


Grooming Connectivity Intents in IP-Optical Networks Using Directed Acyclic Graphs

Filippos Christou 

*Institute of Communication Networks
and Computer Engineering (IKR)
University of Stuttgart
Stuttgart, Germany
filippos.christou@ikr.uni-stuttgart.de*

Andreas Kirstädter

*Institute of Communication Networks
and Computer Engineering (IKR)
University of Stuttgart
Stuttgart, Germany
andreas.kirstaedter@ikr.uni-stuttgart.de*

Abstract—During the last few years, there have been concentrated efforts toward intent-driven networking. While relying upon Software-Defined Networking (SDN), Intent-Based Networking (IBN) pushes the frontiers of efficient networking by decoupling the intentions of a network operator (i.e., what is desired to be done) from the implementation (i.e., how is it achieved). The advantages of such a paradigm have long been argued and include, but are not limited to, the reduction of human errors, reduced expertise requirements among operator personnel, and faster business plan adaptation. In previous work, we have shown how incorporating IBN in multi-domain networks can have a significantly positive impact as it can enable decentralized operation, accountability, and confidentiality. The pillar of our previous contribution is the compilation of intents using system-generated intent trees. In this work, we extend the architecture to enable grooming among the user intents. Therefore, separate intents can now end up using the same network resources. While this makes the intent system reasonably more complex, it indisputably improves resource allocation. To represent the intent relationships of the newly enhanced architecture, we use Directed Acyclic Graphs (DAGs). Furthermore, we appropriately adapt an advanced established technique from the literature to solve the Routing, Modulation, and Spectrum Assignment (RMSA) problem for the intent compilation. We demonstrate a realistic scenario in which we evaluate our architecture and the intent compilation strategy. Our current approach successfully consolidates the advantages of having an intent-driven architecture and, at the same time, flexibly choosing among advanced resource allocation techniques.

Index Terms—architecture, IBN, RMSA, DAG

I. INTRODUCTION

With the dramatic growth of data traffic in IP-optical networks, efficient and scalable network control and management solutions have become increasingly critical. Intent-driven networking has emerged as a promising approach to simplify these tasks by allowing operators to express high-level network objectives as intents. Connectivity intents define the desired end-to-end connectivity between network nodes and are used to automatically configure the underlying network infrastructure. The Intent-Based Networking (IBN) framework responsible for this automatic implementation is logically placed on top of the Software-Defined Networking (SDN) controller and carries the network logic. The SDN controller then gets exclusively dedicated to facilitating the communication between the IBN framework and the network devices; it receives a role similar

to a device driver in computer systems. Thus, the operator will send connectivity intents to the IBN framework. The IBN framework compiles the intents to an implementation, which is then forwarded to the SDN controller to be installed into the devices.

One of the critical design choices in operating an IBN framework is choosing the compilation algorithm. The compilation algorithm receives an abstract high-level network objective and automatically outputs an implementation that can be realized in the network at a specific time. Depending on the objective's nature, different algorithms must be invoked. For example, Network Function Virtualization (NFV) intents could be addressed using algorithms for Virtual Network Functions (VNFs) placement and Service Function Chain (SFC) deployment [1], [2]. Since we consider IP-optical connectivity intents in this work, the serving algorithmic family is that of Routing, Modulation, and Spectrum Assignment (RMSA) [3].

Ideally, we would like to reuse any of the algorithms from the literature in the intent-driven environment. This can be done effortlessly using a one-step compilation procedure and independently invoking the preferred algorithm when new intents must be compiled. The resulting interoperability with legacy algorithms would allow some of the advantages of IBN, like the holistic view and the built-in architecture for the intent lifecycle. However, a mere "re-branding" of the algorithms without truly adapting them to the intent-driven architecture would require re-implementation of procedures like intent monitoring (making sure that installed intents remain successfully installed), intent conflicts resolution (when more than one intent requires the same resources), and intent re-provisioning (in case of network failures) each time according to the legacy algorithm chosen.

In our previous work [4], we have outlined a series of advantages that can be gained by having a multi-step intent compilation approach. Specifically, we have concluded that hierarchical system-generated intent structures, like intent trees, can offer efficient decentralized coordination of multi-domain IP-optical networks. Although possibly requiring more effort to bind to legacy algorithms, this approach promises flexible and scalable communication mechanisms, confidentiality, and accountability. However, there is no inherent possibility to use algorithms that leverage grooming as long as intent trees are used for the representation of the multi-step intent compilation. IP-optical traffic grooming [2] is a popular technique that packs several demands into the same optical spectrum channel,

This work has been performed in the framework of the CELTIC-NEXT EUREKA project AI-NET-ANTILLAS (Project ID C2019/3-3), and it is partly funded by the German BMBF (Project ID 16KIS1312).

thus increasing resource utilization.

