated Annealing outperforms his ILP approach [167, p. 299]. Schnitter *et al.* [247] use Simulated Annealing to improve the routing of traffic on the virtual topology of a global IP/MPLS-over-WDM network with the goal of minimizing the impact of failure scenarios. They found that their approach was able to further improve on the routing which had previously been optimized by a planning tool. Späth *et al.* [255] promote Simulated Annealing for use in multi-layer planning tools and illustrate the efficacy of their approach including VTD, TDR, RWA and PTD for a number of example networks, where the approach is always either equally as effective as their reference method or even better. Sadly, their paper does not detail much about their approach, nor about the test cases. Kucharzak *et al.* [144] solved a multi-layer dimensioning problem including VTD and TDR using several heuristics, including Simulated Annealing. Interestingly, in their evaluation of a 15-node network with scenarios of increasing traffic, they found that their approach only performs better than the simple heuristics based on Dijkstra's algorithm when the traffic demands were low compared to the link capacity. It should be noted that their temperature schedule has a rather steep slope, and the approach terminates after only a few hundred iterations.

Finally, Feller [77] devised a Simulated Annealing- and a MILP approach to solve a complex multi-layer network reconfiguration problem. Therein, the network has to be reconfigured every 15 min to suit time-varying traffic demands, such that the power consumption of installed NEs is minimized. The reconfiguration requires solving VTD, TDR and ICR considering a complex resource hierarchy and the dependencies to the previous configuration. The evaluation, which has been performed mainly for the network topologies "Abilene" and "Géant" with scaled versions of the traffic demands available in SNDlib [183], found that given sufficient time, the MILP approach will result in lower objective values. However, the linear programming approach did not scale arbitrarily and the 22-node "Géant" topology already showed instances, where a first primal solution could only be obtained with a runtime of 1 h [77, p. 119]. While the results of the Simulated Annealing approach were found to be consistently worse, it was only by a comparatively small margin and both approaches outperformed the baseline algorithm in almost all cases.

Since Simulated Annealing requires only a suitable perturbation function as problem adaption, it can easily be used to represent any aspect of a multi-layer network configuration. As it follows a trajectory along small-scale changes, it can explore multi-modal search spaces, but this makes it also more difficult to explore large search spaces without macro-structure. In multi-layer networks the number of neighboring solutions increases exponentially with the number of nodes and links, such that it may take a long sequence of iterations to cover an area of the search space. This effect cannot be efficiently counteracted by parallel implementations, since the single-trajectory approach always requires information of the previous state to determine the subsequent state.

#### *3.5.3.5 Other Metaheuristic Approaches*

A couple of other meta-heuristic approaches have been applied to multi-layer networking as well, but an exhaustive enumeration of all such works and approaches would be well beyond the scope

of this monograph, such that only a few examples are named in the following. Pióro and Medhi solve several networking problems using a metaheuristic called Simulated Allocation [194, pp. 173–176]. It is similar to Simulated Annealing but uses a state-dependent acceptance probability. They also suggest Tabu Search [194, pp. 176–177] as a viable method, which is also used by many others, such as Ding *et al.* [61], who optimize CAPEX in multi-layer networks, and Yao *et al.* [279], who employ Tabu Search to optimize traffic grooming, or Zhang *et al.* [292], who have designed a multi-step approach that combines local-first and best-fit heuristics with Tabu Search to optimize traffic grooming in reconfigurable multi-layer networks.

Valesco *et al.* [274] used Greedy Randomized Adaptive Search Procedure (GRASP) to optimize CAPEX and OPEX of multi-layer networks. Holler *et al.* [126] minimize the cost of NEs in the WDM-layer and found that their approach obtains results for networks of more than 100 nodes. Pedrola *et al.* [188] augment the regular GRASP approach by a path-relinking technique which results in an algorithm able to outperform even their genetic algorithm implementation.

Various other approaches such as Harmony Search, Bee Colony Optimization and hybridizations of algorithms have been used to varying degrees of success.

## 3.6 Overview on Genetic Algorithm-based Network Optimization

Evolutionary and Genetic Algorithms have been used to solve almost all problems commonly found in multi-layer networking. They share the parallel exploration and global information sharing with ACO and PSO methods, but their operators are more flexible. They may be used akin to Simulated Annealing, operating on small changes, but may also incorporate problem-specific knowledge. While the advantages are compelling, Genetic Algorithms also require a meaningful multi-layer problem representation just as PSOs and ACOs do, which may drive or hamper performance. Similar to these metaheuristics, GAs also depend on a large number of adjustable hyperparameters. This section will provide a short overview on methods for sub-problems, as well as integrated solution approaches for entire multi-layer networks and their methods of adapting the GA approach.

### 3.6.1 Research on Topology Optimization

Topology design problems like PTD and VTD are found in many domains of networking and consequently a large body of works focused on finding optimal topologies exists. One way of categorizing them is according to their intended level of connectivity. For many applications, basic connectivity is sufficient, such that the goal is to find tree topologies, which are cost-minimized, and which possibly need to fulfill additional constraints such as staying below link capacity or node degree limits. This type of topology is often required in dimensioning PTD problems, where the cost of establishing new links dominates most other costs. When virtual topologies or QoS requirements are considered, sparse topologies may be insufficient to fulfill constraints, such that there is no upper limit on graph connectivity. Nesmachnov *et al.* [173] list more than 40 scientific works utilizing some form of evolutionary formulation to find tree topologies for different kinds of communication networks. Many of the works in this area use one of the following approaches to represent trees as chromosomes to encode the topology.

Direct edge encodings [15, 55, 192, 193, 250] use binary vectors to represent the presence of possible candidate edges in the tree. The advantage of this approach lies in its simplicity since it yields fixed-size chromosomes of length $|E_c|$ and allows using most common mutation and recombination operations designed for basic GAs with good heritability, which is the property indicating how well traits can be transferred to the following generation of solutions. The

primary problem for trees lies in the fact that a chromosome instance can reflect any topological structure and valid trees are only a relatively small subset of all possible encodings. While penalty functions can be used to guide the algorithm towards valid trees, research has shown that *repair functions* lead to better results [203]. Repair functions are typically run after the genetic operators and alter the offspring chromosomes in such a way that they become valid trees. The drawback of these functions is that they may reduce heritability and increase the computational effort.

Rather than using the large number of edges to directly encode a topology, node sequence-based encodings represent trees by identifying a sequence of nodes that need to be decoded into the proper tree structure. An example for this are encodings based on Prüfer sequences [23, pp. 33–35], which can be found in many works [34, 37, 60, 102, 261, 294]. They represent spanning trees in an elegant and compact way as fixed-length integer sequences. The big advantage of using Prüfer sequences is that they always encode a spanning tree, such that there are no invalid topologies. However, there are two drawbacks. On the one hand, they require a separate decoding algorithm with a runtime of $\mathcal{O}(|V|\log|V|)$ on graphs with $|V|$ nodes and, on the other hand, they exhibit poor locality and heritability [109]. Later works proposed alternatives to Prüfer sequences like the determinant encoding [1, 37], dandelion encoding [187], or network random keys [234]. While these all improve on locality and heritability, they typically require decoding functions of increased runtime complexity [203], often scaling with the number of potential links, rather than nodes.

Weight-based approaches like link-bias [139, 204, 254] or Link and Node Bias (LNB) [185] approaches introduce additional weighting coefficients which modify the original cost values of links or nodes or both. The resulting weighted graphs are then used to run an algorithm such as Prim's or Kruskal's, which identify the minimum spanning trees given the modified weights. Often, these weights are not initialized randomly, but already biased towards graph elements of lower cost or otherwise beneficial traits. The weights may be expressed as vectors of integer- or real-valued numbers which form the chromosome. Depending on what is biased, chromosomes may therefore consist of weight vectors for links, nodes, or both. The largest disadvantages of such approaches lie in the combinatorial complexity of the vectors and the decoding effort given Kruskal's time complexity of $\mathcal{O}(|E|\log|V|)$.

Finally, there is also an approach, which omits an abstract genetic encoding altogether[1] and works directly on sets of edges. These edge set approaches [118, 203] shift the complexity from the encoding to the operators. Mutation and recombination operators are therefore specifically built to operate on sets of edges. While this may increase the computational complexity of operators significantly, the complexity of creating offspring using this system may yet be smaller than the combination of having a complex decoding and potentially repair functions in addition to simple genetic operators. The advantage is that this combination of custom operators and omitting the decoding step can boost performance, but the operators may be fairly difficult to adapt to more complex problems without excessive computational effort.

When topologies with a higher level of connectivity are required, attention in most works shifts from intricate encodings with simple operators to simple encodings with intricate operators. The reason for this is that when any arbitrary topology is a possible solution, a direct edge encoding offers the least amount of combinatorial complexity with high locality and heritability. The problem in communication networks is that unconnected topologies are undesirable and

[1] While it can be argued that such EAs cannot be considered as GAs per se, lacking their defining feature, the presented approaches are otherwise identical to GAs and will therefore be considered as variants thereof within the context of this work.

need to be avoided. As mentioned before, this can be accomplished by adding penalties to the objective function, when connectivity requirements are not met, but this may increase the number of iterations the Genetic Algorithm needs to converge to a good solution.

Most works use repair functions or incorporate them within the operators in order to deterministically enforce connectivity [140, 141, 163, 246, 285]. These repair functions however may vary drastically in their runtime, depending on the exact connectivity requirement, e. g., if simple reachability needs to be established or a number of alternative links is needed. Especially for more complex requirements, this may necessitate an algorithmic examination of the decoded topology, such that other works also use edge sets or immediately work on the entire data structure [36, 122, 191, 252], since the added complexity of their customized operators is comparable to operators with added repair functions.

While this overview covers the most common approaches, there are also many works dealing with more specific topology constraints and designs, such as networks of pre-defined hierarchies or geography-specific constraints which may employ more specific encodings. The interested reader is referred to the survey of Kampstra *et al.* [134], which lists 55 works solving general topology problems and an additional 14 works solving topology problems with constraints from different types of optical networks using various evolutionary computing methods, including GAs.

### 3.6.2 Research on Routing Optimization

Since there are many efficient algorithms for point-to-point shortest path routing, typical applications of genetic algorithms in this area mostly focus on specific sub-types of routing. These include, e. g., identifying efficient multi-cast trees, considering technology-specific constraints or multiple link metrics, or dealing with variability on graph properties. The works cited in the next paragraphs therefore tackle a variety of different routing problems and the common factor highlighted is the form of genetic routing representation. Similar to the case of topologies, there are also a number of encoding approaches that are used by the majority of works in some, possibly adapted or hybridized form.

The direct encoding of candidate edges as binary vectors is a simple and straightforward approach, that is used in a number of works [5, 100, 278, 283]. Most of these works optimize single paths that are subject to complex QoS factors, such that they cannot be aggregated into a single monotonously growing metric, thereby precluding the usage of deterministic shortest path algorithms. Since this encoding offers a large number of codes that correspond to infeasible solutions due to the fact, that selected edges may not form a contiguous path or contain loops, repair functions and operators are required. While this scales reasonably well for single paths in sparse networks [5], the encoding method suffers from the rapid increase of combinatorial complexity in network-wide approaches, where all paths have to be optimized simultaneously. Therefore, these approaches are mostly relegated to small graphs or single-route problems.

Node sequence encodings are a popular choice for routing problems [3, 29, 168, 290]. They reduce the combinatorial complexity of representing all edges to the number of nodes in the network, thereby improving the scalability for problems with high graph density. If the graph is not a full graph, however, they still require some form of repair function to fix situations in which two subsequent nodes are not connected by an edge. A big advantage of this method is that routing loops are easy to detect, even without decoding the graph structure, simply by checking the node sequence for repetitions. This can even be entirely avoided by operators, which treat the node sequences as permutations of node identifiers and maintain this property. The node sequence may be of variable length and only contain the nodes of the path, which

requires problem-specific operators. Fixed-length encodings, on the other hand, may use more simple operators and move runtime complexity to the decoding procedure. If the code is based on permutations, this can be trivially accomplished by skipping the parts of the sequence before the source node and after the destination node.

Precomputed path enumeration encodings follow the notions of precomputed candidate paths also used in ILP formulations. Rather than determining the paths as part of the problem, they are computed by a deterministic algorithm in a preprocessing step. This is a rather popular approach used in many works [10, 108, 117, 164], especially when paths are subject to many constraints or require a large number of links. Since the static path-specific constraints, such as length- or hop-limits, can be evaluated in the preprocessing phase, the additional complexity introduced by them can be removed from the optimization problem itself. Furthermore, this encoding scales to very large networks, since for a constant number of precomputed paths, primarily the preprocessing phase scales up. While this approach does not require any repair function since all codes lead to valid routing configurations, the codes cannot reflect all possible configurations.

Sets of routing weights can also be used to encode a routing configuration in combination with a regular routing algorithm [74, 86, 186, 227, 245]. The encoded values may either directly represent the link weights, or they may also correspond to bias terms which modify a preexisting link weight system. This approach is often used with stochastic traffic models of sequentially arriving and departing traffic demands. Since the routing algorithms, which are typically used, only route one traffic demand at a time, the order in which the routes are determined may alter the resulting routing configuration of the network. This needs to be specifically addressed when using traffic models representing simultaneous flows of traffic between the nodes by establishing a total order relation on the traffic demands. When this order is assigned statically, it may lead to an inability of the coding to express all possible routing configurations. When the ordering of demands is to be performed dynamically, it will be required to encode their order along with the routing weights, which increases the number of combinatorial choices. Furthermore, a QoS-diverse routing would require using different sets of routing weights for different QoS classes or even for each demand, which will drastically impact the combinatorial complexity.

Finally, the approach of skipping the genetic encoding altogether and use custom operators is also used in a smaller number of works [152, 251]. As with the topology problems before, there is a reduction in runtime due to the removal of the decoding step, but the design of fitting operators is more complex. While crossover can be achieved with good heritability, e. g., by exchanging routes between the same source and destination between parents, mutation is often more difficult. It should allow for any possible route, but not drift towards excessively long or convoluted routes. Obtaining a good balance between exploration and refinement is essential, but difficult without analyzing the effects of the decision beforehand by using additional heuristics or restricting the possible solutions representable by the approach.

Similar to the case of topology design, there is a wide variety of works dealing with a plethora of more specific constraints and properties. The survey of Kampstra *et al.* [134] lists 57 works using evolutionary computing techniques to solve general routing, restoration and admission problems and an additional 20 covering routing in optical networks. It is interesting to note, that a number of works use GAs to complement path-selection-based ILP approaches for problem instances where the solver algorithms approach the limits of scalability. For example, de Miguel *et al.* [164], who dimension a WDM network for maximum resource efficiency, found that for their problem, their GA would sporadically outperform their ILP for networks with more than 10 nodes, which does not yield results for networks with more than 14 nodes. Pióro

and Medhi also used a GA-variant[1] and compared its performance to ILP formulations for concave network dimensioning [194, pp. 202–203] and for multi-hour network design with two service classes [194, pp. 467–471]. They found that for their concave dimensioning problem, where the ILP needs to approximate a non-linear function, the GA offers better results, albeit at significantly higher runtimes. For their multi-hour design problem, they used networks between 7 and 50 nodes and found that a solver using a regular MILP formulation could only solve the 7 node problem, while their GA provided solutions to all of them.

### 3.6.3 Research on Multi-Layer Network Optimization

As explained in Chapter 2, a configuration of a multi-layer network, given its (potential) physical topology, typically requires finding solutions for TDR, VTD, ICR, WA, and potentially PTD. Since each of these problems is at least NP-complete, the overall combination of all of them is also at least NP-complete and exhibits a very high level of combinatorial complexity. Most works dealing with such networks therefore focus on its constituent subproblems in one of two ways. They either optimize one of these aspects and deterministically solve the others based on the optimization results or they optimize each of these problems individually using the results of the previous subproblem as input to the following ones. A joint optimization approach, which simultaneously considers all problem components, will most likely be intractable for real-world network sizes and constraints [194, chap. 12.7.4].

The 2016 survey of Rožić *et al.* [236] categorizes a total of 38 scientific works dealing with multi-layer network optimization, emphasizing, that the majority of these use either ILP-based approaches or deterministic heuristics. Only 6 of these employ some form of metaheuristic and among those only two, the works of Morais *et al.* [166] and Ruiz *et al.* [242], make use of GAs. In addition to these two, a number of other works have since been published that address various multi-layer problems using solution approaches that are either centered around or incorporate some form of GA or EA. The following provides a short overview regarding such related works.

Morais *et al.* [166] present a complex multi-layer optimization environment in which they separately optimize several sub-problems. They use a GA to solve the PTD [165] and based on its results, they use an ILP to dimension node resources with a high level of detail down to individual interfaces. After this, they use another ILP to reoptimize the traffic routing to determine, when capacity extensions are required over the network's lifetime. Their overall objective is to obtain minimal CAPEX and OPEX in terms of power consumption, as well as physical footprint. The GA uses a bit-string encoding of candidate links. Selection is performed either by RWS or tournament, while crossover is performed either by UXO or CNCXO operators.

In their evaluations, they found the UXO operator to be consistently more efficient than CNCXO and that RWS offered a smaller, but noticeable advantage over tournament selection. The most interesting aspect about this work is the population generation. They use a modified Waxman model [281] to create the initial population and compare this approach to a population of randomly generated individuals. The figures of their evaluation on 9 different topologies of 9 to 17 nodes, suggest that this initialization sometimes slightly decreased performance for the small networks, but lead to significant improvements for the largest networks. After 100 generations, the gap to the optimal solution was 10 % or less for all instances.

Ruiz *et al.* [242] have developed a BRKGA to minimize CAPEX in terms of ports and different router models while considering different failure scenarios. Both topologies, physical and virtual, are fixed inputs to the routing of traffic and circuits, considering wavelength slots

[1] They refer to it as an EA [194, pp. 172–173], but their approach is consistent with a GA using ($\mu$+$\lambda$) selection.

on the fibers. BRKGA uses encodings based on floating point numbers in $[0,1)$ and randomly generates such numbers as keys for gene values. It inherently uses elitism with a large number of individuals and performs a biased recombination, such that one parent is always selected from the elite population and the other from the regular population. Mutation is performed by removing a number of individuals from the population and replacing them by randomly generated individuals.

Their encoding is an LNB approach with additional genes which provide the sequence in which traffic demands are to be routed. The demands are ordered according to this index, the bias-terms are applied to link and node metrics and a shortest path routing is performed using the resulting values. The population is split such that the elite part represents 20 % of individuals and the lowest 20 % of the population are replaced in the mutation phase. They evaluated their approach in networks of 20 and 21 nodes and found that during the 10 hours of runtime, 80 % of the overall improvement on the initial solution was obtained within 200 minutes.

Balasubramanian *et al.* [9] present a highly integrated GA approach to minimize cost in IP/MPLS over WDM networks, which is in fact very similar to the objective of this monograph. Their approach is scalable to large networks exceeding 100 nodes and considers a very detailed hardware model considering IP, optical and regenerator ports as well as flexible optics. They focus on different protection and restoration schemes and employ additional evaluations to simulate failure scenarios and grade their solutions' survivability. While the resource cost is the primary metric, they also use penalty terms, e. g., for failing to provide traffic demands with sufficient QoS. Their GA is augmented by several other algorithms, e. g., for dimensioning of regenerators, but the work does not contain details, especially not on how exactly WA is solved. It seems that TDR is generally addressed by regular shortest path algorithms using fiber length as the metric.

The chromosome is divided into two parts, one for VTD and one for ICR. The virtual topology is defined by a bit-string that represents the possible virtual links. The second part defines, how these links are routed on the physical network and is referred to as a "tunnel path directory", which may suggest some form of path precomputation, but no further information is given. While recombination is done by a CNCXO operator, information regarding the mutation operators is very sparse. Some form of custom operators are used which analyze the parents and perform mutations targeted towards improving utilization, protection and delay deficiencies. The description places special emphasis on avoiding premature convergence through population management. This is performed by a number of techniques including special initialization algorithms, injecting new candidates to increase diversity and evolving several independent populations in parallel, which regularly exchange candidates using special operators. Their evaluations show cost savings of 25 % and above, compared to their reference method.

Banerjee *et al.* [10] and Ghose *et al.* [103] developed GA approaches to solve different capacitated networking problems. While both works consider delay in the form of propagation and queuing delay, the first uses a multi-objective approach, combining minimization of delay and resources, while the latter is a single-objective approach focused on delay alone. Regardless, both works seem to use the same underlying encoding for the GA. They use a path enumeration approach, where a fixed number $k$ of shortest paths are precomputed between all node pairs. The integer index of the path is expressed as a bit-string, such that $\lceil \log k \rceil$ bits are needed for every pair of nodes. The GA is then used in two phases. First, ICR is solved, such that the genes indicate a set of circuits, with the objective of maximizing the number of connected node pairs without violating a wavelength limit. Following this, VTD is performed by adding edges for each routed circuit and adding circuits on all links, that correspond to physical-layer links. Since

the physical topology is defined as connected, the resulting virtual topology will be connected as well. Based on this topology, the second GA is run in order to solve TDR, and a deterministic heuristic is used for WA, such that now the overall optimization goal can be evaluated.

Both works use a specific NPXO-version to create their offspring chromosomes. They only take full sequences of $\lceil \log k \rceil$ bits corresponding to a node pair, such that the path index value is preserved. For each pair they seem to randomly choose, which parent contributes its bit-sequence. The mutation operator is RRM and is only applied to the worst chromosome in the gene pool, rather than the offspring. While all GA approaches use a rank-based RWS to determine the parents, the single-objective version uses a linear ranking, while the other applies a more complicated algorithm to account for both objectives. Finally, in survivor selection they discard the worst individuals in the same number as new ones have been created to maintain a steady population size.

Risso *et al.* [228, 230] use several different EA approaches to optimize IP/MPLS over WDM networks for resource efficiency while including backup paths in traffic routing. They use a GA, a parallel EA with distributed subpopulations and a hybrid EA. Their chromosomes encode the capacity of logical links, the circuits and the traffic routes for active and backup paths. They are split into two parts. One part addresses TDR by defining a primary and a backup route for each traffic demand. The routes are given as individual node sequences. From this information, the links required in the virtual topology and their capacity can be determined deterministically. The other part of the chromosome encodes the paths for the circuits of the physical layer, such that for each node pair there is one path. These paths are also encoded as node sequences.

The solution method includes a greedy deterministic algorithm that is used to initialize individuals for the population, but also to repair infeasible chromosomes generated by evolutionary processes. It iteratively builds the routing genes and selects the next node in the sequence relative to the connecting link's cost and builds the physical layer genes accordingly. The evolutionary operators are mostly problem-specific. The recombination operator creates a single offspring by selecting routing genes individually from randomly selected parents and copies the required physical-layer genes as well. If no parent can be selected for a routing gene such that the primary and backup path are still disjoint, then the above repair algorithm is invoked.

The approach uses 5 different mutation operators which can be combined with different probabilities. One changes both parts of a chromosome by randomly removing a virtual link from all genes and rebuilds the chromosome using the repair algorithm. The remaining can be split into two pairs. One pair primarily affects the traffic genes and the other pair primarily affects the connection part of the chromosome. All four randomly choose a gene within their respective chromosome parts and rebuilt it. The first operator of each pair uses an unspecified randomized greedy process to do so. The second operator repeats the process a fixed number of times and selects the best solution candidate among the resulting candidates. Risso *et al.* show successful application examples for networks between 50 and 70 nodes, but due to the large combinatorial complexity of the encoding, their single-threaded approach shows runtimes around 100 h. Their parallel implementation consistently improves the average objective value and reduces the runtime to between 30 and 40 h.

Saha *et al.* [244] present a rather holistic and detailed solution method to maximize the throughput of capacitated packet-over-WDM networks. Given the physical topology and the available hardware resources, they use a GA for VTD and different deterministic heuristics to perform ICR, WA, and TDR. In a second run, they consider minimizing the overall delay as the optimization goal. Delay consists of an upper-layer delay per node as well as the propagation delay on the fiber. Their chromosomes contain a gene for each transponder at every node. The

genes have integer values that represent a unique identifier for the target node. In this way, a chromosome not only reflects the connectivity matrix of the virtual topology, but also the capacity between connected nodes in terms of the number of circuits. To establish the initial population, they use a randomly generated Prüfer sequence to create a spanning tree, such that all solution candidates are guaranteed to yield connected topologies. When constructing the spanning trees, they also ensure that the tree does not require nodal degrees for which the nodes have insufficient transceivers. Their GA uses RWS as the parent selection mechanism and CNCXO and RRM as evolutionary operators. They validate offspring candidates and apply a repair function to ensure feasible solutions whenever possible. Survivor selection is performed by sigma truncation, which eliminates solution candidates worse than a certain number of sigmas from the average fitness in the population. Furthermore, they apply the elitism scheme by always migrating the best two solutions to the next population. Their approach is tested on a rather small network with only few transponders but outperforms their reference approaches.

Ahmad *et al.* [2] and the extension towards reconfiguration by Bonetto *et al.* [24] aim to minimize the power consumption of transport networks. They refer to the networks as WR (Wavelength Routing or Wavelength Routed) networks, stating that this is functionally equivalent to WDM networks. In the first work they consider the power consumption of optical transceivers and the upper-layer switching subsystem explicitly, while the second explicitly models line cards, shelves, and the switching system between them, based on IP routers. The GA within their solution methods solves the VTD problem including dimensioning in terms of the number of circuits needed between the nodes, from which the required hardware resources are inferred. Neither work details, how exactly traffic demands and circuits are routed.

The structure of the chromosome in the GAs is as follows. Each locus corresponds to a source and destination node pair and the value of the gene then represents the number of active circuits between them. Recombination is performed by the CNCXO operator and mutation is performed uniformly on all genes, most likely by RRM, although this is not specified in detail. When a solution candidate is created from the chromosome it is checked for feasibility and discarded if not all traffic can be routed. In the first work, they compare the GA approach to the best results from a MILP formulation with a solver runtime limit of 24 h, and a custom heuristic. They determined that the GA is almost always better than the heuristic and often finds the optimal results. Interestingly, their GA seems to struggle in low-load scenarios, where the optimal topology is very sparse. In the second work, they suggest a new heuristic which yields mostly better results than their GA at drastically lower runtimes.

Roy and Naskar [235] propose a GA-based approach to minimize the number of Synchronous Optical NETworking (SONET) devices in a SONET-over-WDM network. Their chromosome structure encodes the sequence of the traffic demands, which are all of identical granularity such that they refer to them as "calls". They use a framework of deterministic algorithms which allocates resources and tries to route the calls on different shortest paths depending on the already available resources and following the sequence established by the chromosome. They initialize 5 % of the population with heuristically generated chromosomes to provide a good starting point without having a negative impact on genetic diversity. Parents are selected randomly, while survivors replace previous individuals with a certain predefined probability. As evolutionary operators[1] they only use CNCXO augmented by a repair function to prevent repetition or omission of traffic demands in the sequence.

[1]Their work uses different terms, as they seem to refer to survivor selection as crossover and to crossover as mutation.

Alshaer and Elmirghani [4] suggest a solution approach for QoS traffic in IP over WDM networks, adapting a differentiated services architecture to include resilience factors. Their traffic model includes three priority classes and each demand requires a certain number of time slot-based channels as well as a delay limit. They model heterogeneous nodes of different switching granularities and use a deterministic algorithm to precompute routing tables of QoS-compliant multi-layer paths. The genes of their GA representation then provide an index into this routing table. Rather than working on integer numbers, they represent each gene by a fixed number of bits representing the integer value.

They seem to employ a URM and most likely a 2-point crossover operator, while selection is performed by RWS using elitism to preserve the most fit individual. They use another deterministic heuristic to create good individuals for their initial population. In their evaluation they used a topology of 20 nodes, but only 4 pairs of nodes have traffic demands between them. They initialize half of the population with their heuristic and the other half is chosen at random. They find that their GA and yet another deterministic heuristic, which they suggest in the same work, outperform the reference methods.

Durán Barros *et al.* [65] suggest a GA to solve a capacitated VTD with the objective of minimizing end-to-end delay. The delay is modeled as a combination of propagation delay and a variable delay for packet processing in the upper layer according to a Jackson Network queuing system. While they do not refer to their approach as applying to a multi-layer network, the description of their "multi-hop scenario" bears all relevant properties to classify it as such. It uses a sequence encoding of source–destination pairs, such that the chromosome consists of $|V| \cdot (|V| - 1)$ genes for a network with $|V|$ nodes. A number of shortest paths are precalculated for each pair and the chromosome defines the sequence in which circuits are routed on these paths. If a circuit cannot be routed due to resource starvation, the respective gene is omitted. If the resulting topology is not connected, a repair function is used that first establishes circuits forming a Hamiltonian cycle and then continues adding circuits according to the chromosome. After the VTD is found, WA is done by a first-fit heuristic and TDR is determined by a shortest path algorithm using the number of hops as the distance metric.

Interestingly, they seem not to employ permutation-preserving operators. Parents are selected by RWS and, depending on a probability, either copied without changes to the offspring pool or by performing a CNCXO operation. The genes in the chromosomes are then uniformly subject to mutation which interchanges the gene for another, randomly chosen gene, which is not explained further. The population is initialized to have a single individual which is the result of a deterministic heuristic. In their evaluation they use very small populations of 2 to 6 individuals, creating 6 to 14 offspring per iteration. They suggest to terminate the algorithm after 10 000 iterations.

The methods used in the works outlined above are summarized in table Table 3.1. Most works deal with resource optimization, and some even include delay as a secondary optimization goal or constraint. However, to the best of our knowledge, there is no solution method incorporating GAs that addresses QoS-diverse traffic in terms of delay and explicit availability figures in multi-layer networks, such as will be presented in the next chapter. The works of Balasubramanian *et al.* and Risso *et al.* are closest in terms of explicitly considering QoS and being intended to scale to large, real-world networks. In contrast to this work, their primary focus is on survivable networks, rather than dealing with diverse traffic demands, which provides an even more complex challenge, especially in terms of routing scalability.

| Src. | Problem(s) | Scenario | Objective | Coding | Population | Selection | Recomb. | Mutation |
|---|---|---|---|---|---|---|---|---|
| [228] [230] | TDR+ICR | IP/MPLS over Optical | CAPEX (Circuit Lengths) | Routes as Node Sequence | Heuristic Init., Distributed | RWS* | Undefined domain-specific Operators | |
| [242] | TDR+ICR | IP/MPLS over WSON | CAPEX (Nodes, Ports, Fiber Lengths) | LNB + Demand Sequence | Undefined Init., Elite Subset | Random, 1 Elite + 1 Regular | BUXO | Injection of random individuals |
| [244] | VTD | Packet over WDM | Throughput | Transponder to node mapping | Prüfer-based Init., Elitism | RWS with Sigma Truncation | CNCXO | RRM |
| [2] [24] | VTD | WR Networks | Power Consumption | Capacity between Node Pairs | Random Init. | Undefined | CNCXO | URM* |
| [235] | ICR | SONET over WDM | CAPEX (Add–Drop Multiplexer) | Demand Sequence | Heuristic Init. | Random | CNCXO | None |
| [4] | RWA | IP over WDM | Throughput | Path index as Bit-String | Random and Heuristic Init., Elitism | RWS | NPXO | URM |
| [65] | VTD | IP over WDM | Delay | Node Pair Sequence | Connected-Graphs Init. | RWS | CNCXO | Copy, RRM* Randomization |
| [166] | PTD | Packet over Optical | CAPEX (TXPs, OLAs, Fiber Lengths) | Links as Bit-String | Waxman-based Init. | RWS, Tournament | CNCXO | URM |
| [9] | VTD+ICR | IP/MPLS over DWDM | CAPEX (IP Ports, Optical Ports, Fibers) | Connectivity as Bit-String + Route Dictionary | Undefined Seeding Init., Distributed | Undefined Custom Approach | CNCXO + Island migration | System of 5 custom mutation operators |
| [10] [103] | TDR+ICR | Packet over WDM | Wavelengths, Delay | Path Index as Bit-String | Random valid paths init. | RWS | Custom NPXO | RRM |

Table 3.1: Scientific works on multi-layer network optimization utilizing solution methods based on Genetic Algorithms

*The description in the respective works is incomplete, but what is given suggests to correspond to the approach as stated here.

## Chapter Summary

In this chapter, we have presented a short introduction to the formal aspects of optimization. It has been shown, that constrained combinatorial optimization problems are especially difficult to solve due to the absence of gradients and due to their complex search space structure. Furthermore, we have established that typical multi-layer optimization problems fall into this category. We have explored their properties in terms of combinatorial complexity, tractability, and modeling approaches.

In the following sections, an introduction to common optimization methods, such as Integer Linear Programs (ILPs) or hill climbing, has been given. The concept of a metaheuristic as a largely domain-neutral solution framework has been explained and the most relevant approaches from this category, both trajectory- and population-based, have been outlined. Special emphasis has been placed on Simulated Annealing, as it is used as part of a reference solution method in this monograph.

Genetic Algorithms (GAs) have been introduced along with the relevant biological terms. We have explored the underlying theoretical constructs and presented the most common variants in terms of the evolutionary operators regarding mutation, recombination, and selection. Among others, this has included N-Point Mutation (NPM), Uniform Random Mutation (URM), Uniform Crossover (UXO), Cut-and-Crossfill Crossover (CNCXO) and as selectors Roulette Wheel Selection (RWS) and its extension Stochastic Universal Sampling (SUS).

The last parts of this chapter have elaborated on different solution approaches to optimization problems common in multi-layer networking. The penultimate section was focused on both, legacy and scientific approaches based on the methods outlined before, and we have discussed aspects pertaining to their scalability and complexity. While many researchers have tackled different objectives in networking using GAs, the works that explicitly address multi-layer network optimization with QoS are relatively few.

The chapter has concluded with a detailed overview on scientific works featuring such integrated solution methods. They have been investigated and categorized regarding their exact use and their approaches to adapting the GA framework. To the best of the author's knowledge, no prior work exists, which optimizes multi-layer network dimensioning and configuration regarding latency and availability utilizing GA-based solution methods. This identifies the gap that is filled by the approaches suggested within this monograph.

# 4 Genetic Algorithm Approach

In the previous chapter, combinatorial optimization has been introduced and basics of Genetic Algorithms (GAs) and related approaches have been presented. While we have stated that the mechanics of GAs are a good fit for combinatorial optimization problems such as they occur in a multi-layer network context, we have also identified a lack of works focusing on different QoS metrics by using more advanced adaptions and efficient encodings. This chapter will present four different encodings together with specific operator adaptions and enhancements, as well as several general-purpose mutation and recombination operators, which are then integrated into an extensible solution method framework.

## 4.1 Requirements and Design Space Delimination

The goal of the developed approaches is to enable optimizing large multi-layer networks in the presence of QoS traffic. These are especially challenging conditions due to the complex interworkings of the individual layers with many degrees of freedom. Rather than finding a chromosome able to encode all relevant parameters separately and run a basic GA version, domain-specific knowledge should be incorporated into the encoding and the surrounding genetic operators. The resulting approach shall be easily extensible to include additional conditions and network models for ready application to related problems.

### 4.1.1 Network Abstraction

The basic network model is the dual-layer architecture as explained in Section 2.1.2. Summarizing its key properties in short, every vertex of the graph is considered to be a PoP with the same basic composition, always consisting of both, upper-layer and lower-layer resources. We can identify each PoP by an ID, which, without loss of generality, we consider to be a unique positive number. A virtual topology can be defined by connections on the lower layer and traffic demands have to be routed on the virtual links of the upper layer. Connections and traffic can be switched between links at each PoP without restrictions, as long as sufficient resources are present at the node. For traffic, this requires line cards and ports, while for circuits, sufficient spectrum on connecting fibers is required. The infrastructure of node locations and candidate edges for physical links are always fixed and form a connected graph. Traversing links and nodes incurs delay and links have known and static availability figures. Furthermore, we make a number of additional, more specific assumptions regarding control, hardware, and physical parameters.

#### *4.1.1.1 Control Instance*

The network is considered to be under the control of a central instance with global knowledge on all aspects of the network. This may be an abstract planning tool for green-field deployments

creating a data sheet of how to build the network, but it may just as well be an SDN controller changing the network's configuration during operation according to some external trigger. Interactions between the actual NEs and this control entity are not part of the optimization, i. e., the signaling part of the network is out of scope.

#### *4.1.1.2 Hardware Model*

For both layers, the basic components such as chassis and switching subsystems are always present, regardless of the amount of traffic that is required, since any PoP will need at least the capacity to maintain connectivity. We consider upper-layer NEs to consist of a chassis containing a number of core-facing line cards with a number of ports. Tributary line cards are not explicitly considered since they offer no room for (re)configuration and need to be dimensioned for the expected peak in client traffic without any potential for optimization. The switching fabric in the upper layer poses no restriction on the amount of traffic that can be switched between the upper-layer ports.

These ports are connected to the lower-layer NEs, such that for every router port there is a corresponding client port in the WCS. Since this is a fixed 1:1 mapping, it also extends to potential TXPs. Under this assumption, there is no difference between the case of router ports with DWDM interfaces to router ports that connect to the WCS via external TXPs from a VTD or ICR perspective. The WCS in each node is considered to be monolithic and features connections to all physical fibers which lead to the neighboring PoPs and there is no internal blocking or wavelength contention. The exact internal composition of the WCS is out of scope and any port of the upper layer may be forwarded to any of the long-haul fibers. The WCS and potential TXPs are treated as being independent from potential shelves or other enclosures since these have little to no influence on the metrics in question.

Furthermore, there is no explicit representation of OLAs, regenerators or other amplifiers. OLAs and amplifiers scale with fiber lengths and fiber connectivity, which are fixed in the infrastructure. Therefore, they can be considered as static costs for the dimensioning case or are already installed and part of the fixed infrastructure for the dynamic case. While 3R regeneration could be included, its effects would require using a more detailed level of intra-node hardware dependencies. While this is possible using the algorithms presented in this chapter by including a local routing heuristic, we shall omit their inclusion here, to focus on the effects of the genetic algorithm design and implementation.

Given these assumptions, the resources that actually vary significantly in number depending on the network configuration are the line cards on the upper layer and the transponders on the lower layer. Additionally, for the green-field dimensioning case, the exact set of physical links used is also an important aspect, since cost for different links may vary and this can result in drastically different overall cost figures. We will therefore consider the number of line cards, ports/transponders, and fibers as the primary contributors to our optimization goals. While the latter assumptions are a simplification, they do not pose restrictions for the GA approaches themselves, but only affect the surrounding solution method. A more complex representation considering the full extent of these aspects would be possible with minor extensions to the integrated solution methods.

#### *4.1.1.3 Physical Parameters*

The physical parameters largely influence two groups of factors. On the one hand, delay and availability characteristics of physical components determine the attainable QoS figures for

services. On the other hand, transparent reaches and corresponding data rates put bounds on the density of the virtual topology and indirectly affect the number of ports and line cards needed.

The capacity of fibers is given as a fixed number of wavelength slots. Therefore, the number of point-to-point fibers can be extracted from the sum of all wavelength slots needed between two nodes. We consider the bidirectional lower-layer circuits to be aligned to the slot-width, such that any circuit will need the same slot on any physical link traversed. Furthermore, we consider all TXPs to utilize the same transmission formats, which we will model as a table of transparent reach and data rate, which corresponds to the functionality of a flexrate transponder.

Processing and forwarding incurs a delay in upper-layer nodes, which represents a variable component in the overall delay of real networks. However, for reasons of simplicity, we consider this delay to be fixed for our studies and identical on all nodes, regardless of the exact port location and traffic load. This is a reasonable assumption given that upper-layer nodes in the core rarely ever experience excessive traffic bursts due to rate-shaping at the customer connection [277, p. 2] and since they introduce only little delay compared to the transmission delay.

The transmission delay on optical fibers, which is the dominating component in the overall latency, is directly dependent on the fiber length and can be statically determined from node-to-node distances. Lower-layer NEs do not add relevant variable delay, such that the overall delay can be bounded and the remaining uncertainty is covered by the SLA. The same is true for availability, which decreases with increasing fiber lengths. The availability of fibers, or rather the lack thereof, is the dominating component in determining the overall availability, such that we neglect the availability figures of all other NEs.

While other physical impairments are currently not considered, we expected them to have a comparatively small impact on the resulting optimization goals presented in this work. Regardless, a more precise model of physical constraints can be integrated into the simulation framework without impact on the design of the methods under investigation.

### 4.1.2 Traffic Demands

We assume that the knowledge on all traffic matrices is available and sufficiently precise for the intended optimization purposes. This may, for example, either reflect a recent measurement with added safety margins for a dynamic resource assignment or a target specification from a NSP's tender for a planned green-field deployment.

As this work is primarily concerned with transport networks, we expect at least some traffic to originate from and terminate at every node in the network. Furthermore, traffic is typically expected between all node pairs, although this is not a strict requirement for transport networks.

We generally assume symmetric traffic demands, i. e., demand pairs where the source of one is the destination of the other and vice versa. The NSP may require that such reciprocal demands are routed on symmetric paths on identical links. Furthermore, such demand pairs do not typically require identical amounts of capacity in each direction, which is an important difference to circuit routing.

Traffic demands may be subject to SLAs and therefore require a certain QoS-class. We assume all such classes to be known a priori. The required parameters are either an upper bound on the delay or a lower bound on availability, or both. In accordance with the hardware model and physical parameters, delay and availability bounds depend mainly on the used routes. Increasing the availability of a path by providing additional backup paths is out of scope of this work, though a very relevant candidate for future extensions.

Finally, we assume that a demand cannot be split into smaller components that can be routed independently, since these components would incur different delay figures, which can result in out-of-order delivery of packets and can impact delay-sensitive traffic.

### 4.1.3 Optimization Goals

The optimization goal may be an arbitrary function based on the above model and its values, but it should be driven by changing the virtual topology or the routing. For the presented cases we primarily consider resource requirements and QoS fulfillment.

#### *Resource Utilization*

For uncapacitated problems reducing the amount of network resources capable of providing the required capacity is an obvious goal. However, it is also a meaningful goal for capacitated problems, since using the least amount of the limited resources immediately translates to saving spare capacity for additional traffic demands. Without loss of generality, we will focus on resource utilization as the primary goal in the investigated objective functions.

#### *QoS Fulfillment*

Regarding the fulfillment of all SLAs, we will prioritize availability over latency. This choice, however, does not affect the design of the solution approach. The violation of QoS constraints will be treated by penalty functions rather than hard constraints for two reasons. One is providing a more smooth solution scape to the algorithm to facilitate the search process. The second reason is that in actual deployments, it may be beneficial to route as many traffic demands as possible, while finding out how many may miss their guarantees, rather than having the binary statement, whether a traffic demand matrix is feasible or not.

### 4.1.4 Solution Formulation

The multi-layer network dimensioning and dynamic resource assignment problems are constrained by the physical parameters, hardware model, infrastructure and traffic demands with their QoS requirements, as described in the previous sections. Under these assumptions, a solution candidate to these problems can be composed of the routes for each traffic demand and the routes for each circuit. From this information, based on the layer-dependencies as explained in Section 2.1.2.3, the precise hardware and fiber requirements, as well as the resulting QoS values can be determined. Circuits can define the required spectral resources, the virtual topology, and the resulting propagation delays on virtual links. They also provide capacity for the traffic routes, such that the number of ports and TXPs and by extension the remaining hardware requirements can be determined.

The traffic routes can then be overlaid on the hardware and circuits and evaluated regarding capacity and other QoS fulfillment. The resulting detailed description of the network and its configuration can then be graded regarding its quality with respect to the chosen objective function. It should be noted that the dynamic resource assignment problem will require an additional algorithm to determine the transition from the previous to the newly determined configuration and additional conditions may be derived from the transition process and considered as secondary objectives in determining the solution.

Since the complexity of the full multi-layer problem, i. e., jointly optimizing all of the aforementioned aspects, is prohibitive to solving very large problem instances, our GA approaches

will be designed to separately solve a subproblem. The result will in turn be used as input to more traditional heuristics to solve the remaining parts and assemble a full solution to the entire problem from these subsolutions. However, using different subproblems as the basis does impact the multi-layer solution quality and scaling behavior differently. We therefore present several GA approaches, which can be used to solve VTD and TDR and integrate these in a larger solution method.

### 4.1.5 Aspects of Genetic Algorithm Design

The central aspect in adapting the GA framework to specific problems is finding a good genetic coding strategy as detailed in Section 3.4.4. A good encoding is paramount in obtaining high solution quality [37], while operators largely determine the rate of convergence.

The design strategy for encodings was therefore not to over- or underrepresent the solution space whenever possible. However, there are some exceptions for specific applications which will be explained later in this chapter. The second part of the design strategy was to facilitate the usage of simple operators and provide good heritability and locality whenever possible.

The primary design goal of operators is efficiency. They should prevent infeasible solutions and exhibit low computational complexity whenever possible. While generally, their function should not impede the inheritance of any genes, in order to avoid excluding meaningful areas of the search space from exploration, they may apply a controlled amount of bias towards specific genes to accelerate the search by focusing on promising regions of the search space.

## 4.2 Genetic Algorithm Adaptations for Topology Design

The GA approaches in this section apply to topology design in general. As explained in Section 2.2.2 and Section 3.2 of the previous chapters, VTD and PTD are very closely related and their main differences lie in their constraints. While the terminology and constraints used in the description of the following encodings is based on a VTD problem, the encoding itself is applicable to PTD problems or even different topology design problems altogether.

The goal of these encodings is therefore to be as neutral to the technology as possible. Regarding the ability to solve large problem instances, the goal is to enable a precise genetic representation, which is conducive to general genetic operators, especially regarding locality and heritability wherever possible.

### 4.2.1 General Aspects of Topology Formation

As explained in Chapter 2, the virtual topology is the topology of the upper-layer graph. Since the virtual links do not have any capacity limits as they simply represent an adjacency in the routing layer, the only information relevant to define are the source and destination nodes for each virtual link. For networks without intermediate O-E-O regeneration, however, the virtual links are constrained by the maximum transparent reach of the optical circuits used to implement them.

Furthermore, it can be argued that it does not make sense to consider the virtual topology to be a directed graph, since commonly used technologies do always maintain symmetrical connections due to the fact that they need to collect channel information including timing which would be problematic in directed graphs, where the connection in the forward-direction can be arbitrarily longer than that of the reverse direction or vice versa.

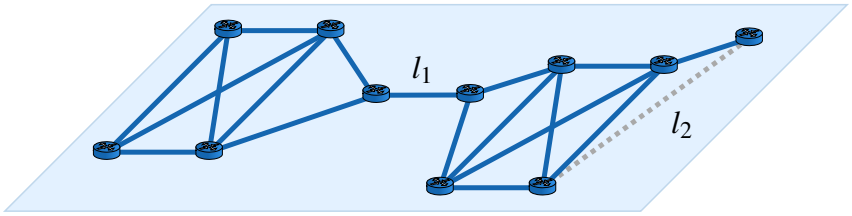



Figure 4.1: Example graph showing different effects on connectivity

It should also be noted, that when the topology design problem is part of a larger context, where a subsequent traffic routing is required to determine the costs, e. g., as part of a dimensioning problem, the effects of topology changes on the objective value may vary drastically. This is caused by the fact that the presence or absence of a single virtual link may just as well affect the entire traffic routing or have little to no effect at all, depending on a combination of the present traffic demand matrix and the geographic properties of the underlying network structure. Consider, e. g., the situation of a shortest-path traffic routing on the virtual topology as shown in Figure 4.1, where the virtual link $l_1$ in the center of the topology connects two well-connected subgroups of nodes. Removing this link will most likely disrupt a large number of paths for many demands, prompting a large change in the objective value. Adding link $l_2$ on the other hand will most likely have little impact on the overall solution, since it will mostly affect traffic between the two nodes it connects.

However, if there were no traffic between the subgroups, then the impact of activating $l_2$ would be larger than the impact of deactivating $l_1$. Due to the wide range of possible effects a topology change can manifest, an encoding that inherently considers such effects would be highly desirable, but most often the precise results of activation or deactivation of individual virtual links is difficult to predict a priori.

### 4.2.2 Virtual Topology Binary Encoding – VTB

This is the most simple topology encoding, which is able to reflect all possible virtual topologies. It represents each potential link as an individual gene of a binary value. It is a regular direct edge encoding, which is very common for pure topology optimization as explained in Section 3.6.1, but rarely seen in the context of multi-layer optimization including traffic and connection routing. Applying this encoding to VTD, we will refer to it as *Virtual Topology Binary* encoding or VTB.

#### *4.2.2.1 Chromosome Structure*

In a virtual topology scenario, this means that every possible virtual link is assigned a single gene representing whether this link is active or inactive by a binary value. Possible virtual links in this context can be considered to be any optically feasible virtual link. Any virtual link between any pair of nodes is considered feasible if the maximum transparent reach is smaller or equal to the length of the shortest path between the node pair within the physical topology. Considering the extreme case of a network of $n_v$ nodes where the transparent reach exceeds the length of all shortest paths the genetic code will consist of $|E_c| = \frac{n_v \cdot (n_v - 1)}{2}$ binary variables. As has been shown in Section 3.2.1, the state space is of size $2^{|E_c|}$, which is in $\mathcal{O}(2^{{n_v}^2})$ for this case.

A chromosome is efficiently representable as a sequence of binary values that can be translated to edges by an array $decodeArr : L \to E_c$ mapping each locus in $L \subseteq \mathbb{N}_0^+$ to the feasible edge candidates $E_c$. While for physical edges in PTD, there may be an explicit cost per edge,

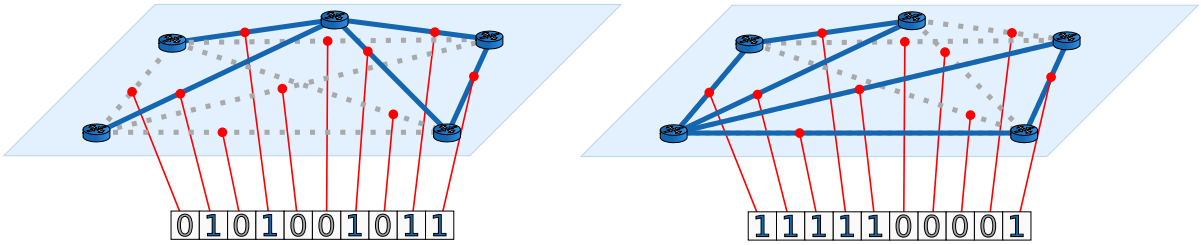



Figure 4.2: Examples for the VTB encoding

this is not typically the case for virtual edges in VTD problems. For VTD there is no obvious order for the links that directly relates to their effects on the cost function. We therefore order them by length, since longer virtual links tend to use spectral resources on more fibers, which is an indirect effect on network resources. Figure 4.2 shows two example topologies for a graph of five nodes, where all virtual links are feasible candidates and active links are shown in blue, while inactive ones are dashed and gray-colored.

#### *4.2.2.2 Advantages and Disadvantages*

The advantage of this formulation lies mainly in avoiding any duplicity in the search space and that it allows for a very efficient and largely adaption-free application of all standard generic operators as presented in Section 3.4. From the perspective of an isolated VTD problem, locality and heredity are good, since a single bit-flip leads to the smallest change in the topology possible and children inherit intact sub-structures. Furthermore, all possible topologies can be expressed by this encoding such that it does not restrict the solution space for VTD problems. This means, that an optimal solution can be found even in the presence of unlikely corner-cases, such as when there is no traffic at all at one or more nodes.

The corresponding drawback lies in the fact, that many solution representations correspond to unconnected topologies that are infeasible for typical traffic scenarios. While this does not make it impossible to identify the optimal solution, it can have a significant impact on the performance. This is especially true, when solving the VTD problem as part of a multi-layer optimization problem. Following the explanation in Section 4.2.1, the VTB encoding can result in a highly multi-modal search space structure which does not lend itself easily to gradual refinements.

Similarly, it is difficult to predict, which groups of links form beneficial substructures, making it impossible for generic recombination operators to exploit any advantages of a gene-sequence since these do not have any clear semantic context relating to the cost values.

#### *4.2.2.3 Evolutionary Operators*

Since the chromosome is a simple bit-vector, all regular mutation and recombination operators intended for binary vectors are immediately applicable. This includes URM, NPM and RRM mutation operations as well as all recombination operators given in Section 3.4.5.5. We have also developed an additional operator specifically for the VTB encoding, which we call Link Block Crossover Operator (LBXO). Rather than exchanging contiguous sequences of genes between chromosomes, it randomly chooses one or more nodes and exchanges all those genes that represent links attaching to the chosen node. Since the number of links that are active for a node can contribute to grooming and since already connected nodes retain their connectivity,

this operator can provide for an inherently meaningful transmission of genes, which would be impossible to encode in an arbitrary, contiguous sequence.

However, as explained in the previous section, a significant portion of the solution space includes topologies that would be infeasible when using hard constraints. Ideally, operators would only apply changes to a chromosome, such that it would result in a similarly fit solution, but validating this by performing the decoding and the computationally more complex evaluation procedures repeatedly, is hardly viable. This leaves two options.

The first is to employ a repair function that can fix issues with the configuration and mirror the required changes back to the chromosome. This is also an approach found in works from topology optimization [192, 246, 285], especially when feasibility is dependent on further conditions such as lower bounds on connectivity. The second approach is to apply advance checks of reduced computational complexity, such that infeasible chromosomes can be discarded early on and replaced in a repeated generation procedure. This is especially useful in situations like a multi-layer context, where the decoding and evaluation procedures require running other algorithms to assemble a full solution, adding further computational cost, while the checks only require a structural analysis of the chromosome itself.

As explained in Section 3.2.1, any connected topology will feature at least $|V| - 1$ edges which are encoded by a corresponding number of 1s in the chromosome. The chromosome can be checked rather quickly for a sufficient number of 1s with a worst-case time complexity of $\mathcal{O}(|V|^2)$. This is typically less then the decode function, which may have a worst-case time complexity of $\mathcal{O}(|V|^4)$ due to repeated path searches in the routing stages of a multi-layer problem. While a minimum number of active edges is a necessary precondition for a connected topology, it is in itself not sufficient to provide this property.

Since the graph with the full candidate edge set is known from the beginning and defines the chromosomes' structure, it is possible to run a preprocessing step before the actual GA, to enable a connectivity verification directly based on the chromosome. To this end, the adjacency matrix composed of all feasible links is computed and stored as an array in the preprocessing step, which requires a worst-case runtime complexity of $\mathcal{O}(|V|^2)$. Afterwards, a feasibility check for connectivity based on the chromosome values and this array can be performed in $\mathcal{O}(|V|^2)$ using a depth-first search approach, starting from the most well-connected node.

An evolutionary operator augmented by this check can therefore perform a *self-healing* function. This can either be achieved by repeating the mutation process until a valid chromosome is created or by adding links to make the topology connected and updating the chromosome accordingly. While the first approach retains good heredity, there is no bound on the number of required mutations to obtain a chromosome for a feasible solution. The second method may reduce heredity by adding traits not previously present in either parent, but it is expected to yield better performance. We follow this notation by implementing Algorithm 3, which uses the adjacency matrix *adj* and the most well-connected node $s_{\text{start}}$ to determine and return, whether the graph is connected. If the Boolean variable $b_{\text{repair}}$ is set, it will alter the chromosome to randomly add previously unused edges to unconnected nodes in order to obtain a structurally similar chromosome, which represents a strongly connected graph.

#### *4.2.2.4 Population Initialization*

A regular randomized population initialization offers the largest diversity, but depending on the sparsity of the graph of feasible edges, may not be particularly useful. In very sparse graphs, this may again result in a large number of infeasible topologies, while very dense graphs may feature an excessive number of edges. This can occur especially for virtual topologies in the

**Algorithm 3** VTB Chromosome Connectivity Check and Repair

**Require:** *chromo* is a binary array representing a VTB-coded chromosome
**Require:** *adj* is a bivariate map associating node pair ids to loci of corresponding links
**Require:** $s_{\text{start}}$ is the id of the most well-connected node
**Require:** $b_{\text{repair}}$ is a Boolean value indicating the use of the repair function

```
function CheckAndRepairConnectivity(chromo,adj,s_start,b_repair)
    let D_u ← {1,...,√|adj|} \ {s_start}        ▷ Set of ids of unconnected nodes
    let D_s ← {s_start}                          ▷ Set of ids of already connected nodes
    let E_r ← ∅                                  ▷ Set of node id pairs representing repair edges
    let w[1] ← s_start                           ▷ Map of node ids to inspect
    let s ← s_start                              ▷ Next id to inspect
    let i ← 1                                    ▷ Counter for remaining ids
    while D_u ≠ ∅ and i > 0 do                   ▷ Connectivity check
        s ← w[i]
        w[i] ← ⊥
        i ← i − 1
        for all d ∈ D_u do
            if adj[s][d] ≠ ⊥ then
                if chromo[adj[s][d]] = 1 then
                    D_u ← D_u \ {d}
                    D_s ← D_s ∪ {d}
                    w[i] ← d
                    i ← i + 1
                else
                    E_r ← E_r ∪ {⟨s,d⟩}
                end if
            end if
        end for
    end while
    while D_u ≠ ∅ and b_repair = true and E_r ≠ ∅ do   ▷ Repair function
        ⟨x,y⟩ ← Rnd(E_r)
        if x ∈ D_s and y ∈ D_u then
            D_u ← D_u \ {y}
            D_s ← D_s ∪ {y}
            chromo[adj[x][y]] ← 1
            E_r ← E_r \ {⟨a,b⟩ ∈ E_r | b = y}
        else if y ∉ D_u then
            E_r ← E_r \ {⟨a,b⟩ ∈ E_r | b = y}
        end if
    end while
    if D_u = ∅ then
        return true
    else
        return false
    end if
end function
```

presence of flexrate devices, which can drastically increase the number of realizable connections. While the extension to the operators explained in the previous section can prevent the creation of unconnected solutions, the initial population can also be built such that it already contains meaningful individuals.

