On Link Harness Optimization of Embedded Ethernet Networks

Jörg Sommer, Elias A. Doumith[1], Quentin Duval[2]

University of Stuttgart, Institute of Communication Networks and Computer Engineering (IKR)
Pfaffenwaldring 47, 70569 Stuttgart, Germany
Email: joerg.sommer@ikr.uni-stuttgart.de

Abstract— **During the last decades, Ethernet progressively became the most widely used Local Area Network (LAN) technology. It evolved from a bus topology to a micro-segmented network with full duplex links. Apart from LAN installations, Ethernet became also attractive for embedded application areas such as industrial, automotive, and avionics. In these areas, the connectivity between the devices and the switches results in link harnesses. These harnesses can be bundled together and installed inside ducts. Since all the links do not have the same endpoints, some full duplex links may leave a duct at points referred to as junction points.**

In this paper, we propose a Simulated Annealing based algorithm to optimize the topology design of embedded Ethernet networks. This algorithm finds the (near-)optimal positions of a given number of switches and their connections to a given set of nodes. When we take into account that links are organized into link harnesses and installed into ducts, we have to find also the number of junction points required as well as their optimal positions. For this purpose, we propose two algorithms and compare them in terms of computation time and the quality of the obtained solution. Finally, we highlight the cost benefits of bundling links and installing them into ducts.

I. 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 keys 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 [1].

[1]From 2007 to 2009, Elias A. Doumith was at the IKR. Now, he is with Telecom ParisTech, Paris, France. Email: elias.doumith@telecom-paristech.fr

[2]At the time of writing, Quentin Duval was a student at the IKR.

A prominent example of Ethernet's application area is the industrial domain where Ethernet is gaining ground over traditional fieldbuses. Another area is avionics, where today's Ethernet proved its ability to fulfill real-time requirements needed in such an environment [2]. Last but not least, the automotive industry is investigating Ethernet as a suitable in-vehicle network technology [3], [4].

A motivation to use Ethernet in embedded networks is the use of commercial off-the-shelf components. This allows manufactures 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 today's embedded environments, wire harnesses are used to interconnect all the network components/devices and to supply them with power. By bundling the wires and cables together and deploying them inside ducts, they can be protected against various adverse effects such as vibrations, moisture, and over-heat [5]. Moreover, installing a wire/cable harness, as opposed to multiple individual wires, reduces the installation time and consequently the deployment cost. However, since all the links do not have the same endpoints, some full duplex links may leave a duct at some points referred to as junction points or breakouts. At these junction points, a bifurcation is created into the duct in order to protect the leaving links as well as the remaining links in the original link harness. In the sequel, we mainly focus on the deployment of full duplex data links that interconnect the devices and the switches of an embedded Ethernet network. We do not consider the cables required to supply these components with electrical power. Hence, we use the term *link harness* to refer to the full duplex links bundled and installed inside a duct. In Figure 1, we give an example of such link harnesses.

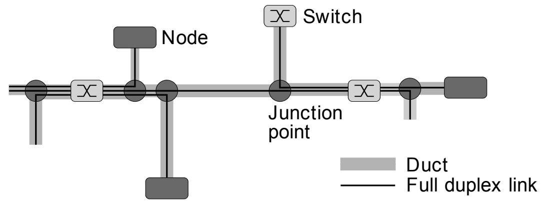


Fig. 1. Example of an embedded Ethernet topology design making use of link harnesses.

When designing an embedded Ethernet network, we have to optimize the number of switches used, their positions, and their connections to a given set of nodes. Moreover, we have to take into account that the links are organized into link bundles and installed into ducts. For this purpose, we have also to optimize the number of junction points required as well as their positions. Thus, the resulting problem is very challenging. In this paper, we propose a Simulated Annealing (SA) based algorithm to optimize the position of the switches. Concerning the latter part of the problem, focusing on junction point optimization, we propose a SA based algorithm and a Descending algorithm.

The rest of the paper is organized as follows. First, we model the environmental conditions by a cost map, discuss the economy of scale of link harnesses and introduce different network designs. Then, we present a link harness cost model. In Section III, we propose algorithms to optimize the link harness cost. In Section IV, we evaluate the performance of the algorithms as well as the cost benefit of link harnesses. Finally, we conclude the paper in Section V and give an outlook for future work.

II. Characteristics and Modeling

A. Cost Map

In contrast to traditional LANs, the environmental conditions impose constraints on the network deployment. For instance, in aircrafts and cars, we have to deploy at some places links with better shielding due to electromagnetic interference or to deploy extra heat-resisting links due to the environment temperature. At other places, it might be extremely expensive or even impossible to deploy a link. Such constraints add up to the design complexity of an embedded network because 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

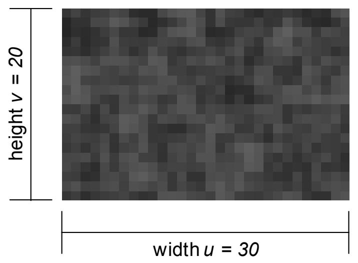


Fig. 2. An example of a 30×20 cost map with an autocorrelation structure.

areas or pixels. For a pixel at position $I(x_I, y_I)$, we assign a non-negative real value γ_{x_I,y_I} in the interval $[0, 1]$ that represents the cost of deploying a single link segment inside a duct at that particular position. Thus, the embedded environment can be represented by a $u \times v$ matrix $\underline{\Gamma} = (\gamma_{x,y})$ referred to as the cost map. Based on this matrix, we compute the minimal cost $\omega_{I,J}$ required to deploy a single link within a duct between the position $I(x_I, y_I)$ and the position $J(x_J, y_J)$ of the embedded environment. The element $\omega_{I,J}$ is a non-negative real value computed by applying the Floyd-Warshall algorithm [6] to the before-mentioned matrix $\underline{\Gamma}$. The minimal cost between any two positions of the embedded environment is stored in a $uv \times uv$ matrix noted $\underline{\Omega} = (\omega_{I,J})$.

