Cost-Based Topology Optimization of Embedded Ethernet Networks

Jörg Sommer, University of Stuttgart, Germany

Elias A. Doumith, University of Stuttgart, Germany

Andreas Reifert, University of Stuttgart, Germany

ABSTRACT

During past decades, Ethernet progressively became the most widely used Local Area Network (LAN) technology. Apart from LAN installations, Ethernet also became attractive for other application areas such as industrial control, automotive, and avionics. In traditional LAN design, the objective is to minimize the network deployment cost. However, in embedded networks, additional constraints and ambient conditions add to the complexity of the problem. In this paper, the authors propose a Simulated Annealing (SA) algorithm to optimize the physical topology of an embedded Ethernet network. The various constraints and ambient conditions are modeled by a cost map. For networks with small number of nodes and/or switches, the authors were able to find the optimal solutions using adapted algorithms. These solutions will serve as a lower bound for the solutions obtained via the SA algorithm. However, the adapted algorithms are time consuming and application specific. The paper shows that the SA algorithm can be applied in all cases and finds (near-) optimal solutions.

Keywords: *Embedded Networks, Ethernet, Optimization, Simulated Annealing, Topology Design*

INTRODUCTION

Today, Ethernet is the predominant Local Area Network (LAN) technology and is considered as a *de facto* standard for network infrastructure. The flexibility and the plug-and-play feature of Ethernet are its key to success. Thanks to the wide availability of its components, its large bandwidth, its reliability, and its backward-compatibility, Ethernet has become an attractive option in many application areas (Sommer et al., 2010).

A prominent example of Ethernet's application area is the industrial domain where Ethernet is gaining ground over traditional fieldbuses (Felser, 2005; Decotignie, 2005). Another area is avionics, where today's Ethernet proved its ability to fulfill real-time requirements needed in such an environment (ARINC 664, 2003). Last but not least, the automotive industry is investigating Ethernet as a suitable in-car network technology (Rahmani, Hillebrand, Hintermaier, Bogenberger, & Steinbach, 2007;

DOI: 10.4018/jertcs.2011010101

Rahmani, Steffen, Tappayuthpijarn, Steinbach, & Giordano, 2008; Rahmani, Tappayuthpijarn, Krebs, Steinbach, & Bogenberger, 2009).

A motivation to use Ethernet in embedded networks is the use of commercial off-the-shelf components (Sommer et al., 2010). This allows integrators to cut down the development cost as well as the time needed to build new components. Furthermore, the large number of Ethernet vendors and the wide range of products promote the competition to provide the best equipment at the lowest price.

In contrast to traditional LANs, an embedded network has to meet additional requirements. For example, in an aircraft or a car, the installation space and the maximum weight tolerated are limited. Furthermore, the ambient conditions impose constraints on the network deployment. For instance, at some places, we need a better cable shielding due to electromagnetic interference or an extra heat-resisting cable due to the environment temperature. At other places, it might be extremely expensive or even impossible to deploy a cable. Moreover, at some places we need additional cables for the power supply of the switches while at other places such as at the positions of the nodes (devices), power supplies are already available. These constraints add to the design complexity of an embedded network.

Ethernet evolved from a bus topology to a so-called micro-segmented network with full duplex links between nodes and switches. On the one hand, deploying a single switch and connecting all the nodes directly to it has the lowest switch cost. However, such a star topology leads to wire bundles, which increase the wiring cost. On the other hand, deploying multiple switches, which increases the switching cost, alleviates the wiring layout, and consequently reduces the wiring cost proportional to the total wire length. Hence, the resulting problem is a multi-objective optimization problem where the optimal solution is a trade-off between the cost penalty due to the additional number of switches and the cost benefit due to the reduction of the total wiring length.

In this paper, we propose a Simulated Annealing (SA) algorithm to optimize the physical topology by minimizing the total link cost of an embedded Ethernet network. We take into account the aforementioned constraints and ambient conditions by modeling them with a cost map. Assuming a constant cost map (fixed cost over all the environment space), we are able to find the optimal solution by means of a Mixed-Integer Linear Program (MILP). For special cases, a Minimum Spanning Tree (MST) algorithm and an exact algorithm for the Steiner Tree (ST) problem can be used. For these cases, we show that the proposed SA algorithm provides optimal solutions in short running time. For general cost map models, the complexity of the problem increases drastically and the ST algorithm is unable to give a solution within reasonable time on today's computers. If we restrict the switches to be placed at the same positions as the nodes, the MILP and the MST algorithms are still able to find optimal solutions, while the proposed SA algorithm is the only general algorithm that keeps its pace with the increasing complexity of the problem.

The rest of the paper is organized as follows. First, we describe the problem and introduce the cost map model. Then, we present in Section *The Simulated Annealing Algorithm* the proposed SA algorithm, and discuss in the following section other related algorithms that find optimal solutions, and state the cases where they can be applied. In Section *Performance Evaluation*, we evaluate the performance of the SA algorithm and compare two embedded network designs. Finally, we conclude the paper and give an outlook for future work.

Related Work

Topology optimization is a frequently encountered problem in network design and planning. Since the early nineties, the problem of topology optimization has been investigated in many fields of application. For instance, networks-on-chip require the reduction of the wire lengths in order to optimize for speed and

power consumption. This can be achieved by keeping the most critical interconnections as short as possible through system partitioning and block placement in the layout (Ahonen, Sigüenza-Tortosa, Bin, & Nurmi, 2004). Power distribution system is another field where topology optimization plays an important role. For instance, the authors of (Wakileh & Pahwa, 1996) investigated the problem of restoring power to a distribution system that has experienced a long-duration outage. This was formulated as an optimization problem where the objective is to find the size of the transformers as well as the number of switches that minimize an annual cost function. However, the most-known application area of topology optimization is the design of transport networks such as mobile networks (GSM, UMTS, WiMax, etc.), fixed and wireless access networks, and optical core networks. In Harmatos, Jüttner, and Szentesi (1999), the authors propose a topology optimization problem for UMTS transport networks. The proposed method optimizes the number and locations of base station controllers together with the transmission network topology. The method is based on an SA algorithm combined with a greedy algorithm. In Pióro and Mehdi (2004), the authors list many other works for a wide variety of topology design problems.

Topology optimization problems can be classified in capacity problems, commonly known as network dimensioning problems, and location problems. The network dimensioning problems consist of determining the minimum capacity requirements that allow the network to satisfy a given traffic matrix. This involves determining a route for each traffic demand and computing the maximum number of channels required between the switches. For example, in Mukherjee, Banerjee, Ramamurthy, and Mukherjee (1996), the authors minimize the network-wide average packet delay for a given traffic matrix and maximize the scale factor by which the traffic matrix can be scaled up. Location problems involve determining where to place particular components and how to connect them. The proposed approaches to solve this problem determine the cost of transmission and the cost of switching, and thereby determine the optimum connection matrix and the location of switches and concentrators while guaranteeing a minimal connectivity. In Chardaire and Lutton (1993), the authors present an SA algorithm that optimizes the cost of telecommunication networks. In this work, the problem consists in finding the number of concentrators, their locations, and the connection of the terminals to concentrators while minimizing the total cost of the network. However, this approach assumes a predefined set of possible locations of the concentrators and the algorithm has to choose among these locations. Moreover, all the concentrators are connected to a central point that simplifies even more the problem of interconnecting the concentrators between them. The connection cost is assumed to be the Euclidean distance. Thus, it does not take into account ambient conditions. Finally, only heuristics were considered and the authors never assessed the quality of the obtained results by comparing them with the optimal results.

Problem Description

In this paper, we propose an SA algorithm to optimize the physical topology of an embedded Ethernet network without redundant paths. Our approach takes into account constraints and ambient conditions modeled by a cost map. The resulting topology guarantees low network cost, but is unable to recover from a link cut or a switch failure. In such topologies, each node is directly connected to a single switch via a point-to-point link. The switches are interconnected in a tree topology. Deploying additional switches gives rise to further optimization issues such as the optimal number of switches, their optimal positions, and the connectivity between them.

In this work, we consider two network designs, based on a tree topology, that differ in the acceptable positions of the switches.

