Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

ANDREAS BLEY

An Integer Programming Algorithm for Routing Optimization in IP Networks

ZIB-Report 08-30 (Extended Version, November 2009)

An Integer Programming Algorithm for Routing Optimization in IP Networks*

Andreas Bley

TU Berlin, Institute of Mathematics
Straße des 17. Juni 136, D-10623 Berlin, Germany
bley@math.tu-berlin.de

Abstract Most data networks nowadays use shortest path protocols to route the traffic. Given administrative routing lengths for the links of the network, all data packets are sent along shortest paths with respect to these lengths from their source to their destination.
In this paper, we present an integer programming algorithm for the minimum congestion unsplittable shortest path routing problem, which arises in the operational planning of such networks. Given a capacitated directed graph and a set of communication demands, the goal is to find routing lengths that define a unique shortest path for each demand and minimize the maximum congestion over all links in the resulting routing. We illustrate the general decomposition approach our algorithm is based on, present the integer and linear programming models used to solve the master and the client problem, and discuss the most important implementational aspects. Finally, we report computational results for various benchmark problems, which demonstrate the efficiency of our algorithm.
Keywords: Shortest Path Routing, Integer Programming

1 Introduction

In this paper, we present an integer programming algorithm to optimize the routing in communication networks based on shortest path routing protocols such as OSPF [25] or IS-IS [17], which are widely used in the Internet. With these routing protocols, all end-to-end traffic streams are routed along shortest paths with respect to some administrative link lengths (or routing weights), that form the so-called routing metric.

The simplicity of this routing policy offers many operational advantages in practice. Shortest path routing permits the use of decentralized and distributed routing algorithms, it has very good scaling properties with respect to the network size, and it causes only little administrative overhead compared to connection oriented routing policies. From the planning perspective, however, shortest path routing is extremely complicated. As all routing paths depend on the same shortest path metric, it is not possible to configure the end-to-end routing paths

* This work has been supported by the DFG Research Center MATHEON "Mathematics for key technologies" in Berlin.

for different communication demands individually. The routing can be controlled only as a whole and only indirectly by changing the routing metric. Finding a routing metric that induces a set of globally efficient end-to-end routing paths is a major difficulty, as the shortest path routing paradigm enforces rather complicated and subtle interdependencies among the paths that comprise a valid routing.

In this paper, we consider the unsplittable shortest path routing variant, where the lengths must be chosen such that the shortest paths are unique and each traffic demand is sent unsplit via its single shortest path. This additional requirement is imposed by several network operators, among them the DFN Verein operating the German national research and education network. Shortest multi-path routing protocols, such as the commonly used equal cost multi-path (ECMP) extension of the open shortest path first protocol (OSPF), permit the use of multiple routing paths for each traffic demand. The mechanisms used to distribute the traffic among these paths, however, in practice either achieve only a rough approximation of the prescribed splitting ratios or may lead to packet reordering and other undesired effects of sending the data packets of a single communication session along different paths. Restricting to the unsplittable shortest path routing variant avoids all these difficulties, permits full traceability of all traffic streams, and sometimes even simplifies the network management.

One of the most important operational planning tasks in communication networks is traffic engineering. Its goal is to improve the service quality in the existing network by (re-)optimizing the routing of the traffic, but leaving the network topology and hardware configuration unchanged. Mathematically, this task can be formulated as the minimum congestion unsplittable shortest path routing problem (Min-Con-USPR). The problem input consists of a directed graph $D = (V, A)$ with arc capacities $c_a \in \mathbb{Z}$ for all $a \in A$, and a set of directed commodities $K \subseteq V \times V$ with demand values $d_{st} \in \mathbb{Z}$ for all $(s, t) \in K$. A feasible solution is an unsplittable shortest path routing (USPR) of the commodities, i.e., a metric of link lengths $\lambda_a \in \mathbb{Z}$, $a \in A$, that induce a unique shortest (s, t)-path for each commodity $(s, t) \in K$. Each commodity's demand is sent unsplit along its shortest path. The objective is to minimize the maximum congestion, i.e., the maximum flow to capacity ratio over all arcs. The maximum congestion is a good measure for the overall network service quality and typically used as a key indicator in traffic engineering.

Due to their great practical relevance, shortest path routing problems have been studied quite intensively in the last decade. Ben-Ameur and Gourdin [2], Broström and Holmberg [14,15] studied the combinatorial properties of path sets that correspond to shortest (multi-)path routings and devised linear programming models to find lengths that induce a set of presumed shortest paths (or prove that no such lengths exist). Bley [4,7], on the other hand, showed that finding a smallest shortest-path conflict in a set of presumed shortest paths or the smallest integer lengths inducing these paths is $\mathcal{NP}$-hard. Furthermore, Bley [6] proved that Min-Con-USPR is inapproximable within a factor of $\Omega(|V|^{1-\epsilon})$ for

any $\epsilon > 0$ and proposed polynomial time approximation algorithms for several special cases of MIN-CON-USPR and related network design problems. Furthermore, In the same paper, Bley also presented examples where the smallest link congestion that can be obtained with unsplittable shortest path routing exceeds the congestion that can be obtained with multi-commodity flow or unsplittable flow routing by a factor of $\Omega(|V|^2)$. The minimum congestion shortest multi-path routing problem has been shown to be inapproximable within a factor less than 3/2 by Fortz and Thorup [19].

Various approaches for the solution of network design and routing problems in shortest path networks have been proposed. Algorithms using local search, simulated annealing, or Lagrangian relaxation techniques with the routing lengths as primary decision variables are presented in [3,9,16,18,19,34], for example. These length-based methods work well for shortest multi-path routing problems, where traffic may be split among several equally long shortest paths, but they often produce only suboptimal solutions for hard unsplittable shortest path routing problems. As they deliver no or only weak quality guarantees, they cannot guarantee to find provenly optimal solutions.

Using mixed integer programming formulations that contain variables for the routing lengths as well as for the resulting shortest paths and traffic flows, shortest path routing problems can – in principle – be solved to optimality. Formulations of this type are discussed in [9,8,20,28,30,33], for example. Unfortunately, the relation between the shortest paths and the routing length always leads to quadratic or very large big-M models, which are computationally extremely hard and not suitable for practical problems.

In this paper, we present an integer programming algorithm that – similar to Bender's decomposition – decomposes the minimum congestion unsplittable shortest path routing problem into the master problem of finding the optimal end-to-end routing paths and the client problem of finding a routing metric that induces these paths. The main advantage of this decomposition approach is that it permits the use of advanced integer linear programming techniques for unsplittable multi-commodity flow problems to compute provenly optimal routings also for real-world scale problems. An implementation of this algorithm is used successfully in the planning of the German national education and research network for several years [10,11,5]. Modifications of this decomposition approach for shortest multi-path and multicast routing problems are discussed in [12,21,31,32,33].

The remainder of this paper is organized as follows. In Section 2, we formally define the minimum congestion unsplittable shortest path routing problem and introduce the notions and notations used throughout this paper. The overall decomposition algorithm and the models and sub-algorithms used for the solution of the master and the client problem are described in Section 3. In Section 4 we discuss additional cutting planes, which proved to be essential for the practical performance of the algorithm. The heuristics used to compute primal solutions are discussed in Section 5, further important aspects of our implementation are discussed in Section 6. In Section 7, we finally report on numerical results ob-

tained with this algorithm for numerous real-world and benchmark problems and illustrate the relevance of optimizing the routing in practice.

2 Notation and Preliminaries

Let $D = (V, A)$ be a directed graph with arc capacities $c_a \in \mathbb{Z}$ for all $a \in A$ and let $K \subseteq V \times V$ be a set of directed commodities with demand values $d_{st} \in \mathbb{Z}$ for all $(s,t) \in K$. For each commodity $(s,t) \in K$, let $\mathcal{P}(s,t)$ denote the set of all (s,t)-paths in D. Furthermore, let $\mathcal{P} := \bigcup_{(s,t)\in K} \mathcal{P}(s,t)$.

Definition 1. *We say that a metric $\lambda = (\lambda_a) \in \mathbb{R}^A$ defines an* unsplittable shortest path routing (USPR) *for the commodity set K, if the shortest (s,t)-path with respect to λ is* uniquely determined *for each commodity $(s,t) \in K$. We denote the shortest (s,t)-path with respect to λ by $P^*_{st}(\lambda)$.*

The demand of each commodity is routed unsplit along the respective shortest path. For a metric λ that defines such an USPR, the total flow through an arc $a \in A$ then is

$$f_a(\lambda) := \sum_{(s,t)\in K:\, a\in P^*_{st}(\lambda)} d_{st} \ . \tag{1}$$

The task in the minimum congestion unsplittable shortest path routing problem Min-Con-USPR is to find a metric $\lambda \in \mathbb{Z}^A$ that defines an USPR for the given commodity set K and minimizes the maximum congestion $L := \max\{f_a(\lambda)/c_a : a \in A\}$. Formally, this problem is defined as follows:

Problem:	Min-Con-USPR
Instance:	A digraph $D = (V, A)$ with arc capacities $c_a \in \mathbb{Z}$, $a \in A$, and a commodity set $K \subseteq V \times V$ with demands $d_{(s,t)} \in \mathbb{Z}$, $(s,t) \in K$.
Solution:	A metric $\lambda \in \mathbb{Z}^A$, such that the shortest (s,t)-path w.r.t λ is uniquely determined for each commodity $(s,t) \in K$.
Objective:	$\min\left(\max_{a\in A} f_a(\lambda)/c_a\right)$, where $f_a(\lambda)$ is as defined in (1).