Typically, we can derive the cost map from any environment skeleton where preferred paths are represented by pixels with lower 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 algorithm. We want to point out, though, that the performance of our algorithm is independent of the model used to create the cost map.

In embedded environments, low cost areas have in most cases low cost neighboring areas. The same also holds for high cost areas. Thus, pure random cost maps are not sufficient, they must exhibit a certain autocorrelation structure given by an autocorrelation matrix. The latter approach is borrowed from mobile radio network simulators where *shadowing* or *shadow fading* effects are modeled by a map with a given autocorrelation structure [7]. For a detailed description of the cost map, the reader can referred to [8] where we provide a method and a mathematical proof for generating cost maps with a given autocorrelation structure. Figure 2 shows an example of such a cost map. The darker the pixels in the map, the lower their corresponding values.

B. Harness Design

When designing embedded Ethernet networks without considering the possibility of bundling links and deploying them inside ducts, we have to optimize the number of switches used, their positions, and their connections to a given number of nodes. For a given set of nodes and given positions of the switches, the optimal topology is obtained by connecting each node to its nearest switch along the path with the lowest cost. As for the switches, they are connected between them using a tree like topology with the lowest cost obtained for example using the Minimum Spanning Tree (MST) algorithm [9], [10].

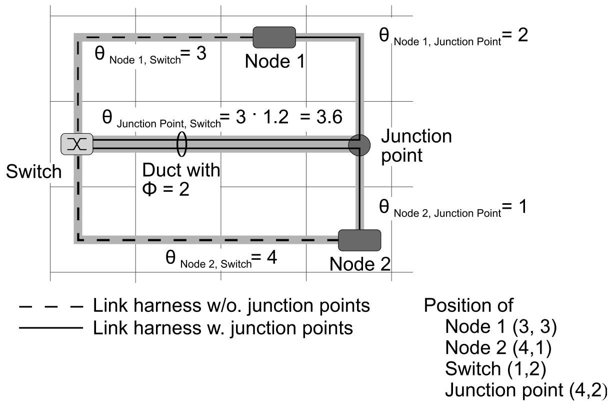
By allowing multiple links following the same path to be bundled together and installed within a single duct, the optimal topology is no more obtained by using the paths/tree with the lowest cost. In fact, by allowing some links to be deployed along paths with higher cost, they can be bundled together along a common path segment and installed in a common duct. Thus, the cost penalty due to the use of individual higher link costs can be compensated by the cost benefit due to the use of a common duct instead of separate ones. This holds if we assume that the cost of ϕ links bundled together and installed within a common duct is smaller than the sum of the costs of the ϕ links deployed separately within ϕ different ducts. Moreover, by binding multiple links into a harness and installing them within a duct, they can be better protected against the adverse effects of heat, abrasions, and moisture. Since the installer has only one link harness to install instead of multiple links, the assembly time is decreased and the process can be easily standardized.

According to the before-mentioned property, we define the cost $\theta_{I,J}(\phi)$ of a single duct comprising ϕ links between the position $I(x_I, y_I)$ and the position $J(x_J, y_J)$ as:

$$\theta_{I,J}(\phi) = (\lambda \cdot \phi + c) \cdot \omega_{I,J} \tag{1}$$

with $\lambda + c = 1$, $\frac{\lambda}{c}$ is the ratio of the cost of a single link to the cost of a single duct for a unit length, and $\omega_{I,J}$ is the minimal cost between I and J obtained from the cost map $\underline{\Gamma}$. This cost function can be generalized to more complex functions without affecting the proposed algorithms. Such generalization can reflects more general assumptions such as the *Economy of Scale* where higher number of links bundled together within one duct results in lower cost per link.

Figure 3 illustrates the benefit of using junction points. For simplicity, we assume a constant cost map with $\gamma_{i,j} = 1$ $(\forall i = 1, \cdots, u, \forall j = 1, \cdots, v)$. We also



θ Node 1, Switch = 3
Node 1
θ Node 1, Junction Point = 2
θ Junction Point, Switch = 3 · 1.2 = 3.6
Junction point
Switch
Duct with Φ = 2
θ Node 2, Junction Point = 1
θ Node 2, Switch = 4
Node 2
Link harness w/o. junction points
Link harness w. junction points
Position of
Node 1 (3, 3)
Node 2 (4,1)
Switch (1,2)
Junction point (4,2)

Fig. 3. Different resulting link harnesses: Neglecting junction points (dashed lines) and placing junction points (solid lines).

assume that the cost of a duct is four times more expensive than the cost of a single link, thus we have $c = 0.8$ and $\lambda = 0.2$. It should be noted that at the endpoints, only half of the cost is required, and that a link connecting two endpoints within the same area has no cost. For instance, *Node 1* is connected to the *Switch* along the shortest path (upper dashed line). The cost of the link and duct along this path is equal to $\theta_{\text{Node 1, Switch}}(1) = 1/2 \cdot 1 + 2 \cdot 1 + 1/2 \cdot 1 = 3$. Similarly, the cost of the duct and link along the shortest path between *Node 2* and the *Switch* (lower dashed line) is equal to $\theta_{\text{Node 2, Switch}}(1) = 1/2 \cdot 1 + 3 \cdot 1 + 1/2 \cdot 1 = 4$. Consequently, the total link cost of this topology is $3 + 4 = 7$.