- In the first design referred to as *Integrated Switches*, the switches can be placed only at the same positions as the nodes (cf., Figure 1 a.)). This design is motivated by the increasing number of devices with a built-in switch available on the market. These integrated switches use the same power supply as the nodes and require minimal additional installation space.
- In the second design referred to as *Self-contained Switches*, the switches can be placed anywhere in the environment space (cf., Figure 1 b.)). Hence, the position of the switches has a higher degree of freedom and thus can achieve a reduced link deployment cost.

It is up to the manufacturers to decide which design is suitable for their embedded networks.

It is to be noted that this problem is harder to solve than traditional network dimensioning problems (Pióro & Mehdi, 2004) where the position of the switches is given and the connectivity between them has to be optimized. It is also harder to solve than already investigated location problems where the position of the switches/concentrators is limited to a small set of possible locations and the connectivity between them is achieved using a central point. In our case, only the nodes' positions are given and we have to optimize the switches' positions as well as the connectivity between them and towards the nodes using a tree topology. When the number of switches is not limited, this problem is reduced to the ST problem that is NP-complete (Garey, Graham, & Johnson, 1977).

Cost Modeling

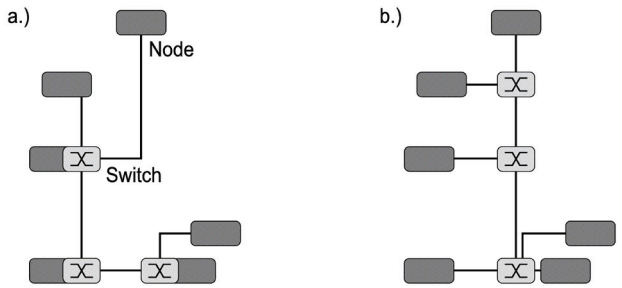
We formulate the total network deployment cost C_{Net} of an Ethernet network without redundant paths (see Box 1.) where:

- n is the number of nodes and thus the number of full duplex links used to connect the nodes to the switches.
- m is the number of switches used. Consequently, in a tree topology, $m-1$ is the number of full duplex links used to connect the switches between them.
- C_C is the cost of a connector (endpoint) composed of a physical layer device and a Medium Access Control (MAC) unit. Due to the micro-segmentation of the

Box 1.

$$C_{\text{Net}} = \underbrace{2\cdot(n+m-1)\cdot C_C + m\cdot C_S}_{\text{Position independent cost}} + \underbrace{C}_{\text{Position dependent cost}} \quad (1)$$

Figure 1. Embedded Ethernet network designs: a.) Tree with integrated switches b.) Tree with self-contained switches



network, there exist only point-to-point (full duplex) links. Therefore, we have to deploy a connector at both ends of each link. In our topology, we have a total number of $n+m-1$ full duplex links that require $2\cdot(n+m-1)$ connectors.

- C_S is the cost of a switching fabric (e.g., a relay unit) without the physical layer devices and the MAC units.
- C is the cost of deploying full duplex links that connect the nodes to the switches and the switches between them. This cost is proportional to the length of the links used to interconnect the nodes and switches, and depends on the positions where these are deployed. In the next sections, we introduce the principle of a cost map and detail how to compute the cost C.

Through all this paper, we assume that we deploy only full duplex links. For sake of simplicity, from now we will use the term *link* to denote a *full duplex link*.

Cost Map

As introduced previously, due to the ambient conditions, the cost of deploying a link depends on its position. In order to take into account this cost variation, we introduce the principle of a cost map. For this purpose, we assume that the environment is reduced to a two-dimensional plane of size $u\cdot v$ rasterized into small areas or pixels. The value assigned to a given pixel represents the cost of deploying a link segment at that particular position. From this cost map, we can compute the cost of any link connecting any two nodes/switches in the network as the sum of the cost of the underlying contiguous pixels where this link is deployed.

Typically, we can derive the cost map from any environment skeleton where already existing link ducts are represented by pixels with the lowest cost. However, this information is application specific and is, in most cases, proprietary for the owner of the embedded environment. For this reason, we chose to use randomly generated maps to test our SA algorithm. We want to point out, though, that the performance of our algorithm is independent of the model used to create the cost map. For the sake of completeness, we will present in the following our approach for generating the cost map.

In embedded environments, low cost areas mostly have low cost neighboring areas. The same usually holds for high cost areas. Thus, pure random maps are not sufficient, they must exhibit a certain autocorrelation structure given by an autocorrelation matrix $\underline{\mathcal{G}}$. Such random maps with a given autocorrelation structure are used in mobile radio network simulators to model the *shadowing* or *shadow fading* effect (Cai & Giannakis, 2003; Patzold & Nguyen, 2004; Forkel, Schinnenburg, & Ang, 2004).

The basic idea of the cost/fading map finds its origin in image processing, where we apply (convolute) a predefined blur filter $\underline{\mathcal{F}}$ to a given image $\underline{P}$. The filtered image $\underline{\Gamma}$ is given by:

$$\underline{\Gamma} = \underline{\mathcal{F}} * \underline{P} \tag{2}$$

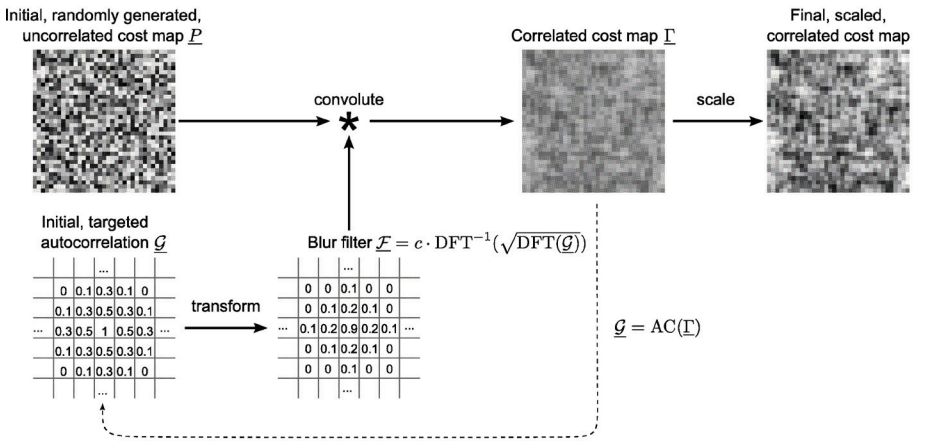
We must determine the filter $\underline{\mathcal{F}}$ in such a way that the filtered image $\underline{\Gamma}$ has a predefined autocorrelation matrix $\underline{\mathcal{G}}$.

Figure 2 illustrates the complete cost map generation process. The process starts with a given uncorrelated random cost map $\underline{P}$, where all values are uniformly and independently chosen from the interval $[-1,1]$. Dark fields represent small values close to 0. We also provide the targeted autocorrelation matrix $\underline{\mathcal{G}} = (g_{i,j})$, with entries:

$$g_{i,j} = \mathrm{e}^{-(|i|+|j|)} \tag{3}$$

From $\underline{\mathcal{G}}$ we construct the filter $\underline{\mathcal{F}} = (f_{i,j})$ using the 2-dimensional Discrete Fourier Transformation (DFT).

Figure 2. Generating the cost map $\underline{\Gamma}$



$$\underline{\mathcal{F}} = \sqrt{\tfrac{3}{uv}} \cdot \mathrm{DFT}^{-1}\left(\sqrt{\mathrm{DFT}(\underline{\mathcal{G}})}\right) \tag{4}$$

We show in the Appendix that $\underline{\mathcal{F}}$ exhibits the required property. The reader finds there the complete derivation of this formula and the definitions of the necessary functions. We want to point out that for reasons of convenience, Figure 2 shows $\underline{\mathcal{F}}$ and $\underline{\mathcal{G}}$ with index (0, 0) in the center. The derivation in the Appendix assumes the index (0, 0) is in the upper left corner due to simpler notation. Both approaches are equivalent, though.

Finally, we apply $\underline{\mathcal{F}}$ to $\underline{P}$ and obtain the correlated cost map $\underline{\Gamma}$. Its entries are within the range $\left[-\sum_{i,j} f_{i,j}, \sum_{i,j} f_{i,j}\right]$. As we require positive cost values as an input for the Floyd-Warshall algorithm (Cormen, Leiserson, Rivest, & Stein, 2001), we linearly scale the values of the resulting map $\underline{\Gamma}$ into the interval $[0,1]$. Again, darker fields in the matrix representations correspond to small values close to 0.