This paper fills this gap by developing a new approach that permits IP-optical grooming by substituting the intent trees with an intent Directed Acyclic Graph (DAG). As a result, we reinvent an intent-driven framework that inherits all the benefits from its predecessor and can successfully integrate any grooming-enabled RMSA algorithm. To demonstrate this universality, we adopt an established RMSA algorithm from the literature [5]. We also proceed to slight modifications to prioritize low-latency paths, showing the algorithm’s preserved flexibility in the intent-driven environment. We validate the operation of the architecture by simulating a realistic scenario with well-known results. We limit our focus on single-domain scenarios to better focus on the task at hand.

In the following section, we introduce the new architecture using an intent DAG as opposed to the one using intent trees. Section III describes the compilation algorithms and the integration steps needed for [5]. In Section IV, we evaluate our architecture in a realistic scenario seeking to reproduce the expected well-known results. In Section V, we conclude the paper and disclose future directions.

II. ARCHITECTURE

In this section, we will first briefly revisit the architecture of [4] using intent trees and then contrast it with the novel version using an intent DAG.

A. Intent Trees

Intent compilation using intent trees recursively generates child intents from higher-level intents. The intent tree is a hierarchical representation of the network intent, where the depth corresponds to a different level of abstraction. The tree’s root represents the high-level user intent, while the leaf nodes represent the low-level intents which are the needed device configurations.

The top of Fig. 1 illustrates two different intent trees corresponding to two different user intents. The generation of the intent tree is specific to the compilation algorithm chosen. One-step compilation algorithms are still possible; in that case, the tree would be two levels deep and be composed exclusively of the root user intent and its low-level intents children.

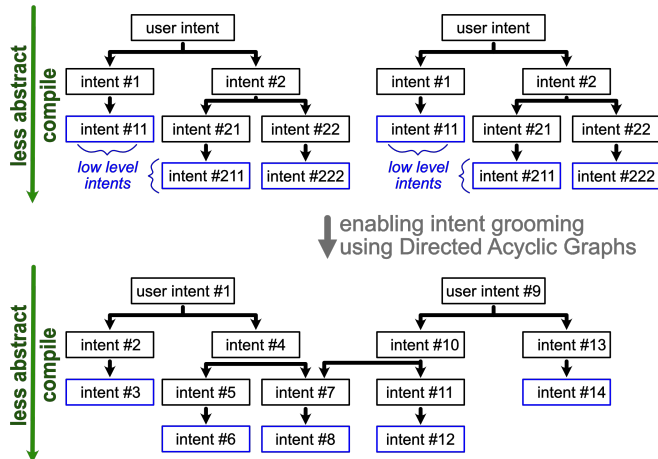


Fig. 1. Comparing intent trees and intent DAG

It is important to notice that separate user intents define separate intent trees, and there is no way of merging them. As a result, the resources reserved by the low-level intents in one intent tree cannot be jointly used by another, rendering support for grooming impossible.

B. Intent DAG

To reuse resources (i.e., low-level intents) across different user intents, we must allow some intent nodes to have several parents, leading us to use a DAG. Furthermore, DAGs preserve the direction the same way it exists in a tree and avoid unwanted cyclic dependencies. In contrast with the intent trees, where the number of intent trees directly depends on the number of user intents, the intent DAG is unique per IBN framework and includes all user intents. However, the intent DAG might be disconnected (when no grooming is done), and then practically several partial DAGs will appear. The bottom of Fig. 1 illustrates how an intent DAG could modify the intent structure. A unified indexing is also necessary to combine all the intent trees.

The following network intents were defined to solve the grooming-enabled RMSA problem.

- *LightpathIntent* is an intent that defines a lightpath across some network nodes. These intents are the grooming points for IP-optical networks. If our IP-optical intent system were applied to the bottom of Fig. 1, then intent #7 would be a *LightpathIntent*.
- *SpectrumIntent* is an intent that defines the spectrum requirements of a parent *LightpathIntent*.
- *NodeTransmoduleIntent* is a low-level intent that requires using a particular transmission module in a multilayer node.
- *NodeRouterPortIntent* is a low-level intent that requires using a port in the router of a multilayer node.
- *NodeSpectrumIntent* is a low-level intent that requires the reservation of some spectrum slots in a link connected to a multilayer node.

Fig. 2 shows an example of a compiled intent involving all the aforementioned intents.