If we allow the connection between the nodes and the *Switch* to take a more expensive path such as the one through the *Junction point* (solid lines), the total cost of this new topology is equal to 6.6. This comprises the cost between the nodes and the *Junction point* plus the cost between the *Junction point* and the *Switch*. Although the connection between *Node 1* and the *Switch* is more expensive, allowing multiple links to be bundled and installed within a duct results in lower total cost.

C. Network Designs

Besides determining the position of junction points, 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 4 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.

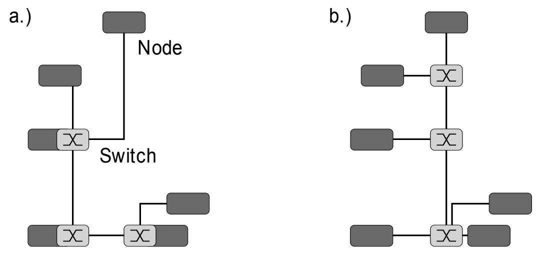


Fig. 4. Embedded Ethernet network designs: a.) Tree with integrated switches and b.) Tree with self-contained switches.

- In the second design referred to as *Self-contained Switches*, the switches can be placed anywhere in the environment space (*cf.* Figure 4 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 the optimization of link harnesses is harder to solve than traditional network dimensioning problems [11] where normally the position of the switches is given and the connectivity between them has to be optimized. Furthermore, traditional topology design problems do not consider enhanced cost functions that enforce link bundles. In our case, only the number of the nodes' and their positions are given as an input. Consequently, we have to find, for a given number of switches, the optimal position of these switches, and the optimal number of junction points and their position as well as the connectivity between the junction points, the nodes, and the switches.

If we assume that each link is deployed in a separate duct, and if we release the constraint on the limited number of switches, the problem we are investigating reduces to the standard Steiner Tree (ST) problem that is known to be NP-complete [12]. Therefore, link harnesses optimization is inherently more difficult than the well-known *NP*-Complete ST problem.

D. Link Harness Cost

We have a given set of n nodes $N_i(x_i, y_i)$ $(i = 1, \ldots, n)$, a number m of switches, and a number l of junction points. As stated before, in an embedded environment the link cost depends on its position. Thus, our objective is to minimize the total link cost C, also called link harness cost, by positioning m switches $S_j(x_j, y_j)$ $(j = 1, \ldots, m)$ and l junction points $B_k(x_k, y_k)$ $(k = 1, \ldots, l)$, connecting the nodes to the switches, and the switches between them either directly or through junction points. For an instance of this problem, we define the following matrices:

- $\underline{\Psi} = (\psi_{i,j})$ is a $n \times (m + l)$ matrix representing the existence of ducts between the nodes and the switches or the nodes and the junction points where $\psi_{i,j}$ is a binary variable specifying the presence or the absence of a duct between node N_i and switch S_j $(1 \leq j \leq m)$ or between node N_i and junction point B_{j-m} $(m + 1 \leq j \leq m + l)$.

$$\psi_{i,j} = \begin{cases} 1 & \text{if } N_i \text{ is connected to } S_j \text{ or } B_{j-m}, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Since we investigate an Ethernet network without redundant paths, there exist exactly one duct between a node and either a switch or a junction point. Thus, we have the following constraint:

$$\sum_{j=1}^{m+l} \psi_{i,j} = 1, \ \forall i = 1, \ldots, n \tag{3}$$

It is to be noted that each of these ducts accommodates exactly one link segment.

- $\underline{\Upsilon} = (\upsilon_{i,j})$ is a $(m+l) \times (m+l)$ matrix representing the existence of ducts between the switches and the junction points where $\phi_{i,j}$ is a binary variable specifying the presence or the absence of a duct between switch S_i/junction point B_{i-m} and switch S_j/junction point B_{j-m}.

$$\upsilon_{i,j} = \begin{cases} 1 & \text{if } S_i/B_{i-m} \text{ is connected to } S_j/B_{j-m}, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

- $\underline{\Phi} = (\phi_{i,j})$ is an $(m+l) \times (m+l)$ matrix representing the number of link segments between switch S_i/junction point B_{i-m} and switch S_j/junction point B_{j-m} where $\phi_{i,j}$ is a non-negative integer variable.

$$\phi_{i,j} = \begin{cases} \geq 1 & \text{if } S_i/B_{i-m} \text{ is connected} \\ & \text{to } S_j/B_{j-m}, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

After placing the switches and the junction points, we calculate the cost of the ducts and the link segments:

- $\underline{\Delta} = (\delta_{i,j})$ is a $n \times (m+l)$ matrix derived from the matrix $\underline{\Omega}$. It contains the cost needed to deploy a duct and a single link between nodes and switches or between nodes and junction points respectively. where $\delta_{i,j}$ is a non-negative real value equal to the minimal cost between node N_i and switch S_j/junction point B_{j-m}.

