# Network Configuration:
# Approaches for Solving the Cable Management Problem*

**O. Niggemann, B. Stein, M. Suermann**
{murray, stein, michel}@uni-paderborn.de

**University of Paderborn**
**Department of Mathematics and Computer Science**
**Warburger Str. 100**
**D-33098 Paderborn**

**Abstract**

Local area networks (LANs) form the basis for worldwide data communication, and, during the last years, performance and size of these networks has been continuously increasing. Inevitably, the configuration of these networks became an important and sophisticated job. This paper presents solutions for the cable management problem, which is a central part of the network configuration process. Among others we show the following: 1. The simplified cable management problem (without bundling resctriction) is an instance of the minimum-cost-flow problem. 2. The standard cable management problem (with bundling resctriction) is NP-complete. From an applicational viewpoint, the central contribution of this paper is the development of a heuristic algorithm that tackles the standard cable management problem. Our approach produces sufficiently exact results in the network configuration domain, and it is much more efficient than exact algorithms that solve the weaker minimum-cost-flow problem.

## 1 Motivation

Local Area Networks (LANs) form the basis for worldwide data communication. During the last years, the performance and the size of these networks has been continuously increasing; networks with more than 1000 nodes are not rare anymore.

Networks are planned and configured by experts. The networks' sizes, the large number of available components, the different technologies as well as the structural and technical restrictions increase the complexity and the time needed when planning a new network. As a consequence, many network configurations are sub-optimum. An undersized network, for example, leads to a number of technical faults during operation. On the other hand, if a network is oversized, a customer can benefit from its performance reserves; however, these reserves are overpaid in the very most cases

* Reworked and extended version of a paper originally published at the 11. Workshop Planen und Konfigurieren (PUK 97).

because of the rapidly decreasing hardware costs. Note that the turn-around time of active components is about 5 years, that of passive components is about 15 years.[1]

Discussions with network experts revealed that there is a need to support the network configuration process. Our long-term objective is the development of a tailored configuration system. In this paper we focus on a central problem within the network configuration process, the so-called *cable management problem*. A practical view to this problem is given in the next section. Section 3 presents both exact and heuristic approaches to solve the cable management problem. Section 4 elaborates on the complexity of the cable management problem; in particular it is shown that along with the bundling restriction the cable management problem becomes NP-complete. The last section contains a run-time comparison between the heuristic and exact approaches when solving the cable management problem for a relevant graph class.

## 2 Cable Management

From a network expert's point of view, network configuration is divided into two independent parts:

(*i*) *Physical Network Planning.* On the physical level one is faced with a routing problem, which stems from the customer's premises. Given is a CAD drawing with designated places, the so-called *sockets*, where a certain number of cables must be made available. The network expert has to connect the sockets with wiring rooms by routing the demanded number of cables on cable channels. In general, the goal is to find a wiring that minimizes the overall cable length.

(*ii*) *Logical Network Planning.* This planning level neglects geometrical details; starting point is the readily routed graph created in the physical planning stage. The points of this graph stand for sockets and active network components, the edges represent data- and communication cables. The goal of this planning step is a suitable selection and dimensioning of active components.

Item (*i*) outlines the cable management problem. Actually, cable management takes place on three levels. On the bottommost level all sockets of a single floor are connected with one ore more wiring rooms. A wiring room contains the active network components such as bridges and hubs. On the second level these wiring rooms are connected via a vertical point-to-point connection to so-called secondary (building) wiring rooms; the secondary wiring rooms in turn are connected with a single primary (campus) wiring room.

The network structure on level 1 and 2 always forms a tree, and thus the related cable management task is rather simple. However, on the bottommost level often a complex routing problem has to be solved: A large number of sockets has to be connected such that technical constraints are considered and the overall routed cable is of minimum length. An important technical constraint shall be pointed out here: the

[1] Active components require an electric power supply to work, passive components do not.

bundling restriction. This restriction claims that all cables of a socket must be routed together to a wiring room, i.e., as one single line.