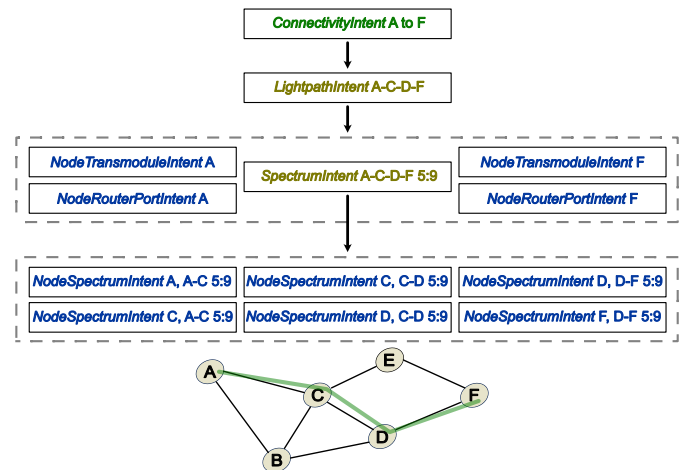


Fig. 2. Example of a compiled intent. The connection from node A to node F is implemented with one lightpath covering the nodes A-C-D-F and using the spectrum slots 5,6,7,8,9 along the involved fibers.

III. INTENT COMPILATION ALGORITHMS

On top of the intent DAG architecture, any grooming-enabled RMSA algorithm could be implemented. In this section, we will look closer at three of such algorithms.

A. Shortest Available Path

The Shortest Available Path (SAP) is the simplest algorithm, also used in [4], and is treated as the baseline. It is an RMSA algorithm that first solves routing using k-shortest-path and then the spectrum assignment using first-fit [6]. If the path is unavailable, the next shortest path is considered. Using algorithms without grooming like this makes no difference whether we have intent trees or an intent DAG.

B. Joint Multilayer

The Joint Multilayer (JML) algorithm has been proposed in [5]. It is a complex algorithm that can be used in an online fashion and is composed of the following steps:

- Create a directed multilayer multigraph.

The layers of the graph correspond to the optical and electrical views. Each topology node is converted to a two-layered node composed of an IP router vertex and an Optical Cross-Connect (OXC) vertex. The edges in the electrical layer signify the established lightpaths (virtual links), and the edges in the physical layer are the fibers. Inter-layer links are the transmission modules connecting the OXC with the IP router and vice versa. Since more than one transmission module might be available, a multigraph is needed to accommodate several edges between the same vertices.

- Calculate the cost vector for all edges.

Each edge e in the multilayer multigraph is described by a vector $(D_e, C_e, P_e, \bar{H}_e, F_e, \bar{W}_e, T_e, I_e, L_e)$. D_e is the distance covered by the last transmission module, i.e., since the last regeneration. C_e is the cost of the transmission module. P_e is the cost of the router port. \bar{H}_e is the vector of transmission module mode tuples $[(r_1, d_1, b_1), (r_2, d_2, b_2), \dots]$, where r_i, d_i, b_i is the transmission rate, the optical reach, and the spectrum slot requirements, respectively. F_e is a boolean variable specifying whether the link is virtual. \bar{W}_e is the boolean

vector indicating the availability of the spectrum slots. The last three components, $T_e, I_e,$ and $L_e,$ are new additions needed for the adaptation as an intent DAG compilation algorithm. T_e is the type of link (virtual, optical, optical-to-virtual, or virtual-to-optical) and is used because different actions are needed based on the link type. I_e is the index of the intent DAG node if the (virtual) link corresponds to an already established *LightpathIntent*. L_e is the physical link length. Several link cost vectors can be added to create a path cost vector with similar components $(D_p, C_p, P_p, \bar{H}_p, \bar{W}_p, I_p, L_p, \bar{R}_p, \bar{p})$, where \bar{R}_p are the transmission modules chosen along the way and \bar{p} is the path in the multilayer multigraph.

- Obtain non-dominated paths.

Here, an algorithm generates a series of candidate multilayer paths. A multilayer path comprises a series of optical, virtual, optical-to-virtual, and virtual-to-optical links in the correct logical order. This collection excludes the paths whose path cost vector is categorically worse than others, i.e., they are dominated by other paths. Such comparison can only happen between paths of the same source and destination. A path p_1 is categorically better (i.e., dominates) than a path p_2 if

$$D_{p_1} \leq D_{p_2} \text{ and } C_{p_1} + P_{p_1} \leq C_{p_2} + P_{p_2} \text{ and } F_{p_1} \leq F_{p_2} \\ \text{and } \max_{\text{rate}} \bar{R}_{p_1} \geq \max_{\text{rate}} \bar{R}_{p_2} \text{ and } \bar{W}_{p_1} \geq \bar{W}_{p_2}$$

During the non-dominated paths generation, care is being taken to only output valid multilayer paths, which respect the limitations imposed by the port rate, optical reach, and bandwidth requirements in \bar{H}_p . Paths picked in this step serve as candidate paths for the next step.

