Exact Algorithms for the Minimum-Cost Embeddings of Reliable Virtual Private Networks into Telecommunication Networks

Rolf H. Möhring
Fachbereich Mathematik, Technische Universität Berlin, Germany

Martin Oellrich
Technologiezentrum Darmstadt, Deutsche Telekom AG, Germany

Andreas S. Schulz
Sloan School of Management, MIT, Cambridge, USA
Fachbereich Mathematik, Technische Universität Berlin, Germany

Summary: When establishing a virtual private network (VPN), a telecommunication provider has to balance two conflicting goals. On the one hand, the customer requires reliability, on the other hand, the provider has to keep costs at a minimum. This article presents two related approaches for the optimal computational solution of the embedding of a VPN which is crucial in modern network planning.

1. Introduction

In telecommunications engineering, there is more to do than accelerating transmission or constructing safer encryption codes. Raising a large technological system typically shows an increasing risk of failure due to unforeseen influences, inherent inconsistencies, or lack of control despite all efforts. Keeping their networks stable and operable are the primary goals of all telecommunication providers. Therefore, safety measures are being called for by their customers. A typical example in this context is the routing of a virtual private network (VPN).

A VPN, as the name indicates, is a network that appears to be exclusively controlled and operated by a customer company alone. In reality, it consists of a number of permanent lines that are leased from a carrier. So control also lies upon the latter to a significant extent. And for this reason, the carrier has to provide these lines subject to a reliability demand. This means that one failure in the carrier's network may at most cause one failure on the VPN level, in the customer's network. Technically speaking, the different leased lines must be routed *disjointly* through the carrier's network.

In order to be competitive, the carrier must also consider the cost side of this demand. So naturally, a reliable embedding of minimum cost is required. The problem of finding this optimum is the mathematical task we treat in this paper.

The basis for this paper was a joint R&D project between the group Combinatorial Optimization and

Graph Algorithms at TU Berlin and the Network Optimization group at Deutsche Telekom AG that has taken place from July 1995 through June 1996. It was initiated by earlier investigations of the Deutsche Telekom AG [6]. One of the authors has written his Diploma thesis [4] about this project containing all presented aspects in detail, along with many examples and important extensions. A more elaborated journal paper [3] is in preparation, featuring more concepts and details of this research.

Terminology

Throughout this text, we will refer to the physical network as the *base graph* $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, and to the requested connections as the *demand graph* $\mathbf{H} = (\mathbf{T}, \mathbf{D})$, $T \subseteq V$. The elements of V and E are called the nodes and edges of G, respectively. An element $d = \{s, t\} \in D$ is called a *demand*. Both input graphs are multi-graphs which may contain parallel edges or demands, respectively, but no loops. For examples of such data, see Figures 1 and 2.

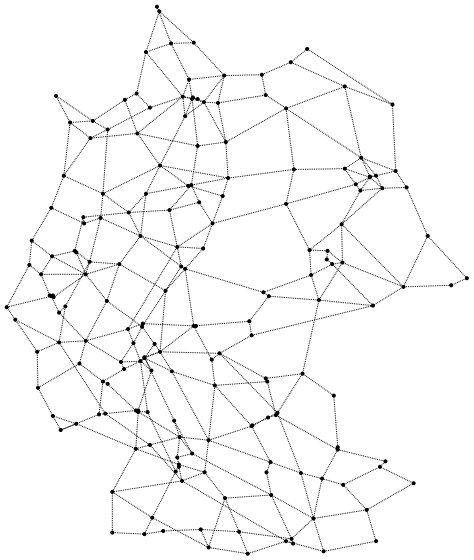

Figure 1: A physical network (base graph)

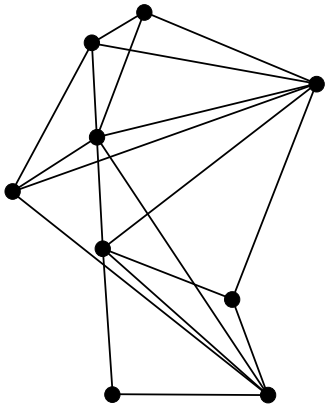

Figure 2: A virtual private network (demand graph)