There are two topologies which are inherently meaningful to consider, although their objective values can be far from the optimal solution's. The first applies to cases, where a meaningful reference topology exists. In case of a multi-layer problem, this may be the topology where all active virtual links correspond exactly to the available physical links. Including an individual realizing such a topology means that the result of the overall optimization can never be worse than this basic reference solution.

The second topology is the full graph of all candidate edges, such that all gene values are set to "1". While this may not in itself be a viable solution, especially not for graphs with dense candidate edge sets, it will form a connected topology. Furthermore, it is guaranteed to include all shortest paths, such that all QoS constraints are fulfilled, if this is at all possible. As such, it may serve as a source of building blocks which can migrate into more sparse chromosomes through recombination.

The least-connected meaningful solution candidates on the other hand always form spanning trees due to the fact that all nodes in the network should be connected. In this way, every traffic demand is guaranteed to be routable. While this is a necessary, but not a sufficient condition to fulfill potential QoS targets, chromosomes representing spanning trees can represent a good starting point for the integration of beneficial traits. Similar to Saha *et al.* [244], a population can therefore be initialized using different randomized spanning trees to provide a certain level of diversity. Contrary to the Prüfer sequence approach by Saha *et al.* [244], we have implemented Algorithm 4, which works very similar to Prim's algorithm. Rather than maintaining a list of

**Algorithm 4** Randomized Spanning Tree Creation

**Require:** $\langle V, E \rangle$ is a connected, non-empty simple graph
  **function** RNDSPANNINGTREE($V$,$E$)
    **let** $E_c \leftarrow \emptyset$ ▷ Set of candidate edges
    **let** $E_t \leftarrow \emptyset$ ▷ Set of edges in the emerging spanning tree
    **let** $v_n \leftarrow \text{RND}(V)$ ▷ Random starting vertex
    **let** $V_w \leftarrow V \setminus \{v_n\}$ ▷ Set of remaining unconnected nodes
    **while** $V_w \neq \emptyset$ **do**
      $E_c \leftarrow E_c \cup \{\langle s,d \rangle \in E | (s = v_n \wedge d \in V_w) \vee (d = v_n \wedge s \in V_w)\}$
      $\langle x,y \rangle \leftarrow \text{RND}(E_c)$
      $E_t \leftarrow E_t \cup \{\langle x,y \rangle\}$
      **if** $x \in V_w$ **then**
        $v_n \leftarrow x$
      **else**
        $v_n \leftarrow y$
      **end if**
      $E_c \leftarrow E_c \setminus \{\langle a,b \rangle \in E_c | a = v_n \vee b = v_n\}$
      $V_w \leftarrow V_w \setminus \{v_n\}$
    **end while**
    **return** $\langle V, E_t \rangle$
  **end function**

candidate edges ordered by their weight, it simply keeps track of the set of unconnected nodes and randomly establishes an adjacency to one of these based on the available candidate edges.

However, most actual transport network topologies are much more well-connected than a spanning tree, as shown in Section 2.2.2. A good solution candidate will therefore also exhibit a higher level of connectivity than a pure spanning tree. To address this, we propose an initialization procedure that will add further edges based on an average target ratio of active to feasible additional links, which presents an additional hyperparameter, we denote as $p_t$. We will refer to this procedure, which is shown in Algorithm 5, as Augmented Spanning Tree Initialization (ASTI).

Additionally, an adapted version of Algorithm 5 can also be used, where the basic topology is not a randomized spanning tree, but rather a known good topology. This approach is especially useful when applied to a reconfiguration scenario, where the previously used configuration will by definition include a good topology. In a multi-layer context and especially in partly dimensioned scenarios, where the physical topology is fixed, it can also be used as the base. When using the physical topology, we will refer to this approach as Augmented Physical Topology Initialization (APTI).

By using one or more of the above strategies, the initial population will not only feature reasonably good configurations, but also drastically decrease the number of evolutionary perturbations required to get to similar or even better states. While these approaches have a non-negligible runtime complexity[1], they are only executed once per individual during initialization, such that the overall performance impact is relatively small.

[1]Worst case time complexity for a simple implementation is in $\mathcal{O}(n^2 \log n)$

**Algorithm 5** Augmented Spanning Tree Initialization (ASTI) Procedure

**Require:** $\langle V, E_f \rangle$ is a connected, non-empty simple graph including all feasible edges
**Require:** *linkToLocus* is the mapping of links to loci
**Require:** $p_t \in [0,1]$ is the target ratio of active to inactive additional links
**function** CREATEINITIALASTCHROMOSOME($V, E_f, linkToLocus, p_t$)
  **let** $n_g \leftarrow |V| \cdot (|V|-1) \cdot 0.5$ ▷ Length of a Chromosome
  **let** $chromosomeArr : \{1,\dots,n_g\} \to \mathbb{B}$ ▷ Chromosome as Array of $n_g$ binary values
  **let** $n_{ones} \leftarrow \lceil (|E_f| - |V| + 1) \cdot p_t \rceil$ ▷ Target number of "1" values
  **let** $L_{zeros} \leftarrow \{1,\dots,n_g\}$ ▷ Set of loci with gene value "0"
  $\langle V, E_c \rangle \leftarrow$RNDSPANNINGTREE($V, E_f$) ▷ Cf. Algorithm 4
  **for all** $e \in E_c$ **do**
    $chromosomeArr[linkToLocus[e]] \leftarrow 1$
    $L_{zeros} \leftarrow L_{zeros} \setminus \{linkToLocus[e]\}$
  **end for**
  **while** $n_{ones} > 0$ **and** $L_{zeros} \neq \emptyset$ **do**
    $l \leftarrow$ RND($L_{zeros}$)
    $L_{zeros} \leftarrow L_{zeros} \setminus \{l\}$
    $chromosomeArr[l] \leftarrow 1$
    $n_{ones} \leftarrow n_{ones} - 1$
  **end while**
  **return** *chromosomeArr*
**end function**

### 4.2.3 Centralized Spanning Tree Encoding – VTCS

For the regular operation of transport networks, the flat encoding's primary disadvantage is the fact that the genes can represent a large number of network configurations that are clearly undesirable. Rather than mitigate this through the use of specific initialization procedures and operators, an encoding may also be designed such that it cannot even represent unconnected topologies. While some problems may require representations for these states, it is not meaningful in a transport network environment, where client traffic may require transport between any two nodes at any time. Therefore, the virtual topology of any transport network will always contain a spanning tree, which we will consider as the basis for the VTCS encoding.

In very simple cases, any spanning tree alone can mark an optimal solution. E. g., when the sum of the data rates of all traffic demands in a network is smaller than the circuit granularity, while the only optimization goal is to achieve the least number of circuits. In this case, any topology forming a spanning tree of minimal edge count is a resource-optimal solution in the absence of further requirements. When additionally considering the number of NEs, e. g., line cards with several ports, then only a specific subset of all possible spanning trees will remain optimal solutions. More specifically, the subset where the number of links at any node does not exceed the number of ports on a line card. This illustrates the point, that depending on the exact problem, some spanning trees can be optimal, while others can not. Furthermore, when adding QoS parameters, nodes will most likely require more direct paths due to latency constraints such that no spanning tree alone can be optimal. Due to these considerations, an encoding which combines a spanning tree with any combination of additional edges, can represent any optimal solution to a VTD problem.

As outlined in Section 3.6.1, a standard approach to encode spanning trees for GAs is to describe them as Prüfer sequences. However, this is not desirable for the present use case for two reasons. First, it's known not to be a good fit for efficient GAs due to problems with heritability and locality [109], which may significantly impede performance. The second reason is that a full representation of all possible spanning trees would significantly inflate the search space. To illustrate this fact, one can consider adding a single link to a node in a spanning tree of $n$ nodes. The graph with the additional link could just as well be the product of a different spanning tree, which had included this new edge in their original tree and to which another edge had been added, which is part of first spanning tree. In the worst case of a topology that becomes a ring after adding a new link, there are $n-1$ spanning trees, which can be considered to be a part of the ring topology. Therefore, an encoding based on arbitrary spanning trees may have many representations for the same topology.

Cayley's formula [31] states that for a fully connected graph of $n$ vertices there are $n^{n-2}$ possible trees. This gives us an upper bound for the number of spanning trees, which exceeds the number of edges in this fully connected graph. Therefore, directly encoding a full enumeration of spanning trees is not a promising approach since its combinatorial complexity exceeds that of the flat encoding significantly, while many trees are redundant for most topologies. To make a spanning tree-based approach viable, the number of spanning trees needs to be reduced to as few relevant candidates as possible.

If the cost contribution of each link were known a priori, Prim's algorithm could be used to determine the subset of minimal-cost spanning trees, but, as has been previously explained in Section 4.2.1, the exact cost contribution can only be determined once the full configuration including all other edges is known. However, as the example of the same section has shown, there are links which are intuitively more important than others, e. g., because the graph would be unconnected without them. Given a heuristic quantifying this importance, edges could be

assigned an artificial weight which could then be used to determine a minimum spanning tree as likely candidates for cost-optimal trees. Furthermore, if a strict total order of edges could be established based on this heuristic, such that no pair of edges have identical weights, the resulting tree generation would always result in the same spanning tree for the same set of weights.

While this importance is difficult to quantify, there are a number of properties, that can be used to formulate a meaningful heuristic. While the closeness centrality used in Section 2.2.2.1 is an interesting candidate for this, it is purely based on the topology, which may be misleading in scenarios with sparse traffic matrices. We therefore suggest an order relation based on betweenness centrality [87], which reflects the number of shortest paths that use a given resource in a graph. While regular betweenness centrality considers the shortest paths for all pairs of nodes, we suggest omitting these nodes, that do not have any traffic demands between them, should such nodes exist. In this way, we can reflect a link's importance given both, topology and traffic demand matrix.

Betweenness centrality measures exist for both, edges and nodes. We will explicitly not use edge betweenness [151] but rather use the betweenness centrality of the nodes a link connects. The reason behind this is to decouple the measure from the exact link representation. E. g., when considering a full virtual topology, every shortest path will only contain the single link between source and target, such that each of these edges have identical edge weights, rendering any order based on edge betweenness ineffective. We therefore assign to each link the sum of the betweenness centrality values of the nodes connected by the respective link. Furthermore, we use the node's unique identifiers as a tie breaker in order to establish a strict total order on edges. While we expect a minimum spanning tree based on these edge weights to be a good candidate to augment with additional links towards a full solution, it may still prevent more sparse topologies from being representable.

Rather than relying on a single spanning tree obtained from Prim's algorithm, we use a gene value to encode a sequence of nodes, which form the spanning tree and use the heuristic from above to determine, which link will be used to connect the selected nodes. In this way, the number of spanning tree candidates is reduced and the resulting trees should reflect the most important variations based on the heuristic. While it cannot be guaranteed, that optimal solutions for all corner cases are encodable, we expect this to be a relatively rare event when applying this scheme to multi-layered transport networks. Given the typical density of such networks it is unlikely that the solution will hinge on the basic spanning tree without any augmenting links. The exact encoding strategy is explained in the following section.

#### *4.2.3.1 Chromosome Structure*

The chromosome consists of two distinct parts. The first part encodes the spanning tree and the following part encodes which of the remaining edges are used to augment the tree topology. We will therefore refer to the genes of the first part as "tree genes" and to the genes of the second part as "augment genes". For this encoding a number of steps can be precomputed using Algorithm 6. The algorithm applies a sort-function using the established strict total order relation on the edges.

Furthermore, it computes $n_{STGenes}$, the number of tree genes, $n_{STValues}$, which is the number of different gene values, and also $n_{VLGenes}$, the number of genes to encode the remaining

**Algorithm 6** Spanning Tree Encoding – Initialization Procedure

**Require:** $\langle V, E_f \rangle$ is a connected, non-empty graph
**Require:** $\succeq_e$ providing a strict total order relation on $E_f$
**procedure** STGINIT($V$,$E_f$,$\succeq_E$)
**let** $edgeArr : \{1,\ldots,|E_f|\} \to E_f$
**let** $edgeMap : V \times V \to E_f$
**let** $sortedEdgeArr \leftarrow$SORT($E_f$,$\succeq_E$)
**let** $n_{STGenes} \leftarrow |V|-2$
**let** $n_{STValues} \leftarrow |V|-1$
**let** $n_{VLGenes} \leftarrow |E_f|-(|V|-1)$
**for all** $v \in V$ **do**
$outEdgeSet \leftarrow \{\langle s,d \rangle \in E_f | s = v\}$
**for all** $\langle s,d \rangle \in outEdgeSet$ **do**
$edgeMap[v][\mathrm{d}] \leftarrow \langle s,d \rangle$
**end for**
**for all** $u \in V$ **do**
**if** $v \neq u$ **and** $edgeMap[v][u] = \perp$ **then**
$edgeMap[v][u] \leftarrow$DIJKSTRA($\langle V, E_f \rangle, v, u)[0]$ ▷ Cf. Algorithm 11
**end if**
**end for**
**end for**
**let** $v_{start} \leftarrow$MAXDEGREENODE($V$,$E_f$) ▷ Cf. Algorithm 13
**end procedure**

augmentation links. Therefore, the total number of genes in such a VTCS chromosome can be determined according to Equation (4.1).

$$n_{g,\mathrm{VTCS}} = (|V|-2)+|E_f|-(|V|-1) \tag{4.1}$$

Note, that $n_{STGenes}$ is smaller than $|V|$, because the last remaining node has to be added without any alternative and the first node is statically assigned as the node with the highest nodal degree in the graph of edge candidates. This results in a state space of size

$$(|V|-1)!+2^{|E_f|-(|V|-1)} \in \mathcal{O}(|V|!+2^{|V|^2}) \tag{4.2}$$

under the assumption of symmetric edges.

The most important part of the initialization procedure is the preparation of the *edgeMap*. This array contains the first edge on a path between any pair of nodes. For neighboring nodes this is their connecting edge, while for indirectly connected nodes, it is the first edge of the shortest path between them. This information is used in the decoding procedure, presented as Algorithm 7. This procedure creates the network data structure from the chromosome, based on the initialization values. Figure 4.3 illustrates this process step by step.

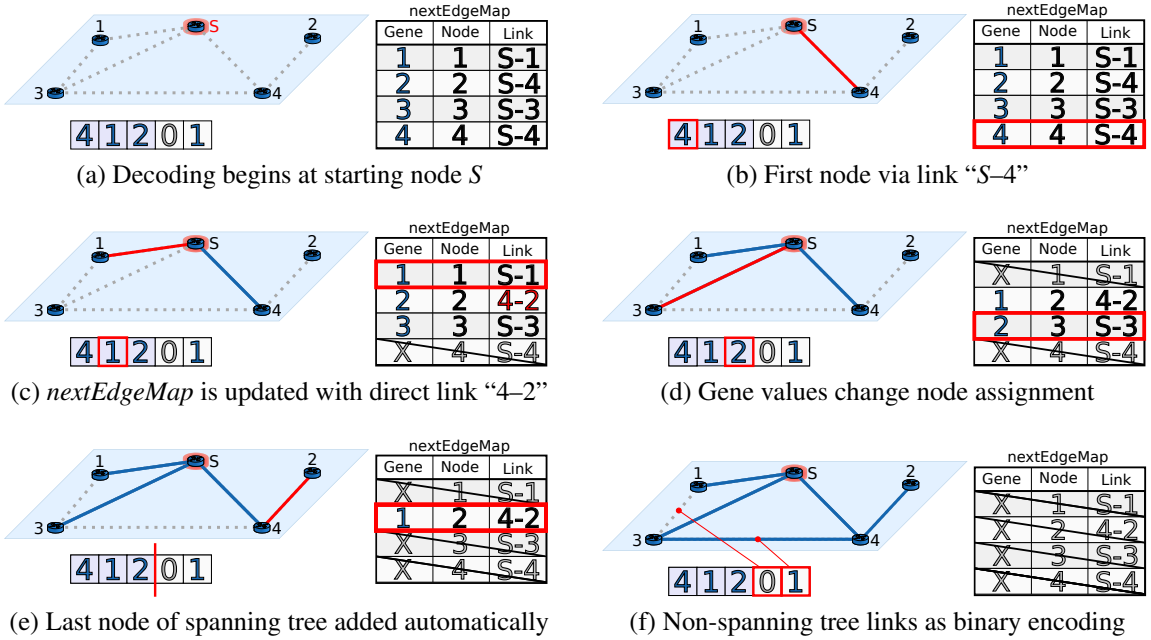
The chromosome is decoded beginning from the starting node “S”, which is highlighted in red, and using the given *nextEdgeMap* indicating choices for next links. The first gene with the value 4 adds node 4 by using the link between “S” and 4 as highlighted in Figure 4.3b. This leaves only three options for the next link, thereby reducing the number of entries in the *nextEdgeMap*. Furthermore, node 2 can now be reached with a more direct link from node 4, such that the

**Algorithm 7** Spanning Tree Encoding – Decoding Function

**Require:** $\langle V, E_f \rangle$ is a connected, non-empty graph
**Require:** *sortedEdgeArr* provides an ordered set of feasible edges
**Require:** *geneArr* provides the sequence of genes
**function** DECODE($V, E_f, sortedEdgeArr, edgeMap, v_{start}, geneArr, n_{STGenes}$)
**let** $workEdgeArr \leftarrow sortedEdgeArr$
**let** $workNodeSet \leftarrow V \setminus \{v_{start}\}$
**let** $i_{target} \leftarrow 0,\ i \leftarrow 0,\ v_{curr} \leftarrow v_{start}$
**let** $nextEdgeMap \leftarrow edgeMap[v_{start}]$
**let** $actEdgesSet \leftarrow \emptyset$
**for** $i_{gene} = 0$ **to** $i_{gene} = n_{STGenes} - 1$ **do**
$i \leftarrow 0$
$i_{target} \leftarrow geneArr[i_{gene}]$
$nextEdgeMap[v_{curr}] \leftarrow \perp$
**for all** $\langle v, \langle s, d \rangle \rangle \in nextEdgeMap$ **do**
**if** $i \leq i_{target}$ **then**
$v_{curr} \leftarrow d$
**if** $v_{curr} \in workNodeSet$ **and** $\exists \langle n, \langle s, d \rangle \rangle \in workEdgeArr$ **then**
**if** $i = i_{target}$ **then**
$workNodeSet \leftarrow workNodeSet \setminus \{v_{curr}\}$
$workEdgeArr[n] \leftarrow \perp$
$\exists! \langle m, \langle d, s \rangle \rangle \in workEdgeArr$
$workEdgeArr[m] \leftarrow \perp$
$actEdgesSet \leftarrow actEdgesSet \cup \{\langle s, d \rangle, \langle d, s \rangle\}$
**end if**
$i \leftarrow i + 1$
**end if**
**end if**
**end for**
UPDATE($nextEdgeMap, v_{curr}, workNodeSet, edgeMap, workEdgeArr$) ▷ Algorithm 8
**end for**
$\exists! v \in workNodeSet$ ▷ Only exactly one element remaining
$v_{curr} \leftarrow v$
$\langle s, d \rangle \leftarrow nextEdgeMap[v_{curr}]$
$\exists! \langle n, \langle s, d \rangle \rangle \in workEdgeArr$
$workEdgeArr[n] \leftarrow \perp$
$\exists! \langle m, \langle d, s \rangle \rangle \in workEdgeArr$
$workEdgeArr[m] \leftarrow \perp$
$actEdgesSet \leftarrow actEdgesSet \cup \{\langle s, d \rangle, \langle d, s \rangle\}$
**for all** $e \neq \perp \in workEdgeArr$ **do**
$i_{gene} \leftarrow i_{gene} + 1$
**if** $geneArr[i_{gene}] = 1$ **then**
$actEdgesSet \leftarrow actEdgesSet \cup \{e\}$
**end if**
**end for**
**return** *actEdgesSet*
**end function**

**Algorithm 8** Spanning Tree Encoding – Subroutine to update *nextEdgeMap* for $v_{curr}$

**procedure** UPDATE(*nextEdgeMap*,$v_{curr}$,*workNodeSet*,*edgeMap*,*workEdgeArr*)
  **for all** $v \in workNodeSet$ **do** ▷ Update next edge table with new adjacencies
    $\langle s,d \rangle \leftarrow edgeMap[v_{curr}][v]$
    **if** $d \in workNodeSet$ **and** $\exists \langle n,\langle s,d \rangle\rangle \in workEdgeArr$ **and** $nextEdgeMap[v] \neq \perp$ **then**
      $\langle o,t \rangle \leftarrow nextEdgeMap[v]$
      **if** $(t \neq v$ **and** $d = v)$ **or** $v_{curr} = t$ **or** $(\langle o,t \rangle \succeq_E \langle s,d \rangle$ **and** $t \neq v)$ **then**
        $nextEdgeMap[v] \leftarrow \langle s,d \rangle$
      **end if**
    **end if**
  **end for**
**end procedure**




(a) Decoding begins at starting node *S*

(b) First node via link "*S*–4"

(c) *nextEdgeMap* is updated with direct link "4–2"

(d) Gene values change node assignment

(e) Last node of spanning tree added automatically

(f) Non-spanning tree links as binary encoding

Figure 4.3: Example of the VTCS decoding procedure

previous entry for gene value 2 is updated with the new link as shown in Figure 4.3c. The decoding continues until we reach the situation illustrated in Figure 4.3e, where only one entry remains in the *nextEdgeMap*, such that no choice is represented in the chromosome. Finally, the remaining augmentation links are added in the same way as with the VTB encoding, which is depicted in Figure 4.3f, where one additional link is chosen.

Figure 4.4 shows the two example topologies from the previous section describing the VTB encoding and how they can be encoded using the VTCS approach. Note, that the binary genes encoding the additional links change their phenotype depending on the spanning tree genes. This is due to the fact that a spanning tree link is not allowed to be changed in subsequent genes and is therefore removed from the binary part of the gene sequence.

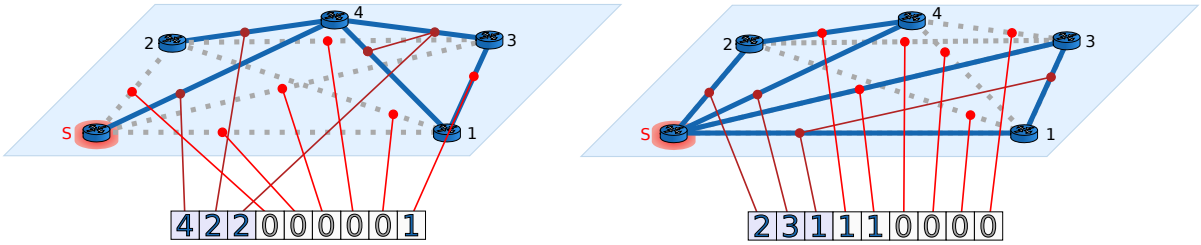



Figure 4.4: Examples for the VTCS encoding

#### 4.2.3.2 *Advantages and Disadvantages*

In comparison to the VTB encoding, the VTCS encoding has the advantage that all possible chromosomes always result in connected topologies, such that no connectivity check or repair functions are necessary. Both, the repair function described in Algorithm 3 and the decoding procedure for VTCS exhibit worst-case runtime complexities within $\mathcal{O}(|V|^2)$ making their performance impact comparatively small. The advantage of the VTCS encoding compared to VTB is that heritability can be improved in certain situations where the solutions are very sparse graphs. For example, when the recombination of two VTB chromosomes results in a chromosome that requires many additional links to become connected again, the repair function may add edges in a way that disrupts effective patterns in the parents. This can result in an offspring with a much lower objective value that detracts from a more localized improvement.

While VTCS has an advantage in these situations, it also has a number of drawbacks regarding state space, locality, and heritability. In situations where the starting node is the same as or close to the one with the highest centrality, but the optimal solution consists solely of links on remote, non-central or non-well-connected nodes, then the VTCS encoding may not be able to represent this solution and a GA based on it is therefore unable to find it. It should therefore not be used in situations where very sparse traffic matrices are expected.

Another drawback are overrepresentations in state space, i. e., the fact that the same topology can be encoded in a number of different ways. While the VTCS decoding procedure and the suggested order relation reduce the amount of representable spanning trees and thereby also the amount of overrepresentation, an inflation of the state space will still occur, which becomes increasingly worse for more densely connected graphs. Finally, there are also implications on locality and heritability. As shown in Figure 4.4, any link used as part of the spanning tree is removed from the binary genes in the chromosome. This means that a mutation in the spanning tree genes can theoretically change the phenotypical meaning of all subsequent genes, such that a small local mutation can lead to a large-scale change in the resulting individual. Similarly, this can have a negative impact on heritability as well, since recombining genes from two parents of different spanning trees can result in offspring where the inherited sequences result in different phenotypes.

#### 4.2.3.3 *Evolutionary Operators*

Due to the segmented nature of this chromosome type and especially the different domains, not all regular operators are immediately applicable.

### *Mutation*

Regarding mutation, purely binary approaches cannot be used, but since binary numbers are a subset of the integers, most[1] regular mutation operators intended for integer numbers can be applied to a VTCS-coded chromosome. This includes the Gaussian and Creep operators as outlined in Section 3.4.5.4. However, using standard operators can result in reduced locality as explained in the previous sections.

When the number of link candidates drastically exceeds the number of nodes and the mutation rate is limited such that only one or two genes are altered, then the probability of changing the tree genes becomes small enough that negative effects on locality remain small. However, for less dense topologies, the reduction in locality can become problematic.

To remedy this aspect, we suggest treating the tree and augmentation genes separately in an encoding-specific mutation operator. Any mutation will only affect either the tree genes or the augmentation genes. The choice is performed randomly according to a configurable threshold. For the tree genes we apply a Creep mutation with step size $n = 1$, while the augmentation genes are altered by a variable secondary operator like URM or RRM. We will refer to this operator as VTCS-Specific Mutation (VSM).

### *Recombination*

Since the ranges of numerical values for the genes within a VTCS-coded chromosome are static for a given candidate topology, any regular, non-reordering recombination approach such as those outlined in Section 3.4.5.5 can be used. However, for these operators in general and UXO in special, the previously explained negative effects of reduced locality and heritability become manifest. Since the edges encoded within the spanning tree cannot be present in the remaining genes, the loci change their meaning depending on the spanning tree. This effect is partly mitigated by retaining the order according to the order relation, but the exact shift depends on the chromosomes to be recombined and can be substantial.

We therefore suggest an alternate recombination operator, that is aware of the phenotypes associated with the alleles in VTCS-chromosomes. We refer to this operator as VTCS-Specific Crossover (VSXO). The recombination of tree genes has a high potential to disrupt any meaningful patterns, since these only emerge as a combination of tree and augment genes. The suggested recombination approach therefore tries to migrate intact subsets of links between chromosomes, rather than flat gene values.

Since the association of phenotypes to genes is known for each parent, the recombination algorithm can establish which genes result in equivalent phenotypes. This is illustrated in Figure 4.5a, where connected genes correspond to the same links in the resulting topologies. Two offspring are created with tree genes that are identical to those of their parents, which means that the equivalence of loci remains identical as well as shown in Figure 4.5b. Thus, the recombination transfers the phenotypical associations of the parents to the children. In the following step, augment genes that correspond to tree genes in the other chromosome need to be activated, which can be seen in Figure 4.5c. Finally, all remaining augment genes are recombined according to a fixed crossover probability $p$. In the example in Figure 4.5d, $p$ is set to 1, such that the resulting offspring each have the tree genes of one parent and the augment genes of the other, where possible.

While heritability is significantly improved through this operator, it does not result in a heritability equal to the VTB encoding, since there are certain phenotypes that cannot be passed

[1] Operators that reorder subsequences of genes, which are used for permutation encodings, are not applicable.


(a) Establishing phenotypical equivalence between loci


(b) Clone tree gene sequences and phenotype equivalence


(c) Activate augment genes corresponding to opposite tree genes


(d) Recombine remaining augment genes (here with $p = 1.0$)

Figure 4.5: Process of the VSXO recombination operator

on to the offspring. This occurs, for every link which is represented in the augment genes of one parent and the tree genes of the other parent. When the augment genes require its absence from the topology, but the offspring shares the same tree as the second parent, it cannot be removed as this would contradict the offspring's tree genes. This is visible in Figure 4.5c, where the $6^{\text{th}}$ gene of the upper offspring cannot have the value 0. Note, that this introduces an imbalance in the heritability between 1 and 0 values in the binary genes, such that offspring topologies tend to be increasingly dense, which may not be desirable depending on the problem at hand.

#### *4.2.3.4 Population Initialization*

Just like with VTB, a population can be seeded with a full candidate topology or randomized spanning trees. To this end, we have implemented an adapted version of the ASTI approach, such that it can be used to initialize populations for VTCS-coded chromosomes. The advantage here, is that the random generation of spanning trees is very simple, since it only requires the randomization of the tree genes. We have also developed an initialization procedure which is analogous to APTI.

However, since VTCS does not allow all topologies to be encoded, the adapted APTI approach may sporadically add additional edges, when it is incapable of creating a chromosome capturing the physical topology exactly. Therefore, the resulting topology may only be an approximation of the physical topology. Since there typically exist several ways to encode a requested topology, the suggested approach attempts to randomly find different encodings, to introduce a meaningful amount of diversity. The full initialization approach is given as Algorithm 9, where $\langle V, E_{\text{init}} \rangle$ should represent the physical topology in order to realize APTI.

**Algorithm 9** Spanning Tree Encoding – Augmented Initial Topology Initialization

**Require:** $\langle V, E_{\text{init}} \rangle$ is a connected, non-empty graph
**Require:** *sortedEdgeArr* provides an ordered set of feasible edges
**function** CREATEINITCHROMOSOME($V$,$E_{\text{init}}$,*sortedEdgeArr*,*linkMap*,$v_{\text{start}}$,$n_{\text{STGenes}}$)
**let** $workEdgeArr \leftarrow sortedEdgeArr$
**let** $workNodeSet \leftarrow V \setminus \{v_{start}\}$
**let** $geneArr \leftarrow \langle i, 0 \rangle$ with $i \in 1, \cdots, n_{\text{STGenes}}$
**let** $i_{target} \leftarrow -1$ , $v_{curr} \leftarrow v_{start}$ , $n_{curr} \leftarrow -1$
**let** $nextEdgeMap \leftarrow linkMap[v_{start}]$
**let** $candEdgesArr$
**for** $i_{gene} = 1$ **to** $i_{gene} = n_{\text{STGenes}}$ **do**
$i_{target} \leftarrow -1$
$candEdgesArr \leftarrow \langle i, \perp \rangle$ with $i \in \{1, \cdots, |V|\}$
$nextEdgeMap[v_{curr}] \leftarrow \perp$
**for all** $\langle v, \langle s, d \rangle \rangle \in nextEdgeMap$ **do**
**if** $d \in workNodeSet$ **and** $\exists \langle n, \langle s, d \rangle \rangle \in workEdgeArr$ **then**
$i_{target} \leftarrow i_{target} + 1$ , $v_{curr} \leftarrow d$ , $n_{curr} \leftarrow n$
**if** $\langle s, d \rangle \in E_{\text{init}}$ **then**
$candEdgesArr[i_{target}] \leftarrow \langle n, \langle s, d \rangle \rangle$
**end if**
**end if**
**end for**
**if** $\exists \langle n, \langle x, e \rangle \rangle \in initEdgesArr$ **and** $e \neq \perp$ **then**
$\langle n, \langle x, \langle s, d \rangle \rangle \rangle \leftarrow \text{RND}(initEdgesArr)$
$i_{target} \leftarrow x$ , $v_{curr} \leftarrow d$ , $n_{curr} \leftarrow n$
**end if**
$geneArr[i_{gene}] \leftarrow i_{target}$
$workNodeSet \leftarrow workNodeSet \setminus \{v_{curr}\}$
$workEdgeArr[n_{curr}] \leftarrow \perp$
$\exists! \langle m, \langle d, s \rangle \rangle \in workEdgeArr$
$workEdgeArr[m] \leftarrow \perp$
UPDATE($nextEdgeMap$,$v_{curr}$,$workNodeSet$,$linkMap$,$workEdgeArr$) ▷ Algorithm 8
**end for**
$\exists! v \in workNodeSet$ ▷ Only exactly one element remaining
$v_{curr} \leftarrow v$
$\exists! \langle n, \langle s, d \rangle \rangle \in workEdgeArr$
$workEdgeArr[n] \leftarrow \perp$
$\exists! \langle m, \langle d, s \rangle \rangle \in workEdgeArr$
$workEdgeArr[m] \leftarrow \perp$
**for all** $e \neq \perp \in workEdgeArr$ **do**
$i_{gene} \leftarrow i_{gene} + 1$
**if** $e \in E_{\text{init}}$ **then**
$geneArr[i_{gene}] \leftarrow 1$
**end if**
**end for**
**return** *geneArr*
**end function**

## 4.3 Genetic Algorithm Adaptations for Traffic Demand Routing

This section details the encodings that are concerned with applying GA mechanisms to traffic routing. While TDR and ICR are in essence both routing problems, as we have explored in Section 2.2.3 and Section 3.2, we will not include any ICR-specific aspects. While most basic approaches apply to both problem versions, ICR adds another layer of complexity due to its relationship with the wavelength assignment problem, suggesting more specialized approaches beyond routing. However, exploring this dependency and the integration of possible heuristics in detail is well beyond the scope of this monograph, such that we will focus on TDR.

The goals regarding the genetic representations are the same as for the topology adaptions in the previous section. They shall be able to support generic operators, provide good locality and heritability, while also allowing for a computationally efficient decoding. However, when it comes to large problem instances, the combinatorial complexity of routing problems quickly becomes prohibitive. A full solution space representation is therefore unlikely to be solvable in a feasible time frame for most problems, such that we will consider performance-enhancing simplifications for the following encodings.

### 4.3.1 General Aspects of Routing Traffic Demands

Routing of a single traffic demand requires an underlying graph, that has sufficient connectivity to permit a path between the demand's source and destination. A solution to a TDR problem will therefore consist of an assignment of paths through the given graph for all demands. Algorithms solving TDR need to consider, whether demands are symmetric, require QoS-compliant paths and if they are routable at all.

Routability may not only depend on the graph's connectivity, but also on predefined capacity limits. For capacitated problems, only a limited amount of demands can be carried on a path, before it runs out of capacity and longer, alternative paths are needed, which may require additional connections. Once resources on all possible paths are depleted, a demand becomes unroutable. For uncapacitated problems, where resources can be added at further cost, all demands will be routable, such that no solution can become infeasible, but the overall cost may be excessive. In both situations, it can make sense to explicitly treat a demand as unroutable. For capacitated problems this can, e. g., be meaningful for a best-effort demand of little revenue and low penalties, when in its place a higher-revenue demand can be routed. A similar situation may occur for uncapacitated problems, when the best-effort demand would require excessive amounts of additional hardware, such that the costs exceed the penalties.

Symmetric demands, meaning that every demand will have a related demand in the reverse direction, are a common occurrence, since most communication is inherently symmetric. Routing a demand and its opposite counterpart on different, non-symmetrical routes can be possible for connectionless packet traffic, but depending on the client traffic or technological constraints it may also be forbidden, posing an important constraint to routing. E. g., classical IP/MPLS is not required to use symmetric routes, whereas MPLS-Transport Profile (MPLS-TP) includes an explicit option to enforce it.

QoS-enabled demands effectively add constraints to the path search, which typically increase the routing problem complexity. This is not only caused by compliance checks, which are likely to increase the runtime, but also because the number of path computations increases. Without QoS constraints, in most cases only one path per node pair needs to be computed, whereas including the constraints will require a path per node pair multiplied by the number of QoS

classes. This may also lead to very different network configurations, since the more diverse paths may require additional or share different existing resources.

When it comes to encoding solutions to TDR, different aspects of a routing can be considered, as explained in Section 3.6.2. The encoding may explicitly address all edges in each path by individual genes, which is done by the direct edge and node sequence encodings. This corresponds to the smallest possible difference between two TDR solutions and therefore allows for very high locality and an exact representation of the search space, but it also entails a high combinatorial complexity and hence limited tractability. Another category of encodings indirectly represents the entire routing at once, using the genes to encode parameters for an algorithm, which derives the routing from these values. An example for this is an encoding of link weights, which are translated to a routing by shortest path algorithms. In fact, a VTD encoding combined with subsequent shortest path routing can also be interpreted as a TDR solution approach and would consequently also fall into the present category. Such encodings are typically much more scalable, but lack locality and solution space coverage with respect to the original degrees of freedom in routing. A compromise between these two categories are the path enumeration encodings, which can be considered to trade locality and search space coverage for tractability by varying the number of precomputed paths.

### 4.3.2 Compact Path Enumeration Encoding – RCPE

Path enumeration encodings are a very common approach used in both, ILP and GA, as explained in Section 3.6.2, and the suggested Compact Path Enumeration Encoding also belongs to this category. For such approaches, a number of paths is precomputed for the required pairs of sources and destinations and the optimization algorithms select among these paths. The precomputation is often based on k-shortest path algorithms with problem-specific metrics, which may incorporate aspects such as the number of hops on a route, the costs or the lengths of the constituent links. Including hops and length generally makes sense for QoS-enabled demands, since longer paths tend to use more resources and incur higher QoS penalties in terms of delay and availability.

Ideally, the selected paths should reflect the most promising candidates for the entire routing solution, but this is difficult to predict a priori. Since the solution quality emerges as a combination of several paths, a path candidate may be essential for a single very good solution, but detrimental to most other solutions. Consider, e. g., the grooming of two demands in a resource minimization, where one takes a shortest path and the other a very long path, but this specific long path allows the re-use of the short paths' resources. Viewed in isolation, the long path makes little sense, but in this combination that allows for grooming, it's more efficient than to use two shortest paths with dedicated resources on each.

While there is no simple metric to measure grooming potential, this observation still highlights two prerequisites. The first is, that considering shortest paths also makes sense when several paths are involved, and the second aspect is, that the selection needs to contain sufficiently long paths. When a very sparse demand matrix is present and the optimal routing may result in a spanning tree, the paths may have to be very long to reflect such a topology. However, when a full demand matrix is used and many QoS-enabled demands are present, the likelihood that long paths are needed decreases, since the QoS demands require resources on short paths anyway, which can be reused by the other demands. However, most real-world problems do not present with such straightforward structures, such that determining, which path length is sufficient, is very difficult. While complete enumerations of all paths are possible in theory, for most larger scenarios it is not practical to consider arbitrarily long paths.

The number of enumerated paths should therefore neither result in a drastic restriction on the diversity of representable paths, nor the representation of a large number of excessively long paths. To allow for a meaningful exploration of the search space, we therefore suggest scaling the number of precomputed paths for each node pair $\langle s,t \rangle$ in a graph $g$ according to a power law related to the number of hops on the shortest path. This accounts for situations where, e. g., the second-shortest path between two immediate neighbors in the topology will typically already result in a relatively long path of dubious utility. Pairs with shortest paths of many hops, on the other hand, will have a large number of paths of similar length, such that a larger number of paths should be considered for enumeration. In RCPE we determine the number of paths according to Equation (4.3), where $k_\mathrm{m}$ is a parameter providing an upper bound to keep the calculations feasible for very large graphs, while $a$ and $b$ are adjustable parameters, where $b$ should be chosen above the average nodal degree. The function $f_\mathrm{Hops}(g,s,t) : \langle V,E \rangle \times V \times V \to \mathbb{N}_0$ returns the number of hops on the shortest path.

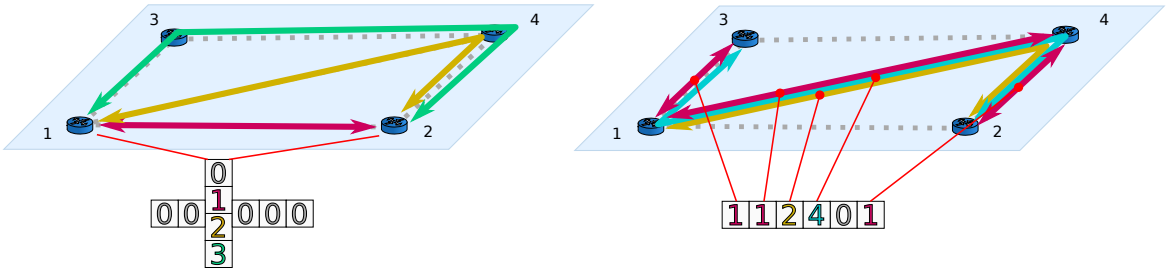
$$f_k(g,s,t,a,b,k_\mathrm{m}) = \min(k_\mathrm{m}, a \cdot b^{f_\mathrm{Hops}(g,s,t)}) \tag{4.3}$$

The effects of a hop-based path enumeration are especially important in the multi-layer network scenario. Whenever short physical links occur in combination with a large transparent reach, the feasible graph may contain a large number of single-hop paths, since the feasible virtual topology can be very dense. In the case of multi-layer networks, we therefore suggest to use the least number of physical hops a virtual link requires as the metric for scaling the number of shortest paths in a virtual topology, i. e., we choose $g$ in Equation (4.3) to represent the physical topology.

For RCPE we use only one gene corresponding to a single selected path for each pair of nodes, such that all demands between a given node pair are routed on the same path. This makes sense, since the demands with same source and destination are an obvious choice for resource sharing, but may lead to situations, in which the selected path may not meet the QoS requirements of all these demands. For this case, we include a fallback mode to divert each individual demand of violated QoS parameters from the selected path to a compliant path. We do this in two steps, following the routing of all compliant traffic on the selected routes according to the gene values.

In the first step, a necessary lower bound on the virtual topology is extracted from the routing performed up to this point. More formally, the union of all edges used in any of the chosen paths according to the chromosome, are used to form a set of links $E_t$, which is used to create a temporary graph $g_t = \langle V,E_t \rangle$. The demands, which had previously violated their QoS contract, are now rerouted on this virtual topology graph using a regular QoS-aware shortest path algorithm based on the same metric as the precomputed paths. This allows for a maximum of resource sharing with the existing routing and results in relatively small diversion from the solution candidate corresponding to the chromosome.

However, this cannot generally result in a QoS-compliant routing for all cases since the temporary topology might still lack essential links. This may, e. g., be caused by a scenario, where there are only very few QoS-enabled demands, such that the majority of best-effort demands suggest a very sparse topology. For this reason, there is a second step, which activates additional links to guarantee a compliant route for those demands, where the first step did not succeed. It simply chooses the shortest route, known to be compliant, which can be identified in the original precomputation. This is more invasive, since it may require a larger number of links to be activated, thereby reducing locality. However, it is computationally very simple, since it can be performed using a lookup table with the index to the original enumeration. In this way,




(a) Alleles and phenotypes for the 3rd gene (b) Chromosome for 6 bidirectional demands

Figure 4.6: Examples for the RCPE encoding

best effort traffic can still be groomed, while as many demands as possible remain aggregated and fulfill their QoS requirements.

#### 4.3.2.1 *Chromosome Structure*

The chromosome represents the primary paths used between all possible node pairs in the graph. Therefore, the length of the chromosome corresponds to the number of node pairs, such that any given node pair can be assigned to one specific locus. The individual values of these genes are an index to the path enumeration. Furthermore, an additional value can be included in the index, which indicates that no demands should be routed between the respective nodes.

Figure 4.6a shows an example for different values for the third gene, which is assigned to the paths between node "1" and node "2" in a symmetric routing scenario. The value 0 is used to indicate that no routing is performed, while the values 1 through 3 indicate all possible paths in order of increasing weight according to the metric. Ordering the index in this way increases locality, since closer indices are more likely to be similar than two paths of distant index values. A full solution candidate with $k_\mathrm{m} \geq 4$ is shown in Figure 4.6b, where the minimum amount of links is used, forming a spanning tree. The demands between nodes "3" and "4" remain unrouted as the respective gene is set to 0. Note, that the fourth gene has a value of 4, since there are 4 possible paths between nodes "3" and "2", highlighting that the range of values can vary for different loci.

The state space for this encoding reaches a size of at most $(k_\mathrm{m}+1)^{|V|^2-|V|}$. Although $k_\mathrm{m}$ is a constant, it should be noted that it has to be chosen carefully in order to maintain a tractable state space, while at the same time it should be selected large enough to match the given graph's structure in order to provide a meaningful number of alternative paths. Following Equation (3.2), the number of possible paths between two nodes in a full graph increases on the order of $|V|!$ such that $k_\mathrm{m}$ has to be scaled drastically to retain coverage. For the extreme case of full coverage in an asymmetric scenario the combinatorial complexity increases according to Equation (4.4).

$$\prod_{s\in V}\prod_{t\in V\setminus\{s\}}(f_k(g,s,t,a,b,k_\mathrm{m})+1) \leq (k_\mathrm{m}+1)^{|V|^2-|V|} \in \mathcal{O}\left(|V|!^{|V|^2}\right) \tag{4.4}$$

#### 4.3.2.2 *Advantages and Disadvantages*

The achievable solution quality of the RCPE encoding depends heavily on the correct choice of $a$, $b$ and $k_\mathrm{m}$. If the number of paths is too small, then routing via long paths, potentially allowing for grooming, may be impossible and therefore it may be impossible to find the global

optimum. This is especially problematic for large graphs with many best-effort demands of very low data rates, where grooming could drastically reduce the required resources. However, if there is an excessive number of precomputed paths, such that any optimal solution requires only much smaller values, then the rate of convergence can be significantly reduced due to the unnecessarily increased combinatorial complexity. This occurs especially when the majority of demands is of high volume and located between nodes in close proximity to each other, as they warrant the usage of dedicated resources on their own.

Finding sufficiently large numbers of paths can also be problematic due to the limited scalability of the routing approaches used in precomputing. As explained in the previous section, $k_\mathrm{m}$ would ideally have to be drastically increased for larger graphs while simultaneously the runtime of the k-shortest path algorithm increases exponentially with graph size as well. This effect is much exacerbated in multi-layer scenarios with flexrate TXPs of large transparent reaches, since they result in very dense feasible virtual topologies where a large amount of the $k$ shortest paths will not diverge significantly from the shortest path anymore, reducing grooming potential even further.

Finally, the RCPE encoding provides only one selectable set of primary paths for all node pairs. The QoS-compliant fallback search does not permit for longer paths, very close to the QoS limits and therefore the grooming potential among QoS-enabled demands is reduced. The advantage on the other side is, that a larger number of QoS classes does not result in an increase in combinatorial complexity. Furthermore, in such situations, where QoS parameters are not close to the limits of what the network can provide, routing the demands on the same path may inherently lead to meaningful grooming between QoS-enabled and best-effort demands.

Another advantage of this approach is the relative simplicity of the decoding function. While computing the required paths between all node pairs for large values of $k_\mathrm{m}$ is computationally expensive, it can be done in a preprocessing step that will typically require only a fraction of the runtime of the GA itself. The primary path will most likely remain compliant for all demands, even for small values of $k_\mathrm{m}$, since shorter paths provide better values for most relevant QoS figures. Furthermore, very long paths are not overly likely to be part of good solutions in many real-world cases and therefore the number of times the shortest path algorithm is invoked, is typically small compared to the number of all demands, such that decoding a RCPE chromosome features a reduced runtime compared to a virtual topology encoding which is followed by routing operations for each demand.

Heritability is generally good, since any allele of any locus will result in the same phenotype for all possible chromosomes. Locality is typically good, but it is dependent on the graph structure. An increase in a gene value will lead to a longer path and paths of increasing length are likely to consume more resources. If the index of a gene is so large that fallback routing is required, the resulting changes are even larger. Locality with respect to a multi-layer scenario, however, is better than for the VTB encoding, since paths can be changed individually, where changing a virtual link can affect an arbitrarily large number of paths simultaneously.

#### *4.3.2.3 Evolutionary Operators*

Since the genes are integer-valued, all regular operators usable on integer vectors described in Chapter 3 can be used in mutation and recombination.

***Mutation***

The Creep-Mutation operator is immediately applicable to the problem without any further adaptions but may be inefficient for large numbers of path candidates. Small step sizes require a large number of mutations to cover the range of possible values, while large maximum step sizes are more likely to miss small refining steps. The Gaussian mutation operator seems better suited for this task, since it inherently provides a meaningful tradeoff between exploration and refinement by randomly sampling step sizes from a normal distribution centered around the previous value.

Since grooming often occurs in combination of one or more short demand routes together with a long demand route as shown in Figure 4.6, a corresponding chromosome is expected to have a similar distribution between small and large values. This suggests that an efficient mutation for such chromosomes can explore values with large strides, but at the same time also refine close candidates by taking small steps. A straightforward way to achieve this behavior, as used in a number of works [104, 145, 288], is to combine two mutation operators of different step sizes. Alternatively, the step size may also be scaled according to a customized distribution. Our approach for integer-valued genes, which we call Longhorn mutation, falls in the latter category and will be detailed in Section 4.4.1.

***Recombination***

Since all genes are integer-valued, all non-reordering regular operators outlined in Chapter 3 can be used. Since each gene represents a single path and since overlapping paths present grooming opportunities, it can be expected that there are combinations of genes that contribute significantly to an improvement in the objective value. This suggests that a bias toward the more fit parent, which is scaled according to the difference in objective value, may be a good choice to proliferate beneficial combinations. We have developed a simple approach for such a recombination operator, which will be elaborated upon in Section 4.4.2.

#### *4.3.2.4 Population Initialization*

Every possible chromosome of the RCPE encoding is guaranteed to result in a valid network configuration without any further enhancement. However, the argument from above, that good solution candidates will most likely combine a larger number of short path indices with a smaller number of longer path indices, is also noteworthy in the initialization phase. To provide initial chromosomes with this property, we have developed a procedure called Randomized Dual-Range Initialization (RDRI). For each locus it determines, if the gene value should be limited to a uniform selection from either a low interval or a high interval of values. If the gene is selected to be neither, it is set to its shortest path. To this end, it has four parameters $c_\mathrm{L}$, $c_\mathrm{H}$, $p_\mathrm{L}$ and $p_\mathrm{H}$ that define the gene value limits and probabilities as follows. The low interval is $[1, c_\mathrm{L}]$ and the high interval is $]c_\mathrm{L}, \min(c_\mathrm{H}, c_\mathrm{max})]$, where $c_\mathrm{max}$ represents the largest possible gene value for the current locus. The probabilities are given by $p_\mathrm{L}$ and $p_\mathrm{H}$ and the remaining loci, are set to the shortest path with a probability of $1 - p_\mathrm{L} - p_\mathrm{H}$. A more detailed description of the entire procedure is given in Algorithm 10.

An alternate initialization approach with the goal of maximizing grooming potential, is to use an adapted version of APTI or ASTI approaches from Section 4.2.2.4 to create basic topologies and then select the fitting path indices, which result in these spanning trees. While individuals generated in this way may not be particularly useful on their own, they can contribute building blocks for efficient grooming to other individuals.

**Algorithm 10** Path Enumeration Encoding – Randomized Dual-Range Initialization

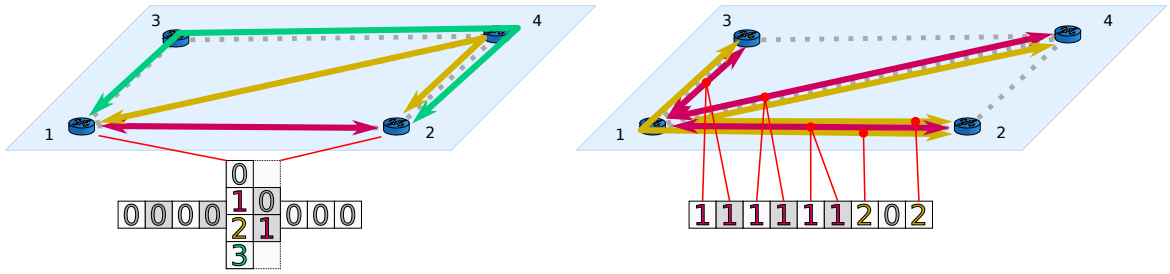
**Require:** *locusToRangeArr* is an array containing the maximum gene value for each locus
**Require:** $c_\mathrm{L}$ is the limit of the lower interval with $c_\mathrm{L} > 1$
**Require:** $c_\mathrm{H}$ is the limit of the higher interval with $c_\mathrm{H} > c_\mathrm{L}$
**Require:** $p_\mathrm{L}$ is the probability for the lower interval
**Require:** $p_\mathrm{H}$ is the probability for the higher interval with $p_\mathrm{L} + p_\mathrm{H} \leq 1$

```
function CreateRDRIChromosome(locusToRangeArr, c_L, c_H, p_L, p_H)
    let n_loci ← max(I) with I = {i | ⟨i, j⟩ ∈ locusToRangeArr}        ▷ Array Length
    let geneArr ← ⟨i, 0⟩ with i ∈ 1, ··· , n_loci
    let r ← 0
    for i_gene = 1 to i_gene = n_loci do
        r ← RND(0, 1)
        if r ≤ p_H then
            if c_L < locusToRangeArr[i_gene] then
                geneArr[i_gene] ← RNDI(min(c_H + 1, locusToRangeArr[i_gene]) − c_L) + c_L
            else
                geneArr[i_gene] ← RNDI(locusToRangeArr[i_gene])
            end if
        else if r − p_H ≤ p_L then
            if p_L > 1 then
                geneArr[i_gene] ← RNDI(1, min(c_L, locusToRangeArr[i_gene])) + 1
            else
                geneArr[i_gene] ← 1        ▷ Assuming that 1 represents the shortest path
            end if
        else
            geneArr[i_gene] ← 1            ▷ Assuming that 1 represents the shortest path
        end if
        if geneArr[i_gene] ≥ locusToRangeArr[i_gene] then
            geneArr[i_gene] ← locusToRangeArr[i_gene] − 1
        end if
    end for
    return geneArr
end function
```

Finally, a typically good, but not optimal candidate can very easily be generated by choosing all indices corresponding to the shortest paths. Since the shortest paths are always included when using this encoding, the results of the overall genetic approach can never be worse than a regular shortest path routing, if elitism is used to ensure that the best solution is kept in the population.

### 4.3.3 Demand-Diverse Compact Path Enumeration Encoding – DCPE

DCPE is very similar to RCPE, but rather than enumerating multiple paths for each node pair, it enumerates paths for each demand separately. This allows for a more fine-grained routing, since using a longer path for best-effort traffic does not have any potentially negative impact on the QoS-traffic demands. Furthermore, it is also possible for demands of different QoS classes to be routed on separate paths. This also enables grooming different QoS classes in different ways, e. g., in situations where a shortest path would use a very unreliable link, such that latency-, but




(a) Alleles and phenotypes for genes 5 and 6

(b) Chromosome for 9 bidirectional demands

Figure 4.7: Examples for the DCPE encoding

not availability-critical demands can use this route, while the others use a longer path on more reliable links.

The path selection for this encoding uses almost the same path reduction approach as RCPE by scaling the number of enumerated paths according to Equation (4.3), but it does not need any of the QoS fallback routing steps. Rather, for each QoS-enabled demand the paths are checked for compliance in precomputing and can even be removed from the enumeration in order to further reduce the search space to meaningful solutions. Again, determining the paths is part of the precomputation and done once for every node pair, between which traffic demands exist.

#### *4.3.3.1 Chromosome Structure*

For this encoding, the chromosome length scales with the number of traffic demands. Depending on the number of QoS classes and source–destination pairs, this chromosome may be longer than that of the previous approach by a factor of the number of QoS classes, but if the traffic demand matrix is sparse, it may also be shorter. The example in Figure 4.7a, again a symmetric routing scenario, has three additional demands between node "1" and the other nodes that require lowest latencies and are therefore represented as slightly shaded. The fifth gene is the best effort class towards node "2" where paths of all lengths are admissible, but the sixth gene, representing the lowest latency class between the same node pair, only allows for the shortest path showcasing the option to remove non-compliant paths to reduce complexity. Figure 4.7b shows a solution candidate where a spanning tree connects node "1" to all others since the low-latency demands require all of these shortest paths offering a grooming possibility to the other best-effort demands.

For transport networks, where full demand matrices are common, the resulting search space complexity will typically be much higher than for RCPE. Given a traffic demand as a tuple $d = \langle s,t,r,q \rangle$ of its source node $s \in V$, destination node $t \in V$, data rate $r \in \mathbb{R}_+$ and QoS class $q \in Q$ and the set of all demands $D$, assuming that all paths are QoS-compliant, the resulting combinatorial complexity can be described by Equation (4.5). This appears quite similar to Equation (4.4), but replaces the multiplication over source and destination nodes by multiplication over the number of demands, which in the limit becomes $|V|^2 \cdot |Q|$ due to the addition of the QoS classes[1]. Note, that $|Q|$ does not scale depending on the graph structure, but is an immediate result of SLA design. Because NSPs tend to keep their portfolios streamlined

[1]While we only consider exactly one $r$ per tuple of node pair and QoS class due to the aggregation of traffic at the core-level, it is absolutely possible to consider several data rates as well. For typical core networks this can be expected to be a relatively small set following the same arguments as for $Q$.

as outlined in Section 2.1.1.1, it can be expected to contain only a relatively small and constant number of elements.

$$\prod_{\langle s,t,r,q\rangle \in D} (f_k(g,s,t,a,b,k_\mathrm{m})+1) \leq (k_\mathrm{m}+1)^{|Q|\cdot|V|\cdot(|V|-1)} \in \mathcal{O}\left(|V|!^{|V|^2}\right) \tag{4.5}$$

It is to be expected, that traffic demands with QoS restrictions will most likely be routed on shorter paths, while best-effort traffic may exhibit much larger path index numbers. When considering availability, however, the choice depends heavily on the underlying model. If the decisive component is not edge length, but rather another figure, e. g., a short path may use much less reliable aerial fiber, while a longer path may use buried fiber, this may lead to increased path index numbers. Due to such effects, given a sufficiently large $k_\mathrm{m}$ and restrictive QoS parameters, the range of gene values can vary drastically between subsequent genes in the chromosome.

