Solving resource constrained shortest path problems with LP-based methods

Markó Horváth[a], Tamás Kis[a,*]

[a]*Institute for Computer Science and Control, Hungarian Academy of Sciences, H1111 Budapest, Kende str. 13–17, Hungary*

Abstract

In the resource constrained shortest path problem (RCSPP) there is a directed graph along with a source node and a destination node, and each arc has a cost and a vector of weights specifying its requirements from a set of resource types with finite capacities. A minimum cost source-destination directed path is sought such that the total consumption of the arcs from each resource type does not exceed the capacity of the resource. In this paper we investigate LP-based branch-and-bound methods and introduce new cutting planes, separation procedures, variable fixing, and primal heuristic methods for solving RCSPP to optimality. We provide detailed computational experiments, and a comparison to other methods in the literature.

Keywords: Resource constrained shortest path, Integer programming, Branch-and-cut, Primal heuristics, Combinatorial optimization

1. Introduction

The resource constrained shortest path problem (RCSPP) is an extension of the familiar shortest path problem in directed graphs. Informally, we have a network with directed arcs and a specific source and a specific destination node. Each arc has a cost, and specifies its requirements from a set of resources with finite capacities. A minimum cost directed path from the source to the destination node is sought such that the total resource consumption of the arcs from each resource type does not exceed its capacity.

We will pursue a branch-and-cut approach, where the linear programming relaxation of the integer programming formulation is strengthened by inequalities valid for the convex hull of feasible (integer) solutions, but which cut off optimal fractional (infeasible) solutions.

Our main results can be summarized as follows:

*Corresponding author, Tel.: +36 1 2796156

Email addresses: marko.horvath@sztaki.mta.hu (Markó Horváth), tamas.kis@sztaki.mta.hu (Tamás Kis)

- New cutting planes. We generalize and extend cutting planes from [8]. In particular we generalize the $s-t$ cut precedence inequalities of Garcia [8] and provide a polynomial time separation procedure for the more general class. Furthermore, we extend the subpath precedence inequalities of Garcia [8]. We prove that it is NP-hard to separate the subpath precedence inequalities, which answers an open question in [8]. We also show that our extension yields a class of cuts which are also NP-hard to separate, and provide a heuristic separation procedure.

- New primal heuristic. We describe a procedure for obtaining a feasible solution in each branch-and-bound node using the data (network) of the corresponding subproblem.

- New variable fixing methods. Our variable fixing techniques for reducing the size of the subproblem are based on analyzing the network of the subproblems corresponding to the branch-and-bound nodes.

- New computational results. We have conducted a thorough computational evaluation and compared our new classes of cuts to their precursors by Garcia [8]. We emphasize that in these tests, all problem instances have 10 resource types. The tests suggest that neither the new, nor the old cuts dominate one-another, and we could characterize the favorable instances for each class. Moreover, the variable fixing and primal heuristic methods have proved very effective in solving all the instances. We have also compared our method to state-of-the-art approaches from the literature. Our main finding is that most of the widely used test instances can be solved in a split of a second (without branch-and-cut) by applying the preprocessing procedure of Dumitrescu and Boland [6] provided that an efficient implementation is at hand, and for the rest of the instances, the computation times can be significantly reduced by applying the variable fixing, and the primal heuristic proposed in this paper.

Structure of the paper. Firstly, we give a formal problem statement along with the notation used throughout the paper (Section 2). We review the related literature in Section 3. We continue with the definition of classes of those valid inequalities of Garcia [8] that will be extended in this paper (Section 4). The new valid inequalities for RCSPP along with the corresponding separation procedures are presented in Section 5. New variable fixing method and a primal heuristic are described in Section 6. Detailed computational evaluation is provided in Section 7, while comparisons with state-of-the-art methods are presented in Section 8. We conclude the paper in Section 9.

2. Formal problem statement

First we define the Shortest Path Problem (SPP) in directed graphs, then we extend it with resources to obtain the Resource Constrained Shortest Path Problem (RCSPP). An instance of the Shortest Path Problem consists of a

directed graph $G = (V, E)$, a cost function $c : E \to \mathbb{Z}$ (negative valuers are allowed), and two distinct nodes from V, the source node s, and the destination node t. Let $\mathcal{P}_{st}$ denote the set of all directed $s-t$ paths in G (a sequence of adjacent directed arcs which do not visit the same node twice). The SPP problem aims at finding the cheapest (least costly) path from s to t in G, that is,

$$\min_{P \in \mathcal{P}_{st}} c(P), \tag{1}$$

where $c(P) := \sum_{e \in P} c(e)$ is the cost of the path.

Now, suppose that in addition to the above problem data, we also have a function $w : E \to \mathbb{Z}^m$ which assigns an m-dimensional weight-vector to each arc, which we call *resource requirements*. There is also an m-dimensional vector $W \in \mathbb{Z}^m$, which represents the maximum available quantities from the resources. In the *Resource Constrained Shortest Path Problem* a directed path P from s to t is sought which has the smallest total cost, and which respects the resource constraints

$$\sum_{e \in P} w_e^r \leq W^r, \quad r = 1, \ldots, m. \tag{2}$$

Clearly, a shortest $s-t$ path may not respect the resource constraints. It is usually assumed that all the resource requirements are non-negative, and there can be lower bounds L as well on the resource consumptions, see e.g., [3]. In the most general case, the w_e^r may take negative values as well, see [11].

Assumption 1. *Throughout this paper we assume that G contains no directed cycle of negative total cost, or of negative total resource consumption for any of the resource types.*

This is a standard assumption adapted in many papers.

We can formulate the problem by a mathematical program in which there is a binary decision variable x_e on each arc $e \in E$ indicating whether the path sought goes through the arc or not.

$$\min \quad \sum_{e \in E} c_e x_e \tag{3}$$

$$\text{s.t.} \quad \sum_{e \in \delta^{out}(i)} x_e - \sum_{e \in \delta^{in}(i)} x_e = \begin{cases} 1 & i = s \\ 0 & i \in V \setminus \{s, t\} \\ -1 & i = t \end{cases}, \quad i \in V \tag{4}$$

$$\sum_{e \in E} w_e^r x_e \leq W^r, \quad r = 1, \ldots, m \tag{5}$$

$$x \in \{0, 1\}^E \tag{6}$$

In the above formulation, $\delta^{in}(i)$ and $\delta^{out}(i)$ denote the set of arcs entering and leaving node i, respectively. The objective function (3) expresses the total cost of the path sought. Constraints (4) along with (6) ensure that the feasible solutions are paths. The resource consumptions of the paths are bounded by (5).

Since G does not admit any negative cost, or negative weight cycle by assumption, if (3)-(6) admits a feasible solution, it has an optimal solution which corresponds to a $s-t$ path in G.

The *RCSPP polytope* is the set of binary vectors satisfying (4)-(6), noting that if G contains directed cycles, then those cycles have corresponding vertices in the polytope. However, it will not create any difficulties for us under Assumption 1.

Notation. For a directed graph $G=(V,E)$, if $e \in E$ is a directed arc from node i to node j, then the *head* of e is node j, and will be denoted by $head(e)$, and the *tail* of e is node i, and will be denoted by $tail(e)$. The set of arcs entering node v is denoted by

$$\delta^{in}(v) := \{e \in E \mid head(e) = v\},$$

whereas those leaving v constitute the set

$$\delta^{out}(v) := \{e \in E \mid tail(e) = v\}.$$

We say that node j is *reachable* from node i if G contains a directed path from node i to node j. The set of nodes reachable from node $i \in V$ on directed paths is denoted by $\rho^{out}(i)$, and symmetrically, let $\rho^{in}(i)$ be the set of nodes from which node i can be reached in G. We assume that $i \in \rho^{out}(i)$ and $i \in \rho^{in}(i)$. For a subset $S \subseteq V$ of nodes, let $\gamma(S)$ be the *set of all arcs spanned by* S, i.e.,

$$\gamma(S) := \{e \in E \mid head(e), tail(e) \in S\}.$$

For a path $\pi = (\{v_0, \ldots, v_p\}, \{e_0, \ldots, e_{p-1}\})$ we denote the subpath consisting of the first i arcs of π by $\pi[0,i]$. The extension of π with an arc e_p with $tail(e_p) = v_p$ is denoted by $\pi \oplus e_p$.

For a subset of arcs $A \subseteq E$ and any weighting $y : E \to \mathbb{R}$, $y(A) := \sum_{e \in A} y(e)$ is the sum of weights of those arcs in A. For a pair of nodes i, j, let σ^r_{ij} denote the cost of the shortest $i-j$ path in G with respect to arc weights w^r_e, whereas σ^c_{ij} denotes that with respect to the arc costs c_e. If no $i-j$ path exists, σ^r_{ij} and σ^c_{ij} are ∞. For arcs $e_1, e_2 \in E$ we say that the arc-pair (e_1, e_2) is *compatible* if

$$\sigma^r_{s,tail(e_1)} + w^r_{e_1} + \sigma^r_{head(e_1),tail(e_2)} + w^r_{e_2} + \sigma^r_{head(e_2),t} \leq W^r$$

holds for all resources r; otherwise we say the arc-pair is *incompatible*.

3. Literature review

The elementary RCSPP has been shown to be NP-hard in the strong sense for graphs containing negative cost cycles by Dror [5]. However, the problem remains NP-hard even if the graph is acyclic, there is a single resource type, and all the arc weights and costs are non-negative (cf. problem [ND30] in Garey and

Johnson [9]). In fact, the latter problem can be solved in pseudo-polynomial time, see e.g., Joksch [13].