In our project, we investigated the mathematical and algorithmic aspects of the so-called *Minimum Cost Disjoint Paths Problem* in two versions, node- and edge-disjoint, respectively. The latter reads as follows:

Minimum Cost Disjoint Paths Problem (MCDPP) (edge-disjoint version)

Given a base graph $G = (V, E)$ with associated edge costs $c : E \to Q^+$ and a demand graph $H = (T, D)$ where $T \subseteq V$. For each demand $d = \{s, t\}$, find an embedding in G as a simple s-t-path $E^d \subseteq E$ such that all paths are mutually edge-disjoint and the total cost of all edges used is a minimum:

$$\begin{array}{ll} \min & \sum_{d \in D} \sum_{e \in E^d} c(e) \\ \text{s. t.} & E^d \cap E^{d'} = \oslash \quad \text{for all } d, d' \in D, d \neq d'. \end{array}$$

The node-disjoint version differs from this one in that the paths may at most mutually share their terminals. For these problems, we ask for only *one* optimal solution and do not require to produce each of them, in case there are more than one.

An abundance of theoretical results is available on the topic of disjoint routing. The most important ones are the *NP*-hardness of the problem, and the existences of feasible solutions as well as polynomial time algorithms in highly restricted settings. A large number of them are surveyed in [4].

2. Lower Bounds on Costs

The key ingredients we need for designing a suitable search method are strong lower and upper bounds on the optimal objective function values. In our case we generate feasible solutions and hence upper bounds by means of a suitable heuristics guided by structural information that comes along with our lower bounds.

In the following discussion, we concentrate on the computation of lower bounds. We denote a lower cost bound on the optimal objective value of an *(MCDPP)* instance (G, H) with cost function c by $\mathbf{LB_c}(G, H)$.

2.1 The Model

The min-cost VPN embedding problem may be considered as a special case of the general multi-commodity flow problem. The latter consists of a certain number of individual minimum cost flow problems, tied together by so-called bundle constraints (for an introduction see [1]). These constraints typically limit the amount of flow on an arc of the underlying network up to a joint capacity value. In our case, this value is uniformly equal to one on all the arcs, in addition to all arc flows being restricted to the binary values of $\{0, 1\}$. This forces the desired disjointness.

We now model the min-cost VPN embedding problem as an integer linear program. To this end, we formalize three types of constraints, which are reflected in the corresponding (in)equalities in the model:

(i) all demands are flows between their associated terminals,
(ii) all edges have got bundle capacities of one,
(iii) all flow values are binary.

In order to get a network $N = (V, A)$ of directed arcs, we split all edges into two anti-parallel arcs with costs equal to $c(e)$:

$$A := \{\, (u,v) \mid \{u,v\} \in E \,\} \cup \{\, (v,u) \mid \{u,v\} \in E \,\}.$$

Each demand $d \in D$ represents a commodity, and the vector of all variables describing its flow values is denoted by x^d. For each $d = \{s,t\}$, the *balance vectors* $b^d \in \{-1,0,1\}^{|V|}$ are given by:

$$b^d(v) := \begin{cases} 1 & \text{if } v = s, \\ -1 & \text{if } v = t, \\ 0 & \text{otherwise.} \end{cases}$$

Finally, $M \in \{-1,0,1\}^{|V| \times |A|}$ denotes the node-arc incidence matrix where

$$M_{va} := \begin{cases} 1 & \text{if } a = (u,v) \text{ for some } u, \\ -1 & \text{if } a = (v,u) \text{ for some } u, \\ 0 & \text{otherwise.} \end{cases}$$

The edge-disjoint model of *(MCDPP)* now reads as follows:

Model (ILP)

$$\begin{array}{lll} \min & \displaystyle\sum_{d \in D} \sum_{a \in A} c(a) x^d(a) & \\ \text{s. t.} \quad \text{(i)} & M x^d = b^d & \text{for all } d \in D \\ \text{(ii)} & \displaystyle\sum_{d \in D} (x^d(u,v) + x^d(v,u)) \le 1 & \text{for all } \{u,v\} \in E \\ \text{(iii)} & x^d \in \{0,1\}^{|A|} & \text{for all } d \in D. \end{array}$$

Here, (i) models flow conservation (Kirchhoff's Law), and (ii) the joint capacity constraints.
The node-disjoint model differs from the above only in constraint (ii) which is transformed to nodes instead of edges in order to control the flow through nodes.

