Design of OSPF networks using subpath consistent routing patterns

Peter Brostrom and Kaj Holmberg

Design of OSPF Networks using Subpath Consistent Routing Patterns

Peter Broström and Kaj Holmberg
Department of Mathematics
Linköping Institute of Technology
SE-581 83 Linköping
Sweden

Abstract

We address the problem of designing IP networks where the traffic is routed using the OSPF protocol. Routers in OSPF networks use link weights set by an administrator for determining how to route the traffic. The routers use all shortest paths when traffic is routed to a destination, and the traffic is evenly balanced by the routers when several paths are equally short. We present a new model for the OSPF network design problem. The model is based on routing patterns and does not explicitly include OSPF weights. The OSPF protocol is modeled by ensuring that all pairs of routing patterns are *subpath consistent*, which is a necessary condition for the existence of weights. A Lagrangean heuristic is proposed as solution method, and feasible solutions to the problem are generated using a tabu search method. Computational results are reported for random instances and for real-life instances.

Keywords: *Internet Protocol, OSPF, Network design, Lagrangean relaxation, Subpath consistency*

1 Introduction

The Open Shortest Path First protocol (OSPF) is one of the most commonly used Interior Gateway Protocols (IGPs) for routing data traffic in Internet Protocol (IP) networks. OSPF is a dynamic protocol where each router has access to a database containing the state of each link and a weight for each link. The database is used by the routers when they decide how the traffic is sent through the network. OSPF is said to be a link state based protocol, since only link related information is used for determining the routing.

A network operator controls the traffic of an OSPF network by assigning suitable values to the link weights. The routers use the link weights for determining the shortest paths to each destination. A router stores all outgoing links (also called interfaces) that belong to a shortest path to a destination in a routing table, and the router alters between these links in a round-robin fashion when traffic is sent to the destination. This means that all shortest paths to the destination are used. A widely accepted approximation is that the traffic is evenly balanced on all outgoing links that belongs to a shortest path, and this approximation is called the Equal-Cost Multi-Path principle (ECMP). A more detailed description of the OSPF protocol can be found in Moy (1998).

In this paper, we consider the problem of designing OSPF networks at minimal cost. We are given the locations of routers, a set of potential links between these routers, and a set of traffic demands. The task is to decide the design of the network and to assign weights to the links such that all traffic demands can be satisfied without violating the installed link capacity on any link. The demands should be routed in accordance with the OSPF protocol. We assume that ECMP splitting applies and that all routers has access to identical databases.

As far as we know, the first optimization model for the design of OSPF networks problem was given in Bley, Grötschel, and Wessäly (2000). A solution to their model is capable of routing a certain percentage of each demand even if a router or a link fails, so network survivability is considered. The ECMP principle is not modeled since the weights are assumed to give rise to unique shortest paths. Holmberg and Yuan (2004) presented the first network design model where the traffic is routed in accordance with the ECMP principle. A sequential method and a weight based search method are proposed as heuristical solution methods. A drawback with this formulation is that ECMP is modeled by introducing several groups of variables, and the model grows in size due to these variables. The ECMP splitting rule is modeled without introducing any additional variables in Broström and Holmberg (2006), where a multiobjective mixed-integer model for the design of survivable OSPF networks is presented. Similar ECMP constraints have independently been developed in De Giovanni, Fortz, and Labbé (2005). ECMP is modeled by introducing a single group of continuous variables in Tomaszewski, Pioro, Dzida, and Zagozdzon (2005), where a mathematical model for the OSPF flow allocation problem is proposed.

All models discussed above are mixed-integer models, and have in common that the link weights are represented by integer variables, and that a binary shortest path variable is set to one if a link belongs to a shortest path towards a destination, and zero otherwise. The shortest path variables and the weight variables are connected by constraints that use a big-M parameter. This parameter makes the linear relaxation weak, and this could be a problem when solution methods are designed. There have for example been unsuccessful attempts of using Lagrangean relaxation on such models.

We present in this paper a new model for the OSPF network design problem. The model is based on routing patterns and does not explicitly include link weights. Since the big-M parameter originates from the link weights, difficulties related to this parameter can be avoided. Previous models use the link weights for ensuring that the demands are routed in accordance with the OSPF protocol, and since our model does not include any weights, we have to find other ways to ensure that the routing patterns are obtainable in an OSPF network.

The question of whether or not a set of desired routing patterns is realizable in an OSPF network has been studied in e.g. Ben-Ameur and Gourdin (2003), Broström and Holmberg (2008) and Broström and Holmberg (2007). These papers contain necessary conditions for the existence of weights, and the routing patterns can not be obtained simultaneously if any of these conditions is violated. One condition is based on subpath consistency, and it states that two patterns must contain identical routing paths between two routers if both patterns contains some routing path between the two routers. A pair of routing patterns that satisfy this condition for all pairs of routers is called subpath consistent. The routing protocol is in this paper modeled using constraints that ensure subpath consistency. This necessary condition has previously been used for the flow allocation problem in Staehle, Köhler, and Kohlhaas (2000), and for network design problems with shortest path routing in Bley and Koch (2002) (point-to-point demands) and Prytz (2002) (multicast demands). Note however that these papers consider non-bifurcated shortest path routing, i.e. routing on unique shortest paths.

As shown in the next section, subpath consistency is not a sufficient condition for the existence of weights, so our model is an approximative model for the OSPF network design problem. An optimal solution to our model is an optimal solution to the the OSPF network design problem if some routing weights yield shortest paths that are identical to the optimal routing patterns. If there are no such weights, the optimal solution to our model provides a lower bound of the optimal objective function value of the OSPF network design problem. We consider solving our model by Lagrangean relaxation and subgradient optimization, and feasible solutions to the OSPF network design problem are generated using a tabu search method. (A Lagrangean approach has previously been used for a closely related problem in Bley (2003).)

The basic conditions for this work are as follows. N is the set of nodes, E is the set of bidirected links, and C is the set of commodities. Each commodity describes a demand of data communication, and our goal is to design a minimum cost network from the bidirected graph $G = (N, E)$, such that all demands can be satisfied without violating the link capacity on any link. Each bidirected link $(i, j) \in E$ has a fixed charge p_{ij}, and a number of optical fibers with capacity u_{ij} can be installed at unit cost f_{ij}. If a bidirected link is installed between two nodes, the link can be used for routing traffic in both directions between the endnodes. The sum of traffic may however not exceed the link capacity. There is also an operating cost that depends on the usage of a link, and the unit cost for link (i, j) is c_{ij}. Each commodity $k \in C$ is represented by a triplet (o_k, d_k, h^k), and the demand volume h^k should be routed from the origin, node o_k, to the destination, node d_k, in accordance with the OSPF protocol. We must also decide how to route the traffic through the network, and this is done by choosing link weights that routers use in their shortest path computations. The weights are directed, so different values can be assigned to opposite directions of each bidirected link. Therefore, we let A be the set of directed links obtained by replacing each undirected link (i, j) in E by the two directed antiparallel links (i, j) and (j, i).

The outline for this paper is as follows. The next section is used for describing routing patterns more in detail, and for defining subpath consistency more precisely. The mathematical model for the OSPF network design problem is presented in Section 3. A Lagrangean heuristic is proposed as solution method, and the method is described in Section 4. Section 5 contains results from computational tests. Concluding remarks are given in the last section.

2 Routing patterns

A routing pattern contains information about the paths used when traffic is routed between pairs of nodes. In principle, a routing pattern could contain the paths used by a single commodity, i.e. one or several paths from one origin to one destination, but a routing pattern could also contain all routing paths to a certain destination (or all paths from a certain origin). When all routing patterns are symmetric, i.e. when the same path should be used in both directions between two nodes, routing patterns could be represented by a set of undirected paths. If the routing patterns are non-symmetric, however, it is necessary to specify directed routing paths. The case with symmetric routing patterns can also be handled using directed paths, namely by specifying two oppositely directed paths between pairs of nodes, see Ben-Ameur, Michel, Liau, and Gourdin (2001).

The question of whether or not a set of prespecified routing patterns can be realized in an OSPF network has previously been studied by a few research groups. A routing pattern can only be realized in practice if it is identical to a shortest path graph with respect to some set of link weights. If a pattern contains several paths between two nodes, all these paths should have the same minimal sum of link weights, while any other path between the two nodes should have a larger sum of weights. We say that a set of weights is compatible with a routing pattern if these requirements are satisfied. The weights should however be compatible with all routing patterns, so we call such weights *compatible weights*. Different routing patterns may contain contradictory information, so it is not certain that compatible weights exist.

Ben-Ameur and Gourdin (2003) study the case when each routing pattern consists of a single undirected path, which means that the same path should be shortest in both directions between the connected node pair. Several necessary conditions for the existence of compatible weights are presented, and one of them states that the set of routing paths must satisfy *the suboptimality condition.*

Definition 1 *A set of routing paths P satisfies the suboptimality condition if, for all pairs of routing paths p_1 and p_2 in P having two nodes i and j in common, p_1 and p_2 share the same subpath between i and j.*

Ben-Ameur et al. (2001) states that “the suboptimality condition excludes traffic load balancing”, and that “routing patterns satisfying the suboptimality condition are necessarily symmetric”, so suboptimality is not directly applicable for the case that we consider. It is however possible to generalize the concept of suboptimality for the case with non-symmetric routing patterns and load balancing, see Broström and Holmberg (2007). This generalization is called *subpath consistency*, since the subpaths of different routing patterns has to be consistent. Subpath consistency will be defined more formally below, but let us first describe how routing patterns are defined in this paper.

We introduce one routing pattern for each node in the network, and the routing pattern associated with node l is denoted A_l. A_l is supposed to describe all shortest paths to node l from any other node in the network, and A_l is represented by the included links. The traffic is never routed in cycles in OSPF networks (since the weights are positive), so each A_l is an acyclic set of links. A_l is in this paper called the *in-graph to node l*, since it contains directed paths to node l from any other node in the node set. Note that A_l is an in-tree for the special case when all shortest path to node l are unique.