#### *4.3.3.2 Advantages and Disadvantages*

Many of the advantages and disadvantages of this encoding are identical to those of the RCPE encoding. The concerns regarding the choice of $b$ and $k_\mathrm{m}$, as well as the remarks on heritability are largely identical. Decoding efficiency is expected to be better for DCPE since it does not require any fallback routing. The main advantage of this encoding is that it can offer grooming for any combinations of demands. However, if there is no meaningful grooming potential among QoS-enabled demands, it cannot offer a significantly better solution quality than the previous approach. In fact, for the case of a full traffic demand matrix without any QoS-enabled traffic RCPE and DCPE become identical. If QoS demands are present, but the resulting problem is inherently devoid of any grooming potential for these demands, then using DCPE will most likely only result in severely degraded performance, since the search space is significantly larger without containing any better solutions, compared to RCPE.

A second advantage, albeit one less relevant to transport networks, appears in case of resources showing conflicting and considerable non-uniform impacts on different QoS metrics, such as an aerial fiber of low availability, but also low delay. Since DCPE is able to explicitly address such situations, it is able to provide better results than other approaches. While the fallback-option of RCPE can mitigate the effects as well, it is restricted to the shortest compliant path, which may lead to non-optimal solutions. Locality can be considered to be slightly improved compared to RCPE, since a single change in a gene value will have an even smaller effect on the overall routing, as the average amount of traffic being diverted becomes slightly smaller in most cases.

The biggest drawback of DCPE, however, is its typically much larger search space. While in fact both, DCPE and RCPE have the same asymptotic worst-case time complexity class, the average search space will increase by an exponent of $|Q|$. Even though this value can be expected to be smaller than 10, the increase is large enough, to drastically reduce tractability, such that this approach does not scale easily to large networks.

#### *4.3.3.3 Evolutionary Operators*

Given DCPE's similarity to RCPE, both approaches can use the same integer-compatible operators for the same reasons. Therefore, further explanations in this chapter will mostly focus on aspects, where a difference between the two emerge.

#### *Mutation*

While Gaussian and Creep mutation operators are applicable, the Longhorn mutation operator, detailed in Section 4.4.1, is expected to yield especially good results for this encoding. This operator was specifically designed to facilitate grooming, which is the primary driver behind the present encoding. A fast and efficient exploration of alternative paths is all the more important given the increased search space.

Beyond that, the mutation operators are required to support vastly different ranges of values. Since $k_{\mathrm{m}}$ may be chosen to be very large, but the number of contained QoS-compliant paths may be small, the mutation operators will have to support large gene value ranges as well as small ranges within the same chromosome. While differing ranges may always exist for any path enumeration encoding in non-complete graphs due to shortest paths of different lengths, QoS requirements may drastically reduce the number of available paths.

There are several ways how to accommodate this property in a mutation operator. While manual tuning of parameters for each gene is possible, it will typically not be feasible, as any combination of QoS requirements and graph properties may result in entirely different ranges. It is suggested to either automatically scale the parameters or to provide a meaningful wrap-around function.

#### *Recombination*

Operators applicable to RCPE can just as well be used for DCPE. As the assignment of demands to loci is fixed, any recombination that maintains this assignment will result in valid offspring, such that the differences in range are not important to such recombination operators.

### *4.3.3.4 Population Initialization*

All the initialization options outlined for RCPE can also be applied to DCPE. Regarding RDRI, it should be noted that the differing gene value ranges can make the choice of parameters more difficult. While this can motivate an extension, where the limits can be scaled, e. g., to represent a proportional selection from the range of available values, we found that Algorithm 10 remains effective for the investigated range of problems. This is due to the fact that for loci of drastically restricted ranges, only little room for different choices is possible, such that any intended distribution between high and low ranges cannot be established in a meaningful way.

Another seemingly obvious choice for an initialization routine would be to exploit the semantic context of genes in the DCPE encoding. Since demands between the same source and destination nodes, can potentially share the same route, this can create grooming opportunities. However, it cannot be generally stated, that explicitly treating such genes identically will present an advantage. Furthermore, this is not always possible given different QoS requirements and link properties.

Rather than explicitly identifying matching path values, we implicitly exploit this opportunity through the adapted ASTI and APTI initialization procedures. In this way, we can not only provide grooming potential for demands between the same nodes, but also to demands of different node pairs. A straightforward example for this are demands on short paths, where these paths are also part of longer paths between other nodes.

## 4.4 Genetic Operator Adaptations

While most encoding approaches have been designed with compatibility to standard operators in mind, more specific and problem-oriented operators have the potential to accelerate the search. To this end, we have developed two further operators, which can be expected to fit the explained problem structures better than the standard operators, while still being sufficiently generic to be applicable to other problems and GA variations as well. The first is a biased recombination operator based on URM, while the second is a mutation operator which combines several step sizes.

### 4.4.1 Longhorn Mutation – LHM

We have developed the so-called Longhorn operator to address a number of shortfalls of common mutation approaches, which seek to efficiently combine refining and exploratory changes to a chromosome. Such approaches can be especially useful for combinatorial problems of large search spaces. This includes path enumeration encodings for problems, where grooming is important to achieve the optimum, such that we expect the corresponding chromosome to feature a combination of many small and few large path index values es explained in Section 4.3.2.3 and Section 4.3.3.3. Therefore, a mutation operator that can provide a combination of small steps for fine-grain adjustment and large step sizes for exploratory mutations, may yield better performance than settling for a single intermediate step size.

The balance between refinement and exploration can be controlled by deterministic or probabilistic means. E. g., a deterministic approach might set a ratio, such that a fixed number of small steps and a single large step are taken in alternation. A probabilistic one might define a probability $p$ for small steps, such that large steps are taken with probability $1 - p$ and draw a random number upon each decision. As already mentioned in Section 4.3.2.3, a straightforward combination of two standard mutations with different step sizes is used in a number of works to achieve the aforementioned properties. As such, versions of both, the Creep [54, p. 263][104] and Gaussian [145] operators, have been developed that consist of two separate mutations with different step sizes or standard deviations, respectively. We will refer to them as Small/Large Creep Mutation (SLCM) and Small/Large Gaussian Mutation (SLGM) operators.

However, both of these approaches have their drawbacks. SLCM is computationally very efficient but allows only for uniform choices within the small and large step sizes. Since SLGM uses the Gaussian normal distribution, it can scale priorities through the standard deviations, but it is incapable of combining a slightly larger probability for very large steps with a very large probability of very small steps. Beyond that, SLGM is computationally more complex than SLCM, since there is no closed-form description of the quantile function to the normal distribution, such that a generating approach needs to resort to other means such as series expansions of the Gauss error function or the Box–Muller transform [28].

We therefore envisage a mutation operator that combines the efficiency of SLCM with a more flexible probability distribution-based approach similar to SLGM. Our approach uses a small number of low-order polynomials to create a simple generating function from which a distribution of values can be sampled. This distribution can be customized with few parameters and provides high probabilities for small increments and low probabilities for large increments, while within the large increments, the low and high ends receive slightly increased probabilities as well.

We define this generating function as

$$f_{\mathrm{LH}}(x,p,v_l,v_h) = \begin{cases} -a\cdot x^3 + b\cdot x^2 + c\cdot x + d & \text{if } -1 \le x < -p \\ \frac{v_l}{p^2}\cdot x^2 & \text{if } -p \le x \le p \\ a\cdot x^3 + b\cdot x^2 - c\cdot x + d & \text{if } \quad p < x \le 1 \end{cases} \tag{4.6}$$

with

$$\begin{aligned} a &= \frac{v_l - v_h}{\alpha + \beta\cdot\gamma} \\ b &= \gamma\cdot a \\ c &= 3\cdot a + 2\cdot b \\ d &= v_h - a - b - c \\ \alpha &= p^3 - 3\cdot p + 2 \\ \beta &= p^2 - 2\cdot p + 1 \\ \gamma &= \frac{3}{2}\cdot\frac{p^2-1}{1-p} \end{aligned}$$

and

$$\begin{aligned} x &\in [-1,1] \\ p &\in (0,1) \\ v_l, v_h &\in \mathbb{R}_+ \\ v_h &> v_l \end{aligned}$$

such that $f_{\mathrm{LH}}$ is continuous in [-1,1], albeit not differentiable for $x = \pm p$. The value $p$ controls the balance between smaller increments, which are derived by the second-order polynomial, and larger increments, which are determined by the third-order polynomials. The parameter $v_l$ marks the upper limit for the small increment size, while $v_h$ bounds the large increments from above. A single random variable $x$ with a uniform distribution in $[-1,1]$ can then be used to create increments following the intended distribution.

In our testing phase we found that, while the choice of these parameters depends heavily on the problem at hand, we still could narrow down the ranges for parameter combinations such that reasonably good results can be expected. We suggest selecting $p$ from $[0.6,1)$ to allow for a meaningful tradeoff. The parameter $v_l$ should be chosen at least as large as the smallest meaningful value that leads to a change in the chromosome, but not much larger than 10 times this value. Finally, we chose $v_h$, such that it is at least 10 times larger than $v_l$ and about 5 % to 25 % of the total value range for the respective gene.

Figure 4.8 shows a visualization of this function, which we call Longhorn function since the shape of its graph resembles the horn shape of Longhorn breeds of cattle. The right half-plane of the graph corresponds to the quantile function of the intended probability distribution. In this example the parameters are $p = 0.8$, $v_l = 5$ and $v_h = 100$. Intervals $I_2$ and $I_3$ result in small increments for refining steps, whereas $I_1$ and $I_4$ result in large increments for exploratory step sizes. In the latter intervals, the function is tapered towards $x = \pm 1$ and $x = \pm p$, such that these increments have higher chances, than those in the middle of the intervals. We consider this property helpful, since larger values are good for exploration and lower values for refinement, but intermediate values are neither the first, nor the latter.

The LHM operator around this function now simply generates $x \in \mathcal{U}(-1,1)$ and uses the Longhorn function to determine the increment. The real-valued increment is then rounded to

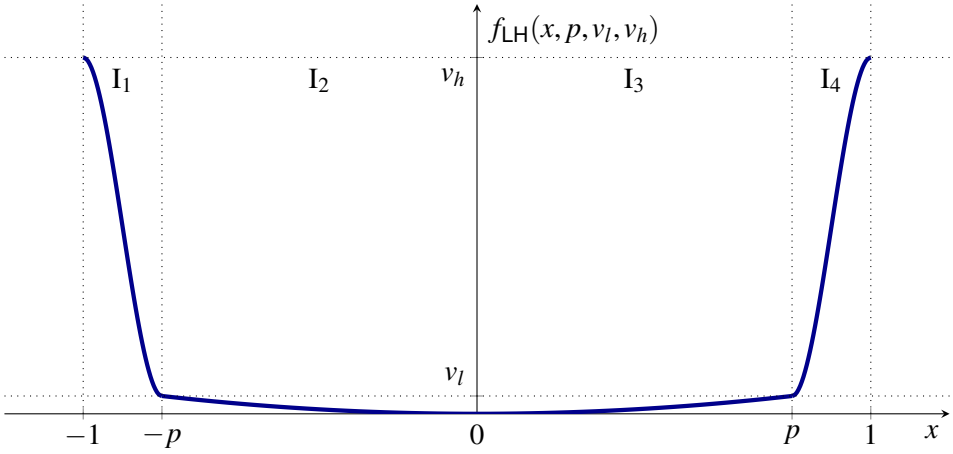



Figure 4.8: Visualization of the Longhorn function

obtain the actual step size, while ensuring that it is non-zero. The step size may have to be wrapped if it exceeds the maximum value for the gene being mutated. To slightly improve efficiency, all constants used in the Longhorn function can be precomputed.

### 4.4.2 Exponentially Biased Crossover – EBXO

An interesting aspect in accelerating the performance of genetic algorithms is the introduction of bias. This can be especially helpful in large search spaces, where otherwise too much time might be dedicated to exploring regions of generally lower solution quality. However, applying too much bias quickly reduces genetic diversity and may overly restrict the exploration to smaller areas of the search space, potentially missing optima in regions of vastly different individuals. In Section 3.4.5.2 we have already touched on the subject of explicitly marking and maintaining elite individuals, i. e., such of above-average fitness, in populations and in Section 3.4.5.5 we have introduced the concept of biased recombination operators.

Elite individuals or sub-populations are an effective way of providing a relatively small bias without significantly interfering with the GA's other mechanisms. Biased recombination, on the other hand, is a more aggressive approach, because it affects all recombinations and not just those, where elite individuals are concerned. If the bias is too strong, many sub-par individuals will be evicted rather quickly from the population threatening diversity. A very effective compromise is presented by the BRKGA approach, where every mutation is between an elite and a non-elite individual, which have equal chances of passing on their genes, combining diversity with known good solutions.

The drawback is that the standard BRKGA cannot immediately combine two elite individuals, which may not always be beneficial in complex combinatorial problems. E. g., in cases, where it can take many attempts to create a non-elite individual, that can be recombined with an elite individual of orthogonal traits, the performance may be decreased. This is especially troublesome, when the problem at hand is so excessively complex, that the GA will most likely encounter only local optima anyway, because the chances of finding the global optimum are very slim.

We therefore suggest a version of the BUXO operator that is adaptive. While approaches exist that scale the hyperparameter $p$ over the runtime of the GA [189], it cannot be guaranteed, that the runtime is matched well to the exploration of the search space due to the inherent randomness of the GA approach. Our approach therefore compares the cost values of the two

individuals and decides based on their relative difference, at which rate genes are passed on. Similar approaches already exist, e. g., [248], but they often rely on more complex evaluations.

The suggested Exponentially Biased Crossover (EBXO) is of very low complexity and therefore does not impede overall computational performance. It uses an exponential weighting of the relative cost difference between the parent individuals according to Equation (4.7).

$$f_{\mathrm{EB}}(c_A, c_B, w, p_l) = \min\left(0.5 + 0.5 \cdot \left(1 - \frac{c_A}{c_B}\right)^w, p_l\right) \tag{4.7}$$

In this equation, $c_A$ is the cost value of the more fit parent and $c_B$ the cost of the other parent. Among the hyperparameters, the weight $w$ scales the bias and the probability limit $p_l$ places an upper bound on the resulting value. This limit prevents the operator from simply cloning very fit individuals in case of vastly different parent cost values. The exponential weighting means that the genes of equally fit individuals have equal chances to be passed on, while more genes are inherited from the more fit parent, the larger the cost difference is.

We also included this weighting approach in our VSXO and LBXO operators, such that exponential bias can be introduced if an activation parameter is set. For LBXO, the number of selected nodes is scaled based on the EBXO approach. For VSXO, the probability according to Equation (4.7) is first used to determine, if both offspring should be using the same spanning tree of the more fit parent or if the offspring retain the trees of both parents. Following this, the EBXO approach is applied for all the remaining genes, which are not part of the spanning trees.

## 4.5 Integrated Solver

As outlined in Section 2.2, multi-layered networks combine topology design and routing at each layer, such that the routing of the lower layer contributes to the topology of the upper layer. An integrated solver for multi-layer problems therefore provides a holistic solution encompassing all of these aspects. However, solving this as a single monolithic problem formulation is suggested to be intractable for real-world problem instances, such that integrated solvers typically work by separating sub-problems. In this section, we describe how the suggested algorithm adaptions can be implemented as components of an integrated solving approach for multi-layer networks.

### 4.5.1 Sub-Problem Focus

In Section 3.6.3 various approaches have been introduced, which solve these problems by addressing the layer coupling in different ways. Rather than exposing the full complexity of all decisions to an optimization algorithm, sub-problems have been separated and solved independently for reasons of tractability, as explained in Section 3.2.2. To further improve tractability of large-scale problems, integrated solvers typically only optimize a single sub-problem and then use its results as inputs to other more simple algorithms to obtain solutions for the remaining sub-problems. The resulting partial solutions are then assembled into a full solution.

While this is often the only way to address very complex and large problems, the drawback is that the objective value of this full solution may not be optimal for the original multi-layer problem. This is due to the fact that the sub-problems are typically not separable in such a way that the combination of their individual optima leads to a global optimum of the entire multi-layer problem as has been explained in Section 3.2.2. It is therefore imperative to select a sub-problem for optimization, which has the largest influence on the overall objective value, while still being tractable, such that the margin of deviation for the remaining heuristics is as small as possible.

Following the design space outlined in Section 4.1, we assume the infrastructure to be fixed in terms of locations and feasible links, thus leaving four major sub-problems in the form of TDR, VTD, RWA (or ICR with WA) and PTD. Furthermore, we are primarily concerned with determining the minimal requirements in terms of line cards and TXPs. Solutions to PTD problems have an immediate effect on physical link costs, but only an indirect effect on the hardware requirements at the nodes, which is directly related to the amount of traffic that terminates or passes through a node.

While RWA and ICR/WA are directly related to the number of TXPs and line cards, they require the amount of point-to-point capacity for traffic as an input, which is typically the result of a previous traffic routing. Consequently, the number of these devices is already defined by the traffic routes and RWA or ICR/WA are mostly concerned with an efficient management of optical resources in terms of optical spectrum and hardware in terms of amplifiers. Therefore, among the aforementioned sub-problems, VTD and TDR have the most immediate effect on the required amount of line cards and TXPs. Selecting one of the upper-layer sub-problems as the center of an integrated approach is therefore expected to yield better results for this purpose, than starting from a lower-layer optimization [77, 129].

TDR requires a virtual topology as input, which can be determined by VTD or assigned statically. When using the full feasible topology, all states representable by any virtual topology are part of the search space of the TDR problem. For this reason, using TDR offers the greatest amount of flexibility and the highest potential for optimization, but also the largest combinatorial complexity. In contrast to this, VTD has a much smaller search space, but requires an additional step to solve the traffic routing. This can be accomplished very efficiently by classic shortest path algorithms as explained in Section 3.5.1.1, but this may restrict the achievable solutions and possibly exclude the actual optimum. The choice between focusing on TDR and VTD is therefore based on a tradeoff between manageable tractability and attainable optimality.

### 4.5.2 Integration of Genetic Algorithm Adaptions

We have developed four different encoding schemes, VTB and VTCS primarily for VTD with possible applications in other topology design problems and two encoding schemes, RCPE and DCPE, geared towards TDR problems, along with several evolutionary operators with varying degrees of problem-specific adaptions. Since an important asset in using GAs is that they present a particularly malleable type of framework, an integrated approach should retain this flexibility and allow for a toolkit-like implementation that enables to mix and match components as required by the problem.

#### *4.5.2.1 Encodings in a Multi-Layer Context*

When considering the developed encodings in the context of solving multi-layer problems, computational complexity and solution space coverage have to be re-evaluated in this context.

VTB and VTCS both determine the virtual topology for the multi-layer problem. An integrated approach based on the virtual topology restricts routing, such that not all possible solutions to the multi-layer problem may be representable, especially when using classic algorithms for traffic routing. Hence, the optimal solution space coverage VTB achieves for the isolated VTD problem translates to a reduced solution space coverage for the full multi-layer problem.

VTB and VTCS require additional path search algorithms for TDR, which significantly impact the computational complexity of an integrated approach. This is especially troublesome

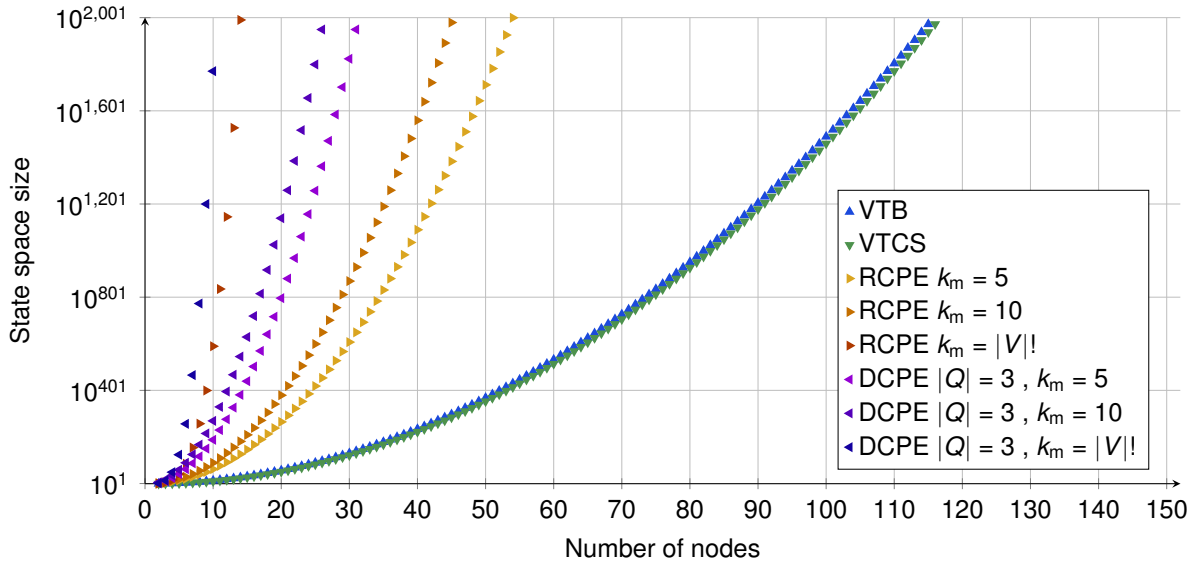



Figure 4.9: State space size for suggested encodings assuming a full graph

for VTB, which on its own only requires inspecting $\mathcal{O}(|V|^2)$ genes for activation, while the subsequent traffic routing has a worst-case time complexity in $\mathcal{O}(|V|^4)$, since the algorithms for point-to-point paths are typically in $\mathcal{O}(|V|^2)$ and need to be run for a potential maximum of $|V|^2$ demands times the number of service classes.

Using TDR as the basis of an integrated approach, does incur less secondary costs by other supporting algorithms, because the virtual topology is implicitly defined by the set of all links used in all paths of all demands. This makes the assembly of a full multi-layer solution computationally more efficient than the VTD-based approaches. The drawback is that a path enumeration-based encoding cannot precompute all possible paths for reasonable problem sizes. To keep combinatorial complexity in check, the number of paths has to be drastically reduced, which detracts from the attainable optimum.

Figure 4.9 illustrates this issue by showing the state space size for different encodings over the number of nodes in a fully connected graph. The state space of the DCPE encoding including all paths for a graph with 10 nodes and 3 QoS classes has about the same size as VTB for a graph of 110 nodes. While limiting $k_m$ to small numbers increases tractability, it is still dramatically lower than for the VTD-focused encodings. It is therefore to be expected that the advantage of the extended search space coverage a path-enumeration encoding can provide, will eventually be nullified by intractability.

Apart from computational complexity and solution space coverage, locality and heritability can also be evaluated in a multi-layer context. As briefly mentioned in the previous sections on the encodings, locality of VTD-based approaches will typically be much lower, than for routing approaches, since a small change in the topology can have a very large effect on many traffic flows and circuit routes. This is closely related to heritability, which in most cases is subject to the same effects as locality. The only property remaining unaffected by the integration is the amount of infeasible representations, since an infeasible solution to VTD or TDR will also be infeasible for the surrounding multi-layer problem.

Table 4.1 provides a qualitative comparison, which summarizes the aforementioned considerations under the assumption that the routing approaches can cover all possible paths. The summary shows that there is not one approach which dominates all others in its prospective

Table 4.1: Qualitative comparison of encodings for multi-layer network optimization

| Aspect | Coding | | | |
|---|---|---|---|---|
| | **VTB** | **VTCS** | **RCPE** | **DCPE** |
| Combinatorial Complexity | + | ++ | − | −− |
| Time Complexity – Initialization | + | ○ | −− | −− |
| Time Complexity – Decoding | − | −− | ○ | + |
| Solution Space Coverage | ○ | − | + | ++ |
| Infeasible Representations | − | ++ | ○ | ++ |
| Locality | ○ | − | + | ++ |
| Heritability | ○ | − | + | + |

Legend
++ Best + Good ○ Intermediate − Bad −− Worst

efficacy regarding multi-layer network problems. For smaller graph sizes or densities, where moderate numbers of enumerated paths lead to very good solutions, routing approaches seem favorable. For large-scale graphs and very dense feasible topologies, VTB or VTCS may remain tractable and yield good solutions. Therefore, a flexible, component-oriented implementation offers the largest potential benefit.

#### *4.5.2.2 Genetic Algorithm Components*

In order to obtain the most amount of flexibility and be able to customize the integrated approach, we have implemented the GA's constituent parts as components. In this way, basic GAs can be configured just as easily as more modern and intricate versions. In addition to our more specific approaches introduced in this chapter, we have also implemented a variety of more generic evolutionary operators and selection strategies. However, while selection is mostly encoding-agnostic, not all of the other components are readily interchangeable, since some are specific to certain encodings and parameter choices. Table 4.2 shows, which of the implemented initialization schemes and genetic operators are applicable to the genetic encodings as presented in this chapter. Combined Augmented Topology Initialization (CATI) refers to a combination of the ASTI and APTI methods. Feasible topology and physical topology initialization means that a single individual corresponding to either the feasible or the physical topology is included. These approaches can be combined with the others and ensure that the end result cannot be worse than what a regular shortest path algorithm would achieve. Note, that the physical topology may not be exactly representable in some rare cases for VTCS and for RCPE and DCPE if the number of paths is too limited.

The distribution initialization is only meaningful for integer-valued chromosomes. It creates a chromosome structure where the sequence of genetic values follows a Gaussian distribution with an expected value uniformly selected among the loci. The distribution is wrapped at the end of the chromosome, and it's always scaled such that the highest point corresponds to the largest value a gene can take at the center locus. Therefore, the standard deviation should be chosen small enough such that the integer-rounded values do not overlap.

For parent selection we have implemented the well-known RWS and SUS as well as uniform and tournament selection as described in Section 3.4.5.2. For survivor selection, we have also

Table 4.2: Implemented initializations and evolutionary operators

| Component \ Coding | | VTB | VTCS | RCPE | DCPE |
|---|---|---|---|---|---|
| Initialization | Random | ✓ | ✓ | ✓ | ✓ |
| | Distribution | X | X | ✓ | ✓ |
| | APTI | ✓ | ✓ | ✓ | ✓ |
| | ASTI | ✓ | ✓ | O | O |
| | CATI | ✓ | ✓ | O | O |
| | RDRI | X | X | ✓ | ✓ |
| | Feasible Topology | ✓ | ✓ | ✓ | ✓ |
| | Physical Topology | ✓ | O | O | O |
| Recombination | UXO | ✓ | ✓ | ✓ | ✓ |
| | CNCXO | ✓ | ✓ | ✓ | ✓ |
| | BUXO | ✓ | ✓ | ✓ | ✓ |
| | EBXO | ✓ | ✓ | ✓ | ✓ |
| | NPXO | ✓ | ✓ | ✓ | ✓ |
| | LBXO | ✓ | ✓ | ✓ | ✓ |
| | VSXO | X | ✓ | X | X |
| Mutation | RRM | ✓ | ✓ | ✓ | ✓ |
| | URM | ✓ | ✓ | ✓ | ✓ |
| | NPM | ✓ | ✓ | ✓ | ✓ |
| | VSM | X | ✓ | X | X |
| | Creep | X | ✓ | ✓ | ✓ |
| | SLCM | X | X | ✓ | ✓ |
| | Gaussian | X | X | ✓ | ✓ |
| | SLGM | X | X | ✓ | ✓ |
| | LHM | X | X | ✓ | ✓ |

Legend
✓ Applicable O Parameter-dependent X Not applicable or not meaningful

considered a number of common approaches. Apart from $(\mu, \lambda)$, round-robin and stochastic tournaments as explained in Section 3.4.5.3, we also included a generation counter system. This can either be used for generational model GAs or as a biasing factor in other selection approaches.

Furthermore, we have also included different standard scaling approaches for the cost values of individuals including proportional and exponential as well as rank-based classification. To increase selection pressure, i. e., emphasize the advantage of good individuals, we can also employ sigma truncation as well as a normalization to the lowest cost value of the current population. We also use a basic elitism approach, where we ensure that the fittest individual of a population always migrates to the following generation.

To maintain diversity, we can also employ a *crowding* approach [56] to population management. Crowding attempts to use a chromosome structure-based similarity measure to monitor which offspring replace which parents. Our approach divides the population into subgroups of chromosomes featuring high similarity values. When the offspring all result in similar chromosomes, crowding admits only the ones with an actual improvement within their subgroup to compete with the remaining population for survival. The group assignment is dynamic and repeated for each generation in order to avoid the situation that the majority of sub-groups are forced to represent groups of excessively bad quality.

All of these individual components can be rearranged and customized in order to achieve the best performance for a given problem. To meet this goal, the two integrated approaches presented in the following sections, start from one of the developed encodings and can utilize all compatible components in a highly flexible manner.

### 4.5.3 Virtual Topology-Based Configuration

The first of the integrated approaches is based on an optimization starting from the virtual topology in a top-down fashion and is derived from a framework developed in prior works of Feller [77, 78, 79] and the author of this monograph [12, 13, 72]. It first solves an uncapacitated VTD problem and uses the resulting virtual links together with the traffic as input to an uncapacitated TDR, which results in an assignment of required capacity to virtual links. This in turn is input to a partly uncapacitated ICR and WA, where the required capacity is broken down into individual circuits, which are routed according to a shortest path metric. From the number and location of circuits, the required number of TXPs can be extracted, which can then be used to assign a sufficient number of upper-layer NEs.

The original framework from prior works mentioned above used Simulated Annealing to solve VTD for reconfiguration problems with a single demand between each node pair, given a physical network structure with unlimited wavelength conversion capability, i. e., wavelength conversions can be provided by dedicated hardware at any node when necessary as explored in Section 2.2.3.2. The extended framework, which has been developed as the basis for the integrated approaches presented in this monograph, retains the Simulated Annealing implementation, which makes it ideal for performance comparisons. While the focus of many developments leading to the new version was on the handling of multiple QoS-diverse demands per node pair, the optical layer has also been subject to further extensions. The presented version includes additional heuristics for flexible TXPs [12], as well as a WA component, augmenting the original resource assignment algorithm given in Feller's work [77, ch. 4.4.1].

The entire Virtual Topology-Based Configuration process is illustrated in Figure 4.10, where blue colors indicate GA-specific functions. While the coding is either VTB or VTCS, the individual functions can be realized by the different components according to Section 4.5.2.2. Green

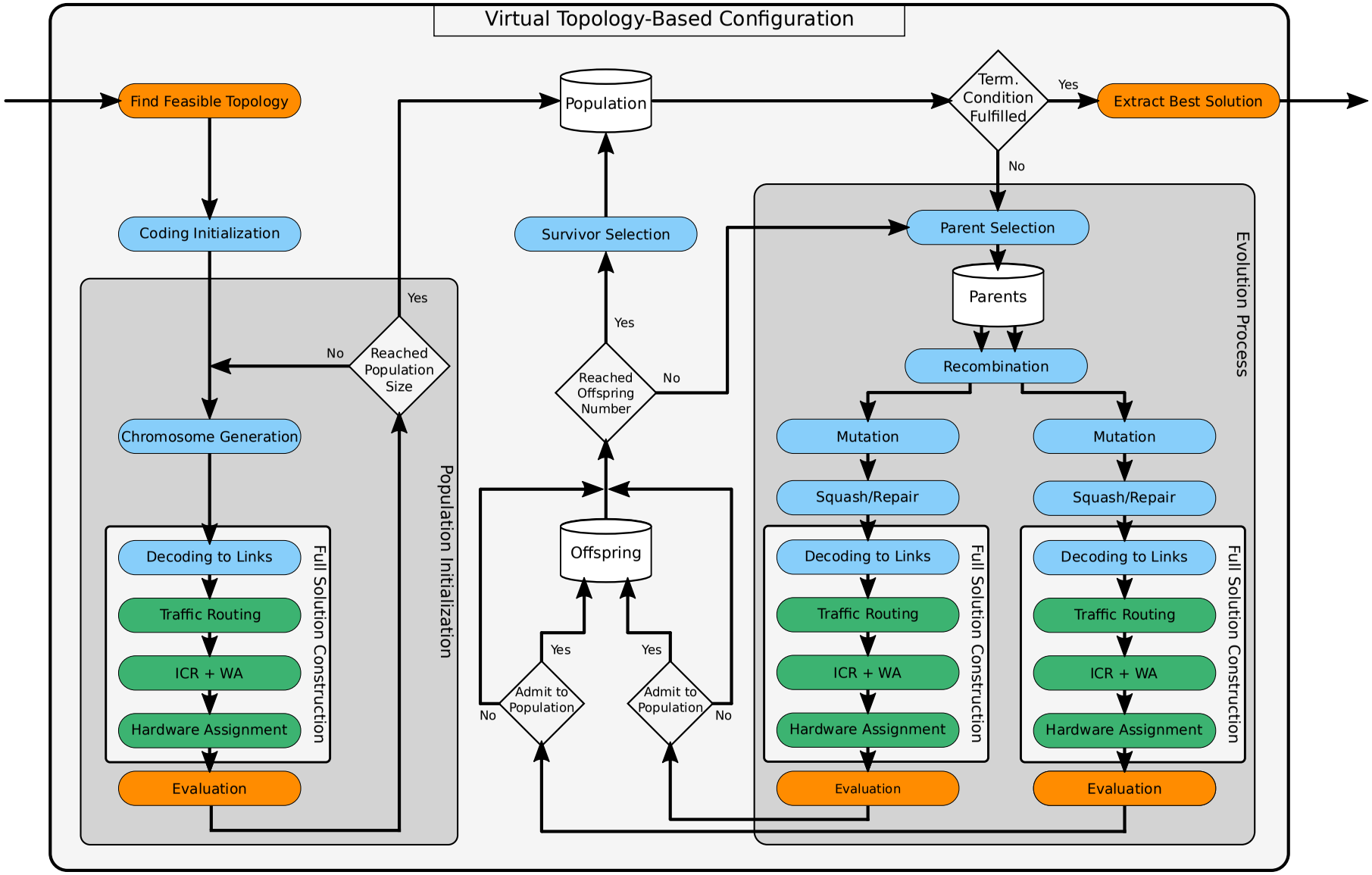



Figure 4.10: Structure of the subcomponents used in the Virtual Topology-Based Configuration

colors indicate functions using heuristics, while orange-colored blocks represent algorithms without any heuristic parts. The first step in the top left corner is to determine the optically feasible topology based on the infrastructure and hardware parameters, such as fiber lengths, available TXP capabilities and maximum transparent reach, given as part of the problem definition. The resulting feasible links are then used to initialize the chromosomes, e. g., by mapping genes to links for VTB.

The following step is the population initialization, where chromosomes are created according to one of the explained algorithms such as ASTI or APTI. Since the objective function typically requires information of the entire solution, the chromosome is decoded and a full solution is assembled from the results of the individual heuristics. First, a traffic routing is done by finding a shortest path using Dijkstra's algorithm for each demand individually. As the metric we use the number of hops in the upper layer biased by the number of hops on the shortest circuit of the lower layer with absolute distance as a tie-breaker. This metric emphasizes minimizing the intermediate terminations, while the bias also allows to include physical-layer properties.

The resulting route is then checked against the demands' QoS constraints. If it does not meet the requirements, a second path search is triggered, which looks for the shortest compliant path. This can be done based on Martins' algorithm [158] or extensions of k-shortest path algorithms. We use the latter approach, since we do not require the entire Pareto front, but only one shortest path. It may seem counter-intuitive to start with Dijkstra's algorithm, rather than to directly compute the compliant path, but we found this approach to be more efficient. Algorithms that consider additional measures often run slower than pure shortest path algorithms considering a single metric. Since the shortest path in terms of distance is very often a compliant path when considering latency and availability, which scale with distance, the drawback of running a second algorithm for a few cases is often smaller than running a more complicated algorithm every time.

Once the traffic routing is completed, the amount of capacity on each virtual link is known and ICR begins by sequentially routing circuits, starting from the shortest virtual link, until the required capacity is met. Yen's algorithm [286] is used to find a shortest path on the physical topology, where the hop count is used as a metric with fiber distance as a tie breaker. After a path has been found, the largest possible circuit capacity is selected, and WA is performed by a simple first-fit assignment. If there are no free channels on a fiber, it is removed from the search graph and the search is repeated. If the fibers in the path do not have overlapping free spectrum, such that a single contiguous channel cannot be found, the path search is repeated with the next shortest path. This is certainly not the most efficient implementation for this task, since cases of resource starvation will drastically reduce performance, but such cases are not the primary focus of this work and the framework is sufficiently flexible to use a different RWA implementation, should it be required.

Once all optical circuits have been set up, the hardware assignment function determines the required amount of TXPs and infers the required amount of line cards and other upper-layer components. While this is a simple counting algorithm, the function block in the diagram is considered as a heuristic, since a second component, originally developed by Feller for use in reconfiguration methods [77] can also be used. When considering a multi-period or reconfiguration problem, NEs like line cards, ports and TXPs may be partly occupied by a previously active configuration and therefore pose the additional question, which of the available NEs should be used. Feller's heuristic approach tracks usage over time and preferably assigns NEs which show high utilization values. However, since such investigations are largely outside the scope of this work, we will not go into further details.

Once an entire population has been created and its individuals evaluated, the evolutionary cycle starts and is repeated, until the termination condition is fulfilled. We use three such conditions that can also be combined to terminate when any of the three is met. There is a limit on the number of solution construction processes, a simple time duration limit for the overall algorithm and a target cost limit for the best solution. Following the genetic operators through selection, recombination and mutation, the process arrives at the *Squash/Repair* block. When VTB is used, this block can either perform an advance check or repair an unconnected chromosome using Algorithm 3. If the repair-function is inactive and the chromosome is found to be unconnected, it is squashed, i. e., no solution construction is performed and the chromosome is immediately discarded.

A resulting individual is then subjected to the population management algorithms. The crowding algorithm determines the respective population sub-group based on chromosomal similarity and compares the evaluation's result to the values within this sub-group as explained in Section 4.5.2.2. If a new chromosome underperforms, it can be excluded from the offspring and the evolutionary cycle has to be repeated. If a sufficient number of offspring has been computed, survivor selection determines the population of the next generation including the elite individual. The result is then reclassified by the crowding algorithm, before either continuing the evolutionary process or meeting a termination condition and returning the best solution found.

In terms of performance, the repeated execution of the solution construction process is the most time-consuming part of the entire approach, due to the necessity of repeatedly determining shortest paths for demands and for circuits. This is especially noteworthy, since the initialization phase is typically finished relatively quickly, when considering that the evolutionary cycle is run thousands of times. In order to accelerate some of these processes, the solution construction is supported by path caches and preloaded data structures for the constant parts of the network, but the computational effort is still large. Memory on the other hand is less of an issue, since the individuals are relatively compact, even for large networks. This allows for very large populations.

As explained in the context of VTD and TDR, this integrated approach is expected to be much more scalable than a routing-based one, but the attainable solution quality may be lower. However, a second drawback with respect to high levels of traffic has to be expected even in smaller networks. When traffic volumes increase, more and more circuits have to be employed to match all demands. Whenever nodes can be bypassed by well-utilized circuits, the conditions are favorable for an improvement in resource efficiency by optimizing VTD. However, when traffic demands between node pairs grow far beyond the circuit capacity, eventually all realizable virtual links are part of the optimal solution. At this point a pure connectivity-based optimization cannot find the optimal network configuration anymore. However, at the same time, the potential benefits in terms of resource efficiency improvements are subject to diminishing returns since the majority of circuits already runs at full utilization.

Not only do the extent and density of the fiber infrastructure, as well as the technological parameters defining the feasible topology exert great influence on the algorithm, but also the traffic demands themselves. This highlights that, while the Virtual Topology-Based Configuration can be applied to many multi-layer problems, the exact scenario has a large influence on both, the efficacy and efficiency of the approach.

### 4.5.4 Compliant Routing-Based Configuration

The second integrated approach uses the same basic layout as the previous one, but rather than focus on the topology, it uses one of the path encodings to create an optimized traffic routing from which the full solution is derived. The uncapacitated TDR solved by the genetic algorithm uses the full feasible topology to determine the routes, thereby entirely omitting VTD. Just like with the previous approach, this is followed by a partly uncapacitated ICR and WA to determine the circuits and their capacity and from this, the required number of TXPs and other NEs is determined.

Figure 4.11 shows an illustration of the entire process of the Compliant Routing-Based Configuration. The function blocks' colors have the same meaning as in Figure 4.10, i. e., GA-specific blocks are blue, other heuristics green and other ancillary functions orange. Since this approach is based on an encoded routing, either RCPE or DCPE can be used for the GA, along with the compatible evolutionary components as explained in Section 4.5.2.2. The approach begins in the top left corner of the illustration by establishing the optically feasible topology. Based on this, it determines shortest paths for each pair of nodes using Yen's algorithm and uses these to initialize the chromosomes. The metric is identical to the last approach, i. e., using upper-layer hops, biased by lower-layer hops with distance as a tie-breaker. Depending on the choice for $k_{\mathrm{m}}$ and the density of the underlying feasible graph, this can be a very computation- and memory-intensive endeavor. The resulting paths are stored in a lookup table accessible by the chromosome decoding functions. For DCPE a local index is created for genes which correspond to QoS-enabled paths, which automatically excludes all non-compliant paths.

After these preparation steps, the population initialization creates the population based on the selected initialization algorithms. Constructing a full solution is slightly different than for the Virtual Topology-Based Configuration. Since VTD is not needed and path computation is mostly relegated to the early preparation steps, determining the entire traffic routing is now done by retrieving paths from the lookup table. The only exception being that for RCPE the selection via any gene value may initially point to a non-compliant path and therefore may trigger an additional search for QoS-compliant paths. However, due to the distance-correlation of availability and delay figures, this can be expected to be only a sporadic occurrence as outlined in Section 4.3.2. Every used path flags the virtual links it uses as active and stores the required capacity. Following the traffic routing, the required virtual link capacities are then extracted by iterating over the used virtual links, which now implicitly define the required virtual topology.

After this, ICR sequentially routes circuits considering the required capacity and connectivity using Yen's algorithm, followed by a first-fit WA and hardware assignment, exactly identical to the virtual topology-based approach. The remaining compliant routing approach also follows the same steps as the virtual topology approach, going through parent selection, recombination, and mutation, before the potential offspring are subjected to the population management algorithms and survivor selection is applied, after which the results are checked regarding the same termination conditions. In terms of differences, the solution construction follows different steps, as explained above, and the choice of evolutionary operators is different, as summarized in Section 4.5.2.2. Furthermore, there is no squash or repair option in this approach, since all chromosomes exclusively encode compliant solutions.

For this approach, there can be two performance bottlenecks. The path enumeration lookup table and the population of full solutions. Regarding memory requirements, the path enumeration can rival the population. A larger number of possible paths immediately results in an increased state space and consequently a more diverse and therefore potentially larger population may have to be used. This implies a certain relationship between $k_{\mathrm{m}}$ and the population size, such that

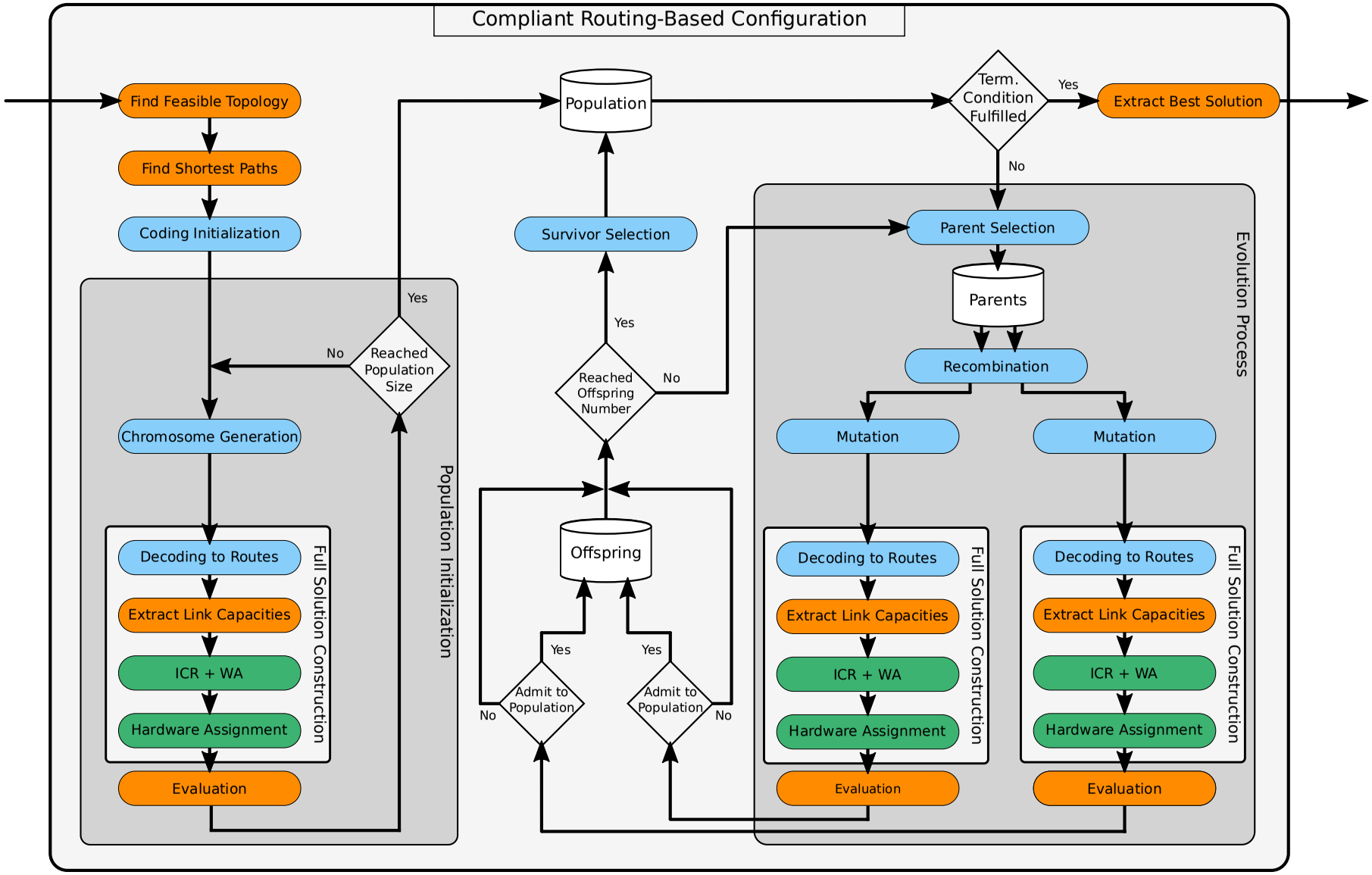



Figure 4.11: Structure of the subcomponents used in the Compliant Routing-Based Configuration

severely limiting one parameter to allow for a larger value of the other may not yield the desired results, such that the tradeoff needs to be carefully adjusted. In terms of computational effort, the full solution construction is more lightweight than for the virtual topology-based approach, but it still requires a large number of path searches for ICR. The initial path enumeration also requires a large number of path searches, which may contribute in large parts to the overall runtime. Depending on the exact scenario, however, a reusable precomputation of the paths may be able to drastically reduce the impact. E. g., when considering a continuous network reconfiguration reacting to changes in the traffic demands, the path enumeration needs to be computed only once for all such problems, as long as the physical parameters do not change. However, when a routing is to be optimized for a continuously changing feasible topology, then the route enumeration has to be recomputed every time. This is also the primary reason, why using VTB or VTCS in conjunction with a subsequent optimization by RCPE or DCPE is not a promising approach.

When considering the scalability of the approach, the arguments regarding RCPE and DCPE made in Section 4.3.2 and Section 4.3.3 respectively, also hold true: Given sufficiently large values of $k_{\mathrm{m}}$, the solution quality can be expected to be higher, but the required runtime to sufficiently explore the state space will increase drastically and approach intractability rather quickly. Sadly, there is no universal rule of thumb, regarding the choice of $k_{\mathrm{m}}$, such that it will have to be empirically determined. This is especially difficult in scenarios incorporating flexrate devices of large reach, since many shortest paths are likely to use long-reach, but low-rate modes, such that the enumeration may miss significant grooming potential.

Just with the virtual topology-based approach, the scenario parameters have a very significant impact on the achievable solution quality. The feasible graph density, as defined by the physical topology and the technological parameters, is even more important for the routing-based approach, as it does not only affect solution quality, but tractability as well. It is therefore especially important for the Compliant Routing approach, to exploit the inherent parallelism of the GA approach by a suitable implementation.

### 4.5.5 Parallelization Approach

Since GAs are inherently parallel processes and scaling out computing hardware is currently much easier than scaling up, a parallel implementation of GA-based approaches promises improved scalability towards larger problems. In order to provide the most acceleration, components executed in parallel should not only handle the most computing-intensive tasks but should also be largely decoupled to avoid synchronization delays. The algorithms within the evolutionary processes as shown in the right-hand side boxes within Figure 4.10 and Figure 4.11 consume the largest share of the overall computation time, making them suitable for parallelization. Between the selection of parents and the admission into the population, all function blocks do not require to exchange any information between each other, such that there is no need to synchronize any recombination, mutation or solution constructing functions.

In contrast to this, the survivor selection function necessarily requires a global view of the current population and offspring, in order to run comparisons between all individuals and establish the next generation of the population. The population management functions, which are run together with the survivor selection, also include the classification analysis, which is needed for the crowding-based admission control mentioned in Section 4.5.3. The actual admission decision for population management, however, can be separated from the survivor selection and population evaluation. It can be run in parallel following the evaluation, since it requires only read access to the population, which does not change before survivor selection.

Finally, in case of the Compliant Routing Configuration, there is the computationally intensive path computation in the early initialization phase. This is completely separated from all remaining function blocks, as it cannot be run without the input of the previous block and its entire output is a prerequisite for the coding initialization. Since the interface to the other processes is very lean, as it only requires read access to the search graph representing the feasible topology and the output of each path search is not dependent on any other, it also presents itself as a candidate for parallelization.

Due to this partitioning of data flows, we used a master–slave concept for the parallelization, where a master process runs all central functions and instructs other worker processes regarding their tasks. The blocks termed "evolution process" in Figure 4.10 and Figure 4.11, followed by the admission control checks, are handled as an individual task. The main process checks the termination conditions and if they are not yet fulfilled, it organizes the evolution of the next generation. It first determines, how many such tasks have to be run in total to meet the given number of offspring and then it evenly divides these among a preset number of worker processes.

Each of these workers maintains their exclusive copy of the graph and evaluation data structures. These data structures can therefore be used over the entire runtime of the Configuration algorithm, such that there is no additional overhead due to frequent allocation and deallocation of memory after every generation. The main process starts the workers and remains suspended, until each of the workers signals their termination and hands over the new individuals, which can then be organized in the survivor selection function by the main process.

Furthermore, the main process also runs the classification for the crowding and the initial path computation for the Compliant Routing Configuration in parallel. For these tasks it uses lightweight temporary worker processes, since — in contrast to the worker processes for the evolution — these do not require maintaining complex data structures and can be discarded after a single use, to better manage resources.

## Chapter Summary

The focus of this chapter are the genetic algorithm adaptions with their prerequisites, design considerations and integration into solution methods. We have presented, how the hardware abstractions developed in Chapter 2 determine our design space and following from this, we have detailed, what constitutes a solution to the established multi-layer problems explained in Chapter 3. We have further laid out a number of aspects to consider in the design of genetic algorithm methods, including computational efficiency, coding complexity, genetic locality, and heredity.

We have presented four different genetic encodings that we have developed for multi-layer optimization, two for Virtual Topology Design problems and two for Traffic Demand Routing problems. The first encoding, which we denote VTB, is a simple binary link-encoding, augmented by specialized initialization algorithms, advance checks and repair functions. The second is a topology-oriented encoding, called VTCS, which combines the resource-efficiency of special spanning trees, constituted by integer-encoded edges of high centrality, with additional augmentation links encoded as binary values, which provide shortcuts indispensable for QoS-enabled demands.

For traffic routing, the RCPE and DCPE encodings have been suggested. While both are based on path enumeration principles, the former tries to enumerate paths between all nodes using an integer-valued index, splitting off and rerouting QoS-enabled demands from non-compliant routes. The latter encoding considers each demand individually and enumerates only compliant paths, such that the resulting solution is always compliant as a whole. We combine these encodings with different initialization schemes, as well as our Longhorn Mutation and exponentially biased crossover operators, which have specifically been crafted to increase the rate of convergence for these encodings.

For each approach, we have discussed how and why they can meet the design criteria established in the beginning of the chapter. Following these individual analyses, we comparatively highlight the implications of integrating these encodings in a multi-layer network context and present an overview on the implemented genetic algorithm components, also including many well-established approaches for reference. It has been explained, how these components have been integrated into a larger, parallelized framework together with other heuristics in order to efficiently solve complex multi-layer configuration problems, highlighting advantages and drawbacks of each approach, which we will evaluate regarding their performance in the next chapter.

# 5 Evaluation

The focus of this chapter is on the performance evaluation of the two integrated multi-layer network configuration approaches, introduced in the last chapter. One of these is based on the virtual topology, one based on routing and each of these can use two different encodings and several GA components in a flexible framework. While the analysis of the last chapter was based on general properties of individual components, the evaluation in this chapter will use several exemplary scenarios to gauge different aspects of performance.

## 5.1 Methodology

Chapter 4 has already provided a theoretical exploration of the methods' properties, especially regarding scalability of the state space. A further analytical evaluation in the context of specific applications, however, is difficult due to the problem dependency and the Monte Carlo-nature of GA-methods. We will therefore experimentally evaluate the performance of the suggested approaches in a number of indicative and representative scenarios, focusing on solution quality and runtime.

Given the flexibility of the framework and the large number of hyperparameters associated with the various components, it is beyond the scope of this monograph to investigate the entire parameter space. Furthermore, due to the considerable influence of the exact problem scenario, it is also difficult to provide a singular set of components and parameters that can be expected to perform well in all scenarios. We have therefore designed a number of scenarios, each highlighting different effects.

### 5.1.1 Scenario Design

To provide the most general perspective, we will investigate our approaches in several steps. Initially, we will validate core functionality and design parameters using a number of artificial scenarios with small graphs and reduced scenario complexity. These problems are designed to be small and simple enough that not only can their optimal solution be determined by an exhaustive solution enumeration, but also that they are still readily comprehensible. This is intended to evaluate, whether the suggested methods are in fact able to achieve fundamental goals such as fulfilling QoS constraints and exploiting grooming potential.

The second evaluation is based on a known, albeit less complex reference problem, for which a lower bound to solution cost is known. For this evaluation, we do not focus on the performance-enhancing effects of different parameters, but rather we employ a very long runtime in order to see, what the lowest attainable cost values are for a given encoding in a basic setting. We compare the results to those obtained from a shortest path routing and a comparable Simulated Annealing approach, known to be effective for multi-layer network optimization. The comparison therefore indicates, if the suggested approaches provide an improvement over much

simpler legacy approaches and if they can compare to an established optimization approach, thereby demonstrating their viability for solving general networking problems.

The third evaluation is concerned with the performance of the entire framework in a full multi-layer configuration setting. For this we use a complex network scenario including all relevant features and parameters with a relatively short runtime limit on the order of minutes, which is closer to operational time frames as explained in Section 2.2.1.1. In this scenario we compare the influence of using different components. We analyze the effects on both, performance and attainable cost values, while providing the results from Simulated Annealing and the legacy approach as a reference. This investigation demonstrates two things. First, it shows that the developed approaches are in fact capable of optimizing multi-layer problems of the intended level of integration including QoS-enabled demands and flexrate TXPs. Second, it highlights, which components of the framework provide the largest contribution to performance, and which turn out to be ineffective.

The final evaluation is focused on demonstrating scalability. It is based on a large-scale scenario, which is inspired by a real transport network of relatively high density and large geographical coverage. Again, we will provide results obtained from the Simulated Annealing and legacy approaches as reference in order to determine the applicability to such real-world problems.

### 5.1.2 Simulation Environment

In order to experimentally evaluate the GA-based framework we have implemented it in Java. This implementation is derived from a preexisting network optimization and simulation tool, originally developed by Feller throughout the STRONGEST [58] and SASER [199] projects and extended by the author of this monograph and others [12, 72, 73]. The original tool provides a network and hardware abstraction along with a traffic demand simulation. Both, the original tool and our version, rely on two further libraries, the IKR SimLib and JGraphT. The IKR SimLib [130, 253] is a library for event-driven network simulation and performance analysis, of which we use version 4.0.0. JGraphT [162, 172] is a library providing graph abstractions and algorithms, which we use as version 1.4.0 with a few custom extensions.

For random number generation we use a customized version of the Mersenne Twister [160], which allows creating independent instances per thread based on an initial seed value from which the seeds for the independent instances are then derived. We use Java 1.8 with parallel garbage collection and manually tune heap sizes to provide sufficient memory for the entire simulation in order to avoid unnecessary garbage collection or heap resizing events, which would otherwise introduce stochastic delays.