Several types of methods have been proposed to solve variants of RCSPP, for an overview, see e.g., Garcia [8], Irnich and Desaulniers [11], Pugliese and Guerriero [17]. For instance, in *path ranking* methods, the first K shortest $s-t$ paths are generated from which the shortest resource feasible is chosen, if it exists. Clever ways of applying path ranking for solving RCSPP have been suggested in e.g., [10, 16]. *Node labeling* type methods are based on dynamic programming, where the nodes of the graph are labeled with the cost of the directed (elementary) paths leading to them from the source node s, and also with the possible resource consumptions on these paths. The first method in this class is from Joksch [13]. If all arc weights are positive, the method of Desrochers and Soumis [4] finds the optimal solution in $O(|E|U)$ time in case of a single resource with upper bound U. An improved algorithm is proposed by Dumitrescu and Boland [6]. The third type of methods uses *Lagrangian relaxation* to relax the resource constraints. For instance, Beasley and Christofides [3] propose branch-and-bound in which lower bounds are computed using a Lagrange dual, and they also apply various preprocessing and variable fixing methods to reduce the number of branch-and-bound nodes. The methods in the fourth category are based on linear programming relaxation and cutting planes. For instance, Avella et al. [2] use cutting planes to solve RCSPP with negative cycles. In Jepsen et al. [12], a model with resource weights on the nodes is considered, and the authors adapt several classes of valid inequalities for the capacitated vehicle routing problem to their problem and also proposed a new class. Their computational experiments show that branch-and-cut is more effective than dynamic programming on large instances with 800 nodes.

Garcia [8] propose several types of valid inequalities for the polytope of feasible solutions of RCSPP, some of which being valid for RCSPP with cycles, while others only for the acyclic special case. A detailed computational evaluation shows the merits of the various classes in solving the two variants of the problem by branch-and-cut. In the proposed algorithms preprocessing plays a very important role as well.

Methods in the fifth class: these are methods which do not completely fit in any of the above categories. Lozano and Medaglia [15] propose the Pulse method, which traverses the graph with depth-first-search, and use bounding and pruning strategies to limit the search-space. The method has proved very effective in practice in solving RCSPP with a single resource only, and the authors propose some very limited results with 10 resource instances. Pugliese and Guerriero [18] devised the Reference Point Method, the main idea being that they consider RCSPP as a multi-criteria optimization problem to direct the search toward a resource feasible solution of smallest cost. The authors state that no pre-processing algorithms are invoked before running their own method. The authors provide extensive computational results on single-resource instances and at a glance, their method is as effective as that of Lozano and Medaglia. Unfortunately, there is no detailed comparison between the two approaches.

In this paper we will pursue a polyhedral approach, and use a branch-and-cut

method to solve the RCSPP.

4. Preliminaries

In this section we recapitulate previous results that we will extend in Sections 5 and 6. The inequalities presented in this section are valid for the RCSPP polytope.

4.1. Node precedence inequalities

The class of these inequalities is based on the idea that a resource-feasible path through some arc e can leave node $head(e)$ only on those arcs $e' \in \delta^{out}(head(e))$ that satisfy the condition

$$\sigma^r_{s,tail(e)} + w^r_e + w^r_{e'} + \sigma^r_{head(e'),t} \leq W^r$$

for each resource r, as shown by Garcia [8]. Let

$$\phi^{out}_{e,r} := \{e' \in \delta^{out}(head(e)) \mid \sigma^r_{s,tail(e)} + w^r_e + w^r_{e'} + \sigma^r_{head(e'),t} \leq W^r\}.$$

Since we aim at finding elementary paths, we can safely drop from $\phi^{out}_{e,r}$ those arcs e' with $head(e') = tail(e)$ (if any), to obtain the set

$$F^{out}_{e,r} := \{e' \in \phi^{out}_{e,r} \mid head(e') \neq tail(e)\}.$$

Then the *node precedence inequality* for arc e with respect to resource r is

$$x_e \leq x(F^{out}_{e,r}). \tag{7}$$

The validity of this inequality for the RCSPP polytope is easily seen from the definitions. One can also define an analogous inequality using the sets

$$\phi^{in}_{e,r} := \{e' \in \delta^{in}(tail(e)) \mid \sigma^r_{s,tail(e')} + w^r_{e'} + w^r_e + \sigma^r_{head(e),t} \leq W^r\},$$

and

$$F^{in}_{e,r} := \{e' \in \phi^{in}_{e,r} \mid tail(e') \neq head(e)\}.$$

The resulting *reverse node precedence inequality* for arc e with respect to resource r is

$$x_e \leq x(F^{in}_{e,r}). \tag{8}$$

Garcia [8] has also provided a polynomial time exact separation algorithm for node precedence inequalities.

4.2. $s-t$ cut precedence inequalities

This class of inequalities generalizes the node precedence inequalities of the previous section. Let $S \subset V$ be a set of nodes with $s \in S$ and $t \notin S$, and $e \in E$ with $head(e) \in S \setminus \{s\}$. Then any resource feasible $s-t$ path π through arc e must cross the $s-t$ cut $\delta^{out}(S)$ on some arc in $\gamma(\rho^{out}(head(e)))$ (the set of arcs $e' \in E$ such that $tail(e')$ is reachable from node $head(e)$ on a directed path in G). In addition, π must pass through some arc $e' \in \delta^{out}(S) \cap \gamma(\rho^{out}(head(e)))$ with

$$\sigma^r_{s,tail(e)} + w^r_e + \sigma^r_{head(e),tail(e')} + w^r_{e'} + \sigma^r_{head(e'),t} \leq W^r, \qquad r = 1, \ldots, m.$$

Again, Garcia [8] defined the sets

$$\Phi^{out}_{e,r} := \{e' \in E \mid \sigma^r_{s,tail(e)} + w^r_e + \sigma^r_{head(e),tail(e')} + w^r_{e'} + \sigma^r_{head(e'),t} \leq W^r\}$$

for each resource r. For any $S \subset V$ with $s \in S$, $t \notin S$, and $e \in E$ with $head(e) \in S \setminus \{s\}$, let

$$F^{out}_{e,r}(S) := \{e' \in \delta^{out}(S) \mid e' \in \Phi^{out}_{e,r}, tail(e') \neq tail(e), head(e') \neq tail(e)\}.$$

Then the *$s-t$ cut precedence inequality* for resource r is

$$x_e \leq x(F^{out}_{e,r}(S)). \tag{9}$$

Clearly, the node precedence inequalities of the previous section are just special cases of $s-t$ cut precedence inequalities. This class of inequalities can be separated by computing a minimum $head(e)-t$ cut in a graph derived from G. For further details, see [8].

For $e \in E$ one can define analogously the set

$$\Phi^{in}_{e,r} := \{e' \in E \mid \sigma^r_{s,tail(e')} + w^r_{e'} + \sigma^r_{head(e'),tail(e)} + w^r_e + \sigma^r_{head(e),t} \leq W^r\}$$

for each resource r, and for any $S \subset V$ with $s \in S$, $t \notin S$, and $tail(e) \in \overline{S} \setminus \{t\}$,

$$F^{in}_{e,r}(S) := \{e' \in \delta^{out}(S) \mid e' \in \Phi^{in}_{e,r}, tail(e') \neq head(e), head(e') \neq head(e)\}.$$

Then the *reverse $s-t$ cut precedence inequality* for resource r is

$$x_e \leq x(F^{in}_{e,r}(S)). \tag{10}$$

4.3. Subpath precedence inequalities

Let $\pi = \left(\{v_1, \ldots, v_p\}, \bigcup_{j=1,\ldots,p-1}\{e_j\}\right)$ be a path from node v_1 to node v_p in G such that $v_1 \neq t$, $v_p \neq s$, $v_2, \ldots, v_{p-1} \notin \{s,t\}$, and e_j being a directed arc of G from node v_j to v_{j+1}. We say that π *is an infeasible subpath with respect to resource r* if

$$\sigma^r_{s,v_1} + \sum_{k=1}^{p-1} w^r_{e_k} + \sigma^r_{v_p,t} > W^r.$$

If e_1 is an arc of a feasible $s-t$ path π^*, then π^* cannot contain all the arcs of π, and therefore, it must leave the subpath π on some arc adjacent to one of the nodes $v_2, \ldots, v_{p-1}$. Suppose π^* leaves π on the arc e' directed from node v_k of π $(2 \leq k \leq p-1)$ to some node $v' \neq v_{k+1}$. Since π^* is feasible for resource r, the condition

$$\sigma^r_{s,v_1} + \sum_{\ell=1}^{k-1} w^r_{e_\ell} + w^r_{e'} + \sigma^r_{head(e'),t} \leq W^r, \qquad r = 1, \ldots, m,$$

is satisfied. Using this observation, Garcia [8] has defined the sets

$$\phi^{out}_{\pi,r}(k) := \{e' \in \delta^{out}(i_k) \mid \sigma^r_{s,i_1} + \sum_{\ell=1}^{k-1} w^r_{e_\ell} + w^r_{e'} + \sigma^r_{head(e'),t} \leq W^r\},$$

for each resource r and $k = 2, \ldots, p-1$, and

$$F^{out}_{\pi,r}(k) := \{e' \in \phi^{out}_{\pi,r}(k) \mid head(e') \neq i_{k+1}, i_1, \ldots, i_{k-1}\}, \qquad k = 2, \ldots, p-1.$$

Letting $F^{out}_{\pi,r} = \bigcup_{k=2}^{p-1} F^{out}_{\pi,r}(k)$, the *subpath precedence inequality with respect to the infeasible subpath* π is

$$x_{e_1} \leq x(F^{out}_{\pi,r}). \tag{11}$$

It is clear again, that a subpath precedence inequality for an infeasible path with length 2 is nothing but a node precedence inequality of the Section 4.1.