In the sequel, we will refer to the cost matrix $\underline{\Gamma}$ generated using this approach as *Arbitrary Cost Map* in contrast to *Constant Cost Map* where the cost matrix has the same value for all entries.

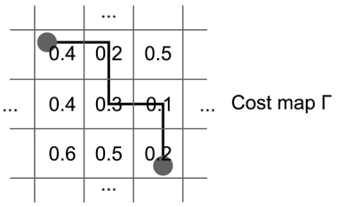
Problem Formulation

For a given set of *n* nodes $N_i(x_i, y_i)$ $(i = 1,\ldots,n)$ and a given number *m* of switches, the network deployment cost C_{Net} (*cf.* Box 1) is a function of the link cost C. As stated before, in an embedded environment the link cost depends on the position where the links are deployed.

Our objective is thus to minimize C by positioning *m* switches $S_j(\alpha_j, \beta_j)$ $(j = 1,\ldots,m)$, connecting the nodes to the switches, and connecting the switches between them. For an instance of this problem, we define the following matrices:

- $\underline{\Gamma} = (\gamma_{i,j})$ is a $u \times v$ matrix representing the discrete cost map where $\gamma_{i,j}$ is a non-negative real value in the interval $[0,1]$ specifying the cost of deploying a link segment at position (i, j).
- $\underline{\Omega} = (\omega_{I,J})$ is a $uv \times uv$ matrix representing the minimum link cost required to connect a device at position $I(x_I, y_I)$ to another device at position $J(x_J, y_J)$. The elements $\omega_{I,J}$ are non-negative real values derived from the matrix $\underline{\Gamma}$ by applying the Floyd-Warshall algorithm (Cormen et

Figure 3. Path calculation based on a discrete cost map



al., 2001). It should be noted that the links can only be deployed along either the *x-axis* or the *y-axis*. Figure 3 shows an example of a link connecting the top left corner with the bottom right corner. The values in the different fields represent the cost $\gamma_{i,j}$ of deploying a full link segment at position (i,j). At the endpoints of the link we assume that only half of a link segment is required, which leads to half of the cost there. Two endpoints in the same field require no link. The cost of the considered link is the sum of the incurred cost of the contiguous fields where it is deployed. Consequently, the link plotted in Figure 3 has a cost of $(1/2 \cdot 0.4) + 0.2 + 0.3 + 0.1 + (1/2 \cdot 0.2) = 0.9$.

When considering a constant cost map $\underline{\Gamma} = (c > 0)_{i,j}$ the minimum cost to connect a device at position $I(x_I, y_I)$ to another device at position $J(x_J, y_J)$ reduces to:

$$\omega_{I,J} = c \cdot d_{I,J} = c \cdot \left(\mid x_I - x_J \mid + \mid y_I - y_J \mid \right), \quad (5)$$

where $d_{I,J}$ is the Manhattan distance between I and J (Black, 2006). In this case, the link plotted in Figure 3 would have a cost of $4c$. Consequently, deploying a link between two endpoints assuming an arbitrary cost map may have higher or lower cost values than the physical length of this same link computed using the Manhattan distance and assuming a constant cost map.

- $\underline{\Psi} = (\psi_{i,j})$ is an $n \times m$ matrix representing the connectivity between the nodes and the switches where $\psi_{i,j}$ is a binary variable specifying the presence or the absence of a link between node N_i and switch S_j.

$$\psi_{i,j} = \begin{cases} 1 & \text{if } N_i \text{ is connected to } S_j, \\ 0 & \text{otherwise.} \end{cases} \quad (6)$$

- $\underline{\Phi} = (\phi_{i,j})$ is an $m \times m$ matrix representing the connectivity between the switches where $\phi_{i,j}$ is a binary variable specifying the presence or the absence of a link between switch S_i and switch S_j. Note that in case of full duplex links, the matrix $\underline{\Phi}$ is symmetric $(\phi_{i,j} = \phi_{j,i})$.

$$\phi_{i,j} = \begin{cases} 1 & \text{if } S_i \text{ is connected to } S_j, \\ 0 & \text{otherwise.} \end{cases} \quad (7)$$

Depending on the positions of the switches, we calculate the cost of the individual links. For this purpose, we also define:

- $\underline{\Delta} = (\delta_{i,j})$ is an $n \times m$ matrix derived from the matrix $\underline{\Omega}$. It contains the costs of the links needed to connect the nodes and the switches where $\delta_{i,j}$ is a non-negative real value equal to the minimal link cost between node N_i and switch S_j.
- $\underline{\Lambda} = (\lambda_{i,j})$ is an $m \times m$ matrix also derived from the matrix $\underline{\Omega}$. It contains the costs of

the links needed to interconnect the switches where $\lambda_{i,j}$ is a non-negative real value equal to the minimal link cost between the switches S_i and S_j.

The link cost C results in:

$$C = \sum_{\substack{i=1,..,n \\ j=1,..,m}} \psi_{i,j} \cdot \delta_{i,j} + \sum_{\substack{i=1,..,m-1 \\ j=i+1,..,m}} \phi_{i,j} \cdot \lambda_{i,j} \quad (8)$$

The Simulated Annealing Algorithm

In this section, we present our Simulated Annealing (SA) algorithm that solves the complex problem formulated in the previous section. Kirkpatrick et al. (1983) introduced SA as a randomized local search algorithm to solve combinatorial optimization problems (Kirkpatrick, Gelatt, Jr., & Vecchi, 1983). Today, it is commonly known that SA is able to find solutions with high quality in a short time (Glover & Laguna, 1997; Aarts & Lenstra, 2003; Gonzales, 2007). The basic idea of SA is to improve a given initial solution iteratively by applying defined rearrangement operations. We describe the complete SA algorithm in Algorithm 1.

As an initial solution S_0 of the SA algorithm, we place the switches randomly within a pre-defined solution space, connect each node to its nearest switch, i.e., the switch that requires the lowest link cost, and interconnect the switches using an MST algorithm (Kruskal, 1956; Prim, 1957). For given switch positions, such a link scheme guarantees the minimal link cost. Let C_0 be the cost of this initial/current solution.

By applying defined rearrangement operations to the current solution, also called *perturbations*, a new solution S_x with cost C_x is obtained (Line 15 of Algorithm 1). In the case of integrated switches, the solution space is restricted to the node positions, and the perturbation moves a randomly selected switch from its current node position to the position of another selected node randomly. In contrast to this, in the case of self-contained switches, the perturbation moves a randomly selected switch to any other position on the cost map. In both cases, the nodes are connected to their nearest switch, while the switches are interconnected using the MST algorithm.

New solutions with lower cost than the current solution are accepted automatically (Line 19 of Algorithm 1). In order to avoid local minima, solutions with higher cost than the current solution are accepted with a probability determined by a system control temperature T. However, the probability that these more expensive solutions are chosen decreases as the algorithm progresses in time to simulate the *cooling* process associated with annealing. This probability is based on a negative exponential function and is inversely proportional to the difference between the cost of the current solution and the cost of the new solution (Line 29 of Algorithm 1). If the cost of the new solution is equal to the cost of the current solution, one can randomly choose to accept the new solution as the current solution or rejected it (Line 26 of Algorithm 1).

By iteratively applying the rearrangement operations and appropriately updating the current solution, the SA algorithm searches the solution space for a solution with minimal cost. The best solution obtained by the SA algorithm during its run is stored in S_{best} with corresponding cost C_{best}.

It is to be noted that we delete unnecessary switches in a post-processing step. A switch is unnecessary, if its non-existence does not affect the link cost. These are switches to which only two full duplex links are connected. Normally, it is possible to delete one or more switches if the number of switches is close to the number of nodes ($m \simeq n$).

Related Reference Algorithms

In the previous section, we presented the SA algorithm that is a powerful stochastic search method (Vidal, 1993). In this section, we introduce exact algorithms that return the optimal solution for particular networks. However, due

to the complexity of the problem, these algorithms are only able to solve *small* instances of the problem with limited number of nodes and switches. These optimal solutions, when computable, will serve as a lower bound for the SA's solutions and will help us to evaluate their quality. In the general case of self-contained switches and arbitrary cost map, to the best of our knowledge, there is not one algorithm that finds the optimal solution in an acceptable running time.