There are several other sources of stochastic effects that may influence the evaluation. A computer system's load situation outside of the simulation and evaluation environment may also impact measurements, since the simulation may be interrupted by high priority tasks, introducing delays. To somewhat mitigate this effect, we run simulations involving time measurements on computers that run no other background tasks than strictly required for the operating system and the evaluation framework. All time-sensitive simulations have been performed using systems with dual-socket Xeon E5-2640 v4 processors with 10 cores and a maximum frequency of 3.4 GHz.

Apart from these environmental effects, there may also exist a dependency on the exact seed values used for the random number generators. In order to account for the mentioned and other stochastic effects, we will run 10 independent repetitions with different seeds of each scenario and present the resulting deviations.

### 5.1.3 Reference Methods

We use two basic reference methods to compare our approaches against. The first is based on a simple shortest path routing, representing an upper bound to the solution quality. The second is based on Simulated Annealing, which is a comparable metaheuristic. Ideally, we would also provide an optimal solution, or at least a lower bound, but for complex and large-scale scenarios, we do not have access to reference problems or exact methods that are sufficiently scalable.

#### *5.1.3.1 Shortest QoS-Compliant Path Routing*

To create the shortest path routing reference, we took our framework for the virtual topology-based encodings and removed all GA-functions, except for the initialization. Therefore, it uses the same routing metric and support functions as our GA-based approaches, which avoids unwanted bias, which could be caused by different ICR or WA implementations. We use the initialization function to create two virtual topologies for the routing. The first is the regular physical network topology without any bypass links. This is the baseline that we consider to be obtainable by all legacy approaches. The second is the full optically feasible topology.

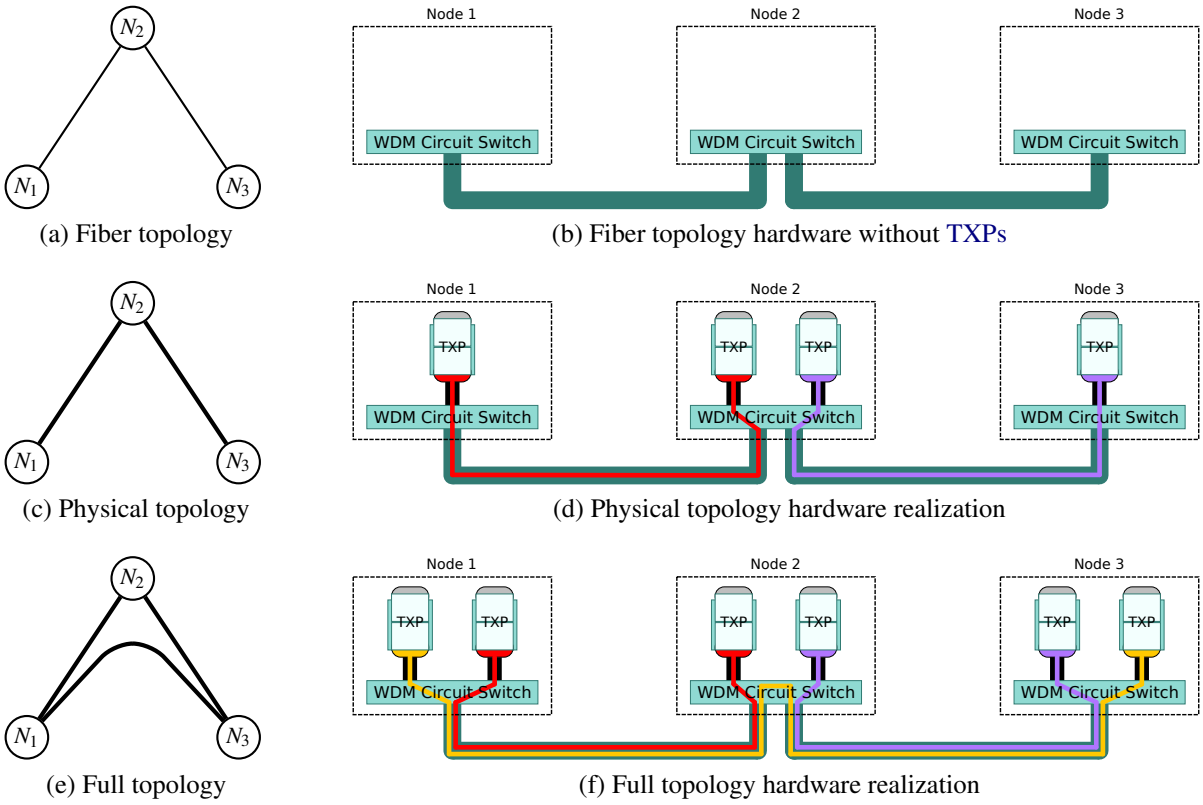
This is illustrated in Figure 5.1 using a small example network of three nodes. The left side shows an abstract view of the topologies while the right side shows a hardware realization for these topologies regarding TXPs. The thin lines in Figure 5.1a show available fibers without active virtual links and consequently Figure 5.1b shows only "empty" fibers without any TXPs installed and therefore no virtual links at all. The physical version shown in Figure 5.1c, where thick lines indicate virtual links, and Figure 5.1d uses one virtual link on each fiber with the minimum number of TXPs to achieve this. For the full feasible topology, as seen in Figure 5.1e, we assume that node 1 and 3 are within transparent reach, such that there are virtual links between all nodes at the expense of a higher number of TXPs. The thick line between node 1 and node 3 is bent towards node 2 to indicate the actual wavelength routing through the fibers as shown in Figure 5.1f.

When flexrate TXP of very large reaches are used, the feasible topology will most likely result in low quality solutions, since using every possible bypass link requires a large number of TXPs and corresponding upper-layer NEs. The physical topology-based routing is therefore the baseline, any optimization has to improve upon in order to be viable. The virtual topology-based routing gives an exaggerated upper bound, that can be considered the most costly of feasible solutions and helps to gauge the extent of influence flexrate devices have on the solution.

#### *5.1.3.2 Simulated Annealing Reference Methods*

Our main reference method is a Simulated Annealing-based approach, which we have already found to be effective in optimizing multi-layer network configurations including flexrate devices [12]. This approach has been extended from Feller's original reconfiguration solution method, that he had determined to be superior to other metaheuristic approaches for multi-layer network problems [77, ch. 4.4.3]. We have integrated it in our virtual topology-based framework, such that it uses exactly the same supporting algorithms and cost functions as our GA, enabling a direct comparison of the optimization method adaptions.

We use two different versions of the Simulated Annealing approach. SVT perturbates the virtual topology by adding or removing links and then performs the entire traffic routing. This is directly comparable to the virtual topology encoding approaches we have developed for our GA. The second version, called SVTR, uses an adapted version of Feller's perturbation approach [77, pp. 100–101], which supports QoS traffic and flexrate devices. The advantage of SVTR is that




(a) Fiber topology

(b) Fiber topology hardware without TXPs

(c) Physical topology

(d) Physical topology hardware realization

(e) Full topology

(f) Full topology hardware realization

Figure 5.1: Fiber topology example and corresponding virtual topologies used as reference

it maintains the previous network configuration and only performs essentially required changes as follows. If it activates a virtual link, it checks for all demands if shorter routes have become available and only replaces the old one, if it is shorter than the previous one. If it deactivates a link, it reroutes only the traffic demands, which had previously used this specific link. Therefore, it is able to passively optimize parts of the routing for paths of equal lengths and since it alters only part of the data structures, it requires less traffic routing operations.

Similar to the shortest path routing, we also provide the entire feasible topology, the physical topology, or a random topology as initialization for Simulated Annealing in order to provide a level ground for the evaluation. We know from Feller's and our own studies that the approach is effective for multi-layer networks and Feller has demonstrated that it can be expected to scale better for large and complex problems than ILP-based approaches. Therefore, we can use it as a reference for both, scalability and solution quality obtainable by metaheuristics.

## 5.2 Core Function Validation

In this section we will investigate a number of test cases that exemplify basic principles and core functionalities that contribute to improving the overall solution quality towards the optimum for a generic type of multi-layer network problem. This will highlight, which of the suggested approaches can make use of such principles as part of their optimization process. This is by no means exhaustive and does not guarantee that the underlying principles will be utilized to their full extent in arbitrary scenarios, nor is it guaranteed that a combination of these principles will always lead to an optimum for every type of problem. However, it is sufficient for a basic valida-

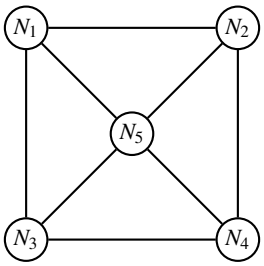



Figure 5.2: Topology *5N8L*

tion, which highlights specific limitations of the presented approaches, that give an indication on the scope of problems that cannot be solved to optimality and the reasons for this.

### 5.2.1 Scenario Parameters

#### *5.2.1.1 Network Topology*

To increase the visibility and aid a ready understanding of the intended and measured effects, we demonstrate them using a small model network topology we will refer to as *5N8L* as it consists of 5 nodes and 8 bidirectional links as shown in Figure 5.2. This generic representation may be regarded as a PTD problem with potential links or a VTD problem with existing physical links.

Table 5.1: Parameters of the basic *5N8L* topology

| Topology | Nodes | Links | Density | Degree | | Closeness | | Betweenness | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Max. | Avg. | Max. | Avg. | Max. | Avg. |
| Basic | 5 | 8 | 0.8 | 4 | 3.2 | 1 | 0.84 | 4 | 2.4 |
| Full | | 10 | 1 | 4 | 4 | 1 | 1 | 0 | 0 |

Since this topology is artificial, there are no actual link lengths associated with the edges of the graph. *5N8L* is already a relatively dense topology compared to the corresponding full topology for five nodes, such that there is only a small difference in the resulting complexity. The basic parameters of this topology and the corresponding full topology of all possible links are given in Table 5.1.

#### *5.2.1.2 Physical Parameters*

Traversing links and nodes adds a fixed amount of delay and links can have a real-valued availability figure between 0 and 1. While the topology itself does not have any lengths, we arbitrarily[1] assign a value of 10 time units and an availability of 0.5 to the horizontal and vertical links and, following the proportions of a 2D-plane, a value of 7 time units and an availability of 0.8 to the diagonal links. A data-carrying connection between two points requires a terminal device at each of those nodes. To simplify the description, we will use the terminology of VTD, such that a connection represents an optical circuit and a terminal device is a single-port line card with integrated TXP. For these tests we only consider circuits of a fixed, singular circuit

[1]Please note, that in this artificial example the numbers are chosen for reasons of clarity and readability. Real fibers can be expected to have much higher availability figures.

capacity and disregard the wavelength continuity constraint. Depending on the test case, the transparent reach is either only sufficient for all physical links or additionally allows for the bypass links $N_1$ to $N_4$ and $N_2$ to $N_3$.

#### 5.2.1.3 Traffic Demands

The traffic in each of the following test cases is purely artificial and designed to result in specific optimal solutions, which serve to demonstrate specific effects. For this part of the evaluation, we consider all traffic demands to be unidirectional. This makes for a clearer scenario and does not reduce the generality, since any bidirectional demand can be considered to consist of two unidirectional demands. We denote each individual traffic demand as $d_{s,d}$ with source vertex $s$ and destination vertex $d$. Every demand requires a certain capacity, which we give relative to the circuit capacity, i. e., a demand with 0.5 capacity units uses 50 % of the capacity of a single circuit. Splitting of a demand is not permitted. Demands may or may not have QoS-constraints, which will be indicated by a superscript, where the prefix "$L$" indicates a latency and the prefix "$A$" an availability constraint. E. g., the demand $d_{1,3}^{L8}$ originates at node $N_1$, is destined for node $N_3$ and requires a delay of at most 8 time units.

#### 5.2.1.4 Optimization Goal

The optimization problems to be solved have an objective function that minimizes the number of required line cards and heavily penalizes any unrouted demands. Depending on the exact test case, the objective function also has a secondary objective. For cases where no QoS-constraints exist, $F_{mLat}$ minimizes the average latency of all demands, whereas for cases including explicit QoS figures, the secondary goal of $F_{QoS}$ is simply to fulfill all such constraints. Beyond these points, no other constraints exist, such that the problem can be considered to be of the uncapacitated[1] category. The objective functions are defined as follows.

$$F_{mLat}(s) = \sum_{v \in V} (n_{\mathrm{LC},v} \cdot \alpha) + n_{\mathrm{UD}} \cdot \beta + \frac{\sum_{d \in D} f_{\mathrm{lat}}(d)}{|D|} \cdot \tau \tag{5.1}$$

and

$$F_{QoS}(s) = \sum_{v \in V} (n_{\mathrm{LC},v} \cdot \alpha) + n_{\mathrm{UD}} \cdot \beta + n_{\mathrm{AV}} \cdot \phi + n_{\mathrm{LV}} \cdot \psi \tag{5.2}$$

with

$$\begin{gathered} \beta > |V| \cdot m_{\mathrm{LC}} \cdot \alpha \\ \alpha > \left( \sum_{e \in E} \delta_f \cdot f_{\mathrm{len}}(e) + \sum_{v \in V} \delta_h \right) \cdot \tau \\ \alpha > |D| \cdot \phi \\ \phi > |D| \cdot \psi \\ \alpha, \beta, \tau, \psi, \phi \in \mathbb{R}_+ \end{gathered}$$

where $s$ is a solution candidate. Lower case Greek letters without indices are scaling parameters to adjust the importance of the functions' components. In these equations $n_{\mathrm{LC},v}$ is the number of TXPs present at node $v$ and $m_{\mathrm{LC}}$ is the maximum number of TXPs installable per node. $n_{\mathrm{UD}}$ is the number of unrouted demands, $n_{\mathrm{AV}}$ represents the number of demands which miss

[1] Note that demands can still be unroutable in this case if the topology of the candidate solution is not sufficiently connected to enable the required reachability as explained in Section 4.3.1.

their availability requirements, whereas $n_{\text{LV}}$ is the number of demands exceeding their latency requirements. $\delta_f$ is the latency per km and $\delta_h$ the latency per hop. Finally, the function $f_{\text{len}}$ returns the physical length of an edge and the function $f_{\text{lat}}(d)$ gives the latency experienced by demand $d$. The inequalities ensure that there is a total order on the precedence of the components within the functions, e. g., for $F_{mLat}$ the maximum accumulated penalty from $\alpha$ cannot exceed the least penalty from $\beta$ and the maximum accumulated penalty from $\tau$ cannot exceed the least penalty from $\alpha$.

#### *5.2.1.5 Algorithmic Parameters*

Due to the small size of the problems, we chose very basic components without any specific performance enhancements and we did not use the shortest path reference approach, since this evaluation is not about performance, but functionality. The parameters chosen for the Simulated Annealing are a more simplified version of the ones suggested by Feller [77, Ch. 5.2.3], where the change between temperature values is either caused by a maximum of iterations per temperature or by the number of accepted candidates per temperature. The exact values are given in Table 5.2. For the GA-approaches we also used a very modest set of parameters, which are given in

Table 5.2: Simulated Annealing hyperparameters for core function test cases

| Type | Initial Topology | Initial Temp. | Cooling Factor | Iterations per Temp. | Accepted per Temp. |
|---|---|---|---|---|---|
| SVT | Physical | 1 | 0.95 | 1000 | 100 |
| SVTR | Physical | 1 | 0.95 | 1000 | 100 |

Table 5.3, where the Creep operator uses a step size of 1. Regarding performance enhancements it should be noted that we always use basic elitism, i. e., we migrate the best individual into the next generation. For all approaches we use the same termination conditions. The algorithms

Table 5.3: Genetic Algorithm hyperparameters for core function test cases

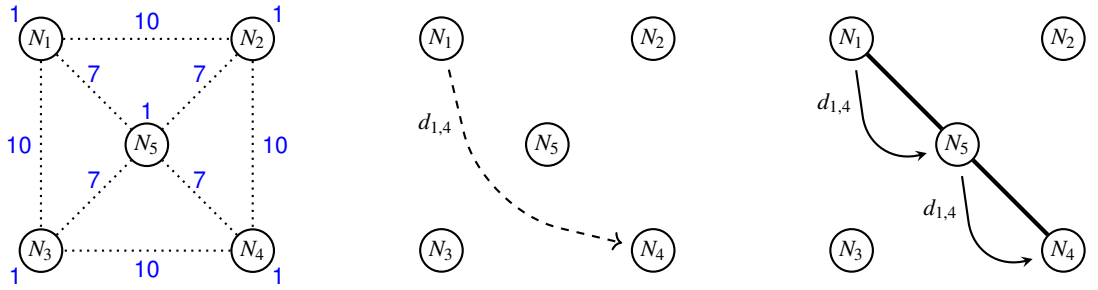
| Type | Population | | | | Selection | | Recomb. | Mutation |
|---|---|---|---|---|---|---|---|---|
| | Init. | Size | Offsp. | Control | Parent | Survivor | | |
| VTB | Random | 10 | 4 | No | RWS | $(\mu+\lambda)$ | CNCXO | URM |
| VTCS | Random | 10 | 4 | No | RWS | $(\mu+\lambda)$ | CNCXO | Creep |
| RCPE | Random | 10 | 4 | No | RWS | $(\mu+\lambda)$ | CNCXO | Creep |
| DCPE | Random | 10 | 4 | No | RWS | $(\mu+\lambda)$ | CNCXO | Creep |

immediately terminate, once the intended solution has been found, which we have validated to be the optimum using exhaustive enumeration. The second condition is a maximum runtime of 60 s, which is ample enough to perform a vast number of full enumerations.

### 5.2.2 Evaluation

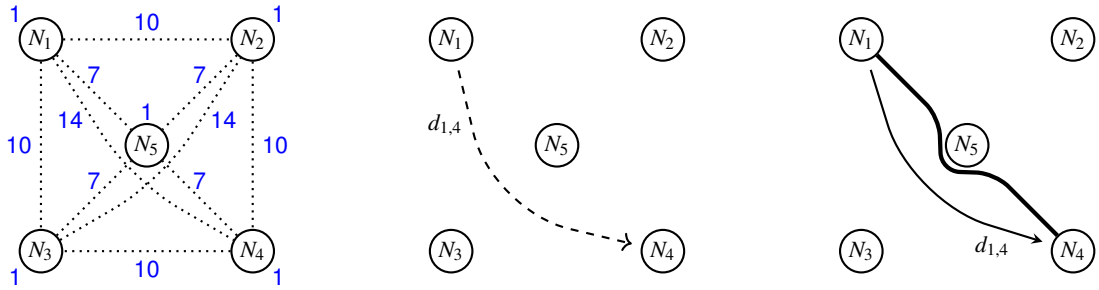
*Test Case* 1 *Shortest Path Routing in Virtual Topology*

This is the baseline test that simply requires to find the shortest path and therefore replicates the




(a) Topology and delays (b) Demand (c) Optimal solution

Figure 5.3: Illustration of Test Case 1 showing shortest path baseline




(a) Topology and delays (b) Demands (c) Optimal solution

Figure 5.4: Illustration of Test Case 2 showing bypass circuit via $N_5$

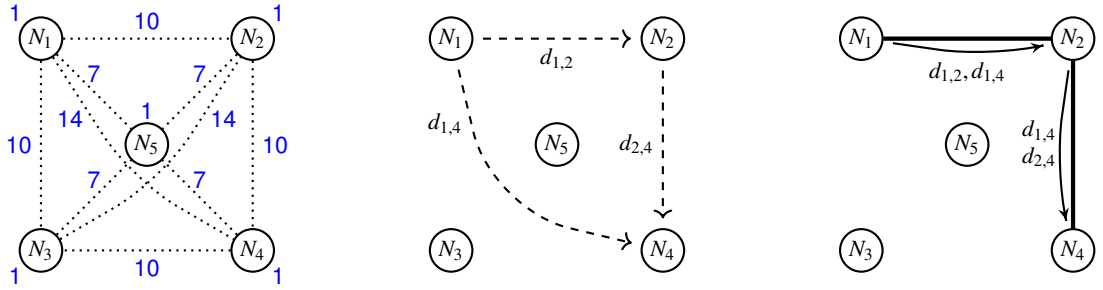
behavior of an algorithm such as Dijkstra's, where the edge weights correspond to the link delay. There is a single demand $d_{1,4}$ from node $N_1$ to node $N_4$ which requires the capacity of one circuit and has no further constraints, such that $F_{mLat}$ is used. The transparent reach does not allow any bypass circuit, thus allowing for virtual links as indicated by the dashed lines in Figure 5.3a.

A resource-minimal solution requires 2 circuits, one from $N_1$ to an intermediate node, i. e., any one of $N_2$, $N_3$, or $N_5$, and another circuit from the intermediate node to $N_4$ resulting in 3 different solutions each of 4 line cards in total. The optimal solution, however, additionally minimizes latency and therefore chooses the path with the least accumulated delay. As shown in Figure 5.3a, the paths with $N_2$ and $N_3$ as intermediate node each induce 10 units of delay per link and 1 unit per node for a total of 21 units each. The optimal choice is the path with $N_5$ as shown in Figure 5.3c, since it adds only 7 units per link, resulting in a total latency of 15 units. All algorithms and encodings reliably find the optimal solution.

*Test Case 2 Bypass Circuit in Virtual Topology*

This test uses an increased transparent reach and therefore requires the approaches to extend the basic topology to consider the full graph as a feasible topology as shown in Figure 5.4a. The demand $d_{1,4}$ in this scenario is identical to the previous case, i. e., requires a route from node $N_1$ to node $N_4$ with the capacity of one circuit.

Using $F_{mLat}$, a resource-minimal solution therefore requires just one diagonal circuit from $N_1$ directly to $N_4$, bypassing any intermediate node and thereby reducing the number of line cards to 2 in total. In addition, this also improves the secondary objective by avoiding the node delay of one unit, such that this bypass link is better than any of the others in the sense of the




(a) Topology and delays (b) Demands (c) Optimal solution

Figure 5.5: Illustration of Test Case 3 showing demand grooming

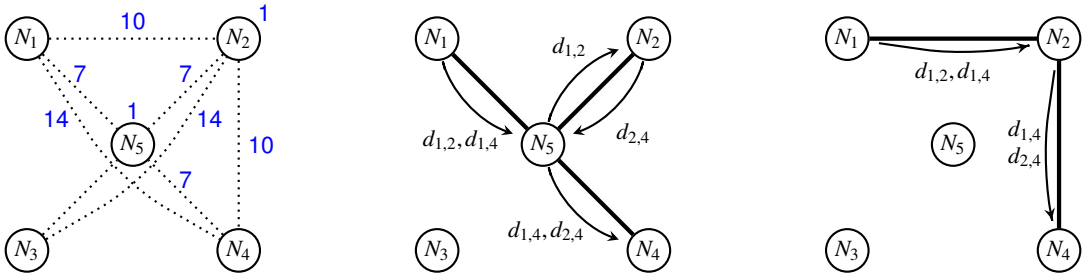
objective function, making it the optimum, which is shown in Figure 5.4c. All algorithms and encodings were able to determine this optimal solution.

*Test Case* 3 *Grooming of Traffic Demands into Circuits*
Grooming, i. e., aggregating demands of fine granularity efficiently into services of coarse granularities (cf. Section 2.1.2.3), is a very potent approach to increase efficiency in networking. To illustrate this principle, we need several demands requiring sub-circuit capacity, such that they can be multiplexed into the same circuit. We use three traffic demands that each require half of a circuit's capacity. These are $d_{1,2}$, $d_{2,4}$ and $d_{1,4}$. The feasible virtual topology is the full graph and $F_{mLat}$ is the objective function.

This scenario creates a conflict between resource efficiency and minimum latency. When starting from the first two demands, they can be routed directly on circuits between their source and destination to achieve minimal resources and minimal latency. The third demand experiences the least amount of delay on a third bypass circuit from $N_1$ to $N_4$. The resource-optimal routing, would groom it onto the circuits of the other two demands, thereby making use of spare capacity at the expense of increased latency. Since the primary goal of $F_{mLat}$ is resource efficiency due to the choice of $\alpha$, the optimal solution is the latter, yielding a solution of 4 line cards.

It should be noted that there exist more solutions of the same amount of line cards. When starting from $d_{1,4}$, one could establish a bypass circuit directly to $N_4$ and the second circuit either between $N_1$ and $N_2$ or between $N_2$ and $N_4$. In the former case, $d_{2,4}$ would be groomed into this link in reverse direction and then groomed into the bypass. In the latter case, $d_{1,2}$ would be groomed into the bypass and then into the other circuit. However, these solutions are inferior, because the average demand latency for both is 16.3, whereas in the optimal case above it's only 13.7 units. As before, this test case is solved to optimality by all methods, but there exist variations of this scenario, where some of the algorithms will fail to do so, which will be highlighted in the following cases.

*Test Case* 4 *Grooming of sparse and remote Demands*
This case is specifically designed as a scenario in which the VTCS encoding is not able to represent the optimal solution. The demands are identical to the previous case and the same objective function is used, but the virtual topology is now limited to those subsets of the physical topology, that do not include the links between $N_1$ and $N_3$ and between $N_3$ and $N_4$. Since these links were never part of the optimal solution to case 3, their omission does not have any impact on the optimal solution, such that it remains identical. All methods solve this case to optimality, except the VTCS encoding, which can neither find the previous optimum, nor any configuration




(a) Topology and delays (b) VTCS solution (c) Optimal solution

Figure 5.6: Illustration of Test Case 4 with comparison of solutions

with an equal number of circuits. By its design, this encoding assumes traffic between all nodes and is therefore generally not well-suited for this test case, where only few nodes have traffic between them, such that no spanning tree can be part of the optimal solution. For all previous test cases this encoding worked nonetheless since the integrated approach does not create circuits where there is no traffic and therefore it could still determine the optimal solutions.

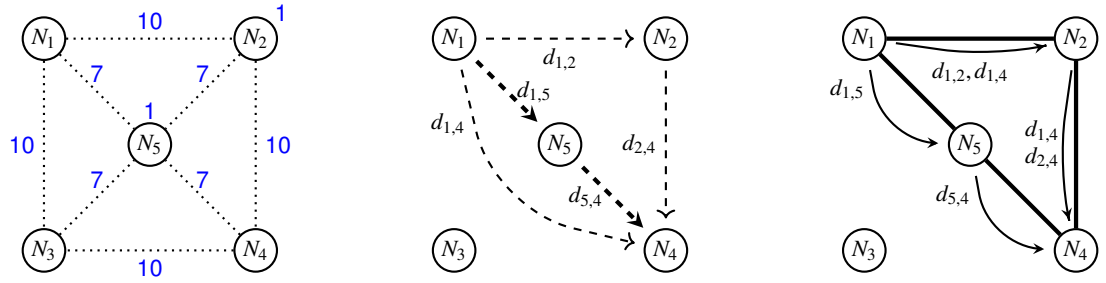
However, the VTCS encoding does not permit all possible spanning trees, relying on the assumption that nodes, which are central in the feasible virtual topology, should also be central in the spanning tree. Due to the omission of the aforementioned links in the feasible virtual topology, $N_5$ has become an articulation vertex[1] in the remaining graph and exhibits drastically increased centrality such that it becomes the center of any spanning tree representable by the VTCS encoding. Although there is no traffic for which $N_5$ is either source or destination, the resulting spanning trees would still be routing traffic via this node, such that a total of 3 circuits would be used as shown in Figure 5.6b. While this is a drawback for generic problems, it is a deliberate design choice since such topologies with barely any traffic, which occurs nowhere near to any central nodes, is a highly unlikely occurrence in real-world transport networks.

*Test Case 5 Alternative Routes and Grooming*

This case has a total of 5 demands and uses $F_{mLat}$. The demands $d_{1,2}$, $d_{2,4}$ and $d_{1,4}$ are identical to test case 3, and therefore each one requires half a circuit capacity. The new demands are $d_{1,5}$ and $d_{5,4}$ and they both require the capacity of an entire circuit, which is indicated by the bold arrows in Figure 5.7b. The feasible virtual topology corresponds to the physical topology without bypasses. Since the last two demands each require their own circuit, they leave no spare capacity, such that grooming is impossible for them. Therefore, these circuits can be set up on the direct edges from the demand's sources to their destinations. For the other three demands the same reasoning as in test case 3 applies, resulting in 2 more circuits and therefore the optimal solution has a total of 4 circuits and 8 line cards.

All of the route-altering algorithms, including SVTR, successfully find the optimal solution to the problem. All algorithms which rely only on a full shortest path routing in an optimized topology fail to find the optimum. The reason for this is that for all demands with the exception of $d_{1,4}$, a direct circuit is optimal for resources and latency, such that they already require the optimal circuit layout shown in Figure 5.7c on their own. This allows for two different routes for $d_{1,4}$ in the resulting topology. The route via $N_2$ allows for grooming and is therefore resource-optimal. The alternative route via $N_5$ prevents grooming, but it is shorter and therefore preferable

[1] An *articulation vertex* in a connected graph is defined as a vertex whose removal disconnects the graph.




(a) Topology and delays (b) Demands (c) Optimal solution

Figure 5.7: Illustration of Test Case 5 showing advanced grooming

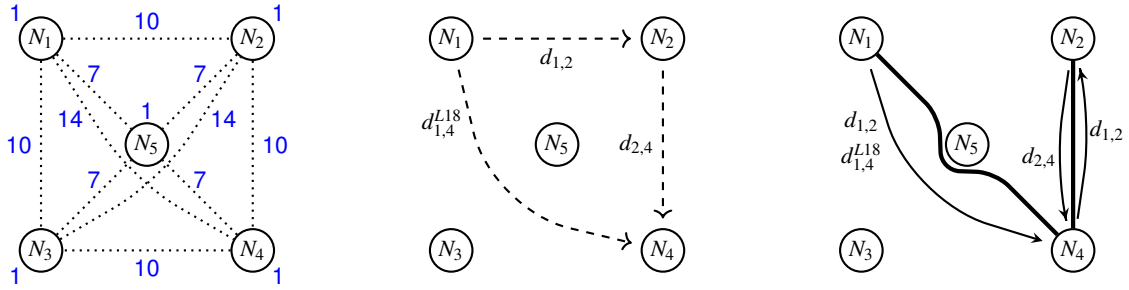
to shortest path algorithms. As the integrated approaches establish the number of circuits only after routing demands in the virtual topology, they can only make a local decision on routes based on the number of hops and the delay incurred. The virtual topology-driven approaches are therefore incapable of finding the optimum for this test case.

*Test Case* 6 *Traffic Demands with Latency Constraints*
This case is similar to test case 3, such that we have 3 demands that each require 0.5 circuit capacities and the virtual topology corresponds to the full graph. While $d_{1,2}$ and $d_{2,4}$ are identical, the difference is that the demand from $N_1$ to $N_4$ now has a SLA specifying a worst-case latency of 18 units, which is indicated as $d_{1,4}^{L18}$. In the previous examples we considered an intermediate node to contribute 1 unit of delay. More precisely, we now consider the line cards to be the source of this delay, such that the delay at intermediate nodes can be divided into 0.5 units for the incoming and 0.5 units for the outgoing line card. Consequently, demands now experience 0.5 units of delay at their line cards of origin and destination as well. For this case the objective function is $F_{QoS}$.

Therefore, in contrast to case 3, the conflict between resource efficiency and minimum latency should now be decided in favor of latency by using a bypass circuit. The resource-optimal route via $N_2$ violates the latency constraint of 18 units since it incurs 10 units per circuit through propagation delay, 0.5 units at the source and at the destination line cards and 1 additional unit for processing and forwarding at $N_2$, totaling 22 units. A route terminating at $N_5$ as intermediate node would reduce the delay significantly, as the diagonal links only result in 7 units of delay. This would result in an overall delay of 16 units. Using the bypass circuit, however, not only reduces resources at $N_5$, but also eliminates the associated delay leading to a total latency of 15 units. Since honoring SLA-parameters is valued higher than resource efficiency in $F_{QoS}$, usage of the diagonal links is mandatory for all optimal solutions.

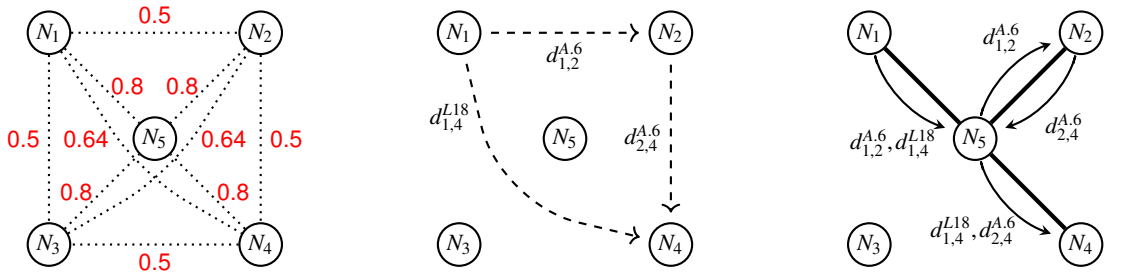
This changes the situation for the remaining two demands. As they do not have any latency constraints, routing each of them on their own circuits, like before, would be a waste of resources. Consequently, the algorithms should create only one additional circuit, either between $N_1$ and $N_2$ or between $N_2$ and $N_4$. The other demand should be groomed into the existing circuits using the remaining capacity; thus, there are two optimal solutions of 4 line cards each. Figure 5.8c illustrates the latter solution. All algorithms and encodings succeed in finding one of these optima.

*Test Case* 7 *Traffic Demands with Availability Constraints*
This scenario differs from the previous by added availability constraints. As before, there are




(a) Topology and delays (b) Demands (c) Optimal solution

Figure 5.8: Illustration of Test Case 6 showing effects of a latency-constrained demand




(a) Topology and availabilities (b) Demands (c) Optimal solution

Figure 5.9: Illustration of Test Case 7 showing effects of a availability-constrained demand

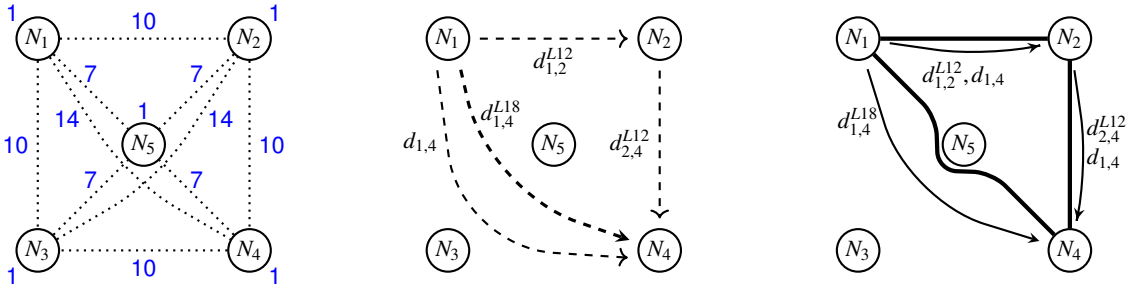
three demands that each require 0.5 circuit capacities. The demand $d_{1,4}^{L18}$ remains unchanged. The demands $d_{1,2}^{A.6}$ and $d_{2,4}^{A.6}$ require an availability figure of at least 0.6. The virtual topology and the delay figures are identical to test case 6. All physical links connecting to $N_5$ have an availability figure of 0.8, while all remaining links have a value of 0.5. Potential circuits bypassing $N_5$ would need to traverse two diagonal links leading to a circuit availability of 0.64 as shown in Figure 5.9a.

The low availability of the direct links between $N_2$ and $N_4$ as well as between $N_1$ and $N_2$ now prevents the previous solution from test case 6. The latency-sensitive demand still needs to remain on one of two diagonal routes. Since both, the bypass circuit via $N_5$ and two separate circuits with an intermediate termination at $N_5$, are below the latency limit of 18 units, the better choice is the one that offers more grooming potential. Therefore, the optimal solution will terminate at $N_5$ since both circuits can be shared with the other demands, leading to an optimal solution with 3 circuits and therefore 6 line cards as illustrated in Figure 5.9c.

Once more, all algorithms and encodings succeed in finding the optimum.

*Test Case* 8 *Multiple QoS-compliant Paths per Source–Destination Pair*

In this case there exist several demands of differing QoS constraints between the same node pair. For reasons of simplicity this is demonstrated using only latency constraints. There are four demands in total. The demands $d_{1,2}^{L12}$ and $d_{2,4}^{L12}$ are required to have latencies of at most 12 units and a data rate equivalent to half of the circuit capacity. The remaining demands both originate from $N_1$ and are destined for $N_4$, but $d_{1,4}^{L18}$ requires at most 18 units of delay and the capacity of




(a) Topology and delays (b) Demands (c) Optimal solution

Figure 5.10: Illustration of Test Case 8 showing effects of QoS-diverse demands

one circuit, whereas $d_{1,4}$ has no latency constraint and needs only half a circuit as illustrated in Figure 5.10b. The feasible topology corresponds to the full graph and $F_{QoS}$ is used.

For the first two demands there is no feasible alternative to using a direct circuit for each, due to their stringent latency requirements. The third latency-constrained demand requires an entire circuit, which offers no room for grooming, such that only a direct circuit fulfills latency and resource goals. Finally, the unconstrained demand $d_{1,4}$ should be routed such that savings through grooming are maximized. This is the case when routing it on the path via $N_2$ since the existing circuits there have spare capacity, yielding an optimal solution of 3 circuits and 6 line cards as shown in Figure 5.10c.

All of the topology-based algorithms cannot solve this problem to optimality, since their routing algorithms favor the shorter routes. While they are able to differentiate between service classes and can rule out edge weight-minimal paths when they do not conform to the QoS constraints, they are unable to make a local decision to distinguish between demands on compliant, but longer paths to facilitate a global optimum. In the present example, any feasible route for $d_{1,4}^{L18}$ is also a feasible route for $d_{1,4}$, such that they will use the same path.

The Simulated Annealing-version incorporating the routing optimization also fails to find the optimum, since there is no sequence of activation and deactivation actions, that can lead demands of the same metric between the same nodes on compliant routes of different weights. To illustrate this point, one can consider the case that two such demands exist, but no route has been found yet. When activating virtual links and a path is found such that the QoS constraints are fulfilled for both demands, both will be treated identically and routed on the same path and both will be removed when one of these links is removed. When activating virtual links such that a path emerges that fulfills the constraints of only one demand, while the more stringent constraints of the other are not fulfilled, the demands will at first be treated separately, by routing only the first. However, as soon as a an additional activation allows for a path fulfilling the constraints of the unrouted demand, this same path will also be a more beneficial path in the sense of the routing metric to the first one, which will then also be rerouted onto this path as explained in algorithm 4.5 of Feller's dissertation [77].

It should be noted that this effect does not occur for all combinations of possible QoS-parameters and link weights. There may exist situations where other factors cause a sufficiently significant difference in routing options; e. g., consider a node pair with a short path of low availability and a long path of high availability, which may occur when comparing aerial fiber deployments with buried lines. However, availability, latency and edge weights are typically

strongly correlated, since the longer a fiber is, the larger is the risk that it may suffer a fiber cut, the larger is the propagation delay and the larger are deployment and usage costs.

### 5.2.3 Summary of Results

Table 5.4: Overview of core function test results

| Test Case | | SA | | GA | | | |
|---|---|---|---|---|---|---|---|
| Nr. | Scenario | **SVT** | **SVTR** | **VTB** | **VTCS** | **RCPE** | **DCPE** |
| 1 | Shortest Path | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 2 | Bypass Circuit | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 3 | Grooming 1 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 4 | Grooming 2 | ✓ | ✓ | ✓ | X | ✓ | ✓ |
| 5 | Route Grooming | X | ✓ | X | X | ✓ | ✓ |
| 6 | Latency | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 7 | Availability | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 8 | QoS-Multipaths | X | X | X | X | ✓ | ✓ |

Legend
✓ Successfully finds optimal solution
X Cannot find the optimal solution

The results of the test cases are summarized in Table 5.4. All proposed GA-based approaches and the Simulated Annealing reference methods generally succeed in finding paths while considering QoS constraints, establish bypasses and perform some level of grooming for typical problem structures. Therefore, they are not only capable of finding solutions equal to a regular QoS-enabled shortest path routing, but also of finding solutions closer to a global optimum, due to their ability to utilize these principles. However, there are specific scenarios where some approaches only find sub-par solutions. For the VTCS encoding there exist scenarios where it is not only unable to determine the optimal solution, but in fact, may be outperformed by the trivial shortest path solution. This may occur when the topology and demand matrices become very sparse, such that a connected graph is not necessary for the optimal solution. Since this encoding has been designed for connected graphs of high density, it should not be used in such scenarios.

Further differences between the approaches emerge in the more advanced cases. Especially, when either the routing has to be adapted to create the optimal solution or when multiple demands of different QoS classes exist between the same node pairs. For SVT, VTB and VTCS, which rely on an optimization of the topology alone, the routing is not an immediate part of the optimization. Therefore, these approaches are not suitable for the more advanced scenarios, although they can still be expected to yield results, which can outperform the shortest path solution. In the scenarios where multiple routes per source–destination pair are necessary, even SVTR may be unable to obtain the desired results.

While these test cases provide a validation of core functions, they cannot be used to assess runtime performance for realistic scenarios, nor do they indicate that a meaningful level of optimization is possible for such scenarios. While improvements in runtime performance largely

depend on the genetic operators and initialization mechanisms, the encodings need to cover the search space sufficiently to include good solutions. We know the optima to the presented test cases by design, but these are of small scale and high symmetry. In order to provide a better indication towards the representable and attainable solution quality, more realistic scenarios have to be considered.

## 5.3 Solution Quality for Reference Scenario

In order to assess the capability of the proposed frameworks to achieve high-quality results at meaningful cost values for common network optimization problems, we investigate a common type of networking problem, which features a higher node count and asymmetries in the topology and traffic demands. If the optimal solution to such a problem is known, the results from the approaches can be compared to the optimum and the result of the shortest path-based approach as an upper bound. However, such problems are typically too complex to be solvable by exhaustive enumeration or other exact solution approaches.

In order to obtain a representative scenario, for which a lower bound can be given, we surveyed the SNDlib database introduced in Section 3.5.2.1 for problems and results from other researchers to compare against. As we have already explained in Section 3.5.2.2, there is no problem instance using modular link capacities and single routes for demands, that has been solved to optimality. However, there exist optimal solutions for some networks with arbitrary flow distribution, which represent dual solutions and therefore lower bounds to the problem type required for this investigation. We have selected the problem called *france--D-B-M-N-S-A-N-N*, for which such a dual solution exists, since it features a reasonably large topology size, fitting traffic and circuit parameters and is based on WDM planning data of a French network operator [183]. While this problem does not include the full intended network complexity, as it does not separate network layers, nor contain flexrate devices or QoS constraints in terms of latency or availability, it is still a viable problem to demonstrate solution quality for representative problems.

### 5.3.1 Scenario Parameters

#### *5.3.1.1 Network Topology*

The network topology shared by all problems named *France* in the SNDlib describes a physical topology of a nationwide French WDM network with 25 nodes and 45 links and is therefore a good fit for the physical layer of a transport network. The graph is shown in Figure 5.11 and an overview of typical properties is shown in Table 5.5.

Table 5.5: Parameters of SNDlib's *France* topology

| Nodes | Links | Density | Degree | | Closeness | | Betweenness | |
|---|---|---|---|---|---|---|---|---|
| | | | Max. | Avg. | Max. | Avg. | Max. | Avg. |
| 25 | 45 | 0.15 | 10 | 3.6 | 0.585 | 0.394 | 313.5 | 38.9 |

The problem description does not contain actual link lengths, and instead of geographical coordinates, the node locations are given as $x$ and $y$ integer values. We therefore infer abstract link lengths using the regular Euclidean metric. Interestingly, the graph is not a planar graph.

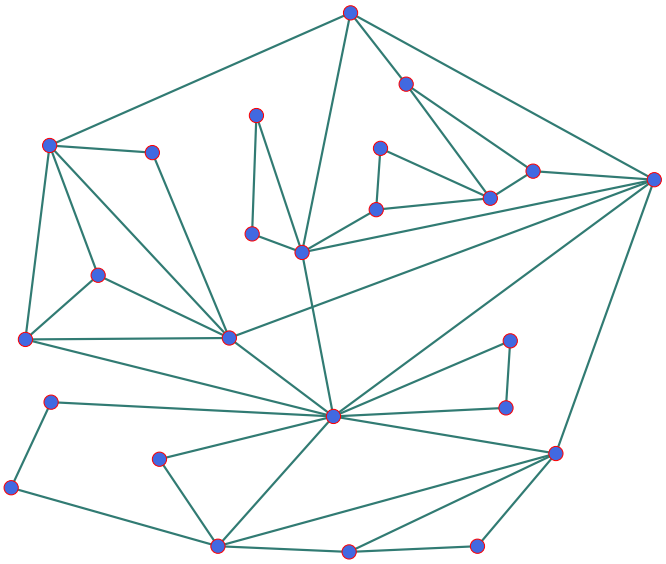


Figure 5.11: *France* topology from SNDlib

#### 5.3.1.2 Physical Parameters

SNDlib assigns capacities in the form of abstract modules, which can be installed on physical links. Each module for the *france* problems has a capacity of 2500 units and comes at a cost of 250 units. Since there is no limit on installable modules, such problems can be interpreted as uncapacitated and therefore can be classified as dimensioning problems including VTD and TDR, or alternatively PTD with TDR.

As SNDlib problems have no notion of wavelengths and modules can only be installed directly on physical links without any bypass options, ICR collapses to finding routes to next-hop neighbors, which trivially are always circuits corresponding to the direct link between these neighbors. Due to these limitations a virtual topology can only be a subset of the physical topology and therefore, VTD and PTD converge to the same problem in the absence of other constraints.

#### 5.3.1.3 Traffic Demands

The *france--D-B-M-N-S-A-N-N* problem contains 300 unidirectional traffic demands, which are not allowed to be split between different routes. The sum of all demand capacities is at 99 830 units, while the average demand capacity is 332.77 units and the maximum 1808 units. The exact distribution is illustrated in Figure 5.12. No QoS features are required by any demand.

#### 5.3.1.4 Optimization Goal

SNDlib contains minimization problems, where costs may be incurred from routing, modules, and links. The problem *france--D-B-M-N-S-A-N-N* requires all demands to be routed and costs are determined by the sum of the costs of all installed modules. We designate this cost function as $F_{SND}$. For this particular problem there is only one type of module with a cost of $\kappa_{\text{mod}} = 250$ as mentioned above. We consider such a module to be equivalent to our single-port line card with integrated TXP. We also relax the hard demand routing constraints into soft constraints to facilitate search space exploration as explained in Section 3.2.3. The relaxed cost function

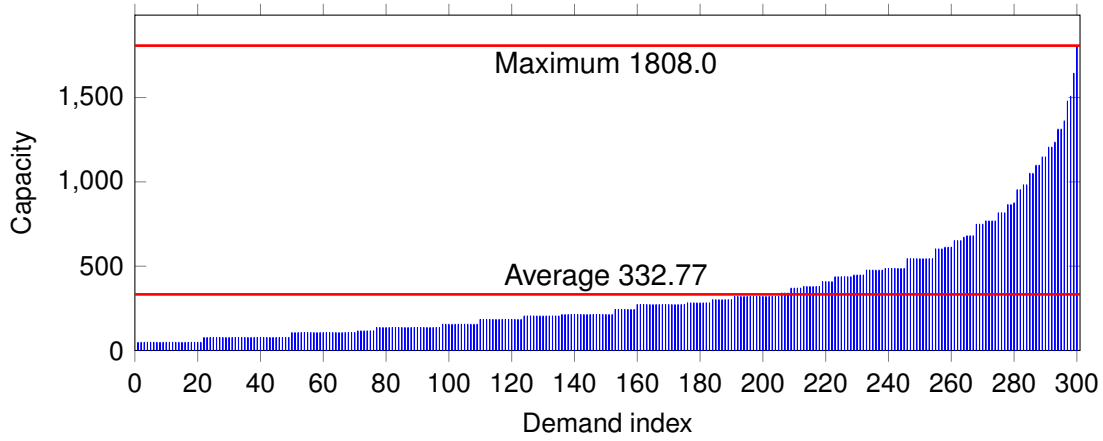



Figure 5.12: Distribution of capacities for traffic demands in *france--D-B-M-N-S-A-N-N*

$F_{RSND}$ heavily penalizes the number of unrouted demands $n_{\mathrm{UD}}$ and the sum of all unrouted traffic capacities $n_{\mathrm{UC}}$ separately. We will therefore use $F_{RSND}$ as the objective function for the optimization, but we will give the results as the value of $F_{SND}$. If all demands are routed successfully, the values of both cost functions will be identical. The objective functions are defined as

$$F_{SND}(s) = \sum_{v \in V} n_{\mathrm{LC},v} \cdot \kappa_{\mathrm{mod}} \tag{5.3}$$

and

$$F_{RSND}(s) = \sum_{v \in V} (n_{\mathrm{LC},v} \cdot \kappa_{\mathrm{mod}}) + n_{\mathrm{UD}} \cdot \alpha + n_{\mathrm{UC}} \cdot \beta \tag{5.4}$$

with

$$\alpha > \beta$$
$$\beta > \kappa_{\mathrm{mod}}$$
$$\alpha + r_{\mathrm{min}} \cdot \beta \gg |E| \cdot \kappa_{\mathrm{mod}}$$
$$\alpha, \beta \in \mathbb{R}_{+}$$

such that the penalty of not routing the smallest demand capacity $r_{\mathrm{min}}$ is always higher than the cost for the longest possible path.

#### *5.3.1.5 Algorithmic Parameters*

Since we focus on high solution quality, we provide each of the metaheuristics ample time to determine the best solution possible and refrain from utilizing any performance enhancement, that may compromise search space exploration. All heuristic approaches use the same termination condition, which is a runtime limit of 12 h.

For this test we also include the shortest path reference approach, which does not require a time limit. We employ it in two different versions by altering the metric for the routing algorithm. The first metric is the regular one also used in the virtual topology-based approaches, where we employ the number of hops with the length of the path as a tie breaker. Henceforth we shall refer to this as the *default metric*. For the second metric, we used only the number of hops, but randomized the iteration order of the edges. Since many paths have identical number of hops, the randomization can lead to very different solutions. In contrast to the other approaches, we did this not for 10, but for 1000 randomizations.

For Simulated Annealing, we did a large number of tests to obtain good hyperparameters for both approaches. Given the very long runtime, the cooling schedule has to be drastically increased to divide the time meaningfully between the more exploratory and more refining stages. This highlights another difficulty in employing Simulated Annealing with runtime limits. Interestingly, we found that the SVT approach required a much slower cooling schedule than the SVTR approach to provide the most consistent high-quality solutions. The exact values used for both are given in Table 5.6.

Table 5.6: Simulated Annealing hyperparameters for SNDlib scenario

| Type | Initial Topology | Initial Temp. | Cooling Factor | Iterations per Temp. | Accepted per Temp. |
|---|---|---|---|---|---|
| SVT | Physical | 4000 | 0.9999 | 10000 | 1000 |
| SVTR | Physical | 2000 | 0.95 | 6000 | 400 |

The GA-approaches were also tested using a wide variety of parameters. We chose large populations and numbers of offspring to facilitate a thorough exploration of the search space. Due to the long runtimes, we also included an age-bias factor in the survivor selection scheme, which favors removing older individuals when cost values are very similar. The best individual is preserved by the elitism rule. Furthermore, all approaches use sigma truncation with 2 sigmas. The simulation software runs the evolutionary tasks on 20 threads in parallel. VTB uses the advance check mechanism without repair and RRM with a mutation rate of 0.02. The path enumeration approaches do not employ any scaling techniques, such that each uses $k_{\mathrm{m}} = 1000$ shortest paths. The chosen parameter combinations are summarized in Table 5.7, where Creep uses a step size of 1 and NPXO uses 3 crossover points.

Table 5.7: Genetic Algorithm hyperparameters for SNDlib scenario

| Type | Population | | | | Selection | | Recomb. | Mutation |
|---|---|---|---|---|---|---|---|---|
| | Init. | Size | Offsp. | Control | Parent | Survivor | | |
| VTB | ASTI | 1e4 | 1000 | No | Tourn. 10 | $(\mu+\lambda)$+A | NPXO | RRM |
| VTCS | Rand. | 1e4 | 1000 | No | Tourn. 5 | $(\mu+\lambda)$+A | NPXO | Creep |
| RCPE | RDRI | 1e4 | 1000 | No | Tourn. 10 | $(\mu+\lambda)$+A | NPXO | Creep |
| DCPE | RDRI | 1e4 | 1000 | No | Tourn. 5 | $(\mu+\lambda)$+A | NPXO | Creep |

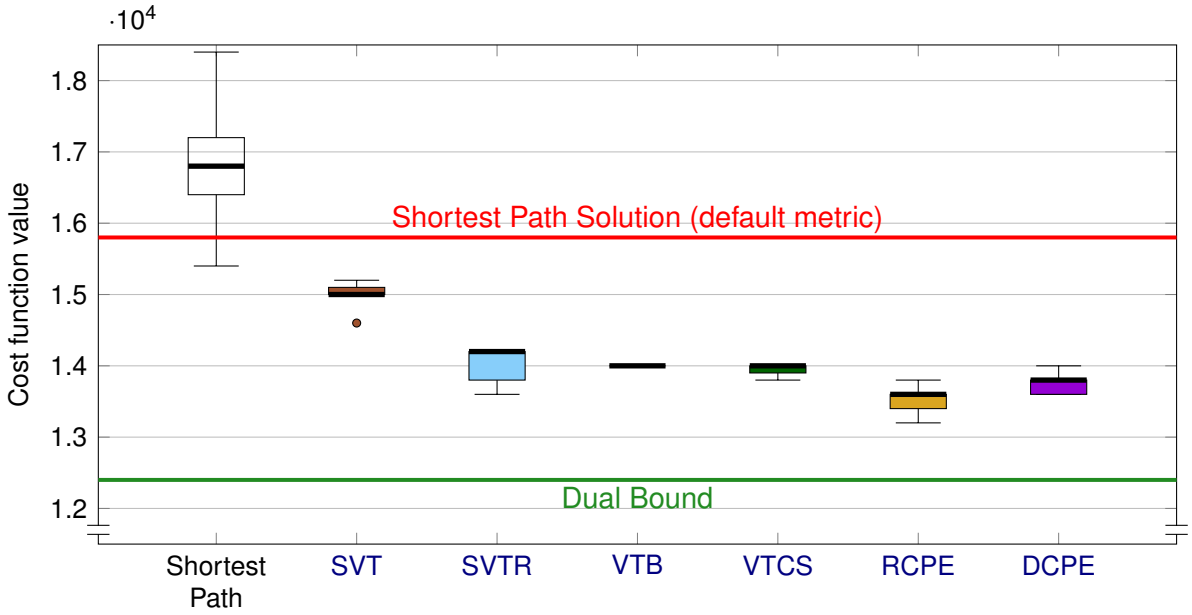



Figure 5.13: Cost function values for different methods after 12 hours of optimization

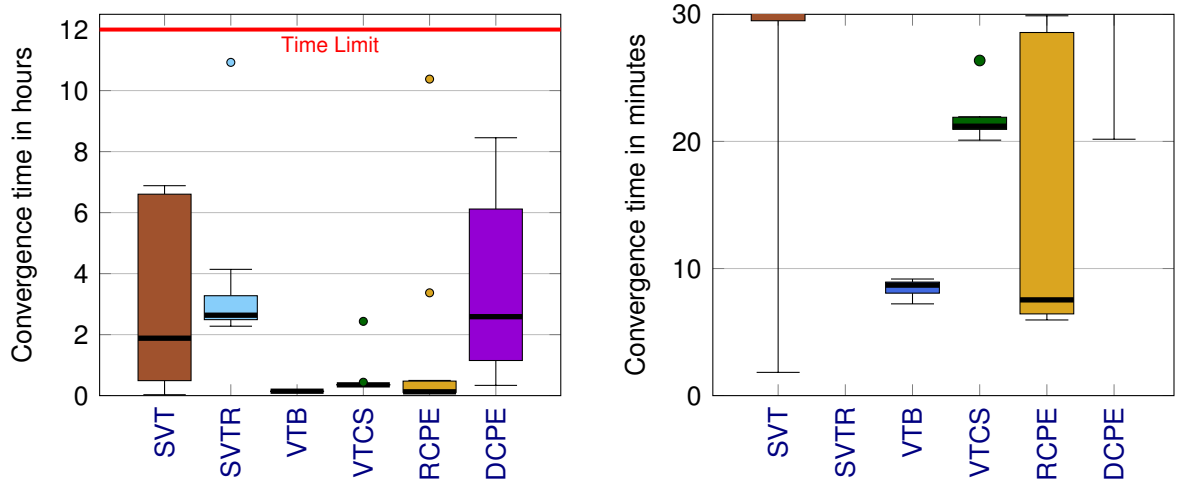
### 5.3.2 Evaluation

Figure 5.13 shows the best obtained cost values for each of the approaches, visualized as box plots[1]. The upper, red line marks the result of the shortest path reference approach using the default metric as explained in the previous section. The lower, green line is the dual bound solution obtained from the SNDlib database. The results of the randomized shortest path approach show significant deviation, which is in fact larger than for any of the metaheuristic approaches. Interestingly, there were even routings which surpass the default metric in solution quality, although not by a large margin. A second observation is that controlling the sequence, in which shortest paths are routed, as used as the core of optimization by some works, is not very promising for the given problem.