In the following, we will shortly refer to the "cable management problem without bundling restriction" with **CM**; the "cable management problem with bundling restriction" is abbreviated with **CMB**. While **CM** can be solved in polynomial time, section 4 shows that **CMB** is NP-complete.

As mentioned above, physical network planning is subject to CAD drawings of the buildings. Essential information such as the demanded sockets, the length and the maximum capacity of cable channels, or even room names must be attached to these drawings; superfluous information must be hidden. Figure 1 gives an example of such a prepared drawing.

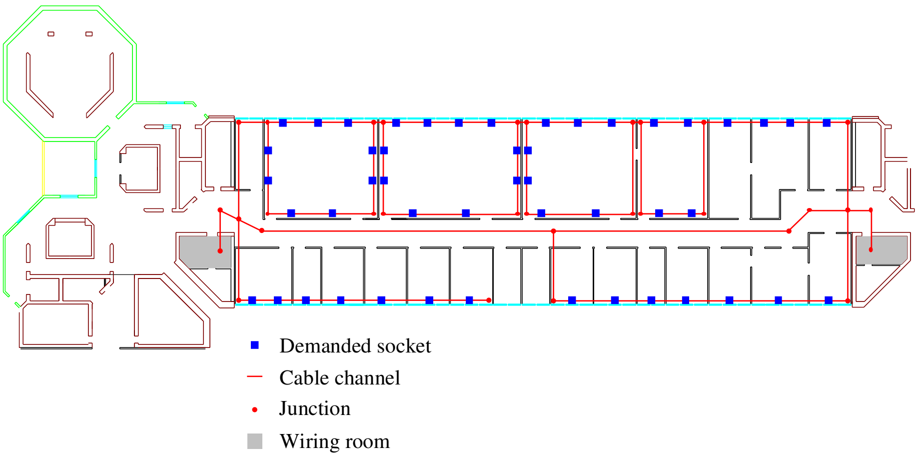


**Figure 1:** *Prepared CAD drawing of a floor with demanded sockets.*

# 3 Solving the Cable Management Problem

This section provides a solution for the cable management problems **CM** and **CMB**. In a first step we will develop a specification of **CM** which relies on the graph-theoretical terms flow network or capacitated network. It becomes clear that solving **CM**, the cable management problem *without* bundling restriction, requires to solve the well-known *minimum cost flow* problem. Hence there exist several algorithms that are able to tackle **CM**; the basic ideas of some of them will be sketched out here. Although these exact algorithms are of a polynomial time complexity, their run-time behavior still is unsatisfactory for our application.[2]

However, we have developed a heuristic approach to solve **CMB**, which will be presented in section 3.3. This approach performs pretty well and produces sufficient results with respect to the cable length optimality criterion.

[2]Pseudo-polynomial algorithms that solve the minimum cost flow run in $O(nm^2CU)$, with $n, m, C, U$ denoting the number of points, the number of edges, the maximum cost value, and the maximum capacity value. There exist also polynomial algorithms running in $O(n^4 log(n))$ .

## 3.1 Mapping CM onto the Minimum Cost Flow Problem

Let us recall the cable management problem. Given are the following three types of nodes: ($i$) sockets $s_1 \dots s_n$, ($ii$) wiring rooms $t_1 \dots t_m$, and ($iii$) junctions $v_1 \dots v_o$.

These nodes are connected by cable channels; the nodes along with the cable channels form a cable channel graph (cf. figure 2). Speaking informally, the solution of **CM** requires that each socket node $s$ is connected to one wiring room node $t$ by routing one or more cables via the cable channel graph. As mentioned in section 2, this routing must happen carefully, considering both cable length restrictions and cable channel capacities. Moreover, the following optimality criterion is stated: The totally routed cable length shall be minimum.