Integrated Switches

Limited Number of Switches. In this case, we are able to find the optimal positions of a given number of switches that guarantees the minimal link deployment cost when these switches can only be placed at the same positions as the nodes. This algorithm is based on a MILP formulation that can be solved by means of linear solvers such as Scip (Zuse Institute Berlin (ZIB), 2009). The MILP problem is formulated as:

Given

- An $n \times n$ matrix $\underline{\Theta} = (\theta_{i,j})$ containing the cost of the links needed to interconnect the nodes where $\theta_{i,j}$ is a non-negative real value equal to the minimal link cost between the node $N_i(x_i, y_i)$ and the node $N_j(x_j, y_j)$ ($i, j = 1,\ldots,n$). This matrix is derived from $\underline{\Omega}$ and can be obtained for arbitrary cost map and constant cost map as described in Section *Cost Map*.
- The number m of the switches to be deployed.

Variables

- The binary variables $\psi_{i,j}$ (*cf.*Equation (6)) representing the connectivity between the nodes and the switches are slightly modified and are now defined as: $i, j = 1,\ldots,n$

$$\psi_{i,j} = \begin{cases} 1 & \text{if } N_i \text{ is connected to a switch at } N_j, \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

- The binary variables o_i ($i = 1,\ldots,n$):

$$o_i = \begin{cases} 1 & \text{if a switch is built-in } N_i(x_i, y_i), \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

- The binary variables $\phi_{i,j}$ (*cf.*Equation (7)) representing the connectivity between the switches are slightly modified and are now defined as: $i, j = 1,\ldots,n$ (see Box 2.)
- The binary variables $\xi_{k,l}^{i,j}$ ($i = 1,\ldots,n-1,\ j = i+1,\ldots,n,\ k,l = 1,\ldots,n$). $\xi_{k,l}^{i,j} = 1$, if the link between the node N_k and the node N_l is used when sending data from a switch placed at node N_i to another switch placed at node N_j. $\xi_{k,l}^{i,j} = 0$, otherwise.

Box 2.

$$\phi_{i,j} = \begin{cases} 1 & \text{if a switch at } N_i \text{ is connected} \\ & \text{to a switch at } N_j, \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

Algorithm 1. SA algorithm

```
1: define
2:     mIter  ▷ maximum number of consecutive iterations
       without any cost improvement
3:     mImpr  ▷ maximum number of cost improvements
4:     mAtmp  ▷ maximum number of attempts
5:     T  ▷ initial temperature
6:     p  ▷ cooling factor
7: end define
8: procedure SA ( N_i(x_i, y_i) , m , Ω )
9:     Construct an initial solution S_0 ; C_0 ← Cost( S_0 )
10:  (S_best, C_best) ← (S_0, C_0)
11:   iter ← 0
12:   while iter < mIter do
13:       impr ← 0 ; atmp ← 0
14:       while (impr < mImpr) ∧ (atmp < mAtmp) do
15:           S_x ← Perturbation( S_0 ) ; C_x ← Cost( S_x )
16:           atmp++
17:           Generate a random number r ∈ [0,1)
18:           if C_x < C_0 then ▷ improved solution
19:               (S_0, C_0) ← (S_x, C_x)
20:               impr++
21:               if C_x < C_best then
22:                   (S_best, C_best) ← (S_x, C_x)
23:                   iter ← 0
24:               end if
25:           else if C_x = C_0 then ▷ equivalent solution
26:               if r < 0.5 then
27:                   (S_0, C_0) ← (S_x, C_x)
28:               end if
29:           else if r < e^{-(C_x - C_0)/(C_0 T)} then ▷ worse solution
30:               (S_0, C_0) ← (S_x, C_x)
31:           end if
32:       end while
33:       if atmp = mAtmp then ▷ maximum number of attempts reached
34:           iter++
35:       end if
36:       T ← pT
37:   end while
38:   return ( S_best , C_best )
39: end procedure
```

Constraints

- We have to choose at most m nodes where to place the switches.

$$\sum_{i=1}^{n} o_i \leq m \tag{12}$$

- The interconnection between the switches is based on the principle of *Flow Conservation* (Pióro & Mehdi, 2004; Atallah & Blanton, 2009) commonly used in the field of *Network Dimensioning*. The basic idea is to assume that every switch has to send some data to all the other switches. For a given data flow between a source switch and a destination switch, the flow conservation constraint states that the data flow is only leaving the source switch, and is only entering the destination switch. At all the other switches, the amount of data entering the switch is equal to the amount of data leaving the same switch. However, in our formulation, we do not know the exact position of the switches but we know their possible locations (at the nodes). Hence, the flow conservation constraint assumes that some data has to be sent between two locations if both locations contain a switch.

$$\forall i = 1,\ldots,n-1,\ \forall j = i+1,\ldots,n,\ \forall k = 1,\ldots,n$$

(See Box 3)

- The connectivity between the switches must contain all the links needed to connect any two switches.

$$\forall i = 1,\ldots,n-1,\ \forall j = i+1,\ldots,n,$$
$$\forall k = 1,\ldots,n,\ \forall l = 1,\ldots,n$$
$$\phi_{k,l} \geq \xi_{k,l}^{i,j} \tag{14}$$

- The connectivity between the switches is symmetric.

$$\forall k = 1,\ldots,n-1,\ \forall l = k+1,\ldots,n\ \phi_{k,l} = \phi_{l,k} \tag{15}$$

- Every node has to be connected to exactly one position of the solution space.

$$\forall i = 1,\ldots,n$$
$$\sum_{j=1}^{n} \psi_{i,j} = 1 \tag{16}$$

- However, this position is only valid, if it contains a switch. $\forall i = 1,\ldots,n$ and $\forall j = 1,\ldots,n$

$$\psi_{i,j} \leq o_j \tag{17}$$

Objective

The objective defined in Equation (8) can be expressed as:

$$C = \sum_{\substack{i=1,\ldots,n \\ j=1,\ldots,n}} \psi_{i,j} \cdot \theta_{i,j} + \sum_{\substack{i=1,\ldots,n-1 \\ j=i+1,\ldots,n}} \phi_{i,j} \cdot \theta_{i,j} \tag{18}$$

Unlimited Number of Switches. It is obvious that, for small number of switches, increasing the number of switches will reduce the total

Box 3.

$$\sum_{\substack{l=1,\ldots,n \\ l \neq k}} \xi_{k,l}^{i,j} - \sum_{\substack{l=1,\ldots,n \\ l \neq k}} \xi_{l,k}^{i,j} = \begin{cases} = 0 & \text{if } k \neq i \text{ and } k \neq j, \\ \geq o_i + o_j - 1 & \text{if } k = i, \\ \leq -o_i - o_j + 1 & \text{if } k = j. \end{cases} \tag{13}$$

link cost. The lower bound on the total link cost can be obtained by the MST algorithm. The MST connects all the nodes of the embedded environment with a tree-like topology having the lowest possible cost. Once the MST is computed, every node that is connected to at least two other nodes is assumed to have a built-in switch. As the number of switches is only determined afterwards, this algorithm is not suited for networks with a limited number of switches. In the worst case, the maximum number of required switches is equal to the number of nodes minus two.

There are two commonly used algorithms to compute the MST: *Prim's* algorithm (Prim, 1957) and *Kruskal's* algorithm (Kruskal, 1956). Both are greedy algorithms that run in polynomial time.

Self-Contained Switches

Limited Number of Switches. When the switches can be placed anywhere in the embedded environment space, the solution space for the position of the switches becomes large. Consequently, if we use the same MILP formulation as in the case of integrated switches, the number of variables and constraints becomes extremely large, which makes it impossible to get any solution in acceptable running time even for small problem instances. However, in the special case of a constant cost map, the minimum link cost to connect two nodes/switches at different positions is proportional to the Manhattan distance between these two positions. The Manhattan distance is not a linear function but can be linearized by means of additional variables and constraints. The linearized model of the problem is formulated as (Sommer & Doumith, 2008).