One can define analogously the sets

$$\phi^{in}_{\pi,r}(k) := \{e' \in \delta^{in}(i_k) \mid \sigma^r_{s,tail(e')} + w^r_{e'} + \sum_{\ell=k}^{p-1} w^r_{e_\ell} + \sigma^r_{i_p t} \leq W^r\},$$

for each resourcr r and $k = 2, \ldots, p-1$, and

$$F^{in}_{\pi,r}(k) = \{e' \in \phi^{in}_{\pi,r}(k) \mid tail(e') \neq i_{k-1}, i_{k+1}, \ldots, i_p\} \qquad k = 2, \ldots, p-1.$$

Letting $F^{in}_{\pi,r} = \bigcup_{k=2}^{p-1} F^{in}_{\pi,r}(k)$, the *reverse subpath precedence inequality with respect to the infeasible subpath* π is

$$x_{e_p} \leq x(F^{in}_{\pi,r}). \tag{12}$$

Both of these classes of inequalities are valid for the RCSPP polytope. However, Garcia [8] neither provided a polynomial time separation procedure, nor a proof that the separation problem is intractable (NP-hard). Nevertheless, he gave a separation heuristic which worked well in practice.

4.4. Strengthening the inequalities

All the inequalities presented in this section so far have one of the following two forms:

$$x_e \leq x(F^{out}_r), \tag{13}$$

or

$$x_e \leq x(F_r^{in}), \tag{14}$$

where F_r^{out} is a set of arcs that lie on a directed $head(e) - t$ path, F_r^{in} is a set of arcs that lie on a directed $s - tail(e)$ path, and the inequalities are derived by considering only the resource weights w^r on the arcs. Garcia [8] argued that inequalities of these forms can be strengthened by the following trick. Consider e.g., the class (13). Let

$$B_{e,r} := \{e' \in \delta^{in}(head(e)) \mid \sigma^r_{s,tail(e')} + w^r_{e'} \geq \sigma^r_{s,tail(e)} + w^r_e\}.$$

Since $e \in B_{e,r}$, inequality (13) can be strengthened:

Proposition 1. *If (13) is valid for the RCSPP polytope, then so is*

$$x(B_{e,r}) \leq x(F_r^{out}).$$

A similar strengthening method applies to inequalities in the class (14). Define the set

$$A_{e,r} := \{e' \in \delta^{out}(tail(e)) \mid w^r_{e'} + \sigma^r_{head(e'),t} \geq w^r_e + \sigma^r_{head(e),t}\}.$$

Notice that $e \in A_{e,r}$.

Proposition 2. *If (14) is valid for the RCSPP polytope, then so is*

$$x(A_{e,r}) \leq x(F_r^{in}).$$

Using a simple observation in case of multiple resources, one can strengthen inequalities of the forms (13) and (14) in the following way:

Proposition 3. *If (13) is valid for the RCSPP polytope for all resource $r = 1, \ldots, m$, then so is*

$$x\left(\bigcap_{r=1}^{m} B_{e,r}\right) \leq x\left(\bigcap_{r=1}^{m} F_r^{out}\right)$$

Proposition 4. *If (14) is valid for the RCSPP polytope for all resource $r = 1, \ldots, m$, then so is*

$$x\left(\bigcap_{r=1}^{m} A_{e,r}\right) \leq x\left(\bigcap_{r=1}^{m} F_r^{in}\right)$$

4.5. Preprocessing and heuristics

Several preprocessing methods have been proposed for the RCSPP to reduce the size of the underlying graph G. Aneja et al. [1] deleted all nodes and arcs that cannot appear in a feasible $s-t$ path in G corresponding to a single resource. That is, they calculated the values σ^r_{si} and σ^r_{it}, i.e., the cost of the shortest $s-i$ path and cost of the shortest $i-t$ path in G, respectively, with arc weights w^r, for each resource r and for every node i. Then they erased all nodes i such that

$\sigma^r_{si} + \sigma^r_{it} > W^r$ holds for some resource r, since such a node cannot appear in a feasible $s-t$ path. Similarly, any arc e such that $\sigma^r_{s,tail(e)} + w^r_e + \sigma^r_{head(e),t} > W^r$ holds for some resource r can be eliminated from the graph. This procedure can be applied repeatedly until no other nodes and arcs can be deleted or no $s-t$ path remains in the reduced graph (which means that the problem is infeasible).

Beasley and Christofides [3] considered cost bounds to erase additional arcs and nodes from the underlying graph G. In a tree search procedure in each subproblem corresponding to a branch-and-bound node they calculated cost bounds through Lagrangean relaxation and eliminated arcs and nodes that could not appear in an optimal $s-t$ path in G. Dumitrescu and Boland [6] combined and simplified these approaches. They used the original arc costs instead of those derived from the Lagrangean dual to obtain upper bounds on the optimum value and created a combined preprocessing method. This preprocessing scheme has an additional advantage, namely, it may return an upper bound on the optimal solution value which can be used to improve the branch-and-cut procedure. For details we refer the reader to [6].

Garcia [8] also extended the preprocessing scheme of Aneja et al. Since the condition $\sigma^r_{si} + \sigma^r_{it} \leq W^r$ is necessary for each resource r, but not sufficient for the existence of an $s-t$ path through node i which is feasible for all resources, Garcia applied the preprocessing procedure for a subgraph of G which contains each $s-t$ path through node i. That is, one can create the graph $G[i] = (V[i], E[i])$ where $V[i] = \rho^{in}(i) \cup \rho^{out}(i)$ and $E[i] = \gamma(\rho^{in}(i)) \cup \gamma(\rho^{out}(i))$, and apply the preprocessing scheme of Aneja et al. for this graph repeatedly. If the procedure terminates because no more $s-t$ path is left in $G[i]$, then node i can be eliminated from the original graph G, since every $s-t$ path through i is infeasible for at least one resource constraint.

The latter approach can be applied not only to a node i, but also to an arc e. That is, we can create the graph $G[e] = (V[e], E[e])$ where $V[e] = \rho^{in}(tail(e)) \cup \rho^{out}(head(e))$ and $E[e] = \gamma(\rho^{in}(tail(e))) \cup \{e\} \cup \gamma(\rho^{out}(head(e)))$, and apply for it the preprocessing scheme of Aneja et al. repeatedly. Since preprocessing of $G[e]$ for all $e \in E$ can be expensive, Garcia [8] did not use this approach as a preprocessing procedure but created a variable fixing method which can be applied in a branch-and-cut procedure. That is, if a fractional solution x^* to the LP relaxation is available, one can preprocess $G[e]$ for all arc e such that $x^*_e > 0$. In this case the deletion of an arc e means to fix variable x_e to 0.

5. New valid inequalities and separation procedures

In this section we generalize the valid inequalities of Section 4.

5.1. Cut based inequalities

Fix a pair of edges e_1, e_2 of G with $e_1 \neq e_2$, and let $i_1 := tail(e_1)$, $j_1 := head(e_1)$ and $i_2 := tail(e_2)$, $j_2 := head(e_2)$. A necessary condition for a resource-feasible path π visiting e_1 and e_2 in this order to exist is that for each resource

r, the inequality

$$\sigma^r_{s,i_1} + w^r_{e_1} + \sigma^r_{j_1,i_2} + w^r_{e_2} + \sigma^r_{j_2,t} \leq W^r \tag{15}$$

holds. However, this condition is weak, and we can make it stronger. We define the set

$$\Phi_{(\cdot,e_1,e_2),r} = \{e \in E \mid \\ \sigma^r_{s,tail(e)} + w^r_e + \sigma^r_{head(e),i_1} + w^r_{e_1} + \sigma^r_{j_1,i_2} + w^r_{e_2} + \sigma^r_{j_2,t} \leq W^r\Big\}$$

and then for any $s - i_1$ cut S, the set

$$F_{(\cdot,e_1,e_2),r}(S) = \left\{e \in \delta^{out}(S) \cap \Phi_{(\cdot,e_1,e_2),r} \mid tail(e), head(e) \notin \{j_1, i_2, j_2, t\}\right\}.$$

The new inequality with respect to the set $F_{(\cdot,e_1,e_2),r}(S)$ is

$$x_{e_1} + x_{e_2} - 1 \leq x(F_{(\cdot,e_1,e_2),r}(S)). \tag{16}$$

Proposition 5. *If G contains no directed path from j_2 to i_1, then the inequality (16) is valid for the RCSPP polytope.*

Proof. Consider any resource-feasible elementary $s-t$ path P, and let x^P be its characteristic vector, i.e., $x^P_e = 1$ if P contains the arc e, otherwise 0. If P does not contain any of $\{e_1, e_2\}$, then the left-hand-side of (16) is at most 0, while the right-hand-side is at least zero by the non-negativity of x, and the claim follows. So, suppose P passes through both of e_1 and e_2, i.e., $x^P_{e_1} = x^P_{e_2} = 1$. Since G contains no directed path from j_2 to i_1, it follows that P passes through e_1 first, and then through e_2. We claim that the right-hand-side of (16) is at least 1. It suffices to verify that $x^P_e = 1$ for some arc $e \in F_{(\cdot,e_1,e_2),r}(S)$. Being a resource-feasible elementary $s-t$ path passing through e_1 and then e_2, P must start in s and visit i_1, therefore, all the arcs e' on the subpath P' from s to i_1 belong to $\Phi_{(\cdot,e_1,e_2),r}$ and P' contains no nodes from $\{j_1, i_2, j_2, t\}$. Since S is an $s - i_1$ cut, at least one edge e' of P' must belong to $F_{(\cdot,e_1,e_2),r}(S)$, and the claim is verified. □

One can define analogously the set

$$\Phi_{(e_1,\cdot,e_2),r} = \{e \in E \mid \\ \sigma^r_{s,i_1} + w^r_{e_1} + \sigma^r_{j_1,tail(e)} + w^r_e + \sigma^r_{head(e),i_2} + w^r_{e_2} + \sigma^r_{j_2,t} \leq W^r\Big\}$$

and then for any $j_1 - i_2$ cut S, the set

$$F_{(e_1,\cdot,e_2),r}(S) = \left\{e \in \delta^{out}(S) \cap \Phi_{(e_1,\cdot,e_2),r} \mid tail(e), head(e) \notin \{s, i_1, j_2, t\}\right\}$$

that gives rise to the new inequality

$$x_{e_1} + x_{e_2} - 1 \leq x(F_{(e_1,\cdot,e_2),r}(S)). \tag{17}$$

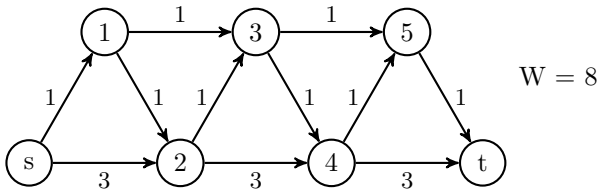

Figure 1: Network of Example 1.