We may assume without loss of generality that D contains an (s,t)-path for each commodity $(s,t) \in K$ and that D is simple: Loops cannot be contained in a uniquely determined shortest path and, if two parallel arcs were contained in two commodities' routing paths, then none of these paths would be a unique shortest path. Furthermore, we may assume that there are no parallel commodities: If there were two or more parallel commodities from s to t, these would have to use the same (uniquely determined shortest) flow path in any unsplittable shortest path routing and, therefore, could be aggregated into one commodity.

For any bijection $\mathrm{idx} : A \leftrightarrow \{1, \ldots, |A|\}$, the metric $\lambda_a := 2^{\mathrm{idx}(a)}$ induces unique shortest paths between all node pairs. Hence, any instance of Min-Con-USPR has a feasible solution, provided that the underlying digraph D contains at least one (s,t)-path for each $(s,t) \in K$. One easily observes that Min-Con-USPR contains the classical disjoint paths problem as a special case. This immediately implies that it is $\mathcal{NP}$-hard to approximate Min-Con-USPR within

a factor strictly less than 2 even in the special case where all arc capacities are 1 and only two commodities with demand value 1 are given. For general arc capacities and commodities, MIN-CON-USPR is hard to approximate within a factor $\Omega(|V|^{1-\epsilon})$ for any $\epsilon > 0$; see [6].

Note that the problem MIN-CON-USPR defined above permits arbitrarily large values for the routing lengths λ_a. The routing protocols used in practice, however, can handle only small lengths. The most commonly used shortest path routing protocol OSPF, for example, permits only lengths between 1 and $2^{16}-1$. Nevertheless, we can safely neglect the upper bounds on the lengths for the purpose of routing planning. For the currently deployed routing protocols, they impose no binding constraints for practically interesting network sizes. We may assume that two routing metrics are equivalent for MIN-CON-USPR if they induce the same set of shortest paths.

The integer programming algorithm discussed in this paper is based on a special representation of the shortest path routing, which describes the routing as a set of node-arc pairs $F \subset V \times A$. For the USPR defined by a metric λ, the corresponding node-arc set $F(\lambda)$ is $F(\lambda) := \{(t, a) : a \in P^*_{st}(\lambda) \text{ for some } (s, t) \in K\}$, that is, $F(\lambda)$ contains all those node-arc pairs (t, a), for which a is contained in a (shortest) routing path towards destination t. In the MIN-CON-USPR problem, we are interested in all those sets $F \subset V \times A$ that correspond to a valid unique shortest path routing.

Definition 2.

(i) A metric λ is said to be compatible *with a set of node-arc pairs $F \subset V \times A$, if, for each $(t, a) \in F$, arc $a = (u, v)$ is contained in* all *shortest (u, t)-paths with respect to λ.*

(ii) A set $F \subset V \times A$ is a unique shortest path forwarding (USPF) *if there exists a compatible metric for F. Otherwise we call F an* (USPF-) conflict.

Note that this definition of USPF is independent from the commodity set K. One easily verifies that a set F is an USPF if and only if there exists a metric λ that defines a unique shortest path between all node pairs in D and, for each given pair $(t, a) \in F$, arc $a = (u, v)$ is contained in the unique shortest path from u to t. Note that a given USPF does not necessarily define the shortest path for each commodity. Furthermore, a USPF does not necessarily define complete end-to-end paths; it may as well prescribe only some arcs to be on the shortest paths towards some destination, but still leave some choice about the complete end-to-end shortest paths.

Clearly, any subset (including the empty set) of an USPF is an USPF as well. Hence, the family of all USPF in the digraph D forms an independence system $\mathcal{I} \subset 2^{V \times A}$. The circuits of this independence system are exactly the irreducible (i.e., inclusion-wise minimal) USPF conflicts. In the following, we denote the family of all irreducible conflicts by $\mathcal{C} \subset 2^{V \times A}$. Based on the independence system of unique shortest path forwardings, we immediately get the following completely characterization of all valid unsplittable shortest path routings for a given set of commodities.

Lemma 3 ([5]). *The path set $Q \subseteq \mathcal{P}$ is (the path set of) a valid unique shortest path routing for the commodity set K if and only if*

(i) $|Q \cap \mathcal{P}(s,t)| = 1$ *for all* $(s,t) \in K$*, i.e.,* Q *contains exactly one path for each commodity, and*
(ii) no irreducible conflict $C \in \mathcal{C}$ *is fully contained in the set* $F := \bigcup_{P \in Q} \{(t,a) : t$ *is destination of* P*,* $a \in P\}$*.*

The independence system $\mathcal{I}$ of all unique shortest path forwardings can be extremely complex; see [5]. In general digraphs, the rank quotient of $\mathcal{I}$ may be arbitrarily small and the size of irreducible conflicts may be arbitrarily large. Given an arbitrary set $F \subset V \times A$, it is $\mathcal{NP}$-hard to approximate the size (with respect to the number of node-arc pairs) of the largest USPF contained in F by a factor less than $8/7$. It is also $\mathcal{NP}$-hard to approximate the size of the smallest conflict in F by a factor less than $7/6$ [7]. However, as we will see in Section 3.2, one can decide in polynomial time whether or not a given set $F \subset V \times A$ is a valid USPF and, depending on that, either find a compatible metric or some (not necessarily minimal) irreducible conflict in F. This is the foundation of the algorithm described in the following section.

3 Integer Programming Algorithm

Similar to Benders' decomposition, our algorithm decomposes the problem of finding an optimal shortest path routing into the master problem of finding the optimal end-to-end paths and the client problem of finding compatible routing lengths for these paths.

The master problem is formulated as an integer linear program and solved with a branch-and-cut algorithm. Instead of using routing weight variables, the underlying formulation contains special inequalities based on USPF conflicts to exclude routing path configurations that are no valid unsplittable shortest path routings. These inequalities are generated dynamically as cutting planes by the client problem during the execution or the branch-and-cut algorithm.

Given a set of routing paths computed by the master problem's branch-and-cut algorithm, the client problem then is to find a metric of routing lengths that induce exactly these paths. As we will see in Section 3.2, this problem can be formulated and solved as a linear program. If the given paths indeed form a valid shortest path routing, the solution of this linear program yields a compatible metric. If the given paths do not form a valid unsplittable shortest path routing, the client linear program is infeasible. In this case, the given routing paths contain a conflict that must not occur in any admissible shortest path routing. This conflict, which can be derived from the dual solution of the infeasible client linear program, then can be turned into an inequality for the master problem, which is valid for all admissible shortest path routings, but violated by the current routing. Adding this inequality to the master problem, we then cut off the current non-admissible routing and proceed with the master branch-and-cut algorithm to compute another candidate routing.

3.1 Master Problem

There are several ways to formulate the master problem of Min-Con-USPR as a mixed integer program. For notational simplicity, we present a variation of the disaggregated arc-routing formulation used in our algorithm, which contains additional artificial variables that describe the unique shortest path forwarding defined by the routing.

The primary decision variables used in this formulation are the variables $x_a^{st} \in \{0,1\}$ for all $(s,t) \in K$ and $a \in A$. These variables describe which arcs are contained in the routing paths. Variable x_a^{st} is supposed to be 1 if and only if arc a is contained in the routing path for commodity (s,t). A single variable $L \in \mathbb{R}$ represents the maximum congestion that is attained by the routing. The additional artificial variables $y_a^t \in \{0,1\}$ for all $t \in V$ and $a \in A$ describe the forwarding defined by the routing paths. Variable y_a^t is supposed to be 1 if there is a routing path towards t that contains arc a. With these variables the master problem of Min-Con-USPR can be formulated as follows:

$$\min \quad L \tag{2a}$$

$$\text{s.t.} \quad \sum_{a \in \delta^+(v)} x_a^{st} - \sum_{a \in \delta^-(v)} x_a^{st} = \begin{cases} -1 & \text{if } v = s \\ 1 & \text{if } v = t \\ 0 & \text{else} \end{cases} \qquad (s,t) \in K,\ v \in V \tag{2b}$$

$$\sum_{(s,t) \in K} d_{st} x_a^{st} \leq c_a L \qquad a \in A \tag{2c}$$

$$x_a^{st} \leq y_a^t \qquad (s,t) \in K,\ a \in A \tag{2d}$$

$$\sum_{a \in \delta^+(v)} y_a^t \leq 1 \qquad t \in V,\ v \in V \tag{2e}$$

$$\sum_{(a,t) \in C} y_a^t \leq |C| - 1 \qquad C \in \mathcal{C} \tag{2f}$$

$$x_a^{st} \in \{0,1\} \qquad (s,t) \in K,\ a \in A \tag{2g}$$

$$L \geq 0. \tag{2h}$$

Subproblem (2a)–(2c) together with the integrality and non-negativity constraints (2g) and (2h) is a standard arc-routing formulation for the unsplittable multi-commodity flow problem, whose objective is to minimize the congestion L.

Inequalities (2d) force the artificial variables y_a^t to be (at least) 1 for all arcs a that are contained in some routing path towards destination t. Together with the out-degree constraints (2e) this ensures that, for each destination $t \in V$, the routing paths towards t form an anti-arborescence, which is clearly necessary for the paths in any valid unsplittable shortest path routing.

Constraints (2f) finally ensure that no integer solution of (2) contains all node-arc pairs of any (irreducible) USPF-conflict $C \in \mathcal{C}$. As the irreducible conflicts are exactly the circuits of the independence system formed by all valid

unique shortest path forwardings, this implies that the artificial variables y_a^t describe a valid USPF. Consequently, the routing given by any integer feasible solution of (2) is a valid unsplittable shortest path routing. In general, the number of these conflict constraints (2f) can be exponentially large. They are separated via the client problem during the branch-and-cut solution process.