The in-graphs are supposed to be shortest path graphs obtained from the same set of unknown weights, and such weights exist only if the different in-graphs contain similar paths. More specifically, if one in-graph contains a set of directed paths between two nodes, then these paths are supposed to be shortest, which means that the same directed paths should be included in any other in-graph where the two nodes are connected. If we let $S_l(s,t)$ denote all subpaths from node s to node t in A_l, subpath consistency can be defined as follows.

Definition 2 *A_k and A_l are subpath consistent if $S_k(s,t) = S_l(s,t)$ for all $s \in N$ and $t \in N$ such that $S_k(s,t) \neq \emptyset$ and $S_l(s,t) \neq \emptyset$.*

It is shown in Broström and Holmberg (2007) that a necessary condition for the existence of compatible weights is that *all* pairs of routing patterns are subpath consistent.

3 Mathematical model

We will now present a mixed-integer model of the OSPF network design problem. The model is based on routing patterns formed as in-graphs, and we ensure that each pair of in-graphs satisfies the necessary condition based on subpath consistency. The model will from this point be called the Subpath Consistent IP Network Design model (SCIPND).

The SCIPND model uses the following groups of variables. The binary variable q_{ij} is 1 if link $(i,j) \in E$ is installed, and 0 otherwise. The integer variable z_{ij} denotes the number of capacity steps installed on link $(i,j) \in E$. Recall that p_{ij} is the fixed charge for $(i,j) \in E$, and that f_{ij} is the cost for a capacity step of size u_{ij}. The flow variable x^k_{ij} denotes the fraction of commodity $k \in C$ which is routed on the directed link $(i,j) \in A$. The flow variables are continuous since the demand volume h^k can be routed on several paths towards the destination. The total flow of commodity k on link (i,j) is $h^k x^k_{ij}$, and the operating cost is c_{ij} per unit. The in-graphs are modeled using binary variables, and y^l_{ij} is 1 if $(i,j) \in A$ is included in the in-graph with destination $l \in N$. Each in-graph $A_l = \{(i,j) \in A : y^l_{ij} = 1\}$ is, as mentioned earlier, a desired shortest path graph to node l. Finally, the binary variable τ^l_{in} is 1 if node n is reachable from node i in A_l, and 0 otherwise. This group of variables will be used for ensuring that the in-graphs do not contain any directed cycles, and for ensuring that each pair of in-graphs is subpath consistent.

The objective function minimizes the cost of the network. The cost function consists of a sum of fixed charges, a sum of capacity costs and a sum of operating costs. The objective function is given below.

$$\min \quad v = \sum_{(i,j)\in E} p_{ij} q_{ij} + \sum_{(i,j)\in E} f_{ij} z_{ij} + \sum_{(i,j)\in A} \sum_{k\in C} c_{ij} h^k x^k_{ij} \qquad (1)$$

The multicommodity network design part of the model ensures that links and link capacities are installed, and that the demand volume of each commodity $k \in C$ is routed from node o_k to node d_k without violating the installed link capacities.

$$\sum_{j:(i,j)\in A} x_{ij}^k - \sum_{j:(j,i)\in A} x_{ji}^k = \begin{cases} 1 & i = o_k \\ -1 & i = d_k \\ 0 & i \neq o_k,\ d_k \end{cases} \quad \forall\ i \in N, \forall\ k \in C \tag{2}$$

$$\sum_{k\in C} h^k(x_{ij}^k + x_{ji}^k) \leq u_{ij} z_{ij} \qquad \forall\ (i,j) \in E \tag{3}$$

$$x_{ij}^k + x_{ji}^k \leq q_{ij} \qquad \forall\ (i,j) \in E, \forall\ k \in C \tag{4}$$

Constraints 2 are standard flow conservation constraints, and ensure that the entire demand volume of each commodity is routed from the origin to the destination. Constraints 3 ensure that the total traffic on each bidirected link does not exceed the installed link capacity, and constraints 4 ensure that traffic is only routed on installed links. We will later ensure that a commodity can use only one direction of a bidirected link, so we have $x_{ij}^k = 0$ and/or $x_{ji}^k = 0$. This yields $x_{ij}^k + x_{ji}^k \leq 1$, which is used in 4.

The next part of the model ensures that the y variables describe a set of in-graphs, $A_l = \{(i,j) \in A : y_{ij}^l = 1\}$, $\forall l \in N$. This part of the model also ensures that τ_{in}^l is set to 1 when A_l contains a directed path from i to n.

$$\sum_{j:(i,j)\in A} y_{ij}^l \geq 1 \qquad \forall\ i \in N : i \neq l, \forall\ l \in N \tag{5}$$

$$\tau_{ij}^l \geq y_{ij}^l \qquad \forall\ (i,j) \in A, \forall\ l \in N \tag{6}$$

$$y_{ij}^l + \tau_{jn}^l - \tau_{in}^l \leq 1 \qquad \forall\ (i,j) \in A, \forall\ n \in N, \forall\ l \in N \tag{7}$$

$$\tau_{ii}^l = 1 \qquad \forall\ i \in N, \forall\ l \in N \tag{8}$$

$$\tau_{in}^l + \tau_{ni}^l \leq 1 \qquad \forall\ i \in N : i \neq n, \forall\ n \in N, \forall\ l \in N \tag{9}$$

Constraints 5 ensure that all nodes, except for the root node, has out-degree greater or equal to one. It thus follows that each node is connected to some other node, and this is obviously necessary if the node should be connected to node l.

Constraints 6 ensure that $\tau_{ij}^l = 1$ when j is reachable from i by a path of length one, i.e. when $y_{ij}^l = 1$. Constraints 7 does the same thing for paths that consist of several links. This set of constraints can also be written as $\tau_{in}^l \geq \max_j(y_{ij}^l + \tau_{jn}^l - 1)$, so we get $\tau_{in}^l = 1$ if link (i,j) is included in A_l, i.e. $y_{ij}^l = 1$, and if node n is reachable from node j in A_l, i.e. $\tau_{jn}^l = 1$.

Constraints 8 state that each node is reachable from itself. Due to 8, constraint set 7 can also handle paths of length one. In this case we have $n = j$, so $y_{ij}^l = 1$ and $\tau_{jj}^l = 1$ yields $\tau_{ij}^l = 1$. Constraints 6 are therefore redundant and can be omitted, but we keep these constraints for explanatory reasons.

Constraints 6 and 7 ensure that $\tau_{in}^l = 1$ and $\tau_{ni}^l = 1$ if the y variables are set to one in a directed cycle including nodes i and n. It thus follows from constraints 9 that A_l may not contain any directed cycles.

Let us now show that if constraints 5-9 are satisfied, $A_l = \{(i,j) : y_{ij}^l = 1\}$ is an in-graph to node l. We have already verified that A_l may not contain directed cycles, due to 6-9, so it only remains to verify that all nodes are connected to node l. This is shown by assuming the opposite and getting a contradiction.

Let S is the set of nodes reachable from node n in A_l, and let us assume that A_l does not contain a directed path from node n to node l. We then have $l \not\subseteq S$. Summing up 5 for all $i \in S$ yields $\sum_{i \in S} \sum_{j:(i,j) \in A} y_{ij}^l \geq |S|$. We must have $y_{ij}^l = 0$ for all $(i,j) \in A : i \in S, j \notin S$, since node j would otherwise be reachable from node n via node i. It thus follows that at least $|S|$ links starts and ends in S, so S contains a directed cycle. This contradicts constraints 6-9, so we conclude that node l is reachable from node n. Since n was arbitrary chosen, node l is reachable from all $n \in N$, so A_l is an in-graph to node l.

So far, we have ensured that τ_{in}^l is 1 when A_l contains a path from node i to node n. However, it is not certain that τ_{in}^l is set to 0 when n is not reachable from i in A_l, and this is ensured by the next part of the model.

$$2\,\delta_{ijn}^l \leq y_{ij}^l + \tau_{jn}^l \qquad \forall\ (i,j) \in A, \forall\ n \in N, \forall\ l \in N \tag{10}$$

$$\sum_{j:(i,j) \in A} \delta_{ijn}^l \geq \tau_{in}^l \qquad \forall\ i \in N, \forall\ n \in N, \forall\ l \in N \tag{11}$$

Due to constraints 10, the auxiliary binary variable δ_{ijn}^l is 0 when node n is not reachable from node i in A_l via link (i,j) (we may have $n = j$). Constraints 11 use δ to ensure that τ_{in}^l is 0 when node n is not reachable from node i in A_l via *any* link (i,j). These constraints can also be written as $2\tau_{in}^l \leq \max_j(y_{ij}^l + \tau_{jn}^l)$, so we can only get $\tau_{in}^l = 1$ if some link (i,j) is included in A_l and if node n is reachable form node j in A_l, i.e. if $y_{ij}^l = 1$ and $\tau_{jn}^l = 1$ for some $(i,j) \in A$.