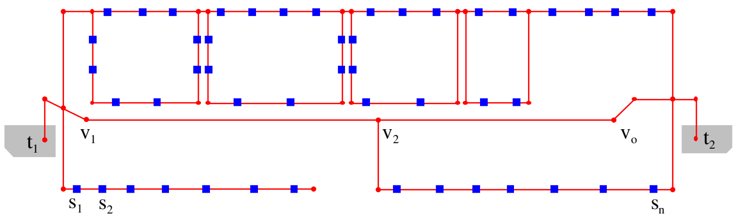


**Figure 2:** *Cable channel graph*

The considerations of this section rely on a few standard definitions of graph theory. For the sake of completeness, the necessary definitions are listed in this place:

1. A *graph* $G$ is a pair $\langle V, E \rangle$ where $V \neq \emptyset$ and $E$ is a set of two-element sets. $V$ is called the set of *points*, $E$ is called the set of (undirected) *edges*. A *directed graph* is a graph whose set of edges, $E$, consists of ordered pairs, which are also called *arcs*. Let $(v_i, v_j)$ be an arc; then $v_i$ and $v_j$ are called the arc's *tail* and *head* respectively.

2. A *flow network* $N$ is a tuple $\langle G, c, s, t \rangle$ comprising a graph $G = \langle V, E \rangle$, two points $s, t \in V$, and a mapping $c : E \rightarrow \mathbf{R}^+$. $c(e)$ is called *capacity* of $e$, $s$ and $t$ are called *source* and *sink* respectively. A mapping $f$ is called *flow* on $N$ if $f$ fulfills the following conditions:

   F1) $\forall e \in E : \; 0 \leq f(e) \leq c(e)$

   F2) $\forall v \in V, v \neq s, t : \; \sum_{e^+=v} f(e) = \sum_{e^-=v} f(e)$. $e^+$ and $e^-$ denote the tail and the head of $e$ respectively.

   Condition F1 claims the capacity restrictions not to be violated. Condition F2 defines the mass balance constraints for all points except the source and the sink.

*Minimum Cost Flow Problem.* Let $\langle G, c, s, t \rangle$ be a flow network. Moreover let $\gamma : E \rightarrow \mathbf{R}$ be a cost function that defines the cost per unit flow on each edge $e \in E$. Objective of the minimum cost flow problem is to minimize $\sum_{e \in E} \gamma(e) f(e)$ subject to the conditions F1 and F2.

*Remarks.* Note that in the minimum cost flow problem as defined here the lower capacity bound is set to zero. Note further that it is assumed that the flow cost vary linearly with the amount of flow.

Key idea when mapping **CM** on the minimum cost flow problem is the interpretation of routed cables as a flow, which can be generated on the cable channel graph. Taking this view, each socket node produces an outflow $\sigma$, which defines the number of cables required at the node, while the wiring room nodes solely have some inflow. For junction nodes a "mass balance constraint" must be fulfilled, i. e., for each junction node the outflow must equal the inflow. The creation of a flow network $N_{CM} = \langle G_{CM}, c_{CM}, s, t\rangle$ and a cost function $\gamma_{CM}$ related to the cable channel graph happens within the following steps.

1. $V_{CM} := \{s_1 \dots s_n\} \cup \{t_1 \dots t_n\} \cup \{v_1 \dots v_o\} \cup \{s, t\}$. I.e., $V_{CM}$ is comprised of the socket nodes, the wiring room nodes, the junction nodes, and an additional source and sink respectively.

2. $E_{CM}$ contains all edges of the cable channels; their capacity values correspond to the original values of the channels. Note that these edges are not directed. Moreover $E_{CM}$ contains for each socket node $s_i$ an arc $(s, s_i)$ and for each wiring room node $t_j$ an arc $(t_j, t)$. The capacity values of the arcs $(s, s_i)$ is equal to $\sigma(s_i)$, the number of cables required at node $s_i$; the capacity values of the arcs $(t_j, t)$ are unlimited.

3. The length of a cable channel is interpreted as the cost function $\gamma_{CM}$ by which a unit of "cable flow" is charged when passed along that channel. With respect to the additional arcs $(s, s_i)$ and $(t_j, t)$, the value of $\gamma_{CM}$ is zero.

Based on the cable channel graph of figure 3, the subsequent figure shows how the mapping comes to effect.