Note that the model contains no explicit constraints forcing the artificial variables y_a^t to attain only values 0 or 1. These constrains are not necessary. Any solution $(\mathbf{x}, \mathbf{y}, L)$ with $x_a^{st} \in \{0, 1\}$ for all $(s, t) \in K$ and $a \in A$ can be easily turned into an equivalent solution with $y_a^t \in \{0, 1\}$ for all $t \in V$ and $a \in A$ by setting $y_a^t := \max\{x_a^{st} : s \text{ with } (s, t) \in K\}$ for all t and a.

3.2 Client Problem

Now suppose we are given an integer solution $(\mathbf{x}, \mathbf{y}, L)$ of formulation (2) or, more precisely, of a subsystem of (2) containing only some of the conflict constraints (2f) so far.

Let F be the presumed unique shortest path forwarding given by this solution, i.e., $F = \{(t, a) : y_a^t = 1\}$. Our goal in the client problem is to find a compatible metric λ for F. However, if the given solution $(\mathbf{x}, \mathbf{y}, L)$ violates some of the conflict constraints (2f) that have not yet been added to the master formulation, such a metric does not exist. In this case, the task is to generate one of these violated inequalities.

The first part of this problem can be solved with linear programming techniques. A number of alternative formulations for this so-called *inverse shortest paths problem (ISP)* have been proposed in the literature [2,29]. In the following, we present the aggregated formulation used in our algorithm together with the arc-routing formulation for the master problem.

Let F be the given presumed unique shortest path forwarding. For each pair $(t, a) \in F$, arc $a = (u, v)$ is assumed to be on a *unique* shortest path from u to t. Hence, the arcs $a' \in \delta^+(u) \setminus \{a\}$ must not be on any shortest (u, t)-path. The set of all these implied non shortest path node-arc pairs is $\bar{F} = \bigcup_{(t,(u,v)) \in F} (\delta^+(u) \setminus \{(u, v)\})$. For each pair $(t, a) \in \bar{F}$, arc $a = (u, v)$ must *not* be on a shortest path from u to t. Note that we cannot simply assume $\bar{F} = V \times A \setminus F$, because F does not necessarily prescribe shortest paths between all node pairs. Arcs that are not relevant for the routing of the given commodities may or may not be on shortest paths and, thus, the corresponding node-arc pairs may be missing in both F and $\bar{F}$.

Our formulation of the inverse shortest paths problem uses a variable $\lambda_a \in \mathbb{Z}$ for the length of each arc $a \in A$ and a variable $\pi_v^t \in \mathbb{R}$ for the so-called potential of each node $v \in V$ with respect to each destination $t \in V$ and the metric λ. (If $\pi_t^t = 0$, the smallest possible potential π_v^t of node v is exactly the distance from v to t with respect to the arc lengths λ_a.) With these variables, the inverse shortest paths problem for the given forwarding F, can be formulated as follows:

$$
\begin{aligned}
&\min \quad \lambda_{\max} && && \text{(3a)}\\
\text{s.t.} \quad &\lambda_{(u,v)} - \pi_u^t + \pi_v^t = 0 && (t,(u,v)) \in F && \text{(3b)}\\
&\lambda_{(u,v)} - \pi_u^t + \pi_v^t \geq 1 && (t,(u,v)) \in \bar{F} && \text{(3c)}\\
&\lambda_{(u,v)} - \pi_u^t + \pi_v^t \geq 0 && (t,(u,v)) \in (V \times A) \setminus \{F \cup \bar{F}\} && \text{(3d)}\\
&1 \leq \lambda_a \leq \lambda_{\max} && a \in A && \text{(3e)}\\
&\pi_v^t \in \mathbb{R} && t \in V,\, v \in V && \text{(3f)}\\
&\lambda_a \in \mathbb{Z} && a \in A\ . && \text{(3g)}
\end{aligned}
$$

Constraints (3b),(3d), and (3e) ensure that the lengths λ_a in any solution of (3) form a compatible metric for the given forwarding F. The term $\lambda_{(u,v)} - \pi_u^t + \pi_v^t$ is the difference between the length of the shortest path starting in node u, passing through arc (u,v), and ending in node t, and the distance from node v to node t. This difference must be 0 for all arcs (u,v) that are on a shortest path and strictly greater than 0 for all arcs that must not be on a shortest path, as expressed in constraints (3b) and (3c). For all remaining arcs it must be non-negative.

It is easy to verify that formulation (3) has a solution if and only if there exist a compatible metric for the given forwarding F. Furthermore, there is a compatible metric with lengths in the range $\{1, 2, \ldots, M\}$ if and only if the optimal solution value $\lambda_{\max}$ of formulation (3) is less or equal to M.

Note that formulation (3) is an integer program and may be computationally hard. In fact, Bley [4] proved that it is already $\mathcal{NP}$-hard to approximate its optimum value within a factor less than 9/8 in general. In our application, however, we only want to find *some* feasible solution of (3), or prove that none exists. We do not really care about the size of the length values λ_a. Using Cramer's rule, one easily verifies that the integer program (3) has a solution if and only if its linear relaxation has: Multiplying a solution of the linear relaxation with the determinant of the corresponding linear programming basis, for example, always yields a feasible integer solution of (3). As the linear relaxation of (3) can be solved in polynomial time, we also can decide in polynomial time whether the given set F is an USPF or not.

In our algorithm, we solve the linear relaxation of (3) in a first step and, if it is feasible, scale and round its optimal fractional solution to an integer feasible solution of (3) afterwards. Using the rounding scheme proposed by Ben-Ameur and Gourdin [2], we obtain lengths that exceed the minimal possible ones by a factor of at most $\min(|V|/2, |P_{\max}|)$, where $P_{\max}$ is the longest prescribed shortest path. For practically relevant network sizes, the lengths computed with this approximate method easily fit into the admissible range of all modern routing protocols. So, we can safely ignore the integrality constraint (3g) in practice.

If the linear relaxation of (3) is infeasible, then the given solution $(\mathbf{x}, \mathbf{y}, L)$ of the (incomplete) master formulation is not a valid routing. In this case, the presumed forwarding F is not a valid unsplittable shortest path forwarding. It

contains at least one (irreducible) conflict $C \in \mathcal{C}$, whose corresponding inequality (2f) is violated by the given solution $(\mathbf{x}, \mathbf{y}, L)$. To find one of these conflicts, we iteratively try to remove each node-arc pair from F. In each iteration, we remove one pair (t, a) from F, update the set $\bar{F}$ of implied non-shortest path node-arc pairs, and solve the corresponding linear relaxation of (3). If this linear program remains infeasible, we remove the pair (t, a) permanently from F. Otherwise, we reinsert it into F and keep it permanently. If no more node-arc pair can be removed, the remaining set F defines an irreducible conflict, whose corresponding conflict inequality (2f) for $C = F$ is violated by the given solution $(\mathbf{x}, \mathbf{y}, L)$. In our implementation, we improved the practical performance of this procedure significantly by removing initially all those pairs $(t, (u, v))$ from F, for which the dual variables of the corresponding constraint (3b) and the dual variables of all constraints (3c) implied by $(t, (u, v)) \in F$ are 0. If these constraints are not active in the infeasible subsystem of (3), there is at least one (irreducible) conflict that is not related to the fact that $(t, (u, v)) \in F$.

Note that this iterative method finds an irreducible conflict inequality (2f), but not necessarily the most violated one. Finding the most violated such inequality is $\mathcal{NP}$-hard, even if the given solution of the master problem is integer [7]. Furthermore, note that this approach solves the separation problem over the conflict inequalities (2f) only for integer solutions $(\mathbf{x}, \mathbf{y}, L)$. For fractional solutions $(\mathbf{x}, \mathbf{y}, L)$, the presumed forwarding F is not well-defined, so we cannot construct and solve the client problem's linear program (3). In our application we try to overcome this difficulty by constructing the presumed forwarding as $F := \{(t, a) : y_a^t \geq 0.8\}$. This way, we can apply the simple and fast separation algorithm described above also as a separation heuristic for fractional solutions $(\mathbf{x}, \mathbf{y}, L)$ of the master problem. Yet, the overall decomposition algorithm is an exact algorithm for the Min-Con-USPR problem, because our iterative method solves the separation problem for integer solutions $(\mathbf{x}, \mathbf{y}, L)$ of the master problem exactly.

An alternative approach to solve the separation problem for the conflict inequalities (in a variant appropriate for shortest multi-path routing) was proposed by Tomaszewski et al. [32,33]. Tomaszewski et al. formulate the separation problem over these inequalities as an integer linear program, which they then solve heuristically. For fractional solutions of the master problem, Tomaszewski's approach might find more violated conflict inequalities than our iterative approach, as the latter one does not take the effect of the fractional variables correctly into account. Broström and Holmberg [14,15] studied a special subclass of conflict constraints (again in a variant appropriate for shortest multi-path routing), called valid cycle constraints, for which they derived a polynomial time separation algorithm that also is applicable for fractional solutions of the master problem. From a computational perspective, however, both the general separation algorithm proposed by Tomaszewski et al. and the polynomial time separation algorithm for the special valid cycle constraints proposed by Broström and Holmberg are much more demanding that our simple iterative approach. As the conflict inequalities are only necessary to ensure feasibility of the final solution

but have no substantial effect on the value of the linear programming relaxation of the master problem, it is not clear, whether it is worth to spend more computation time on a better separation for these inequalities. The computational results reported in [32] and [5] indicate that this is not the case.