- $\underline{\Theta} = (\theta_{i,j})$ is a $(m+l) \times (m+l)$ matrix containing the minimal cost of the ducts and links between the switches and the junction points where $\theta_{i,j}$ is a non-negative real value. This matrix is derived from the matrices $\underline{\Omega}$ and $\underline{\Phi}$ according to the cost function given by Equation (1). For instance, the minimal cost of a duct between i^{th} switch at position $I(x_I, y_I)$ and the k^{th} junction point at position $J(x_J, y_J)$ is

$$\theta_{i,k+m} = \theta_{I,J}(\phi_{i,k+m}) \tag{6}$$

The total link cost C of the harness results in

$$C = \sum_{\substack{i=1,\dots,n\\ j=1,\dots,m+l}} \psi_{i,j} \cdot \delta_{i,j} + \sum_{\substack{i=1,\dots,m+l-1\\ j=i+1,\dots,m+l}} \upsilon_{i,j} \cdot \theta_{i,j} \tag{7}$$

In the next section, we present algorithms to minimize the link harness cost of embedded Ethernet networks.

III. Algorithms

In this section, we present several algorithms that, when combined, can solve the complex problem formulated in the previous section. We propose a Simulated Annealing (SA) based algorithm to find the optimal positions of a given number of switches. It is commonly known that SA is able to find a (near-)optimal solution in a short time [13]. The basic idea of SA is to improve a given initial solution iteratively by applying defined rearrangement operations.

In order to find the optimal number of junction points and their optimal positions, we present two algorithms: A SA based algorithm and a Descending algorithm. In order to solve the link harness problem, we combine the optimization of the switches' position and that of the junction points' position either sequentially or iteratively.

A. Simulated Annealing based Switch Positioning Algorithm

In this section, we propose an SA based algorithm, called *SA/S*, which is able to find the (near-)optimal positions of a given number of switches. As an initial solution S_0 of this algorithm, which is listed in Algorithm 1, we place m switches randomly within a pre-defined solution space, interconnect them applying a MST algorithm [9], [10], and then connect each node to its nearest switch, i.e., the switch that requires the lowest cost. By (inter)connecting we mean installing a duct. For given switches' 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, also called perturbations (Line 15 of Algorithm 1) to the current solution, a new solution S_x of cost C_x is obtained. Based on the duct structure, we are able to deploy the full duplex links between nodes and switches and between switches themselves. Due to the constraint defined in Equation (3) and applying a MST algorithm to interconnect the switches, there exist exactly one path between nodes and switches and exactly one path between two switches.

In the case of integrated switches, the solution space is restricted to the nodes' position, and the perturbation consists in moving a randomly selected switch from its current node position to the position of another randomly selected node. In contrast to this, in the case of self-contained switches, the perturbation can move 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 20 of Algorithm 1). In order to avoid local minima, solutions with higher cost than the current solutions 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 30 of Algorithm 1). If the costs of the new solution and the current solution are equal, one can randomly choose to accept the new solution as the current solution or rejected it (Line 27 of Algorithm 1).

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

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 harness cost. These are switches to which are only two full duplex links are connected $\forall j = 1, \dots, m$:

$$\sum_{i=1}^{n} \psi_{i,j} + \sum_{i=1}^{m} \upsilon_{i,j} \leq 2 \tag{8}$$

Normally, it is possible to delete one or more switches if the number of switches is close to the number of nodes ($m \simeq n$).

Algorithm 1 SA based algorithm

```
1: define
2:    mIter  ▷ maximum number of consecutive iterations without any improvement
3:    mImpr  ▷ maximum number of improvements
4:    mAtmp  ▷ maximum number of attempts
5:    $T$  ▷ initial temperature
6:    $p$  ▷ cooling factor
7: end define
8: procedure SA($\{N_i(x_i, y_i)\}$, $m$, $\underline{\Omega}$))
9:    Construct an initial solution $S_0$; $C_0 \leftarrow$ COST($S_0$)
10:   $(S_{\text{best}}, C_{\text{best}}) \leftarrow (S_0, C_0)$
11:   iter $\leftarrow 0$
12:   while iter $<$ mIter do
13:      impr $\leftarrow 0$; atmp $\leftarrow 0$
14:      while (impr $<$ mImpr) $\wedge$ (atmp $<$ mAtmp) do
15:         $S_{\text{x}} \leftarrow$ PERTURBATION($S_0$)
16:         $C_{\text{x}} \leftarrow$ COST($S_{\text{x}}$)
17:         atmp++
18:         Generate a random number $r \in [0, 1)$
19:         if $C_{\text{x}} < C_0$ then  ▷ improved solution
20:            $(S_0, C_0) \leftarrow (S_{\text{x}}, C_{\text{x}})$
21:            impr++
22:            if $C_{\text{x}} < C_{\text{best}}$ then
23:               $(S_{\text{best}}, C_{\text{best}}) \leftarrow (S_{\text{x}}, C_{\text{x}})$
24:               iter $\leftarrow 0$
25:            end if
26:         else if $C_{\text{x}} = C_0$ then  ▷ equivalent solution
27:            if $r < 0.5$ then
28:               $(S_0, C_0) \leftarrow (S_{\text{x}}, C_{\text{x}})$
29:            end if
30:         else if $r < \mathrm{e}^{-\frac{(C_{\text{x}} - C_0)}{C_0 T}}$ then  ▷ worse solution
31:            $(S_0, C_0) \leftarrow (S_{\text{x}}, C_{\text{x}})$;
32:         end if
33:      end while
34:      if atmp $=$ mAtmp then  ▷ lot of attempts
35:         iter++
36:      end if
37:      $T \leftarrow pT$
38:   end while
39:   return $(S_{\text{best}}, C_{\text{best}})$
40: end procedure
```