2.2 Lagrangian Relaxation

We assume that the reader is familiar with the basics of Lagrangian relaxation, subgradient optimization, and their properties, see [1, 2]. Lagrangian relaxation applied to the disjointness constraint (ii) of Model *(ILP)* reads:

Model ($\mathbf{LR}_\lambda$)

$$\begin{array}{lll} \min & \displaystyle\sum_{d \in D} \sum_{\{u,v\} \in E} (c(u,v) + \lambda(u,v))(x^d(u,v) + x^d(v,u)) \; - \sum_{\{u,v\} \in E} \lambda(u,v) & \\ \text{s. t.} \quad \text{(i)} & M x^d = b^d & \text{for all } d \in D \\ \text{(iii)} & x^d \in \{0,1\}^{|A|} & \text{for all } d \in D. \end{array}$$

Now for any value of vector $\lambda \ge 0$, the remaining problem is to find a minimum cost flow for all demands $d \in D$ *independently* from each other with respect to the new cost function $c+\lambda$. Apparently, this is equivalent to $|D|$ individual shortest path problems.

3. The Branch-and-Bound Method

Our first solution method is based on the branch-and-bound paradigm. For an introduction to this principle, see for instance, [5].
We note that a branch-and-bound approach is well suited for the solution of *(MCDPP)*: we have fast methods for computing good lower bounds, and we are able to partition each solution set into disjoint subsets representing again *(MCDPP)* type problems.

3.1 Branching Strategy

We can achieve a disjoint partitioning of the partial solution set in a search tree vertex by observing that every demand $d = \{s, t\}$ must be embedded using exactly one edge e incident to its terminal s. Hence, we choose every neighboring edge e once as a candidate for the embedding where we have chosen the terminal s at random from the two terminals of d. Then, we partition the set of all solutions into subsets differing by these edges e which are respectively added to the embedding of d (and thereby excluded from the embeddings of all other demands).

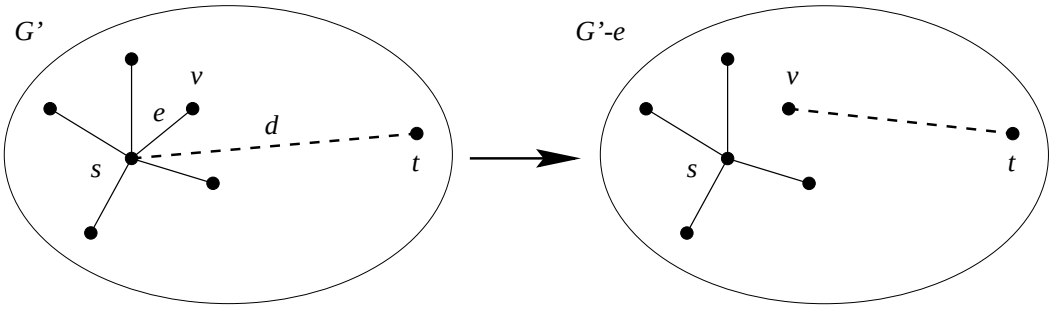

Figure 3: Reduced problem ($G' - e,\ H' - d + \{v, t\}$) after embedding edge e for demand d.

The node-disjoint case differs only slightly in that we reduce the base graph to $G' - v$ instead of $G' - e$.