4 Valid Inequalities

The integrality gap of the integer programming formulation (2) for the master problem can be very large. Even for the solution of small problem instances, it is necessary to improve the linear relaxation of (2) with additional strong inequalities. In this section, we describe the problem specific inequalities that are used in our implementation of this algorithm.

4.1 Routing Inequalities

Numerous types of valid inequalities can be derived from the so-called Bellman or subpath consistency condition [2,5]. Two paths P_1 and P_2 are said to satisfy the *subpath consistency condition* if there are no nodes u and v, such that P_1 and P_2 both contain a subpath $P_1[u,v]$ and $P_2[u,v]$, respectively, and $P_1 \neq P_2$. A path set Q is said to satisfy the subpath consistency condition, if all pairs of paths in Q satisfy this condition.

Clearly, any path set that comprises a unique shortest path routing satisfies the subpath consistency condition. If two paths P_1 and P_2 would contain different subpath $P_1[u,v]$ and $P_2[u,v]$, then none of the two paths P_1 and P_2 can be a unique shortest path between its terminals. The subpath consistency is one of the simplest conditions that must be satisfied by the paths of an unsplittable shortest path routing. All USPF conflicts that may occur between two paths are in fact violations of the subpath consistency; see [5].

The out-degree inequalities (2e) in the formulation of the master problem are a special class of inequalities implied by the subpath consistency condition: As there must be a unique (sub-)path from v to t for all nodes v and t, the routing paths towards t must form an anti-arborescence with root t.

Analogous to the out-degree inequalities, we also add in-degree inequalities to the master problem's formulation. These ensure that the routing paths emanating from each source s form an arborescence. For notational simplicity, assume that we have artificial variables $z_a^s := \max\{x_a^{st} : t \text{ with } (s,t) \in K\}$ for all $s \in V$ and $a \in A$. With these artificial variables, the in-degree inequalities can be easily formulated as

$$\sum_{a \in \delta^-(v)} z_a^s \leq 1 \qquad\qquad v, s \in V \ . \tag{4}$$

These inequalities are clearly valid for any integer solution of (2e). As for the out-degree inequalities, the separation problem over these inequalities can be solved in a straightforward way with an enumerative algorithm.

It is not difficult to derive many types of valid inequalities from the subpath consistency condition. In our algorithm, we add the following three types of inequalities to the master problem formulation (2):

$$x_a^{s,v} - x_a^{s,t} + \sum_{e \in \delta^-(v)} x_e^{s,t} \leq 1 \quad (s,t),(s,v) \in K,\ a \in A\ , \tag{5}$$

$$x_a^{v,t} - x_a^{s,t} + \sum_{e \in \delta^-(v)} x_e^{s,t} \leq 1 \quad (s,t),(v,t) \in K,\ a \in A\ , \tag{6}$$

$$x_a^{s,v} + x_a^{v,t} - x_a^{s,t} - 2(1 - \sum_{e \in \delta^-(v)} x_e^{s,t}) \leq 0 \quad (s,v),(v,t),(s,t) \in K,\ a \in A\ . \tag{7}$$

One easily verifies that these inequalities are valid for each integer solution of (2). Consider the inequalities of type (5) for the two commodities $(s,t),(s,v) \in K$. If $\mathbf{x}$ corresponds to a valid unsplittable shortest paths routing, the term $\sum_{e \in \delta^-(v)} x_e^{s,t}$ will be 1 if the routing path for commodity (s,t) passes through node v and 0 otherwise. In the first case, the corresponding inequalities (5) for (s,t) and (s,v) reduce to $x_a^{s,v} \leq x_a^{s,t}$ for all $a \in A$, which expresses the condition that that the routing path of commodity (s,v) must be subpath of the routing path of commodity (s,t). In the second case, these inequalities reduce to $x_a^{s,v} \leq x_a^{s,t} + 1$ for all $a \in A$, which is satisfied trivially.

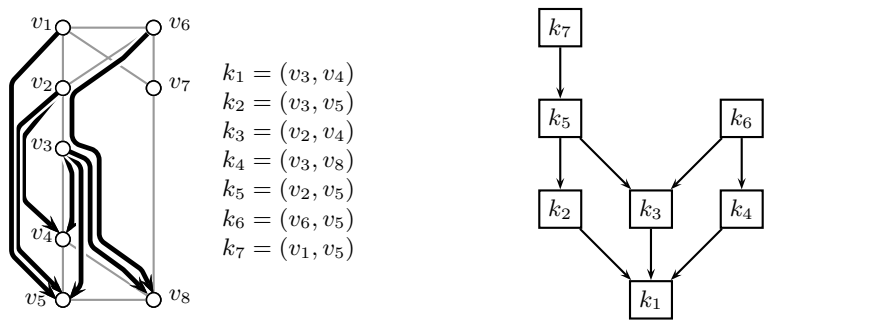
Analogously, inequalities (6) express the condition that the routing path for commodity (v,t) must be a subpath of the routing path for commodity (s,t), if v is contained in the latter path. Inequalities (7) finally ensure that the routing paths of the two commodities (s,v) and (v,t) are disjoint subpaths of the routing path for commodity (s,t), if the latter path contains v.

Although in general none of these inequalities is facet-defining for the polytope associated with (2), they all proved to be very useful in practice. In our implementation, we solve the separation problem over these inequalities with a straightforward enumerative algorithm.

4.2 Precedence Constrained Knapsack Inequalities

The inequalities discussed above have been derived by considering only the routing paths of valid unsplittable shortest paths routings. The traffic demands and link capacities have been completely ignored so far. The inequalities discussed in the following combine the combinatorial restrictions on the valid routing path and the given link capacities and demand volumes.

For each arc a, the subproblem defined by its capacity constraint and the arc routing variables on this arc can be regarded as a precedence constrained knapsack. The items in this knapsack problem correspond to the commodities $(s,t) \in K$, the weights of the items are the demand volumes $d_{(s,t)}$ of the commodities. The capacity of the knapsack is $L_{\max} c_a$, where $L_{\max}$ is some upper bound for the optimal congestion value. The subpath consistency condition in the original shortest path routing problem implies additional precedence relations among the commodities. Consider two commodities $(s,t),(u,v) \in K$ and



(a) Possible routing of commodities across arc (v_3, v_4).

(b) Precedence relations among the commodities.

Figure 1. Precedence relations in the routing optimization instances.

suppose that every (s,t)-path across arc a contains the two nodes u and v in its subpaths from s to a and from a to t, respectively. Then commodity (u,v) must be routed across arc a whenever commodity (s,t) is routed across arc a, which leads to the so-called precedence (or order) relation $x_a^{u,v} \geq x_a^{s,t}$. These precedence relations define a partial order on the commodities. For each commodity $(s,t) \in K$, we denote by $p(s,t)$ the set of commodities (u,v) which are immediate predecessors of commodity (s,t) in this order. Figure 1 illustrates this partial order for a small example.

Together with the capacity constraint (2c), these precedence relations define the precedence constrained knapsack polytope

$$\begin{aligned} PCK(a, L_{\max}) := \operatorname{conv}\{\, x_a^{s,t} \in \{0,1\}^K : & \\ & \sum_{(s,t)\in K} d_{st} x_a^{st} \leq c_a\, L_{\max} \\ & x_a^{u,v} \geq x_a^{s,t} \quad (s,t) \in K,\ (u,v) \in p(s,t)\ \} \end{aligned}$$

associated with arc a and $L_{\max}$. Valid inequalities for the associated precedence constrained knapsack polyhedra carry over in a straightforward way to valid inequalities for the polyhedron associated with (2): An inequality that is valid for $PCK(a, L_{\max})$ is valid for each solution $(\mathbf{x}, \mathbf{y}, L)$ of (2) that satisfies $L \leq L_{\max}$. Hence, all inequalities that are valid for $PCK(a, L_{\max})$ are also valid for the optimal solution $(\mathbf{x}^*, \mathbf{y}^*, L^*)$ of (2), provided that $L_{\max}$ is a valid upper bound on the optimal congestion value L^*. Of course, these inequalities are valid only for the optimal solution of (2), not for all solutions. As not only the capacity constraints but also the precedences among the routing variables are respected, the inequalities derived from facets of the precedence constrained knapsack polyhedron are typically much stronger than those derived from facets of the knapsack polyhedron defined by the capacity constraint alone.

The polyhedral structure of the precedence constrained knapsack polytope was first investigated by Boyd [13], who generalized cover and $(1,k)$-configuration

inequalities for the standard knapsack polytope to the precedence constrained case. In our algorithm, we apply only the so-called induced cover inequalities introduced by Boyd [13]. Let $a \in A$ and $L_{\max}$ be an upper bound for the optimal congestion value derived from the best known feasible solution, for example. For each commodity $(s,t) \in K$, we denote by $P(s,t)$ the set of commodities (u,v) for which the routing variable $x_a^{u,v}$ is a (not necessarily immediate) predecessor of the routing variable $x_a^{s,t}$ in the precedence order, i.e., $P(s,t)$ contains *all* predecessors of $x_a^{s,t}$. We also assume that $(s,t) \in P(s,t)$. For a set of commodities $C \subseteq K$, let $P(C) := \bigcup_{(s,t)\in C} P((s,t))$ and $d(P(C)) := \sum_{(s,t)\in P(C)} d_{st}$.