B. Simulated Annealing based Junction Point Positioning Algorithm

1) Principle: In this section, we propose an SA based algorithm, called SA/JP, which finds (near-)optimal positions of junction points. This algorithm works similar to the one introduced in Section III-A. We also define the parameters mIter, mImpr, mAtmp, T, and p. The differences are the input parameters and the perturbation. The former are a set of nodes and their positions $\{N_j(x_i, y_i)\}$, a set of switches and their positions $\{M_j(x_j, y_j)\}$ as well as the initial number of junction points and the matrix $\underline{\Omega}$.

Similar to the SA/S algorithm, as an initial solution S_0, we position l junction points randomly. Then, we connect each node to its nearest switch or its nearest junction point and we interconnect the switches and the junction points using a MST algorithm. By (inter)connecting we mean installing a duct between two entities (node, switch, or junction point).

By applying the perturbation to the current solution, a new solution S_x with cost C_x is obtained. The new solution S_x includes all the ducts. Based on this solution, we are able to deploy the full duplex links between nodes and switches and between switches themselves. Due to the constraint defined in Equation (3), there exist exactly one path between nodes and switches and exactly one path between two switches. Based on this link harness and the cost function, we compute the cost C_x of solution S_x. The perturbation selects with a probability of 80% a junction point randomly and moves this one from its current position to another position also selected randomly. With a probability of 10%, the perturbation deletes a randomly selected junction point and with a probability of 10%, the perturbation adds a new junction point on a randomly selected position. In case of zero junction points, the perturbation adds a new junction point.

Since the perturbation removes and adds junction points, the sizes of the matrices that we introduced in Section II-D can vary from solution to solution.

In a post-processing step, the SA/JP algorithm deletes unnecessary junction points. A junction point is unnecessary, if its non-existence does not affect the total cost. To delete unnecessary junction points, the algorithm tests for each of them, if the deletion affects the cost $\forall j = m+1, \ldots, m+l$:

$$\sum_{i=1}^{n} \psi_{i,j} + \sum_{i=m+1}^{m+l} \upsilon_{i,j} \leq 2 \qquad (9)$$

2) Configuration:

a) Nested: We nest the SA/JP algorithm into the SA/S algorithm (between Line 15 and Line 16 of Algorithm 1), where the nested algorithm is responsible to find the (near-)optimal positions of the junction points

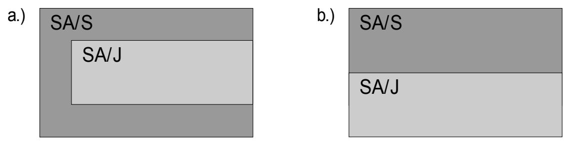


Fig. 5. Possible Configurations: a.) Nested processing configuration, called *N/SA/JP* and b.) Sequential processing, called *S/SA/JP*.

and the outer algorithm to find the (near-)optimal positions of the switches.

b) Sequential: Instead of nesting the SA/JP algorithm into the SA/S algorithm, we run both algorithms SA/S and SA/JP sequentially. Practically, we run the SA/S algorithm that positions the switches without considering junction points in the topology and, afterwards, we use this positioning as the input of the additional SA/JP algorithm. Figure 5 depicts the two configurations.

C. Descending Junction Point Positioning Algorithm

1) Principle: In this section, we propose a Descending algorithm, called D/JP, which finds (near-)optimal positions of junction points. The main difference with respect to the previous SA based algorithm is in the perturbation function.

As an initial solution S_0, we position l junction points randomly and save this solution as the best solution. While the current number of iterations iter is smaller than the maximum number of iterations mIter a new perturbation is applied. It consists in always adding a given number of new junction points to the current solution. Each node is connected to its nearest switch or its nearest junction point. The switches and the junction points are interconnected using a MST algorithm.

Finally, we remove unnecessary junction points. By this way, since the positions of the junction points of the current best solution are kept in the newly constructed one, the link harness cost can only decrease or remain unchanged. Thus, the Descending algorithm can be viewed as a SA algorithm that only accepts better solutions.

2) Configuration: As for the SA/JP algorithm, the D/JP algorithm can be nested into the SA/S algorithm (N/D/JP) or can be run sequentially (S/D/JP).