3.2 Bounding Subproblems

We test whether a subproblem (G', H') at hand can possibly deliver a global optimal solution by comparing

$$(*) \qquad cost(E \setminus E') + LB_c(G', H') < cost(S_{best})$$

where E, E' are the edge sets of G, G', respectively. The term $E \setminus E'$ means all edges hitherto embedded for some demand, and $cost(S_{best})$ is the cost of the actual found best solution S_{best}. Obviously, this is not the case otherwise.

4. The Backtracking Method

Our second method is related to branch-and-bound. It partitions the set of all solutions into smaller subsets, yet the divisions are made differently. The general idea is to simultaneously reduce G *and*

H in every step, thereby recursively defining a search tree. For this purpose, we successively embed entire paths which are delivered by an appropriate path generation sub-algorithm, one at a time.
On every level of the search tree, we attempt to solve an *(MCDPP)* type problem (G', H') by embedding one demand from H' successively by all paths that are possible in G'. With each of these paths, we define a subproblem by removing its nodes or edges from G' (depending on the disjointness mode) and the embedded demand from H'. The bounding works analogously to equation $(*)$.

5. Conclusion

Confronted with a real-world problem of the Deutsche Telekom AG, we have investigated the optimization problem *(MCDPP)* on arbitrary base and demand graphs in both versions, edge- and node-disjoint. Due to their *NP*-hardness, we pursued exact approaches lacking a polynomial time guarantee.
We have set up and implemented a method for computing lower bounds on the optimal objective value of *(MCDPP)*, representing the total embedding cost. They are the core ingredients in both practical algorithms we have devised for obtaining optimal solutions to the embedding problems.
Subsequently, we have developed two combinatorial codes, one using a branch-and-bound method and a backtracking type one. While the former approach had to be carefully adapted in order to actually produce disjoint paths, the backtracking method is based on entire paths generated by an appropriate subroutine.
Both methods have the advantage that execution can be terminated at any time while retaining the best feasible solution hitherto found. This may be automated by finishing when the total cost $cost(S)$ of a solution S drops below a certain percentage p above the global lower cost bound $LB_{\mathcal{C}}(G, H)$.
We performed extensive experiments on several example classes which are from or close to real-world situations. Throughout all our test runs, we have used the lower cost bounds $LB_{\mathcal{C}}$ that were always taken from the Lagrangian relaxation approach using subgradient optimization.
We have benchmarked both of our algorithms against the standard LP and IP solver cplex, version 3.0. This is a general tool for arbitrary optimization problems encoded by linear models. It does not explicitly take into account the graph-theoretical structure of our problem *(MCDPP)*, but exclusively operates with the model variables. We observed in general that our algorithms mostly excelled in performance over cplex. Some significant results can be seen in Table 5.below. Note that, although in the cases considered feasible solutions always exist, this need not necessarily be true in general. Methods for checking the existences of feasible solutions are described in [3, 4].
The code developed and implemented in this project has been integrated into a toolbox which is used by the Deutsche Telekom AG.

Acknowledgments

We wish to acknowledge Joachim von Puttkamer, Wilfried Wieser, and Eckart Wollner from the Center of Technology of the Deutsche Telekom AG who have identified the problem and initiated the joint project. Also, we like to thank Olaf Jahn of the TU Berlin who has had a considerable part in the implementation process.

time [min]	size of base graph	*back-tracking*	*branch-and-bound*	cplex
example6e	185	0.0	0.0	0.3
example6n	nodes	0.1	0.3	0.2
example19e	354	1.5	2.0	8.1
example19n	edges	2.4	0.9	16.9
example42e		0.2	0.1	60.1
example42n		0.7	0.3	20.9
example117e	676 n. 1107 e.	2.6	2.7	> 500
ger17e	164	1.4	7.1	6.5
ger17n	nodes	0.0	0.6	5.4
ger73e	386	0.1	0.2	186.6
ger73n	edges	0.0	0.0	8.4
ger16e	434	2.3	8.2	29.5
ger16n	nodes	19.3	33.8	22.1
ger28e	978	3.6	13.5	93.9
ger28n	edges	> 200	94.8	151.8
usa7e	176	0.0	0.2	0.2
usa7n	nodes	0.5	0.6	2.8
usa12e	314	0.1	0.1	2.7
usa12n	edges	0.4	0.1	2.0
usa26e	617	0.2	0.1	83.8
usa26n	nodes	8.4	2.3	100.0
usa37e	1039	2.8	1.8	304.2
usa37n	edges	0.1	0.0	182.7