According to Boyd we call a commodity set $C \subseteq K$ an *induced cover* for arc a and congestion $L_{\max}$, if $d(P(C)) > c_a\, L_{\max}$, i.e., the total demand of the commodities in and implied by C exceeds the capacity available on arc a for solution with congestion $L_{\max}$ or less. For any induced cover $C \subseteq K$, the *induced cover inequality* [13]

$$\sum_{(s,t)\in C} x_a^{st} \leq |C| - 1 \tag{8}$$

is valid for $PCK(a, L_{\max})$ and, consequently, for (2). If C is an induced cover, not all commodities in C can be routed simultaneously across arc a in an optimal solution of (2).

Boyd [13], Park and Park [27], and van de Leensel et al. [24] also derived conditions under which these inequalities are facet-defining for lower dimensional faces of the precedence constrained knapsack polytope and proposed sequential lifting procedures to lift these inequalities into facets of the precedence constrained knapsack polytope. These lifting techniques, however, are computationally too demanding to be applied effectively in a branch-and-bound algorithm. Both the separation problem for the basic induced cover inequalities and the problem of determining the right coefficients in the lifting process are $\mathcal{NP}$-hard in general.

In our algorithm, we apply a simple greedy heuristic to find violated induced cover inequalities for each arc a and the currently best known upper bound $L_{\max}$. In the first phase, we try to find an induced cover whose induced cover inequality is violated. For this, we build a commodity set S that is closed under the precedence order. Starting with some minimal commodity in the precedence order, we iteratively add a commodity whose predecessors are all already contained in S. If the total demand of the commodities in the set S exceeds the available capacity $c_a\, L_{\max}$, we consider the set C of maximal commodities w.r.t. the precedence order in S. This commodity set C is an induced cover for a and $L_{\max}$. If the corresponding induced cover inequality is violated, we keep this inequality and go to the third phase of the heuristic. Otherwise we continue with the second phase, where we keep on adding commodities to the set S, but from now on in such a way that the size of the set C of maximal commodities decreases. For this, we prefer in each iteration the addition of commodities that are predecessors of several currently maximal commodities. Every time the size of C reduces, the corresponding induced cover inequality is evaluated. The second phase stops if

no violated induced cover inequality can be generated by adding further items. If some violated induced cover inequality was found, then, in the third phase, we finally remove as many commodities as possible from the minimal induced cover C such that C remains an induced cover. Further strengthening techniques, such as lifting further commodities into the inequality, are computational expensive and not performed.

In each cut generation step of the master problem's branch-and-cut algorithm, this greedy heuristic is applied three times with three different strategies of adding the commodities in the first and in the second phase for each arc a and the currently best known congestion value $L_{\max}$. When applied for the very first time, we also have to compute the precedence order among the flow variables across arc a. For this, we check for each pair of commodities (s,t) and (u,v) if u is contained in every path from s to the source node of a and if v is contained in every path from the target node of a to t. This is done by computing the connected components in the digraphs obtained by removing u and v, respectively.

Another simple and very effective type of constraints can be derived for commodities whose terminals are directly connected. For any $(s,t) \in K$ with $(s,t) \in A$, the subpath property implies that either commodity (s,t) is routed across arc (s,t), or none of the commodities in K is routed across arc (s,t). Furthermore, it is always possible to turn a feasible routing where no routing path contains arc (s,t) into another feasible routing by replacing the (s,t)-subpath of all commodities that are routed via s and t by the arc (s,t) and leaving the rest of the routing unchanged. Thus, if the capacity of arc (s,t) is at least as large as the capacities of any other arc that may be contained in an (s,t)-path, the solution not containing arc (s,t) cannot be optimal. Consequently, the equalities

$$x^{s,t}_{(s,t)} = 1 \qquad (s,t) \in K \cap A \text{ with } c_{(s,t)} \geq c_a \text{ for all } a \in A \setminus \{(s,t)\} \tag{9}$$

must hold for some optimal solution of (2). There are at most $|A|$ equalities of this type, so they can be generated in polynomial time. Adding these equalities to the formulation of the master problem proved to be very useful in practice. Note, however, that these equalities are not valid for all feasible solutions of (2); they only hold for some optimal solutions.

5 Primal Heuristics

It is extremely important for the performance of our algorithm to quickly construct good feasible solutions. The upper bound on the optimal solution value derived from heuristic solutions is not only used to prune non-optimal branches in the branch-and-bound tree based on the objective value, it is also used to generate strong induced cover inequalities that cut-off non-optimal solutions. Getting a tight upper bound for the minimal congestion in the beginning of the branch-and-bound algorithm thus helps to strengthen the formulation and speed up the search in the sequel.

In our implementation, we use four heuristics. Two heuristics are used to compute initial solutions from scratch. The first initial heuristic simply generates a set of random routing metrics, biased on the geographic link lengths and the link capacities. For each of these candidate metrics, we then apply a perturbation to ensure the uniqueness of all shortest paths. For the given integer-valued metric λ, we consider the perturbed metric λ' defined as $\lambda'_a := 2^{|A|}\lambda_a + 2^{idx(a)}$ for all $a \in A$, where idx is an arbitrary bijection $idx : A \leftrightarrow \{0, \ldots, |A|\}$. One easily verifies that all shortest paths with respect to λ' are unique and that any shortest paths with respect to λ' is also a shortest path with respect to λ. For the perturbed metrics, we finally compute the induced routing paths and traffic flows.

Our second initial heuristic is the Lagrangian solution approach presented in [3]. In this approach, we solve the linear relaxation of the corresponding minimum congestion unsplittable flow problem via a Lagrangian relaxation, which relaxes the capacity constraints (2c) and, thereby, decomposes the problem into $|K|$ shortest path problems. The dual variables of the capacity constraints can be interpreted naturally as a routing metric. The Lagrangian algorithm iteratively updates this candidate metric based on the violation of the (relaxed) capacity constraints. Our initial 'heuristic' simply takes the metric defined by the dual variables after each iteration of the Lagrangian algorithm, perturbs these metrics to ensure unique shortest paths, and computes the shortest paths and traffic flows induced by the perturbed metrics. As a by-product, the Lagrangian algorithm also computes a lower bound on the optimal solution value. A more detailed description of this solution approach can be found in [3].

Two different heuristics are used within the branch-and-bound tree. Analogous to the initial heuristics, both generate candidate routing metrics, perturb them if necessary, and then compute a feasible solution from these metrics. The first heuristic tries to construct solutions by combining the current dual variables of the capacity constraints and of the induced cover constraints, which can be interpreted as arc lengths quite naturally again, with several random metrics.

The second heuristic is closely related to and combined with the separation routine for USPF-inequalities. In a first step, we determine the forwarding F that is given by the arc routing variables that are integer or almost integer. In our implementation, we let $F := \{(t, a) : y^t_a \geq 0.8\}$. Then we try to find a compatible metric for this forwarding by solving the linear programming relaxation of (3). If this linear program has a feasible solution, we use its optimal solution as a candidate routing metric and then perturb and evaluate this metric as in the other heuristics. If the linear program has no feasible solution, the near-integer routing variables form a non-USPF. In this case, the heuristic proceeds like the separation algorithm for the USPF-inequalities.

To avoid spending too much time in these heuristics, both are applied only at branch-and-bound nodes whose depth is a power of two, or if the forwarding defined by the almost-integer routing variables at the current branch-and-bound node differs from the one at the parent node by more than two node-arc pairs. Thus, the heuristics are applied more frequently at the top of the branch-and-

bound tree than in the lower parts of the tree, and they are applied after very significant changes.

6 Implementation

The presented algorithm has been implemented as part of the network optimization library DISCNET [1]. The data structures and algorithms are based on the standard c++ library and LEDA [23], the linear programs arising in the solution process are solved with CPLEX 12.1 [22].

In the following, we describe our implementation for the problem version where one seeks for a symmetric unsplittable shortest path routing. In this case, the routing lengths of anti-parallel arcs must be equal and, consequently, the routing path from s to t is the reverse path of the routing path from t to s for all $s, t \in V$. Corresponding models for the master and client problems are obtained by either adding the equalities

$$x^{s,t}_{(u,v)} = x^{t,s}_{(v,u)} \qquad (s,t),(t,s) \in K,\ (u,v),(v,u) \in A$$

to the master model (2) and the equalities

$$\lambda_{(u,v)} = \lambda_{(v,u)} \qquad (u,v),(v,u) \in A$$

to the client model (3). In our implementation, we perform the corresponding variable substitutions in (2) and (3).

The initial formulation of the master problem contains all arc-routing variables x^{st}_a, all tree variables y^t_a, and the congestion variable L. Furthermore, it contains the flow conservation equalities (2b), the capacity inequalities (2c), and out-degree inequalities (2e). All other model constraints and additional inequalities are generated via separation subroutines during the execution of the algorithm, together with the standard integer linear programming cuts generated automatically by CPLEX.

The indegree inequalities (4) are omitted in the case of symmetric unsplittable shortest path routing, because they are equivalent to the outdegree inequalities (2e). Violated linking constraints (2d) and subpath consistency inequalities (5), (6), and (7) are separated at all nodes of the branch-and-bound tree for at most 5 rounds. The respective separation problems are solved exactly using ad-hoc enumeration methods. Although there is only a polynomial number of these inequalities, we add them via a cutting plane approach only if they are violated absolutely by more than a given threshold (by default 0.01 for the linking constraints (2d) and 0.1 for the subpath consistency constraints). Adding all of these inequalities to the initial formulation would increase the size of the master problem too much and is practically prohibitive.