- Select the *winner path*.

An optimization function is chosen based on the preferences, and a path is selected from all the candidates that minimize this function. The legacy optimization function the JML algorithm uses aggregates all electrical and optical costs $C_p + P_p$. As a result, the path is selected that minimizes these costs at best.

- Allocate spectrum slots.

After choosing the winner path, the spectrum allocation is conducted with the preferred algorithm. The legacy algorithm uses the first fit.

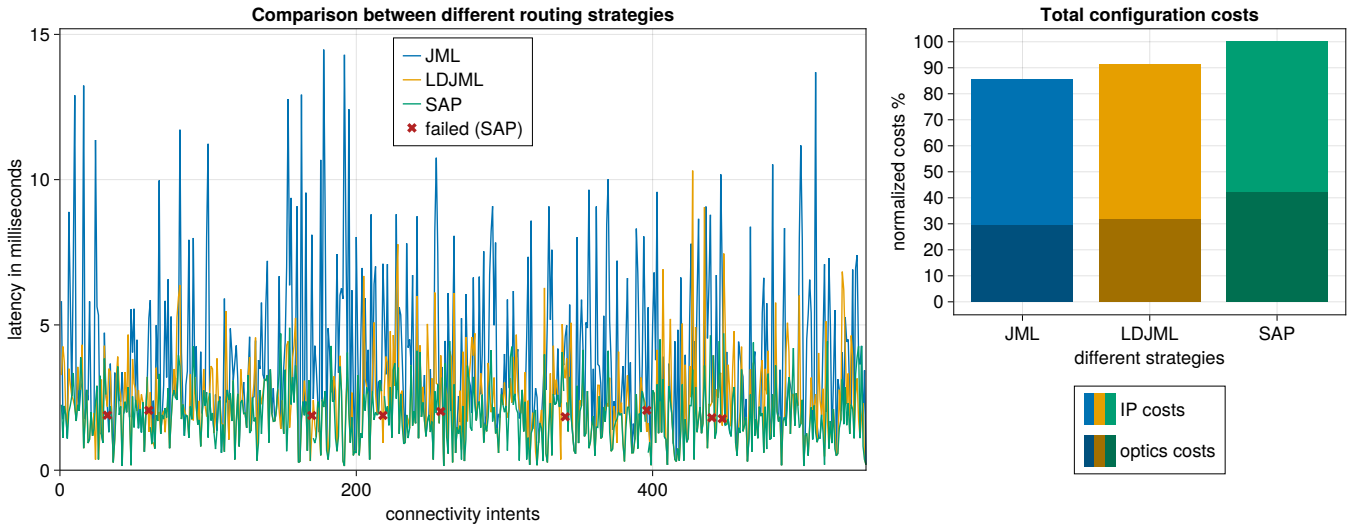


Fig. 3. Single simulation comparison. On the right, the IP and optics costs are the same as the attributes P_p and C_p in the path cost vector.

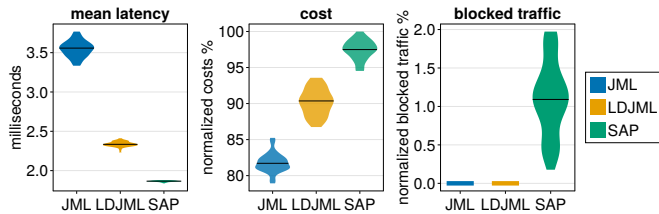


Fig. 4. Multiple simulations comparison. The black line in the violin plots is the median.

Following are the extra steps introduced in the procedure for the adaptation to the intent DAG compilation.

- Break solution into predefined intents.

Once the algorithm yields an output path with all the needed resource allocations, this information is mapped into the intents defined in Section II. The result will be similar to Fig. 2.

- Attach intents to the intent DAG.

In the final step of the compilation algorithm, the newly generated nodes are added to the intent DAG as descendants of the user intent. In case of no grooming, all intents involved will create new child intents. If grooming takes place, not all involved intents will be added to the DAG, but instead, an edge will be created between the involved intents and a *LightpathIntent* already existing in the DAG identified by the I_e attribute.