A pair of in-graphs is subpath inconsistent if the the two in-graphs contain different paths from one node to another. In order to ensure that a pair of in-graphs is subpath consistent, we must therefore ensure that if both in-graphs contain paths from node i to node j, then the set of all paths from i to j must be identical for the two in-graphs. This is ensured by constraints 12, as shown below.

$$\tau_{ij}^l + \tau_{ij}^{l'} + \tau_{in}^l + \tau_{mj}^l \leq 4 - y_{nm}^l + y_{nm}^{l'} \ \ \forall\ (n,m) \in A, \forall\ i \in N, \forall\ j \in N, \forall\ l' \in N, \forall\ l \in N \tag{12}$$

Suppose that A_1 and A_2 contain paths from node i to node j. We then have $\tau_{ij}^1 = 1$ and $\tau_{ij}^2 = 1$. Now consider a path p from i to j. If p is included in A_1, we have $\tau_{in}^1 = 1$, $y_{nm}^1 = 1$ and $\tau_{mj}^1 = 1$ for all $(n,m) \in p$. The constraints in set 12, defined for $l = 1$ and $l' = 2$, then state $1 \leq y_{nm}^2\ \forall (n,m) \in p$, so the path p is included in A_2. This holds for any path p, so each path from i to j in A_1 is also included in A_2. On the other hand, if the path p is included in A_2, we have $\tau_{in}^2 = 1$, $y_{nm}^2 = 1$ and $\tau_{mj}^2 = 1$ for all $(n,m) \in p$. The constraints in set 12, defined for $l = 2$ and $l' = 1$, then state $1 \leq y_{nm}^1\ \forall (n,m) \in p$, so the path p is included in A_1. This holds for any path p, so each path from i to j in A_2 is also included in A_1.

We have thus shown that under the assumption that A_1 and A_2 contains paths from node i to node j, constraints 12 ensure that the same paths between the two nodes are included in A_1 and A_2. This holds for all $i, j \in N$, so A_1 and A_2 are subpath consistent. This also holds for all $l, l' \in N$, so constraints 12 ensure that all pairs of in-graphs are subpath consistent.

A disadvantage with 12 is the large number of constraints. The following constraints are a relaxation of 12.

$$\tau^l_{il'} + \tau^l_{jl'} \leq 2 + y^{l'}_{ij} - y^l_{ij} \qquad \forall (i,j) \in A, \forall l' \in N, \forall l \in N \tag{13}$$
$$\tau^l_{il'} + \tau^l_{jl'} \leq 2 - y^{l'}_{ij} + y^l_{ij} \qquad \forall (i,j) \in A, \forall l' \in N, \forall l \in N \tag{14}$$

Constraints 13 are obtained if we let $j = l'$ and $n = i$ in 12, and constraints 14 are obtained if we replace l and l' in 12, let $j = l'$ and $n = i$ and add the valid inequality $\tau^l_{jl'} \leq 1$.

Constraints 13 and 14 ensure that each path from node i to node l' in A_l also is included in $A_{l'}$. These constraints are defined for all $l \in N$, so A'_l will contain all paths from i to l' which are included in any other in-graph. (However, 13 and 14 do not ensure that all paths from i to l' in $A_{l'}$ are included in any other in-graph that contains a path from i to l'.)

The last part of the model ensures that each commodity is routed on links that belongs to the in-graph rooted at the destination, i.e. on desired shortest paths. The traffic should also be routed in accordance with the ECMP principle, so the traffic leaving a node should be evenly balanced on all outgoing links that belong to the in-graph rooted at the destination.

$$x^k_{ij} \leq y^{d_k}_{ij} \qquad \forall (i,j) \in A, \forall k \in C \tag{15}$$
$$y^{d_k}_{ij'} + y^{d_k}_{ij} \leq 2 - x^k_{ij'} + x^k_{ij} \qquad \forall j' : (i,j') \in A, \forall j : (i,j) \in A, \forall i \in N, \forall k \in C \tag{16}$$

Constraints 15 ensure that $x^k_{ij} = 0$ if $y^{d_k}_{ij} = 0$, so commodity k can not use links that are not included in A_{d_k}. An in-graph may not contain directed cycles (due to 6-9), so these constraints also ensure that each commodity uses at most one direction of each bidirected link. This is why the left-hand-sides in constraints 4 can not be greater than 1.

Let us now consider the ECMP constraints, set 16. First we note that the right-hand-sides can not be less than 1, since we have $0 \leq x^k_{ij} \leq 1$ for all $i \in N$ and $k \in C$. Furthermore, the left-hand-sides can only take values 0, 1, or 2, so a constraint is clearly redundant unless $y^{d_k}_{ij} = 1$ and $y^{d_k}_{ij'} = 1$. Set 16 contains two constraints for each pair of links, so the remaining case yields $2 \leq 2 - x^k_{ij'} + x^k_{ij}$ and $2 \leq 2 - x^k_{ij} + x^k_{ij'}$. These inequalities can only be satisfied if $x^k_{ij} = x^k_{ij'}$, so constraint set 16 ensures that the traffic of commodity k is evenly balanced on pairs of outgoing links that belong to a shortest path to the destination.

The ECMP constraints has previously been presented in Broström and Holmberg (2005) and Broström and Holmberg (2006). Compared to the first linear formulation of ECMP, see Holmberg and Yuan (2004), we do not introduce any additional variables, and this implies that the size of the model is significantly reduced. De Giovanni et al. (2005) has independently developed ECMP constraints for aggregated flow variables, and their constraints are very similar to ours. Compared to our formulation, one y variable has been eliminated from the left-hand-side, and this implies that the constant in the right-hand-side of 16 can be reduced to 1. The right-hand-side is also divided with a constant, which is equal to the total demand volume at the destination.

The SCIPND model consists of minimizing 1 with respect to 2-16 and constraints defining the variable domains, i.e. $0 \leq x \leq 1;\ q,\ y,\ \tau,\ \delta \in \{0,1\}$ and $z \geq 0,$ integer.

4 The Lagrangean heuristic

Preliminary experiments show that the SCIPND model is too hard to solve to optimality using a MIP solver, even for small and medium sized instances. If the SCIPND model is studied in detail, one finds that the variables x, z and q are only connected to the variables y and τ by the ECMP constraints. Furthermore, the subpath consistency constraints are the only ones connecting variables y and τ with different indices l. We will therefore propose a solution approach based on Lagrangean relaxation, where constraints 12, 13, 14, 15 and 16 are relaxed with Lagrangean multipliers $r \geq 0$, $s \geq 0$, $t \geq 0$, $u \geq 0$ and $v \geq 0$, respectively. This gives the following Lagrangean function.

$$\begin{aligned} L(r,s,t,u,v) = & \sum_{(i,j)\in E} p_{ij}q_{ij} + \sum_{(i,j)\in E} f_{ij}z_{ij} + \sum_{(i,j)\in A}\sum_{k\in C} c^x_{ijk}x^k_{ij} \\ & + \sum_{(i,j)\in A}\sum_{l\in N} c^y_{ijl}y^l_{ij} + \sum_{i\in N}\sum_{n\in N}\sum_{l\in N} c^\tau_{inl}\tau^l_{in} - \Gamma, \end{aligned}$$

where

$$c^x_{ijk} = c_{ij}h^k + u^k_{ij} + \sum_{m:(i,m)\in A}\left(v^k_{ijm} - v^k_{imj}\right),$$

$$\begin{aligned} c^y_{ijl} = & \sum_{l'\in N}\left(s^{l'l}_{ij} - s^{ll'}_{ij} - t^{l'l}_{ij} + t^{ll'}_{ij} + \sum_{n\in N}\sum_{m\in N}\left(r^{ll'}_{nmij} - r^{l'l}_{nmij}\right)\right) - \sum_{k\in C:d_k=l} u^k_{ij} \\ & + \sum_{k\in C:d_k=l}\sum_{m:(i,m)\in A}\left(v^k_{ijm} + v^k_{imj}\right), \end{aligned}$$

$$\begin{aligned} c^\tau_{inl} = & \sum_{l'\in N}\left(\sum_{j\in N}\left(\sum_{m:(n,m)\in A} r^{ll'}_{ijnm} + \sum_{m:(m,i)\in A} r^{ll'}_{jnmi}\right) + \sum_{(j,m)\in A}\left(r^{ll'}_{injm} + r^{l'l}_{injm}\right)\right) \\ & + \sum_{m:(i,m)\in A}\left(s^{nl}_{im} + t^{nl}_{im}\right) + \sum_{m:(m,i)\in A}\left(s^{nl}_{mi} + t^{nl}_{mi}\right), \end{aligned}$$

$$\begin{aligned} \Gamma = & \sum_{(i,j)\in A}\sum_{l'\in N}\sum_{l\in N}\left(2s^{l'l}_{ij} + 2t^{l'l}_{ij} + \sum_{n\in N}\sum_{m\in N} 4r^{ll'}_{nmij}\right) \\ & + \sum_{k\in C}\sum_{i\in N}\sum_{m:(i,m)\in A}\sum_{j:(i,j)\in A} 2v^k_{imj}. \end{aligned}$$

Note that c^τ_{inl} is non-negative, while c^x_{ijk} and c^y_{ijl} may become negative. Also note that Γ is a constant.