The general conflict constraints (2f) are separated via solving the client problem as described in Section 3.2. Whenever an integer solution candidate for the (incomplete) master formulation is found, we solve the client problem to decide

whether or not it defines a valid unsplittable shortest path routing and to find a compatible metric or a violated conflict inequality (2f).

In our implementation, we solve the client problem not only for the fully integer solutions at the leaves of the master problem's branch-and-bound tree, but also for non-integer solutions arising within the branch-and-bound tree. At each node of the master problem's branch-and-bound tree, we consider the potential forwarding $F \subseteq V \times A$ defined by the integer and near integer routing variables. In our implementation, we let $F := \{(t, a) : y_a^t \geq 0.8\}$. We solve the client problem whenever this presumed forwarding differs from the one at the parent node in the branch-and-bound tree by more than two node-arc pairs, if the depth of the current node in the branch-and-bound is a power of 2 or if all arc-routing variables are integer. If the linear relaxation of the client problem (3) is feasible for this forwarding F, the computed link lengths define a heuristic solution for the Min-Con-USPR problem, which may improve on the best known solution. Otherwise, if the linear relaxation of the client problem (3) is infeasible, we generate a conflict inequality (2f) using the greedy approach discussed in Section 3.2. This inequality is violated at least by the rounded master problem's solution used to construct the potential forwarding F. If it is also violated by the original fractional solution of the master problem, we add it to the master formulation. Solving the client problem also for rounded fractional solutions of the master problem at inner nodes of the branch-and-bound tree drastically reduced the running time of the overall algorithm in practice, because good solutions and inequalities excluding invalid routing patterns are found early in the branch-and-bound tree.

The branch-and-bound tree for the master problem is constructed and explored in such a way, that we first perform the branching decisions that have the biggest impact on the lower bound. These are the decisions that fix the routing of the commodities with big demands on those arcs, where the maximum congestion is attained. Given the fractional linear programming solution at the current node of the branch-and-bound tree, we first determine the set of all arcs, for which the capacity constraint or one of the induced cover inequalities is tight. Then we determine the maximum demand $d_{\max}$ of those commodities that are routed fractionally across any of these arcs and, after that, restrict our attention to only those fractional arc routing variables whose associated demand value is at least $0.9\, d_{\max}$. Among these variables, we finally choose the one whose fractional value is closest to the target value of 0.8.

The next branch-and-bound node to explore is selected by the following strategy. For a given number of iterations (32 by default), we choose the node with the best lower bound. Then, we 'dive' for a good feasible solution deep into the tree. The dive starts at an unexplored node with minimal lower bound and at each step chooses the child node whose arc routing variable was fixed to 1 in the previous branching decision. If a new feasible solution is found, we switch back to the best lower bound strategy. If we reach a node that is infeasible or too costly, we backtrack. If no feasible solution is found after 5 backtracking steps, we switch back to the best bound strategy.

7 Results

Table 1 shows the computational results obtained by running the presented algorithm for a collection of benchmark problems taken from the Survivable Network Design Library SNDlib [26] and six real-world traffic engineering problems stemming from the German national research and education network between 1999 and 2005. All computations were performed on a machine with an Intel Core2 Duo E8400 CPU at 3.00 GHz and 2 GB RAM running Linux 2.6. the linear programs arising in the solution process are solved with Cplex 12.1 [22]. The algorithm was run with a total CPU time limit of 5 hours on each instance.

The underlying networks are bidirectional and have the same capacity for both directions of all links. The given traffic demands are asymmetric, i.e., the commodities (s, t) and (t, s) may have are different demand values. The task is to find a symmetric unsplittable shortest path routing. In the six real-world problems, there are additional upper bounds on the lengths of the admissible routing paths for each commodity, which correspond to restrictions on the transmission delay in the real networks. These constraints are incorporated into the master model (2) in a straightforward way via linear constraints. In the SNDlib instances and the two real-world instances Bwin and Gwin4, the goal is to minimize the maximum link utilization over all links, as in the definition of the Min-Con-USPR problem. The reported objective values are 100 times the maximum congestion in these instances. In the four instances Gwin1, Gwin2, Gwin3, and Xwin, we consider a generalized version of congestion minimization: In these instances, some arcs of the digraph model interconnection links between the German national research network and other networks. As these links are known a priory to be highly loaded, they form a separate group of links. The goal is to minimize a linear combination of the maximum congestion attained on these special link and the maximum congestion attained on the normal links. More details on these data sets and the extended model can be found in [5].

The numbers of nodes, bidirected links and non-zero traffic demands are shown in the first columns of Table 1. The columns DE, PCK, SC, USPF, and VUB show how many inequalities of types (9), (8), (5)–(7), (2f), and (2d) have been generated in total during the execution of the algorithm. The best proven lower bound and the best solution value found by our algorithm within 5 hours are reported in columns LB and UB. The remaining columns show the residual optimality gap, the number of explored branch-and-bound nodes, and the total CPU time until either optimality was proven or the time limit was exceeded.

The results show that our algorithm can be used to solve traffic engineering problems of realistic size. All real-world instances and most of the small and medium size benchmark instances have been solved optimally within seconds or minutes. For the large problem instances that could not be solved to optimality, our algorithm always found better solutions than the length-based heuristic and Lagrangian approaches that have been used to compute initial feasible solutions. Our algorithm also clearly outperforms all other integer programming approaches presented in the literature so far, which typically even fail to achieve gaps below 30% for networks larger than 10 nodes.

Table 1. Computational results with all additional inequalities enabled.

Problem	Size			Number of added cuts					LB	UB	Gap	B&B	Time
	$\|V\|$	$\|A\|$	$\|K\|$	De	Pck	Sc	Uspf	Vub			(%)	Nodes	(h:mm:ss)
Atlanta	15	22	210	10	46	831	15	853	861	861	0.0	12	0:00:02.7
Dfn-bwin	10	45	90	3	28	0	0	106	699	699	0.0	1	0:00:00.1
Dfn-gwin	11	21	110	9	28	96	13	347	510	510	0.0	54	0:00:01.4
Di-yuan	11	42	22	3	0	0	1	71	625	625	0.0	1	0:00:00.1
France	25	45	300	33	89	4874	13	5070	747	747	0.0	219	0:08:37.8
Germany50	50	88	662	58	219	12366	194	15221	648	655	1.2	754	5:01:30.0
NewYork	16	49	240	39	56	1214	12	2882	620	620	0.0	39	0:01:06.1
Nobel-Eu	28	41	378	26	82	9724	17	9571	444	446	0.3	178	5:00:08.1
Nobel-Ger	17	26	121	16	25	2391	11	1849	733	733	0.0	20	0:00:12.5
Nobel-Us	14	21	91	21	44	1521	125	2116	494	494	0.0	1930	0:02:56.7
Norway	27	51	702	51	126	12179	1	13731	546	616	12.7	1007	5:00:12.1
PDH	11	34	24	14	23	3	0	115	800	800	0.0	1	0:00:00.1
Polska	12	18	66	18	21	760	9	1191	938	938	0.0	373	0:00:33.6
TA1	24	55	396	7	58	161	6	636	933	994	0.0	1	0:00:01.6
Bwin	10	12	90	2	0	14	0	130	53	53	0.0	6	0:00:01.2
Gwin1	11	19	110	2	0	26	5	150	132	132	0.0	5	0:00:00.2
Gwin2	11	27	110	16	0	51	1	254	43	43	0.0	18	0:00:00.6
Gwin3	11	23	109	11	0	61	8	226	36	36	0.0	20	0:00:00.6
Gwin4	11	23	104	7	0	19	2	160	11	11	0.0	1	0:00:00.1
Xwin	43	58	222	17	0	1680	64	6253	52	52	0.0	1749	0:18:49.0

In order to evaluate the effectiveness of the additional inequalities discussed in Section 4, we also run tests in which we selectively enabled and disabled the separation subroutines of the different inequality types. The effect of adding the different inequalities on the lower bound and on the best solution obtained after exploring 20 branch-and-bound nodes is shown in Table 2. The column group labeled 'Without cuts' report results for the runs where none of the inequalities (5)–(9) has been added. The column group 'With SC cuts' contains the results of the runs where only the inequalities (5)–(7) have been added, the columns labeled 'With PCK cuts' report the results of the runs where only inequalities (8) and (9) have been added, and the group 'With all cuts' corresponds to the runs where all inequalities (5)–(9) have been enabled. For each group, the columns LB, UB, and Gap show the best lower bound, the value of the best solution, and the remaining optimality gap after exploring 20 branch-and-bound nodes, respectively. Inf indicates cases where no solution was found and the resulting gap is infinity.

It is evident from the results in Table 2 that both groups of inequalities are extremely helpful in practice. Adding only the inequalities (5)–(7) derived from the subpath consistency gave substantially improved lower bounds in 2 and improved solutions in 5 of the 20 cases, while adding only (in)equalities (8) and (9) substantially improves the lower bounds in 5 cases and the solutions in 5 cases. Of course, adding these inequalities to the model has side effects on the exploration of the branch-and-bound tree and on the addition of other inte-

Table 2. Effect of additional inequalities after 20 branch-and-bound nodes.