Given

- The position of n nodes $N_i(x_i, y_i), i = 1,\ldots,n$.
- The number m of the switches to be deployed.
- The parameter $\aleph$ defined as follows:

$$\aleph \gg 1 \tag{19}$$

- The constant cost c of the fields in the cost map.

Variables

- The binary variables $\psi_{i,j}$ (*cf.*Equation (6)) representing the connectivity between the nodes and the switches.
- The non-negative real variables $\delta_{i,j}^X$ and $\delta_{i,j}^Y$ ($i = 1,\ldots,n,\ j = 1,\ldots,m$) containing the distances between the nodes and the switches along the *x-axis* and the *y-axis*, respectively.
- The non-negative real variables $\delta_{i,j}$ ($i = 1,\ldots,n,\ j = 1,\ldots,m$) containing the Manhattan distances between the nodes and the switches.
- The binary variables $\phi_{i,j}$ (*cf.*Equation (7)) representing the connectivity between the switches.
- The non-negative real variables $\mu_{i,j}$ ($i = 1,\ldots,n,\ j = 1,\ldots,m$) equal to the product of the two variables $\delta_{i,j} \cdot \psi_{i,j}$.
- The abscissae α_j ($\alpha_j \in \mathbb{R}^+$) and the ordinates β_j ($\beta_j \in \mathbb{R}^+$) of the switches ($j = 1,\ldots,m$).
- The binary variables $\xi_{j,k}^i$ ($i = 2,\ldots,m,\ j,k = 1,\ldots,m$). $\xi_{j,k}^i = 1$, if the link between the switch S_j and the switch S_k is used when sending data from the switch S_1 to the switch S_i. $\xi_{j,k}^i = 0$, otherwise.
- The non-negative real variables $\lambda_{i,j}^X$ and $\lambda_{i,j}^Y$ ($i = 1,\ldots,m-1,\ j = i+1,\ldots,m$) containing the distances between the switches along the *x-axis* and the *y-axis*, respectively.

- The non-negative real variables $\lambda_{i,j}$ ($i = 1,\ldots,m-1, j = i+1,\ldots,m$) containing the Manhattan distances between the switches.
- The non-negative real variables $\nu_{i,j}$ ($i = 1,\ldots,m-1, j = i+1,\ldots,m$) equal to the product of the two variables $\lambda_{i,j} \cdot \phi_{i,j}$.

Constraints

The interconnection between the switches is also based on the principle of *Flow Conservation*. In contrast to the previous section, the switches can be placed anywhere in the environment space. In this case, it is obvious that if all switches are able to exchange data with switch S_1, then any two switches can exchange data between them. Consequently and without loss of generality, we can reduce the number of constraints to only insure that node S_1 is able to send data to all the other nodes.

$$\forall i = 2,\ldots,m\ ,\ \forall k = 1,\ldots,m$$

$$\sum_{\substack{j=1,\ldots,m \\ j\neq k}} \xi_{k,j}^{i} - \sum_{\substack{j=1,\ldots,m \\ j\neq k}} \xi_{j,k}^{i} = \begin{cases} = 0 & \text{if } k \neq 1 \text{ and } k \neq i, \\ +1 & \text{if } k = 1, \\ -1 & \text{if } k = i. \end{cases} \tag{20}$$

The connectivity between the switches must contain all the links needed to connect any two switches. $\forall i = 2,\ldots,m$,

$$\forall j = 1,\ldots,m, \forall k = 1,\ldots,m$$

$$\phi_{j,k} \geq \xi_{j,k}^{i} \tag{21}$$

The connectivity between the switches is symmetric.

$$\forall j = 1,\ldots,m-1, \forall k = j+1,\ldots,m$$

$$\phi_{j,k} = \phi_{k,j} \tag{22}$$

The distance between the switches along the *x-axis* is given by:

$$\forall j = 1,\ldots,m-1, \forall k = j+1,\ldots,m$$
$$\lambda_{j,k}^{X} = \mid \alpha_j - \alpha_k \mid \cdot c = \max\left\{(\alpha_j - \alpha_k)\cdot c,\ (\alpha_k - \alpha_j)\cdot c\right\} \tag{23}$$

In linear form, this equation is equivalent to:

$$\forall j = 1,\ldots,m-1, \forall k = j+1,\ldots,m$$

$$\begin{aligned} \lambda_{j,k}^{X} &\geq (\alpha_j - \alpha_k)\cdot c \\ \lambda_{j,k}^{X} &\geq (\alpha_k - \alpha_j)\cdot c \end{aligned} \tag{24}$$

- Similarly, the distance between the switches along the *y-axis* is given by:

$$\forall j = 1,\ldots,m-1, \forall k = j+1,\ldots,m$$

$$\begin{aligned} \lambda_{j,k}^{Y} &\geq (\beta_j - \beta_k)\cdot c \\ \lambda_{j,k}^{Y} &\geq (\beta_k - \beta_j)\cdot c \end{aligned} \tag{25}$$

- Consequently, the Manhattan distances of the links between the switches is equal to:

$$\forall j = 1,\ldots,m-1, \forall k = j+1,\ldots,m$$

$$\lambda_{j,k} = \lambda_{j,k}^{X} + \lambda_{j,k}^{Y} \tag{26}$$

- The product $\mu_{j,k}$ is then given by:

$$\forall j = 1,\ldots,m-1, \forall k = j+1,\ldots,m$$

$$\begin{aligned} \mu_{j,k} &= \phi_{j,k}\cdot\lambda_{j,k} \\ &= \min\left\{\lambda_{j,k}^{X} + \lambda_{j,k}^{Y}, \aleph\cdot\phi_{j,k}\right\} \end{aligned} \tag{27}$$

In linear form, this equation is equivalent to: $\forall j = 1,\ldots,m-1, \forall k = j+1,\ldots,m$

$$\begin{aligned} \mu_{j,k} &\leq \lambda_{j,k}^{X} + \lambda_{j,k}^{Y} \\ \mu_{j,k} &\leq \aleph\cdot\phi_{j,k} \\ \mu_{j,k} &\geq \lambda_{j,k}^{X} + \lambda_{j,k}^{Y} + \aleph\cdot(\phi_{j,k} - 1) \end{aligned} \tag{28}$$

- Every node has to be connected to exactly one switch. $\forall i = 1,\ldots,n$

$$\sum_{j=1}^{m} \psi_{i,j} = 1 \tag{29}$$

- The linear formulation of the distance along the *x-axis* between the nodes and the switches is given by:

$$\forall i = 1,\ldots,n, \forall j = 1,\ldots,m$$

$$\begin{aligned} \delta_{i,j}^{X} &\geq (x_i - \alpha_j)\cdot c \\ \delta_{i,j}^{X} &\geq (\alpha_j - x_i)\cdot c \end{aligned} \tag{30}$$

- Similarly, the linear formulation of the distance along the *y-axis* between the nodes and the switches is given by:

$$\forall i = 1,\ldots,n, \forall j = 1,\ldots,m$$

$$\begin{aligned} \delta_{i,j}^{Y} &\geq (y_i - \beta_j)\cdot c \\ \delta_{i,j}^{Y} &\geq (\beta_j - y_i)\cdot c \end{aligned} \tag{31}$$

- Consequently, the Manhattan distances of the links between the nodes and the switches is equal to:

$$\forall i = 1,\ldots,n, \forall j = 1,\ldots,m$$

$$\delta_{i,j} = \delta_{i,j}^{X} + \delta_{i,j}^{Y} \tag{32}$$

- The product $\nu_{i,j}$ is then given by:

$$\forall i = 1,\ldots,n, \forall j = 1,\ldots,m$$

$$\begin{aligned} \nu_{i,j} &= \psi_{i,j}\cdot \delta_{i,j} \\ &= \min\left\{\delta_{i,j}^{X} + \delta_{i,j}^{Y}, \aleph\cdot \psi_{i,j}\right\} \end{aligned} \tag{33}$$

- In linear form, this equation is equivalent to:

$$\forall i = 1,\ldots,n, \forall j = 1,\ldots,m$$

$$\begin{aligned} \nu_{i,j} &\leq \delta_{i,j}^{X} + \delta_{i,j}^{Y} \\ \nu_{i,j} &\leq \aleph \cdot \psi_{i,j} \\ \nu_{i,j} &\geq \delta_{i,j}^{X} + \delta_{i,j}^{Y} + \aleph\cdot(\psi_{i,j} - 1) \end{aligned} \tag{34}$$

Objective

The objective as defined in Equation (8) can now be expressed as

$$C = \sum_{\substack{i=1,..,n \\ j=1,..,m}} \nu_{i,j} + \sum_{\substack{i=1,..,m-1 \\ j=i+1,..,m}} \mu_{i,j} \tag{35}$$

As stated before, the MILP formulation for a given number of switches and a constant cost map can be solved by means of linear solvers such as Scip (Zuse Institute Berlin (ZIB), 2009).