Both Simulated Annealing approaches achieve better cost values than any of the shortest path routings. SVT is more consistent than SVTR, but the solution quality is lower. More than half of the simulations converged at a cost value of 15 000, but the single outlier at 14 600 shows that there is still room for improvement for a purely topology-based method. The much better performance of SVTR may be explained by the fact that it can also affect routes, covering a larger area of the search space including these better solutions. However, it is more likely that the reduced amount of changes from one perturbation to the next allows for a more gradual refinement, where small improvements point towards more beneficial areas. While the median value of 14 200 is only a 10 % improvement over the shortest path reference, it already exploits almost half of the theoretical improvement potential towards the lower bound.

The results for all GA-based optimizations are very consistent. Most remarkable in this respect is the result of the VTB encoding, where all independent runs converged to the same cost value of 14 000. VTCS shows, that this value is still not the lowest for a topology-based encoding, as it manages to reach a value of 13 800. Both GA-based topology optimizations show better median values than SVTR, but the latter included the best solution of the topology-based approaches.

[1]For more information on reading and interpreting box plots, see section D.3 in the appendix.




(a) Full 12-hour range

(b) Zoomed view of initial 30 minutes

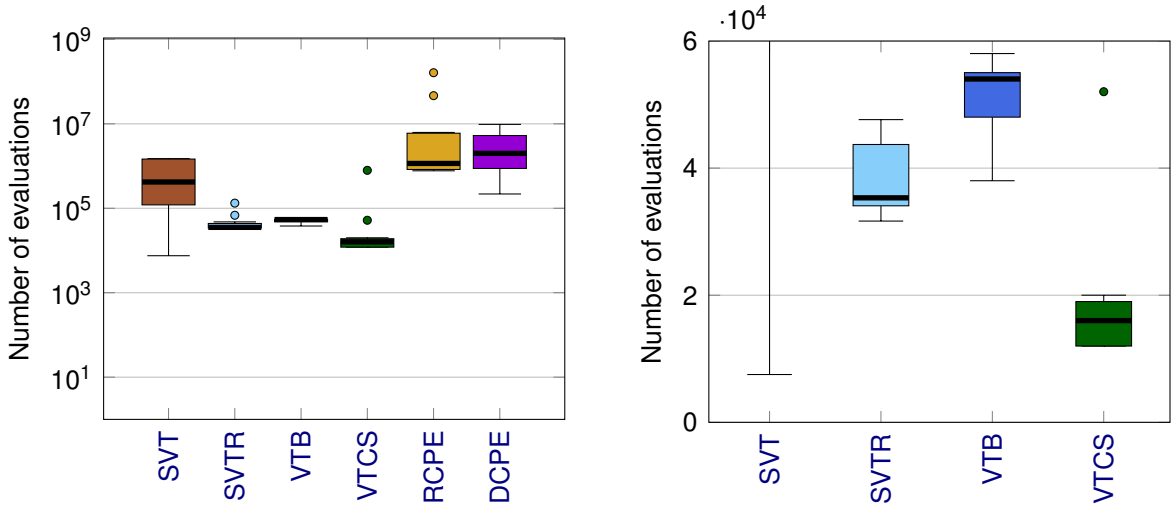
Figure 5.14: Duration until the optimization could not further improve the objective value

Finally, the routing optimizations using GA yield the best results among all of the approaches, with their median values surpassing the medians of all other approaches. In fact, the best result for RCPE is less than 10 % worse than the unachievable dual bound solution, exploiting the largest share of the headroom. However, it should be noted that the best results for the topology optimizations can compete with the worse results from the routing approaches. Surprisingly, DCPE underperforms compared to RCPE, although in the absence of QoS parameters and the number of demands corresponding to the number of node pairs, both of these approaches should have almost equal performance.

Figure 5.14 shows the duration after which the individual approaches first encountered a solution of an objective value equal to the final value after 12 hours. The left figure shows the full range of 12 hours, while the right figure shows a zoomed view of the initial 30 minutes of these 12 hours. Since the purpose of this analysis was to obtain the best solution quality, the algorithm parameters have not been tuned for short runtimes, but rather to make the most of the given 12-hour interval.

For Simulated Annealing this means using a very slow and steady cooling schedule to avoid premature convergence. Consequently, the corresponding optimizations show higher median values for convergence time, especially for SVT, which utilizes a slower schedule than SVTR. Another explanation for the difference might also lie in the full rerouting SVT needs to perform in every perturbation. This can make the search space more uneven and harder to probe, such that SVT is more susceptible to local minima making premature convergence more likely, while also requiring more steps to refine a good solution. While the best solution found exhibited the largest runtime, most other good solutions were below the median time. For SVTR the best solutions are close to the median time, while the outlier in time is about average in result. Note that SVTR is not visible in Figure 5.14b, since its smallest value is larger than the scope of the figure.

The GA-based approaches have a tremendous advantage in runtime due to their parallel execution. Their drawback is that they cannot converge very early in case of simple problems, because initializing the population also takes time. For the relatively large populations in the present example, this amounts to about 300 s for VTB, 1200 s for VTCS, 100 s for RCPE and




(a) Full range

(b) Zoomed view of lower values

Figure 5.15: Number of evaluations until the optimization reached the final objective value

120 s for DCPE. Remarkably, the median convergence time for VTB, VTCS and RCPE is less than 30 minutes, with the maximum for VTB being below 10 minutes. However, VTCS and RCPE have outliers reaching almost 3 and 11 hours, respectively. Very surprising is the comparatively long runtime of DCPE. While the median runtime is still on par with the Simulated Annealing approaches, the large distribution of values suggests that convergence is difficult to achieve for this approach.

Figure 5.15 provides a different perspective on the computational effort by comparing the number of evaluations until the solution cost could not be improved any further. For Simulated Annealing every perturbation comes with an evaluation, whereas for the GA approaches, two evaluations are performed for each execution of the evolutionary process, such that one generation requires an evaluation for each offspring. This perspective removes the overhead of the population initialization, but it allows to compare Simulated Annealing and GA approaches without any distortion from the parallel execution.

This shows some interesting differences to the runtime graphs. Comparing SVTR to VTB in Figure 5.15b, we can notice that the first approach actually requires less evaluations than the GA-based one, although it has a drastically higher runtime as shown in Figure 5.14a. Considering that a single decoding and evaluation for VTB is much more computation-intensive than a SVTR perturbation, indicates the level of acceleration contributed by the parallel execution alone.

Comparing VTB to VTCS we can also observe an inverted behavior to the runtime statistics, where VTCS requires much fewer evaluations, but the more complex decoding results in a larger runtime. Note, that Figure 5.15a is a log scale plot and that the zoomed view in Figure 5.15b does not show all outliers, nor does it include the routing approaches, which require much larger numbers of iterations to converge.

### 5.3.3 Summary of Results

The evaluation using the reference scenario has shown, that the suggested approaches are applicable to scientifically relevant network optimization problems. Furthermore, all metaheuristics employed are able to surpass the solution quality of the shortest path reference approach. Even the best hop-oriented routing solution is still worse than the worst metaheuristic solution, demonstrating the superiority of optimized configurations. The suggested GA-based encodings were consistently able to obtain better results than a simple Simulated Annealing approach and were either on par or even better than the more advanced SVTR approach, which has been proven to be effective in complex multi-layer scenarios.

The routing-based optimizations showed the best overall cost values, as expected. RCPE obtained the best cost value, which is within 10 % of the unobtainable dual bound solution, such that the headroom for improvement can be expected to be well-exploited. It should also be noted that the virtual topology approaches are rather close to the route encodings in cost values, but achieve this within a much lower time frame, hinting at their potential for scalability.

The acceleration by parallel execution of the evolutionary process within the framework, greatly boosts performance. This is especially clear for VTB in comparison to SVTR, given that the GA approach needs more evaluations, but only a fraction of the runtime of the Simulated Annealing approach. However, VTCS seems successful in exploring the search space efficiently, such that it would outperform SVTR even without a parallel implementation. Finally, we also found the parameter tuning for the GA less complicated than for Simulated Annealing. While the GA approaches have a larger number of parameters, their effects are more localized, whereas adjusting the cooling schedule ideally requires knowledge on the entire runtime of the algorithm to get consistent results without premature convergence.

## 5.4 Performance Improvements for QoS-enabled Scenario

While the framework and the developed encodings have proven to be capable of working with QoS constraints and capable of achieving good results in a reference scenario, the next step is to combine a realistic scenario with realistic constraints. To the best of our knowledge, there are no comprehensive reference scenarios that include all the parameters required for this type of evaluation. We have therefore defined such a scenario, consisting of a network topology, based on an actual transport network, traffic demands, based on an established forecast of traffic patterns, and a flexrate TXP, based on data of commercial TXP.

Since this part of the evaluation is more focused on runtime performance, we have set a more rigorous time limit of 1000 seconds, which is closer to the time scales of network operation as explained in Section 2.2.1.1. In order to determine the relative performance gains of the suggested approaches, we will vary the selection of components used in the framework. Starting from a baseline GA approach, we apply each of our suggested mechanisms individually, followed by a combination of the most successful components. We also show the results obtained from the three reference methods, demonstrating that relevant gains in objective values are possible even in the given time frame.

### 5.4.1 Scenario Parameters

#### *5.4.1.1 Network Topology*

We have designed two transport network topologies based on publicly available data on different networks of a large NSP [32]. The network topology *CL–DC* connects 19 core locations, at

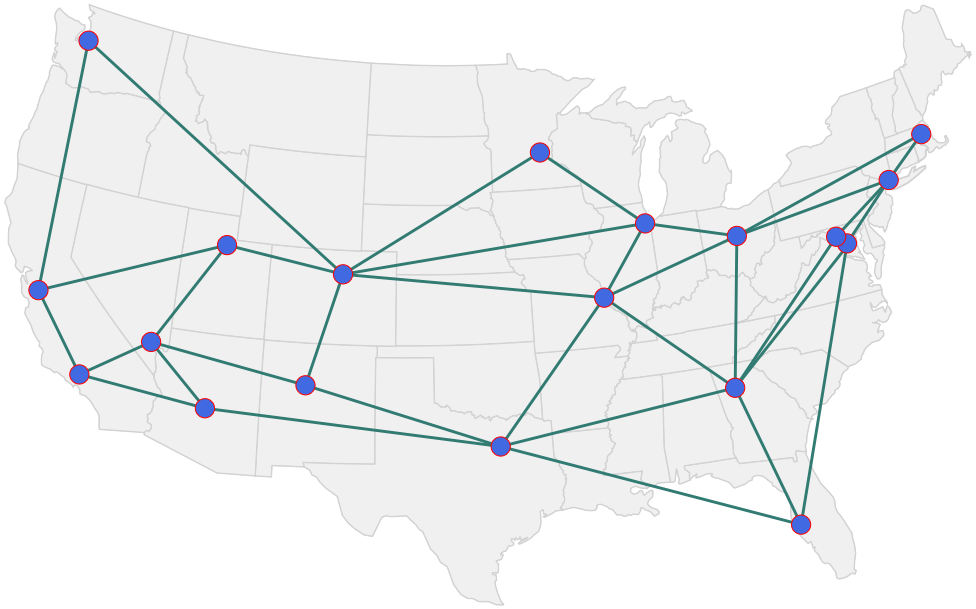


Figure 5.16: *CL–DC* topology spanning the continental USA

which the NSP operates data centers[1]. The links have been derived from fiber infrastructure data of the same NSP. The resulting topology is shown in Figure 5.16. Since the available data does not contain information on fiber lengths, we based the lengths of our links on the orthodromic distance between the city centers. More information on the design process can be found in Section B.3 of the appendix.

We compare the basic parameters for the physical *CL–DC* topology with that of a full graph of the same nodes in Table 5.8, since the virtual topology can indeed become a full graph. Note, that betweenness centrality is zero for the full graph, since any shortest path never needs to incorporate any other node than source and destination. When considering the other known transport networks from Section 2.2.2.1, it is very similar, although it remains on the lower end in node count, while featuring slightly elevated degree and centrality values.

Table 5.8: Parameters of the *CL–DC* topology

| Topology | Nodes | Links | Density | Degree | | Closeness | | Betweenness | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Max. | Avg. | Max. | Avg. | Max. | Avg. |
| Physical | 19 | 35 | 0.2 | 6 | 3.68 | 0.529 | 0.402 | 90.8 | 27.7 |
| Full | | 171 | 1 | 18 | 18 | 1 | 1 | 0 | 0 |

The proposed *CL–DC* topology is not only realistic, because it is based on a real network, but it is also a good candidate for a QoS-based optimization, since inter-data center traffic flows are likely to carry business-relevant traffic. Since the topology also covers a large, geographically diverse area, it offers interesting opportunities for bypass links and routing alternatives alike, since longer routes will result in noticeable differences in latency and availability. Figure 5.17 shows the potential virtual links of the full graph in blue, while the links of the physical *CL–DC*

[1]The NSP actually operates a total of 57 data centers, but metropolitan areas have been aggregated.

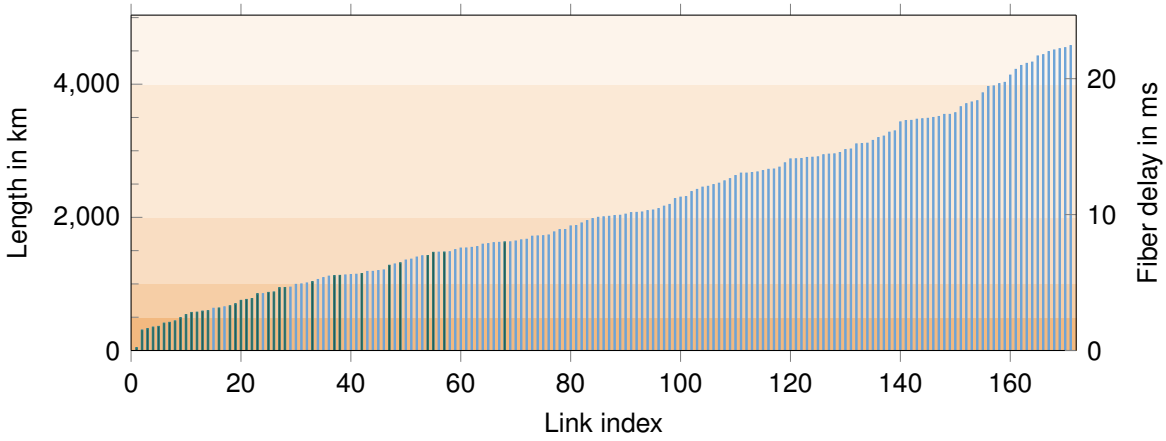



Figure 5.17: Distribution of link lengths in the *CL–DC* topology

topology are green in color. Note, that the indicated delay is purely based on propagation delay in the fibers. Any intermediate termination would add additional delay.

#### *5.4.1.2 Physical Parameters*

This scenario includes both, latency and availability figures. As explained in Section 4.1.1, we incur a delay proportional to the fiber distance and a fixed delay for every node traversed. We chose the fiber delay to be $\delta_f = 4.8985\,\mu s/km$, based on values for a common fiber brand [51], and the delay per hop $\delta_h = 1\,ms$, which we consider to be a meaningful upper bound for what can commonly be expected[1].

The availability of a fiber is also determined by a factor, which scales the figure with distance. We set this factor to $a_f = 2.55e{-}6\,km^{-1}$, which translates to an availability of 0.99745 for 1000 km of fiber. This value is based on the observation that long-haul networks like transport networks suffer about 1 to 5 fiber cuts per 1000 miles annually [62, 167, 263] with a reported mean time to repair of 12 hours [263].

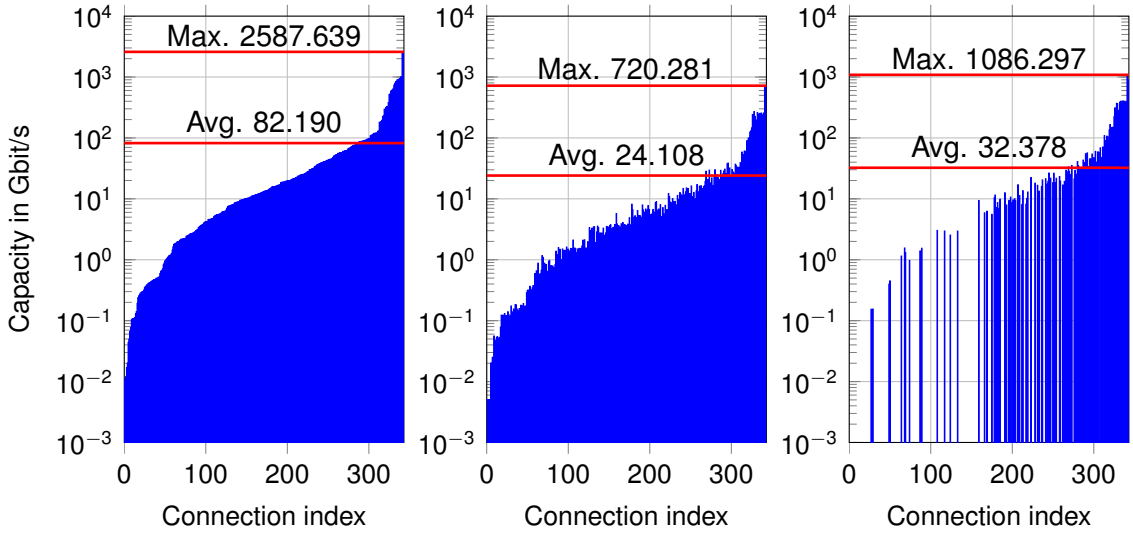
Regarding hardware, we consider a line card to be able to serve up to four TXPs. We consider full band-tunable flexrate TXPs as explained in Section 2.1.3 and that every fiber can support 80 wavelengths. Based on the data rate and transparent reach values of the different commercial devices shown in Figure 2.5, we define the different modes of the TXP under consideration as listed in Table 5.9.

Table 5.9: Flexrate modes of the suggested TXP

| Data Rate in Gbit/s | 50 | 100 | 150 | 200 | 250 |
|---|---|---|---|---|---|
| Transparent Reach in km | 8000 | 4000 | 2000 | 1000 | 500 |
| Active for % of Virt. Links | 7.6 | 42.1 | 32.2 | 12.8 | 5.3 |
| Active for % of Phy. Links | – | – | 28.6 | 45.7 | 25.7 |

The percentages in the lower half of the table show the fraction of links for which the given mode has the highest possible transmission rate. The color code of the table matches the

[1] Cf. Section 2.2.3.1 for details on sources and magnitudes of delays in NEs.




(a) Best effort class (b) Availability class (c) Latency class

Figure 5.18: Capacity distribution for traffic demands of different QoS classes for *CL–DC*

background of the respective links in Figure 5.17. Note, that the lowest data rate for any physical link in *CL–DC* is still 150 Gbit/s and that the maximum transparent reach of 8000 km permits a full graph as the virtual topology.

#### *5.4.1.3 Traffic Demands*

In order to obtain meaningful traffic demands, we have utilized a traffic generation method developed by Enderle and the author in previous works [72]. It considers expected traffic shares derived from Cisco's estimates [39, 40] and results [240] of the DISCUS project [241], as well as population data [6, 7], and time zone information.

We chose to use three different QoS-classes for this evaluation, one with a latency constraint of 10 ms, one with an availability constraint of 0.99 and a best-effort class. This results in 812 unidirectional demands, which can also be considered as 406 bidirectional traffic demands with non-symmetrical data rates. Splitting of demands is not allowed in routing. The sum of all traffic demands amounts to 47.4 Tbit/s. With 59.3 %, best-effort traffic marks the largest share of the overall capacity, followed by 23.3 % latency and 17.4 % availability-constrained traffic. Figure 5.18 shows the demands, grouped by a connection index, i. e., a unique index for each pair of source and destination nodes corresponding to the unidirectional flows. The index is ordered by ascending capacity requirements for demands between the pairs in the best effort traffic class. Note, that Figure 5.18c does not feature demands for each possible interconnection, since there are node pairs for which a latency of 10 ms is impossible to achieve due to their distance and accordingly long fiber lengths.

#### *5.4.1.4 Optimization Goal*

The objective function for *CL–DC* is $F_{QoS}$, defined in Equation (5.2), from the initial small-scale example. While the scenario includes the hardware previously explained, there is only little to gain from including line cards explicitly here, since we treat the present problem as being

uncapacitated, such that fractured resource configurations are only a minor issue, applying to at most one line card.

#### *5.4.1.5 Algorithmic Parameters*

Since the present problem is limited to a runtime of 1000 seconds, the parameters of the Simulated Annealing reference approaches have been adjusted to provide a cooling schedule matching this time frame. However, we will add a second version to each approach, where we use a randomly generated initial solution. The purpose of this is to highlight the influence of this starting point. The exact parameters are listed in Table 5.10, where the version column contains a name for the respective parameter set.

Table 5.10: Simulated Annealing hyperparameters for QoS scenario

| Type | | Initial | Initial | Cooling | Iterations | Accepted |
|---|---|---|---|---|---|---|
| Approach | Version | Topology | Temp. | Factor | per Temp. | per Temp. |
| SVT | Rnd | Random | 1 | 0.95 | 1000 | 100 |
| SVT | SPR | Physical | 1 | 0.95 | 1000 | 100 |
| SVTR | Rnd | Random | 1 | 0.95 | 1000 | 100 |
| SVTR | SPR | Physical | 1 | 0.95 | 1000 | 100 |

For the GA-based approaches we will investigate the effects of the initialization, mutation and, recombination, as well as a combination of the most successful components. Each will be evaluated against the combination of basic components. We used known good hyperparameters for each component, which will be detailed in the evaluation part. The basic hyperparameters for the GA are chosen to be identical for each of the combinations. The population size is fixed at 400 individuals, from which 100 offspring are created with every generation in a steady-state model. Parent and survivor selection are both performed by tournament selection, each held with 10 contending individuals. All other hyperparameters and components are subject to variation for evaluation.

### 5.4.2 Evaluation

All encodings are paired with different genetic operators and other components in order to gauge their individual efficacy and contribution to the attainable solution quality in isolation, before considering the effects of combinations of such components. The hyperparameters for the individual components are chosen identically for each component to enable a fair comparison. They were determined by prior parameter exploration and are the best known to us.

#### *Virtual Topology Binary Encoding*

First, we present the results for different combinations of the VTB encoding. The exact combination of the primary components is listed in Table 5.11, where the version column provides a concise designation for the respective combination. The column entitled “Control” indicates, whether population diversity control measures are active. The hybrid initialization uses a mix of ASTI and APTI with an activation probability of 70 %, while also including a solution only consisting of the physical links and another solution considering all virtual links to be active.

Table 5.11: Genetic Algorithm hyperparameters for the VTB encoding

| Type | | Population | | Recombination | | Mutation | Squash | Repair |
|---|---|---|---|---|---|---|---|---|
| Enc. | Version | Init. | Control | Type | Param. | | | |
| VTB | Basic | Random | No | NPXO | 3 | RRM | No | No |
| VTB | Squash | Random | No | NPXO | 3 | RRM | Yes | No |
| VTB | SqRp | Random | No | NPXO | 3 | RRM | Yes | Yes |
| VTB | EBXO | Random | No | EBXO | 2, 0.95 | RRM | No | No |
| VTB | LBXO | Random | No | LBXO | 2, 0.95 | RRM | No | No |
| VTB | H-Init | Hybrid | No | NPXO | 3 | RRM | No | No |
| VTB | E/R | Hybrid | No | EBXO | 2, 0.95 | RRM | Yes | Yes |
| VTB | E/R/C | Hybrid | Yes | EBXO | 2, 0.95 | RRM | Yes | Yes |
| VTB | L/R | Hybrid | No | LBXO | 2, 0.95 | RRM | Yes | Yes |
| VTB | L/R/C | Hybrid | Yes | LBXO | 2, 0.95 | RRM | Yes | Yes |

Including the physical topology solution means that the results of the GA can never be worse than the shortest path reference solution based on this topology. Regarding crossover operators, the parameter for NPXO is the number of crossover points, while for EBXO and LBXO the parameters given are $w$ and $p_l$.

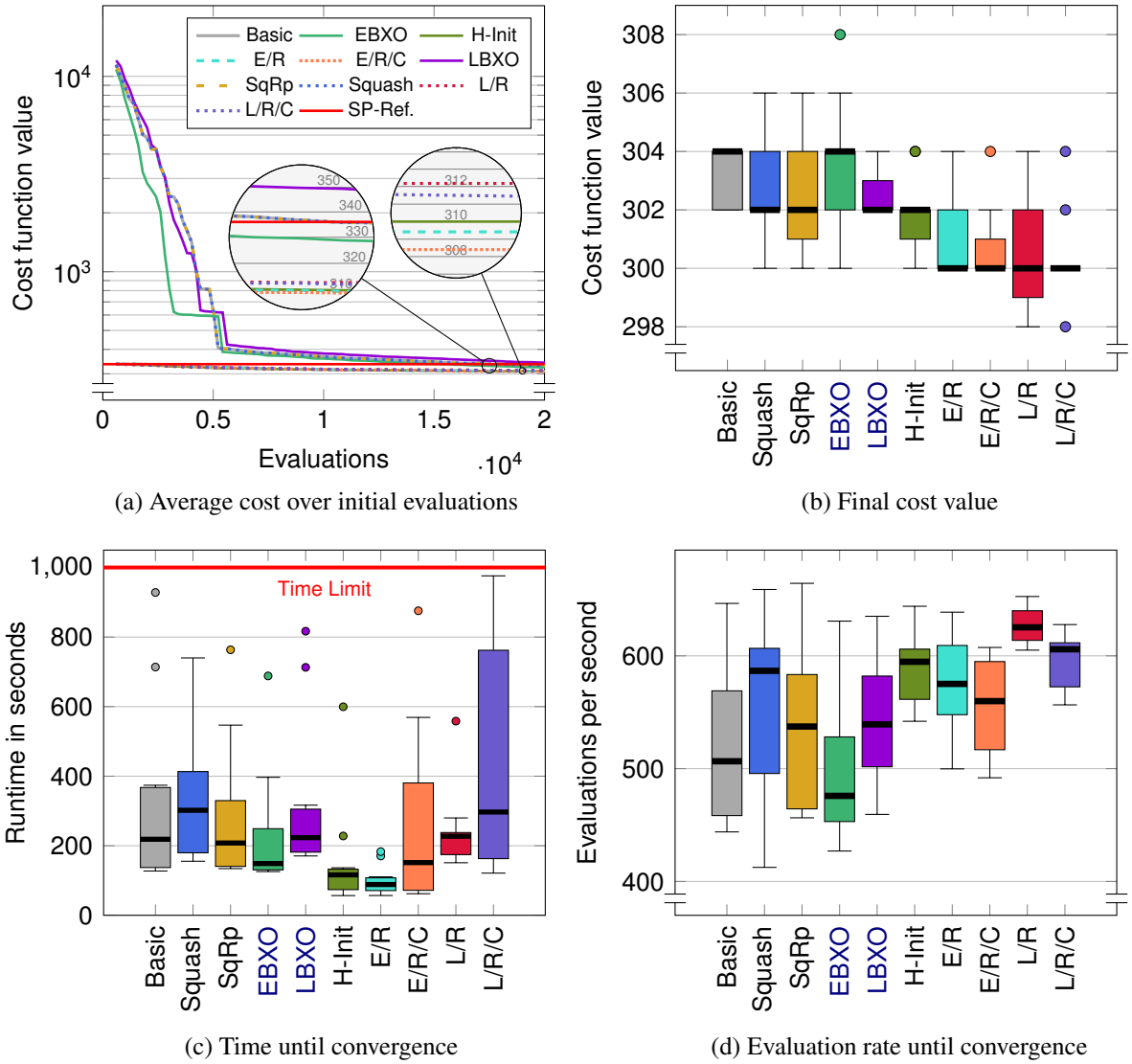
Figure 5.19a shows the average cost values for each combination during the first 20 000 candidate evaluations out of a total of between 550 000 and 660 000. The difference in the number of total iterations is caused by two effects. The optimization was run with a time limit and faster approaches can test more candidates in the same time frame. The second effect is that an evaluation is not a constant time operation, e. g., due to solutions with larger number of virtual links resulting in more complex traffic and circuit routing tasks.

The large difference in cost value between the first random initializations and the shortest path reference solution, which is indicated by the red line “SP-Ref” at a cost value of 336, is caused by the fact that many random topologies incur penalties for violated QoS limits. Due to the high number of potential virtual links in relation to the number of nodes, all random topologies happened to be connected, but this is generally not guaranteed.

It can be observed in the first magnified area of Figure 5.19a, that all approaches outperform the shortest path reference solution within these first iterations. The sole exception is LBXO, which catches up with the shortest path solution after an average of about 26 500 evaluations. The versions Basic, SqRp and Squash are exactly identical in the shown area and only diverge after about 23 100 evaluations on average. As can be seen in the second magnified area, the E/R/C combination is initially the best among the ones using the hybrid initialization.

Figure 5.19b shows the final cost values obtained after the entire runtime. The basic variant turns out to be the most consistent with all runs having either 304 or 302 as the final cost value. Squash and SqRp show better median values, but less consistency. The approach also seems successful in terms of evaluation performance, as Figure 5.19d shows that it generally increases for Squash, while additionally using the repair function in SqRp reduces this gain.

EBXO accelerates early improvements as expected, with the average outperforming all other randomly initialized combinations as shown in Figure 5.19a, but it can lead to premature convergence. The LBXO operator seems more balanced and improves upon the cost values of the basic version, moving the median value to 302, but it is incapable of reaching lower values on




(a) Average cost over initial evaluations

(b) Final cost value

(c) Time until convergence

(d) Evaluation rate until convergence

Figure 5.19: Evaluation results for combinations of the VTB encoding

its own. When combined with the hybrid initialization and the squash/repair feature within L/R, however, it shows the best evaluation performance and when additionally including population diversity control, then designated as L/R/C, it shows very good and consistent cost values at the drawback of a less consistent convergence time[1] as shown in Figure 5.19c.

### *Centralized Spanning Tree Encoding*

For VTCS we have a larger choice of components and consequently a larger number of combinations, which are listed in Table 5.12. The parameters for EBXO and VSXO are $w$ and $p_l$. Regarding mutation parameters we specify the increment for the Creep operator and for VSM the underlying mutation operator and the tree-to-augment ratio. The hybrid initialization uses the same mix as in the case of VTB, but depending on the exact graph and the resulting betweenness values, there is a chance that it is impossible to represent the physical topology as explained

[1]Note, that in the context of this evaluation, convergence time is the time until no further improvement in the objective value was possible within the limited runtime.

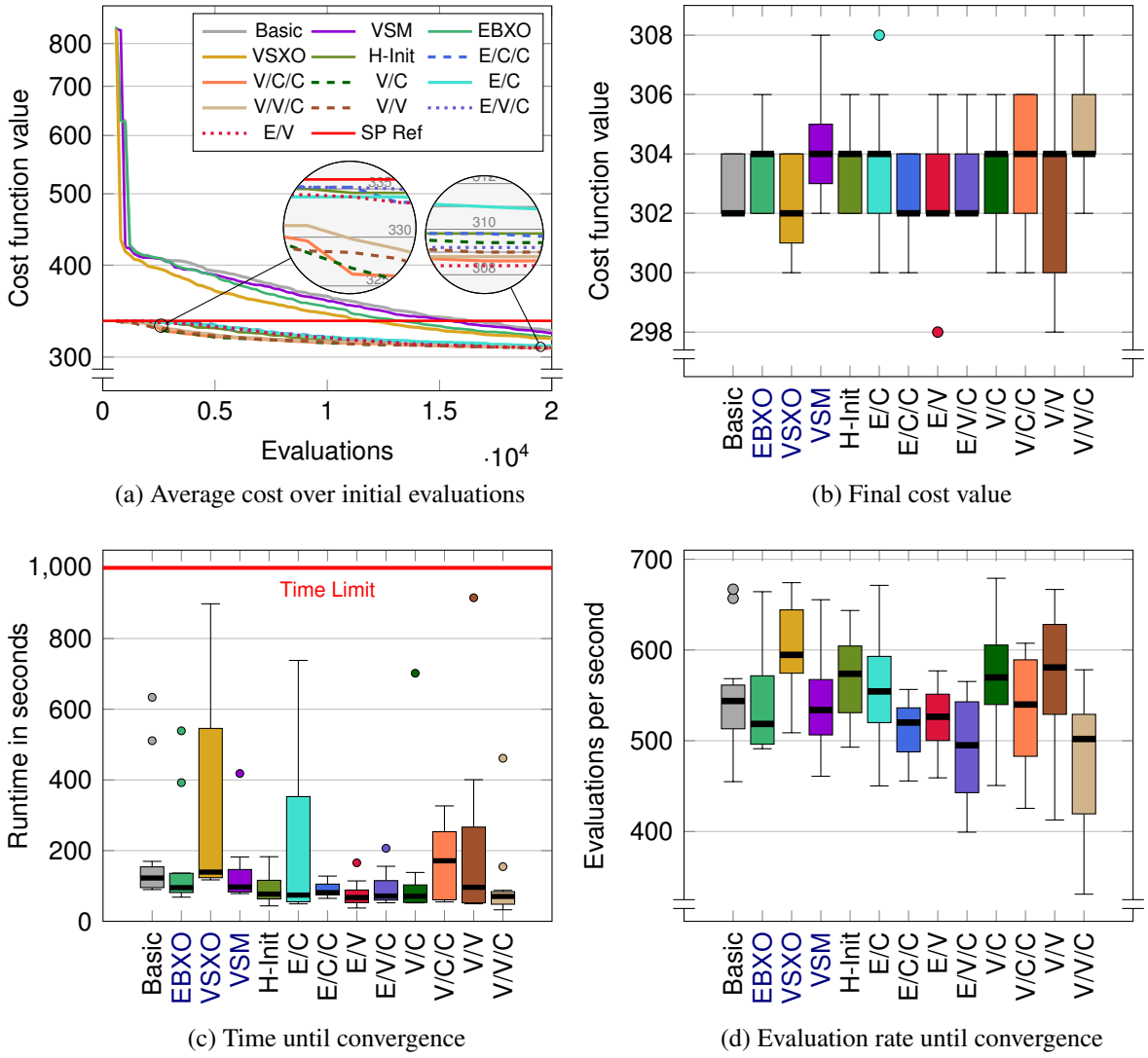
Table 5.12: Genetic Algorithm hyperparameters for the VTCS Encoding

| Type | | Population | | Recombination | | Mutation | |
|---|---|---|---|---|---|---|---|
| Enc. | Version | Init. | Control | Type | Param. | Type | Param. |
| VTCS | Basic | Random | No | NPXO | 3 | Creep | 1 |
| VTCS | EBXO | Random | No | EBXO | 2, 0.95 | Creep | 1 |
| VTCS | VSXO | Random | No | VSXO | 3 | Creep | 1 |
| VTCS | VSM | Random | No | NPXO | 3 | VSM | URM, 0.15 |
| VTCS | H-Init | Hybrid | No | NPXO | 3 | Creep | 1 |
| VTCS | E/C | Hybrid | No | EBXO | 2, 0.95 | Creep | 1 |
| VTCS | E/C/C | Hybrid | Yes | EBXO | 2, 0.95 | Creep | 1 |
| VTCS | E/V | Hybrid | No | EBXO | 2, 0.95 | VSM | URM, 0.15 |
| VTCS | E/V/C | Hybrid | Yes | EBXO | 2, 0.95 | VSM | URM, 0.15 |
| VTCS | V/C | Hybrid | No | VSXO | 2, 0.95 | Creep | 1 |
| VTCS | V/C/C | Hybrid | Yes | VSXO | 2, 0.95 | Creep | 1 |
| VTCS | V/V | Hybrid | No | VSXO | 2, 0.95 | VSM | URM, 0.15 |
| VTCS | V/V/C | Hybrid | Yes | VSXO | 2, 0.95 | VSM | URM, 0.15 |

in Section 4.2.3.4. Therefore, it cannot be guaranteed that the results of the GA will always be equal or better than the shortest path solution, contrary to the situation for VTB.

Figure 5.20a shows that even the random initialization provides candidates of much better solution quality than the random initialization for the VTB encoding. It does not only inherently ensure connected topologies, avoiding unroutable demands, but the node betweenness-based order relation also happens to successfully create topologies as intended, where all demands can fulfill their QoS requirements. All three custom approaches, VSM, EBXO and VSXO, show a lower average cost value than the basic approach after only about 2000 evaluations. The recombination approaches seem to have a larger impact than VSM and VSXO is showing the steepest decrease in average cost value. As shown in the first magnified area, V/C and V/V along with their diversity-controlled variants show the fastest improvement among the hybrid initialized approaches. The second magnified area shows that the average value for E/V eventually catches up and is able to surpass the others. However, the final cost values in Figure 5.20b show a more ambiguous result. The Basic variant is once more the most consistent with an even lower median cost value than for VTB. EBXO and VSM, while initially showing promising values, result in worse final cost values than observed for Basic. Surprisingly, Basic results in slightly better cost values than H-Init alone.

Among the combination approaches, E/C/C is the most consistent with a solution quality slightly better than for Basic, but at a 29 % lower convergence time as shown in Figure 5.20c. The V/V and E/V versions obtained the best results at a cost value of 298, identical to the best value of VTB. Remarkably, E/V showed a 33 % lower median in convergence time than V/V, although it has a lower evaluation rate as shown in Figure 5.20d. The inclusion of diversity control mechanisms reduces the evaluation rate of the respective combination due to the computational overhead as expected, but it also has a surprisingly strong negative impact on the solution quality for V/V/C. A possible explanation for this is that V/V is too aggressive in focusing on the current best individuals, which would also explain its relatively large span of cost function




(a) Average cost over initial evaluations

(b) Final cost value

(c) Time until convergence

(d) Evaluation rate until convergence

Figure 5.20: Evaluation results for combinations of the VTCS encoding

values in Figure 5.20b, such that the presence of a diversity control mechanism slows the rate of convergence drastically, since it prevents overly rapid exploitation of current good solutions.

### *Compact Path Enumeration Encoding*

For the RCPE encoding we chose the same values for all combinations. For the basic path enumeration we use $k_{\mathrm{m}} = 5000$, $a = 18$ and $b = 6$. The choice for $k_{\mathrm{m}}$ is based on a previous analysis which showed that realizing the physical topology requires a minimum of 4630 paths. By exceeding this number, it can be guaranteed that the shortest path reference solution can be included in the population initialization. The values for $a$ and $b$ have been derived from experiments.

Table 5.13 lists the combinations of components used for this evaluation. The parameter for the Creep mutation is the stepsize, while SLCM and SLGM each receive three parameters, one for the small increment, one for the large increment and one for the ratio of small to large increments.

Table 5.13: Genetic algorithm hyperparameters for the RCPE encoding

| Type | | Population | | Recombination | | Mutation | |
|---|---|---|---|---|---|---|---|
| Enc. | Version | Init. | Control | Type | Param. | Type | Param. |
| RCPE | Basic | Random | No | NPXO | 3 | Creep | 1 |
| RCPE | EBXO | Random | No | EBXO | 2, 0.95 | Creep | 1 |
| RCPE | SLCM | Random | No | NPXO | 3 | SLCM | 1, 800, 10 |
| RCPE | SLGM | Random | No | NPXO | 3 | SLGM | 1, 800, 10 |
| RCPE | LHM | Random | No | NPXO | 3 | LHM | 2, 200, 0.8 |
| RCPE | RDRI | RDRI | No | NPXO | 3 | Creep | 1 |
| RCPE | H-Init | Hybrid | No | NPXO | 3 | Creep | 1 |
| RCPE | E/L | Hybrid | No | EBXO | 2, 0.95 | LHM | 2, 200, 0.8 |
| RCPE | E/L/C | Hybrid | Yes | EBXO | 2, 0.95 | LHM | 2, 200, 0.8 |

For LHM the parameters are $p$, $v_l$ and $v_h$. The values for NPXO and EBXO are identical to the ones used for the previous encodings. The hybrid initialization uses a mix of APTI and ASTI and includes the physical topology reference solution as before. Additionally, it includes a solution where only the shortest paths are used.

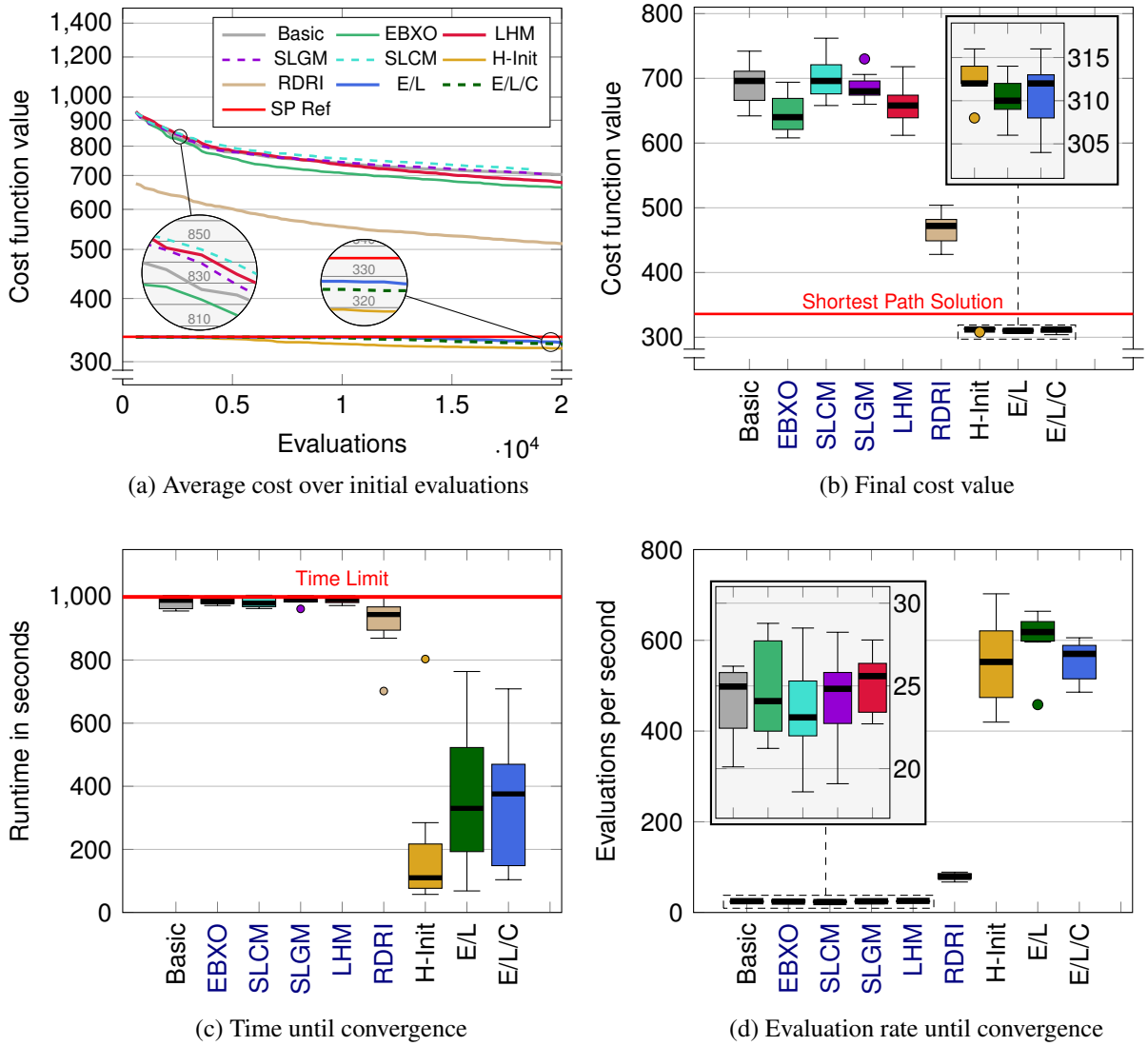
Figure 5.21a shows the comparison of these combinations during the first 20 000 of between 18 000 and 700 000 evaluations. The difference is even larger than for the virtual topology-based approaches, because all approaches that do not use the hybrid initialization evaluate excessively complex network configurations over the entirety of their runtime. Some of the SLGM and SLCM runs do not reach 20 000 evaluations, which is visible in the top right, where their graphs end prematurely.

The average values of the SLCM, SLGM and LHM mutation operators show little difference in the early stages and initially the basic version achieves better cost values as shown in the first magnification of Figure 5.21a. The second magnified area shows that this also holds true when comparing H-Init against combinations of these operators with the same hybrid initialization. The RDRI approach is successful in providing initial candidates, which show better cost values than the randomly initialized ones, but it is still insufficient compared to the hybrid initialization.

Figure 5.21b shows that the distribution-based mutation operators, SLGM and LHM, ultimately deliver better cost values than the basic version, but they are still far from the shortest path reference solution. EBXO is once more successful in accelerating the cost value improvement beyond the capabilities of the mutation operators but is also insufficient to reach relevant cost value ranges on its own.

Figure 5.21c shows that the last improvement in solution quality happens immediately before the time limit is reached, which indicates that the time limit is much too low for these approaches and the complexity of the solution candidates. This is especially evident in Figure 5.21d, where the inset shows that only about 25 evaluations per second could be performed. Only the combinations using the hybrid initialization, which are H-Init, E/L and E/L/C show evaluation rates comparable to the virtual topology approaches.

While they manage to obtain cost values below the shortest path reference, as shown in the inset of Figure 5.21b, the improvement in cost value is less than for the virtual topology-based approaches.




(a) Average cost over initial evaluations

(b) Final cost value

(c) Time until convergence

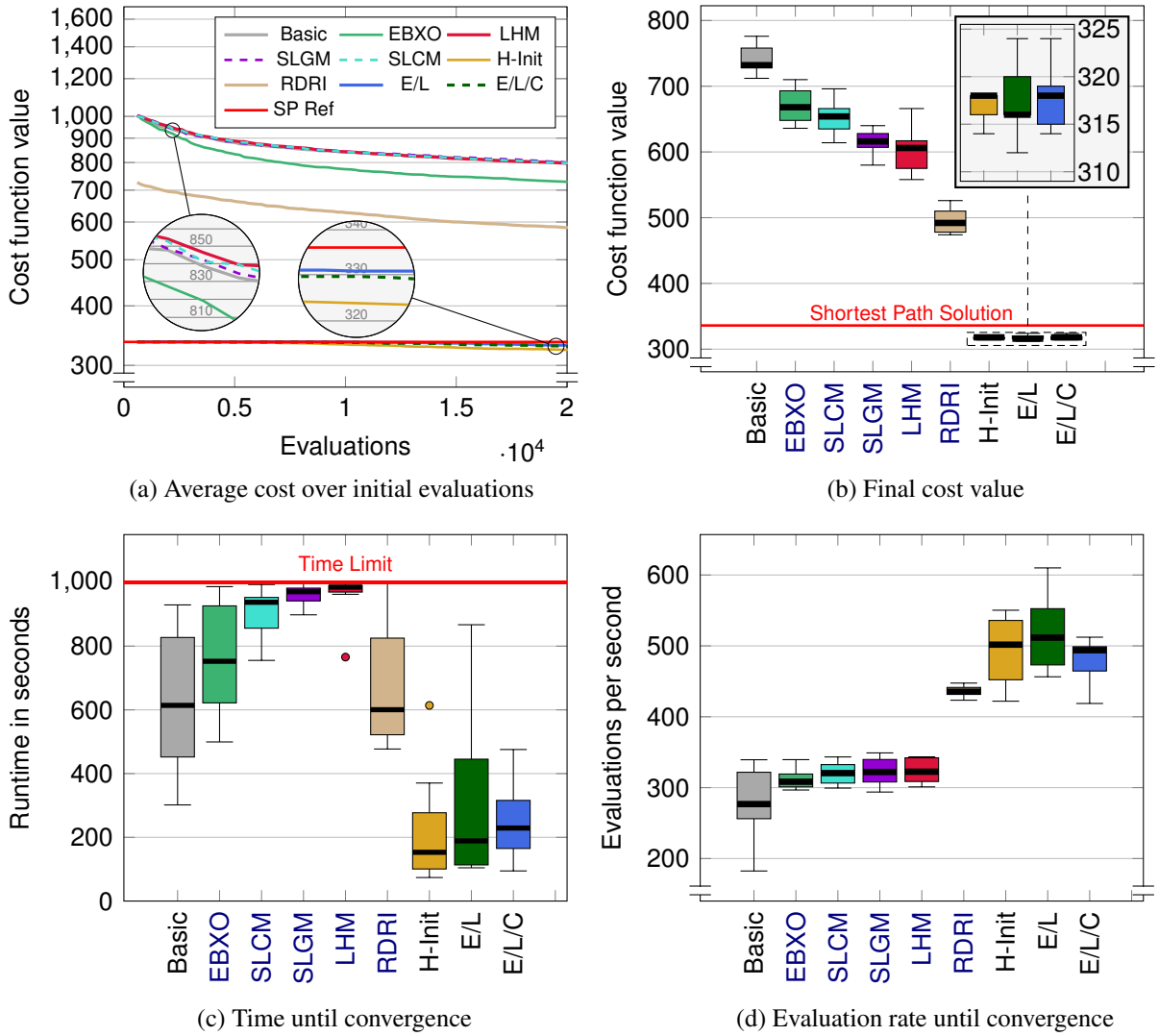
(d) Evaluation rate until convergence

Figure 5.21: Evaluation results for combinations of the RCPE encoding

### *Demand-Diverse Compact Path Enumeration Encoding*

For the DCPE encoding we use the exact same parameters for the path enumeration and for all of the individual operators and their combinations as for the RCPE encoding with the same rationale. Figure 5.22a shows that the randomly initialized values start at slightly higher costs than for RCPE, which is caused by the fact that RCPE's fallback routing mode implicitly improves grooming by rerouting demands from non-compliant routes onto existing links. The drawback of this mode is clearly visible in Figure 5.22d, where the randomly initialized versions of DCPE show a drastically higher evaluation performance as compared to the inset in Figure 5.21d, as DCPE only performs table lookups for all demands, where RCPE needs to run a QoS-aware routing algorithm for each non-compliant route.

Generally, the results and relative performance of the different combination versions is very similar to the results for RCPE. EBXO seems to be slightly more effective in the early stages and the mutation operators present larger differences in the late cost values, which may be attributable to the slightly increased locality and heritability of DCPE compared to RCPE.




(a) Average cost over initial evaluations

(b) Final cost value

(c) Time until convergence

(d) Evaluation rate until convergence

Figure 5.22: Evaluation results for combinations of the DCPE encoding

Somewhat surprisingly, E/L and E/L/C show no clear improvement over the more basic H-Init version, as shown in the inset in Figure 5.22b.

Figure 5.22c shows wide distributions of improvement increments over time, which can be attributed to the fact that the increased combinatorial complexity reduces the chances of improvement per mutation. For the randomly initialized versions, the more complex SLGM and LHM mutation operators seem to be more effective, as they deliver improvements up until the time limit and result in better final cost values than the others, despite a virtually identical evaluation rate as shown in Figure 5.22d.

### *Genetic Algorithm and Simulated Annealing Comparison*

The previous sections have shown that there are several combinations which can considerably improve the performance of the basic approaches. However, not all were able to compete with the shortest path reference solution in the given time frame of 1000 seconds. Figure 5.23 shows the final cost values for a selection of the most successful combinations of components for each encoding and compares them to the results obtained from the two Simulated Annealing

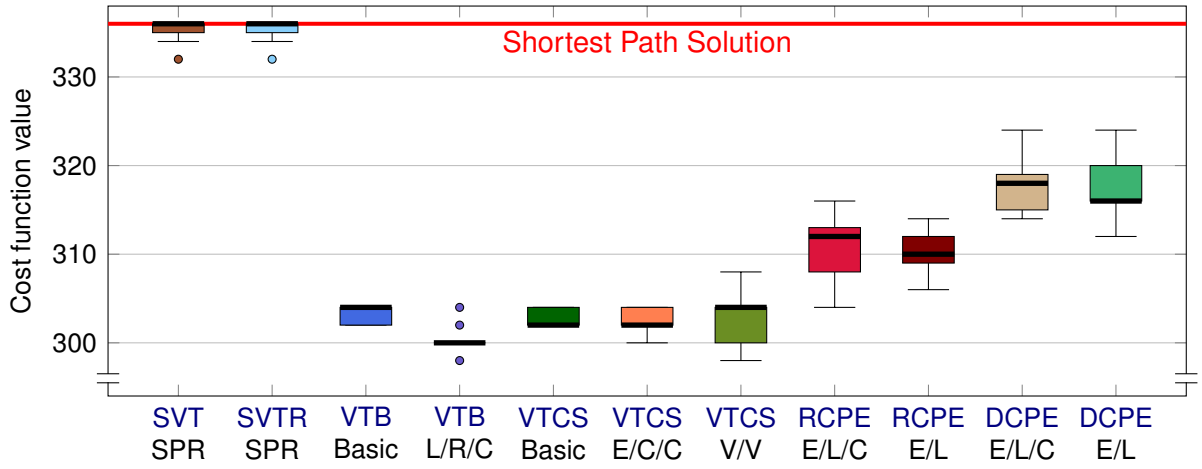



Figure 5.23: Final cost values for selected GA and Simulated Annealing methods

approaches, which have also been initialized with the physical topology solution to provide a level comparison. The best overall cost value found was 298, which was obtained by VTB in the L/R and L/R/C combinations, as well as by VTCS in the E/V and V/V combinations. The path enumeration-based encodings already suffer from the combination of the short time limit together with the complex problem structure, such that their cost values are much worse than for the virtual topology-based encodings, especially for DCPE, which has an even larger state space than RCPE. When only basic operators are used, VTCS shows slightly better results than the more simple VTB encoding.

When comparing them to the Simulated Annealing versions, most GA approaches result in substantially better cost function values. In fact, the median solution value of both Simulated Annealing versions is exactly the initial shortest path reference solution at a value of 336, such that for these cases no improvements were made. Surprisingly, both versions result in the same distribution of values, where the best result obtained was a value of 332, for exactly one run for each. The more complex SVTR method with its faster permutation scheme and passive routing optimization shows no advantage over the basic SVT.

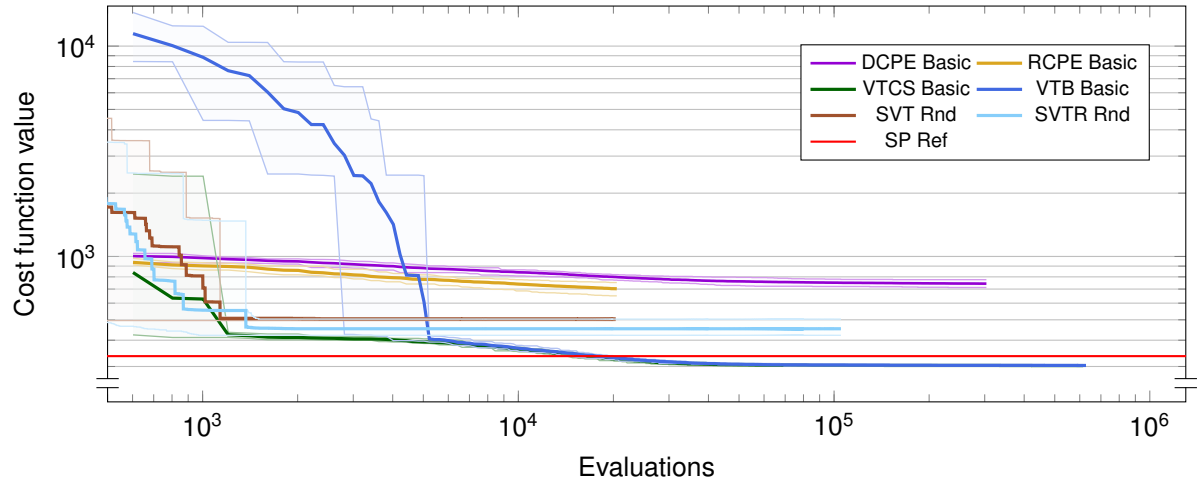



Figure 5.24: Cost function values over evaluations for basic metaheuristic approaches

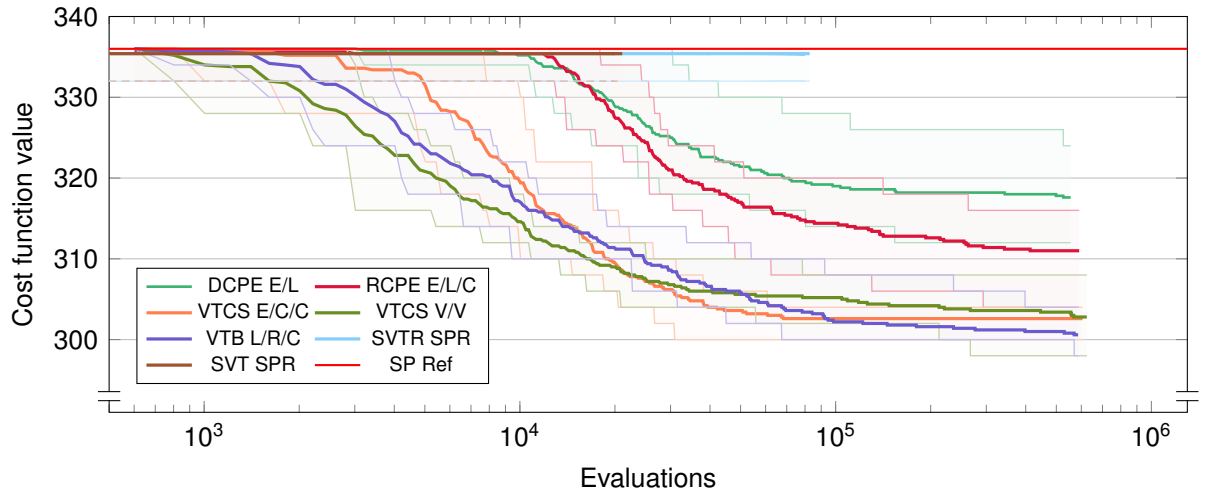



Figure 5.25: Cost function values over evaluations for advanced metaheuristic approaches

However, this raises the question if the GA approaches are only superior due to their parallel execution which allows a higher number of candidate evaluations within the time limit. To investigate this, Figure 5.24 shows the cost improvement over the number of evaluations for all basic methods, such that effects of the evaluation rate and time limit are largely irrelevant. Note, that the ordinate is not only log-scaled, but also transposed such that the lowest value shown is 400. The initial 500 evaluations are required by the GA to create and evolve its initial population, such that there are no earlier values available, while the Simulated Annealing has already improved upon its initial value. The figure shows average values with the shaded areas representing the minimum and maximum deviation of all recorded runs.