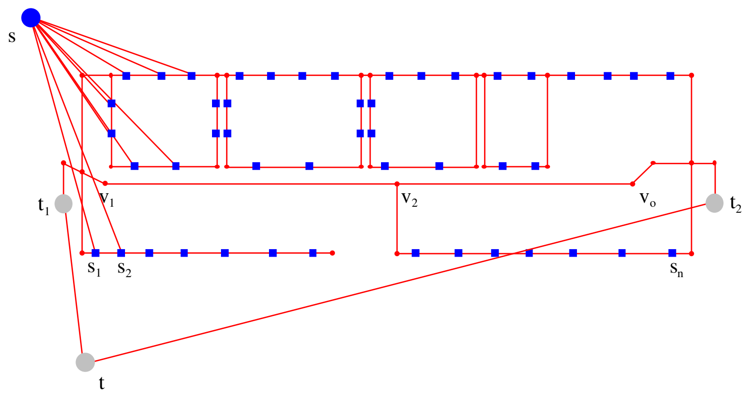


**Figure 3:** *Modified cable channel graph*

Given are the flow network $N_{CM} = \langle G_{CM}, c_{CM}, s, t\rangle$ and the cost function $\gamma_{CM}$ related to some cable channel graph. Then each flow $f$ that minimizes $\sum_{e \in E_{CM}} \gamma_{CM}(e) f(e)$ solves **CM** and vice versa. This assertion follows direct from the mapping.

## 3.2 Algorithms that Solve the Minimum Cost Flow Problem

There exist several algorithms which solve the minimum cost flow problem. Due to the complexity of the problem we cannot discuss these algorithms in detail here. However, we will outline some basic principles on which most of the algorithms rely. This is useful for both getting an idea of the complexity of the problem and assessing the heuristic approach presented in section 3.3.

Many algorithms for solving the minimum cost flow problem combine ingredients of algorithms for shortest path problems and algorithms for maximum flow problems.

- *All Pairs Shortest Path Problem.* Given are a graph $G = \langle V, E \rangle$ and a cost (distance) function $\gamma : E \rightarrow \mathbf{R}^+$. The objective is to compute for each point in $V$ the shortest paths to all other nodes in $G$.
  The algorithm of Floyd-Warshall solves the all pairs shortest path problem and has a runtime complexity of $O(|V|^3)$.

- *Maximum Flow Problem.* Given is a flow network $N = \langle G, c, s, t \rangle$. The objective is to determine a flow $f$ that maximizes $\sum_{e^-=s} f(e) - \sum_{e^+=s} f(e)$.
  The preflow-push algorithm and the algorithm of Malhotra, Kumar, & Maheshewari determine a maximum flow $f$ and have a runtime complexity of $O(|V|^3)$.

An algorithm that solves the minimum cost flow problem is the *cycle-canceling algorithm*. The algorithm first establishes a feasible flow in the network by solving the maximum flow problem. Then it uses shortest path computations to find cycles with negative flow costs in the graph; it then augments flows along these cycles and iteratively repeats the computations for detecting negative cycles and augmenting flows. The correctness of the cycle-canceling algorithm is founded on the following theorem:

*Theorem (Negative Cycle Optimality Condition).* A feasible flow $f$ is an optimal solution of the minimum cost flow problem for a given network $N$ and a cost function $\gamma$ if and only if $f$ satisfies the negative cycle optimality condition: namely, the residual network $N_f$ contains no negative cost cycle. ($N_f$ can be directly derived from $N$ in $O(|E|)$ steps; in first place $N_f$ differs from $N$ with respect to its capacity function $c_f$.)

In the following a pseudo-code representation of the cycle-canceling algorithm is listed.