4.1 The Lagrangean subproblems

The proposed relaxation results in a Lagrangean subproblem that is separable into $|N| + 1$ subproblems. The first subproblem is a capacitated network design problem, which is well-known to be NP-hard. This problem is denoted DS_0, and could by itself be solved using solution techniques based on Lagrangean relaxation, for example by applying Lagrangean relaxation of the node-balance constraints, as in Holmberg and Yuan (1998), or of the capacity constraints, as in Crainic, Frangioni, and Gendron (2001).

$$
\begin{array}{lll}
[DS_0] & \min\limits_{x,z,q} \sum\limits_{(i,j)\in E} p_{ij}q_{ij} + \sum\limits_{(i,j)\in E} f_{ij}z_{ij} + \sum\limits_{(i,j)\in A}\sum\limits_{k\in C} c^x_{ijk}x^k_{ij} & \\
& \text{s.t.} \sum\limits_{j:(i,j)\in A} x^k_{ij} - \sum\limits_{j:(j,i)\in A} x^k_{ji} = \begin{cases} 1 & i = o_k \\ -1 & i = d_k \\ 0 & i \neq o_k,\ d_k \end{cases} & \forall\ k \in C \\
& \sum\limits_{k\in C} h^k(x^k_{ij} + x^k_{ji}) \ \leq\ u_{ij}z_{ij} & \forall\ (i,j) \in E \\
& x^k_{ij} + x^k_{ji} \ \leq\ q_{ij} & \forall\ (i,j) \in E, \forall\ k \in C \\
& 0 \leq x^k_{ij} \leq 1 & \forall\ (i,j) \in A, \forall\ k \in C \\
& q_{ij} \in \{0,1\} & \forall\ (i,j) \in E \\
& z_{ij} \geq 0,\ \text{integer} & \forall\ (i,j) \in E
\end{array}
$$

The remaining subproblems are minimization problems over the y, τ and δ variables under constraints 5-11 and $y, \tau, \delta \in \{0,1\}$. The feasible sets describe in-graphs, and we get one subproblem for each in-graph since the constraints are separable in $l \in N$. The objective functions state that costs for links and costs for subpaths for each in-graph should be minimized. The subproblems are denoted $DS_l\ \forall l \in N$, so subproblem DS_l consists of finding a min-sum in-graph rooted at node l.

$$
\begin{array}{lll}
[DS_l] & \min\limits_{y,\tau} \sum\limits_{(i,j)\in A} c^y_{ijl}y^l_{ij} + \sum\limits_{i\in N}\sum\limits_{n\in N} c^\tau_{inl}\tau^l_{in} & \\
& \text{s.t.} \sum\limits_{j:(i,j)\in A} y^l_{ij} \ \geq\ 1 & \forall\ i \in N : i \neq l \\
& \tau^l_{ij} \ \geq\ y^l_{ij} & \forall\ (i,j) \in A \\
& y^l_{ij} + \tau^l_{jn} - \tau^l_{in} \ \leq\ 1 & \forall\ (i,j) \in A, \forall\ n \in N \\
& \tau^l_{ii} \ =\ 1 & \forall\ i \in N, \forall\ l \in N \\
& \tau^l_{in} + \tau^l_{ni} \ \leq\ 1 & \forall\ i \in N : i \neq n, \forall\ n \in N \\
& y^l_{ij} \in \{0,1\} & \forall (i,j) \in A \\
& \tau^l_{in} \geq 0 & \forall\ i \in N,\ \forall\ n \in N
\end{array}
$$

DS_l has been simplified by relaxing constraints 10-11 and the integrality requirements on the τ variables. The δ variables can then be discarded since they only appear in constraints 10-11. As shown below, the optimal objective function value of the min-sum in-graph problem is not affected by these relaxations.

Constraints 10-11 ensure that τ^l_{in} is set to zero when the optimal in-graph contains no path from node i to node n, so if these constraints are relaxed, τ^l_{in} could be set equal to one even if the optimal in-graph does not contain a path from node i to node n. (This relaxation does not affect the y variables since the feasible set of DS_l ensures that the y variables form an in-graph to node l.) Now we note that all τ variables have non-negative cost coefficients, so the optimal objective function value is either increased or unchanged if the values of some τ variables are increased. We thus conclude that the optimal objective function value can not be improved if constraints 10-11 (and variables δ) are discarded.

Subproblem DS_l has also been simplified by relaxing the integrality requirements on the τ variables. For the same reasons as above, the optimal objective function value can not be improved by increasing the values of τ variables, and this also holds for rational values. Now it only remains to verify that the objective function value can not be improved by decreasing the values of τ variables that should be set equal to one.

Let us consider a path p from i_1 to i_n in the optimal in-graph, i.e. we have $y^l_{ij} = 1$ for all $(i, j) \in p$. Summing up constraints 7 for all links in p and for $j = i_n$ yields

$$\sum_{(i,j)\in p} y^l_{ij} + \tau^l_{i_n i_n} - \tau^l_{i_1 i_n} \leq |p|,$$

since all other τ variables cancel out. We have $\sum_{(i,j)\in p} y^l_{ij} = |p|$ and $\tau^l_{i_n i_n} = 1$, so we get $\tau^l_{i_1 i_n} \geq 1$. This holds for any path in the in-graph, so we conclude that τ^l_{in} will be set to one if the in-graph contains a path from node i to node n (even if the integer requirements on τ are relaxed). The optimal objective function value can therefore not be improved by decreasing the values of τ variables that should be set equal to one.

The min-sum in-graph problem is closely related to the min-sum arborescence problem, see e.g. Gondran and Minoux (1984), but there are also some main differences between these optimization problems. One difference is that a feasible solution to the min-sum arborescence problem contains at most one directed path between a pair of nodes, while a feasible solution to the min-sum in-graph problem may contain more than one directed path between a pair of nodes. Another difference is that the sum of link costs is minimized in the min-sum arborescence problem, while the sums of link costs and costs for subpaths are minimized in the min-sum in-graph problem.

Chu and Liu (1965) and Edmonds (1967) has independently developed polynomial methods for the min-sum arborescence problem, and an $O(|N^2|)$ algorithm based on the Edmonds' algorithm has been presented by Fischetti and Toth (1993). The differences between the min-sum arborescence problem and the min-sum in-graph problem unfortunately prevent us from using these solution methods.

The integrality property does not hold for DS_l, so an optimal solution can not be found by solving the LP-relaxation. A feasible solution can however be generated by including at least one leaving link at each node (except at node l) and by ensuring that the included links do not form a directed cycle, i.e. by generating an in-graph to node l. If the cost coefficient of all τ variables are zero, it is easy to determine the cost for including a link into the in-graph, since the only cost associated with this decision is the link cost. It is however much more difficult to determine the actual cost for the decision when τ variables are associated with positive costs. In this case, the total cost depends of the link cost, the costs for subpaths that include the link, and the costs for subpaths that include the link and are formed by decisions taken later on. We have not been able to avoid this difficulty, and DS_l is therefore solved using a MIP solver. We believe that DS_l is NP-hard, but have not been able to prove it.

4.2 Subgradient optimization

A lower bound of the optimal objective value for the OSPF network design problem can be obtained by solving the Lagrangean subproblems for fixed Lagrangean multipliers. The best lower bound can be obtained by solving the Lagrangean dual problem (LD). If we let μ denote the Lagrangean multipliers (i.e. let $\mu = (r, s, t, u, v)$), and if we let $g^l(\mu)$ denote the optimal objective function value of subproblem DS_l for fixed multipliers μ, the Lagrangean dual problem can be stated as

$$[LD] \qquad v_L = \max_{\mu \geq 0} \; g(\mu)$$

where

$$g(\mu) = \sum_{l=0}^{|N|} g^l(\mu) - \Gamma.$$

The Lagrangean multipliers will here be updated using subgradient optimization, see e.g. Poljak (1967), Held, Wolfe, and Crowder (1974) and Shor (1985), and we let $\mu^{(\gamma)}$ denote the Lagrangean multipliers in iteration γ. The dual function $g(\mu)$ is a piecewise linear function and its function value is found by solving the Lagrangean subproblems. The optimal solution of the Lagrangean subproblems, $(\bar{x}, \bar{z}, \bar{q}, \bar{y}, \bar{\tau})$, provides a subgradient, $\xi^{(\gamma)} = (\xi^r, \xi^s, \xi^t, \xi^u, \xi^v)$, which is computed as

$$\begin{aligned}
\xi^r_{ijl'l} &= \bar{\tau}^l_{ij} + \bar{\tau}^{l'}_{ij} + \bar{\tau}^l_{in} + \bar{\tau}^l_{mj} - 4 + \bar{y}^l_{nm} - \bar{y}^{l'}_{nm} && \forall (n,m) \in A, \forall i \in N, \forall j \in N, \\
& && \forall l' \in N, \forall l \in N, \\
\xi^s_{ijl'l} &= \bar{\tau}^l_{il'} + \bar{\tau}^l_{jl'} - 2 - \bar{y}^{l'}_{ij} + \bar{y}^l_{ij} && \forall (i,j) \in A, \forall l' \in N, \forall l \in N, \\
\xi^t_{ijl'l} &= \bar{\tau}^l_{il'} + \bar{\tau}^l_{jl'} - 2 + \bar{y}^{l'}_{ij} - \bar{y}^l_{ij} && \forall (i,j) \in A, \forall l' \in N, \forall l \in N, \\
\xi^u_{ijk} &= \bar{x}^k_{ij} - \bar{y}^{d_k}_{ij} && \forall (i,j) \in A, \forall k \in C, \\
\xi^v_{ij'jk} &= \bar{y}^{d_k}_{ij'} + \bar{y}^{d_k}_{ij} - 2 + \bar{x}^k_{ij'} - \bar{x}^k_{ij} && \forall j' : (i,j') \in A, \forall j : (i,j) \in A, \\
& && \forall i \in N, \forall k \in C.
\end{aligned}$$

Note that $\xi^{(\gamma)}$ is an $\epsilon-$subgradient if the subproblems are approximately solved. The Lagrangean multipliers are updated by taking a step of length $\alpha^{(\gamma)}$ in the direction given by $\xi^{(\gamma)}$ from the current iterate, i.e. $\mu^{(\gamma+1)} = (\mu^{(\gamma)} + \alpha^{(\gamma)}\xi^{(\gamma)})_+$. We use the so called Poljak step as step length, which can be computed as

$$\alpha^{(\gamma)} = \lambda^{(\gamma)} \frac{\bar{v} - g(\mu^{(\gamma)})}{||\xi^{(\gamma)}||^2},$$

where $0 < \lambda^{(\gamma)} < 2$ and $\bar{v}$ is a target value and should be equal to the optimal objective function value.