Within the first 1000 evaluations, the deviation of SVTR Rnd, SVT Rnd and VTCS are very large, but shrink significantly after that. However, this also marks the point from which the Simulated Annealing approaches do not improve in cost value anymore until termination. Interestingly, for the random initialization, SVTR does have a slight advantage over SVT. Despite the continuous improvement obtained by the RCPE and DCPE methods, their high complexity prevents them from competing with the Simulated Annealing methods. Basic VTCS and VTB both eventually outperform SVTR Rnd and SVT Rnd, but VTB needs almost five times as many iterations as VTCS to achieve this.

Figure 5.25 shows a similar comparison of the combined approaches which are all initialized with the shortest path reference solution. As has already been shown, SVT SPR and SVTR SPR only slightly improve upon the initial solution, but interestingly, this improvement happens within the first 20 evaluations. The routing-based GA approaches require a very large number of iterations, before they can significantly improve upon the initial shortest path solution. In the early stages, VTCS with V/V provides the best cost values, before eventually VTB using the L/R combination obtains better average results after about 100 000 evaluations.

### 5.4.3 Summary of Results

The evaluation has shown that the developed encodings, methods, and operators are effective in either improving the rate of convergence or the final cost value in most of the cases. The largest influence in all cases was observable for the hybrid initialization, which saved about 11 000 evaluations for the virtual topology-based approaches and allowed the path enumeration-based

ones to outperform the Simulated Annealing and shortest path reference approaches. When comparing the results to the *france* scenario, it is important to note that the runtime limit was shortened from 12 hours to the present 1000 seconds. At the same time the present problem's complexity is much higher, due to an increased number of links and therefore the routing approaches did not stand much of a chance.

Regardless, they were still able to achieve improvements upon the shortest path solution using select combinations of the developed components. However, none of them were able to compete with the virtual topology-based methods. VTCS was most successful in quickly improving upon the objective function early on, while VTB, which profited the most of the problem-specific adaptions, gave the best average and most consistent results in the L/R/C combination. Both encodings were able to achieve a reduction of 10 % in cost function value relative to the shortest path reference, which corresponds to an according reduction in hardware. The best solution found requires 298 TXPs and 81 line cards compared to 336 TXPs and 91 line cards for the reference solution, while fulfilling all QoS requirements. This means that the suggested approaches can achieve significant improvements over the legacy method even within a very short time frame and furthermore, outperform established metaheuristics such as Simulated Annealing.

## 5.5 Scalability towards Large Networks

The previous section has demonstrated that the developed framework and its GA-based components are able to solve a realistic, full-featured multi-layer problem in a constrained time frame. Although the example presented by the *CL–DC* topology is already of high complexity, the basic graph itself features a relatively small number of nodes, which is slightly below the average of the transport networks presented in Section 2.2.2 and certainly below what can be expected from future developments as outlined in Section 1.5.

While the path enumeration-based approaches had shown superior solution quality in the *france* scenario, they had already approached the limits of their scalability in in the *CL–DC* scenario. The virtual topology-based encodings however had mostly achieved their final results after about five minutes, suggesting that larger problems may still be solvable by these approaches. In order to test the limits of their scalability, we have developed a very large topology, based on the same data as the *CL–DC* topology, but with a drastically increased number of nodes and physical links.

### 5.5.1 Scenario Parameters

#### *5.5.1.1 Network Topology*

We designed a network topology, we refer to as *CL–Max*, as a scalability benchmark. With its 149 nodes and 206 links it is larger than any of the other layer-3 topologies given in Section 2.2.2. The locations, as well as the connecting fiber infrastructure are based on the same public data set as the *CL–DC*, such that the 19 data center locations are included in *CL–Max* as well. Additional information on how this topology was derived from the data set, can be found in Section B.3 of the appendix. The topology is shown in Figure 5.26 and relevant parameters are given in Table 5.14, where fractional values are rounded.

In the table are also three intermediate versions of this network, which we have obtained by limiting our TXP model as follows. The small version contains all virtual links, which can be created using only the 200 and 250 Gbit/s modes, while the medium version additionally uses

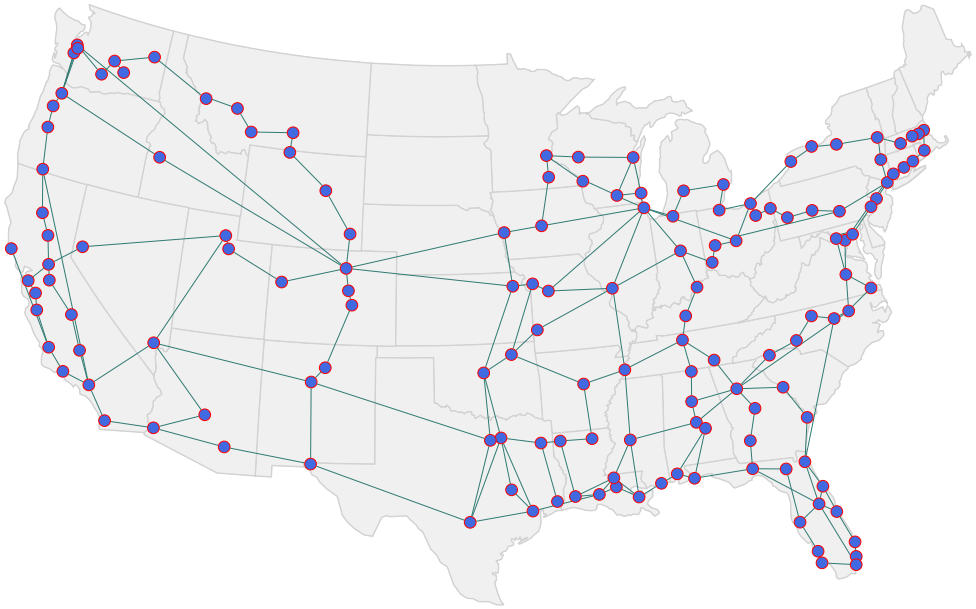


Figure 5.26: Physical topology of the *CL–Max* network

Table 5.14: Parameters of the *CL–Max* topology

| Topology | Nodes | Links | Density | Degree | | Closeness | | Betweenness | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Max. | Avg. | Max. | Avg. | Max. | Avg. |
| Physical | 149 | 206 | 0.0187 | 8 | 2.77 | 0.182 | 0.129 | 7061.9 | 1037.7 |
| Small | | 1755 | 0.159 | 42 | 23.6 | 0.427 | 0.316 | 5110 | 319 |
| Medium | | 4725 | 0.429 | 102 | 63.4 | 0.722 | 0.559 | 2995 | 110 |
| Large | | 9345 | 0.848 | 148 | 125 | 1 | 0.879 | 313 | 22.6 |
| Full | | 11026 | 1 | 148 | 148 | 1 | 1 | 0 | 0 |

the 150 Gbit/s mode and the large topology also includes the 100 Gbit/s mode. The different versions provide a more gradual approach towards the full topology and furthermore highlight the increase in topological density.

With a dauntingly large number of 11 026 bidirectional links in the full virtual topology, the computational complexity of the state space suggests that it is most likely not only intractable for most optimization approaches, but also that finding an acceptable local optimum may prove difficult even for metaheuristic approaches.

#### *5.5.1.2 Physical Parameters*

The physical parameters regarding latency per distance, availability per distance and hardware are identical to the ones given in Section 5.4.1.2. Once more, the proposed flexrate device is able to realize a circuit on the longest of all shortest paths, permitting a full graph as the virtual topology. Table 5.15 shows the same data rate and reach values, but the connection ratios are updated for *CL–Max*. Due to the reduced length of the physical links, caused by lower average node distances compared to *CL–DC*, the overwhelming majority can now utilize the fastest transmission mode. This also means that virtual links between nodes that are far apart from

each other now need to traverse more fiber links with small detours. This increase in virtual link length therefore also increases the share for the slowest mode.

Table 5.15: Flexrate modes and feasible circuits for *CL–Max*

| Data Rate in Gbit/s | 50 | 100 | 150 | 200 | 250 |
|---|---|---|---|---|---|
| Transparent Reach in km | 8000 | 4000 | 2000 | 1000 | 500 |
| Active for % of Virt. Links | 15.3 | 41.9 | 26.9 | 10.0 | 5.9 |
| Active for % of Phy. Links | – | – | 1.5 | 5.8 | 92.7 |

#### *5.5.1.3 Traffic Demands*

We have generated a traffic demand matrix following the same procedure as outlined in Section 5.4.1.3, based on the same Cisco, DISCUS and population data. We also use the same three QoS-classes, i. e., a latency class with a constraint of 10 ms, an availability class with a constraint of 0.9 and a best-effort class. For the present network, the generation resulted in 47 094 unidirectional traffic demands, where for each demand there exists a demand in reverse direction of non-symmetrical data rate. Splitting of demands to ease the routing process is not allowed. The sum of all traffic demands amounts to 44.8 Tbit/s, where best-effort traffic again occupies the largest share with 57.9 %, followed by the latency class traffic with 22.9 % and 19.2 % remaining for the availability class. More details can be found in Section B.4 of the appendix.

#### *5.5.1.4 Optimization Goal*

Since the physical parameters and traffic demands follow the same parameters as for the *CL–DC* network, we use the same objective function for *CL–Max*, i. e., $F_{QoS}$ defined in Equation (5.2). Due to the drastic increase in size of most aspects, the values can be expected to be much higher than for the previous network. While the sums of the traffic demand capacities are comparable, there are also more nodes that require connectivity.

#### *5.5.1.5 Algorithmic Parameters*

For each of the virtual topology encodings, we use one of the combinations of components which has turned out to be the most successful in the analysis based on *CL–DC*. We use a time limit of 3 hours given the drastic increase in complexity and aggressively tune the parameters towards quick convergence. As reference method, we use the shortest path solution based on the physical topology and for Simulated Annealing, we will also use this as the initial solution. Furthermore, we will use the same cooling schedule, but we only use the faster SVTR version. For the GA-based methods, we reduce the population size to a mere 10 and the offspring per generation to 20 individuals. Consequently, we reduce the number of contenders for the tournament-based parent and survivor selection to 3.

For VTB we do not make use of the hybrid initialization mixing APTI and ASTI, but instead we rely solely on APTI, as it is more computationally efficient. We will also deactivate the repair function to further focus the available runtime on developing new individuals as quickly as possible. For VTCS there is inherently no performance impact from a spanning tree initialization and the cost of using APTI, such that the hybrid initialization is used for this approach.

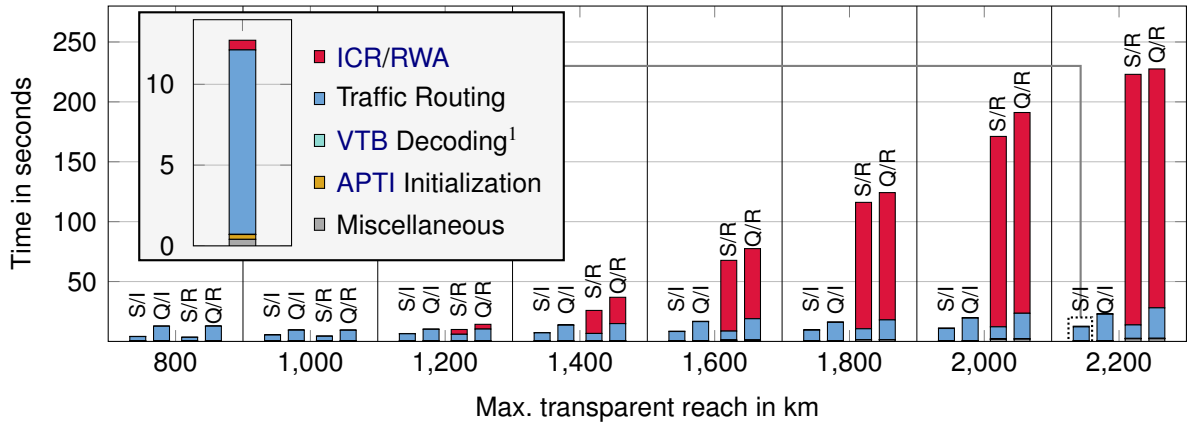



Figure 5.27: Average generation and evaluation time for different transparent reaches

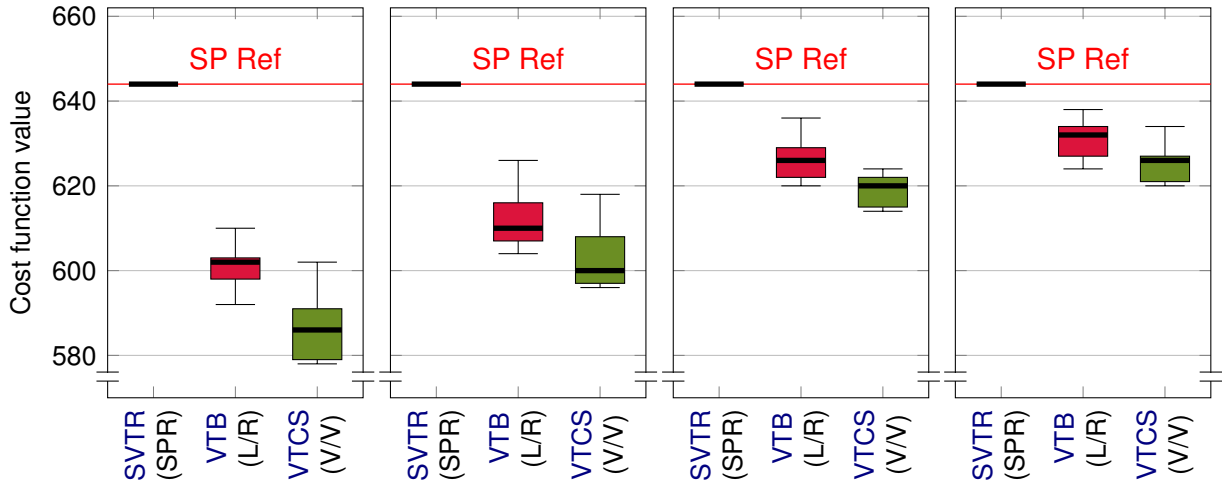
Furthermore, we also change aspects of ancillary algorithmic components in the framework for both metaheuristics providing a problem relaxation in order to speed up the computation, which will be explained in the following section.

### 5.5.2 Evaluation

In order to allow for any improvement over the reference solution, any iterative optimization approach needs to explore as many solution candidates as possible. There are three important factors limiting scalability. The exponential increase in search space size for increasing problem sizes, as explored in Section 4.5.2.1, means that the number of evaluations required for a meaningful improvement increases accordingly. The second factor is the performance of the genetic encodings and operators, as evaluated in the last section. The final factor is the performance of the overall framework.

To investigate the limits of the framework, we artificially scaled the maximum transparent reach between 800 km and 2200 km in increments of 200, which explores the topology expansion between the small and medium virtual topologies. We randomly generated individuals based on the VTB encoding using the APTI initialization and we measured the time to create and evaluate an individual using different ancillary algorithms, repeating the process for each combination 100 times. Figure 5.27 shows the average times obtained. For traffic routing, "S" denotes a standard shortest path algorithm, while "Q" is a QoS-aware algorithm. "I" represents ICR without WA and "R" is a full RWA, such that we obtain the four combinations "S/I", "S/R", "Q/I" and "Q/R".

The inset shows that the traffic routing is mostly the dominant contributor to the overall time due to the quadratic worst case scaling behavior of routing algorithms. QoS-aware routing requires almost twice the time of the shortest path routing, because it iteratively needs to consider more alternative path candidates whenever the initial ones fail to meet the QoS requirements. However, beyond 1400 km RWA becomes the dominant share and the algorithm's performance degrades so quickly, that it becomes prohibitive to solving larger problem instances. Since we are interested in determining, when the combinatorial complexity becomes too large for the topology encodings, we have decided to use ICR without WA in the remainder of this evaluation. This effectively means disregarding the wavelength continuity constraint, which reduces the realism of the scenario. However, we are confident that a more efficient RWA implementation could easily replace the present component and prevent this algorithm from becoming the performance




(a) Small topology (b) Medium topology (c) Large topology (d) Full Topology

Figure 5.28: Final cost values for selected GA and Simulated Annealing methods

bottleneck early on. Note, that the actual VTB decoding procedure is so efficient that its runtime is dwarfed[1] by all other time shares.

Moving on to the actual configuration optimization, our shortest path reference approach resulted in a cost function value of 644, which is represented as the red line "SP Ref" in Figure 5.28. Since the physical links are included in all other scenarios, we will retain this as a reference for all of the four network problems from small[2] to full. At this point, the Simulated Annealing-based SVTR is unable to provide any improvement beyond its initial value of 644.

Both, the VTB and VTCS encodings were able to achieve significant improvements, with VTCS consistently showing better median cost value than the simple VTB encoding, which is in line with the observation from the previous section, that VTCS is more efficient during the initial stages of the runtime. For the small topology, VTCS achieved an average improvement of 8.7 %, whereas VTB could only achieve 6.55 % improvement. This corresponds to an average reduction of at least 40 TXPs and 12 line cards. However, as the network density increases from the small to the full topology, the reduction decreases down to an average 2.89 % for VTCS and 1.96 % for VTB, finally approaching the limits of scalability due to the increased search space.

### 5.5.3 Summary of Results

Increasing the number of nodes while considering the capabilities of flexrate devices, rapidly increases the combinatorial complexity of multi-layer networking problems beyond the point, where significant cost savings can be achieved within small time frames. Our relatively simple RWA implementation became the performance bottleneck when increasing the number of links to upwards of 3000, such that we decided to discard the wavelength continuity constraint in order to focus on effects on our GA approaches for the largest of topologies. Under these circumstances we found that for the small topology with 1755 links, the proposed GA-based algorithms were able to achieve a reduction in cost value between 5.28 and 10.2 %. These values diminished

[1]The VTB decoding procedure requires so little time, that it is not even visible in the plot in Figure 5.27.

[2]While 1.5 % of the physical links are normally infeasible in the small virtual topology, we added an exception applying to the purely physical links to prevent a split into unconnected eastern and western networks, because the few long links in *CL–Max* happen to be the ones exclusively connecting the east to the west.

to between 0.93 and 3.73 % for the full topology with 11 026 links. Given sufficiently potent ancillary algorithms, our approach is therefore efficient even for very large networks, given a runtime of several hours.

## 5.6 Discussion

Summarizing the results of the previous sections, it can be stated that the developed approaches definitely show potential for application in multi-layer problems as they show considerable improvements even compared to other metaheuristics. However, the presented scenarios are limited in scope and a generalized statement regarding arbitrary problems is difficult at best.

### 5.6.1 Scenario Aspects

While the evaluation has addressed combinatorial complexity as the primary influence in network optimization, there are numerous other aspects which can change the nature of the problem and impact solubility. The traffic distribution and magnitude exert a certain influence on the routing algorithms, since larger demands can require routing more circuits and thereby add to the overall complexity. The effect is lessened in our approach by including path caching, but once the limits of fiber capacity are approached, the performance will degrade substantially. However, it can be argued that optimizing such a network without any spare fiber capacity is not a realistic use case.

The composition and number of QoS classes also has an influence, which is especially large for the DCPE encoding, since it immediately increases the input space. A very high number of classes may therefore be prohibitive to the use of DCPE. The other approaches are less sensitive to the number of classes, but the underlying latency and availability models may increase the runtime of auxiliary algorithms such as QoS-aware routing approaches. If, e. g., node delay values are within similar numeric ranges as transmissions delays and even vary non-uniformly in the same network, path searches increase in complexity, because delay accumulation may not scale proportional to the distance and number of hops. In extreme cases, this may even lead to situations, where the VTCS encoding may focus on problematic configurations, because its suggested order relation is oblivious to the varying QoS figures and may misguide link selection.

More complex hardware models including choices between heterogeneous devices may not be efficiently integratable into the present approach. E. g., including TXPs of differing capacity, cost and rack space requirements can become problematic when using many low-rate devices is much cheaper than a single piece of equipment, since a local auxiliary algorithm may not be able to deterministically find a hardware composition, that is optimal for the network, since many low-rate devices occupy far more spectrum. Since these local choices can have a large impact, they would have to be included in the encoding such that this impact can be actively controlled by the optimization.

### 5.6.2 Algorithmic Parameter Dependence

A general problem for many metaheuristic frameworks is their large number of parameters and the difficulty of finding meaningful sets of values for them. For any GA the population size, number of offspring per generation and termination condition need to be determined relative to the problem. Long and complex chromosomes generally require larger populations to ensure sufficient diversity to explore the search space. However, it is generally impossible to determine

what the best set of parameter values is for a given problem, other than through experimental exploration.

We chose large populations and numbers of offspring on the order of thousands or higher whenever runtime was not an issue and solution quality was the objective. For short runtimes we had to limit both, such that a reasonable number of generations could emerge within the given time limit. Based on tests, varying the parameters between those for the *france*, *CL–DC* and *CL–Max* problems, the solution quality was gradually affected, such that even for less-than-ideal parameter sets, an improvement beyond the reference solution was possible in most cases.

For mutation operators, small step sizes generally resulted in better results than large step sizes, most likely due to the limited locality of the encodings in the multi-layer context, which warrants a fine-granular exploration. Drastically increasing step sizes, especially beyond half of a gene's domain, has resulted in degraded performance in all previous tests. For recombination, however, the choice is more difficult. While preserving larger subsequences by using few crossover points has generally led to better results, this can also be taken to the extreme by choosing almost all genes from a single parent and applying them to both offspring. Being too aggressive at this point results in a very high probability for premature convergence.

Overall, we expect the chosen parameters to work reasonably well for similar problems, but it is possible that combinations of scenarios and genetic operators with specific parameters exist, that can result in different relative operator performance, than what has been presented in this evaluation.

### 5.6.3 Limits of Applicability

The performance of the presented algorithms and approaches largely hinges on the combinatorial complexity of the problem. Even relatively small networks can exhibit prohibitive levels of complexity in the presence of flexrate devices and complex QoS models. While the evaluation has shown that networks of over 10 000 usable links still have some potential to be optimized, the time frame needed to obtain a reasonably good result has also significantly increased. Such detailed large-scale scenarios are therefore only addressable in long-term planning tasks, where at least some level of parameter tuning and ample runtime are available. When solution quality is required and a sufficient number of paths can be generated, the RCPE or DCPE encodings can be expected to deliver the best results.

Given a sufficiently small problem, either through a limited level of detail or a limited network complexity, a configuration can be optimized in relatively short time frames of about 15 minutes, which is within the realm of network operation tasks, which also makes this approach viable for network reconfiguration. However, a real-world network might require a much finer level of detail or must be augmented by a more problem-specific framework, which deals with its specific idiosyncrasies. Path enumeration-based encodings are not readily applicable to typical problem scenarios.

### 5.6.4 Shortfalls and Extensions

For the present framework implementation, the factors limiting scalability are the QoS-aware routing algorithm and the RWA algorithm. Both are based on not overly efficient library implementations of a simple k-shortest path algorithm. Replacing these by more efficient algorithms, augmented by caching and lookup tables is expected to significantly improve the scalability of the overall framework, although not by several orders of magnitude.

Scaling the approach to even larger problems, the framework could also be extended from a multi-threaded parallelization approach to a multi-computer approach, such that it can be run on a large cluster. There are several approaches, how different populations can be maintained and managed in parallel to further enhance the performance [69, pp. 95–97].

### 5.6.5 Applicability to other Problems

The present evaluation and framework design are geared towards (re)configuration of multi-layer networks using a packet-routed over a circuit-switched technology, but most of the basic aspects of the encodings and operators are in fact applicable to other problems in networking as well. E. g., 5G Cloud-RAN systems [35] or novel reconfigurable microwave access networks [179] are also a good fit for the presented framework, since they regularly need to adapt their topologies and routing to the changing conditions and demands. Similarly, reconfigurable data centers [83] need to manage their resources and interconnects, which could also present a use case for our approaches.

The overlay graph structure present in our approaches can also be entirely removed from computer networking and applied to other areas, where an interconnection topology and traffic flows are important. This may include tasks as diverse as dynamic road traffic routing, planning extensions to railway networks, or optimizing logistics.

## Chapter Summary

The objective of this chapter is the evaluation of the integrated multi-layer network configuration approaches along with their GA components. It contains explanations regarding the methodology of the evaluation and the design goals for the scenarios employed in the performance analysis. Furthermore, the reference methods, which represent a basic legacy approach and Simulated Annealing as a competitive metaheuristic, are motivated and their relevance for the present endeavor is explained.

We use four evaluation scenarios, each serving a different purpose. The first is a basic function validation on a small-scale model topology. It shows the ability of the developed approaches to exploit simple concepts such as finding shortest paths, performing grooming and providing QoS-compliant routing. It also highlights important design-inherent tradeoffs.

The second scenario is a basic reference network design problem from the SNDlib database, for which a lower bound to the optimal solution is known. Therefore, we can determine whether the proposed encodings can represent the solution space reasonably well, such that their solutions can compete with those of established approaches. Due to the focus on quality rather than speed, the algorithms were provided with a 12-hour runtime. All metaheuristics consistently outperform the legacy approach. The GA-based methods are either on par with the Simulated Annealing methods or yield even better solutions.

The third scenario presents a realistic network with QoS-enabled traffic, where the usage of flexrate devices increases the combinatorial complexity significantly. This evaluation provides details on the performance of different GA components and enhanced versions. Results show that our adaptions are able to considerably improve solution quality, especially when short runtimes on the order of 15 minutes are mandatory. The enhanced versions not only improve upon the legacy approach's solution by 5 % to 11 %, but they also outperform the reference metaheuristic, which only achieves a 1.2 % improvement at best.

The final scenario demonstrates scalability and shows where tradeoffs emerge for problems of increasing complexity. We scale a network topology of 149 nodes to between 1755 and 11 026 potential links, which is more complex than any of the investigated reference networks, and combine it with over 20 000 simultaneous traffic demands. While this is by all accounts a largely intractable problem, our topology-based GA approaches were still able to achieve improvements compared to the legacy approaches in a 3 hour time frame when relaxing the wavelength continuity constraint.

Finally, we presented a discussion about the limits of the evaluation and the algorithms' scalability and applicability to other problems.

# 6 Conclusion and Outlook

The topic of this monograph was motivated by observing four current trends in networking and their implications for network operation and planning as outlined in Chapter 1. First, emerging novel applications and services around 5G and the tactile internet will more often require QoS guarantees. Second, businesses increasingly rely on cloud-based services. Given the first two observations, SLA-based services will be of growing importance to NSPs and ISPs. Third, new degrees of flexibility in multi-layered transport networks become available through novel hardware and control planes. Fourth, NSPs and ISPs follow the global trend of virtualization, considering more flexible and short-term deployments of services and capacity. The implication of the latter two observations is that complexity in network operation and planning tasks increases, while lead times decrease.

We have explored the details of multi-layer networks, including legacy and new technologies, as well as common networking problems in Chapter 2. It was established, that creating a configuration for such a network, i. e., determining the required NEs and their configurations and interconnections, is a complex problem consisting of solving VTD, TDR, RWA and their dependencies. Further investigations showed, that known transport networks have a relatively low density, which might substantially increase in the near future, especially due to the adoption of flexrate TXPs, which can lead to a significant increase in complexity for each of these subproblems. Dealing with this increase therefore requires scalable algorithms to solve the multi-layer configuration problem considering QoS and flexrate devices.

In Chapter 3 we have analyzed the theoretical properties of topology design and network routing optimization problems and explored possible solution methods and their applicability to these problems. We have reasoned that exact algorithms such as mathematical optimization are not sufficiently scalable, such that metaheuristic approaches are the most promising candidates. The principles and components of GAs, as well as Simulated Annealing were introduced, since the latter is applicable as a reference method. The chapter concludes by providing an overview on related research regarding network optimization approaches, in which we find that there are several works dealing with multi-layer networks using metaheuristics, but none of them addresses the present combination of factors including QoS and flexrate devices.

We chose GAs as our approach, since they provide a flexible and inherently parallel framework. Based on the prior problem analysis from Chapter 2, we determined that the complexity of a holistic optimization including all parameters from VTD, TDR and RWA does not scale to larger networks and therefore we chose to apply optimization only to a single subproblem and derive the remaining configuration deterministically from the results of the optimization as a more viable alternative. Research by Idzikowski *et al.* [129] suggests, that optimizing the upper layer yields a larger impact on resource requirements, such that we focus on VTD and TDR.

We have developed two genetic encodings for each of these subproblems, which are described in detail within Chapter 4. VTB uses a simple and efficient binary encoding of virtual links, but it is also capable of representing undesirable solutions, such as unconnected topologies.

The more advanced VTCS avoids this by creating topology solutions around spanning trees of links connecting to nodes of high betweenness centrality within the graph formed by the physical layer. The TDR approaches both use a path enumeration encoding. RCPE encodes paths between all node pairs, such that each gene can take an index value corresponding to a specific path. After establishing these paths, the approach checks all demands for QoS compliance and selectively reroutes demands of violated constraints onto compliant paths using a shortest path algorithm on the topology implied by the established paths. DCPE uses path genes for each demand, thereby gaining the advantage of grooming all traffic, while incurring the drawback of an increased search space.

Furthermore, we have developed several techniques and genetic operators to enhance the performance of the developed encodings. We have implemented a system of advance checks and repair functions to eliminate undesirable solutions for VTB, designed encoding-specific mutation and crossover operators for VTCS, as well as problem-specific initialization and mutation techniques for RCPE and DCPE. We have integrated these encodings and operators into a larger multi-threaded solution framework, combining it with ancillary heuristics to derive two integrated solution approaches for multi-layer network configuration.

We have evaluated the proposed genetic approaches and the surrounding framework regarding its properties in four different stages. An initial basic function test used small-scale problems, each engineered to demonstrate a different effect, where all approaches behaved exactly as expected. The second stage was built around a known reference problem from the SNDlib database, for which a dual bound is known. A regular shortest path algorithm was used to determine an upper bound, such that it can be tested if the encodings are able achieve meaningful results between these bounds. Furthermore, results from the Simulated Annealing approaches were also given as a reference. During the 12-hour runtime all encodings went below the upper bound and the topology-based approaches were on par with the more advanced Simulated Annealing method, while the routing-based approaches obtained even better results.

In the third stage we have tested the performance using a strict time limit of only 1000 seconds or roughly 15 minutes for a smaller, but more complex scenario including QoS traffic and flexrate devices. We had consciously chosen the time limit to be very low in order to investigate if the proposed acceleration techniques and operators could lead to a meaningful improvement even for shorter network operation time scales. The increased problem complexity allowed the simpler virtual topology approaches to outperform the RCPE and DCPE. In fact, the routing approaches were unable to compete with the shortest path reference, unless our APTI initialization scheme was used. Similarly, the Simulated Annealing approaches could only beat the reference solution, when they used it as a starting point and even then, only little improvement was possible, while VTB and VTCS were able to reduce the number of required TXPs by up to 10 %. For all algorithms, a good starting solution/population was the most important influence. We also showed that the superior performance of the GA methods is not solely the result of parallelism, by showing the improvement over the number of evaluations for each algorithm.

The fourth and final stage used a large, realistic network including all constraints and featuring more nodes and links than any of the transport networks investigated in Chapter 2. The objective was to highlight the limits of scalability of our present implementation. We found that our framework is primarily limited due to the ancillary algorithms, such that we applied a number of relaxations to them, in order to see how long the GA approach could improve upon the reference solution. While the problems quickly became intractable to the routing approaches, the VTB and VTCS encodings augmented by our customized operators were able to achieve improvements up to the maximum network size.

Finding good parameters is still an issue, but our investigations have shown that the range of reasonably good parameters is relatively wide and that the whole framework is surprisingly robust against minor deviations. However, all configuration should be made with the problem complexity, goal and time frame in mind. Short time frames only allow for limited improvement and necessitate a good initialization at the risk of missing global optima by a wide margin. Using the topology-based approaches is also helpful in these situations, since their lower combinatorial complexity can lead to larger improvements more quickly. If solution quality is paramount and time irrelevant, routing approaches and large populations, number of offspring and time frames should be considered.

However, while the evaluation within a monograph can only cover so much, we still feel that our scenarios and choices are diverse, yet sufficiently representative for common cases to state that we have successfully demonstrated functionality, quality and scalability of our proposed encodings, operator approaches and their surrounding framework. They were able to compete with an established metaheuristic like Simulated Annealing and even surpass its results considerably in complex and realistic multi-layer network scenarios.

Regardless, there are still many areas in which the existing methods could be improved even further. The most limiting aspect are certainly the traffic routing and RWA algorithms we have used. More memory-efficient path storage, more intelligent caching and generally faster algorithms, would certainly improve scalability and solution quality through an elevated evaluation rate. Especially highly scalable routing algorithms, which allow to offload calculations to a preprocessing step are an interesting aspect, since this basic path data can be made available to genetic encodings for optimization as well.

Another interesting research question is concerned with the limits of parallel execution. Our implementation is not designed to run distributed on a cluster system, but it can easily be extended in this way. This is especially interesting for a branch of genetic algorithms, that run multiple populations in parallel and exchange individuals after a certain number of generations, which can be more efficient than managing a single large population. Another interesting aspect to consider are self-adaptive operators, which can vary their parameters according to the current results. While our EBXO and VSXO approaches scale the number of genes to be exchanged according to cost difference, this is a rather limited approach. More advanced operators might consider the remaining time, present population diversity or time since last improvement to scale their aggressiveness.

Finally, while our integrated approaches are designed with multi-layer networks in mind, the encodings and operators are in essence applicable to any scenario that can be expressed as a graph problem, where either connections or paths are subject to optimization. This can include classical networking problems from local networks to radio networks, but it also applies to transportation, logistics, hardware, and mechanical design, among many other potential fields, to which our GA-based approaches can be applied.

# A Further Details on Multi-Layer Networks

## A.1 Aspects of Layer Separation in Multi-Layer Networks

The multi-layer structure of transport networks has several reasons and implications regarding functional, technological, and administrative aspects. From a functional perspective, a layered architecture allows for the combination of the advantages of different communication paradigms. E. g.,connection-oriented and connection-less approaches or routed and switched forwarding domains can be employed simultaneously as needed on the same infrastructure. This also includes layers as a means of functional adaption. E. g., allowing many low-rate client traffic flows to be multiplexed into larger flows thereby simplifying the number of forwarding decisions. Another example are legacy services, which can be emulated as an overlay on top of the actual network as a means of providing a migration path to new technology.

When the technologies are vastly different, a layered architecture is typically less complex in terms of individual NEs. Specialized NEs on each layer can focus on their own technological requirements, rather than implement all possible peculiarities. E. g., when packet and optical technologies are integrated into the same NE, then this device needs to simultaneously manage aspects of optical transmission like wavelengths and amplifier settings (cf. Section 2.1.2.2) as well as aspects of packet transmission like buffering strategies and packet classification.

Each of these technologies require their own set of specific hardware components and software functions for forwarding, control and management, such that NSPs employ different experts to work on their respective layers, often leading to an administrative split as well. While this separate mode of operation leads to a reduced complexity within the layers, it also has drawbacks for the network as a whole. Making changes that affect the whole system is cumbersome as it requires sequential changes layer by layer where the individual heterogeneous systems are unaware of implications beyond their interconnection points. This is especially relevant when considering service provisioning times, as the human factor can often significantly delay a timely deployment.

Alternatively, the layers can share a common or hierarchically overarching multi-layer control and/or management plane [148] which collects information of the NEs on all layers of the network and can therefore establish a global network state. With such an approach, changes can be effectuated much more rapidly and safely since cross-layer effects are inherently considered. Furthermore, a combination of this comprehensive perspective and an SDN-based control plane enable holistic and optimized operation and planning where services can be controlled on a fine-grained level and margins can be tailored to a precise fit [135].

## A.2 Multi-Layer Architecture Examples

Several multi-layer architectures for transport networks have been proposed and are in use with NSPs and ISPs. As stated in Section 2.1.1.1, most client traffic is packet-based, such that the top-most layer is implemented using a packet-based technology, while the point-to-point transport is handled by an optical layer providing high data rates and long-range connectivity.

The optical layer is typically based on WDM-technologies [90] which have largely replaced SONET and Synchronous Digital Hierarchy (SDH) [95] in core networks [63, ch. 13.1.4], due to higher capacities per fiber and better adaption to the now dominating packet- and especially IP-based client services [201]. While Course WDM (CWDM) is a less costly technology compared to DWDM, it offers less capacity per physical link [92, 93] and does not provide long-reach interfaces for datarates beyond 10 Gbps [30, 94], making it unsuitable for geographically large transport networks. Due to these considerations, we limit the scope to architectures which are designed with a focus on packet traffic and DWDM as the optical transmission technology.

IPoDWDM is an architecture where IP traffic routing is directly coupled to DWDM-ports on routers. This allows for a reduced number of hardware devices, since no additional and potentially expensive adaption technologies and layers are required. On the other hand, the complexity and cost of the router components is increased, since a tight integration of both technologies is required in a single type of NE. While this allows for a fine-grained handling of packet traffic, it also means that control, forwarding decisions and Traffic Engineering (TE) tasks become more complex as well. This becomes apparent for leased lines, ATM and especially TDM client services, which cannot be directly mapped to DWDM and therefore need emulation over the IP-layer adding transmission overhead to these services. Such an emulation of a point-to-point abstraction in a packet-switched network is called *pseudowire* and can, e. g., be implemented using the Pseudo Wire Emulation Edge-to-Edge (PWE3) architecture [212]. While IPoDWDM networks can make use of traditional control and management plane protocols such as Generalized Multi-Protocol Label Switching (GMPLS), some known real-world deployments use proprietary SDN-controllers to manage NEs on both layers [149].

IP/MPLS [208] over DWDM is a widely deployed architecture that simplifies traffic forwarding and improves scalability at the expense of adding a new technology. Regular IP nodes make routing decisions at every hop along the path for every flow, potentially involving repeated path calculations. In an IP/MPLS network, path computation is done only once by the node, that receives IP traffic from a client outside of the transport network. The paths, which are identified by labels and hence called Label-Switched Paths (LSPs), are communicated to all following nodes on the LSPs and all IP packets to use the respective LSPs, will be prefixed by the according labels. This also allows to map multiple IP traffic flows to the same LSP, effectively aggregating them. Since MPLS is designed to forward layer-3 protocols such as IP just as well as layer-2 protocols, client services can be transported in a uniform way. With the MPLS Traffic Engineering (MPLS-TE) extension LSPs and pseudowires can be explicitly set up by a Network Management System (NMS) via Resource Reservation Protocol - TE (RSVP-TE) [209] which allows even legacy TDM services to be used.

MPLS-TP over DWDM is the most recent of the presented architectures. MPLS-TP [99, 217] is a simplified version of MPLS, augmented by functions to improve operations, administration, and maintenance (OAM or OA&M), as well as resiliency. While the IP/MPLS architecture was designed to be driven by existing IP control plane protocols and provides a connection-oriented abstraction for uni-directional flows, the present architecture is designed closer to the optical layer, such that LSPs can be set up as bidirectional connections. It uses the GMPLS and PWE3

control planes [220] and is intended to be managed by an external NMS, which may centrally perform all routing and distribute the labels. The centralized approach makes this architecture very suitable for use in an SDN-environment.

IP/MPLS over Optical Transport Network (OTN) over DWDM adds an additional layer to the network. OTN in this architecture refers to the electrical switching part of ITU-T recommendation G.709 [96], which specifies optical and electrical switching and multiplexing capabilities. It establishes a hierarchy of optical containers which can be embedded into each other in a TDM-fashion similar to SDH. The electrical containers are synchronized to these optical signals and designed to accommodate various clients from Ethernet and SDH to different MPLS flavors. The subcontainers, both electrical and optical can be multiplexed into each other according to the hierarchy and therefore allow for an efficient adaption of packetized traffic streams to optical connections. Since electrical containers and optical TDM-containers match each other, virtual connections of very low jitter can be realized for a variety of client services. The drawback of this architecture is that additional OTN-switching components are required and need to be managed together with the other layers introducing additional complexity. While OTN systems often come with their own proprietary NMS systems, they can also be controlled by GMPLS [213, 221, 223].

Many of the protocols in these architectures predate the advent of SDN and therefore were designed with individual control planes in mind. The development of the ASON architecture [98] and the related protocols in GMPLS [211] provide a framework allowing NEs to directly interact with control plane functions on other NEs. This concept was further extended to support an external system for path computation, the so-called Path Computation Element (PCE) [214]. A stateful and active PCE, which is essentially an SDN-controller, can make use of the traffic engineering-extensions to Interior Gateway Protocols (IGPs) [210, 215, 225] to gather network state information and control connection setup by GMPLS via the PCE Communication Protocol (PCEP) [216].

Alternatively, new protocols like OpenFlow [181] with its optical extensions [182] or protocols using YANG[1] models [218] such as Network Configuration Protocol (NETCONF) [219] or gNMI[2] can be utilized. These go beyond high-level control plane interfaces on NEs and allow a centralized network controller to directly query and instruct NEs regarding their configuration details and forwarding behavior in a true SDN-architecture [222]. Several multi-layer-capable SDN-controllers, with and without GMPLS as a mediator, have been developed and deployed in large-scale testbeds [127] showing the feasibility of the approach for various data plane architectures.

[1] RFC 6020 introduces the name as is, while RFC 8328 [224] states it to mean “Yet Another Next Generation”.
[2] Recursive acronym for “gRPC Remote Procedure Call Network Management Interface”.

# B Networking Problems and Constraints

## B.1 Other Networking Problems not explored in this Thesis

Apart from the aforementioned problems, many others exist, which we will not or at least not explicitly tackle in this monograph. Following is a short overview of such related problems.

The problem of identifying a link cost or link weight system for an entire topology, i. e., a set of link cost values to be used by a shortest path routing algorithm as explained in Section 2.2.3.1, is a common problem in networking, since such systems are a typical input to non-SDN control planes. Finding a single set of cost values such that the resulting routing implicitly fulfills the required criteria in the face of changing network demands is a complex optimization problem. Several versions of such problems and solution approaches can be found in Pióro and Medhi [194, ch. 7].

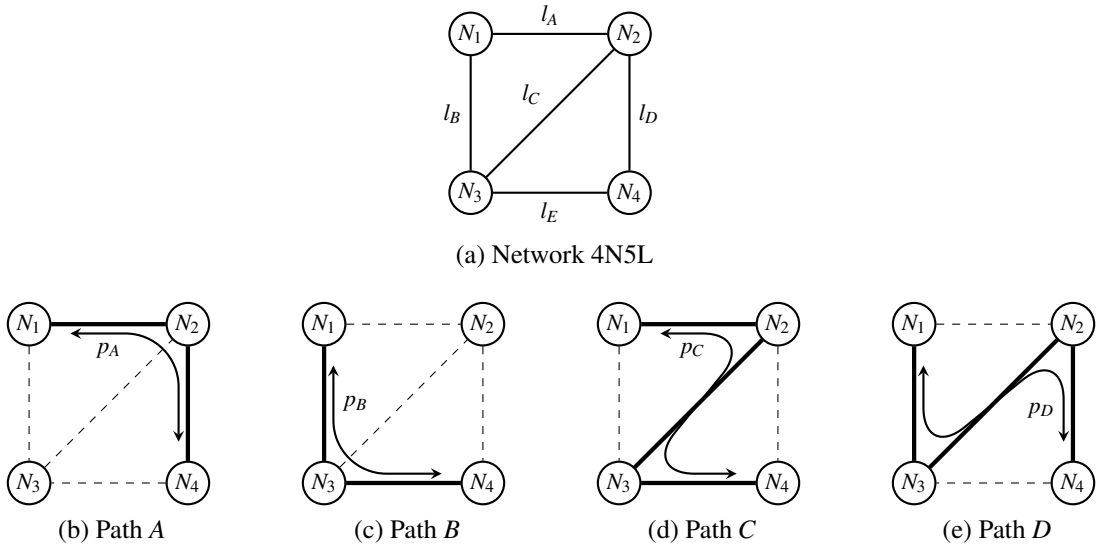
Several sub-problem types are concerned with the effects of temporal variations in traffic demands and how to address them directly as part of the optimization process. In so-called *multi-period design* problems, the network's behavior in reaction to evolving traffic demands over longer periods is considered in the design phase [194, ch. 11.2]. Alternatively, uncertainty regarding future traffic behavior may also be explicitly modeled and incorporated into the planning problem [257].

*Multi-hour design* problems [194, ch. 11.1] exploit predictable short-term traffic fluctuations such as traffic peaks in geographical areas of different time zones being shifted from another. Rather than designing the network for a combined maximum of all areas that never actually occurs, resources are placed such that they can be re-dedicated for traffic from different areas. This is especially relevant for reconfiguration problems, which may yield differing results for different schedules or triggers [282].

Apart from temporal aspects there exist a number of spatial allocation problems as well. In physical network design there are several versions of the node location problem with and without link connectivity [194, pp. 212–230], typically constrained by the requirement of connecting given geographical areas. Beyond the nodes itself, there are also individual placement problems which aim to allocate different NEs to node locations, e. g., SDN controllers [284], 3R regenerators [171], wavelength converters [101], and virtualized network functions [48].

Service *orchestration* problems include the management of heterogeneous systems such as hybrid networks or NFV requirements such as data center resources colocated with network nodes in network planning and operation [131] and *virtual network embedding* [82] create isolated customer networks on the dual-layer network infrastructure of the NSP.

For network operation, finding a sequence of transitional steps between two pre-determined configurations under various timing and resource constraints is also a relevant problem [293], especially in situations of resource shortage, where make-before-break is not an option. Such serialization problems are also relevant for spectrum defragmentation representing a specialized version of network reconfiguration with the goal of reducing stranded spectrum [260].




(a) Network 4N5L

(b) Path $A$ (c) Path $B$ (d) Path $C$ (e) Path $D$

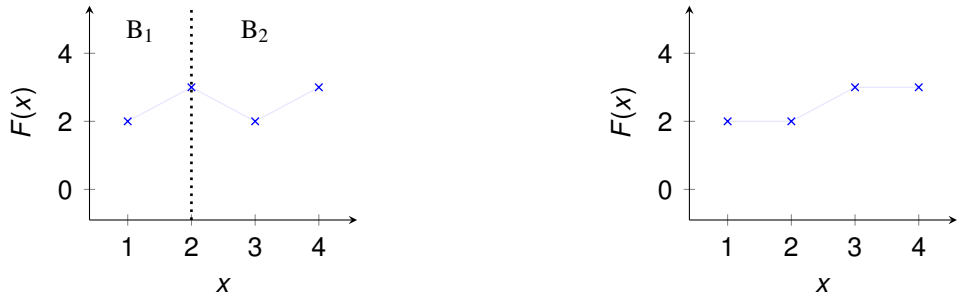
Figure B.1: Example graph enumeration of paths between $N_1$ and $N_4$

Finally, in routing various types of grooming problems [167, chs. 13 & 14] exist. Since traffic demands and circuit granularities rarely match, it becomes a complex task to aggregate traffic and circuits such that demands meet capacities with the least amount of wastage. This becomes especially challenging when considering the resource hierarchies of ports, line cards and shelves, where wasting a little bit of capacity in a circuit can help save on line cards.

## B.2 Routing Problem Example

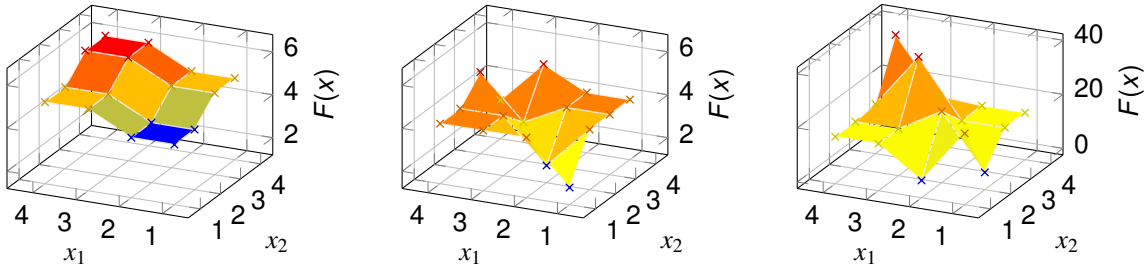
To illustrate the occurrence of the effects mentioned in Section 3.2.3, this section will provide a small-scale example problem. Since topology design and routing problems for multi-layer networks share many traits, this example will focus on routing problems due to their more intuitive relation between search space and index set. The first version of the example only requires routing of a single traffic demand without any QoS constraints. We consider every link to require an active circuit, i. e., no bypass circuits are allowed, and will use the number of required circuits as the measure of resource efficiency. Given the topology in Figure B.1 and Equation (3.2), we can determine an upper bound for the number of possible paths for a node pair to be 5, which can easily be enumerated. Since the graph 4N5L is not a full graph there are only 4 possible routes, denoted as $p_A$ to $p_D$, between nodes $N_1$ and $N_4$ as illustrated in the subfigures.

Furthermore, the example assumes that the entire topology in Figure B.1a is available and that a single demand exists between $N_1$ and $N_4$ which requires exactly the capacity offered by a circuit. It may therefore be considered as an uncapacitated ICR problem. A solution using either path $A$ or path $B$ will then require two circuits in total, while paths $C$ and $D$ each require three circuits. For a loop-free routing problem the combinatorial choice is therefore between these four paths. In an indexed set of these paths, each can be represented by a unique integer index to this set, such that a solution can be described by $x \in \{1,2,3,4\}$. In this case there are 24 permutations to assign indices to these paths, each leading to different relative gradients and number of basins. In the following solution enumeration figures, we will indicate local gradients by connecting the dots of individual neighboring solutions. The index order $\langle p_A, p_C, p_B, p_D \rangle$, which would assign




(a) Index order $\langle p_A, p_C, p_B, p_D \rangle$ (b) Index order $\langle p_A, p_B, p_C, p_D \rangle$

Figure B.2: Search space for simple routing problem (lines indicate local gradients)



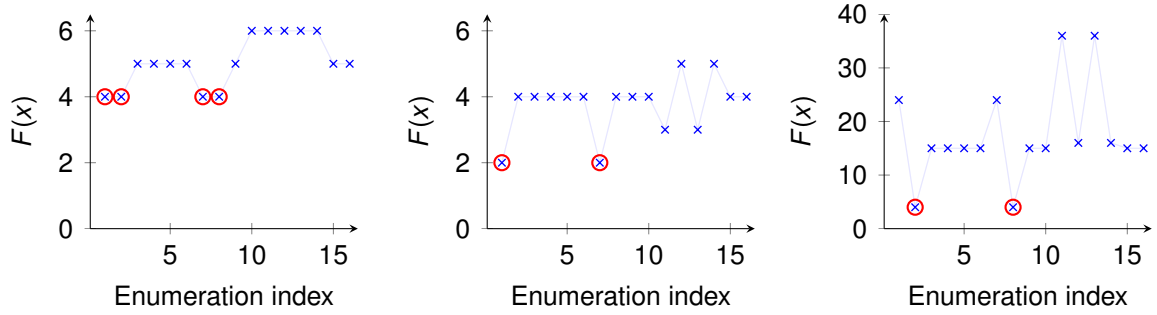

(a) Additive circuits (b) Shared circuits (c) Constrained circuits

Figure B.3: Search spaces for routing problem of 2 demands (surfaces indicate local gradients)

index 1 to $p_A$, index 2 to $p_C$ and so on, would have a basin $B_1$ for indices 1 and 2, as well as a second basin $B_2$ for indices 2, 3 and 4 as shown in the solution enumeration in Figure B.2a. In comparison, the index order reflecting the number of hops of each path as shown in Figure B.2b is superior, since it results in a more simple gradient approximation and a smaller number of basins in the search space. Even if the exact objective function were unknown, knowing that it somehow measures resources and that more hops typically require more resources, makes this order a good choice for the index.

When adding a second demand between the same nodes, there is a second choice among those paths for the new demand, such that combinations of the path for the first demand $x_1$ and the path for the second demand $x_2$ need to be considered. This results in a drastic increase of the input space to a total of 16 possible states, highlighting the fast growth of combinatorial problems. A solution to this problem can therefore be represented by a tuple $\langle x_1, x_2 \rangle = x$ with $x \in \{1,2,3,4\} \times \{1,2,3,4\}$. In Figure B.3 the horizontal axes correspond to the indices of the two demands and the surfaces indicate local gradients. Figure B.3a shows the search space when both demands require their own circuits and cannot be groomed into one circuit. The four combinations where both demands use either path A or path B, which mark the corners of the blue surface in the figure, are the optimal solutions in this case.

When the capacity required by both demands is low enough, such that their sum is less than provided by one circuit, they can be groomed into the same circuits. This is a common occurrence for TDR problems in multi-layer networks. Therefore, whenever the selected paths share edges, this edge requires only one circuit and thus out of the original 4 optima, only $\langle 1,1 \rangle$ and $\langle 2,2 \rangle$ are now optimal, each requiring a total of 2 circuits instead of the original 4 as shown in Figure B.3b. Figure B.3c shows the search space for a constrained problem where




(a) Additive circuits (b) Shared circuits (c) Constrained circuits

Figure B.4: Search spaces for routing problem of 2 demands (lines indicate local gradients)

each demand requires their own circuit, but the link capacity is limited. As a result of this, any link may only carry one circuit at a time. If more circuits are required, a penalty function will add a value of 10 to the original resource objective function, such that the global optima are now $\langle 1,2\rangle$ and $\langle 2,1\rangle$.

In order to ease comparing the values of the search space, especially for higher dimensional problems, we will use a universal search space representation, based on a single contiguous index to enumerate the solutions. Finding a general, yet meaningful order relation to assign a single index to all combinatorial states is difficult, since structurally similar vectors of high dimensions cannot all be placed next to each other in a linear order. We will use an enumeration that starts from a vector where all components are initialized to the smallest value and then increments the vector's components following a Gray code sequence generalized for integer values.

The same solution spaces as in Figure B.3 are visualized using this method in Figure B.4. The advantage of this visualization is that subsequent solutions in the index only differ by a value of "1" in a single component of their respective vectors. This means that all solutions that are neighbors in the index are also neighbors in search space, but not all neighbors in search space are also neighbors in the index. This can be observed in Figure B.4a, where the four optima are not all immediate neighbors in the index, while they are in search space as visualized by the corners of the blue surface in Figure B.3a.

While the search space of the original version with the purely additive objective function metric has a clear macro-structure with only a single basin in search space, adding capacity sharing or resource limitation constraints lead to complex search spaces with additional local minima even for this very small-scale problem. Furthermore, the effects of graph symmetries, as introduced in Section 3.2.3, are also clearly visible around index 5 of Figure B.4a and by the axial symmetry of Figure B.3. It is noteworthy that the global optima of the complex problems still reside in the same area of the search space, as for the case of one demand. Both, capacity sharing and resource constraints are the hallmarks of typical multi-layer problems, such that the outlined effects are commonly present in large-scale problems as well.

## B.3 Reference Network Design

Searching for a representative multi-layer transport network, we analyzed several sources. The remarkable work of the team behind the Rocketfuel Topology Engine [256] includes a dataset[1] of ISP topologies. While very informative, the topologies contained are by now almost 20 years old and mostly limited to the IP layer, since the team inferred the topologies from public IP addresses of router interfaces. SNDlib and The Internet Topology Zoo[2] provide newer data, which has been assembled from various sources and includes fiber topologies as well as higher layers. However, none of these sources include matching packet- and fiber-layer topologies with representative numbers of nodes and links.

We therefore decided to create our own reference network based on a large commercial network and we found a provider that publishes information about their networks and capabilities, including data center and peering sites, layer-2 and layer-3 capabilities, as well as fiber connectivity, on their website [32]. While the individual maps have differing geographic scopes, they all contain data on the continental USA, such that we decided to focus on this area and only consider nodes and fiber links contained therein. In order to create a homogeneous transport network, we made a number of changes to the original data.

We removed the three international fiber links to Tokyo and London and another to Honolulu for our version of the layer-1 network. Furthermore, we realized that some fiber links seem to end at different sites within the same metropolitan areas, that sometimes have no connections between them in the map. We decided to merge these nodes, because it is highly unlikely that they are not connected to each other and because optimizing traffic and optical circuits from one side of town to the other is on a different detail level than optimizing for the same for links between cities or states. We have therefore merged

- Burbank, Anaheim and Emeryville into Los Angeles,
- Palo Alto into San Francisco,
- Los Osos into San Luis Obispo,
- Highlands Ranch into Denver,
- Lamy into Santa Fe,
- Cermak into Chicago,
- Dublin (OH) into Columbus,
- Westfield (MA) into Springfield (MA),
- Winter Park into Orlando and
- Newark into New York City.