And one can also define analogously the set

$$\Phi_{(e_1,e_2,\cdot),r} = \{e \in E \mid \\ \sigma^r_{s,i_1} + w^r_{e_1} + \sigma^r_{j_1,i_2} + w^r_{e_2} + \sigma^r_{j_2,tail(e)} + w^r_e + \sigma^r_{head(e),t} \leq W^r\Big\}$$

and then for any $j_2 - t$ cut S, the set

$$F_{(e_1,e_2,\cdot),r}(S) = \left\{e \in \delta^{out}(S) \cap \Phi_{(e_1,e_2,\cdot),r} \mid tail(e), head(e) \notin \{s, i_1, j_1, i_2\}\right\}$$

that gives rise to the new inequality

$$x_{e_1} + x_{e_2} - 1 \leq x(F_{(e_1,e_2,\cdot),r}(S)). \tag{18}$$

Proposition 6. *If G contains no directed path from j_2 to i_1, then the inequalities (17) and (18) are valid for the RCSPP polytope.*

The proof is similar to that of Proposition 5. Now we give an example showing that a member of this new class of inequalities may be violated, while all $s - t$ cut precedence inequalities (9), (10) are satisfied.

Example 1. *Consider the graph in Figure 1. There is one resource, and the resource weights are indicated by the arcs. The only resource-infeasible path $\pi = (s, 2, 4, t)$ is not cut off by any $s - t$ cut precedence inequalities, but for the arcs $(s, 2)$ and $(4, t)$, and cut $S = \{s, 1, 2\}$, the inequality (17)*

$$x_{s,2} + x_{4,t} - 1 \leq x_{2,3}$$

is violated by the incidence vector of π.

Given a feasible solution x^* of the LP relaxation of (3)-(6), possibly augmented by some valid inequalities for RCSPP, and in which some variables may be fixed to 0 or 1 due to branching or preprocessing. To separate inequalities in this class, we fix e_1 and e_2 such that there is not any directed path from j_2 to i_1, and we consider the inequalities (16), (17), and (18) in turn. Suppose we want to find violated (16) inequalities. Firstly, we define a capacity function $g : E \to \mathbb{R}$ as follows. If an arc e is in $\Phi_{(\cdot,e_1,e_2),r}(S)$ and $tail(e) \notin \{j_1, i_2, j_2, t\}$ and $head(e) \notin \{j_1, i_2, j_2, t\}$, than let $g(e)$ be equal to x^*_e, otherwise $g(e) = 0$. Then we determine a minimum capacity $s - i_1$ cut S with respect to g. If

the minimum capacity is strictly smaller than $x^*_{e_1} + x^*_{e_2} - 1$, then a violated inequality is found (determined by S). It is easy to see that this procedure is of polynomial time, and since the number of pairs of arcs is $O(|E|^2)$, the inequalities (16), (17), and (18) can be separated in polynomial time.

Finally notice that the strengthening methods of Section 4.4 can be applied to (16), (17), and (18) as well.

5.2. Infeasible subpath based inequalities

Let $\pi = (\{i_1, \ldots, i_p\}, \{e_1, \ldots, e_{p-1}\})$ be an infeasible subpath of G for resource r (cf. Section 4.3), with $p \geq 4$. We argue that any resource-feasible path visiting e_1 and e_{p-1} must leave the subpath π at some node $i_2, \ldots, i_{p-2}$, otherwise it would contain all the arcs $e_1, \ldots, e_{p-1}$, and thus it would be infeasible for resource r. Therefore, for each $k = 2, \ldots, p-2$, we define the set of arcs

$$\tilde{\phi}^{out}_{\pi,r}(k) = \{e \in \delta^{out}(i_k) \mid \sigma^r_{s,i_1} + \sum_{\ell=1}^{k-1} w^r_{e_\ell} + w^r_e + \sigma^r_{head(e),i_{p-1}} + w^r_{e_{p-1}} + \sigma^r_{i_p,t} \leq W^r\},$$

and using $\tilde{\phi}^{out}_{\pi,r}(k)$, the arc set

$$\tilde{F}^{out}_{\pi,r}(k) = \{e \in \tilde{\phi}^{out}_{\pi,r}(k) \mid head(e) \neq i_{k+1}, head(e) \neq i_1, \ldots, i_{k-1}\}.$$

Using $\tilde{F}^{out}_{\pi,r} = \bigcup_{k=2}^{p-2} \tilde{F}^{out}_{\pi,r}(k)$, we define the inequalities

$$x_{e_1} + x_{e_{p-1}} - 1 \leq x(\tilde{F}^{out}_{\pi,r}). \tag{19}$$

Proposition 7. *If G contains no directed path from i_p to i_1, then the inequality (19) is valid for the RCSPP polytope.*

Proof. Let P be a resource-feasible path and x^P the corresponding vertex of the RCSPP polytope. If $x^P_{e_1} = 0$ or $x^P_{e_{p-1}} = 0$, then (19) is satisfied because the left-hand side is at most 0, while the right-hand-side is non-negative. Now suppose $x^P_{e_1} = x^P_{e_{p-1}} = 1$, i.e., P goes through both of e_1, and e_{p-1}. Since G contains no directed path from i_p to i_1 by assumption, P passes through e_1 first. Clearly, P cannot contain all the arcs e_2 through e_{p-2} as well, because π is a resource infeasible subpath for resource r. Hence, P must contain an arc emanating from one of the nodes $i_2, \ldots, i_{p-2}$. So let e' be the first arc on P emanating from node i_k of π. Since the path is simple, $head(e') \neq i_1, \ldots, i_{k-1}$, and $head(e') \neq i_{k+1}$. Moreover, since P is resource-feasible, $e' \in \tilde{F}^{out}_{\pi,r}(k)$ follows, and then $x^P_{e'} = 1$, and the statement of the proposition is proved. □

One can define analogously the sets

$$\tilde{\phi}^{in}_{\pi,r}(k) = \{e \in \delta^{in}(i_k) \mid \sigma^r_{s,i_1} + w^r_{e_1} + \sigma^r_{i_2,tail(e)} + w^r_e + \sum_{\ell=k}^{p-1} w^r_{e_\ell} + \sigma^r_{i_p,t} \leq W^r\},$$

for each $k = 3, \ldots, p-1$, and using $\tilde{\phi}^{in}_{\pi,r}(k)$, the arc set

$$\tilde{F}^{in}_{\pi,r}(k) = \{e \in \tilde{\phi}^{in}_{\pi,r}(k) \mid tail(e) \neq i_{k-1}, tail(e) \neq i_{k+1}, \ldots, i_p\}.$$

Let $\tilde{F}^{in}_{\pi,r} = \bigcup_{k=3}^{p-1} \tilde{F}^{in}_{\pi,r}(k)$, and we define the inequalities

$$x_{e_1} + x_{e_{p-1}} - 1 \leq x(\tilde{F}^{in}_{\pi,r}). \tag{20}$$

Proposition 8. *If G contains no directed path from i_p to i_1, then the inequality (20) is valid for the RCSPP polytope.*

The following two examples show that neither the subpath precedence (defined in Section 4.3), nor the Infeasible subpath based inequalities dominate the other.

Example 2. *Let us consider the graph in Figure 2. The arcs are indicated with the single resource requirement. There are two infeasible subpaths: $\pi_1 = (s, 2, 5, t)$ and $\pi_2 = (s, 2, 3, 5, t)$. Let us write the subpath precedence inequalities (11) and the infeasible subpath inequalities (19):*

$$\textit{subpath precedence inequality for } \pi_1 \quad : \quad x_{s,2} \leq x_{2,3} + x_{2,4} + x_{5,6} \tag{21}$$

$$\textit{subpath precedence inequality for } \pi_2 \quad : \quad x_{s,2} \leq x_{2,4} + x_{2,5} + x_{5,6} \tag{22}$$

$$\textit{infeasible subpath inequality for } \pi_1, \pi_2 \quad : \quad x_{s,2} + x_{5,t} \leq x_{2,4} + 1 \tag{23}$$

Condition (23) excludes both π_1 and π_2, but (21) excludes only π_1 and (22) excludes only π_2.