Problem	Without cuts			With Sc cuts			With Pck cuts			With all cuts		
	LB	UB	Gap	LB	UB	Gap	LB	UB	Gap	LB	UB	Gap
Atlanta	776	992	27.7	803	868	8.1	861	861	0.0	861	861	0.0
Dfn-bwin	678	699	3.1	383	699	82.4	699	699	0.0	699	699	0.0
Dfn-gwin	508	510	0.3	508	510	0.3	508	510	0.3	508	510	0.3
Di-yuan	625	625	0.0	625	625	0.0	625	625	0.0	625	625	0.0
France	602	Inf	Inf	602	Inf	Inf	711	Inf	Inf	711	776	9.2
Germany50	648	755	16.6	648	680	5.0	648	800	23.6	648	730	12.7
NewYork	445	620	39.2	445	620	39.2	463	620	34.1	472	620	31.3
Nobel-Eu	444	483	8.8	444	479	7.8	444	496	11.6	444	483	8.8
Nobel-Ger	644	733	13.8	644	800	24.1	644	733	13.8	733	733	0.0
Nobel-Us	484	516	6.6	484	494	2.1	484	506	4.5	484	494	2.1
Norway	546	724	32.5	546	760	39.1	546	736	34.7	546	728	33.2
PDH	548	800	46.0	800	800	0.0	800	800	0.0	800	800	0.0
Polska	829	Inf	Inf	829	953	14.9	829	953	15.0	829	950	14.6
TA1	933	933	0.0	841	947	12.5	933	933	0.0	933	933	0.0
Bwin	53	53	0.0	53	53	0.0	53	53	0.0	53	53	0.0
Gwin1	132	132	0.0	132	132	0.0	132	132	0.0	132	132	0.0
Gwin2	43	43	0.6	42	43	2.7	43	43	0.0	43	43	0.0
Gwin3	36	36	1.3	36	36	1.3	36	36	0.9	36	36	0.0
Gwin4	11	11	0.0	11	11	0.0	11	11	0.0	11	11	0.0
Xwin	50	54	8.2	50	54	8.2	54	54	8.2	50	54	8.2

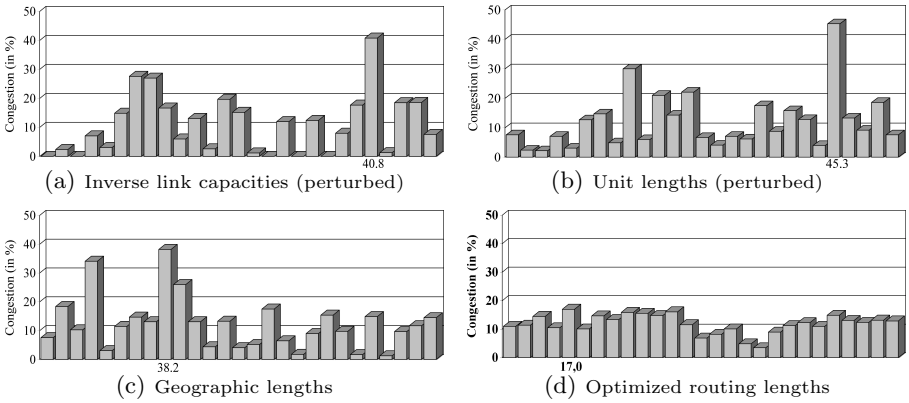
ger programming cuts via Cplex's build-in separation subroutines, so there are also some cases where adding inequalities (5)–(7) slightly deteriorates the lower bounds and solutions. In the runs where all types of inequalities are added, the lower bound improves in 6 cases and better solutions are found in 7 cases, leading to substantially smaller residual integrality gaps in 11 of the 20 cases. Furthermore, there is only one case with a slightly worse solution and no case with a worse lower bound than in the corresponding run where none of the addition inequalities is added. Comparing the results of the different runs, one also finds that the vast majority of the improvement in the lower bound is due to adding (in)equalities (8) and (9), which are derived from the dependencies between the the link capacities and the subpath consistency property. Inequalities (5)–(7), on the other hand, help to find better feasible solutions earlier in the search tree, but they have only little effect on the lower bound of the linear relaxation. Note, however, that the outdegree and linking constraints (2d) and (2e) already enforce that the routing paths form in- and out-arborescences, which covers a large and practically important subclass of the subpath consistency conditions. Adding inequalities (5)–(7) thus only enforces those subpath consistency conditions that involve two or three commodities and do not fall into the class of in- or outdegree constraints.

Table 3 provides a more detailed report on the effect of the different inequalities at the root node of the branch-and-bound tree and on their impact on the overall branch-and-bound procedure. For each of the four strategies, the first

columns in group 'Root' show how many inequalities of the corresponding types have been added at the root node of the branch-and-bound tree. The column LP-Gap reports the integrality gap of the linear relaxation after all CPLEX preprocessing steps and 5 rounds of adding cuts. This gap is the relative error with respect to the optimal integer solution value or, if marked by an asterix, to the best lower bound obtained for this instance by our algorithm after 5 hours. The second column group 'Branch&Bound' describes the impact of the additional inequalities on the overall branch-and-bound algorithm. These columns show the residual optimality gap, the number of explored branch-and-bound nodes, and the total CPU time until either optimality was proven or the time limit was exceeded.

The results in Table 3 again show that both groups of inequalities are necessary to solve large problem instances. In many cases, adding (in)equalities (8) and (9) alone, already yields a much better linear programming bound at the root node and leads a substantial improvement of the performance of the overall branch-and-bound algorithm. In several cases, however, our approach achieves its maximum performance only if all inequalities (5)–(9) are added. For the instances Germany50 and Norway, for example, the best know solutions are found only if all of these inequalities are used. For the instances Atanta, Dfn-gwin, France, and NewYork, better solution were found more quickly and, thus, optimality could by proved faster by using all inequalities.

The computational results also show that instances with dense networks and lots of potential routing paths for most demand pairs are more difficult than those where the underlying networks are fairly sparse. One of the reasons for this behavior is that the induced cover inequalities (8) are much stronger for sparse networks than for dense networks. In dense networks, the values of the flow variables x_a^{st} may become extremely small on all arcs in an optimal LP solution, which makes these inequalities almost useless in these cases. Furthermore, lots of violated subpath consistency constraints (5)–(7) and linking constraints (2d) are found at all depths of the branch-and-bound tree for instances with dense networks and many commodities. For the benchmark instance Germany50, for example, 12,366 subpath consistency constraints and 15,221 linking constraints have been added during the exploration of the 754 branch-and-bound nodes, even though only inequalities with an absolute violation of more than 0.1 have been generated. Adding all these inequalities to the formulation drastically slows down the solution of the linear relaxations and reduces the number of branch-and-bound nodes that can be explored, which can be seen in the number of explored branch-and-bound nodes reported for this instance and for the different strategies in Table 3. For the larger and more difficult instances Germany50, Nobel-EU, and Norway, more than 90% of the overall solution time was spent in (re-)solving the linear programs in the branch-and-bound tree. Also in the other problem instances, lots of violated subpath consistency constraints are found, while only a moderate number of induced cover inequalities and very few general conflict inequalities are generated. However, not adding the subpath



(a) Inverse link capacities (perturbed) (b) Unit lengths (perturbed)

(c) Geographic lengths (d) Optimized routing lengths

Figure 2. Link congestion values in G-WiN for several routing metrics.

consistency inequalities is not practically either, as it would lead to numerous invalid branches in the branch-and-bound tree.

Figure 2 finally illustrates the importance of optimizing the routing weights in practice. It shows the different link loads that would result with unsplittable shortest path routing from three commonly used default weight settings and those resulting from the optimized routing weights in the German national research and education network G-WiN with capacities and traffic demands of August 2001. Even without using the traffic splitting possibilities of the ECMP routing version, the traffic is distributed much more evenly for the optimized metric. The peak congestion is not even half of that for the default settings, which significantly reduces packet delays and loss rates and improves the network's robustness against unforeseen traffic changes and failures.

Acknowledgments

This work was motivated by the problems arising in the planning of the German national research and education network operated by the DFN Verein. I wish to thank everyone at DFN and in particular Marcus Pattloch for fruitful discussions and an excellent cooperation.

References

1. atesio GmbH, Sophie-Taeuber-Arp-Weg 27, D-12205 Berlin, Germany: discnet – Network optimization software library (2000–2005). URL `http://www.atesio.de`
2. Ben-Ameur, W., Gourdin, E.: Internet routing and related topology issues. SIAM Journal on Discrete Mathematics **17**, 18–49 (2003)
3. Bley, A.: A Lagrangian approach for integrated network design and routing in IP networks. In: Proceedings of the 1st International Network Optimization Conference (INOC 2003), Paris, France, pp. 107–113 (2003)