**begin**
  establish a feasible flow $f$ on $N$;
  compute the residual network $N_f$ from $N$ and $f$;
  **while** $N_f$ contains a negative cycle **do**
    **begin**
    identify a negative cycle $W = ((v_1, v_2), (v_2, v_3), \ldots, (v_n, v_1))$;
    $\delta := min\{c_f((v_i, v_j)) \mid (v_i, v_j) \in W\}$;
    augment $\delta$ units of flow in the cycle $W$;
    compute the new residual network from $N_f$ and $\delta$
  **end**
**end**

*Remarks.* A feasible flow $f$ in the network can be established by solving the maximum flow problem for $N$. Negative cycles can be detected with a modified Floyd-Warshall algorithm. Both steps have a runtime complexity of $O(|V|^3)$. Note that the generic cycle-canceling algorithm is only a pseudo-polynomial-time algorithm since its number of iterations depends on the functions $c$ and $\gamma$.
However, in our cable routing application the following assumptions can be stated: $(i)$ The maximum capacity $c(e)$ of an edge $e$ is bound by $O(|V|)$ since the number of socket nodes is bound by $V$ and each socket node produces at most $max(\sigma) = 4$ units of flow. $(ii)$ The cost function $\gamma$, which represents the lengths of the cable channels, will only take integral values less then 100.

Another well-known representative that solves the minimum cost flow problem is the *out-of-kilter algorithm*. At every iteration, it solves a shortest path problem and augments flow along the shortest paths. Unlike the cycle-canceling algorithm, the out-of-kilter algorithm permits transient violations of the flow capacity.

Both the out-of-kilter algorithm and the cycle-canceling algorithm have been implemented and tested within the system LANeCo. Because of their insufficient run-time behavior we started to develop a heuristic approach to solve **CM**. A result is the subsequently presented algorithm, which establishes a good compromise between the quality of the solution and the performance.

## 3.3 A Heuristic Approach to Solve CM

Our heuristic approach is based on the observation that if the cables are routed via the shortest path from a socket to a wiring room, there exist only a few *bottle-necks* in the graph in many realistic configurations. In this connection, a bottle-neck is an edge whose capacity restriction is violated.
Stated another way, if one forgot about the capacity restrictions on the edges and just used Dijkstra's algorithm to send all flow from the $\{s_1, \ldots, s_n\}$ to $t$, one would end up quite close to the optimal solution for **CM**.

Exactly this is the strategy pursued by our heuristic, which can be outlined by the following three steps.

1. Computation of an initial flow on the uncapacitated network, using Dijkstra's shortest path algorithm.
2. Identification and removal of all bottle-necks from the graph.
3. Re-routing of the excess flow of all bottle-necks on the modified network, using again Dijkstra's algorithm.

In fact, the algorithm presented below tries not only to solve **CM** but the harder problem **CMB**: Recall that all cables from a single socket should be routed on the same path to the same wiring room, thus posing the bundle restriction constraint.[3]

[3]One may also think about $CMB$ as the problem of embedding a star graph into an arbitrary

To precisely describe the algorithm, we enhance the graph definition by a function *path*. $path : \{s_1, \ldots, s_n\} \rightarrow \mathcal{P}(E)$, where the edges in $path(s_i)$ denote a path from $s_i$ to $t$.

We make also use of a function *guests*, which is implicitly defined by *path*. $guests : E \rightarrow \mathcal{P}(\{s_1, \ldots, s_n\})$; $guests(e) = \{s | s \in \{s_1 \ldots s_n\}, e \in path(s)\}$.

**Procedure** Heuristic ($CMB$)
*bn*: List of bottle-necks
*pbn*: List of possible bottle-necks
*new.path*: A path through the graph
*terminate*: Boolean
*gl*: List of sockets

1. **for** $s_i \in \{s_1 \ldots s_n\}$ **do**
   $path(s_i)$ := shortest path for $s_i$

2. $bn := \{\}$
   **for** $e \in E$ **do**
     **if** $f(e) > c(e)$ **then**
       $bn := bn \cup e$

   **for** $e \in bn$ **do**
     remove $e$ from graph

3. **while** $terminate = FALSE$ **do**
     *terminate* :=TRUE
     **for** $e \in bn$ **do**
       $pbn := \{\}$
       $gl := guests(e)$