IV. Evaluation

In the previous section, we presented algorithms to minimize link harness cost of an embedded Ethernet network. In this section, we investigate the benefit of bundling links and installing them into ducts. Furthermore, we compare the performance of the algorithms

Algorithm 2 Descending algorithm

```
1: define
2:     mIter  ▷ maximum number of iterations
3: end define
4: procedure SA($\{N_i(x_i, y_i)\}$, $\{M_i(x_i, y_i)\}$, $l$, $\underline{\Omega}$))
5:     Construct an initial solution $S_0$; $C_0 \leftarrow$ COST($S_0$)
6:     $(S_{\text{best}}, C_{\text{best}}) \leftarrow (S_0, C_0)$
7:     iter $\leftarrow 0$
8:     while iter $<$ mIter do
9:         $S_x \leftarrow$ PERTURBATION($S_{\text{best}}$)
10:        $C_x \leftarrow$ COST($S_x$)
11:        if $C_x < C_{\text{best}}$ then  ▷ improved solution
12:            $(S_{\text{best}}, C_{\text{best}}) \leftarrow (S_x, C_x)$
13:            iter $\leftarrow 0$
14:        end if
15:        iter++
16:    end while
17:    return $(S_{\text{best}}, C_{\text{best}})$
18: end procedure
```

and the obtained final solution. In [1] and [14], we have already evaluated the performance of the SA/S algorithm by comparing its chosen solutions with the optimal solutions found by a Mixed-Integer Linear Program (MILP). Although in [14] we apply the SA/S algorithm to optimize the topology of in-vehicle multimedia communication systems, the algorithm is not domain specific and can be applied to optimize any types of embedded Ethernet networks. We have shown that this algorithm achieves (near-)optimal solutions in short computation time. In addition, we have shown that the computation time of the MILP increases enormously with an increasing number of switches, although we did not consider junction points.

For the evaluation, we investigate several networks with different number of nodes where we have to place different number of switches. In all networks, we use a 50×50 cost map with an autocorrelation structure as well as a linear cost function (*cf.* II-B) with $c = 0.8$ (fixed duct cost) and $\lambda = 0.2$ (relative link cost). The probability of finding an optimal solution decreases with an increasing solution space, which results from the size of the cost map, the number of nodes, the number of switches, and if we consider junction points. Consequently, the parameters of the algorithms should depend on the solution space. Based on a large number of experiments and analytical approximations, we use the following parameters for the SA/S algorithm: $\mathrm{mIter} = 15 \cdot \ln(u \cdot v)$, $\mathrm{mAtmp} = n \cdot m \cdot \ln(u \cdot v)$,

$\mathrm{mImpr} = 0.1 \cdot \mathrm{mAtmp}$, $T = 0.5$, and $p = 0.9$. For the S/SA/JP algorithm, we use the same parameters as for the SA/S algorithm. For the N/SA/JP algorithm, we use a tenth of the values of the S/SA/JP algorithm. For the N/D/JP algorithm, we use the same number of maximum iterations as for the S/SA/JP algorithm. For comparable in accuracy with the S/SA/JP algorithm, we double the number of maximum iterations for the S/D/JP algorithm ($2 \cdot \mathrm{mIter}$). We choose the initial number of junction points equal to half the number of nodes ($l = n/2$).

A. Algorithms' Performance

In this section, we compare the performance of the SA/JP and the D/JP algorithm as well as both configurations *sequential* and *nested*.

a) D/JP vs. SA/JP: Table I shows the minimal total link cost C of solutions chosen by different algorithms. The solutions of the SA/JP are most of the time better than the one found by the D/JP with *equivalent parameters* (parameters that allow similar level of precision), but require considerable longer computational time. For example, for a network with 8 integrated switches and 15 nodes, the computation time is 385 times longer. The D/JP algorithm is suitable if a quick answer is preferred to the accuracy of the solution or if we drive at comparing a large number of networks.

If we choose *worse* parameters for the SA/JP, as it is, in the Table I, the case for the nested configuration, the computational time is equivalent, but the D/JP finds better solutions.

b) Configuration: Sequential vs. Nested: Generally, the switches' positions are not identical if we use junction points or not. Therefore, running the SA/JP or the D/JP algorithm after finding optimized switches' positions, i.e., using the sequential configuration, provides generally only a sub-optimal solution, since we try to find the optimal positions of the junction points based on exactly one solution chosen by the SA/S algorithm. On the contrary, nesting the SA/JP or the D/JP into the SA/S considers a larger solution space by trying to find the optimal positions of the junction points for different settings of switches' positions. This again influences the switches' positions. Thus, the chosen solutions have generally lower cost in comparison to the sequential configuration as shown in Table II. However, considering a larger solution space leads to an enormously increasing computation time, which is the major drawback of the nested configuration.

Although, the sequential configuration considers a smaller solution space, its chosen solution is not far away from the nested configuration. If we consider a set of positions of switches and junction points, the sets are similar in both cases. For example, if the nested configuration finds a solution with two switches on position $S_1(x_1, y_1)$ and $S_2(x_2, y_2)$, and one junction point on position JP(x_3, y_3), the sequential configuration finds normally a solution with $S_1(x'_2, y'_2)$, $S_2(x'_3, y'_3)$, and JP(x'_1, y'_1), where $x_i \approx x'_i$ and $y_i \approx y'_i$. Due to the smaller solution space considered, the computation time of the sequential configuration is in general shorter. Consequently, we should use the sequential configuration in order to find a good solution in a short time and to get a feeling of the possible improvement of using junction points.