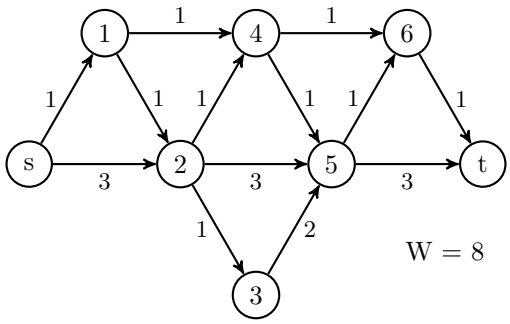

Figure 2: Network for Example 2.

Example 3. *Let us consider the graph in Figure 3. The arcs are indicated with the single resource requirement. There are three infeasible subpaths: $\pi_1 =$*

$(s, 2, 4, t)$, $\pi_2 = (s, 2, 4, 6, t)$ *and* $\pi_3 = (s, 2, 4, 6)$. *Let us write the subpath precedence inequalities (11) and the infeasible subpath inequalities (19):*

$$\begin{aligned}
\textit{subpath precedence inequality for } \pi_1, \pi_2, \pi_3 &: x_{s,2} \leq x_{2,3} + x_{4,5} && (24)\\
\textit{infeasible subpath inequality for } \pi_1 &: x_{s,2} + x_{4,t} \leq x_{2,3} + 1 && (25)\\
\textit{infeasible subpath inequality for } \pi_2 &: x_{s,2} + x_{4,6} \leq x_{2,3} + 1 && (26)\\
\textit{infeasible subpath inequality for } \pi_3 &: x_{s,2} + x_{6,t} \leq x_{2,3} + 1 && (27)
\end{aligned}$$

Condition (24) excludes all of π_1, π_2 *and* π_3, *but (25), (26) and (27) only exclude* π_1, π_2 *and* π_3, *respectively.*

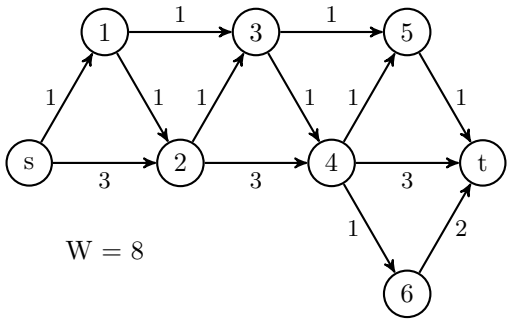

Figure 3: Network for Example 3.

The separation of the inequalities (19) and (20) is not an easy problem. Firstly, we prove that the separation of the subpath precedence inequalities (11) and (12) is NP-hard, thus solving an open problem raised in [8].

Problem (SPP-SEP). Subpath Precedence Separation
Instance: We are given a directed graph $G = (V, E)$, a fractional solution $x^* \in [0, 1]^E$ for the LP relaxation of (3) - (6), arc weights $w_e \in \mathbb{R}$ for each $e \in E$, two vertices $s, t \in V$, and capacity $W \in \mathbb{R}$.
Question: Is there an $s - t$ subpath $\pi = (\{i_1, \ldots, i_p\}, \{e_1, \ldots, e_{p-1}\})$ in G such that $p \geq 4$, $\sigma_{s,i_1} + w(E_\pi) + \sigma_{i_p,t} > W$ and $x^*_{e_1} > x^*(F^{out}_\pi)$?

In the following NP-hardness proof we will make use of the well-known Knapsack Problem.

Problem (KP). Knapsack Problem
Instance: We are given a set of n items, each with a non-negative profit p_i, and a non-negative weight a_i. Additionally we are given a capacity value c, and a desired profit value P. All problem data is integer.
Question: Is there a subset J of items such that $p(J) > P$ and $a(J) < c$?

Let $p_{sum} = \sum_{i=1}^{n} p_i$ denote the sum of all profits in an instance of the knapsack problem. After these preliminaries, we are ready to prove the following:

Proposition 9. *SPP-SEP is NP-hard.*

Proof. We reduce the NP-hard KP problem to SPP-SEP. Given an instance of the KP problem, renumber the items such that $a_1 \leq a_2 \leq \ldots \leq a_n$. Without loss of generality we may assume that $c > 0$, $1 \leq a_i \leq c$ and $1 \leq p_i \leq P$ for all $i = 1, \ldots, n$. Divide every weight and capacity value by $a_n + c$ to obtain the values $\bar{a}_i$ and $\bar{c}$:

$$\bar{a}_i = \frac{a_i}{a_n + c} \quad (i = 1, \ldots, n) \quad \text{and} \quad \bar{c} = \frac{c}{a_n + c}$$

If it is necessary, multiply the values $p_1, \ldots, p_n$, and P by a suitable integer to ensure that

$$\bar{c} \leq (P + p_{sum} + 1)/(P + p_{sum} + 2). \tag{28}$$

Clearly, this can be done, since $\bar{c}$ is smaller than 1. For the sake of simplicity we do not use another notations for these modified values, that is, we assume that (28) is met for the values $p_1, \ldots, p_n$, and P.

We create an acyclic digraph G as follows. G has $n + 3$ nodes denoted by $0, 1, \ldots, n + 2$, where $s = 0$ is the source and $t = n + 2$ is the sink. Every pair of nodes $(i, i + 1)$, $i = 1, \ldots, n + 1$, is connected by two arcs, $e^+_{i,i+1}$ and $e^0_{i,i+1}$. Furthermore, there are n more arcs from node 0 to each of the nodes $1, \ldots, n$: $e_{0,i} = (0, i)$ for $i = 1, \ldots, n$, and finally from node 0 to node 1: $e^+_{0,1} = (0, 1)$ and from node 1 to node $n + 2$: $e_{1,n+2} = (1, n + 2)$. We define the arc-weights w, and an $s - t$ flow x^* as follows:

$$\begin{array}{lll}
w(e^+_{0,1}) = p_{sum} + 1, & x^*(e^+_{0,1}) = \bar{c} & \\
w(e^+_{i,i+1}) = p_i, & x^*(e^+_{i,i+1}) = 0, & i = 1, \ldots, n \\
w(e^+_{n+1,n+2}) = p_{sum} + 1, & x^*(e^+_{n+1,n+2}) = \bar{a}_n & \\
w(e^0_{i,i+1}) = 0, & x^*(e^0_{i,i+1}) = \bar{a}_i, & i = 1, \ldots, n \\
w(e^0_{n+1,n+2}) = 0, & x^*(e^0_{n+1,n+2}) = 0 & \\
w(e_{0,i}) = 0, & x^*(e_{0,i}) = \bar{a}_i - \bar{a}_{i-1}, & i = 1, \ldots, n \\
w(e_{1,n+2}) = P + p_{sum} + 2, & x^*(e_{1,n+2}) = \bar{c} &
\end{array}$$

where $\bar{a}_0 = 0$. Let $W = P + 2p_{sum} + 2$. This network along with the flow x^* is depicted in Figure 4. It is clear that x^* is an $s - t$ flow of value $\bar{a}_n + \bar{c} = 1$, and x^* is a feasible solution for the LP relaxation of the RCSPP problem, since $\sum_{e \in E} x^*_e w_e = (\bar{a}_n + \bar{c})(p_{sum} + 1) + \bar{c}(P + p_{sum} + 2) \leq P + 2p_{sum} + 2 = W$ holds, due to (28). We claim there exists a solution for KP if and only if there exists a solution for SPP-SEP in G.

First suppose the separation problem SPP-SEP admits a solution, and let $\pi = (i_1, \ldots, i_p)$ be an infeasible subpath with $p \geq 4$ such that $x^*(F^{out}_\pi) < x^*(\pi[0, 1])$. Since arc $e_{1,n+2}$ cannot appear in an infeasible subpath with minimum length 3, E_π does not contain $e_{1,n+2}$. It is clear that $\sigma_{s,i} = \sigma_{i,t} = 0$ for all $i = 0, \ldots, n + 2$, and $\sum_{i=1}^{n} \max\{w(e^+_{i,i+1}), w(e^0_{i,i+1})\} = p_{sum}$, hence an infeasible subpath with minimum length 3 contains both of the arcs $e^+_{0,1}$ and $e^+_{n+1,n+2}$, and thus π is an $s - t$ path. Now we determine F^{out}_π. Firstly, notice that

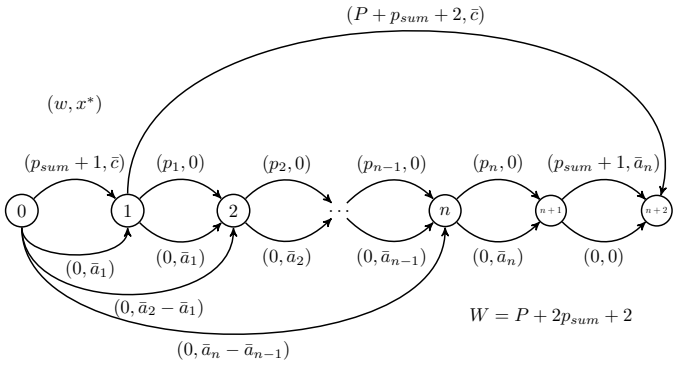

Figure 4: The constructed graph

$e_{1,n+2} \notin F_\pi^{out}$, since $w(e_{0,1}^+) + w(e_{1,n+2}) = P + p_{sum} + 3 > W$, i.e., the $s-t$ path consisting of the edges $e_{0,1}^+$ and $e_{1,n+2}$ is not resource feasible. If π contains an arc $e_{i,i+1}^+$ for some $i = 1, \ldots, n+1$, then F_π^{out} comprises the arc $e_{i,i+1}^0$, since $w(\pi[0,i]) \leq 2p_{sum} + 1$, $w(e_{i,i+1}^0) = 0$, and $\sigma_{i+1,n+2} = 0$. On the other hand, if π contains an arc $e_{i,i+1}^0$ for some $i = 1, \ldots, n$, then F_π^{out} comprises the arc $e_{i,i+1}^+$, since $w(\pi[0,i]) \leq 2p_{sum} + 1$, $w(e_{i,i+1}^+) = p_i$, and $\sigma_{i+1,n+2} = 0$. Therefore, letting $J := \left\{ i \in \{1, \ldots, n\} \ : \ e_{i,i+1}^+ \in E_\pi \right\}$, we have proved that

$$F_\pi^{out} = \left(\bigcup_{i \in J} \{e_{i,i+1}^0\} \right) \cup \left(\bigcup_{i \notin J} \{e_{i,i+1}^+\} \right) \cup \{e_{n+1,n+2}^0\}.$$

Thus, if π is a solution for SPP-SEP, i.e., π is an infeasible subpath and x^* violates (11), i.e., $x^*(F_\pi^{out}) < x^*(e_{0,1}^+) = \bar{c}$, then J is a solution for KP, since $p(J) = w(\pi) - w(e_{0,1}^+) - w(e_{n+1,n+2}^+) > P$, and $\bar{a}(J) = x^*(F_\pi^{out}) < \bar{c}$ if and only if $a(J) < c$.