       **until** $gl = \emptyset \ \vee \ (f(e) \leq c(e))$ **do**
         $s := select\_guest(gl)$
         $gl := gl \setminus \{s\}$
         *new.path* := shortest path from $s$ to $t$
         **if** $\forall\ e' \in new.path$: $f(e') + \sigma(s) \leq c(e')$ **then**
           $path(s) := new.path$
           *terminate* :=FALSE
         **else**
           $pbn := pbn \ \cup \ \{e' | e' \in new.path \ \wedge \ f(e') + \sigma(s) > c(e')\}$

       **if** $f(e) > c(e)$ **then**
         **for** $e' \in pbn$ **do**
           remove $e'$ from graph
           *terminate* :=FALSE
       **else**
         $bn := bn \setminus \{e\}$

graph. From this point of view the capacity restriction correlates to a congestion constraint while the optimum criteria of $CMB$ becomes a dilation criteria.

*Remarks.* ($i$) The *select_guest* function defines an order by which the guests are chosen from a bottle-neck. It turned out to be a good heuristic to select the guests according to the additional length, that a rerouting would cause. ($ii$) Note that within step 3 no new bottle-necks are created. ($iii$) Unlike classical Operations Research algorithms the heuristics points out bottle-necks even if no solution can be found.

*Theorem.* The heuristic algorithm has a time-complexity of $O(|E| * (|E| + n \log n))$.

*Proof Idea.* One may notice that it is sufficient to compute a Dijkstra solution every time an edge is removed from the graph. Therefore in the worst case we remove edges individually from the graph. This results in $|E|$ Dijkstra runs.

Note that the heuristic works for most realistic examples, but it cannot solve problems, where, in order to find a solution, two bottle-necks have to exchange guests.

# 4 On the Complexity of CMB

In this section a complexity result is presented. Clearly, **CM** is in P since it can be solved with several polynomial-time algorithms (cf. section 3.2). On the other hand, as the following theorem states, for the cable management problem with bundling restriction this is not the case.

***Theorem 4.1 (NP-Completeness of* CMB*).*** The cable management problem with bundling restriction, **CMB**, is NP-complete.

***Proof.*** It must be shown that ($i$) **CMB** is in NP, and that ($ii$) a problem known to be NP-complete can be polynomially reduced onto **CMB**.

**CMB** has been introduced as an optimization problem; note that it is sufficient to proof theorem 4.1 for the decision problem variant of **CMB**, where also a maximum length $L$ is specified. Then the question is whether for a given flow network a routing exists that is bound by $L$.

ad ($i$) Let $\Pi$ be an instance of **CMB**. To show that **CMB** is in NP, guess a solution respecting $\Pi$. A solution consists of all paths from the sources $s_i$ to the sink $t$. Check whether the lengths of all paths is less than the maximum length, i.e., if $\sum_{s_i \in \{s_1, \ldots, s_m\}} \sigma(s_i)|path(s_i)| \leq L$, and if this solution complies with the capacity and the bundling restriction respectively. Obviously this can be verified within polynomial time in the input length of $\Pi$.

ad ($ii$) We reduce the Knapsack problem, **KP**, onto **CMB**. The Knapsack Problem is defined as follows: Given are two constants $B$, $W$, and $r$ objects $o_1, \ldots, o_r$, each of which coming with a weight $w_i$ and a benefit $b_i$. The question is whether there exists a subset $S \subseteq \{o_1, \ldots, o_r\}$ such that $\sum_{o_i \in S} b_i \geq B$ and $\sum_{o_i \in S} w_i \leq W$.

First, we define a transformation $f, f \in P$, that maps an instance of **KP** onto an instance of **CMB**. Second, for this $f$ it will be shown that $\Pi \in$ **KP** $\Leftrightarrow f(\Pi) \in$ **CMB**.