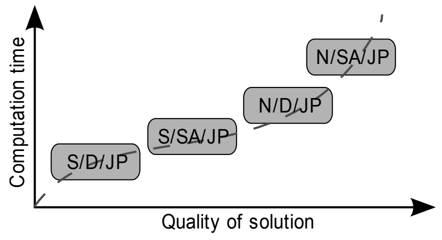


Fig. 6. Comparison of the algorithms' performance.

Considering Table I and II, it might make more sense to execute a number of independent runs with the sequential configuration instead of using the nested configuration. Figure 6 depicts the performance of the algorithms. Normally, the S/SA/JP and S/D/JP algorithm find an acceptable solution within a short computation time, where the former performs better. If we are interested in solutions with a high quality (low cost), we have to nest the algorithms into the SA/S; but the price is an enormously increasing computation time.

B. Network Designs

In this section, we compare the network designs mentioned in Section II-C. Without positioning junction points, i.e., applying only the SA/S algorithm, the total link cost of the design with self-contained switches is smaller than the link cost with the integrated switches. This is because the solution space of the latter design is included in the former one.

As we mentioned before, the optimal switches' positions are not the same with and without using junction points. Consequently, positioning of the switches in a post-processing step, i.e., sequential configuration, the cost of the design with self-contained switches must not to be mandatory less than the cost of the design with integrated switches, although it is often the case. For

TABLE I

Comparison of the algorithms (with configuration *Sequential* and 10 independent runs).

Network design	n	m	SA/S		S/D/JP				S/SA/JP				Cost
			C	t [min]	C	l	max(ϕ)	t [min]	C	l	max(ϕ)	t [min]	reduction [%]
Self-Cont. Switches	10	1	**111.25**	< 1	74.40	8	7	< 1	**73.80**	10	7	$\simeq 10$	33.7
		2	**82.75**	< 1	64.45	8	4	< 1	**63.45**	11	4	$\simeq 30$	23.3
		5	**63.81**	< 1	59.42	4	2	$\simeq 1$	**58.82**	7	2	$\simeq 175$	7.8
		8	**57.19**	< 1	**57.19**	0	1	$\simeq 1$	**57.19**	0	1	$\simeq 230$	0.0
	15	2	**117.84**	< 1	89.87	10	4	$\simeq 2$	**88.62**	15	5	$\simeq 180$	24.8
		5	**92.64**	< 1	80.15	9	3	$\simeq 8$	**80.06**	14	4	$\simeq 1300$	13.6
		8	**83.02**	< 1	78.12	6	2	$\simeq 10$	**77.59**	12	2	$\simeq 3350$	6.5
		12	**76.04**	< 1	75.02	2	2	$\simeq 8$	**75.41**[2]	4	2	$\simeq 6100$	0.8
Integrated Switches	10	1	**119.32**	< 1	71.14	7	5	< 1	**70.77**	13	5	$\simeq 7$	40.7
		2	**92.20**	< 1	67.57	6	4	< 1	**66.86**	11	5	$\simeq 30$	27.5
		5	**67.84**	< 1	62.62	3	3	< 1	**62.38**	8	3	$\simeq 135$	8.0
		8	**66.26**	< 1	**66.02**[1]	1	2	$\simeq 1$	**66.02**[1]	2	2	$\simeq 300$	0.4
	15	2	**124.35**	< 1	88.60	11	4	$\simeq 3$	**87.87**	15	4	$\simeq 240$	29.3
		5	**99.02**	< 1	83.57	8	3	$\simeq 5$	**83.46**	11	3	$\simeq 1000$	15.7
		8	**85.15**	< 1	80.04	5	3	$\simeq 7$	**79.84**	8	3	$\simeq 2700$	6.2
		12	**83.50**	< 1	80.89[2]	3	2	$\simeq 11$	**80.76**[2]	5	2	$\simeq 6500$	3.3

[1] Post-processing switch deletion: 7 switches used instead of (given) 8.
[2] Post-processing switch deletion: 10 switches used instead of (given) 12.

TABLE II

Comparison of the algorithms' computation time (for n =10 nodes and with 1 independent run).

		Sequential						Nested			
Network	m	SA		S/D/JP		S/SA/JP		N/D/JP		N/SA/JP	
design		C	t [min]	C	t [min]	C	t [min]	C	t [min]	C	t [min]
Self-Cont. Switches	1	111.25	< 1	75.30	< 1	75.77	$\simeq 1$	69.54	$\simeq 100$	71.70	$\simeq 70$
	2	82.75	< 1	65.30	< 1	64.08	$\simeq 6$	63.25	$\simeq 800$	64.98	$\simeq 1000$
Integrated Switches	1	119.32	< 1	71.22	< 1	71.19	$\simeq 1$	70.77	$\simeq 100$	71.86	$\simeq 100$
	2	92.20	< 1	69.21	< 1	67.06	$\simeq 3$	64.73	$\simeq 500$	65.62	$\simeq 1000$

example, in Table I, with 10 nodes and 1 switch, the design with integrated switches achieves better results with both algorithms SA/JP and D/JP.

Another reason that the design with integrated switches achieves sometimes better results is that the solution space is smaller compared to the design with self-contained switches. If we execute the algorithms with the same parameters, the probability to find a (near-)optimal solution is a priori higher.