Conversely, let J be a solution for KP. We define the path π with

$$E_\pi = \{e_{0,1}^+, e_{n+1,n+2}^+\} \cup \left(\bigcup_{i \in J} \{e_{i,i+1}^+\} \right) \cup \left(\bigcup_{i \notin J} \{e_{i,i+1}^0\} \right).$$

It is easy to verify that $F_\pi^{out} = \left(\bigcup_{i \in J} \{e_{i,i+1}^0\} \right) \cup \left(\bigcup_{i \notin J} \{e_{i,i+1}^+\} \right) \cup \{e_{n+1,n+2}^0\}$, and the inequality (11) is violated by x^*. □

Now we turn to the separation problem for the inequalities (19) and (20).

Problem (IS-SEP). Infeasible Subpath Separation
Instance: We are given a directed graph $G = (V, E)$, arcs e^1, e^2, a fractional

solution $x^* \in [0,1]^E$, arc weights $w_e \in \mathbb{R}$ for each $e \in E$, two vertices $s, t \in V$, and capacity $W \in \mathbb{R}$.
Question: Is there a subpath $\pi = (\{i_1, \ldots, i_p\}, \{e_1, \ldots, e_{p-1}\})$ in G such that $p \geq 5$, $e_1 = e^1$, $e_{p-1} = e^2$, $\sigma_{s,i_1} + w(E_\pi) + \sigma_{i_p,t} > W$ and $x^*_{e_1} + x^*_{e_{p-1}} - 1 > x^*(\tilde{F}^{out}_\pi)$?

Proposition 10. *IS-SEP is NP-hard.*

Proof. The construction is almost the same as that in the proof of Proposition 9. To be suitable for the present claim, extend the graph G with a new node and a new arc, denoted by $n+3$ and $e_{n+2,n+3}$, respectively, where $head(e_{n+2,n+3}) = n+3$ and $tail(e_{n+2,n+3}) = n+2$. The weight of the new arc is 0, and let $x^*_{e_{n+2,n+3}} = 1$. Accordingly, the new sink node is $t = n+3$, the source node remains $s = 0$. Let $e^1 = e^+_{0,1}$ and $e^2 = e_{n+2,n+3}$.

One may verify that the KP problem admits a solution J if and only if there is an infeasible subpath π with length at least 4, with $e^+_{0,1}$ as the first edge, $e_{n+2,n+3}$ as the last edge, and $x^*(e^+_{0,1}) + x^*(e_{n+2,n+3}) - 1 > x^*(\tilde{F}^{out}_\pi)$. □

For separating the inequalities (19), we propose the heuristic method shown in Algorithm 1. By using the procedure INFEASIBLE_SUBPATH_DFS we find an infeasible $s-t$ subpath which lies (or partially lies) in the support graph of the solution x^*. In the general step we have a feasible subpath π consisting of the arcs $e_1, \ldots, e_{p-1}$ in this order (line 8), and we revise the outgoing arcs from its last node $head(e_{p-1})$ (line 10). If the current arc does not create a cycle, the solution value on the arc is positive, and the extended path is also resource feasible, we store the arc in the set $\mathcal{F}$ (line 13). Otherwise, if the extended path is infeasible and still elementary we call the procedure EVAL_OUT to create inequality (19) for that path (line 17). This simple procedure verifies all the arcs that leave the subpath (lines 28-29) and checks whether an arc is in $\tilde{F}^{out}_{\pi;r}$ (line 30). Finally, if we get a violated inequality, we store it in the set $\mathcal{C}$ (line 36).

One can devise a similar method to separate inequalities (20) by modifying the procedure EVAL_OUT. It is possible to combine the separation of inequalities (19) and (20), because the procedure INFEASIBLE_SUBPATH_DFS is the same in both cases. That is, after an infeasible path is found, we call procedure EVAL_OUT and a similar procedure EVAL_IN to separate inequalities (19) and (20), respectively.

6. Variable fixing and primal heuristic

In this section we present our depth-first-search based feasible solution search heuristic, and our variable fixing procedure based on the preprocessing method of Dumitrescu and Boland.

Algorithm 1 Heuristic for separating inequalities (19) for $x^* \in [0,1]^E$

$\mathcal{C} \leftarrow \emptyset$
for $e_1 \in E^*$ **do**
  $\pi \leftarrow e_1$
  INFEASIBLE_SUBPATH_DFS(π)
end for
return $\mathcal{C}$

procedure INFEASIBLE_SUBPATH_DFS($\pi = (e_1, e_2, \ldots, e_{p-1})$):
$\mathcal{F} \leftarrow \emptyset$
for $e_p \in \delta^{out}(head(e_{p-1}))$ **do**
  if $\sigma^r_{s,tail(e_1)} + w^r(\pi) + w^r_{e_p} + \sigma^r_{head(e_p),t} \leq W^r$ **then**
    if $x^*_{e_p} > 0$ and $head(e_p) \notin V_\pi \cup \{s\}$ **then**
      $\mathcal{F} \leftarrow \mathcal{F} \cup \{e_p\}$
    end if
  else
    if $head(e_p) \notin V_\pi \cup \{s\}$ **then**
      EVAL_OUT($\pi \oplus e_p$)
    end if
  end if
end for
for $e_p \in \mathcal{F}$ such that $head(e_p) \neq t$ **do**
  INFEASIBLE_SUBPATH_DFS($\pi \oplus e_p$)
end for
end procedure

procedure EVAL_OUT($\pi = (e_1, e_2, \ldots, e_p)$):
$y \leftarrow x^*_{e_1} + x^*_{e_p} - 1$
for $j = 1, \ldots, p-2$ **do**
  for $e' \in \delta^{out}(head(e_j)) - \{e_{j+1}\}$ **do**
    if $\sigma^r_{s,tail(e_1)} + w^r(\pi[0,j]) + w^r_{e'} + \sigma^r_{head(e'),tail(e_p)} + w^r_{e_p} + \sigma_{head(e_p),t} \leq W^r$ and $head(e') \notin V(\pi) \cup \{s\}$ **then**
      $y \leftarrow y - x^*_{e'}$
    end if
  end for
end for
if $y > 0$ **then**
  $\mathcal{C} \leftarrow \mathcal{C} \cup \left\{ x_{e_1} + x_{e_p} - 1 \leq x(\tilde{F}^{out}_{\pi;r}) \right\}$
end if
end procedure

6.1. Primal solution search heuristic

The aim of this procedure is to find an $s-t$ path which is feasible for all resource constraints starting out from a fractional solution x^* to the LP relaxation. If such an $s-t$ path is found we may strengthen the actual upper bound in the branch-and-cut procedure, thus we may improve its performance. Briefly stated, we perform a depth-first-search from s on the support graph of x^*, i.e., $G^* = (V, E^*)$ where $E^* = \{e \in E \mid x^*_e > 0\}$. Once we reach a previously processed node i, we may improve the best upper bound along an $s-t$ path through i. The sketch of our procedure can be seen in Algorithm 2.

At the beginning we create an empty stack S and insert s into it. During the procedure the following two conditions always hold:

- When we process a node u, the label $cost_s(u)$ denotes the cost of an $s-u$ path $\pi_s(u)$, and the label $\vartheta^r_s(u)$ denotes the resource consumption from the resource r of the same path. Furthermore, $path(v)$ is true if and only if $\pi_s(u)$ contains v.
- After a node u has been processed, i.e., $processed(v)$ is $true$, the label $cost_t(u)$ denotes the cost of an $u-t$ path $\pi_t(u)$, and the label $\vartheta^r_t(u)$ denotes the resource consumption from the resource r of the same path.

After the initialization steps (lines 1 - 7), these conditions clearly hold. In a general step, we consider the node u most recently inserted on stack S. If u has already been processed, i.e., $processed(u) = true$, we remove u from S. Otherwise, we set $path(u)$ to $true$, and visit each outgoing arc $e \in \delta^{out}(u)$ in turn. If adding some $e \in \delta^{out}(u)$ to the path $\pi_s(u)$ would create a cycle, that is, the head node $v = head(e)$ is on the path $\pi_s(u)$ (i.e., $path(v) = true$) we finish processing this arc. Otherwise, we distinguish between two cases: (i) if $v = head(e)$ is not processed, we change the labels of v with respect to the path $\pi_s(v) = \pi_s(u) \oplus e$ (lines 21-22), and insert v into S; (ii) if v has already been processed (being part of a path explored before), we change the label of u corresponding to the $u-t$ path $e \oplus \pi_t(v)$ (lines 26-27). After we revised all outgoing arcs from u, we set $processed(u)$ to $true$.