*Transformation.* $f(\Pi)$ consists of a flow network and a constant $L$. $L$ is set to $r \cdot B_O - B$,

with $B_O := \sum_{i=1,\dots,r} b_i$.
The graph of the flow network contains a vertex $s_i$ for each object $o_i$. The vertices $s_i$ can be interpreted as sockets each of which having $w_i$ in-going cables. Furthermore, the graph does also contain a vertex $t$, associated with the server, and two auxiliary vertices $\alpha$ and $\beta$. No other vertices are part of the graph.
The edges in the graph are constructed as follows. Each $s_i$ is connected with both $\alpha$ and $\beta$ via edges whose capacities are unlimited. The costs (lengths) of these edges are defined as $\gamma((s_i, \alpha)) = \frac{B_O - b_i}{w_i}$, and $\gamma((s_i, \beta)) = \frac{B_O}{w_i}$. Furthermore $\alpha$ and $\beta$ are connected with $t$, and we set $c((\alpha, t)) = W, c((\beta, t)) = \infty, \gamma((\alpha, t)) = 0$, and $\gamma((\beta, t)) = 0$. No other edges are part of the graph. Figure 4 illustrates such a graph. $f \in P$ since this graph can be constructed in $O(|r|)$.

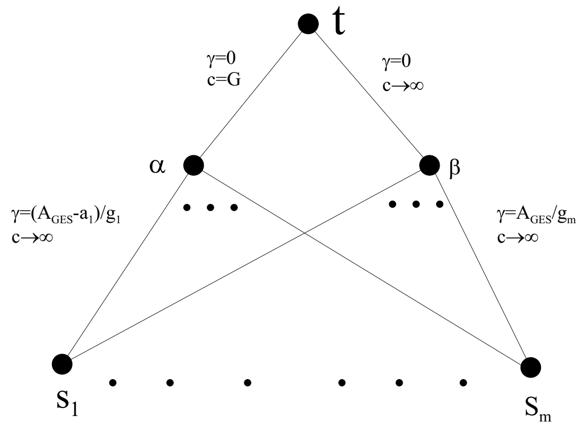


**Figure 4:** *Transforming* KP *onto* CMB

*Equivalence.* It is useful to show first some characteristics of the above graph construction. Any solution of an instance $f(\Pi)$ of **CMB** divides $\{s_1, \dots, s_m\}$ into two sets, $O_\alpha$ and $O_\beta$. $O_\alpha$ comprises all sockets routed via $\alpha$, while $O_\beta$ comprises those sockets routed via $\beta$. In these terms the overall cable length of a solution is

$$\sum_{s_i \in O_\beta} w_i \cdot \frac{B_O}{w_i} + \sum_{s_i \in O_\alpha} w_i \cdot \frac{B_O - b_i}{w_i} = r \cdot B_O - \sum_{s_i \in O_\alpha} b_i$$

Notice that the capacity restriction can only be violated on the edge $(\alpha, t)$.

To show that $\Pi \in$ **KP** $\Leftrightarrow f(\Pi) \in$ **CMB** we will proof that $O_\alpha$ is equivalent to the solution $S$ of the Knapsack Problem.

**KP** $\Rightarrow$ **CMB**. Let $\Pi$ be an instance of **KP** with solution $S$. To solve $f(\Pi)$ we route all $s_i$ with $o_i \in S$ via $\alpha$ and all other sockets via $\beta$. Now the following holds:
$(i)$ $r \cdot B_O - \sum_{s_i \in O_\alpha} b_i \leq L := r \cdot B_O - B$, because $\sum_{o_i \in S} b_i \geq B$, and
$(ii)$ $f(\alpha, t) \leq W$, because $\sum_{o_i \in S} w_i \leq W$. Hence, $f(\Pi) \in CMB$.

**CMB** $\Rightarrow$ **KP**. Now let $f(\Pi)$ be an instance of **CMB**, and let $O_\alpha$ and $O_\beta$ define a solution for this instance. Then $S = \{o_i | s_i \in O_\alpha\}$ defines a solution for the instance $\Pi$ of **KP**:
$(i)$ $\sum_{o_i \in S} w_i \leq W$, because of the capacity restriction on edge $(\alpha, t)$, and
$(ii)$ $\sum_{o_i \in S} b_i \geq B$, because $r \cdot B_O - \sum_{s_i \in O_\alpha} b_i \leq L := r \cdot B_O - B$. Hence, $\Pi \in KP$. $\diamond$

# 5 Some Test Results

Aside from processing graphs that are descended from real-world applications, we have implemented a test bench to compare our different approaches. Within this test bench, grid graphs are generated randomly, where parameters like number and contribution of sockets as well as server rooms can be altered.