C. Latency-Driven Joint Multilayer

After being ported into the intent-based regime, the adapted algorithm JML should remain easy to modify as designed by the original authors of [5]. Indeed, we can easily modify the optimization function to adjust the algorithm to focus on finding the shortest paths, rendering the intent-driven approach equally flexible. For this reason, we had to introduce the extra attribute L_p to quantify the physical length of the path, which now needs to be minimized. With a lower priority and in case of a tie between paths, we fall back to the previous objective function $C_p + P_p$. We called this slight variation Latency-Driven Joint Multilayer (LDJML), which is further used during the evaluation to attain more variety of use cases.

IV. EVALUATION

In this section, we will evaluate the presented architecture. The results should not invoke surprise, as the core algorithms used are well-known and should yield the expected results. This section has the role of proof of concept, where we validate that the adapted compilation algorithm JML and its slight variation LDJML operate as expected.

For the simulation, we used the Nobel-Germany topology from [7]. The demand matrix is generated using a truncated normal distribution for every node pair, aggregating to circa 62 Tbps for the whole network. Each entry on the demand matrix is then used to issue a connectivity intent. The cost and network equipment model is derived from [8].

A. Single Simulation

Fig. 3 shows the results of running a single simulation for each compilation algorithm presented in Section III. On the left, we can see the latency for every connectivity intent for all compilation algorithms. SAP generally delivers the lowest latency, JML the highest, and LDJML is, as designed,

in the middle. However, SAP does not leverage grooming, and 62 Tbps are already too much to accommodate with such a naive approach. As a result, this leads to certain connectivity intents being blocked. Blocking does not happen for JML and LDJML, which both leverage grooming and thus manage their resources more efficiently.

We witness similar results also regarding the cost of the three compilation algorithms. SAP presents the highest costs as it allocates more resources for every new connectivity intent. JML presents the lowest costs as it is the objective of the optimization function used. LDJML again stands in the middle as it uses grooming, which helps reduce the costs, but also tries to minimize latency with a higher priority.

B. Multiple Simulations

To achieve more confidence in the results, we repeat the simulation 40 times using different seeds for the random generation of the demand matrix. Fig. 4 confirms the same results as described previously. LDJML always stands in the middle between JML and SAP, with SAP having the lowest latency, highest cost, and some blocking and JML having the highest latency, lowest cost, and no blocking. LDJML also shows no blocking due to the grooming capabilities enabled by the intent DAG compilation design.

V. CONCLUSION

In this paper, we designed a modular architecture for intent compilation algorithms. We used an intent DAG to represent the realization of any grooming-enabled RMSA algorithm. Using the proposed architecture, any related algorithm can be adapted while also preserving the possibility for further modifications. This work enables the migration of legacy algorithms to the intent-based regime for single-domain scenarios. We demonstrated the validity of our approach by making simulations using three different intent compilation algorithms and confirming the well-known results. As new, exciting benefits are still to be gained for decentralized multi-domain operation, future work will focus on expanding and analyzing the presence of intent DAGs in such use cases.

REFERENCES

- [1] B. Yi, X. Wang, K. Li, S. k. Das, and M. Huang, "A comprehensive survey of network function virtualization," *Computer Networks*, vol. 133, pp. 212–262, 2018.
- [2] S. Miladić-Tešić, G. Marković, and V. Radojčić, "Traffic grooming technique for elastic optical networks: A survey," *Optik*, vol. 176, pp. 464–475, 2019.
- [3] F. Shirin Abkenar and A. Ghaffarpour Rahbar, "Study and analysis of routing and spectrum allocation (rsa) and routing, modulation and spectrum allocation (rmsa) algorithms in elastic optical networks (eons)," *Optical Switching and Networking*, vol. 23, pp. 5–39, 2017.
- [4] F. Christou, "Decentralized intent-driven coordination of multi-domain ip-optical networks," in *2022 18th International Conference on Network and Service Management (CNSM)*, 2022, pp. 359–363.
- [5] V. Gkamas, K. Christodouloupoulos, and E. Varvarigos, "A joint multi-layer planning algorithm for ip over flexible optical networks," *Journal of Lightwave Technology*, vol. 33, no. 14, pp. 2965–2977, 2015.
- [6] B. C. Chatterjee and E. Oki, "Performance evaluation of spectrum allocation policies for elastic optical networks," in *2015 17th International Conference on Transparent Optical Networks (ICTON)*, 2015, pp. 1–4.
- [7] S. Orłowski, R. Wessály, M. Pioro, and A. Tomaszewski, "Sndlib 1.0—survivable network design library," *Networks*, vol. 55, pp. 276–286, 01 2009.
- [8] F. Christou, T. Enderle, and A. Witt, "Towards a hybrid architecture by introducing coherent pluggable transceivers in ip-optical core networks with optical cross-connects," in *Photonic Networks; 23th ITG-Symposium*, 2022.