Unlimited Number of Switches. For small number of switches, increasing the number of switches will reduce the total link cost. However, this cost cannot be reduced indefinitely, and the lower bound on the total link cost can be obtained in this case by solving the ST problem. The classical ST problem consists in connecting all the nodes of the embedded environment with a tree-like topology that has the lowest possible cost while adding additional points, called *Steiner points* (Du & Hu, 2008). These Steiner points as well as the nodes with two or more connections represent in our case the positions of the switches. As the number of switches is only determined afterwards, this algorithm is not suited for networks with a limited number of switches. In the worst case, the maximum number of required switches is equal to the number of nodes minus two.

As we mentioned before, the ST problem is NP-complete. In practice, it is solved by means of heuristics. In the special case of a constant cost map, the GeoSteiner package (Skiena, 2009) gives the optimal solution for rectilinear ST problems.

Performance Evaluation

We presented an SA algorithm to minimize the total link cost of an embedded Ethernet network. We also introduced algorithms that find the globally optimal solutions for *small* instances of the problem. In Figure 4, we classify these algorithms according to their use cases.

In this section, we evaluate the performance of the SA algorithm in terms of the quality of the obtained final solution as well as the running time. In order to find the optimal solution by means of the exact algorithms, which we discussed in the previous section, we have to consider problem instances with small number of nodes and/or switches. For this reason, we consider two different sets of 15 and 20 nodes, respectively. For each set of nodes, we consider two cost maps: a constant cost map and an arbitrary cost map. In both cases, the minimum cost to connect any two positions of the embedded environment is pre-computed beforehand and stored in the matrix $\underline{\Omega}$. We point out that all the discussed algorithms (SA, MILP for integrated switches, ST, and MST) make use of this matrix, and thus the pre-computation step does not affect the comparisons' solutions. In the case of the MILP with self-contained switches, and due to the large solution space, some elements of the matrix $\underline{\Omega}$ are computed online during the simulation as in Equations (24), (25), (30), and (31). The parameters of the SA algorithm are $\mathrm{mIter} = 500$, $\mathrm{mAtmp} = 500$, $\mathrm{mImpr} = 100$, $T = 1$ (initial temperature), and $p = 0.9$ (cooling factor).

In the case of a constant cost map and the integrated switches design, the SA algorithm finds in all cases the same solution as the MILP (the optimal solution). As shown in Table 1, the running time of the SA is always less than one minute while the running time of the MILP increases with the increasing number of nodes and switches and exceeds two weeks in some cases. For large number of switches (number of nodes minus two), the SA algorithm maintains its excellent performance and finds the same solution as the MST in a comparable running time.

By replacing the constant cost map with an arbitrary cost map with the same average cost of 0.5, all the algorithms exhibit an increased running time, except for the SA algorithm. As shown in Table 2, the performance of the SA algorithm has not changed. In fact, it still finds the globally optimal solutions within less than one minute.

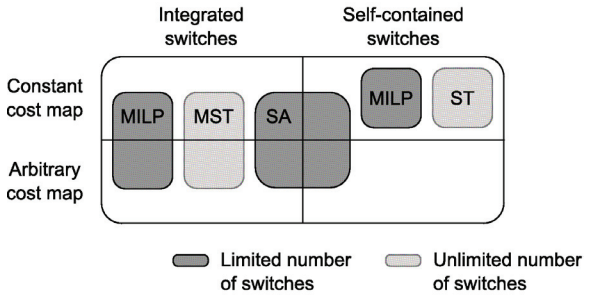
By considering the self-contained switches design, the solution space for the positions of the switches becomes large. In this case, the optimal link deployment cost can only be obtained for the constant cost map. For small number of switches, as shown in Table 3, the SA algorithm finds the same solutions as the MILP for self-contained switches in considerably reduced running time. Even for large number of switches (number of nodes minus two), the SA algorithm keeps its pace and finds optimal solutions, too.

Network Design Evaluation

In this section, we compare two network designs mentioned in Section *Problem Description*. For this purpose, we consider several cost maps. For each of them, we consider a set of 50 nodes and we distribute them uniformly on the map. As depicted in Figure 4, only the SA algorithm is able to find the (near-) optimal solutions in an acceptable amount of running time for such large networks. In order to ensure the best quality of the obtained solution, we make 25 independent runs for each case and record the lowest cost. Due to the complexity of the problem instances, we do not know the globally optimal solutions. However, the proposed SA algorithm has proven its excellent performance for small network scenarios and it is expected to keep such performance for larger problem instances. Therefore, we use the term *(near-) optimal* solutions to qualify the solution of the SA algorithm as there is no way to guarantee that the obtained result is optimal.

As expected, the link cost in the case of self-contained switches design is smaller than the link cost in the case of the integrated switches design. This is because the solution

Figure 4. Classification and application areas of the various algorithms



space of the latter design is included in the former one. However, in most cases the difference between the two designs is less than 5%. Only in the cases with unlimited number of switches ($m = n - 2$), the difference can exceed 10%. Moreover, by deploying additional switches, we can reduce the total link cost. However, as shown in Table 4 and 5, the higher the number of switches, the lower the total link cost reduction.

Finally, we evaluate the impact of the ambient conditions on the embedded network design problem. For this purpose, we consider an arbitrary cost map with an average value of approximately 0.5 and a constant cost map where all the fields are equal to $c = 0.5$. As shown in Table 4 and Table 5 (constant vs. arbitrary), the SA algorithm benefits from the map fields with low cost to interconnect the nodes and the switches. These solutions result mostly in lower link cost for the arbitrary cost map.

CONCLUSION AND OUTLOOK

During the last decade, apart from LAN installations, Ethernet became attractive for other application areas such as industrial control and avionics. In these areas, we have to consider additional constraints and ambient conditions, while reducing the deployment cost. In this paper, we proposed an SA algorithm that minimizes the total link cost of an embedded Ethernet network. We modeled the ambient conditions by means of a cost map and used it for deriving the link cost.

The proposed SA algorithm is applicable for two network designs that are relevant in embedded networks: integrated switches and self-contained switches. We evaluated the performance of the SA algorithm by comparing its solutions with the optimal solutions for small networks or simplified network designs. We have shown that, in our comparisons, the SA algorithm achieves optimal solutions in short running time.

Part of our ongoing work is to extend the SA algorithm. We are working on further perturbations. For example, we decrease the size of the space where a switch can move when it is subject to a perturbation. The aim is to approximate a local optimum carefully. Besides, we are extending the SA algorithm in order to consider ducts. These ducts are used to protect the cables that pass through and can carry one or more links. In this case, we also have to consider junction points where a link joins/leaves a duct. The duct cost depends on the number of links. With this enhancement, we will further reduce the link cost.

Until now, we focused on the physical topology. We did not take into account higher layer constraints. In the future, we will extend the SA algorithm to take into account a given traffic demand matrix, while optimizing the topology. The resulting topology should fulfill the traffic demands at a low cost.