Table 1: Running times of our *(MCDPP)* codes compared to cplex 3.0. The postfix 'e' refers to the edge-disjoint case, 'n' to the node-disjoint one while the decimals form the numbers of demands. Entries of "0.0" mean "below 6 seconds." All times were measured on a Sun SPARC 20 machine under SunOS 4.1.4.

time [sec]	cheapest paths		Lagrangian		LP-relaxed		optimum	
	bound	time	bound	time	bound	time	value	time
example6e	3567	0.0	3643.0	0.2	3643.0	12.7	3643	17.8
example6n	3567	0.0	3673.5	0.5	3673.5	10.3	3677	11.4
example19e	6918	0.0	7688.0	1.4	7688.0	483	7688	483
example19n	6966	0.0	8303.1	1.2	8303.5	662	8462	1011
example42e	8263	0.0	8707.0	1.5	8707.0	3605	8707	3605
example42n	8263	0.0	9102.6	1.7	9102.7	1237	9105	1251
example117e	10478	0.03	11302.9	4.2	11303.0	$> 10^4$	11303	$> 10^4$
ger17e	54215	0.0	56336.5	1.3	56340.5	208	56467	390
ger17n	54679	0.0	58949.0	0.3	58949.0	321	58949	321
ger73e	88969	0.0	92132.0	1.7	92132.0	11198	92132	11198
ger73n	89601	0.0	98146.0	1.1	98146.0	502	98146	502
ger16e	53321	0.0	55265.8	2.9	55266.0	1771	55266	1771
ger16n	53423	0.0	57024.2	2.4	57086.0	1323	57086	1323
ger28e	80356	0.0	83094.8	4.6	83095.0	5634	83095	5634
ger28n	80125	0.0	87348.0	5.3	87404.4	8903	87498	9108
usa7e	60810	0.0	61498.8	0.6	61499.0	11	61499	11
usa7n	61214	0.0	66354.0	0.6	66398.0	8	69942	168
usa12e	67835	0.0	69898.9	0.5	69899.0	68	69899	162
usa12n	67835	0.0	73092.4	0.7	73104.7	43	74556	121
usa26e	104293	0.0	109493.7	3.9	109494	5029	109494	5029
usa26n	105108	0.0	123412.4	3.5	123601.1	2659	123835	5997
usa37e	105412	0.0	112486.0	2.1	112486.0	18036	112486	18252
usa37n	105412	0.0	121550.0	2.0	121550.0	10961	121550	10961

Table 2: Comparison of different lower cost bounds: values and computation times are given. The LP-relaxed and optimum bounds were computed with cplex, the cheapest paths with the same Dijkstra-subroutine underlying the Lagrangian relaxation. It can be seen that the Lagrangian bounds represent a good compromise between accuracy and time.