4. Bley, A.: Inapproximability results for the inverse shortest paths problem with integer lengths and unique shortest paths. Networks **50**, 29–36 (2007)
5. Bley, A.: Routing and capacity optimization for IP networks. PhD thesis, Technische Universität Berlin (2007)
6. Bley, A.: Approximability of unsplittable shortest path routing problems. Networks **54**(1), 23–46 (2009)
7. Bley, A.: On the hardness of finding small shortest path routing conflicts. In: Proceedings of the 4th International Network Optimization Conference (INOC 2009), Pisa, Italy (2009)
8. Bley, A., Fortz, B., Gourdin, E., Holmberg, K., Klopfenstein, O., Pióro, M., Tomaszewski, A., Ümit, H.: Optimization of OSPF routing in IP networks. In: A. Koster, X. Muñoz (eds.) Graphs and Algorithms in Communication Networks: Studies in Broadband, Optical, Wireless and Ad Hoc Networks, chap. 8, pp. 199–240. Springer (2009)
9. Bley, A., Grötschel, M., Wessäly, R.: Design of broadband virtual private networks: Model and heuristics for the B-WiN. In: N. Dean, D. Hsu, R. Ravi (eds.) Robust Communication Networks: Interconnection and Survivability, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, vol. 53, pp. 1–16. American Mathematical Society (1998)
10. Bley, A., Koch, T.: Integer programming approaches to access and backbone IP-network planning. In: Modeling, Simulation and Optimization of Complex Processes: Proceedings of the 3rd International Conference on High Performance Scientific Computing, pp. 87–110. Hanoi, Vietnam (2006)
11. Bley, A., Pattloch, M.: Modellierung und Optimierung der X-WiN Plattform. DFN-Mitteilungen **67**, 4–7 (2005)
12. Bourquia, N., Ben-Ameur, W., Gourdin, E., Tolla, P.: Optimal shortest path routing for Internet networks. In: Proceedings of the 1st International Network Optimization Conference (INOC 2003), Paris, France, pp. 119–125 (2003)
13. Boyd, E.: Polyhedral results for the precedence-constrained knapsack problem. Discrete Applied Mathematics **41**, 185–2001 (1993)
14. Broström, P., Holmberg, K.: Determining the non-existence of compatible OSPF weights. In: D. Yuan (ed.) Nordic MPS 2004, no. 14 in Linköping Electronic Conference Proceedings, pp. 7–21. Linköping University Electronic Press (2004)
15. Broström, P., Holmberg, K.: Valid cycles: A source of infeasibility in OSPF routing. Networks **52**, 206–215 (2008)
16. Buriol, L., Resende, M., Ribeiro, C., Thorup, M.: A hybrid genetic algorithm for the weight setting problem in OSPF/IS-IS routing. Networks **46**, 36–56 (2005)
17. Callon, R.: Use of OSI IS-IS for routing in TCP/IP and dual environments. IETF Internet RFC 1195 (1990)
18. Ericsson, M., Resende, M., Pardalos, P.: A genetic algorithm for the weight setting problem in OSPF routing. Journal of Combinatorial Optimization **6**, 299–333 (2002)
19. Fortz, B., Thorup, M.: Increasing Internet capacity using local search. Computational Optimization and Applications **29**, 13–48 (2004)
20. de Giovanni, L., Fortz, B., Labbé, M.: A lower bound for the Internet protocol network design problem. In: Proceedings of the 2nd International Network Optimization Conference (INOC 2005), Lisbon, Portugal, pp. 402–408 (2005)
21. Holmberg, K., Yuan, D.: Optimization of Internet protocol network design and routing. Networks **43**, 39–53 (2004)
22. ILOG CPLEX 12.1 (2009). URL `http://ilog.com/products/cplex/`

23. LEDA: Library of Efficient Data types and Algorithms (1998–2003). URL `http://www.algorithmic-solutions.com`
24. van de Leensel, R., van Hoesel, C., van de Klundert, J.: Lifting valid inequalities for the precedence constrained knapsack problem. Mathematical Programming **86**, 161–186 (1999)
25. Moy, J.: OSPF version 2. IETF Internet RFC 2328 (1998)
26. Orlowski, S., Pióro, M., Tomaszewski, A., Wessäly, R.: SNDlib 1.0–Survivable Network Design Library. Networks (2009). DOI 10.1002/net.20371. To appear. Preprint version available as ZIB report ZR-07-15
27. Park, K., Park, S.: Lifting cover inequalities for the precedence-constrained knapsack problem. Discrete Applied Mathematics **72**, 219–241 (1997)
28. Parmar, A., Ahmed, S., Sokol, J.: An integer programming approach to the OSPF weight setting problem. Optimization Online (2006)
29. Pioro, M., Medhi, D.: Routing, flow, and capacity design in communication and computer networks. Morgan Kaufmann (2004)
30. Pióro, M., Szentesi, A., Harmatos, J., Jüttner, A.: On OSPF related network optimization problems. In: 8th IFIP Workshop on Performance Modelling and Evaluation of ATM & IP Networks, pp. 70/1–70/14. Ilkley, UK (2000)
31. Prytz, M.: On optimization in design of telecommunications networks with multicast and unicast traffic. Ph.D. thesis, Royal Institute of Technology, Stockholm, Sweden (2002)
32. Tomaszewski, A., Pióro, M., Dzida, M., Mycek, M., Zagożdżon, M.: Valid inequalities for a shortest-path routing optimization problem. In: Proceedings of the 3rd International Network Optimization Conference (INOC 2007), Spa, Belgium (2007)
33. Tomaszewski, A., Pióro, M., Dzida, M., Zagożdżon, M.: Optimization of administrative weights in IP networks using the branch-and-cut approach. In: Proceedings of the 2nd International Network Optimization Conference (INOC 2005), Lisbon, Portugal, pp. 393–400 (2005)
34. Ümit, H., Fortz, B.: Fast heuristic techniques for intra-domain routing metric optimization. In: Proceedings of the 3rd International Network Optimization Conference (INOC 2007), Spa, Belgium (2007)

Table 3. Effect of additional inequalities on the root node and on the overall branch-and-bound algorithm.

Problem	Without Sc and Pck cuts				With Sc cuts					With Pck cuts						With all cuts						
	Root	Branch&Bound			Root		B&B			Root			B&B			Root				B&B		
	LP Gap (%)	Fin. Gap (%)	B&B Nodes	Time (h:m:s)	Sc cuts	LP Gap (%)	Fin. Gap (%)	B&B Nodes	Time (h:m:s)	De cuts	Pck cuts	LP Gap (%)	Fin. Gap (%)	B&B Nodes	Time (h:m:s)	De cuts	Pck cuts	Sc cuts	LP Gap (%)	Fin. Gap (%)	B&B Nodes	Time (h:m:s)
Atlanta	23.6	0.0	44	0:00:06	118	23.6	0.0	17	0:00:02	4	34	5.1	0.0	13	0:00:02	6	40	566	5.1	0.0	12	0:00:03
Dfn-bwin	0.3	0.0	24	0:00:02	50	0.3	0.0	0	0:00:00	3	28	0.3	0.0	0	0:00:00	3	28	0	0.3	0.0	0	0:00:00
Dfn-gwin	53.3	0.0	142	0:00:04	0	53.3	0.0	82	0:00:01	0	16	0.0	0.0	96	0:00:03	0	16	0	0.0	0.0	54	0:00:01
Di-yuan	19.4	0.0	23	0:00:02	46	19.4	0.0	0	0:00:00	3	0	4.8	0.0	0	0:00:00	3	0	0	4.8	0.0	0	0:00:00
France	0.0	0.0	806	2:15:59	914	0.0	0.0	342	0:12:44	6	48	0.0	0.0	257	0:10:32	7	46	1592	0.0	0.0	219	0:08:38
Germany50	*28.2	16.6	965	5:00:13	2546	*28.2	5.0	190	6:03:17	1	76	*28.2	23.6	278	5:24:30	0	25	2686	*28.2	1.2	754	5:01:31
NewYork	48.6	34.6	982	5:00:08	136	48.1	0.0	79	0:01:19	12	60	0.0	0.0	54	0:01:58	10	56	180	0.0	0.0	39	0:01:06
Nobel-Eu	*11.6	0.3	1768	5:00:13	2174	*11.6	3.1	72	5:00:08	0	19	*11.6	0.3	2211	5:00:18	0	29	2216	*11.6	0.3	178	5:00:08
Nobel-Ger	66.8	0.0	72	0:00:24	567	62.2	0.0	17	0:00:11	0	15	0.0	0.0	25	0:00:10	0	20	576	0.0	0.0	20	0:00:13
Nobel-Us	3.3	0.0	739	0:01:51	15	3.3	0.0	963	0:01:17	0	24	3.3	0.0	620	0:00:54	0	25	15	3.3	0.0	1930	0:02:57
Norway	*6.1	32.5	1029	5:00:07	2221	*3.8	39.1	488	5:00:10	1	90	*7.5	13.1	1111	5:01:04	4	90	2350	*3.8	12.7	1007	5:00:43
PDH	7.0	0.0	31	0:00:01	21	7.7	0.0	0	0:00:00	14	23	4.1	0.0	0	0:00:00	14	23	3	4.1	0.0	0	0:00:00
Polska	13.8	0.0	567	0:00:48	134	14.5	0.0	363	0:00:29	2	18	10.0	0.0	304	0:00:21	2	21	113	8.2	0.0	373	0:00:34
TA1	0.0	0.0	10	0:00:10	902	0.0	0.0	0	0:00:01	1	0	0.0	0.0	0	0:00:02	7	58	149	0.0	0.0	0	0:00:02
Bwin	48.6	0.0	8	0:00:01	1	48.6	0.0	6	0:00:01	1	0	0.0	0.0	6	0:00:01	1	0	2	0.0	0.0	6	0:00:01
Gwin1	0.0	0.0	5	0:00:00	11	0.0	0.0	5	0:00:00	3	0	0.0	0.0	4	0:00:00	1	0	16	0.0	0.0	5	0:00:00
Gwin2	12.1	0.0	26	0:00:01	78	12.1	0.0	18	0:00:00	12	0	12.1	0.0	19	0:00:01	12	0	31	12.1	0.0	18	0:00:01
Gwin3	2.0	0.0	21	0:00:01	39	2.0	0.0	20	0:00:00	8	0	2.0	0.0	21	0:00:01	9	0	49	2.0	0.0	20	0:00:01
Gwin4	0.0	0.0	0	0:00:00	18	0.0	0.0	0	0:00:00	7	0	0.0	0.0	6	0:00:00	7	0	19	0.0	0.0	0	0:00:00
Xwin	4.2	0.0	4600	0:37:37	40	4.2	0.0	1749	0:16:02	0	0	4.2	0.0	1484	0:11:55	0	0	40	4.2	0.0	1749	0:18:49