4.3 Generating feasible solutions

The solution to the Lagrangean subproblem, $(\bar{x}, \bar{z}, \bar{q}, \bar{y}, \bar{\tau})$, can also be used as input to a heuristic for generating upper bounds of the optimal objective function value for the OSPF network design problem. The in-graphs, $A_l = \{(i,j) \in A : \bar{y}^l_{ij} = 1\}\ \forall l \in N$, are supposed to be shortest path graphs obtained from certain link weights, but it is not certain that such weights exist. If compatible weights exists, we want to find them, and if no such weights exist, we want to find a set of weights that yields shortest path graphs that are similar to the in-graphs. Such weights can be found by solving the Artificial Weight Finding Problem (AWFP), which is based on a model from Broström and Holmberg (2008).

$$
\begin{array}{llll}
[AWFP] & \min \sum\limits_{(i,j)\in A} \sum\limits_{l \in N} a^l_{ij} & & \\
& \text{s.t. } w_{ij} + \pi^l_i - \pi^l_j = M a^l_{ij} & \forall (i,j) \in A_l, l \in N & (17) \\
& w_{ij} + \pi^l_i - \pi^l_j \geq 1 - a^l_{ij} & \forall (i,j) \notin A_l, l \in N & (18) \\
& a^l_{ij} \leq 1 & \forall (i,j) \notin A_l, l \in N & (19) \\
& w_{ij} \geq 1, \text{integer} & \forall (i,j) \in A & (20) \\
& a^l_{ij} \geq 0 & \forall (i,j) \in A, \forall l \in N & (21)
\end{array}
$$

The input to AWFP is the in-graphs $A_l\ \forall l \in N$, which define the feasible set. AWFP uses the following variables; w_{ij} is the weight of link (i,j), π^l_i is the node potential for node i and in-graph A_l, and a^l_{ij} is an artificial variable associated with link (i,j) and in-graph A_l. The model in Broström and Holmberg (2008) is obtained by letting $a^l_{ij} = 0\ \forall (i,j) \in A,\ \forall l \in N$.

It is shown in Frank (1995) that for fixed positive integer weights $\bar{w}$, there exists node potentials π satisfying $\bar{w}_{ij} + \pi_i - \pi_j \geq 0\ \forall (i,j) \in A$. It is also shown that the potentials can be chosen such that $\bar{w}_{ij} + \pi^l_i - \pi^l_j = 0$ is satisfied for links (i,j) that belong to a shortest path to node l, and such that $\bar{w}_{ij} + \pi^l_i - \pi^l_j \geq 1$ is satisfied for links (i,j) that do not belong to a shortest path to node l.

If there exists weights w (and node potentials π) such that $w_{ij} + \pi^l_i - \pi^l_j = 0\ \forall (i,j) \in A_l, \forall l \in N$, and $w_{ij} + \pi^l_i - \pi^l_j \geq 1\ \forall (i,j) \notin A_l, \forall l \in N$, then these weights give shortest path graphs identical to the in-graphs. All artificial variables can in this case be set equal to zero, and this yields an optimal solution to AWFP since its objective function value can not be negative. The artificial variables ensure that AWFP has a feasible solution even if there are no compatible weights, i.e. when it is impossible to satisfy $w_{ij} + \pi^l_i - \pi^l_j = 0\ \forall (i,j) \in A_l$ and $w_{ij} + \pi^l_i - \pi^l_j \geq 1\ \forall (i,j) \notin A_l$. The artificial variables give a measure of the deviations from the desired in-graphs, and since the sum of artificial variables is minimized in the objective function, an optimal solution to AWFP yields a set of weights that violates the desired constraints as little as possible.

We will now present how feasible solutions to the OSPF network design problem are generated using the in-graphs. The heuristic first generates a set of weights by solving

AWFP, and a shortest path graph to each node in the network is then computed using the weights. When all shortest paths to each destination are known, it is easy to compute OSPF flows with respect to the weights, simply by balancing the flow leaving a node evenly on all leaving links that belongs to a shortest path to the destination. A feasible solution to the OSPF network design problem is then obtained by installing links and link capacities such that no link is overloaded. The cost for the solution is computed by summing up fixed charges, capacity costs and operating costs i.e. by evaluating the objective function value for the SCIPND model.

The next part of the heuristic tries to find better feasible solutions by applying a tabu search method, which is based on the weights. Let w^0 be the set of weights found by solving AWFP, let w^i be the set of weights in iteration i, and let $N(w^i)$ be the neighborhood of w^i. $N(w^i)$ contains sets of weights obtained by changing one weight in w^i. We consider the same changes as in Holmberg and Yuan (2004) and Broström and Holmberg (2006), so a weight is either (a) decreased such that *some* demand is routed differently, (b) increased such that *some* demand is routed differently, or (c) increased such that *no* demand uses the link. Each weight is thus changed in three different ways, and the weight is not changed more than necessary. The OSPF protocol uses positive weights, so a neighbor solution is discarded if the changed weight becomes non-positive. For implementational reasons, a neighbor solution is also discarded if the value of the weight exceeds 100. The weights in the neighborhood are evaluated in the same manner as w^0, and the set of weights that yields the lowest objective function value is chosen as the next iterate. The same weight may not be changed for the next T iterations.

Let us now describe the heuristic in a more algorithmic fashion. Here, $\bar{v}$ is the upper bound to the optimal objective function value of the OSPF network design problem, i is the iteration counter, T is the length of the tabu list, and K is used as stopping criterion.

The OSPF heuristic

1. Initialize the vector of link weights, w^0, by solving AWFP. Let $i = 0$, $\bar{v} = \infty$ and initialize K and T to appropriate values.

2. Compute/update the shortest path trees with respect to w^i (see below).
   Compute OSPF flows x (see above).
   Install link capacities, i.e. let $z_{ij} = \left\lceil \sum_{k \in C} h^k (x_{ij}^k + x_{ji}^k) \right\rceil \ \forall (i,j) \in E$.
   Install links, i.e. for each $(i,j) \in E$, let $q_{ij} = 1$ if $z_{ij} > 0$ and let $q_{ij} = 0$ if $z_{ij} = 0$.
   Compute the design cost $v(w^i) = \sum_{(i,j) \in E} p_{ij} q_{ij} + \sum_{(i,j) \in E} f_{ij} z_{ij} + \sum_{(i,j) \in A} \sum_{k \in C} c_{ij} h^k x_{ij}^k$.

3. If $v(w^i) < \bar{v}$, store the solution and let $\bar{v} = v(w^i)$. If $\bar{v}$ has not been improved for K iterations, go to 5.

4. For each $w \in N(w^i)$; update the shortest path trees, compute OSPF flows, install link capacities, install links, and compute the design cost $v(w)$.
   Let $w^{i+1} = \arg\min w \in N(w^i) v(w)$, let $i = i + 1$, update the tabu list, and go to 2.

5. Terminate the procedure with approximate objective function value $\bar{v}$.

Note that we may return to step 3 from step 4 if the best solution in the neighborhood (and its corresponding shortest path tree) is stored.

Dijkstra's method is used for computing shortest paths the first time step 2 is entered, and the shortest paths to each destination are stored in *shortest path trees.* A shortest path tree contains one shortest path from each node to the destination, and a vector of node potentials allows us to find all shortest paths. A thread index enables a preorder traversal of the tree, and it is possible to compute how much a certain weight has to be changed before some shortest path is affected by traversing a part of a tree and by checking all entering/leaving links. Since changing a single weight in general have minor impact on the shortest paths, substantial computational savings can be obtained if the shortest path are updated (and not recomputed) in step 4 and when step 2 is re-entered. This data structure is further described in Ahuja, Magnanti, and Orlin (1993) (Section 11.3), and has previously been used for OSPF networks in Broström and Holmberg (2006).

4.4 Algorithm summary

Let us now summarize the Lagrangean heuristic for the OSPF network design problem. Recall that $\mu^{(\gamma)}$ denotes the Lagrangean multipliers in iteration γ, and that $g(\mu)$ is the dual function. We let $\bar{v}$ and $\underline{v}$ denote the best known upper and lower bounds of the optimal objective value, and we let T be the available solution time. Note that the step length parameter λ is increased when either $\bar{v}$ or $\underline{v}$ is improved, and that the parameter is decreased when neither bounds have been improved for 5 iterations.

The Lagrangean heuristic

1. Let $\gamma = 0$, $\lambda^{(0)} = 1.0$, $\underline{v} = 0$ and $\bar{v} = \infty$. Initialize the Lagrangean mutipliers $\mu^{(0)}$ to zeros.

2. Evaluate $g(\mu^{(\gamma)})$ by solving the Lagrangean subproblems. If $g(\mu^{(\gamma)}) > \underline{v}$, let $\underline{v} = g(\mu^{(\gamma)})$.

3. Generate feasible solutions using the OSPF heuristic. Let $h(\mu^{(\gamma)})$ be the objective function value for the best found solution. If $h(\mu^{(\gamma)}) < \bar{v}$, let $\bar{v} = h(\mu^{(\gamma)})$.

4. Compute the subgradient $\xi^{(\gamma)}$, the steplength $\alpha^{(\gamma)}$ and let $\mu^{(\gamma+1)} = (\mu^{(\gamma)} + \alpha^{(\gamma)}\xi^{(\gamma)})_+$.
   If $\underline{v}$ was updated in 2 or if $\bar{v}$ was updated in 3, let $\lambda^{(\gamma+1)} = 1.2 \cdot \lambda^{(\gamma)}$.
   If $\underline{v}$ and $\bar{v}$ have not be updated for five iterations, let $\lambda^{(\gamma+1)} = 0.95 \cdot \lambda^{(\gamma)}$.