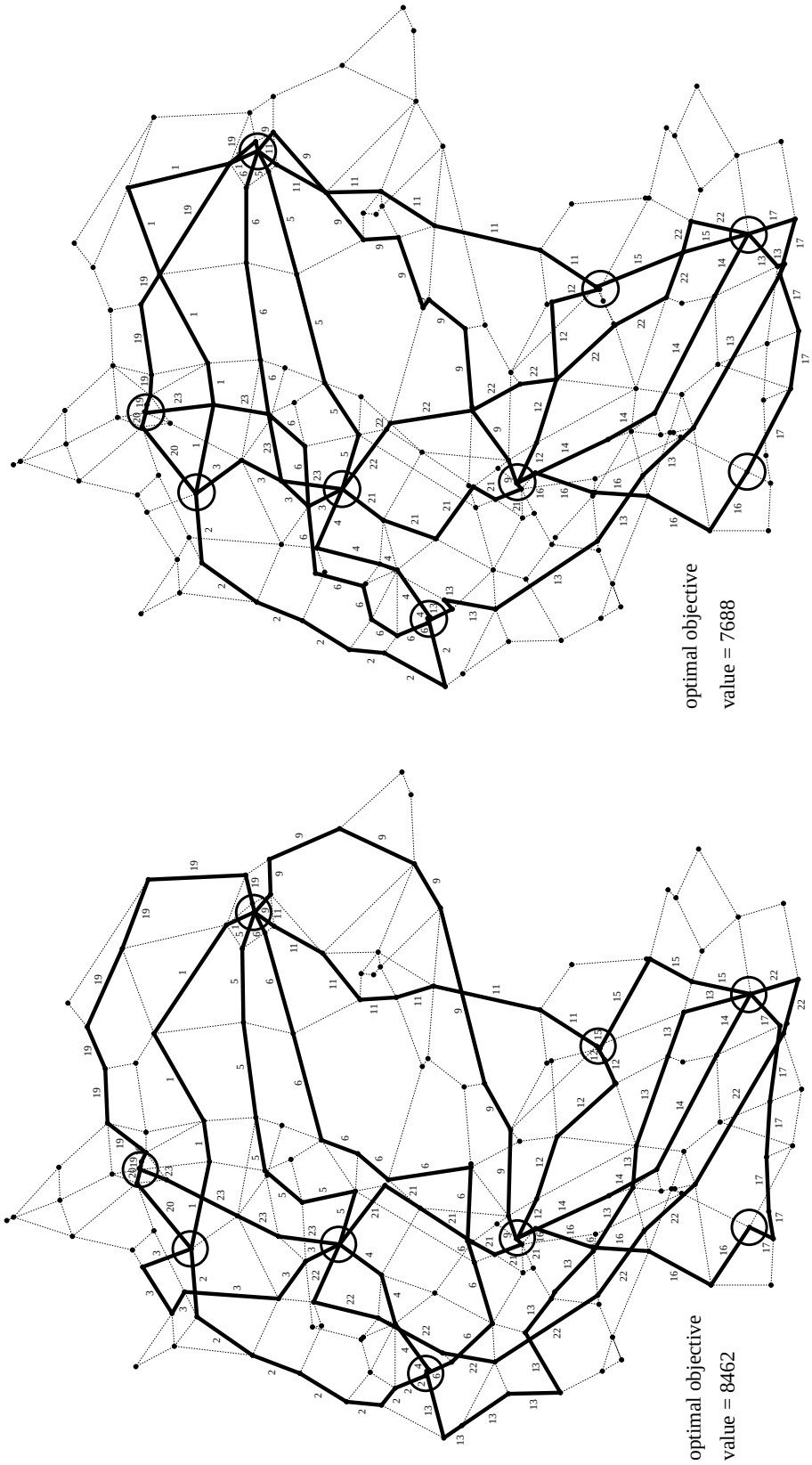

Figure 4: Optimal solutions to *(MCDPP)* on the example given in Figures 1 and 2. Left: node-disjoint case, right: edge-disjoint case.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin: *Network Flows*, Prentice Hall, Englewood Cliffs (1993).

[2] M. Held, P. Wolfe, and H. Crowder: Validation of Subgradient Optimization, in *Mathematical Programming* 6 (1974), pp. 62–88.

[3] R. H. Möhring, M. Oellrich, and A. S. Schulz: Optimal Embeddings of Reliable Virtual Private Networks into Telecommunication Networks, in preparation.

[4] M. Oellrich: Algorithms for the Construction of Reliable Network Platforms in Telecommunication Networks, unpublished Diplomarbeit, Fachbereich Mathematik, Technische Universität Berlin (1997).

[5] C. H. Papadimitriou and K. Steiglitz: *Combinatorial optimization: algorithms and complexity*, Prentice Hall, Englewood Cliffs (1982).

[6] W. Wieser: Planning of Survivable Network Platforms, Deutsche Telekom AG, Preprint (1996).