Table 1. Integrated switches: SA vs. optimal solutions using a constant cost map (size 50×50)

# switches		15 nodes		20 nodes	
		C	t [min]	C	t [min]
5	MILP	112.0	$\simeq 52$	115.5	$\simeq 412$
	SA	112.0	< 1	115.5	< 1
10	MILP	93.5	$\simeq 70$	99.0	$> 30d$
	SA	93.5	< 1	99.0	< 1
15	MILP	93.5	$\simeq 5$	98.0	$> 30d$
	SA	93.5[1]	< 1	98.0[2]	< 1
unlimited $\max = n - 2$	MST	93.5	< 1	98.0	< 1
	SA	93.5[1]	< 1	98.0[2]	< 1

Table 2. Integrated switches: SA vs. optimal solutions using an arbitrary cost map (size 50×50)

# switches		15 nodes		20 nodes	
		C	t [min]	C	t [min]
5	MILP	99.0	$\simeq 68$	104.4	$\simeq 471$
	SA	99.0	< 1	104.4	< 1
10	MILP	83.5	$\simeq 40$	90.2	$> 30d$
	SA	83.5	< 1	90.2	< 1
15	MILP	83.5	$\simeq 5$	88.9	$> 30d$
	SA	83.5[1]	< 1	88.9[3]	< 1
unlimited $\max = n - 2$	MST	83.5	< 1	88.9	< 1
	SA	83.5[1]	< 1	88.9[3]	< 1

Table 3. Self-contained switches: SA vs. optimal solutions using a constant cost map (size 50×50)

# switches		15 nodes		20 nodes	
		C	t [min]	C	t [min]
2	MILP	139.5	< 1	164.5	< 1
	SA	139.5	< 1	164.5	< 1
3	MILP	122.0	$\simeq 4$	132.0	$\simeq 17$
	SA	122.0	< 1	132.0	< 1
5	MILP	107.0	> 7*d*	111.5	> 7*d*
	SA	107.0	< 1	111.5	< 1
unlimited $\max = n-2$	ST	83.0	< 1	83.0	< 1
	SA	83.0	< 1	83.0[4]	< 1

Table 4. Comparison of networks with $n = 50$ nodes, a large number of integrated switches, and different cost maps

Map size	m = 10		m = 25		$m = n - 2$ [5]	
	constant	arbitrary	constant	arbitrary	constant	arbitrary
50×50	221.0	203.6	167.0	157.9	158.0	150.3
75×75	328.5	298.5	243.5	227.5	230.0	215.9
100×100	401.5	345.7	310.5	274.9	294.0	262.4

Table 5. Comparison of networks with $n = 50$ nodes, a large number of self-contained switches, and different cost maps

Map size	m = 10		m = 25		$m = n - 2$ [6]	
	constant	arbitrary	constant	arbitrary	constant	arbitrary
50×50	216.5	195.5	163.5	151.5	141.5	133.0
75×75	316.5	286.7	234.0	219.1	207.0	197.3
100×100	392.0	339.4	297.0	263.1	258.5	237.1

[1] Post-Processing switch deletion: 10 switches needed
[2] Post-Processing switch deletion: 12 switches needed
[3] Post-Processing switch deletion: 14 switches needed
[4] Post-Processing switch deletion: 17 switches needed
[5] Post-Processing switch deletion: 38 switches needed
[6] Post-Processing switch deletion: 47 switches needed

ACKNOWLEDGMENT

The authors would like to thank Matthias Kaschub for many fruitful discussions and Quentin Duval for his contribution to the implementation of the SA algorithm and the cost map generator.

REFERENCES

Aarts, E. L., & Lenstra, J. K. (2003). *Local search in combinatorial optimization*. Princeton, NJ: Princeton University Press.

Ahonen, T., Sigüenza-Tortosa, D. A., Bin, H., & Nurmi, J. (2004). *Topology optimization for application-specific networks-on-chip*. In *Proceedings of the 2004 international workshop on System Level Interconnect Prediction (SLIP '04)* (pp. 53-60). New York: ACM.

ARINC 664. (2003). *Aircraft Data Network, Part 7: Deterministic Networks* (Computer software manual).

Atallah, M. J., & Blanton, M. (2009). *Algorithms and Theory of Computation Handbook* (*Vol. 2*). Boca Raton, FL: CRC Press LLC.

Black, P. E. (2006, May). *Manhattan distance.* Retrieved from http://www.itl.nist.gov/div897/sqg/dads/

Cai, X., & Giannakis, G. (2003, November). A two-dimensional channel simulation model for shadowing processes. *IEEE Transactions on Vehicular Technology*, *52*(6), 1558–1567. doi:10.1109/TVT.2003.819627

Chardaire, P., & Lutton, J. L. (1993). In Vidal, R. V. (Ed.), *Applied Simulated Annealing* (pp. 173–199). New York: Springer Verlag.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2001). *Introduction to Algorithms* (2nd ed.). Cambridge, MA: MIT Press.

Decotignie, J.-D. (2005, June). Ethernet-based real-time and industrial communications. In *Proceedings of the IEEE*, Neuchatel, Switzerland (Vol. 93, p. 1102-1117). Washington, DC: IEEE.

Du, D., & Hu, X. (2008). *Steiner Tree Problems in Computer Communication Networks*. New York: World Scientific. doi:10.1142/9789812791450

Felser, M. (2005, June). Real-Time Ethernet – Industry Prospective. *Proceedings of the IEEE*, *93*(6), 1118–1129. doi:10.1109/JPROC.2005.849720

Forkel, I., Schinnenburg, M., & Ang, M. (2004, September). Generation of Two-Dimensional Correlated Shadowing for Mobile Radio Network Simulation. In *Proceedings of the 7th International Symposium on Wireless Personal Multimedia Communications (WPMC)*, Abano Terme (Padova), Italy (p. 5).

Garey, M. R., Graham, R. L., & Johnson, D. (1977, June). The complexity of computing Steiner minimal trees. *SIAM Journal on Applied Mathematics*, *31*(4), 835–859. doi:10.1137/0132072

Glover, F., & Laguna, M. (1997). *Tabu Search*. Dordrecht, The Netherlands: Kluwer Academic Publisher.

Gonzales, T. F. (Ed.). (2007). *Handbook of Approximation Algorithms and Metaheuristics*. Boca Raton, FL: Chapman & Hall CRC.

Harmatos, J., Jüttner, A., & Szentesi, A. (1999, September). Cost-based UMTS Transport Network Topology Optimisation. In *Proceedings of the International Conference on Computer Communication (ICCC'99),* Tokyo, Japan.

Kirkpatrick, S., Gelatt, C. D. Jr, & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*, 671–680. doi:10.1126/science.220.4598.671

Kruskal, J. B. (1956, February). On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. *Proceedings of the American Mathematical Society*, *7*(1), 48–50. doi:10.1090/S0002-9939-1956-0078686-7

Mukherjee, B., Banerjee, D., Ramamurthy, S., & Mukherjee, A. (1996). Some principles for designing a wide-area WDM optical network. [TON]. *IEEE/ACM Transactions on Networking*, *4*(5), 684–696. doi:10.1109/90.541317

Patzold, M., & Nguyen, V. (2004, September). A spatial simulation model for shadow fading processes in mobile radio channels. In *Proceedings of the 15th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2004)* (Vol. 3, pp. 1832-1838).

Pióro, M., & Mehdi, D. (2004). *Routing, Flow, and Capacity Design in Communication and Computer Networks*. San Francisco: Morgan Kaufmann Publishers.

Prim, R. (1957). Shortest connection networks and some generalizations. *The Bell System Technical Journal*, *36*, 1389–1401.

Rahmani, M., Hillebrand, J., Hintermaier, W., Bogenberger, R., & Steinbach, E. (2007, May). A Novel Network Architecture for In-Vehicle Audio and Video Communication. In *Proceedings of the 2nd IEEE/IFIP International Workshop on Broadband Convergence Networks (BcN '07)* (pp. 1-12).

Rahmani, M., Steffen, R., Tappayuthpijarn, K., Steinbach, E., & Giordano, G. (2008, February). Performance Analysis of Different Network Topologies for In-vehicle Audio and Video Communication. In *Proceedings of the 4th International Telecommunication Networking Workshop on QoS in Multiservice IP Networks* (pp. 179-184).

Rahmani, M., Tappayuthpijarn, K., Krebs, B., Steinbach, E., & Bogenberger, R. (2009). Traffic Shaping for Resource-Efficient In-Vehicle Communication. *IEEE Transactions on Industrial Informatics*.

Skiena, S. (2009). *GeoSteiner: Software for Computing Steiner Trees*.

Sommer, J., & Doumith, E. A. (2008, July). Topology Optimization of In-vehicle Multimedia Communication Systems. In *Proceedings of the First Annual International Symposium on Vehicular Computing Systems (ISVCS),* Dublin, Ireland.

Sommer, J., Gunreben, S., Feller, F., Köhn, M., Mifdaoui, A., Saß, D., et al. (2010). Ethernet – a survey on its fields of application. *IEEE Communications Surveys & Tutorials, 12*(2).