5. Terminate if the solution time exceeds T. Otherwise let $\gamma = \gamma + 1$ and go to 2.

5 Numerical experiments

Numerical results have been obtained from experiments on randomly generated test problems and on a few real life networks from the *survivable network design data library* SNDlib 1.0 (2005). The random networks are generated as follows. Coordinates for the

Table 1: Groups of test instances

	fixed charges	step costs	demands
T1	large	large	large
T2	large	large	small
T3	large	small	large
T4	small	large	large
T5	small	large	small

Table 2: Settings

	DS_0		DS_l
	q	z	B&B
S1	bin	cont	∞
S2	bin	cont	400
S3	cont	cont	∞
S4	cont	cont	400

nodes are randomly chosen, and each node is connected to its three (or four) closest neighbors by a directed link. If a node pair only has a link in one direction between the two nodes, a directed link is introduced in the other direction as well. This makes the network bidirected, and each node is connected to at least three (or four) other nodes. If the network is not connected, or if the removal of a single bidirected link makes the network disconnected, the network is discarded and new coordinates are randomly chosen.

The link costs are based on the Euclidean distance between the nodes, with a random variation added. The cost structure of the links, and the structure of demands, may affect the performance on the solution method. We have therefore generated instances with different types of data, and the different groups are displayed in Table 1.

The properties of the random instances has been investigated by solving a capacitated network design problem (we minimize 1 with respect to constraints 2-4 and variable domains). The solution time was limited to 120 minutes, and the topologies of the best found solutions are found to be very different. Group T4 has large step costs, large demand volumes and small fixed charges, so the solutions consist of a large number of links. The topology is sparser for the instances in groups T1 and T5, and much sparser for the instances in groups T2 and T3.

5.1 Implementational details

Computations are performed on a 750MHz Sun-Blade-1000 with 3Gb physical memory. The Lagrangean subproblems are solved using the CPLEX solver version 6.5.1, and the heuristic has been implemented in C++. The solution time is limited to 120 minutes, which can be considered as fairly short solution time for this difficult problem.

To enable a larger number of iterations, which is important for finding good feasible solutions, the Lagrangean subproblems are solved approximately in the first 90 minutes of the solution time. Preliminary experiments show that a promising solution approach is to relax the integrality requirements on variables z in DS_0 while solving DS_l to optimality, and this approach will be called setting S1. If it is too time-consuming to solve DS_0 using setting S1, the integrality requirements on variables q can be relaxed. If DS_l is too time-consuming to solve, the number of branch-and-bound nodes can be limited to 400. This gives us three additional solution approaches, which are called settings S2, S3 and S4. The solution approaches are summarized in Table 2.

Solving the Lagrangean subproblems approximately affects the quality of the lower bounds, so in order to get strong lower bounds, integrality requirements on variables z

and q are reintroduced when 30 minutes remains of the solution time. In this part of the method, the time limit for solving DS_0 is set to 30 minutes.

A further relaxation used in these tests is to fix r to zero. The reason for this is the large number of these multipliers ($|A||N|^4$), coupled with what we believe is a fairly limited effect of these constraints. Not doing this would most probably require more time for finding good values for the multipliers. If a longer time was allowed for the total solution process, this relaxation should probably not be done, so this is only a practical consideration for these specific tests. Due to the difficulty of the problem, different relaxations must be considered, and here we have three possibilities; Lagrangean relaxation, relaxation without Lagrangean multipliers and relaxation of integer requirements. Note however that with the help of the multipliers s and t constraints 13 and 14 are taken into account.

The Lagrangean multipliers are initialized to zero, and the tabu search heuristic uses parameter values $T = 0.2|A|$ and $K = 50$ as long as no other values are specified. A large tabu list reduces the number of solutions investigated in each iteration, which is preferable for large-scale instances, and a small value of K enables a larger number of iterations to be performed. This allows the heuristic to be restarted from a larger number of initial solutions.

5.2 Experiments on sparse random networks

The solution method has first been used on random networks where each node is connected to at least three other nodes. Computational results are displayed in Tables 3-7, and the tables are organized as follows. The first column shows the numbers of nodes, bidirected arcs and commodities for each test instance. The second column shows the setting used in the experiments. The lower and upper bounds of the optimal objective function value are given in columns $\underline{v}$ and $\bar{v}$, and the next column shows the relative gap. Column *Top* shows the number of (bidirected) links in the best found solution. The next three columns show how large part of the solution time that is spent in the different parts of the method. (An asterix indicates that some subproblem requires more than one hour solution time.) The last column shows the total number of iterations performed.

Table 3 contains computational results for group T1. The table shows that for instances with 14 to 22 nodes, more than 80 percentage of the solution time is spent in the tabu search heuristic, and this indicates that the Lagrangean subproblems are easily solved. For the largest instance, however, setting S1 uses almost 80% of the solution time for solving DS_l, so if these subproblems are approximately solved (as in S2), more than tree times as many iterations can be performed. Settings S1 and S2 use a small part of the available solution time for solving DS_0, so settings S3 and S4 have not been used.

The relative gaps vary between 1.1% and 4.3%, so both solution approaches work well for this group of instances. A difference is that setting S2 finds better feasible solution for some instances.

The instances in group T2 have smaller demand volumes than in group T1, so a smaller part of the objective function value originates from capacity costs. Setting S1 yields relative gaps between 3.7% and 12.3%, and setting S2 yields relative gaps between 3.3%

Table 3: Computational results for group T1.

(N,E,C)	*Setting*	$\underline{v}$	$\bar{v}$	$\frac{\bar{v}-\underline{v}}{\bar{v}}$	*Top*	*Part of sol. time*			*iter*
						DS_0	DS_l	TS	
(14,26,66)	S1	265236	268056	0.011	15	0.07	0.07	0.86	251
	S2	264967	268056	0.012	15	0.08	0.08	0.84	285
(16,29,96)	S1	340848	352242	0.032	17	0.10	0.11	0.80	141
	S2	341200	351992	0.031	18	0.08	0.10	0.81	135
(18,34,112)	S1	401057	414840	0.033	22	0.04	0.13	0.83	85
	S2	402438	411376	0.021	21	0.03	0.10	0.88	83
(20,37,128)	S1	432121	442298	0.023	22	0.05	0.07	0.88	74
	S2	432289	442776	0.024	22	0.05	0.07	0.88	71
(22,41,170)	S1	603911	624144	0.032	28	0.03	0.15	0.82	30
	S2	603597	621490	0.029	28	0.03	0.16	0.81	31
(24,44,190)	S1	647825	676864	0.043	29	0.01	0.78*	0.21	6
	S2	647514	667873	0.030	27	0.03	0.13	0.83	21

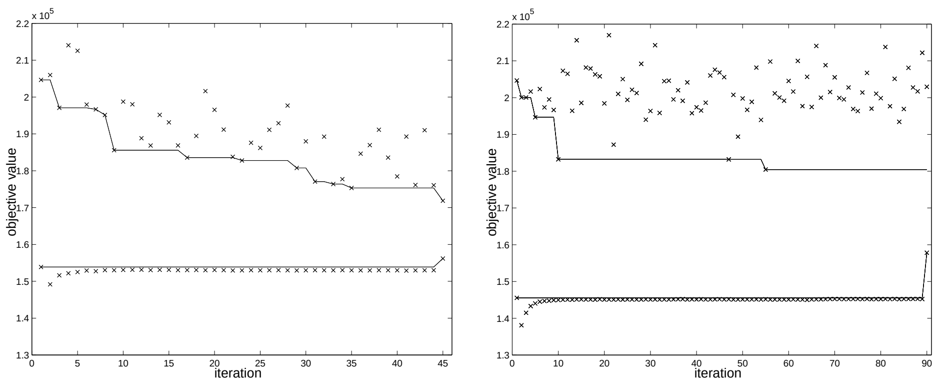

Figure 1: Lower and upper bounds of the optimal objective function value found using S1 and S3 on the 16 node instance in group T2.

and 11.3%. Setting S2 yields better results than S1 for the largest instance since a larger part of the solution time is used for finding good feasible solutions.

The bounds found by settings S1 and S3 are shown in Figure 1. The diagrams illustrate the bounds for the instance with 16 nodes, but they are representative for the other instances as well. The diagrams show that the lower bounds are much improved when integrality requirements on variables z and q are reintroduced. The diagrams also show that S1 improves the upper bound more frequently, so it seems possible that the upper bounds could be further improved if the solution time for S1 is longer, while this might be less likely for S3.

The instances in group T3 are identical to the ones in group T1, except for smaller step costs. Table 5 shows that setting S1 yields the smallest relative gaps for most instances. Settings S3 and S4 usually find slightly stronger lower bounds of the optimal objective function value, but the upper bounds are in general weaker than the ones found by S1 and S2.

The relative gaps are smaller in group T3 than in T2, and this indicates that the

Table 4: Computational results for group T2.

(N,E,C)	*Setting*	$\underline{v}$	$\bar{v}$	$\frac{\bar{v}-\underline{v}}{\bar{v}}$	*Top*	*Part of sol. time* DS_0	DS_l	TS	*iter*
(14,26,66)	S1	130863	135882	0.037	14	0.81	0.01	0.18	43
	S2	131360	135882	0.033	14	0.82	0.01	0.17	45
	S3	135218	151662	0.108	16	0.04	0.06	0.90	137
	S4	134774	152730	0.117	16	0.04	0.06	0.90	140
(16,29,96)	S1	156189	171849	0.091	18	0.50	0.06	0.44	45
	S2	156491	174384	0.103	18	0.48	0.05	0.47	45
	S3	157810	180445	0.125	18	0.03	0.14	0.83	90
	S4	157828	181315	0.130	19	0.03	0.06	0.90	91
(18,34,112)	S1	186431	212591	0.123	20	0.64	0.06	0.31	25
	S2	186658	210325	0.113	19	0.61	0.05	0.47	25
	S3	190565	223408	0.147	21	0.03	0.09	0.88	48
	S4	190257	224912	0.154	22	0.04	0.07	0.89	47
(20,37,128)	S1	218189	230002	0.051	20	0.57	0.03	0.40	22
	S2	218272	234134	0.068	21	0.59	0.03	0.38	24
	S3	219862	241226	0.089	22	0.03	0.05	0.91	37
	S4	218990	238659	0.082	22	0.04	0.06	0.90	40
(22,41,170)	S1	286497	324066	0.116	27	0.17	0.17	0.65	22
	S2	287582	324365	0.113	26	0.18	0.10	0.72	23
	S3	288628	333069	0.133	29	0.01	0.67	0.32	9
	S4	288468	327512	0.119	27	0.02	0.16	0.82	24
(24,44,190)	S1	297599	330104	0.098	27	0.05	0.79*	0.16	5
	S2	297599	310902	0.043	24	0.43	0.13	0.44	17
	S3	299444	341260	0.122	29	0.00	0.90*	0.10	6
	S4	299337	332656	0.100	29	0.03	0.21	0.76	17

Table 5: Computational results for group T3.