C. Harness Design

The objective of this paper is to optimize the total link cost of an embedded Ethernet network by placing a given number of switches as well as by deploying junction points and finding their (near-)optimal positions. In section III-A, we introduce algorithms for this purpose. As shown in Table I, by deploying additional switches, we can reduce the harness cost. As higher the number of switches, as lower the link harness cost.

As shown also in the Table I and II, by deploying junction points, we are able to decrease further the harness cost. As we can state from this table, the larger the number of switches, the smaller the benefit of the junction points. The reason is that the more switches we have, the less we have useful possible junction points spots to decrease the cost. Indeed the switches and junction points are similar but for the multiplexing property, that junction points do not share with switches and thus, simplifying to the extreme situation, we can say that one additional switch equals at least minus one useful junction point. Besides the number of junction points, also the maximum number of links in a duct max(ϕ) decreases also if the number of switches in a network increases. This confirms the statement that the benefit of installing link bundles into ducts is smaller if we deploy a large number of switches.

Eventually, passed a given number of switches, which depends on the network, no more junction points are needed, and the cost reduction goes to zero. Thus, as for considering the link harness cost only, position junction points is out performed by deploying additional switches instead. However, this statement is wrong if we consider the overall deployment cost, which include the cost of all network components, and not only the link harness cost. Indeed, adding a switch represents a non-negligible cost, whereas adding a junction point does not (or significantly lesser). Depending on this switch cost, both the optimal number of switches and the improvement achieved by link bundles vary. For example, by selecting a switch cost of 10, and add this cost multiplied by the number of switches to each line, we can conclude that the minimal cost without junction points is 102.75 (10 nodes and 2 switches), and 79.54 with junction points (10 nodes and 1 switch), leading to a 22.6% cost reduction.

V. Conclusion and Outlook

During the last decade, apart from LAN installations, Ethernet became also attractive for other application areas such as industry and avionics. In these areas, we have to consider additional constraints and environmental conditions, while reducing the cost. Since link harnesses have a number of advantages, in embedded fields they are deployed instead of single links. In this paper, we proposed algorithms to optimize link harnesses. We evaluated the performance of the algorithms and discussed the trade-off between computation time and the quality of the obtained solution. Furthermore, we highlighted the cost benefits of bundling links and installing them into ducts.

The algorithms in this paper are not limited to optimize the link harness of embedded Ethernet networks. With minor modifications, the algorithms can be generalized and can take into account, e.g., power wires.

Currently, we are working on approaches to find proper switch and junction point positions for the initial solutions, instead of using randomly choosing positions. This may further improve the results.

Until now, we focused on the link harnesses. We did not take into account higher layer constraints. In the future, we will extend the algorithms that we consider a given traffic demand matrix, while optimizing the network cost. The objective is that the resulting topology fulfills the traffic demands of the network with minimal cost. For this purpose, the algorithms take into account the traffic demands and, if necessary, deploy additional links.

Acknowledgment

The authors would like to thank Andreas Reifert for many fruitful discussions. The author would also like to thank the reviewers who have read and helped to improve the quality of this paper.

References

[1] J. Sommer, S. Gunreben, F. Feller, M. Köhn, A. Mifdaoui, D. Saß, and J. Scharf, "Ethernet – A Survey on its Fields of Application," February 2009, accepted for publication in the IEEE Communications Surveys and Tutorials.

[2] ARINC 664, *Aircraft Data Network, Part 7: Deterministic Networks*, 2003.

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

[4] M. Rahmani, K. Tappayuthpijarn, B. Krebs, E. Steinbach, and R. Bogenberger, "Traffic Shaping for Resource-Efficient In-Vehicle Communication," *IEEE Transactions on Industrial Informatics*, 2009, accepted for future publication.

[5] E. Aguirre and B. Raucent, "Performances of wire harness assembly systems," *IEEE International Symposium on Industrial Electronics, ISIE '94*, pp. 292–297, May 1994.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Second Edition*. MIT Press, September 2001.

[7] I. Forkel, M. Schinnenburg, and M. Ang, "Generation of two-dimensional correlated shadowing for mobile radio network simulation," in *Proceedings of The 7th International Symposium on Wireless Personal Multimedia Communications, WPMC 2004*, Abano Terme (Padova), Italy, September 2004, p. 5.

[8] J. Sommer, E. A. Doumith, and A. Reifert, "Cost-based Topology Optimization of Embedded Ethernet Networks," 2009, submitted to the International Journal of Embedded and Real-time Communication Systems (IJERTCS).

[9] J. J. B. Kruskal, "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem," *Proceedings of the American Mathematical Society*, vol. 7, no. 1, pp. 48–50, February 1956.

[10] R. Prim, "Shortest connection networks and some generalizations," *Bell System Technical Journal*, vol. 36, pp. 1389–1401, 1957.

[11] M. Pióro and D. Mehdi, *Routing, Flow, and Capacity Design in Communication and Computer Networks*. Morgan Kaufmann Publishers, 2004.

[12] M. R. Garey, R. L. Graham, and D. Johnson, "The complexity of computing Steiner minimal trees," *SIAM Journal on Applied Mathematics*, vol. 31, no. 4, pp. 835–859, June 1977.

[13] T. F. Gonzales, Ed., *Handbook of Approximation Algorithms and Metaheuristics*. Chapman & Hall CRC, 2007.

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