If an $s-t$ path is found (lines 11 and 24), we may update U (lines 12 and 29), that is, if $\vartheta^r_s(u) + \vartheta^r_t(u) \leq W^r$ for each resource r and $cost_s(u) + cost_t(u) < U$ we replace U with $cost_s(u) + cost_t(u)$.

6.2. Variable fixing based on Dumitrescu-Boland preprocessing

Now, we describe our variable fixing procedure based on the preprocessing method of Dumitrescu and Boland (DB-preprocessing) to be applied in each search-tree node prior to solving the corresponding node LP. To goal is to fix arcs (i.e., the corresponding variables) to 0 if they cannot appear in any optimal solution of the branch-and-bound subtree rooted at the node. In addition, this method may also find primal solutions.

If a variable x_e ($e \in E$) is fixed to 0 (to 1) in a search-tree node then we seek a minimal cost, resource feasible $s-t$ path π, such that π does not use (uses)

Algorithm 2 DFS based feasible solution search heuristic

$U \leftarrow$ best upper bound
$\vartheta_s^r(s) \leftarrow 0$ and $\vartheta_t^r(t) \leftarrow 0$ for all r
$\vartheta_s^r(u) \leftarrow \infty$ for all $u \in V - s$ and $\vartheta_t^r(u) \leftarrow \infty$ for all $u \in V - t$ for all r
$cost_s(s) \leftarrow 0, cost_s(u) \leftarrow \infty$ for all $u \in V - s$
$cost_t(t) \leftarrow 0, cost_t(u) \leftarrow \infty$ for all $u \in V - t$
$processed(u) \leftarrow false$ for all $u \in V$
$path(u) \leftarrow false$ for all $u \in V$
insert s into an empty stack S
while $S \neq \emptyset$ **do**
  $u \leftarrow S.top()$
  if $processed(u) = true$ **then**
    update U
    $path(u) \leftarrow false$
    $S.pop()$
  else
    $path(u) \leftarrow true$
    for $e \in \delta^{out}(u) \cap E^*$ **do**
      $v \leftarrow head(e)$
      if $path(v) = false$ **then**
        if $processed(v) = false$ **then**
          $cost_s(v) = cost_s(u) + c(e)$
          $\vartheta_s^r(v) \leftarrow \vartheta_s^r(u) + w_e^r$ for all r
          insert v into S
        else
          if $cost_t(u) < c(e) + cost_t(v)$ and $\vartheta_t^r(u) \leq w_e^r + \vartheta_t^r(v)$ for all r **then**
            $cost_t(u) \leftarrow c(e) + cost_t(v)$
            $\vartheta_t^r(u) \leftarrow w_e^r + \vartheta_t^r(v)$ for all r
          end if
          update U
        end if
      end if
    end for
    $processed(u) \leftarrow true$
  end if
end while

arc e. Thus, before we optimize a branch-and-bound node we create a subgraph $\bar{G}$ of G such that $\bar{G}$ contains all paths that are feasible according to the fixed variables (however, $\bar{G}$ may contain other paths as well), and then we perform DB-preprocessing on subgraph $\bar{G}$. If the preprocessing finds that there is no resource feasible path in $\bar{G}$, then we can cut off the current branch-and-bound node. If the preprocessing erases an arc e from $\bar{G}$ (that is, there is no optimal $s-t$ path through e) we can fix the corresponding variable x_e to 0. If the preprocessing finds a resource feasible $s-t$ path in $\bar{G}$ (i.e., a primal solution for the global problem) we may update the best upper bound on the optimal solution, moreover, if the preprocessing proves that the found path is optimal in $\bar{G}$ we can cut off the current branch-and-bound node.

For the sake of completeness we describe our method to create subgraph $\bar{G}$. The sketch of the procedure can be seen in Algorithm 3. Let $E_0 \subseteq E$, and $E_1 \subseteq E$ denote the set of arcs previously fixed to 0, and to 1, respectively. First, we initialize $\bar{G}$ as G, and erase all arcs from $\bar{G}$ that belong to E_0. Then, for each arc $e \in E_1$ we determine the set of nodes P^- from which $tail(e)$ is reachable avoiding $head(e)$ in the current subgraph. Symmetrically, we determine set P^+. If for an arc $f \in E(\bar{G}) \setminus \{e\}$ both of $\{head(f), tail(f)\} \nsubseteq P^-$ and $\{head(f), tail(f)\} \nsubseteq P^+$ hold, it means that there is no $s-t$ path in $\bar{G}$ consisting both of the arcs e end f. If such an arc is in E_1 we can terminate with $E(\bar{G}) = \emptyset$, because there is no $s-t$ path in the reduced graph that is feasible according to the fixed arcs. Otherwise, we erase this arc from $\bar{G}$.

Algorithm 3 Creating subgraph $\bar{G}$

Input: G; $E_0, E_1 \subseteq E(G)$ such that $E_0 \cap E_1 = \emptyset$
$\bar{G} \leftarrow G$
for $e \in E_0$ **do**
  $\bar{G} \leftarrow \bar{G} \setminus e$
end for
for $e \in E_1$ **do**
  $P^- \leftarrow \rho^{in}_{\bar{G} \setminus head(e)}(tail(e))$
  $P^+ \leftarrow \rho^{out}_{\bar{G} \setminus tail(e)}(head(e))$
  for $f \in E(\bar{G}) \setminus \{e\}$ **do**
    if $\{head(f), tail(f)\} \nsubseteq P^-$ **and** $\{head(f), tail(f)\} \nsubseteq P^+$ **then**
      if $f \in E_1$ **then**
        return $\emptyset$
      else
        $\bar{G} \leftarrow \bar{G} \setminus f$
      end if
    end if
  end for
end for
return $\bar{G}$

7. Evaluation of cutting planes and heuristics

In this section we summarize our computational experiments. The main goals of the experiments were

- to show that some of the new cutting planes can significantly improve the performance of a branch-and-cut type algorithm for solving RCSPP,
- to assess the effectiveness of the new preprocessing and heuristic algorithms,
- to find the best combination of the various techniques for solving hard instances.

In the following sections we sketch our computational framework, then then we address the above points in turn. We conclude these experiments in Section 7.7.

7.1. Test environment and implementation

All the computational experiments were performed on a workstation with 4GB RAM, and XEON X5650 CPU of 2.67 GHz, and under Linux operating system using a single thread only.

Our branch-and-cut solver has been implemented in the C++ programming language using the FICO Xpress Optimization Suite (version 23.01.06) [7] (Xpress) as a branch-and-cut framework. We also used the LEMON C++ library (version 1.3.1) [14] to handle graphs and to perform graph algorithms.

In all of the following experiments, before building an IP model of the problem, the input graph was preprocessed by two of the procedures from the literature (see Section 4.5). Firtly, the procedure of Dumitrescu and Boland [6] (DB-preprocessing) was applied to remove some arcs, and if it also returned a feasible solution, we used its value as an upper bound on the optimum in the branch-and-cut procedure. In addition, the method of Garcia [8] was applied (GQ-preprocessing) to eliminate some nodes from the input graph.

7.2. Instances

We performed experiments on several instance sets. First, we used the instances of Beasley and Christofides [3] that range in size from 100 nodes and 955 arcs to 500 nodes and 4978 arcs, and can be grouped into two sets. The first set consists of 12 instances with 1 resource function, while the other 12 instances in the second set use 10 resources. However, we found that these instances are easy to solve and cannot be used to challenge our branch-and-cut procedure. That is, 22 instances out of 24 can be solved using just preprocessing, and the remaining two instances are solved by Xpress at the root node. For results we refer to Appendix B. Therefore, we decided to generate our own instances using the similar generation method as in [8] that we describe next.

To construct a directed graph we used a method similar to that of [3]. Let n be the number of desired nodes, and denote $V = \{1, 2, \ldots, n\}$ the set of nodes with $s = 1$ and $t = n$. For all $i = 1, \ldots, n-1$ and for all $j = i+1, \ldots, \min\{n, i+$

Table 1: Summary of the problem instances.

Class	Nodes	Res	Method
G1	500	10	Beasley-Christofides ($p = 20$ %)
G2	500	10	Uniform ($W = 20$)
G3	500	10	Uniform ($W = 30$)
G4	500	10	Uniform ($W = 40$)
G5	1000	10	Beasley-Christofides ($p = 20$ %)
G6	1000	10	Uniform ($W = 20$)
G7	1000	10	Uniform ($W = 30$)
G8	1000	10	Uniform ($W = 40$)
G9	1500	10	Beasley-Christofides ($p = 20$ %)
G10	1500	10	Uniform ($W = 20$)
G11	1500	10	Uniform ($W = 30$)
G12	1500	10	Uniform ($W = 40$)

$\lfloor n/4 \rfloor\}$ we randomly include arc (i, j) with a probability such that the expected value of the number of arcs is $10n$. Since for all arcs (i, j), $j - i \leq n/4$, every $s-t$ path consisted of at least 4 arcs. Clearly, the generated graphs are directed, acyclic, and do not contain parallel arcs.

In each of our instances all arc weights and all arc costs are integers. The weights were uniformly and independently generated from $[0, 5]$, and arc costs were uniformly and independently generated from $[-5, 0]$. To create resource limits we used two different methods. The first one is similar to that in [3], that is, we searched a minimal cost $s - t$ path and computed its resource consumptions. The resource limits were derived from these values, reduced by a given percentage p (see Beasley and Christofides [3]). In the second method we chose a fixed uniform limit W for all resources, like in [8].