(N, E, C)	*Setting*	$\underline{v}$	$\bar{v}$	$\frac{\bar{v}-\underline{v}}{\bar{v}}$	*Top*	*Part of sol. time* DS_0	DS_l	TS	*iter*
(14,26,66)	S1	112223	112435	0.002	13	0.83	0.03	0.14	48
	S2	112018	112435	0.004	13	0.78	0.02	0.20	77
	S3	112375	122722	0.084	15	0.04	0.06	0.90	119
	S4	112329	127199	0.117	16	0.04	0.07	0.89	130
(16,29,96)	S1	141124	152269	0.073	18	0.53	0.06	0.41	38
	S2	141281	153198	0.078	18	0.54	0.05	0.40	41
	S3	141479	157165	0.099	18	0.03	0.24	0.73	68
	S4	141513	157629	0.102	19	0.03	0.12	0.85	78
(18,34,112)	S1	157235	157896	0.004	17	0.51	0.06	0.43	37
	S2	157068	167280	0.061	19	0.53	0.05	0.41	35
	S3	157776	177482	0.111	20	0.03	0.09	0.88	48
	S4	157758	180398	0.125	21	0.03	0.08	0.88	50
(20,37,128)	S1	180719	181337	0.003	20	0.69	0.03	0.29	22
	S2	180755	181585	0.005	20	0.71	0.03	0.26	22
	S3	181204	193114	0.062	22	0.04	0.12	0.85	36
	S4	181204	192446	0.058	22	0.04	0.09	0.87	39
(22,41,170)	S1	233629	254641	0.083	27	0.40	0.17	0.43	13
	S2	233387	260537	0.104	29	0.50	0.08	0.42	14
	S3	233519	260776	0.105	29	0.01	0.77	0.23	7
	S4	233717	252434	0.074	27	0.03	0.23	0.74	24
(24,44,190)	S1	256700	266576	0.037	26	0.16	0.71*	0.14	5
	S2	256700	267857	0.041	27	0.46	0.08	0.46	10
	S3	256362	266576	0.038	26	0.01	0.78	0.21	5
	S4	256872	273680	0.061	28	0.04	0.20	0.76	17

heuristic is better at handling large demand volumes than large step costs. We note that in a few cases the gaps are less than 0.5%.

Group T4 contains instances with small fixed charges, large step costs and large demand volumes. Table 6 shows that the topologies of the best found solutions are denser compared to the previous groups. The tabu search heuristic has previously been successful for instances with large demand volumes, so it is not surprising that the best found solutions are less than 5.3% worse than the optimal solution.

Finally, group T5 is comparable with group T1, since the demand volumes and the fixed charges has been decreased by similar factors. The solution method generates relative gaps between 3.5% and 8.9%, which is approximately twice as much as for group T1.

Our initial experiments on sparse random networks have shown that setting S2 seems to be a suitable solution approach when the fixed charges are not too large compared to the costs for capacity. This solution approach does not work as well when the fixed charges are too large, and this is because subproblem DS_0 is very time-consuming to solve for such instances.

Table 6: Computational results for group T4.

(N,E,C)	*Setting*	$\underline{v}$	$\bar{v}$	$\frac{\bar{v}-\underline{v}}{\bar{v}}$	*Top*	*Part of sol. time* DS_0	DS_l	TS	*iter*
(14,26,66)	S1	200514	209048	0.041	21	0.04	0.06	0.90	223
	S2	200484	209048	0.041	21	0.04	0.06	0.90	226
(16,29,96)	S1	274717	290240	0.053	23	0.02	0.08	0.90	106
	S2	275722	289150	0.046	23	0.02	0.08	0.90	114
(18,34,112)	S1	332046	344575	0.036	23	0.04	0.15	0.81	94
	S2	331776	344575	0.037	23	0.03	0.10	0.87	91
(20,37,128)	S1	356791	367664	0.030	24	0.03	0.08	0.89	72
	S2	356823	367664	0.029	24	0.03	0.08	0.89	72
(22,41,170)	S1	513225	534326	0.039	33	0.01	0.30	0.68	27
	S2	513353	532474	0.036	33	0.02	0.13	0.86	33
(24,44,190)	S1	551233	577776	0.046	34	0.00	0.87*	0.13	6
	S2	551832	577268	0.044	33	0.01	0.17	0.82	26

Table 7: Computational results for group T5.

(N,E,C)	*Setting*	$\underline{v}$	$\bar{v}$	$\frac{\bar{v}-\underline{v}}{\bar{v}}$	*Top*	*Part of sol. time* DS_0	DS_l	TS	*iter*
(14,26,66)	S1	78859	84498	0.067	15	0.44	0.04	0.52	124
	S2	78839	84498	0.067	15	0.45	0.03	0.52	124
(16,29,96)	S1	103155	108190	0.047	17	0.22	0.12	0.66	90
	S2	102985	106671	0.035	17	0.22	0.10	0.68	86
(18,34,112)	S1	130504	144460	0.097	21	0.20	0.13	0.68	64
	S2	131312	144160	0.089	20	0.21	0.07	0.72	68
(20,37,128)	S1	152168	160715	0.053	21	0.22	0.06	0.71	57
	S2	152153	160715	0.053	21	0.23	0.06	0.71	57
(22,41,170)	S1	213682	235431	0.092	29	0.04	0.39	0.57	21
	S2	214091	232888	0.081	29	0.07	0.15	0.78	33
(24,44,190)	S1	220827	246354	0.104	33	0.01	0.80*	0.19	6
	S2	219976	234414	0.062	28	0.07	0.19	0.73	28

Table 8: Computational results for setting S2 and denser networks.

		S2/TS1				S2/TS2			
(N, E, C)	*Group*	$\underline{v}$	$\bar{v}$	$\frac{\bar{v}-\underline{v}}{\bar{v}}$	*Top*	$\underline{v}$	$\bar{v}$	$\frac{\bar{v}-\underline{v}}{\bar{v}}$	*Top*
(14,33,66)	T1	209406	215450	0.028	14	211914	215555	0.017	14
	T4	154721	163381	0.053	17	154909	163384	0.052	17
	T5	73038	82390	0.114	15	74026	78628	0.059	14
(16,38,96)	T1	295759	303373	0.025	17	297313	303373	0.020	17
	T4	233451	247766	0.058	23	233063	245891	0.052	23
	T5	106627	123560	0.137	18	107110	123268	0.131	19
(18,43,112)	T1	359809	387308	0.070	23	361097	375889	0.039	22
	T4	279959	300406	0.068	25	279992	298213	0.061	27
	T5	129811	159251	0.185	30	129725	149424	0.132	23
(20,52,128)	T1	377050	418140	0.098	27	377590	402828	0.063	22
	T4	293880	317334	0.074	30	294807	313830	0.061	27
	T5	130412	169280	0.229	33	130412	150918	0.136	26
(22,56,170)	T1	573489	633420	0.095	34	573683	622856	0.079	33
	T4	472785	515077	0.082	40	473346	510541	0.073	34
	T5	182326	229319	0.205	38	182326	213582	0.146	33
(24,58,190)	T1	648663	716492	0.095	39	649058	680643	0.046	29
	T4	544411	591854	0.080	43	545082	576686	0.055	36
	T5	208101	252546	0.176	39	208101	234691	0.113	34

5.3 Experiments on dense random networks

We now study random networks with a larger number of bidirected links (each node is connected to at least four other nodes). The dense and the sparse instances has the same structure of demands and the same structure of link costs. We use setting S2 for the instances in groups T1, T4 and T5, since this solution approach has been preferable in our previous experiments. We use setting S4 for the instances in groups T2 and T3, since DS_0 is too hard to solve to optimality for larger instances when the fixed charges are dominating and when z has to be integer-valued.

Two different versions of the tabu search heuristic will be used, TS1 and TS2. TS1 is the version used on sparse random networks, and it evaluates all solutions in the neighborhood in each iteration and terminates when the best found solution has not been updated for 50 iterations. TS2 uses the same neighborhood as TS1, but the entire neighborhood is only evaluated when a best found solution recently has been found. If more than five iterations has passed since the best found solution was updated, each neighbor solution is evaluated with probability 0.15 and discarded otherwise. TS2 is terminated when the best found solution has not been improved for 300 iterations.

Computational results for groups T1, T4 and T5 are displayed in Table 8. The table shows that setting S2/TS1 finds the best feasible solution in 2 instances, setting S2/TS2 finds the best feasible solution in 15 instances, and that the two settings find feasible solutions with the same objective value in one instance. The strongest lower bound is usually found using S2/TS2, so this approach yields the smallest relative gap in all

Table 9: Computational results for setting S4 and denser networks.