We generated two different sets of cases, one consisting of graphs with no or only very few bottlenecks ($\Phi_0$) and one with networks, where about 1 percent of the edges were bottlenecks ($\Phi_1$). These are quite realistic assumptions.

$\sigma(s_i)$ has been set to 1 for all sockets $s_i$, so that the Out-of-Kilter and the Cycle-Cancelling algorithms are able to find the optimal solution for **CMB**. Our tests were performed on a Pentium 166; the programs are written in Common Lisp.

The tests were of interest with respect to the following questions:

- What about the quality of the solutions?
  For all solvable network in $\Phi_0 \cup \Phi_1$ the heuristic approach found a solution. This solution was never, concerning the overall length, more than 3 percent away from the optimal solution.

- What about the runtime behaviour?
  It is mainly interesting to examine the run-time behaviour for some inputs from $\Phi_1$. The Out-of-Kilter algorithms proofed to be superior to the Cycle-Cancelling algorithm but even the Out-of-Kilter is much too slow for our desired graph size (1000-10000 vertices). The heuristic algorithm could handle this input size quite well.

- When does the heuristic approach start to fail?
  Related to **CM** ($\sigma(s_i) = 1 \; \forall i$) the heuristic approach will fail, if more than 20 percent of the edges in the network are bottlenecks.

# 6 Conclusion

A central problem when configuring local area networks is the cable management problem. By defining a mapping of the cable management problem onto the minimum cost flow problem we were able to employ existing graph algorithms to exactly solve the cable management problem.

However, we made the experience that the runtime complexity of exact algorithms is not acceptable when solving real-world cable management problems. As a consequence we have developed alternative heuristic algorithms. Empirical tests with real cable channel systems have shown that the heuristic approach efficiently computes optimum or nearly optimum solutions in most cases.

Aside from the presented heuristic we have also developed a genetic and a simulated annealing algorithm. Both can handle $\Pi_b$ quite well, but they are here no match for the heuristic approach.

The goals in our future research concerning the cable management problem are twofold: $(i)$ improvement of the heuristic approach presented here, and $(ii)$ development of concepts that tackle synthesis jobs, like the modification of the cable channel graph.

# References

[1] Ravindra K. Ahuja, Thomas L. Magnanti and James B. Orlin. *Network flows: Theory, Algorithms and Applications.* Prentice Hall, Englewood Cliffs, N. J., 1993.

[2] Mokhtar S. Bazaraa and John J. Jarvis. *Linear programming and network flows.* Wiley, New York, 1977.

[3] T. B. Boffey. *Graph theory in operations research.* Macmillan Computer Science Series, London, 1982.

[4] Nicos Christofides. *Graph theory: an algorithmic approach.* Academic Press, London, 1986.

[5] Dieter Jungnickel. *Graphen, Netzwerke und Algorithmen.* Wissenschaftsverlag, Mannheim, 1994.

[6] Edward Minieka. *Optimization for networks and graphs.* Dekker, New York, 1978.

[7] Heiner Müller-Merbach. *Operation Research: Methoden und Modelle der Optimalplanung.* Vahlen, München, 1988.

[8] Thomas Ottmann and Peter Widmeyer. *Algorithmen und Datenstrukturen.* Wissenschaftsverlag, Mannheim, 1993.

[9] Christos H. Papadimitriou and Kenneth Steiglitz. *Algorithms and Complexity.* Prentice Hall, Englewood Cliffs, N. J., 1982.

[10] Günther Ruhe Dordrecht. *Algorithmic aspects of flows in network.* Kluwer Academic Publ., 1991.