Consequently, we have also eliminated the four fiber links that had existed between the merged nodes in the original data. Furthermore, we had realized that some links passed right by major population centers, where other sites are located, such that it is unlikely, that there is no common site. The fiber between Ft. Worth and Las Vegas in all likelihood passes Albuquerque following Interstates 40, 27 and 20, such that we replaced it with a fiber from Ft. Worth to Albuquerque

[1]Accessible at https://research.cs.washington.edu/networking/rocketfuel/maps/rocketfuel_maps_cch.tar.gz, visited on 2025-05-09

[2]Accessible at http://www.topology-zoo.org/dataset.html, visited on 2019-06-24

and another from Albuquerque to Las Vegas. We did the same for the link from Washington, DC to Rocky Mount, NC as it passes by Richmond.

Finally, some of the sites were not biconnected, such that we chose to add an additional link to another node, typically following large roads. These sites are

- Augusta, with a new fiber link to Atlanta,
- White Plains, with a new fiber link to Bridgeport, and
- New London (CT), with a new fiber link to Providence.

The resulting topology, which we have referred to as *CL–Max*, covers the entire continental USA and consists of 149 nodes and 206 fiber links. It is shown in Figure 5.26. Using the information about the data center locations, we derived the second topology, which we have referred to as *CL–DC*, that consists of the 19 sites, which feature data centers. We created the 35 links between them based on the shortest paths between the nodes in the *CL–Max* topology with some additional manual tweaks.

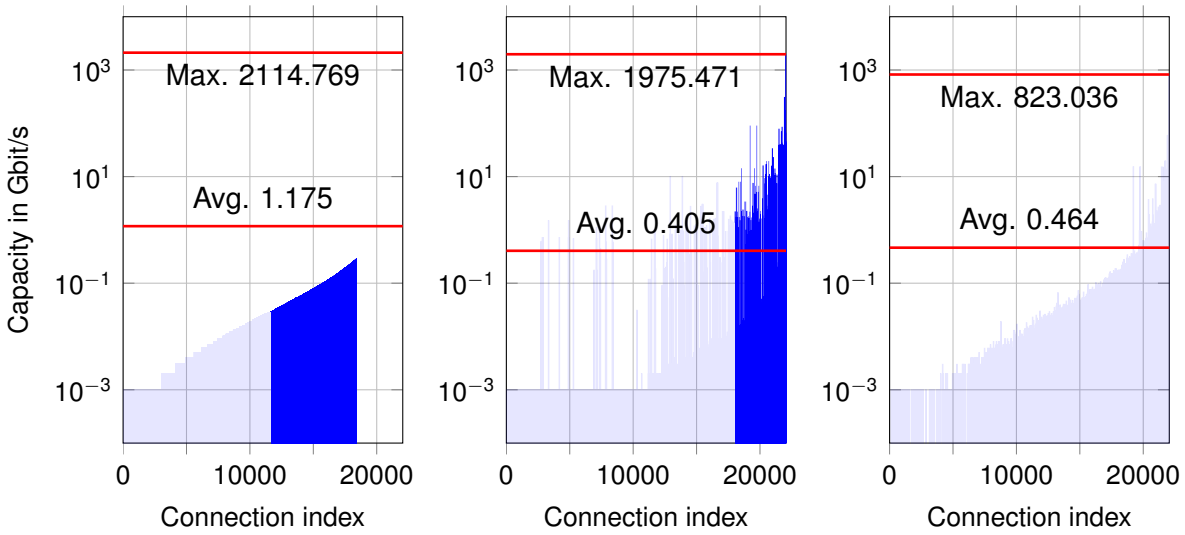
Comparing *CL–Max* to *CL–DC*, it is interesting to note that not only is the average degree lower than for *CL–DC*, but the density is even lower by an order of magnitude. This is a common effect for networks with large numbers of nodes, where only a smaller number of nodes serve as connection hubs, while the vast majority only fulfills the 2-connectedness requirement in order to allow for a minimum of geographic diversity. This also means that having many 2-connected nodes between these hubs, leads to a certain amount of detours resulting in increased fiber lengths between these hubs. This also results in a reduced average fiber length due to the nodes in-between. A prominent example for both effects is visible in the northwest of Figure 5.26, where both cases exist in parallel: A direct fiber connects Denver in Colorado directly to Tukwila in Washington State, while a much longer sequence of fibers between these cities serves many smaller nodes throughout Wyoming and Montana.

## B.4 Traffic Generation

The traffic demand matrices used in the evaluations for *CL–DC* and *CL–Max* were both created using a traffic generation algorithm developed by Enderle and the author [72]. This algorithm creates demand matrices based on a number of different values. First, it uses traffic shares, which have been determined by Cisco and published in their VNI [40] and GCI studies [39]. This is information about which type of applications will consume which share of bandwidth.

The second data source is population data, since larger groups of people produce more data traffic and communicate the most with other large groups of people. We have obtained population size information from the 2018 estimate by the U.S. Census Bureau [6] for each city or metropolitan area, where a node is located and also data on the state population [7]. All population numbers have been rounded to the nearest thousand figures, except for settlements smaller than 1000 which have been rounded to 1000. We then redistributed the population of each state proportional to the population size for each contained settlement that houses a PoP with the rationale, that access networks will carry the traffic of the surrounding areas to these transport network nodes.

The third data source are public results [240] of the DISCUS project [241], which resulted in a traffic simulation model based on activity profiles of different services, we had correlated with the Cisco data and internet exchange and data center locations, since the traffic profiles at




(a) Best effort class (b) Availability class (c) Latency class

Figure B.5: Capacity distribution for traffic demands of different QoS classes for *CL–Max*

such locations are different from regular nodes. Finally, we also included time zone information to adjust relative activity profiles relative to their local time.

As previously stated, we chose to model three different QoS-classes. The first is a best-effort class, the second requires a latency of at most 10 ms, and the final one requires a minimum availability requirement of 0.99. We fed the data of *CL–Max* and *CL–DC* to the generator to obtain the traffic matrices as used in the evaluation. For the smaller topology, this resulted in 812 unidirectional or 406 bidirectional demands, which are given as unidirectional since their required data rates are not symmetric in both directions. The sum of all traffic reaches 47.4 Tbit/s of which 59.3 % are best-effort, 23.3 % are latency-sensitive and 17.4 % are availability constrained traffic demands. Figure 5.18 in Section 5.4.1.3 shows the exact distribution of demands.

The demand matrix for *CL–Max* is obviously much larger and contains 47 094 unidirectional traffic demands, the sum of which amounts to 44.8 Tbit/s of which 57.9 % are best-effort and 22.9 % are latency constraint demands, finally leaving 19.2 % for the availability class. The exact demand profiles for each class are shown in Figure B.5 and listed by a connection index, where each index represents a unique pair of source and destination nodes.

Note, that the lowest resolution for demand capacity used is 1 Mbit/s, which leads to the staircase effect in the figures. The relatively large amount of low-speed traffic demands is caused by the fact, that many small nodes in rural areas attract very little traffic due to the absence of large population centers or infrastructure like internet exchanges or data centers. However, there are some business-related activities even in rural areas, which lead to the spikes visible in Figure B.5.

# C Further Evaluations and Results

## C.1 Acceleration by Parallel Execution

As stated in Section 5.1.2 a very import factor for genetic algorithm methods is their potential for increased runtime performance by parallel execution thanks to their inherent parallelism. In order to quantify the attainable improvement incurred by multithreaded execution, the problem given in Section 5.3 has been solved repeatedly with the same parameter set. However, rather than using a runtime limit, for this experiment the algorithm is run with a fixed number of 100 generations each of which required 200 evolutions. We varied the number of threads used for the evolution tasks and we used three different servers, each with two CPUs, as shown in Table C.1. To control for statistical effects, each of these simulations has been performed 10 times using different sets of random numbers. Figure C.1 shows runtime and speedup, i. e., the

Table C.1: Servers used for runtime comparisons

| CPU Type | Num. CPUs | Frequency Turbo | Frequency Base | Cores | Cache | Threads Hw | Threads Sw | RAM |
|---|---|---|---|---|---|---|---|---|
| Xeon E5-2630 v2 | 2 | 3.1 GHz | 2.6 GHz | 6 | 15 MB | 12 | 24 | 220 GB |
| Xeon E5-2650 v2 | 2 | 3.4 GHz | 2.6 GHz | 8 | 20 MB | 16 | 32 | 220 GB |
| Xeon E5-2640 v4 | 2 | 3.4 GHz | 2.4 GHz | 10 | 25 MB | 20 | 40 | 378 GB |

ratio of the present runtime to the runtime of the single-threaded case, for the three servers with varying numbers of threads. The data points indicating runtime include error bars representing the minimum and maximum values observed over the ten independent repetitions. The deviation from the average value always remained below 5%, such that most of the error bars in this figure are too small to be visible, except for the initial runs with few threads.

As expected, the faster and larger CPUs outperform the slower ones in almost all cases and an increasing number of threads significantly boosts performance. The largest speedup on the three models is between 6.5 and 7.25. Figure C.2 shows the runtime values for larger core numbers in more detail. With the exception of the 2x8-core machine, the best performance is reached when the number of evolution threads is identical to the total number of cores in the system, as indicated by the vertical bars at 12, 16 and 20 cores. Beyond that, the CPUs use multi-threading to accommodate up to 2 threads per core which incurs a significant penalty, most likely due to the increased number of required context switches and reduced cache locality. When operating in this regime, there is a second local minimum for each of the machines as indicated by the vertical bars at 24, 32 and 40 threads, which are the maximum number of threads that can be simultaneously supported on the respective machines. After this point, performance slowly, but continuously degrades for each of the three data sets.

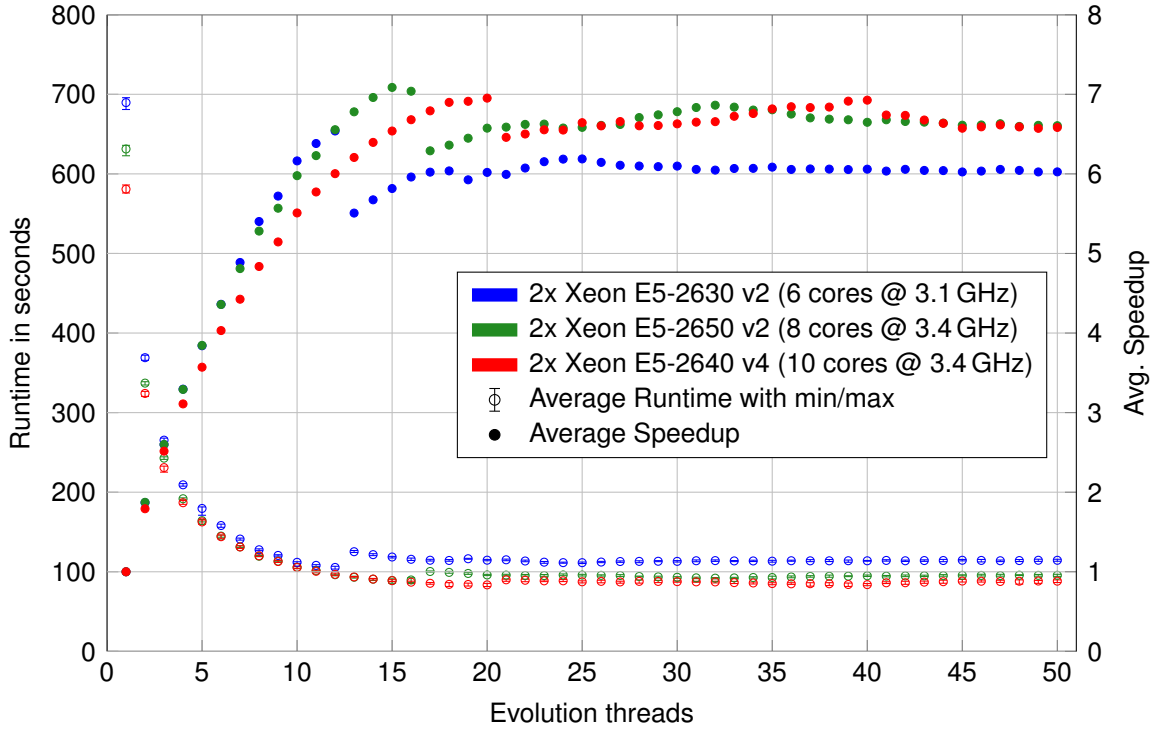



Figure C.1: Runtime and speedup over number of threads used for evolutionary operations

It should be noted that the given number of threads is not the total number of threads instantiated by the software. Safe for the initial single-threaded case, there is always an additional thread managing the population, which runs anticyclically to the evolution threads. Also, to maximize throughput, Java's *ParallelGC* has been used which creates additional threads according to an internal heuristic. For the presented experiments, 6 to 12 threads have been observed. In the single threaded case, the user time of the software thread has been compared to the system time of the entire run to determine an upper bound for the time spent on garbage collection and other system duties occurring in parallel. This time has been found to be consistently less than a few percent of the overall runtime.

Further scenarios have been run, showing a number of secondary effects, e. g., more complicated network data structures or excessively large pools of offspring showed effects which might be attributed to reduced cache locality and memory access latencies, but two core observation held true for all experiments. One, that a significant performance increase is visible when increasing the number of evolution threads up to a certain limit, and two, that this limit lies close to or matches the number of CPU cores, rather than the maximum number of simultaneously executable threads through hardware multithreading.

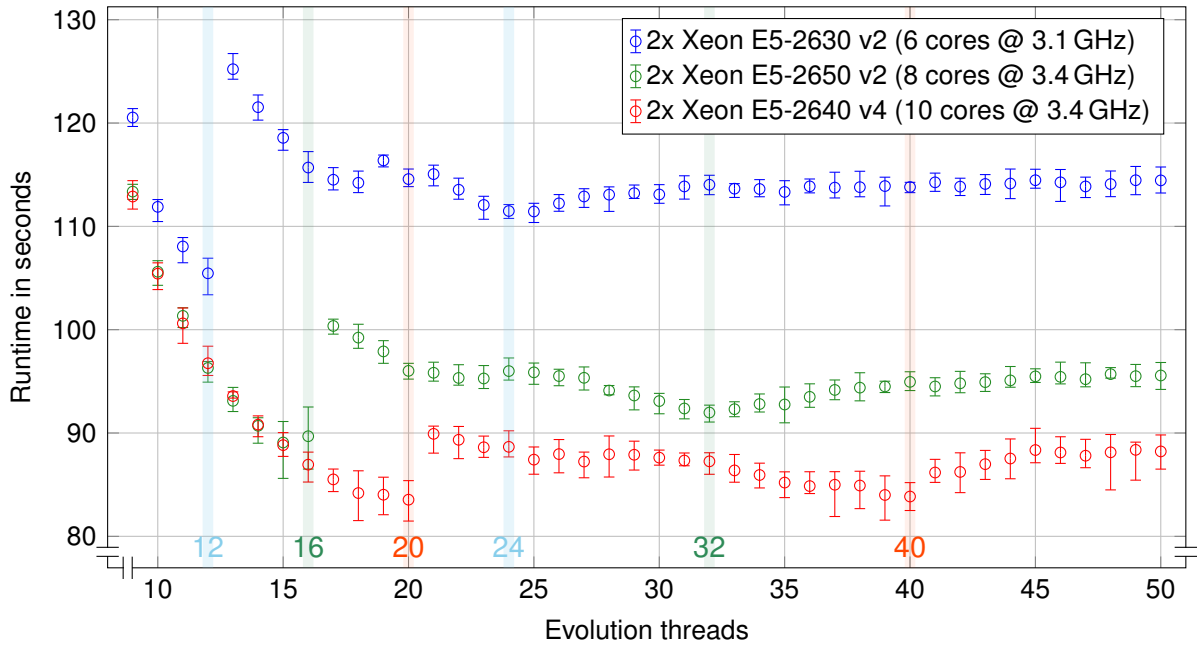
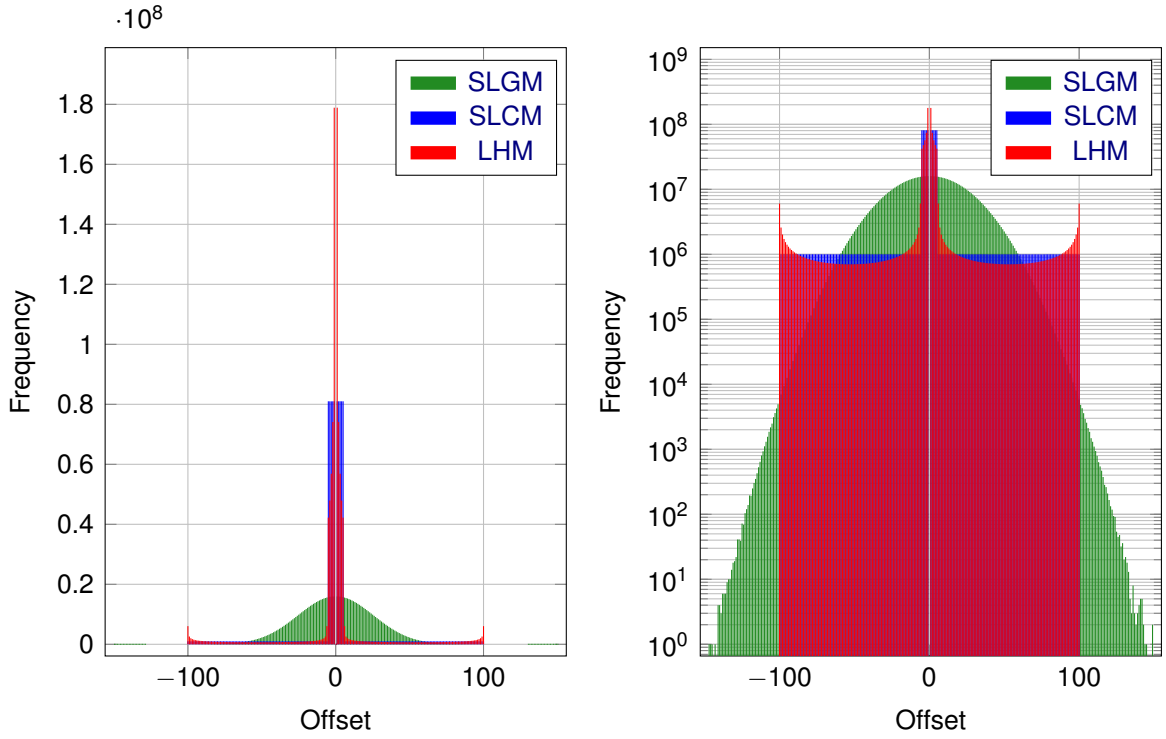



Figure C.2: Runtime over number of threads used for evolutionary operations, detailed view

## C.2 Mutation Operator Evaluation

In order to evaluate the performance of our LHM operator against its competing operators, we created an individual chromosome and applied a mutation operator one billion times, recording the resulting step size and the required time. We did this for LHM, SLGM, and SLCM on each of three servers presented in Table C.1. Figure C.3 shows the histogram of the obtained step sizes in a linear and a logarithmic view. The linear view shows, that LHM successfully achieves the intended pattern of very high probabilities around 0 and slightly elevated probabilities close to $v_h = 100$. The comparison in the logarithmic plot shows the difference compared to SLGM, with a relatively wide distribution.

Figure C.4 shows the time required for this computation. Our LHM operator not only achieved the intended distribution but was also faster an all the systems under investigation.




(a) Linear Axis View

(b) Logarithmic Axis View

Figure C.3: Histograms of mutation operators

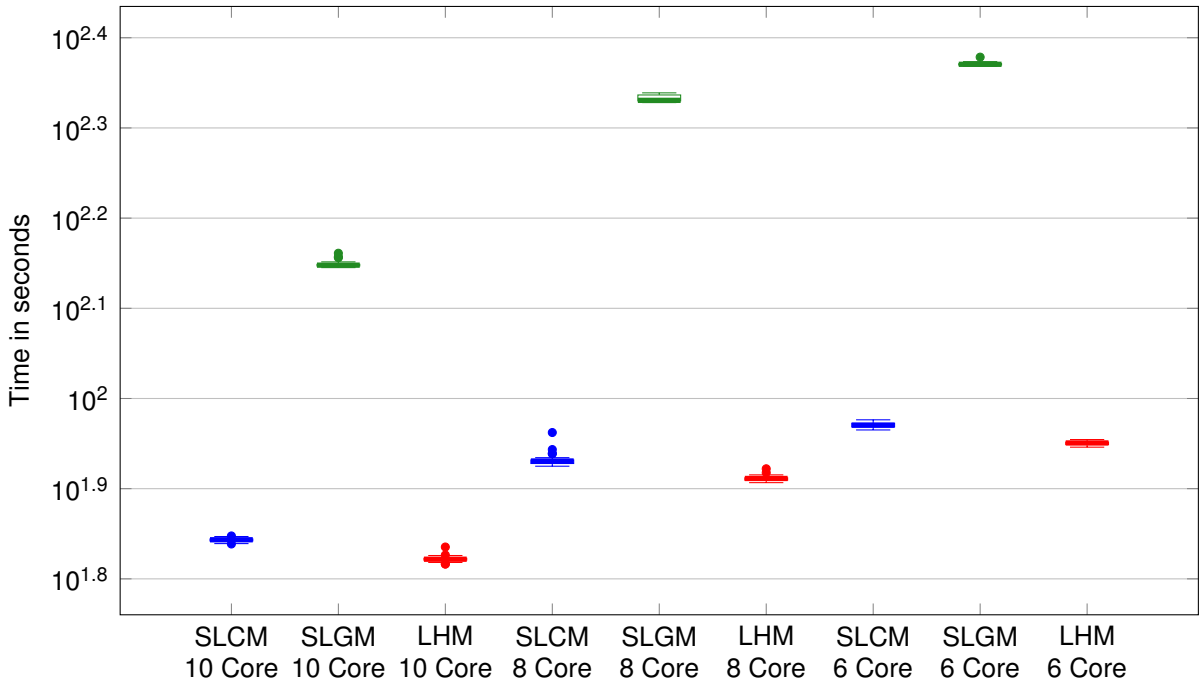



Figure C.4: Time needed to compute 1 billion mutations for different operators

# D Mathematical Foundations

## D.1 Closeness Centrality

Closeness centrality [233] is a measure that describes, how central a given vertex $s$ is to a given connected graph $g$. It is based on the idea of considering the shortest distance from any vertex in the graph to the vertex $s$. Normalized closeness centrality is the inverse of the sum of all these distances, multiplied by the number of vertices in the graph excluding $s$. More formally, given a graph $g = \langle V, E \rangle$ with $s, d \in V$ and $s \neq d$, the normalized closeness centrality is defined as follows.

$$C(g,s) = \frac{|V| - 1}{\sum_{d \in (V \setminus \{s\})} f_{\text{dist}}(g,s,d)} \tag{D.1}$$

Here, $f_{\text{dist}}$ is the length of the shortest path between $s$ and $d$ in graph $g$. Furthermore, we determine the average closeness centrality of the entire graph $g$ according to Equation (D.2)

$$C(g) = \frac{\sum_{s \in V} C(g,s)}{|V|} = \sum_{s \in V} \sum_{d \in (V \setminus \{s\})} \frac{1}{f_{\text{dist}}(g,s,d)}. \tag{D.2}$$

Note, that centrality values depend on the number of nodes as well as the structure of the graph, such that the average closeness centrality of a ring topology decreases with an increasing number of nodes. For $|V| \to \infty$, a line graph will converge to 0, a star graph converges to 0.5, while a full mesh converges to 1.

## D.2 Number of loop-free Paths in a complete, bidirectional Graph

Given a bidirectional, complete simple graph $G$ of $n$ nodes, i. e., a loop-free graph where any node $s$ has exactly one bidirectional link to any other node $t$, the number of all possible paths $c_{\text{paths}}(n)$ between any node $s$ and any node $t$ with $t \neq s$ can be calculated by Equation (D.3).

$$c_{\text{paths}}(n) = \sum_{i=0}^{n-2} \frac{(n-2)!}{i!} \tag{D.3}$$

**Proof**

A graph with at least two distinct nodes $s$ and $t$ has at least two nodes. Since we consider bidirectional simple graphs, a graph of two nodes has exactly one edge between these nodes, such that exactly one path exists. Adding one more node $a$ to the graph, allows for a single alternative path via the added node. Adding a fourth node $b$, however, adds several new paths. There is the choice of using none, one or both of the added nodes as intermediate hops. In case of one intermediate hop, this can either be $a$ or $b$. For two intermediate hops, the paths are either first via $a$, then $b$ or first via $b$, then $a$. This results in a total of 5 paths. Further investigating the smallest four graphs, yields the following values.

$$\begin{aligned} c_{\text{paths}}(2) &= 1 &&= 1 \\ c_{\text{paths}}(3) &= 1+1 &&= 2 \\ c_{\text{paths}}(4) &= 1+2+2 &&= 5 \\ c_{\text{paths}}(5) &= 1+3+6+6 &&= 16 \end{aligned}$$

Therefore, adding an extra node results in two things. First, it increases the maximum length of a single path by one more node, adding all possible choices for a path of this length. Second, it presents itself as an additional choice in all shorter paths, except for the direct path. The choice for a given path length is among all the nodes that are neither $s$, nor $t$. These are $n-2$ distinct nodes. Since we require loop-free paths, every node can appear at most once in a path. The first choice is among all $n-2$ nodes. If a second hop is taken in a path, this leaves $n-3$ choices and so on. Therefore, we can reformulate the initial observations as follows.

$$\begin{aligned} c_{\text{paths}}(2) &= 1 \\ c_{\text{paths}}(3) &= 1+(3-2) \\ c_{\text{paths}}(4) &= 1+(4-2) &&+(4-2)(4-3) \\ c_{\text{paths}}(5) &= 1+(5-2) &&+(5-2)(5-3)+(5-2)(5-3)(5-4) \end{aligned}$$

Consequently, we can infer the following for the general case of $n$ nodes.

$$\begin{aligned} c_{\text{paths}}(n) &= 1+\prod_{j=2}^{2}(n-j)+\prod_{j=2}^{3}(n-j)+\prod_{j=2}^{4}(n-j)+\cdots \\ &= 1+\frac{(n-2)!}{(n-3)!}+\frac{(n-2)!}{(n-4)!}+\frac{(n-2)!}{(n-5)!}+\cdots \\ &= \frac{(n-2)!}{(n-2)!}+\frac{(n-2)!}{(n-3)!}+\frac{(n-2)!}{(n-4)!}+\frac{(n-2)!}{(n-5)!}+\cdots \\ &= \sum_{i=0}^{n-2}\frac{(n-2)!}{i!} \end{aligned}$$

□

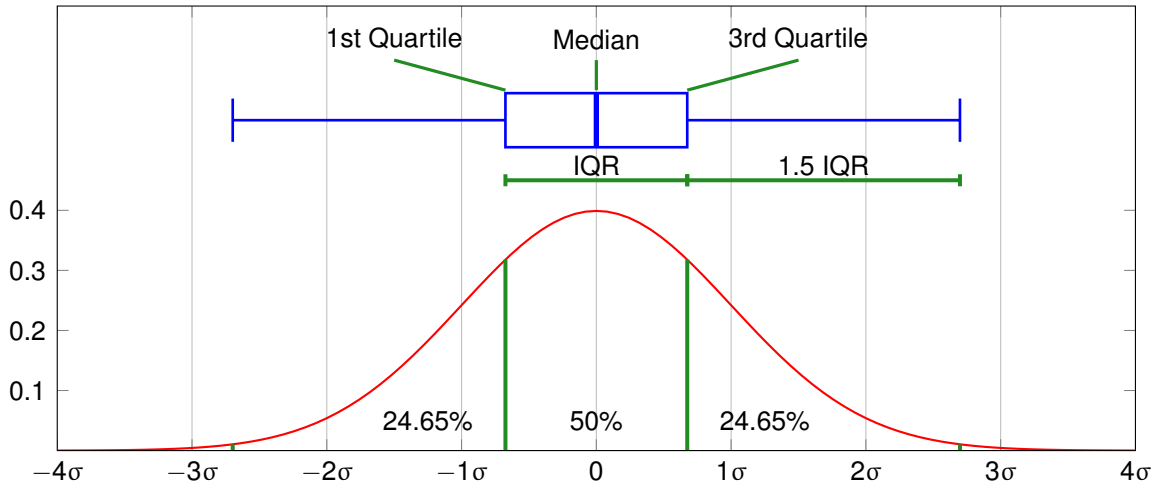



Figure D.1: Box plot of the Gaussian distribution $\mathcal{N}(0, \sigma)$.

## D.3 Box Plots

The box plot, or in this case more precisely the box-and-whisker plot as originally developed by Tukey [267], is a visualization method from descriptive statistics, which provides a fast, but detailed overview on the distribution within a numerical data set. Unlike other statistical tools the box plot is non-parametric, i. e., it does not require any assumptions on the actual distribution behind the investigated data set, nor is it dependent on the correct choice of other parameters like bin sizes.

It uses the following characteristic figures to describe a data set. The first quartile and the third quartile form the bounds of the central box as shown in Figure D.1. The line within the box marks the median value and allows to judge the skewness of the data. The whiskers either stop at the largest value or represent at most 1.5 times the inter-quartile range. Figure D.2 illustrates this for a set of random data set indicated by red triangles, where the samples between 8 and 10 have values larger than $3\mathrm{Q} + 1.5 \cdot \mathrm{IQR}$ and therefore they are represented by separate dots, since they may be considered as outliers. In this case, the whisker on the right side extends to its maximum size of $1.5 \cdot \mathrm{IQR}$. The left whisker is shorter than the right, since the smallest sample value is larger than $1\mathrm{Q} - 1.5 \cdot \mathrm{IQR}$ and therefore the whisker only extends to the smallest sample value. We generate our box plots using PGFPLOTS and compute the quartiles for discrete samples using method “R7” [81, p. 502].

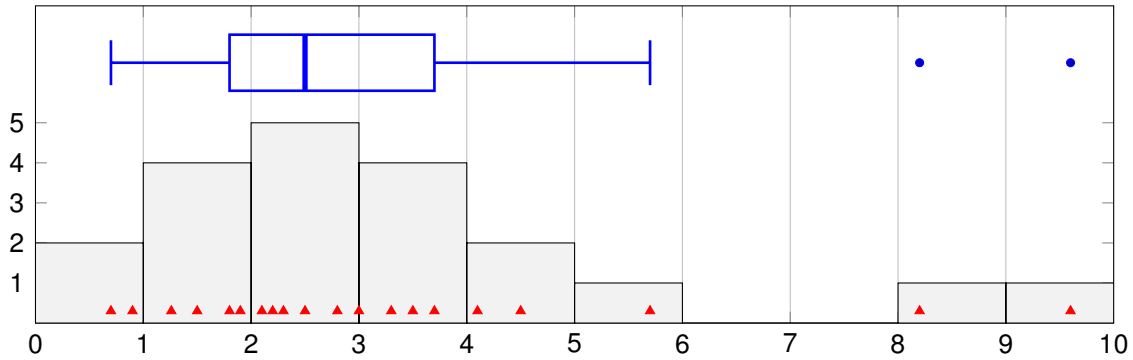



Figure D.2: Example box plot and histogram of 20 samples between 0 and 10

## D.4 Simulated Annealing in Detail

In addition to the short introduction and example in Section 3.3.4, the following sections will provide a more detailed explanation of Simulated Annealing. It is a probabilistic metaheuristic approach pioneered by Kirkpatrick *et al.* in 1983 [136] that is related to Stochastic Hill Climbing and ILS. Their approach is inspired by the annealing process from metallurgy in which metal is heated and slowly cooled according to a pre-defined schedule in order to support the crystallization processes which helps reduce defects in the material. At high temperatures atoms can reposition themselves more freely in the material. If cooled quickly, they will settle in a locally optimal position, but if cooled slowly they will gradually position themselves in equilibrium states, ultimately resulting in a globally homogeneous crystalline structure.

### D.4.1 Algorithmic Approach

As stated in Section 3.3.4, the approach to realize this is based on the Metropolis-Hastings algorithm [120, 161]. This is a Markov Chain Monte Carlo method used to create a sequence of samples which approximate an unknown distribution. Metropolis had originally used the Boltzmann distribution as a proxy for the distribution to be sampled when investigating the interaction of molecules, since it states that in a system of temperature $T$ the probability $p_s$ of encountering a state $s$ of energy $E(s)$ follows

$$p_s \propto e^{-\frac{-E(s)}{k \cdot T}}$$

with $k$ being the Boltzmann constant. Metropolis' algorithm therefore selects a candidate state $x_c$ from the neighborhood of the previous state $x_i$ in the Markov Chain according to the acceptance probability

$$p = \min\left(1, e^{-\frac{E(x_c)-E(x_i)}{k \cdot T}}\right)$$

such that it accepts if either $x_c$ has a smaller energy value than $x_i$ or their difference in energy values is likely to be encountered at the system's temperature given the Boltzmann constant. When applying this to the annealing context, a large system temperature translates to a high acceptance probability since atoms move more freely, while lowering the temperatures reduces this probability, until for $T = 0$, the candidate solution $x_c$ will only be accepted if it is in a less energetic state.

Kirkpatrick *et al.* therefore adapted this equation as the acceptance probability akin to its usage in Stochastic Hill Climbing, but using an abstract perturbation operation while also changing the simulated temperature of the system according to a cooling schedule of monotonically decreasing temperature values. This allows for a broad exploration of the search space at high initial temperatures and gradually translates to a more localized search for low temperatures. The basic Simulated Annealing procedure is shown as Algorithm 1 in Section 3.3.4.

### D.4.2 Theoretical Behavior

An important aspect of Simulated Annealing is the cooling schedule. The cooling schedules in metallurgy are precisely defined for a number of materials and other parameters which influence the crystallization process. For Simulated Annealing, such pre-defined schedules do not exist, since they would depend heavily on the unknown behavior of the black box objective function and also the perturbation procedure.

The initial temperature has to be sufficiently high and remain so for a sufficiently long number of iterations to allow for a meaningful coverage of the search space. Subsequent reductions in temperature should not be too rapid to avoid missing small basins, but not too slow to avoid excessively long runtimes. While Simulated Annealing has theoretically been proven to converge to the global optimum under a number of conditions [110], it is often difficult to guarantee these and identify an efficient cooling schedule without experimentation for the specific problem to be solved.

The method works best, when there is a macro-structure to the search space, such that reduction in temperature can match this macro-slope. On the contrary, search spaces with many small, but deep basins are difficult to navigate, because they are hard to differentiate early on, but difficult to escape from in later iterations. It is unsurprising that there exists a vast multitude of cooling schedules in literature including linear, logarithmic, exponential and other versions [180]. An idealized illustration of how a solution space may be sampled is presented in Figure 3.4, where the color of the arrows indicates the virtual temperature.

### D.4.3 Problem Adaption

Simulated Annealing requires a starting solution and a perturbation procedure. The starting solution does not need to have any specific form and the only requirement is that the perturbation procedure can accept it as an argument and create a solution of the same form. This makes Simulated Annealing very easy and straightforward to adapt to specific problems, especially in computer software, since it can directly act on any data structure.

The perturbation is therefore the essential part of the problem adaption. For the optimization of integer vectors the perturbation can be identical to the selection process in Stochastic Hill Climbing, such that it iterates through the solutions in the immediate neighborhood $\mathcal{N}_1$ sequentially until an acceptable candidate has been found. For arbitrary structures, however, this can be more difficult to define. It is essential that the perturbation can, possibly through an arbitrarily long sequence of applications, transform any solution to any other possible solution of the search space.

### D.4.4 Hyperparameters

Apart from potential perturbator-specific parameters, Simulated Annealing only requires a termination condition and a cooling schedule. The initial temperature needs to be selected according to the expected differences in the objective function values. This can be determined by choosing a temperature and drawing a number of samples and comparing the prospective probabilities for acceptance. Initially the probability should be close to 1.

The length of the cooling schedule should be designed according to the tolerable runtime duration, but its exact cooling rate is harder to determine. If local minima are good enough, the temperature can be decreased very slowly in the beginning, but rapidly in the end. If the goal is to match a global optimum as close as possible in a flat search space, a very slow cooling schedule is advisable.

Algorithm-specific termination conditions are often tied to the number of iterations, such that the algorithm terminates after a given number of steps. Another option is linking the termination to the temperature value. Once this value reaches 0, it is no better than a local search and may therefore not be able to improve the best solution any further.

### D.4.5 Aspects of Implementation

The memory footprint of Simulated Annealing is typically very low, since it only needs to store the best known solution, the current solution and the candidate solution. The fact that it is a trajectory-based method means that it sequentially generates and analyzes candidate solutions. The basic approach therefore offers little potential for parallelization.

The computational complexity of all operations is fairly low, except for the perturbation, where it varies with the exact implementation. As stated for the Hill Climbing methods, up to $2^n$ solutions may be probed for acceptance at low temperatures when the algorithm operates on $n$-dimensional integer vectors. However, for many combinatorial optimizations a permutation of a structure needs to be generated, which can be done by a simple swapping operation in constant time.

In Algorithm 1 a total of 4 evaluations of the objective function is required in every iteration. However, in situations where this evaluation is computationally complex, the already determined values can be stored and only the evaluation for the candidate solution has to be performed for each iteration.

## D.5 Ancillary Algorithms

For the sake of completeness, a number of mostly primitive ancillary algorithms used as part of the other algorithms are presented in this section.

#### Dijkstra's Algorithm

We did not implement Dijkstra's algorithm ourselves, but rather used a library function in JGraphT [162], which uses an implementation based on Fibonacci Heaps. For more information on Dijkstra's algorithm and related algorithms, cf. Grover [115, pp. 189–192]. The signature of the algorithm as it appears in our pseudocode is shown in Algorithm 11

**Algorithm 11** Dijkstra's algorithm to find a shortest path in a graph

**Require:** $\langle V, E \rangle$ is a graph
**Require:** $s, d \in V$
  **function** DIJKSTRA($\langle V, E \rangle$,$s$,$d$)
    **let** pathArr : $\{0..n\} \to E$
    ...  ▷ Library implementation
    **return** *pathArr*
  **end function**

#### K-Shortest Path algorithm

Just like with Dijkstra's algorithm, we did not implement a K-Shortest Path algorithm ourselves, but rather relied on the implementations in JGraphT [162]. For more information on K-Shortest Path algorithms, cf. Grover [115, pp. 195–199] or Pióro [194, sect. C.3]. The signature of the algorithm as it appears in our pseudocode is shown in Algorithm 12

#### Other Algorithms

Algorithm 13, which finds the node of largest degree in a given graph, and Algorithm 14, which implements the arg max function, are used within some of the other algorithms in this thesis.

**Algorithm 12** K-Shortest Path algorithm to find the k shortest paths in a graph

**Require:** $\langle V, E \rangle$ is a graph
**Require:** $s, d \in V$
  **function** KSHORTESTPATH($\langle V, E \rangle$,$s$,$d$,$k$)
    **let** pathArrArr : $\{0..n\} \to \{0..n\} \to E$
    ⋯ ▷ Library implementation
    **return** *pathArrArr*
  **end function**

**Algorithm 13** Find a vertex of maximum degree $deg_{max}$ in a given graph

**Require:** $\langle V, E \rangle$ is a connected, non-empty graph
  **function** MAXDEGREENODE($V$,$E$)
    **let** $nodeMap : V \to \mathbb{N}_0$
    **for all** $\langle s, d \rangle \in E$ **do**
      $nodeMap[s] \leftarrow nodeMap[s] + 1$
      $nodeMap[d] \leftarrow nodeMap[d] + 1$
    **end for**
    $node \leftarrow$ ARGMAX($nodeMap, V$) ▷ See Algorithm 14
    **return** node
  **end function**

**Algorithm 14** Determine $\arg\max_{x \in X} f(x)$ for finite mappings to natural numbers

**Require:** *map* is a mapping of a finite set $X$ to $\mathbb{N}_0$
  **function** ARGMAX(*map*,$X$)
    **let** $n \leftarrow 0$
    **let** $m \leftarrow \perp$
    **for all** $x \in X$ **do**
      **if** $map[x] \geq n$ **then**
        $n \leftarrow map[x]$
        $m \leftarrow x$
      **end if**
    **end for**
    **return** $m$
  **end function**

# Bibliography


[1] F. N. Abuali, R. L. Wainwright, and D. A. Schoenefeld. “Determinant Factorization: A New Encoding Scheme for Spanning Trees Applied to the Probabilistic Minimum Spanning Tree Problem.” In: *ICGA*. 1995, pp. 470–477.

[2] A. Ahmad et al. “Power-aware logical topology design heuristics in Wavelength-Routing networks”. In: *15th International Conference on Optical Network Design and Modeling - ONDM 2011*. Feb. 2011, pp. 1–6.

[3] C. W. Ahn and R. S. Ramakrishna. “A genetic algorithm for shortest path routing problem and the sizing of populations”. In: *IEEE Transactions on Evolutionary Computation* 6.6 (Dec. 2002), pp. 566–579. ISSN: 1941-0026. DOI: 10.1109/TEVC.2002.804323.

[4] H. Alshaer and J. M. H. Elmirghani. “Differentiated resilience services support in heterogeneous IP over wavelength division”. In: *IET Communications* 4.6 (Apr. 2010), pp. 645–662. ISSN: 1751-8636. DOI: 10.1049/iet-com.2009.0480.

[5] P. T. Anh Quang et al. “Multi-objective multi-constrained QoS Routing in large-scale networks: A genetic algorithm approach”. In: *2018 International Conference on Smart Communications in Network Technologies (SaCoNeT)*. Oct. 2018, pp. 55–60. DOI: 10.1109/SaCoNeT.2018.8585634.

[6] *Annual Estimates of the Resident Population for Incorporated Places of 50,000 or More, Ranked by July 1, 2018 Population: April 1, 2010 to July 1, 2018*. U.S. Census Bureau, Population Division. May 2019. URL: https://factfinder.census.gov/bkmk/table/1.0/en/PEP/2018/PEPANNRSIP.US12A (visited on 2019-08-18).

[7] *Annual Estimates of the Resident Population: April 1, 2010 to July 1, 2018*. U.S. Census Bureau, Population Division. May 2019. URL: https://factfinder.census.gov/faces/tableservices/jsf/pages/productview.xhtml?pid=PEP_2017_PEPANNRES&src=pt (visited on 2019-08-18).

[8] J. E. Baker. “Reducing bias and inefficiency in the selection algorithm”. In: *Proceedings of the second international conference on genetic algorithms*. Vol. 206. July 1987, pp. 14–21.

[9] S. Balasubramanian et al. “Multilayer Planning for Facebook Scale Worldwide Network”. In: *2017 International Conference on Optical Network Design and Modeling (ONDM)*. May 2017, pp. 1–6. DOI: 10.23919/ONDM.2017.7958520.

[10] N. Banerjee, V. Mehta, and S. Pandey. “A genetic algorithm approach for solving the routing and wavelength assignment problem in WDM networks”. In: *3rd IEEE/IEE international conference on networking, ICN*. 2004, pp. 70–78.

[11] C. Barnhart, N. Krishnan, and P. H. Vance. “Multicommodity flow problemsMulticommodity Flow Problems”. In: *Encyclopedia of Optimization*. Ed. by C. A. Floudas and P. M. Pardalos. Boston, MA: Springer US, 2009, pp. 2354–2362. ISBN: 978-0-387-74759-0. DOI: 10.1007/978-0-387-74759-0_407.

[12] U. Bauknecht. “Resource Efficiency and Latency in Dynamic IP-over-WSON Networks utilizing Flexrate Transponders”. In: *Proceedings of the 18th ITG-Symposium in Photonic Networks 2017*. 2017, pp. 36–41.

[13] U. Bauknecht and F. Feller. “Dynamic Resource Operation and Power Model for IP-over-WSON Networks”. In: *Proceedings of the 19th Open European Summer School and IFIP TC6.6 Workshop (EUNICE 2013)*. 2013.

[14] T. Bauschert et al. “Network planning under demand uncertainty with robust optimization”. In: *IEEE Communications Magazine* 52.2 (Feb. 2014), pp. 178–185. ISSN: 0163-6804. DOI: 10.1109/MCOM.2014.6736760.

[15] L. Berry et al. “A genetic-based approach to tree network synthesis with cost constraints”. In: *Second European Congress on Intelligent Techniques and Soft Computing-EUFIT*. Vol. 94. 1994, pp. 626–629.

[16] D. P. Bertsekas. *Network optimization: continuous and discrete models*. Belmont, MA, USA: Athena Scientific Belmont, 1998. ISBN: 1-886529-02-7.

[17] H.-G. Beyer and H.-P. Schwefel. “Evolution strategies – A comprehensive introduction”. In: *Natural Computing* 1.1 (Mar. 2002), pp. 3–52. ISSN: 1572-9796. DOI: 10.1023/A:1015059928466.

[18] R. Bhandari. “Optimal physical diversity algorithms and survivable networks”. In: *Proceedings Second IEEE Symposium on Computer and Communications*. IEEE. 1997, pp. 433–441.

[19] A. P. Bianzino et al. “A Survey of Green Networking Research”. In: *IEEE Communications Surveys Tutorials* 14.1 (Dec. 2012), pp. 3–20. ISSN: 1553-877X. DOI: 10.1109/SURV.2011.113010.00106.

[20] K. Bogineni et al. *SDN-NFV Reference Architecture v1.0*. Technical Report. Verizon Network Infrastructure Planning, 2016. URL: http://innovation.verizon.com/content/dam/vic/PDF/Verizon_SDN-NFV_Reference_Architecture.pdf (visited on 2018-02-19).

[21] R. Bolla et al. “Energy Efficiency in the Future Internet: A Survey of Existing Approaches and Trends in Energy-Aware Fixed Network Infrastructures”. In: *IEEE Communications Surveys Tutorials* 13.2 (July 2011), pp. 223–244. ISSN: 1553-877X. DOI: 10.1109/SURV.2011.071410.00073.

[22] R. Bolla et al. “The potential impact of green technologies in next-generation wireline networks: Is there room for energy saving optimization?” In: *IEEE Communications Magazine* 49.8 (Aug. 2011), pp. 80–86. ISSN: 0163-6804. DOI: 10.1109/MCOM.2011.5978419.

[23] J. A. Bondy and U. S. R. Murty. *Graph theory with applications*. Vol. 290. The Macmillan Press Ltd., 1976. ISBN: 0-444-19451-7.

[24] E. Bonetto et al. "Algorithms for the Multi-Period Power-Aware Logical Topology Design With Reconfiguration Costs". In: *J. Opt. Commun. Netw.* 5.5 (2013), pp. 394–410. DOI: 10.1364/JOCN.5.000394.

[25] M. R. Bonyadi and Z. Michalewicz. "A locally convergent rotationally invariant particle swarm optimization algorithm". In: *Swarm Intelligence* 8.3 (Sept. 2014), pp. 159–198. ISSN: 1935-3820. DOI: 10.1007/s11721-014-0095-1.

[26] G. Bosco et al. "On the Performance of Nyquist-WDM Terabit Superchannels Based on PM-BPSK, PM-QPSK, PM-8QAM or PM-16QAM Subcarriers". In: *J. Lightwave Technol.* 29.1 (Jan. 2011), pp. 53–61.

[27] R. Bowden et al. "Planarity of Data Networks". In: *Proceedings of the 23rd International Teletraffic Congress*. ITC '11. San Francisco, California: International Teletraffic Congress, 2011, pp. 254–261. ISBN: 978-0-9836283-0-9.

[28] G. E. P. Box and M. E. Muller. "A Note on the Generation of Random Normal Deviates". In: *Ann. Math. Statist.* 29.2 (June 1958), pp. 610–611. DOI: 10.1214/aoms/1177706645.

[29] J. Brandao, T. Noronha, and C. Ribeiro. "A genetic algorithm for maximizing the accepted demands in routing and wavelength assignment in optical networks". In: *Proceedings of the 11th Metaheuristics International Conference*. 2015.

[30] D. Breuer et al. "5G Transport in Future Access Networks". In: *ECOC 2016; 42nd European Conference on Optical Communication*. Sept. 2016, pp. 1–3.

[31] A. Cayley. "A Theorem on Trees". In: *The Collected Mathematical Papers of Arthur Cayley, Sc.D., F.R.S.* Vol. XIII. Cambridge University Press, 1897, pp. 26–28.

[32] *CenturyLink Network Maps*. CenturyLink Business. 2019. URL: http://www.centurylink-business.com/demos/network-maps.html (visited on 2019-08-12).

[33] E. K. Çetinkaya et al. "Multilevel resilience analysis of transportation and communication networks". In: *Telecommunication Systems* 60.4 (Dec. 2015), pp. 515–537. ISSN: 1572-9451. DOI: 10.1007/s11235-015-9991-y.

[34] G. Chen et al. "The multi-criteria minimum spanning tree problem based genetic algorithm". In: *Information Sciences* 177.22 (2007), pp. 5050–5063. ISSN: 0020-0255. DOI: 10.1016/j.ins.2007.06.005.

[35] M. Chen et al. "Cloud-based Wireless Network: Virtualized, Reconfigurable, Smart Wireless Network to Enable 5G Technologies". In: *Mobile Networks and Applications* 20.6 (Dec. 2015), pp. 704–712. ISSN: 1572-8153. DOI: 10.1007/s11036-015-0590-7.

[36] S.-T. Cheng. "Topological optimization of a reliable communication network". In: *IEEE Transactions on Reliability* 47.3 (Sept. 1998), pp. 225–233. ISSN: 1558-1721. DOI: 10.1109/24.740489.

[37] H. Chou, G. Premkumar, and C.-H. Chu. "Genetic algorithms for communications network design-an empirical study of the factors that influence performance". In: *IEEE Transactions on Evolutionary Computation* 5.3 (2001), pp. 236–249.

[38] Ciena. *OnePlanner Unified Design System*. https://media.ciena.com/documents/OnePlanner_Unified_Design_System_DS.pdf. Data Sheet. 2016. (Visited on 2025-03-06).

[39] Cisco. *Cisco Global Cloud Index: Forecast and Methodology, 2016–2021*. White Paper. June 2017.

[40] Cisco. *Cisco Visual Networking Index: Forecast and Methodology, 2016–2021*. White Paper. June 2017. URL: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.pdf (visited on 2018-11-13).

[41] *Cisco NCS 1002*. Data Sheet. Cisco. 2017. URL: https://www.cisco.com/c/en/us/products/collateral/optical-networking/network-convergence-system-1000-series/datasheet-c78-733699.pdf (visited on 2020-01-05).

[42] *Cisco NCS 2000 100-Gbps Coherent DWDM Trunk Card with CPAK Client Interface Data Sheet*. Data Sheet. Cisco. Dec. 2015. URL: https://www.cisco.com/c/en/us/products/collateral/optical-networking/network-convergence-system-2000-series/data_sheet_c78-729401.pdf (visited on 2019-04-26).

[43] *Cisco NCS 2000 200-Gbps Multirate DWDM Line Card Data Sheet*. Data Sheet. Cisco. 2018. URL: https://www.cisco.com/c/en/us/products/collateral/optical-networking/network-convergence-system-2000-series/datasheet-c78-733699.pdf (visited on 2020-01-05).

[44] *Cisco NCS 2000 Series Hardware Installation Guide*. Manual. Last Modified: 2019-10-04. Cisco. Oct. 2016. URL: https://www.cisco.com/c/en/us/td/docs/optical/hardware/ncs/guide/b-hig-ncsinstall.pdf (visited on 2020-01-03).

[45] *Cisco Network Convergence System 5500 Series: 1.2-Tbps IPoDWDM Modular Line Card Data Sheet*. Data Sheet. Cisco. 2018. URL: https://www.cisco.com/c/en/us/products/collateral/routers/network-convergence-system-5500-series/datasheet-c78-739372.pdf (visited on 2020-01-05).

[46] *Cisco Network Convergence System 6008 Single-Chassis System*. Data Sheet. Last Modified: 2018-01-22. Cisco. Aug. 2017. URL: https://www.cisco.com/c/en/us/products/collateral/routers/network-convergence-system-6000-series-routers/data_sheet_c78-728048.pdf (visited on 2019-03-06).

[47] *Cisco Network Convergence System NCS 5500 Modular Platform Architecture*. White Paper. Cisco. 2017. URL: https://www.cisco.com/c/dam/en/us/products/collateral/routers/network-convergence-system-5500-series/ncs5500-modular-platform-architecture-white-paper.pdf (visited on 2019-03-22).

[48] S. Clayman et al. "The dynamic placement of virtual network functions". In: *2014 IEEE Network Operations and Management Symposium (NOMS)*. May 2014, pp. 1–9. DOI: 10.1109/NOMS.2014.6838412.

[49] A. Colorni, M. Dorigo, V. Maniezzo, et al. “Distributed optimization by ant colonies”. In: *Proceedings of the first European conference on artificial life*. Vol. 142. Cambridge, MA. 1992, pp. 134–142.

[50] *Coriant® mTera ROADM*. Data Sheet. Infinera. 2018. URL: https://cdn.extranet.coriant.com/resources/Data-Sheets/DS_mTera_ROADM_74C0226.pdf (visited on 2019-03-20).

[51] Corning. *Corning SMF-28e+ Photonic Optical Fiber*. Data Sheet. 2010. URL: https://www.corning.com/media/worldwide/csm/documents/Corning%20SMF28e+%C2%AE%20Photonic%20Specialty%20Fiber.pdf (visited on 2020-05-09).

[52] D. Crawford. “Fiber optic cable dig-ups: Causes and cures”. In: *Proc. Network Reliability: A Report to the Nation–Compendium of Technical Papers, National Engineering Consortium* (1993).

[53] M. L. Daggitt, A. J. Gurney, and T. G. Griffin. “Asynchronous convergence of policy-rich distributed bellman-ford routing protocols”. In: *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. ACM. 2018, pp. 103–116.

[54] L. Davis. *Handbook of genetic algorithms*. VNR computer library. Van Nostrand Reinhold, 1991. ISBN: 9780442001735.

[55] L. Davis et al. “A genetic algorithm for survivable network design”. In: *Proceedings of the 5th International Conference on Genetic Algorithms*. Morgan Kaufmann Publishers Inc. 1993, pp. 408–415.

[56] K. A. De Jong. *Analysis of the behavior of a class of genetic adaptive systems*. Dissertation. University of Michigan, Aug. 1975.

[57] A. de Sousa, A. Tomaszewski, and M. Pióro. “Bin-packing based optimisation of EON Networks with S-BVTs”. In: *2016 International Conference on Optical Network Design and Modeling (ONDM)*. May 2016, pp. 1–6. DOI: 10.1109/ONDM.2016.7494082.

[58] A. Di Giglio and E. Vezzoni. *STRONGEST Deliverable D1.1. Project Presentation*. Deliverable. Version version 1.0 public, Final. Jan. 31, 2010.

[59] E. W. Dijkstra. “A note on two problems in connexion with graphs”. In: *Numerische Mathematik* 1.1 (Dec. 1959), pp. 269–271. ISSN: 0945-3245. DOI: 10.1007/BF01386390.

[60] D.-R. Din. “Genetic algorithm for virtual topology design on MLR WDM networks”. In: *Optical Switching and Networking* 18 (2015), pp. 20–34. ISSN: 1573-4277. DOI: 10.1016/j.osn.2015.03.003.

[61] H. Ding, B. Ramamurthy, and P. Yi. “CAPEX optimized routing for scheduled traffic in multi-layer optical networks”. In: *2013 19th IEEE Workshop on Local Metropolitan Area Networks (LANMAN)*. Apr. 2013, pp. 1–6. DOI: 10.1109/LANMAN.2013.6528272.

[62] J. Doucette and W. D. Grover. “Shared-risk logical span groups in span-restorable optical networks: Analysis and capacity planning model”. In: *Photonic Network Communications* 9.1 (2005), pp. 35–53.

[63] R. Doverspike and P. Magill. “13 - Commercial optical networks, overlay networks, and services”. In: *Optical Fiber Telecommunications V B (Fifth Edition)*. Ed. by I. P. Kaminow, T. Li, and A. E. Willner. Fifth Edition. Optics and Photonics. Burlington: Academic Press, 2008, pp. 511–560. DOI: 10.1016/B978-0-12-374172-1.00013-8.

[64] R. Durairajan et al. “InterTubes: A Study of the US Long-haul Fiber-optic Infrastructure”. In: *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*. SIGCOMM ’15. London, United Kingdom: ACM, 2015, pp. 565–578. ISBN: 978-1-4503-3542-3. DOI: 10.1145/2785956.2787499.

[65] R. J. Durán Barroso et al. “Minimisation of end-to-end delay in reconfigurable WDM networks using genetic algorithms”. In: *European Transactions on Telecommunications* 20.8 (2009), pp. 722–733. DOI: 10.1002/ett.1344.

[66] R. Eberhart and J. Kennedy. “A new optimizer using particle swarm theory”. In: *MHS’95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*. IEEE. 1995, pp. 39–43.

[67] ECI. *MUSE Network Planner*. https://www.ecitele.com/wp-content/uploads/2019/07/Muse-Network-Planner-Brochure-v2.pdf. Data Sheet. 2019. (Visited on 2019-10-08).

[68] A. E. Eiben and J. E. Smith. “Evolutionary Programming”. In: *Introduction to Evolutionary Computing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 89–99. ISBN: 978-3-662-05094-1. DOI: 10.1007/978-3-662-05094-1_5.

[69] A. Eiben and J. Smith. *Introduction to Evolutionary Computing (2nd Ed.)* 2nd. Berlin, Heidelberg: Springer-Verlag, 2016. ISBN: 978–3-662-44873-1. DOI: 10.1007/978-3-662-44874-8.