		S4/TS1				S4/TS2			
(N, E, C)	*Group*	$\underline{v}$	$\bar{v}$	$\frac{\bar{v}-\underline{v}}{\bar{v}}$	*Top*	$\underline{v}$	$\bar{v}$	$\frac{\bar{v}-\underline{v}}{\bar{v}}$	*Top*
(14,33,66)	T2	124375	126440	0.016	13	123959	125782	0.014	13
	T3	102247	104709	0.024	13	102233	103675	0.013	13
(16,38,96)	T2	160400	176830	0.093	16	160063	174940	0.085	16
	T3	128568	133506	0.037	16	132595	128565	0.030	15
(18,43,112)	T2	192438	222715	0.136	20	191494	213978	0.105	19
	T3	151119	165346	0.086	19	151325	163516	0.075	19
(20,52,128)	T2	197901	230904	0.143	23	197955	227588	0.130	21
	T3	161359	181572	0.111	21	160269	179760	0.108	21
(22,56,170)	T2	257812	312494	0.175	27	257194	308817	0.167	25
	T3	208583	250094	0.166	28	208583	243341	0.143	25
(24,58,190)	T2	279391	326257	0.143	27	278911	329504	0.154	27
	T3	231089	265920	0.131	26	231089	264168	0.125	24

Table 10: Real life networks.

Network	(N, E, C)	Fixed charge	Step cost	Routing Cost
POLSKA	(12,18,66)	Yes	Yes (2)	No
FRANCE	(25,45,300)	Yes	Yes (1)	No

instances. The table clearly shows that solution approach S2/TS2 is preferable for the instances in groups T1, T4 and T5, especially for large-scale instances.

Computational results for groups T2 and T3 are displayed in Table 9. The table shows that solution approach S4/TS2 yields smaller relative gaps than S4/TS1 for all instances except one, but the relative gaps are in most cases quite large. If the fixed charges are dominating, it is even more important to solve DS_0 accurately in order to get strong lower bounds of the objective function value. When DS_0 is solved with integer requirements in the last iteration, the gap is still very large when the maximal solution time is reached, and this explains the large relative gaps.

5.4 Experiments on real life networks

We have finally used some real life instances from the *survivable network design data library*, SNDlib 1.0 (2005), see Table 10. The first two columns show the name and the size of each network, and the remaining columns describe the cost structure. The parenthesis in column four shows if different types of capacity modules are available, and (2) means that small and large capacity steps can be installed.

The network called FRANCE fits exactly into the SCIPND model. The POLSKA network allows small and large capacity steps to be installed, and this possibility is not considered in the SCIPND model. We have therefore constructed two new instances, POLSKA-1 and POLSKA-2, where it is only possible to install small capacity steps in the POLSKA-1

Table 11: Computational results for real life networks.

Network	*Setting*	$\underline{v}$	$\bar{v}$	$\frac{\bar{v}-\underline{v}}{\bar{v}}$	*Top*
POLSKA-1	S2/TS1	34746	35418	0.019	18
POLSKA-1	S2/TS2	34770	35418	0.018	18
POLSKA-2	S2/TS1	27798	28886	0.038	14
POLSKA-2	S2/TS2	27808	28612	0.028	17
FRANCE	S2/TS1	50695	60457	0.161	25
FRANCE	S2/TS2	50849	57714	0.118	29
FRANCE	S4/TS1	50324	91596	0.451	38
FRANCE	S4/TS2	51156	60523	0.155	26

network and large capacity steps in the POLSKA-2 network.

The POLSKA-1 network has demand volumes that are in the same approximate size as a capacity step and the fixed charges are equal to the costs for one capacity step, so this instance is similar to the instances in group T1. The capacity steps and the capacity costs are larger for the POLSKA-2 network, but the fixed charges are the same, so this instance is more similar to the instances in group T5. We use solution approach S2/TS2 for both instances since this approach has been preferable for groups T1 and T5. We use the same solution approach for the FRANCE network, but also S4/TS2 since this instance is quite large. The FRANCE network has small demand volumes compared to the size of a capacity steps so we expect this network to be more challenging for our solution method. Computational results are displayed in Table 11.

The relative gaps are very small for POLSKA-1 and POLSKA-2, so for these instances our solution approach perform well. Setting S2/TS2 yields the best result for the FRANCE network. Due to the problem size and the small demand volumes, this problem really would need a longer solution time, so the larger relative gaps are not unexpected. We have noticed that TS1 generates solutions with similar (or the same) objective function values in subsequent iterations, while the solutions generated by TS2 are more different. This indicates that the regions around a local optimum can be left more rapidly when a smaller part of the neighborhood is investigated in each iteration, and this is why S4/TS2 is capable of generating reasonable good solutions even if the initial solution is poor.

6 Conclusions

We present a new mixed integer model for the OSPF network design problem. The model is based on routing patterns in the form of in-graphs and does not explicitly include OSPF link weights. The routing protocol is modeled by ensuring that all pairs of routing patterns are subpath consistent, which is a previously known necessary condition for the existence of compatible weights. The equal-cost multi-path principle is modeled without introducing any additional variables. The validity of this complex model is proved, and its solvability is demonstrated by a solution method based on Lagrangean relaxation and subgradient optimization. Feasible solutions are generated using a tabu search heuristic. Computational tests are performed on random networks and on real life networks from SNDlib, and computational results show that the relative maximal errors of the solutions usually are decreased to reasonable levels, even when the available solution time is fairly limited.

Acknowledgment: This work has been financed by the Swedish Research Council.

References

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993), Network Flows. Theory, Algorithms and Applications, Prentice Hall.

Ben-Ameur, W., Michel, N., Liau, B., and Gourdin, E. (2001), “Routing strategies for IP networks”, Telektronikk 2/3, 145–158.

Ben-Ameur, W., and Gourdin, E. (2003), “Internet routing and related topology issues”, SIAM Journal on Discrete Mathematics 17, 18–49.

Bley, A. (2003), “A Lagrangian approach for integrated network design and routing in IP networks”, in: Proceedings of 1st International Network Optimization Conference INOC 2003, 107–113.

Bley, A., Grötschel, M., and Wessäly, R. (2000), “Design of broadband virtual private private networks: Model and heuristics for the B-WiN”, in: Dean, N., Hsu, D. F., and Rav, R. (eds.), Robust Communication Networks : Interconnection and Survivability, vol. 53 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science, AMS, 1–16.

Bley, A., and Koch, T. (2002), “Integer programming approaches to access and backbone IP-network planning”, Technical report ZR-02-41, Konrad-Zuse-Zentrum für Informationstechnik, Berlin.

Broström, P., and Holmberg, K. (2005), “Design of IP/OSPF networks using a Lagrangean heuristic on an in-graph based model”, in: Gouveia, L., and Mourao, C. (eds.), INOC 2005, 702–709. University of Lisbon, Lisbon, Portugal.

Broström, P., and Holmberg, K. (2006), “Multiobjective design of survivable IP networks”, Annals of Operations Research 147, 235–253.

Broström, P., and Holmberg, K. (2007), “Compatible weights and valid cycles in non-spanning OSPF routing patterns”, Research report LiTH-MAT-R-2007-04, Department of Mathematics, Linköping Institute of Technology, Sweden. Accepted for publication in *Algorithmic Operations Research*.

Broström, P., and Holmberg, K. (2008), “Valid cycles: A source of infeasibility in OSPF routing”, Accepted for publication in Networks. Online Jan 11, 2008.

Chu, Y. J., and Liu, T. H. (1965), “On the shortest arborescence of a directed graph”, Science Sinica 14, 1396–1400.

Crainic, T. G., Frangioni, A., and Gendron, B. (2001), “Bundle-based relaxation methods for multicommodity capacitated fixed charge network design”, Discrete Applied Mathematics 112, 73–99.

De Giovanni, L., Fortz, B., and Labbé, M. (2005), “A lower bound for the internet protocol network design problem”, in: Gouveia, L., and Mourao, C. (eds.), INOC 2005, 401–408. University of Lisbon, Lisbon, Portugal.

Edmonds, J. (1967), “Optimum branchings”, Journal of research of the National Bureau of Standards 71B, 233–240.

Fischetti, M., and Toth, P. (1993), "An efficient algorithm for the min-sum arborescence problem on complete digraphs", ORSA Journal on Computing 5/4, 426–434.

Frank, A. (1995), "Connectivity and network flows", in: Graham, A., Grötschel, M., and Lovász, L. (eds.), Handbook of Combinatorics, vol. 1, North-Holland, 111–177.

Gondran, M., and Minoux, M. (1984), Graphs and Algorithms, Wiley-Interscience.

Held, M., Wolfe, P., and Crowder, H. P. (1974), "Validation of subgradient optimization", Mathematical Programming 6, 62–88.

Holmberg, K., and Yuan, D. (1998), "A Lagrangean approach to network design problems", International Transactions in Operational Research 5, 529–539.

Holmberg, K., and Yuan, D. (2004), "Optimization of Internet Protocol network design and routing", Networks 43, 39–53.

Moy, J. (1998). "OSPF: Anatomy of an internet routing protocol", Addison-Wesley.

Poljak, B. T. (1967), "A general method of solving extremum problems", Soviet Mathematics Doklady 8, 593–597.

Prytz, M. (2002), On Optimization in Design of Telecommunications Networks with Multicast and Unicast Traffic. Phd dissertation, Royal Institute of Technology, Sweden. TRITA-MAT-02-OS-05.

Shor, N. Z. (1985), Minimization Methods for Non-Differentiable Functions, Springer-Verlag, Berlin.

SNDlib 1.0 (2005). "SNDlib 1.0 – Survivable network design data library", http://sndlib.zib.de.

Staehle, D., Köhler, S., and Kohlhaas, U. (2000), "Towards an optimization of the routing parameters for IP networks", Technical report 258, Department of Computer Science, University of Würzburg, Germany.

Tomaszewski, A., Pioro, M., Dzida, M., and Zagozdzon, M. (2005), "Optimization of administrative weights in IP networks using the branch-and-cut approach", in: Gouveia, L., and Mourao, C. (eds.), INOC 2005, 393–400. University of Lisbon, Lisbon, Portugal.