Vidal, R. V. (Ed.). (1993). *Applied Simulated Annealing* (*Vol. 1*). New York: Springer Verlag.

Wakileh, J., & Pahwa, A. (1996, November). Distribution system design optimization for cold load pickup. *IEEE Transactions on Power Systems*, *11*(4), 1879–1884. doi:10.1109/59.544658

Zuse Institute Berlin (ZIB). (2009). *SCIP: Solving Constraint Integer Programs*.

Jörg Sommer received his Diploma degree in information technology from the Baden-Württemberg Cooperative State University, Heidenheim, Germany, and his Master degree in computer science from the University of Ulm, Germany, in 2002 and 2004 respectively. Since 2005, he has been with the Institute of Communication Networks and Computer Engineering (IKR) at the University of Stuttgart, Germany. His research interests include performance evaluation of communication networks as well as topology design and optimization problems. He is a member of the German Gesellschaft für Informatik (Computer Science Society).

Elias A. Doumith is Assistant Professor in the Department of Networks and Computer Science at TELECOM ParisTech, France. He received his M.Sc. degree (2003) and Ph.D. degree (2007) from the Ecole Nationale Supérieure des Télécommunications, France. Between 2007 and 2009, he worked as Junior Research Engineer at the Institute of Communication Networks and Computer Engineering (IKR) at the University of Stuttgart, Germany. His domain of interest covers network planning and traffic engineering for networks, ranging from access networks to core networks including embedded networks. His current research works deal with cloud computing, radio over fiber, monitoring in optical networks, and scalable optical network design.

Andreas Reifert received his degree as Dipl.-Inform. in Computer Science from the Julius-Maximilians-Universität Würzburg, Germany, in 2003. He then moved to the Institute of Communication Networks and Computer Systems (IKR) at the University of Stuttgart as a member of the research staff. There, he worked in various national and international projects focusing on peer-to-peer technologies and service architectures for future telecommunication infrastructures. Mr. Reifert is currently working towards his PhD in Electrical Engineering. He concentrates on service component placement in wide-area communication infrastructures, specializing in planning aspects and resource provisoning.

APPENDIX: PRINCIPLE OF AUTOCORRELATION MASK GENERATION

Before going into the detail of the construction of the Filter $\underline{\mathcal{F}}$, let us introduce some notations and definitions. Unless stated otherwise, all the matrices are two dimensional $u \times v$ matrices with indices in the range $I = [0, u-1] \times [0, v-1]$ (index (0, 0) corresponds to the upper left entry of the matrix). All following operations are cyclic in contrast to using 0-padded matrices. The infinitely wrapped-around matrix $\underline{A}^E = (a^E_{x,y})$ with infinite index ranges $E = [-\infty, \infty] \times [-\infty, \infty]$ is defined as:

$$a^E_{x,y} = a_{x \bmod u, y \bmod v} \qquad \forall (x,y) \in E \tag{36}$$

The point inverted matrix $\underline{A}^- = (a^-_{x,y})$ of a matrix $\underline{A} = (a_{x,y})$ is given by:

$$a^-_{x,y} = a_{-x \bmod u, -y \bmod v} \tag{37}$$

The cyclic convolution $\underline{C} = (c_{x,y}) = \underline{A} * \underline{B}$ of two matrices $\underline{A} = (a_{x,y})$ and $\underline{B} = (b_{x,y})$ of the same size is given by:

$$c_{x,y} = \sum_{(k,l) \in I} a_{k,l} \cdot b^E_{x-k, y-l} \qquad \forall (x,y) \in I \tag{38}$$

The cyclic autocorrelation $\underline{B} = (b_{x,y}) = \mathrm{AC}(\underline{A})$ of a matrix $\underline{A} = (a_{x,y})$ is given by:

$$b_{x,y} = \sum_{(k,l) \in I} \overline{a}_{k,l} \cdot a^E_{k+x, l+y} \qquad \forall (x,y) \in I \tag{39}$$

and is related to the cyclic convolution via $\mathrm{AC}(\underline{A}) = \overline{\underline{A}^-} * \underline{A}$. $\overline{\underline{A}^-}$ is the complex conjugate of $\underline{A}^-$. Another property of the autocorrelation function is that it is equal to the complex conjugate of its point inverted matrix ($\underline{B} = \overline{\underline{B}^-}$), and has a maximum at index (0, 0).

The 2-dimensional Discrete Fourier Transform $\underline{B} = (b_{x,y})$ of a matrix $\underline{A} = (a_{x,y})$, noted $\underline{B} = \mathrm{DFT}(\underline{A})$, and its inverse, noted $\underline{A} = DFT^{-1}(\underline{B})$, are matrices of the same size:

$$b_{x,y} = \sum_{(k,l) \in I} a_{k,l} \cdot \mathrm{e}^{-2\pi i \left(\frac{x \cdot k}{u} + \frac{y \cdot l}{v} \right)} \qquad \forall (x,y) \in I \tag{40}$$

$$a_{k,l} = \frac{1}{uv} \sum_{(x,y) \in I} b_{x,y} \cdot \mathrm{e}^{2\pi i \left(\frac{x \cdot k}{u} + \frac{y \cdot l}{v} \right)} \qquad \forall (k,l) \in I \tag{41}$$

As all our operations are based on cyclic matrices, an interesting property follows for matrices $\underline{A}$ and $\underline{B}$ of the same size

$$\mathrm{DFT}(\underline{A} * \underline{B}) = \mathrm{DFT}(\underline{A}) \odot \mathrm{DFT}(\underline{B}) \tag{42}$$

where $\odot$ is the element-wise product of the two matrices.

In our case, the values of the original map $\underline{P}$ are uniformly and independently chosen in the interval $[-1,1]$. Hence, the map $\underline{P}$ has an average value $\mu = 0$, thus a second moment equal to the variance $\sigma^2 = 1/3$. The autocorrelation matrix $\underline{\mathcal{R}} = \mathrm{AC}(\underline{P})$ has a maximum value equal to $uv\sigma^2$ at index (0, 0). The remaining values are all 0 as the values of $\underline{P}$ are independent of each other. Consequently, $\mathrm{DFT}(\underline{\mathcal{R}})$ is a matrix with all its values equal to $uv\sigma^2$.

We want to apply a filter $\underline{\mathcal{F}}$ to the original map $\underline{P}$ such that the resulting map $\underline{\Gamma} = \underline{\mathcal{F}} * \underline{P}$ has a predefined autocorrelation matrix $\underline{\mathcal{G}}$. By applying the DFT transformation, we get:

$$\begin{aligned}
\mathrm{DFT}(\underline{\mathcal{G}}) &= \mathrm{DFT}\Big(\overline{(\underline{\mathcal{F}} * \underline{P})^-} * (\underline{\mathcal{F}} * \underline{P})\Big) \\
&= \mathrm{DFT}\Big(\overline{\underline{\mathcal{F}}^-}\Big) \odot \mathrm{DFT}\Big(\overline{\underline{P}^-}\Big) \odot \mathrm{DFT}\Big(\underline{\mathcal{F}}\Big) \odot \mathrm{DFT}\Big(\underline{P}\Big) \\
&= \mathrm{DFT}\Big(\overline{\underline{P}^-} * \underline{P}\Big) \odot \mathrm{DFT}\Big(\overline{\underline{\mathcal{F}}^-}\Big) \odot \mathrm{DFT}\Big(\underline{\mathcal{F}}\Big) \\
&= uv\sigma^2 \cdot \mathrm{DFT}\Big(\overline{\underline{\mathcal{F}}^-}\Big) \odot \mathrm{DFT}\Big(\underline{\mathcal{F}}\Big)
\end{aligned} \tag{43}$$

Among the many possible filters $\underline{\mathcal{F}}$ that satisfy this equation, we can choose the one that is point invertible to its conjugate $\underline{\mathcal{F}} = \overline{\underline{\mathcal{F}}^-}$. We can directly calculate it via:

$$\underline{\mathcal{F}} = \tfrac{1}{\sigma\sqrt{uv}} \cdot \mathrm{DFT}^{-1}\left(\sqrt{\mathrm{DFT}(\underline{\mathcal{G}})}\right) \tag{44}$$