[70] *EKINOPS PM 200FRS02 Single slot FlexRate pluggable transponder/muxponder*. Data Sheet. Ekinops. 2018. URL: https://www.ekinops.com/products/flexrate-modules/transport-modules/item/621-pm-200frs02 (visited on 2019-03-27).

[71] *EKINOPS PM 400FRS04 Pluggable 400G Flexrate Muxponder*. Data Sheet. Ekinops. 2018. URL: https://www.ekinops.com/products/flexrate-modules/transport-modules/item/836-pm-400frs04-sf (visited on 2019-03-27).

[72] T. Enderle and U. Bauknecht. “Modeling Dynamic Traffic Demand Behavior in Telecommunication Networks”. In: *Proceedings of the 19th ITG-Symposium in Photonic Networks 2018*. 2018, pp. 18–25.

[73] T. Enderle et al. “Reconfigurable Resource Allocation in Dynamic Transport Networks”. In: *Proceedings of the 20th ITG-Symposium in Photonic Networks 2019*. 2019.

[74] M. Ericsson, M. Resende, and P. Pardalos. “A Genetic Algorithm for the Weight Setting Problem in OSPF Routing”. In: *Journal of Combinatorial Optimization* 6.3 (Sept. 2002), pp. 299–333. ISSN: 1573-2886. DOI: 10.1023/A:1014852026591.

[75] S. Even, A. Itai, and A. Shamir. “On the complexity of time table and multi-commodity flow problems”. In: *16th Annual Symposium on Foundations of Computer Science (sfcs 1975)*. Oct. 1975, pp. 184–193. DOI: 10.1109/SFCS.1975.21.

[76] A. Farrel and I. Bryskin. *GMPLS: Architecture and Applications*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2006. ISBN: 978-0-12-088422-3. DOI: 10.1016/B978-0-12-088422-3.X5000-X.

[77] F. Feller. "A Reconfiguration Method for Energy-Efficient Operation of Multi-Layer Core Networks - Communication Networks and Computer Engineering Report No. 113". PhD thesis. Universität Stuttgart, 2016.

[78] F. Feller. "Evaluation of a Centralized Method for One-Step Multi-Layer Network Reconfiguration". In: *Proceedings of the 24th Tyrrhenian International Workshop on Digital Communications (TIWDC 2013)*. 2013.

[79] F. Feller. "Evaluation of Centralized Solution Methods for the Dynamic Optical Bypassing Problem". In: *Proceedings of the 17th Conference on Optical Network Design & Modeling (ONDM 2013)*. Apr. 2013.

[80] C. Ferreira. "Gene expression programming in problem solving". In: *Soft computing and industry*. Springer, 2002, pp. 635–653.

[81] C. Feuersänger. *Manual for Package pgfplots. 2D/3D Plots in LaTeX*. Version 1.16. Mar. 28, 2018. URL: http://sourceforge.net/projects/pgfplot (visited on 2019-05-22).

[82] A. Fischer et al. "Virtual Network Embedding: A Survey". In: *IEEE Communications Surveys Tutorials* 15.4 (Apr. 2013), pp. 1888–1906. ISSN: 1553-877X. DOI: 10.1109/SURV.2013.013013.00155.

[83] K.-T. Foerster, M. Ghobadi, and S. Schmid. "Characterizing the Algorithmic Complexity of Reconfigurable Data Center Architectures". In: *Proceedings of the 2018 Symposium on Architectures for Networking and Communications Systems*. ANCS '18. Ithaca, New York: Association for Computing Machinery, 2018, pp. 89–96. ISBN: 9781450359023. DOI: 10.1145/3230718.3230722.

[84] L. R. Ford and D. R. Fulkerson. "Maximal Flow Through a Network". In: *Canadian Journal of Mathematics* 8 (1956), pp. 399–404. DOI: 10.4153/CJM-1956-045-5.

[85] X. Foukas et al. "Network Slicing in 5G: Survey and Challenges". In: *IEEE Communications Magazine* 55.5 (2017), pp. 94–100. ISSN: 0163-6804. DOI: 10.1109/MCOM.2017.1600951.

[86] F. Francois et al. "Green IGP link weights for energy-efficiency and load-balancing in IP backbone networks". In: *2013 IFIP Networking Conference*. May 2013, pp. 1–9.

[87] L. C. Freeman. "A Set of Measures of Centrality Based on Betweenness". In: *Sociometry* 40.1 (1977), pp. 35–41. ISSN: 00380431. DOI: 10.2307/3033543.

[88] *FSP 3000*. Data Sheet. Adva. Mar. 2019. URL: https://www.adva.com/-/media/adva-main-site/resources/data-sheets/pdfs/fsp-3000.ashx?la=en (visited on 2020-01-03).

[89] *FSP 3000 QuadFlex*. Data Sheet. Adva. June 2019. URL: https://www.adva.com/-/media/adva-main-site/resources/data-sheets/pdfs/fsp-3000-quad-flex-data-sheet.ashx?la=de-de (visited on 2020-01-05).

[90] ITU-T. *Transmission characteristics of optical components and subsystems*. Rec. G.671. ITU-T, Feb. 2012.

[91] ITU-T. *Characteristics of multi-degree reconfigurable optical add/drop multiplexers*. Rec. G.672. ITU-T, Nov. 2018.

[92] ITU-T. *Spectral grids for WDM applications: DWDM frequency grid*. Rec. G.694.1. ITU-T, Feb. 2012.

[93] ITU-T. *Spectral grids for WDM applications: CWDM wavelength grid*. Rec. G.694.2. ITU-T, Dec. 2003.

[94] ITU-T. *Optical interfaces for coarse wavelength division multiplexing applications*. Rec. G.695. ITU-T, July 2018.

[95] ITU-T. *Network node interface for the synchronous digital hierarchy (SDH)*. Rec. G.707/Y.1322. ITU-T, Dec. 2003.

[96] ITU-T. *Interfaces for the Optical Transport Network (OTN)*. Rec. G.709/Y.1331. ITU-T, Mar. 2003.

[97] ITU-T. *Unified functional architecture of transport networks*. Rec. G.800. ITU-T, Apr. 2016.

[98] ITU-T. *Architecture for the automatically switched optical network (ASON)*. Rec. G.8080/Y.1304. ITU-T, Nov. 2001.

[99] ITU-T. *Architecture of the Multi-Protocol Label Switching transport profile layer network*. Rec. G.8110.1/Y.1370.1. ITU-T, Dec. 2011.

[100] J. Galán-Jiménez. "Minimization of energy consumption in IP/SDN hybrid networks using genetic algorithms". In: *2017 Sustainable Internet and ICT for Sustainability (SustainIT)*. Dec. 2017, pp. 1–5. DOI: 10.23919/SustainIT.2017.8379802.

[101] S. Gao et al. "An Optimization Model for Placement of Wavelength Converters to Minimize Blocking Probability in WDM Networks". In: *J. Lightwave Technol.* 21.3 (Mar. 2003), p. 684.

[102] M. Gen, K. Ida, and J. Kim. "A spanning tree-based genetic algorithm for bicriteria topological network design". In: *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*. May 1998, pp. 15–20. DOI: 10.1109/ICEC.1998.699068.

[103] S. Ghose et al. "Multihop Virtual Topology Design in WDM Optical Networks for Self-Similar Traffic". In: *Photonic Network Communications* 10.2 (Sept. 2005), pp. 199–214. ISSN: 1572-8188. DOI: 10.1007/s11107-005-2484-2.

[104] G. M. Gibson. "Simultaneous optimisation of both scalar parameters and agent reaction strategies using genetic algorithms". In: *Proceedings of IEEE/IAS International Conference on Industrial Automation and Control*. Jan. 1995, pp. 405–410. DOI: 10.1109/IACC.1995.465806.

[105] V. Gkamas, K. Christodoulopoulos, and E. Varvarigos. "A Joint Multi-Layer Planning Algorithm for IP Over Flexible Optical Networks". In: *Journal of Lightwave Technology* 33.14 (July 2015), pp. 2965–2977. DOI: 10.1109/JLT.2015.2424920.

[106] A. V. Goldberg and R. E. Tarjan. "A New Approach to the Maximum-Flow Problem". In: *J. ACM* 35.4 (Oct. 1988), pp. 921–940. ISSN: 0004-5411. DOI: 10.1145/48014.61051.

[107] D. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Artificial Intelligence. Addison-Wesley Publishing Company, 1989. ISBN: 978-0201157673.

[108] L. Gong et al. "A Two-Population Based Evolutionary Approach for Optimizing Routing, Modulation and Spectrum Assignments (RMSA) in O-OFDM Networks". In: *IEEE Communications Letters* 16.9 (Sept. 2012), pp. 1520–1523. ISSN: 2373-7891. DOI: 10.1109/LCOMM.2012.070512.120740.

[109] J. Gottlieb et al. "Prüfer Numbers: A Poor Representation of Spanning Trees for Evolutionary Search". In: *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*. GECCO'01. San Francisco, California: Morgan Kaufmann Publishers Inc., 2001, pp. 343–350. ISBN: 1-55860-774-9.

[110] V. Granville, M. Krivánek, and J.-P. Rasson. "Simulated annealing: A proof of convergence". In: *IEEE transactions on pattern analysis and machine intelligence* 16.6 (1994), pp. 652–656.

[111] A. Graves et al. "Hybrid computing using a neural network with dynamic external memory". In: *Nature* 538.7626 (2016), pp. 471–476. DOI: 10.1038/nature20101.

[112] H. J. Greenberg. *Klee-Minty Polytope Shows Exponential Time Complexity of Simplex Method*. 1997.

[113] I. Griva, S. G. Nash, and A. Sofer. *Linear and Nonlinear Optimization: Second Edition*. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics, 2009. ISBN: 9780898716610. DOI: 10.1137/1.9780898717730.

[114] M. Grötschel, C. Raack, and A. Werner. "Towards optimizing the deployment of optical access networks". In: *EURO Journal on Computational Optimization* 2.1 (June 2014), pp. 17–53. ISSN: 2192-4414. DOI: 10.1007/s13675-013-0016-x.

[115] W. D. Grover. *Mesh-based Survivable Transport Networks: Options and Strategies for Optical, MPLS, SONET, and ATM Networking*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2004. ISBN: 0-13-494576-X.

[116] A. Gumaste and S. Akhtar. "Evolution of packet-optical integration in backbone and metropolitan high-speed networks: a standards perspective". In: *IEEE Communications Magazine* 51.11 (Nov. 2013), pp. 105–111. ISSN: 0163-6804. DOI: 10.1109/MCOM.2013.6658660.

[117] D. T. Hai. "Multi-objective genetic algorithm for solving routing and spectrum assignment problem". In: *2017 Seventh International Conference on Information Science and Technology (ICIST)*. Apr. 2017, pp. 177–180. DOI: 10.1109/ICIST.2017.7926753.

[118] L. Han and Y. Wang. "A Novel Genetic Algorithm for Multi-criteria Minimum Spanning Tree Problem". In: *Computational Intelligence and Security*. Ed. by Y. Hao et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 297–302. ISBN: 978-3-540-31599-5.

[119] A. Hassan and C. Phillips. "Particle swarm optimization-based DRWA for wavelength continuous WDM optical networks using a novel fitness function". In: *Artificial Intelligence Review* 29.3 (Oct. 2009), p. 305. ISSN: 1573-7462. DOI: 10.1007/s10462-009-9142-5.

[120] W. K. Hastings. “Monte Carlo sampling methods using Markov chains and their applications”. In: *Biometrika* 57.1 (Apr. 1970), pp. 97–109. ISSN: 0006-3444. DOI: 10.1093/biomet/57.1.97.

[121] R. Hemmecke et al. “Nonlinear Integer Programming”. In: *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*. Ed. by M. Jünger et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 561–618. ISBN: 978-3-540-68279-0. DOI: 10.1007/978-3-540-68279-0_15.

[122] J. Hewitt, A. Soper, and S. McKenzie. “Charley: a genetic algorithm for the design of mesh networks”. In: *First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*. Sept. 1995, pp. 118–122. DOI: 10.1049/cp:19951035.

[123] K. Hinton et al. “Power consumption and energy efficiency in the internet”. In: *IEEE Network* 25.2 (Mar. 2011), pp. 6–12. ISSN: 0890-8044. DOI: 10.1109/MNET.2011.5730522.

[124] C. Hogendorn. “Excessive(?) entry of national telecom networks, 1990–2001”. In: *Telecommunications Policy* 35.11 (2011), pp. 920–932. ISSN: 0308-5961. DOI: 10.1016/j.telpol.2011.09.003.

[125] J. H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Cambridge, MA, USA: MIT Press, 1992. ISBN: 0262082136.

[126] H. Höller and S. Voß. “A heuristic approach for combined equipment-planning and routing in multi-layer SDH/WDM networks”. In: *European Journal of Operational Research* 171.3 (2006). Feature Cluster: Heuristic and Stochastic Methods in Optimization Feature Cluster: New Opportunities for Operations Research, pp. 787–796. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2004.09.006.

[127] T. Huang et al. “A Survey on Large-Scale Software Defined Networking (SDN) Testbeds: Approaches and Challenges”. In: *IEEE Communications Surveys Tutorials* 19.2 (Oct. 2017), pp. 891–917. ISSN: 1553-877X. DOI: 10.1109/COMST.2016.2630047.

[128] F. Idzikowski et al. “A Survey on Energy-Aware Design and Operation of Core Networks”. In: *IEEE Communications Surveys Tutorials* 18.2 (Oct. 2016), pp. 1453–1499. ISSN: 1553-877X. DOI: 10.1109/COMST.2015.2507789.

[129] F. Idzikowski et al. “Saving energy in IP-over-WDM networks by switching off line cards in low-demand scenarios”. In: *2010 14th Conference on Optical Network Design and Modeling (ONDM)*. Feb. 2010, pp. 1–6. DOI: 10.1109/ONDM.2010.5431569.

[130] *IKR Simulation Library (IKR SimLib)*. Version 4.0.0. Institute of Communication Networks and Computer Engineering (IKR), University of Stuttgart. June 24, 2019. URL: http://www.ikr.uni-stuttgart.de/IKRSimLib/ (visited on 2019-06-24).

[131] P. Iovanna et al. “Main challenges on WAN due to NFV and SDN: multi-layer and multi-domain network virtualization and routing”. In: *2015 International Conference on Optical Network Design and Modeling (ONDM)*. May 2015, pp. 74–79. DOI: 10.1109/ONDM.2015.7127277.

[132] P. Iovanna et al. “Multilayer control for packet-optical networks [invited]”. In: *IEEE/OSA Journal of Optical Communications and Networking* 5.10 (Oct. 2013), A86–A99. ISSN: 1943-0620. DOI: 10.1364/JOCN.5.000A86.

[133] V. Joseph and B. Chapman. “Chapter 6 - QoS Service Assurance”. In: *Deploying QoS for Cisco IP and Next Generation Networks*. Ed. by V. Joseph and B. Chapman. Boston: Morgan Kaufmann, 2009, pp. 187–197. ISBN: 978-0-12-374461-6. DOI: 10.1016/B978-0-12-374461-6.00006-9.

[134] P. Kampstra, R. Van der Mei, and A. Eiben. *Evolutionary computing in telecommunication network design: A survey*. Tech. rep. Vrije Universiteit, Faculty of Exact Sciences and CWI, Advanced Communication Networks. Amsterdam, Netherlands, 2006.

[135] M. Khaddam, L. Paraschis, and J. Finkelstein. “SDN multi-layer transport benefits, deployment opportunities, and requirements”. In: *2015 Optical Fiber Communications Conference and Exhibition (OFC)*. Mar. 2015, pp. 1–3. DOI: 10.1364/OFC.2015.Th1A.1.

[136] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. “Optimization by simulated annealing”. In: *science* 220.4598 (1983), pp. 671–680.

[137] V. Klee. “Combinatorial optimization: what is the state of the art”. In: *Mathematics of Operations Research* 5.1 (1980), pp. 1–26.

[138] A. Klekamp. “Multi-layer network optimization: Benefits of elastic optical networks”. In: *2012 14th International Conference on Transparent Optical Networks (ICTON)*. IEEE. 2012, pp. 1–5.

[139] J. D. Knowles and D. W. Corne. “A comparison of encodings and algorithms for multiobjective minimum spanning tree problems”. In: *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*. Vol. 1. May 2001, 544–551 vol. 1. DOI: 10.1109/CEC.2001.934439.

[140] K.-T. Ko et al. “Using genetic algorithms to design mesh networks”. In: *Computer* 30.8 (Aug. 1997), pp. 56–61. ISSN: 1558-0814. DOI: 10.1109/2.607086.

[141] A. Konak* and A. E. Smith. “Capacitated network design considering survivability: an evolutionary approach”. In: *Engineering Optimization* 36.2 (2004), pp. 189–205. DOI: 10.1080/03052150310001633223.

[142] A. M. C. A. Koster and X. Muñoz. “Graphs and Algorithms in Communication Networks on Seven League Boots”. In: *Graphs and Algorithms in Communication Networks: Studies in Broadband, Optical, Wireless and Ad Hoc Networks*. Ed. by A. Koster and X. Muñoz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 1–59. ISBN: 978-3-642-02250-0. DOI: 10.1007/978-3-642-02250-0_1.

[143] J. B. Kruskal. “On the shortest spanning subtree of a graph and the traveling salesman problem”. In: *Proceedings of the American Mathematical society* 7.1 (1956), pp. 48–50.

[144] M. Kucharzak, L. Koszalka, and A. Kasprzak. “Shortest Multilayered Path for Dimensioning in Two-Layer Connection-Oriented Networks”. In: *2009 10th International Symposium on Pervasive Systems, Algorithms, and Networks*. Dec. 2009, pp. 332–337. DOI: 10.1109/I-SPAN.2009.79.

[145] M. M. Lankhorst. *Iterated function systems optimization with genetic algorithms*. type. University of Groningen, Department of Mathematics and Computing Science, Jan. 1995.

[146] E. L. Lawler. *Combinatorial optimization: networks and matroids*. Mineola, New York: Dover Publications, Inc., 2001. ISBN: 9780486414539.

[147] W. Liang, G. Havas, and X. Shen. "Improved lightpath (wavelength) routing in large WDM networks". In: *Proceedings. 18th International Conference on Distributed Computing Systems (Cat. No.98CB36183)*. May 1998, pp. 516–523. DOI: 10.1109/ICDCS.1998.679797.

[148] L. Liu, T. Tsuritani, and I. Morita. "Experimental demonstration of OpenFlow/GMPLS interworking control plane for IP/DWDM multi-layer optical networks". In: *2012 14th International Conference on Transparent Optical Networks (ICTON)*. IEEE. 2012, pp. 1–4.

[149] P. Lothberg. "Optical networking in DTAG's TeraStream project". In: *2016 Optical Fiber Communications Conference and Exhibition (OFC)*. Mar. 2016, pp. 1–3.

[150] H. R. Lourenço, O. C. Martin, and T. Stützle. "Iterated Local Search". In: *Handbook of Metaheuristics*. Ed. by F. Glover and G. A. Kochenberger. Boston, MA: Springer US, 2003, pp. 320–353. ISBN: 978-0-306-48056-0. DOI: 10.1007/0-306-48056-5_11.

[151] L. J. Lu and M. Zhang. "Edge Betweenness Centrality". In: *Encyclopedia of Systems Biology*. Ed. by W. Dubitzky et al. New York, NY: Springer New York, 2013, pp. 647–648. ISBN: 978-1-4419-9863-7. DOI: 10.1007/978-1-4419-9863-7_874.

[152] T. Lu and J. Zhu. "Genetic Algorithm for Energy-Efficient QoS Multicast Routing". In: *IEEE Communications Letters* 17.1 (Jan. 2013), pp. 31–34. ISSN: 2373-7891. DOI: 10.1109/LCOMM.2012.112012.121467.

[153] ITU-T. *Generic network information model*. Rec. M.3100. ITU-T, Apr. 2005.

[154] G. Maier et al. "Optical Network Survivability: Protection Techniques in the WDM Layer". In: *Photonic Network Communications* 4.3 (July 2002), pp. 251–269. ISSN: 1572-8188. DOI: 10.1023/A:1016047527226.

[155] M. Mao and M. Humphrey. "A performance study on the vm startup time in the cloud". In: *2012 IEEE Fifth International Conference on Cloud Computing*. IEEE. 2012, pp. 423–430.

[156] D. M. Marom et al. "Survey of Photonic Switching Architectures and Technologies in Support of Spatially and Spectrally Flexible Optical Networking [Invited]". In: *J. Opt. Commun. Netw.* 9.1 (Jan. 2017), pp. 1–26. DOI: 10.1364/JOCN.9.000001.

[157] R. Martínez et al. "Experimental evaluation of delay-sensitive traffic routing in multi-layer (packet-optical) aggregation networks for fixed mobile convergence". In: *39th European Conference and Exhibition on Optical Communication (ECOC 2013)*. Sept. 2013, pp. 1–3. DOI: 10.1049/cp.2013.1457.

[158] E. Q. V. Martins. "On a multicriteria shortest path problem". In: *European Journal of Operational Research* 16.2 (1984), pp. 236–245. ISSN: 0377-2217. DOI: 10.1016/0377-2217(84)90077-8.

[159] R. Martı, M. G. Resende, and C. C. Ribeiro. “Multi-start methods for combinatorial optimization”. In: *European Journal of Operational Research* 226.1 (2013), pp. 1–8.

[160] M. Matsumoto and T. Nishimura. “Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator”. In: *ACM Trans. Model. Comput. Simul.* 8.1 (Jan. 1998), pp. 3–30. ISSN: 1049-3301. DOI: 10.1145/272991.272995.

[161] N. Metropolis et al. “Equation of state calculations by fast computing machines”. In: *The journal of chemical physics* 21.6 (1953), pp. 1087–1092.

[162] D. Michail et al. “JGraphT–A Java library for graph data structures and algorithms”. In: *arXiv preprint arXiv:1904.08355* (2019).

[163] Z. Michalewicz. “Step towards optimal topology of communication networks”. In: *Data Structures and Target Classification*. Vol. 1470. International Society for Optics and Photonics. 1991, pp. 112–122.

[164] I. de Miguel et al. “Genetic Algorithm for Joint Routing and Dimensioning of Dynamic WDM Networks”. In: *J. Opt. Commun. Netw.* 1.7 (Dec. 2009), pp. 608–621. DOI: 10.1364/JOCN.1.000608.

[165] R. Morais et al. “Genetic Algorithm for the Topological Design of Survivable Optical Transport Networks”. In: *IEEE/OSA Journal of Optical Communications and Networking* 3.1 (Jan. 2011), pp. 17–26. ISSN: 1943-0639. DOI: 10.1364/JOCN.3.000017.

[166] R. M. Morais, J. Pedro, and A. N. Pinto. “Planning and dimensioning of multilayer optical transport networks”. In: *2015 17th International Conference on Transparent Optical Networks (ICTON)*. July 2015, pp. 1–5. DOI: 10.1109/ICTON.2015.7193723.

[167] B. Mukherjee. *Optical WDM Networks*. New York, NY, USA: Springer Science+Business Media, Inc., 2004. ISBN: 0-12-557189-5. DOI: 10.1007/0-387-29188-1.

[168] M. Munetomo, Y. Takai, and Y. Sato. “A migration scheme for the genetic adaptive routing algorithm”. In: *SMC’98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.98CH36218)*. Vol. 3. Oct. 1998, 2774–2779 vol.3. DOI: 10.1109/ICSMC.1998.725081.

[169] *MX2000 Universal Routing Platforms*. Data Sheet. Juniper. Feb. 2019. URL: https://www.juniper.net/elqNow/elqRedir.htm?ref=https://www.juniper.net/assets/us/en/local/pdf/datasheets/1000417-en.pdf (visited on 2019-03-06).

[170] J. C. Nash. “The (Dantzig) simplex method for linear programming”. In: *Computing in Science & Engineering* 2.1 (2000), pp. 29–31. DOI: 10.1109/5992.814654.

[171] I. Nath, M. Chatterjee, and U. Bhattacharya. “A survey on regenerator Placement Problem in translucent optical network”. In: *2014 International Conference on Circuits, Systems, Communication and Information Technology Applications (CSCITA)*. Apr. 2014, pp. 408–413. DOI: 10.1109/CSCITA.2014.6839295.

[172] B. Naveh and Contributors. *JGraphT. a Java library of graph theory data structures and algorithms*. Version 1.3.0. June 24, 2019. URL: https://jgrapht.org/ (visited on 2019-06-24).

[173] S. Nesmachnow, H. Cancela, and E. Alba. “Nature-inspired informatics for telecommunication network design”. In: *Networking and Telecommunications: Concepts, Methodologies, Tools, and Applications*. IGI Global, 2010, pp. 302–350.

[174] *NetEngine9000 Series Converged Core Router*. Data Sheet. Huawei. 2017. URL: http://carrier.huawei.com/~/media/CNBG/Downloads/Product/Fixed%20Network/carrierip-router/Huawei-NE9000-Product-Brochure-2-25.pdf (visited on 2019-03-06).

[175] *Network Optimization in the 600G Era*. White Paper. Acacia. 2018. URL: https://acacia-inc.com/wp-content/uploads/2018/12/Network-Optimization-in-the-600G-Era-WP1218.pdf (visited on 2019-04-01).

[176] P. Ngatchou, A. Zarei, and A. El-Sharkawi. “Pareto Multi Objective Optimization”. In: *Proceedings of the 13th International Conference on, Intelligent Systems Application to Power Systems*. Nov. 2005, pp. 84–91. DOI: 10.1109/ISAP.2005.1599245.

[177] *Nokia 1830 PSS 500G Muxponder*. Data Sheet. Nokia. 2017. URL: https://onestore.nokia.com/asset/194076 (visited on 2019-03-27).

[178] *Nokia 7950 Extensible Routing System*. Data Sheet. Release 16. Nokia. Feb. 2019. URL: https://resources.nokia.com/asset/205346 (visited on 2019-03-06).

[179] A. Nordrum. “Facebook pushes networking tech: The company’s Terragraph technology will soon be available in commercial gear”. In: *IEEE Spectrum* 56.4 (2019), pp. 8–9.

[180] Y. Nourani and B. Andresen. “A comparison of simulated annealing cooling strategies”. In: *Journal of Physics A: Mathematical and General* 31.41 (1998), p. 8373.

[181] ONF. *OpenFlow Switch Specification*. Tech. rep. ONF TS-025. ONF, Mar. 2015.

[182] ONF. *Optical Transport Protocol Extensions*. Tech. rep. ONF TS-022. ONF, Mar. 2015.

[183] S. Orlowski et al. “SNDlib 1.0—Survivable Network Design Library”. In: *Networks* 55.3 (May 2010), pp. 276–286. ISSN: 1097-0037. DOI: 10.1002/net.20371.

[184] S. Pachnicke. *Fiber-Optic Transmission Networks - Efficient Design and Dynamic Operation*. Springer Berlin Heidelberg, 2012. ISBN: 978-3-642-21055-6. DOI: 10.1007/978-3-642-21055-6.

[185] C. C. Palmer and A. Kershenbaum. “An approach to a problem in network design using genetic algorithms”. In: *Networks* 26.3 (1995), pp. 151–163.

[186] I. Papanagiotou and M. Howarth. “Delay-based quality of service through intra-domain differentiated routing with optimised link weight setting”. In: *The IEEE symposium on Computers and Communications*. June 2010, pp. 622–627. DOI: 10.1109/ISCC.2010.5546518.

[187] T. Paulden and D. K. Smith. “Recent Advances in the Study of the Dandelion Code, Happy Code, and Blob Code Spanning Tree Representations”. In: *2006 IEEE International Conference on Evolutionary Computation*. July 2006, pp. 2111–2118. DOI: 10.1109/CEC.2006.1688567.

[188] O. Pedrola et al. “A GRASP with path-relinking heuristic for the survivable IP/MPLS-over-WSON multi-layer network optimization problem”. In: *Computers & Operations Research* 40.12 (2013), pp. 3174–3187. ISSN: 0305-0548. DOI: 10.1016/j.cor.2011.10.026.

[189] E. Pellerin, L. Pigeon, and S. Delisle. “Self-adaptive parameters in genetic algorithms”. In: *Data Mining and Knowledge Discovery: Theory, Tools, and Technology VI*. Ed. by B. V. Dasarathy. Vol. 5433. International Society for Optics and Photonics. SPIE, 2004, pp. 53–64. DOI: 10.1117/12.542156.

[190] L. Peterson et al. “Central office re-architected as a data center”. In: *IEEE Communications Magazine* 54.10 (Oct. 2016), pp. 96–101. ISSN: 0163-6804. DOI: 10.1109/MCOM.2016.7588276.

[191] M. Pickavet and P. Demeester. “A zoom-in approach to design SDH mesh restor-able networks”. In: *Heuristic Approaches for Telecommunications Network Management, Planning and Expansion*. Springer, 2000, pp. 107–130.

[192] S. Pierre and G. Legault. “A genetic algorithm for designing distributed computer network topologies”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 28.2 (Apr. 1998), pp. 249–258. ISSN: 1941-0492. DOI: 10.1109/3477.662766.

[193] P. Piggott and F. Suraweera. “Encoding graphs for genetic algorithms: An investigation using the minimum spanning tree problem”. In: *Progress in Evolutionary Computation*. Ed. by X. Yao. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 305–314. ISBN: 978-3-540-49528-4.

[194] M. Pióro and D. Medhi. *Routing, Flow, and Capacity Design in Communication and Computer Networks*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2006. ISBN: 978-0-12-088422-3. DOI: 10.1016/B978-0-12-557189-0.X5000-8.

[195] *PL-1000RO*. Data Sheet. PacketLight Networks. 2019. URL: https://www.packetlight.com/images/Data-Sheets/PL-1000RO.pdf (visited on 2019-03-20).

[196] R. Poli and J. Koza. “Genetic Programming”. In: *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*. Ed. by E. K. Burke and G. Kendall. Boston, MA: Springer US, 2014, pp. 143–185. ISBN: 978-1-4614-6940-7. DOI: 10.1007/978-1-4614-6940-7_6.

[197] K. V. Price. “Differential Evolution”. In: *Handbook of Optimization: From Classical to Modern Approach*. Ed. by I. Zelinka, V. Snášel, and A. Abraham. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 187–214. ISBN: 978-3-642-30504-7. DOI: 10.1007/978-3-642-30504-7_8.

[198] R. C. Prim. “Shortest connection networks and some generalizations”. In: *The Bell System Technical Journal* 36.6 (Nov. 1957), pp. 1389–1401. ISSN: 0005-8580. DOI: 10.1002/j.1538-7305.1957.tb01515.x.

[199] *Project SASER*. 2012. URL: https://www.celticnext.eu/project-saser/ (visited on 2019-04-27).

[200] *PTX1000, PTX10001, PTX10002, and PTX10003 Fixed-Configuration Packet Transport Routers*. Data Sheet. Juniper. Jan. 2019. URL: https://www.juniper.net/assets/de/de/local/pdf/datasheets/1000538-en.pdf (visited on 2019-03-22).

[201] V. Puglia and O. Zadedyurina. “Optical transport networks: From all-optical to digital”. In: *2009 ITU-T Kaleidoscope: Innovations for Digital Inclusions*. Aug. 2009, pp. 1–8.

[202] *QFX10000 Coherent DWDM Line Card*. Data Sheet. Juniper. July 2017. URL: https://www.juniper.net/assets/us/en/local/pdf/datasheets/1000615-en.pdf (visited on 2020-01-05).

[203] G. R. Raidl and B. A. Julstrom. "Edge sets: an effective evolutionary coding of spanning trees". In: *IEEE Transactions on evolutionary computation* 7.3 (2003), pp. 225–239.

[204] G. R. Raidl and B. A. Julstrom. "A Weighted Coding in a Genetic Algorithm for the Degree-constrained Minimum Spanning Tree Problem". In: *Proceedings of the 2000 ACM Symposium on Applied Computing - Volume 1*. SAC '00. Como, Italy: ACM, 2000, pp. 440–445. ISBN: 1-58113-240-9. DOI: 10.1145/335603.335888.

[205] R. W. Callon. *Use of OSI IS-IS for routing in TCP/IP and dual environments*. RFC 1195. IETF, Dec. 1990.

[206] G. Malkin and R. Minnear. *RIPng for IPv6*. RFC 2080. IETF, Jan. 1997.

[207] J. Moy. *OSPF Version 2*. RFC 2328. IETF, Apr. 1998.

[208] E. Rosen, A. Viswanathan, and R. Callon. *Multiprotocol Label Switching Architecture*. RFC 3031. IETF, Jan. 2001.

[209] D. Awduche et al. *RSVP-TE: Extensions to RSVP for LSP Tunnels*. RFC 3209. IETF, Dec. 2001.

[210] D. Katz, K. Kompella, and D. Yeung. *Traffic Engineering (TE) Extensions to OSPF Version 2*. RFC 3630. IETF, Sept. 2003.

[211] E. Mannie (Editor). *Generalized Multi-Protocol Label Switching (GMPLS) Architecture*. RFC 3945. IETF, Oct. 2004.

[212] S. Bryant (Editor) and P. Pate (Editor). *Pseudo Wire Emulation Edge-to-Edge (PWE3) Architecture*. RFC 3985. IETF, Mar. 2005.

[213] D. Papadimitriou (Editor). *Generalized Multi-Protocol Label Switching (GMPLS) Signaling Extensions for G.709 Optical Transport Networks Control*. RFC 4328. IETF, Jan. 2006.

[214] A. Farrel, J.-P. Vasseur, and J. Ash. *A Path Computation Element (PCE)-Based Architecture*. RFC 4655. IETF, Aug. 2006.

[215] T. Li and H. Smit. *IS-IS Extensions for Traffic Engineering*. RFC 5305. IETF, Oct. 2008.

[216] J. Vasseur (Editor) and J. Le Roux (Editor). *Path Computation Element (PCE) Communication Protocol (PCEP)*. RFC 5440. IETF, Mar. 2009.

[217] M. Bocci (Editor) et al. *A Framework for MPLS in Transport Networks*. RFC 5921. IETF, July 2010.

[218] M. Bjorklund (Editor). *YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)*. RFC 6020. IETF, Oct. 2010.

[219] R. Enns (Editor) et al. *Network Configuration Protocol (NETCONF)*. RFC 6241. IETF, June 2011.

[220] L. Andersson (Editor) et al. *MPLS Transport Profile (MPLS-TP) Control Plane Framework*. RFC 6373. IETF, Sept. 2011.

[221] F. Zhang (Editor) et al. *GMPLS Signaling Extensions for Control of Evolving G.709 Optical Transport Networks*. RFC 7139. IETF, Mar. 2014.

[222] E. Haleplidis (Editor) et al. *Software-Defined Networking (SDN): Layers and Architecture Terminology*. RFC 7426. IETF, Jan. 2015.

[223] Z. Ali et al. *GMPLS Signaling Extensions for Control of Evolving G.709 Optical Transport Networks*. RFC 7892. IETF, May 2016.

[224] W. Liu et al. *Policy-Based Management Framework for the Simplified Use of Policy Abstractions (SUPA)*. RFC 8328. IETF, Mar. 2018.

[225] H. Long et al. *OSPF Traffic Engineering (OSPF-TE) Link Availability Extension for Links with Variable Discrete Bandwidth*. RFC 8330. IETF, Feb. 2018.

[226] E. Riccardi et al. "An Operator view on the Introduction of White Boxes into Optical Networks". In: *J. Lightwave Technol.* 36.15 (Aug. 2018), pp. 3062–3072.

[227] A. Riedl. "A hybrid genetic algorithm for routing optimization in IP networks utilizing bandwidth and delay metrics". In: *IEEE Workshop on IP Operations and Management*. Oct. 2002, pp. 166–170. DOI: 10.1109/IPOM.2002.1045774.

[228] C. Risso, S. Nesmachnow, and F. Robledo. "A Parallel Evolutionary Algorithm for Multilayered Robust Network Design". In: *2012 Seventh International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*. Nov. 2012, pp. 291–296. DOI: 10.1109/3PGCIC.2012.4.

[229] C. Risso. "Using GRASP and GA to design resilient and cost-effective IP/MPLS networks". PhD thesis. Universidad de la República de Uruguay, May 2014.

[230] C. Risso, S. Nesmachnow, and F. Robledo. "Metaheuristic approaches for IP/MPLS network design". In: *International Transactions in Operational Research* 25.2 (2018), pp. 599–625. DOI: 10.1111/itor.12418.

[231] G. Rizzelli et al. "Assessing the Scalability of Next-Generation Wavelength Switched Optical Networks". In: *Journal of Lightwave Technology* 32.12 (June 2014), pp. 2263–2270. ISSN: 0733-8724. DOI: 10.1109/JLT.2014.2315759.

[232] G. Rizzelli et al. "Impairment-aware design of translucent DWDM networks based on the k-path connectivity graph". In: *IEEE/OSA Journal of Optical Communications and Networking* 4.5 (May 2012), pp. 356–365. ISSN: 1943-0620. DOI: 10.1364/JOCN.4.000356.

[233] F. A. Rodrigues. "Network Centrality: An Introduction". In: *A Mathematical Modeling Approach from Nonlinear Dynamics to Complex Systems*. Ed. by E. E. N. Macau. Cham: Springer International Publishing, 2019, pp. 177–196. ISBN: 978-3-319-78512-7. DOI: 10.1007/978-3-319-78512-7_10.

[234] F. Rothlauf, D. E. Goldberg, and A. Heinzl. "Network random keys–a tree representation scheme for genetic and evolutionary algorithms". In: *Evolutionary Computation* 10.1 (2002), pp. 75–97.

[235] K. Roy and M. K. Naskar. "Genetic evolutionary algorithm for static traffic grooming to SONET over WDM optical networks". In: *Computer Communications* 30.17 (2007). Special Issue Concurrent Multipath Transport, pp. 3392–3402. ISSN: 0140-3664. DOI: 10.1016/j.comcom.2007.06.009.

[236] Ć. Rožić, D. Klonidis, and I. Tomkos. "A survey of multi-layer network optimization". In: *2016 International Conference on Optical Network Design and Modeling (ONDM)*. IEEE. May 2016, pp. 1–6. DOI: 10.1109/ONDM.2016.7494053.

[237] G. Rudolph. “Convergence analysis of canonical genetic algorithms”. In: *IEEE Transactions on Neural Networks* 5.1 (Jan. 1994), pp. 96–101. DOI: 10.1109/72.265964.

[238] D. F. Rueda, E. Calle, and J. L. Marzo. “Robustness Comparison of 15 Real Telecommunication Networks: Structural and Centrality Measurements”. In: *Journal of Network and Systems Management* 25.2 (Apr. 2017), pp. 269–289. ISSN: 1573-7705. DOI: 10.1007/s10922-016-9391-y.

[239] M. Ruffini. “Multidimensional Convergence in Future 5G Networks”. In: *Journal of Lightwave Technology* 35.3 (Feb. 2017), pp. 535–549. ISSN: 0733-8724. DOI: 10.1109/JLT.2016.2617896.

[240] M. Ruffini. *trafficGen – GitHub*. Jan. 16, 2016. URL: https://github.com/ruffinim/trafficGen (visited on 2018-04-06).

[241] M. Ruffini et al. “DISCUS: an end-to-end solution for ubiquitous broadband optical access”. In: *IEEE Communications Magazine* 52.2 (2014), S24–S32.

[242] M. Ruiz et al. “Survivable IP/MPLS-Over-WSON Multilayer Network Optimization”. In: *IEEE/OSA Journal of Optical Communications and Networking* 3.8 (Aug. 2011), pp. 629–640. DOI: 10.1364/JOCN.3.000629.

[243] R. Sabella et al. “Flexible packet-optical integration in the cloud age: Challenges and opportunities for network delayering”. In: *IEEE Communications Magazine* 52.1 (Jan. 2014), pp. 35–43. ISSN: 0163-6804. DOI: 10.1109/MCOM.2014.6710062.

[244] D. Saha, M. Purkayastha, and A. Mukherjee. “An approach to wide area WDM optical network design using genetic algorithm”. In: *Computer Communications* 22.2 (1999), pp. 156–172. ISSN: 0140-3664. DOI: 10.1016/S0140-3664(98)00238-2.

[245] H. Saini and A. K. Garg. “Investigations of Optimized Optical Network Performance Under Different Traffic Models”. In: *Advances in Computing and Data Sciences*. Ed. by M. Singh et al. Singapore: Springer Singapore, 2018, pp. 197–204. ISBN: 978-981-13-1813-9.

[246] H. Sayoud, K. Takahashi, and B. Vaillant. “Designing communication network topologies using steady-state genetic algorithms”. In: *IEEE Communications Letters* 5.3 (Mar. 2001), pp. 113–115. ISSN: 2373-7891. DOI: 10.1109/4234.913157.

[247] S. Schnitter et al. “Benefits from 2-layer traffic engineering for IP/MPLS networks”. In: *Networks 2008 - The 13th International Telecommunications Network Strategy and Planning Symposium*. Vol. Supplement. Sept. 2008, pp. 1–6. DOI: 10.1109/NETWKS.2008.6231337.

[248] E. Semenkin and M. Semenkina. “Self-configuring Genetic Algorithm with Modified Uniform Crossover Operator”. In: *Advances in Swarm Intelligence*. Ed. by Y. Tan, Y. Shi, and Z. Ji. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 414–421. ISBN: 978-3-642-30976-2.

[249] M. Simsek et al. “5G-Enabled Tactile Internet”. In: *IEEE Journal on Selected Areas in Communications* 34.3 (Mar. 2016), pp. 460–473. ISSN: 0733-8716. DOI: 10.1109/JSAC.2016.2525398.

[250] M. C. Sinclair. “Minimum Cost Topology Optimisation of the Cost 239 European Optical Network”. In: *Artificial Neural Nets and Genetic Algorithms*. Vienna: Springer Vienna, 1995, pp. 26–29. ISBN: 978-3-7091-7535-4.

[251] M. C. Sinclair. “Minimum cost wavelength-path routing and wavelength allocation using a genetic-algorithm/heuristic hybrid approach”. In: *IEE Proceedings - Communications* 146.1 (Feb. 1999), pp. 1–7. ISSN: 1350-2425. DOI: 10.1049/ip-com:19990281.

[252] M. C. Sinclair. “NOMaD: Applying a Genetic Algorithm/Heuristic Hybrid Approach to Optical Network Topology Design”. In: *Artificial Neural Nets and Genetic Algorithms*. Vienna: Springer Vienna, 1998, pp. 299–303. ISBN: 978-3-7091-6492-1.

[253] J. Sommer and J. Scharf. “IKR simulation library”. In: *Modeling and Tools for Network Simulation*. Springer, 2010, pp. 61–68.

[254] A. J. Soper and S. McKenzie. “The use of a biased heuristic by a genetic algorithm applied to the design of multipoint connections in a local access network”. In: *Second International Conference On Genetic Algorithms In Engineering Systems: Innovations And Applications*. Sept. 1997, pp. 113–116. DOI: 10.1049/cp:19971165.

[255] J. Späth and V. Feil. “Planning methods for photonic backbone networks”. In: *Proceedings of the NOC’97-III (Photonic Networks, Optical Technology and Infrastructure)*. 1997, pp. 47–54.

[256] N. Spring, R. Mahajan, and D. Wetherall. “Measuring ISP Topologies with Rocketfuel”. In: *SIGCOMM Comput. Commun. Rev.* 32.4 (Aug. 2002), pp. 133–145. ISSN: 0146-4833. DOI: 10.1145/964725.633039.

[257] U. Steglich et al. “A Generic Multi-layer Network Optimization Model with Demand Uncertainty”. In: *Advances in Communication Networking*. Ed. by T. Bauschert. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 13–24. ISBN: 978-3-642-40552-5.

[258] *STRONGEST Deliverable D2.1. Efficient and optimized network architecture: Requirements and reference scenarios*. Deliverable. Version version 2.0, AMENDED. Nov. 30, 2011.

[259] J. W. Suurballe and R. E. Tarjan. “A quick method for finding shortest pairs of disjoint paths”. In: *Networks* 14.2 (1984), pp. 325–336. DOI: 10.1002/net.3230140209.

[260] T. Takagi et al. “Disruption Minimized Spectrum Defragmentation in Elastic Optical Path Networks that Adopt Distance Adaptive Modulation”. In: *37th European Conference and Exposition on Optical Communications*. Optical Society of America, 2011, Mo.2.K.3. DOI: 10.1364/ECOC.2011.Mo.2.K.3.

[261] F. Teng and G. Zhou. “The Research of an Approach to Design Local Area Network Topology Based on Genetic Algorithm”. In: *2009 Second International Symposium on Computational Intelligence and Design*. Vol. 1. Dec. 2009, pp. 184–189. DOI: 10.1109/ISCID.2009.53.

[262] C. F. Teo et al. “Optimal placement of wavelength converters in WDM networks using particle swarm optimizer”. In: *2004 IEEE International Conference on Communications (IEEE Cat. No.04CH37577)*. Vol. 3. June 2004, pp. 1669–1673. DOI: 10.1109/ICC.2004.1312793.

[263] M. To and P. Neusy. “Unavailability analysis of long-haul networks”. In: *IEEE Journal on Selected Areas in Communications* 12.1 (1994), pp. 100–109.

[264] A. Todimala and B. Ramamurthy. “A Scalable Approach for Survivable Virtual Topology Routing in Optical WDM Networks”. In: *IEEE J.Sel. A. Commun.* 25.6 (Aug. 2007), pp. 63–69. ISSN: 0733-8716. DOI: 10.1109/JSAC-OCN.2007.020605.

[265] J. Triay and C. Cervello-Pastor. “An ant-based algorithm for distributed routing and wavelength assignment in dynamic optical networks”. In: *IEEE Journal on Selected Areas in Communications* 28.4 (May 2010), pp. 542–552. DOI: 10.1109/JSAC.2010.100504.

[266] S. Tse and G. Choudhury. “Real-Time Traffic Management in AT&T’s SDN-Enabled Core IP/Optical Network”. In: *Optical Fiber Communication Conference*. Optical Society of America, 2018, Tu3H.2. DOI: 10.1364/OFC.2018.Tu3H.2.

[267] J. W. Tukey. “Easy summaries – numerical and graphical”. In: *Exploratory Data Analysis*. Reading, Mass.: Addison-Wesley Pub. Co., 1977. Chap. 2, pp. 27–56. ISBN: 0-201-07616-0.

[268] S. Türk, R. Radeke, and R. Lehnert. “Network migration using ant colony optimization”. In: *2010 9th Conference of Telecommunication, Media and Internet*. June 2010, pp. 1–6. DOI: 10.1109/CTTE.2010.5557713.

[269] S. Türk et al. “Particle Swarm Optimization of network migration planning”. In: *2013 IEEE Global Communications Conference (GLOBECOM)*. Dec. 2013, pp. 2230–2235. DOI: 10.1109/GLOCOM.2013.6831406.

[270] G. N. Varela and M. C. Sinclair. “Ant colony optimisation for virtual-wavelength-path routing and wavelength allocation”. In: *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*. Vol. 3. July 1999, 1809–1816 Vol. 3. DOI: 10.1109/CEC.1999.785494.

[271] J.-P. Vasseur, M. Pickavet, and P. Demeester. *Network Recovery: Protection and Restoration of Optical, SONET-SDH, IP, and MPLS*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., July 2004. ISBN: 012715051X.

[272] L. Velasco et al. “A service-oriented hybrid access network and clouds architecture”. In: *IEEE Communications Magazine* 53.4 (Apr. 2015), pp. 159–165. ISSN: 0163-6804. DOI: 10.1109/MCOM.2015.7081090.

[273] L. Velasco et al. “Saving CAPEX by extending flexgrid-based core optical networks toward the edges [invited]”. In: *IEEE/OSA Journal of Optical Communications and Networking* 5.10 (Oct. 2013), A171–A183. ISSN: 1943-0620. DOI: 10.1364/JOCN.5.00A171.

[274] L. Velasco et al. “Value optimization of survivable multilayer IP/MPLS-over-WSON networks”. In: *Photonic Network Communications* 23.3 (June 2012), pp. 246–258. ISSN: 1572-8188. DOI: 10.1007/s11107-011-0354-7.

[275] M. Vissers et al. *SDN Architecture for Transport Networks, TR-522*. Technical Recommendation. Open Networking Foundation, 2016.

[276] G. I. Vjaceslavs Bobrovs Sandis Spolitis. “Latency causes and reduction in optical metro networks”. In: vol. 9008. 2014. DOI: 10.1117/12.2041736.

[277] D. Wagner. “Rate Adaptation for Hierarchical Packet Schedulers Considering Traffic Differentiation by Congestion Control”. PhD thesis. Universtiät Stuttgart, 2019.

[278] Z.-X. Wang, Z.-Q. Chen, and Z.-Z. Yuan. “QoS routing optimization strategy using genetic algorithm in optical fiber communication networks”. In: *Journal of Computer Science and Technology* 19.2 (Mar. 2004), pp. 213–217. ISSN: 1860-4749. DOI: 10.1007/BF02944799.

[279] Wang Yao and B. Ramamurthy. “Survivable traffic grooming with path protection at the connection level in WDM mesh networks”. In: *First International Conference on Broadband Networks*. Oct. 2004, pp. 310–319. DOI: 10.1109/BROADNETS.2004.80.

[280] *Waveserver*. Data Sheet. Ciena. 2018. URL: https://media.ciena.com/documents/Waveserver_DS.pdf (visited on 2019-03-28).

[281] B. M. Waxman. “Routing of multipoint connections”. In: *IEEE Journal on Selected Areas in Communications* 6.9 (1988), pp. 1617–1622.

[282] J. Wu. “A survey of WDM network reconfiguration: Strategies and triggering methods”. In: *Computer Networks* 55.11 (2011), pp. 2622–2645. ISSN: 1389-1286. DOI: 10.1016/j.comnet.2011.05.012.

[283] F. Xiang et al. “QoS routing based on genetic algorithm”. In: *Computer Communications* 22.15 (1999), pp. 1392–1399. ISSN: 0140-3664. DOI: 10.1016/S0140-3664(99)00113-9.

[284] P. Xiao et al. “A K self-adaptive SDN controller placement for wide area networks”. In: *Frontiers of Information Technology & Electronic Engineering* 17.7 (July 2016), pp. 620–633. ISSN: 2095-9230. DOI: 10.1631/FITEE.1500350.

[285] Y. Xin, G. N. Rouskas, and H. G. Perros. “On the Physical and Logical Topology Design of Large-Scale Optical Networks”. In: *J. Lightwave Technol.* 21.4 (Apr. 2003), p. 904.

[286] J. Y. Yen. “Finding the K Shortest Loopless Paths in a Network”. In: *Management Science* 17.11 (1971), pp. 712–716. DOI: 10.1287/mnsc.17.11.712.

[287] P. Yi, H. Ding, and B. Ramamurthy. “Cost-optimized joint resource allocation in grids/clouds with multilayer optical network architecture”. In: *IEEE/OSA Journal of Optical Communications and Networking* 6.10 (Oct. 2014), pp. 911–924. ISSN: 1943-0620. DOI: 10.1364/JOCN.6.000911.

[288] H. L. Yin, Y. M. Wang, and G. Z. Zhao. “A Novel Hybrid Algorithm for Big Resource Allocation Problems”. In: *2012 IEEE 12th International Conference on Computer and Information Technology*. 2012, pp. 124–128.

[289] İ. Yüksel Ergün. “Multi-Layer Network Design Problems In Telecommunication”. PhD thesis. Middle East Technical University, Sept. 2013.

[290] S. Yussof, R. A. Razali, and O. H. See. “A Parallel Genetic Algorithm for Shortest Path Routing Problem”. In: *2009 International Conference on Future Computer and Communication*. Apr. 2009, pp. 268–273. DOI: 10.1109/ICFCC.2009.36.

[291] M. S. Zefreh et al. “CAPEX optimization in multi-chassis multi-rate IP/optical networks”. In: *2016 International Conference on Computing, Networking and Communications (ICNC)*. Feb. 2016, pp. 1–6. DOI: 10.1109/ICCNC.2016.7440611.

[292] S. Zhang and B. Ramamurthy. “Dynamic traffic grooming algorithms for reconfigurable SONET over WDM networks”. In: *IEEE Journal on Selected Areas in Communications* 21.7 (Sept. 2003), pp. 1165–1172. DOI: 10.1109/JSAC.2003.815844.

[293] Y. Zhang et al. “Traffic-Based Reconfiguration for Logical Topologies in Large-Scale WDM Optical Networks”. In: *J. Lightwave Technol.* 23.10 (Oct. 2005), p. 2854.

[294] G. Zhou and M. Gen. “Genetic algorithm approach on multi-criteria minimum spanning tree problem”. In: *European Journal of Operational Research* 114.1 (1999), pp. 141–152.

[295] H. Zhu et al. “A novel generic graph model for traffic grooming in heterogeneous WDM mesh networks”. In: *IEEE/ACM Transactions on Networking* 11.2 (Apr. 2003), pp. 285–299. DOI: 10.1109/TNET.2003.810310.

[296] K. Zhu and B. Mukherjee. “Traffic grooming in an optical WDM mesh network”. In: *IEEE Journal on Selected Areas in Communications* 20.1 (Jan. 2002), pp. 122–133. ISSN: 0733-8716. DOI: 10.1109/49.974667.

[297] Y. Zu et al. “The routing algorithm of multi-layer multi-domain intelligent optical networks based on ant colony optimization”. In: *2013 15th IEEE International Conference on Communication Technology*. Nov. 2013, pp. 350–354. DOI: 10.1109/ICCT.2013.6820399.

# Acknowledgments

This thesis represents the epitome of six years of working as a research staff member at the Institute of Communication Networks and Computer Engineering (IKR). A time ripe with scholarly discussions about fascinating topics, inspiring interactions with interesting people and educational endeavors in communicating the merits of science and engineering to a younger generation. The agglomeration of these experiences has not only culminated in a more profound understanding of the scientific method and its accompanying skill set, but has also bestowed a more diversified perception upon my persona, to which I am very grateful.

I would like to thank Prof. Dr.-Ing. Andreas Kirstädter, the head of the institute, not only for his counsel and guidance surrounding my thesis, but especially for his ready support in all manners research and teaching. His suggestions and insights have provided me with many ideas and the impetus to bring them to fruition. I am also very grateful to Prof. Kirstädter and to Prof. Dr.-Ing. Thomas Bauschert, whom I had the fortune to meet and discuss with at many conferences, for their readiness to assess this thesis.

I also would like to thank Ulrich Gemkow for his continuous support of all my plans and ideas. His guidance and experience regarding research and teaching had always been a formative influence on my decision making. I am especially grateful for many discussions, regarding a wealth of topics including many aspects of my thesis, which always helped me see things with a heightened sense of clarity.

Furthermore, I would also like to extent my thanks to my fellow researchers at the IKR, both past and present, who have shaped the institute to be the nurturing environment of interesting persons and ideas that it is. These are in chronological order Matthias Meyer, Matthias Kaschub, Frank Feller, Magnus Proebster, Thomas Werthmann, Mirja Kühlewind, Sebastian Meier, Domenic Teuchert, Johannes Häussler, David Wagner, Alexander Vensmer, Sebastian Scholz, Kristian Ulshöfer, Hani Samara, Simon Blum, Tobias Enderle, Arthur Witt, Filippos Christou, Christian Köhler, and Nicolas Hornek.

I am especially grateful to Matthias Meyer, Frank Feller, David Wagner, Thomas Werthmann and Sebastian Meier for teaching me the ins and outs of working at the institute and to Magnus Proebster and Kristian Ulshöfer for providing a friendly office space. While all of these people contributed to my work, directly or indirectly, I am particularly indebted to Frank Feller, who introduced me to network optimization, and Tobias Enderle, who reviewed almost the entirety of my manuscript. Both dedicated vast amounts of their time to discussions about my research and helped form many essential ideas.

Many ideas were also formed in a number of student theses, which thereby contributed to my research. For these collaborations I want to extend my thanks to Karsten Schöck, Thilo Rachlitz, Sebastian Eckardt, Lukas Billinger, Daniel Nägele, Xingbo Peng and Xiaochang Wang. I also want to extend a special thanks to Prof. em. Dr.-Ing. Dr. h.c. mult. Paul J. Kühn, who was always

available to spark interesting discussions, and to Kathrin Braust and Silke Loureiro, who were always supportive and helped behind the scenes.

I also profited significantly from collaborations and discussions with other researchers in the SASER and SENDATE projects, as well as in support of student theses. I particularly enjoyed discussions with Lars Dembeck, Wolfram Lautenschläger, Ulrich Gebhard, Jens Milbrandt, Frank Ilchmann, Konstantinos Christodoulopoulos, Tobias Röthlingshöfer, Roland Wessäly, Norbert Ascheuer, Christian Raack, Robert Protzmann, Sebastian Kiesel, Sebastian Neuner, Zigmund Orlov, Prof. Dr. Thomas Zinner, and Prof. Dr.-Ing. Michael Scharf.

Last, but certainly not least, I want to express my deep gratitude towards my friends and family, who endured my moods, my absent-mindedness and my actual physical absence, but never stopped supporting me. I am especially grateful to my parents, who had always encouraged me to pursue my dreams and made me who I am, to my brother Arndt, who helped me through the most difficult of times with his courage and steadfastness, and –most importantly– my fantastic wife Susan, who helped me in innumerable ways from consolation over failed experiments, proofreading my thesis to wholehearted encouragement, never to give up.