We generated 20 graphs for each $n \in \{500, 1000, 1500\}$. For each graph we generated a cost function and 10 resource functions, and then we derived four instances by the four ways of setting the resource limits. That is, we used the Beasley-Christofides method with $p = 20\%$. The other 3 instances had uniform resource limits with $W = 20, 30$, and 40, respectively. Since every $s - t$ path consists of at least 4 arcs, and the maximum arc weight is 5, each RCSPP instance with uniform resource limits has a feasible solution.

In summary, we have created $240 = 20 \times 3 \times 4$ RCSPP instances which can be grouped into 12 classes according to their sizes, and the method used to generate their resource limits. Table 1 contains the parameters of the instances in the different classes, namely the number of nodes (Nodes), the number of resource functions (Res), and the resource limit generating method (Method). Tables A1, A2 and A3 of the Supplementary Material contain more information about the instances.

7.3. Heuristic and variable fixing experiments

In this section we present the results of the experiments of heuristic and variable fixing methods described in Section 6. Our purpose was to investigate

how these methods can improve a simple branch-and-bound procedure. For the sake of a fair comparison, in these experiments we turned off every Xpress presolving and heuristic method (i.e., `XPRS_HEURSTRATEGY`, `XPRS_PRESOLVE` and `XPRS_MIPPRESOLVE` were set to 0) and we did not add any cutting plane to the problem (i.e., `XPRS_CUTSTRATEGY` was set to 0). We call these the PLAIN settings.

The ARCFIX setting refers to the use of the variable fixing method of Garcia [8] at the root node. As described in Section 6 we used our Dumitrescu-Boland preprocessing based variable fixing method (setting DBVARFIX) in every branch-and-bound node before solving the LP relaxation, while our heuristic solution search method (setting HEURSOL) were used in every branch-and-bound node after an optimal solution for the LP relaxation had been found. On each instance every method was tested separately, and all together, which is the ALL method.

The summary of the experiments can be found in Table 2; whereas the detailed computations are provided in Table A4 of the Supplementary Material. In Table 2 we only indicate the average number of the explored branch-and-bound nodes (BBN) and the average running time (Time) of the entire branch-and-bound procedure in seconds.

The most successful among the above methods in is definitely the Dumitresu-Boland based variable fixing technique, as it can reduce the total time by 54,8–89,6% with respect to the PLAIN method. Moreover, in most of the cases we obtained the best results by the ALL method, that is, when we combined all of the fixing and heuristic search techniques.

7.4. Experiments with cutting planes based on $s-t$ cuts

In this section we summarize the experiments with inequalities described in Section 4.2 and Section 5.1. Our purpose was to compare the performance of the $s-t$ cut precedence inequalities and that of our generalized inequalities. In these experiments we turned off every Xpress presolving and heuristic method (i.e., `XPRS_HEURSTRATEGY`, `XPRS_PRESOLVE` and `XPRS_MIPPRESOLVE` were set to 0) and we forbade Xpress to add any cutting plane of his own to the problem (i.e., `XPRS_CUTSTRATEGY` was set to 0). Moreover, in these experiments we gave the optimal solution value to the solver (i.e., `XPRS_MIPABSCUTOFF` was set to the optimal value).

The summary of the experiments can be found in Table 3, 4 and 5, respectively; whereas the detailed computations are provided in Table A5 of the Supplementary Material. In Tables 3, 4 and 5 we only indicated the average number of the investigated branch-and-bound nodes (BBN), the average total running time (Time) of the branch-and-cut procedure in seconds, and the average number of generated cuts (Inequalities/{stcp,aca,caa,aac}).

The STCP setting refers to the use of the strengthened $s-t$ cut precedence inequalities (9) and (10). We generated these inequalities for an arc e only if the solution value on the arc was positive (i.e., $x_e^* > 0$). The CAA, ACA and AAC settings refer to the use of our cut based inequalities (16), (17) and (18),

Table 2: Summary of fixing and heuristic experiments.

Setting	Class	BBN	Time	Class	BBN	Time	Class	BBN	Time
PLAIN	G1	1360.1	57.7	G5	1499.2	235.3	G9	2165.9	644.7
ARCFIX		1153.0	52.0		1328.2	211.9		1884.0	554.4
HEURSOL		1191.3	49.6		1223.9	175.2		1841.0	501.1
DBVARFIX		881.7	19.0		1000.3	54.0		1101.4	93.5
ALL		775.3	19.4		877.8	53.6		1090.3	103.5
PLAIN	G2	1883.6	88.9	G6	2621.3	392.9	G10	2308.4	679.9
ARCFIX		1982.6	93.7		2464.7	375.0		1932.0	623.4
HEURSOL		1769.3	81.5		2303.7	339.2		2092.1	580.8
DBVARFIX		1351.3	33.5		1149.5	63.5		782.5	71.0
ALL		1265.9	31.8		1086.1	66.8		660.6	83.2
PLAIN	G3	4275.3	187.1	G7	4877.5	709.0	G11	5533.7	1400.7
ARCFIX		4275.3	187.7		4877.5	706.8		5533.7	1407.4
HEURSOL		3261.0	134.2		4410.4	615.0		4637.2	1087.5
DBVARFIX		3775.9	84.5		4281.7	235.4		4864.3	432.6
ALL		3412.2	76.6		3722.3	213.0		3990.3	370.9
PLAIN	G4	3699.0	143.4	G8	4222.5	560.8	G12	6044.6	1434.0
ARCFIX		3699.0	141.9		4222.5	556.6		6044.6	1429.1
HEURSOL		2742.5	101.0		3644.5	444.9		5150.7	1073.3
DBVARFIX		3294.8	63.9		4620.6	223.2		5636.2	469.2
ALL		3102.2	62.4		3982.5	197.9		5067.4	432.3

respectively. We generated these inequalities for a pair of arcs (e_1, e_2) such that $x^*_{e_1} > 0$, $x^*_{e_2} > 0$ and the pair (e_1, e_2) is compatible for all resources. The ALL setting refers to the simultaneous use of all the previous inequalities in the same experiments. Cuts were generated in each node with depth at most 8 in one round, except the root node where we separate inequalities in 20 rounds.

We can observe that the efficiency of the procedure depends on the types of the instances rather than their sizes. That is, for all problem sizes, for resource limit types Beasley-and-Christofides(80) and Uniform(20) we obtained the best results either by the ACA or by the ACA + CAA + AAC method; furthermore, for resource limit types Uniform(30) and Uniform(40) the STCP method gave the best results in almost all cases. One of the reasons for this is that RCSPP instances with resource limit types Uniform(30) and Uniform(40) contain a few incompatible arc-pairs. Thus very few inequalities can be generated, however, the generation of inequalities (16), (17) and (18) is more expensive in total than the generation of the $s - t$ cut precedence inequalities.

7.5. Experiments with cutting planes based on infeasible subpaths

In this section we summarize the experiments with cutting planes based on infeasible subpaths as described in Section 4.3 and Section 5.2. Our purpose was to compare the performance of the subpath precedence inequalities and our generalized inequalities. In these experiments we turned off every Xpress presolving and heuristicsolution search methods (i.e., `XPRS_HEURSTRATEGY`, `XPRS_PRESOLVE` and `XPRS_MIPPRESOLVE` were set to 0) and we forbade Xpress to add any cutting

Table 3: Results with $s-t$ cut based cutting planes on 500-node instances.

		Inequalities					
Class	Settings	stcp	aca	caa	aac	BBN	Time
G1	STCP	121.4	0.0	0.0	0.0	1193.8	46.3
	ACA	0.0	241.6	0.0	0.0	1089.2	44.3
	STCP + ACA	130.4	218.5	0.0	0.0	1194.1	48.8
	ACA + CAA + AAC	0.0	234.8	132.1	148.2	1080.2	44.7
	ALL	128.0	194.4	122.6	108.4	1134.4	49.0
G2	STCP	65.2	0.0	0.0	0.0	1548.4	67.6
	ACA	0.0	365.5	0.0	0.0	1511.1	66.4
	STCP + ACA	69.1	396.4	0.0	0.0	1503.3	67.1
	ACA + CAA + AAC	0.0	310.1	196.0	187.6	1429.4	65.2
	ALL	57.6	297.7	178.1	172.6	1484.5	66.2
G3	STCP	0.3	0.0	0.0	0.0	2812.4	107.7
	ACA	0.0	89.3	0.0	0.0	2815.0	113.1
	STCP + ACA	0.7	89.0	0.0	0.0	2818.8	113.4
	ACA + CAA + AAC	0.0	80.3	37.8	53.2	2809.1	119.0
	ALL	0.4	80.4	37.3	53.0	2809.4	119.3
G4	STCP	0.0	0.0	0.0	0.0	2403.2	86.9
	ACA	0.0	2.2	0.0	0.0	2416.4	95.9
	STCP + ACA	0.0	2.2	0.0	0.0	2416.4	96.2
	ACA + CAA + AAC	0.0	2.1	0.6	1.1	2405.2	109.1
	ALL	0.0	2.1	0.6	1.1	2405.2	109.6

Table 4: Results with $s-t$ cut based cutting planes on 1000-node instances.

		Inequalities					
Class	Settings	stcp	aca	caa	aac	BBN	Time
G5	STCP	163.2	0.0	0.0	0.0	1068.4	140.5
	ACA	0.0	449.2	0.0	0.0	1011.9	132.7
	STCP + ACA	178.8	454.3	0.0	0.0	976.0	134.3
	ACA + CAA + AAC	0.0	398.4	178.1	291.5	993.1	132.0
	ALL	162.7	410.5	190.4	288.9	1000.4	133.1
G6	STCP	172.4	0.0	0.0	0.0	2104.8	284.6
	ACA	0.0	472.3	0.0	0.0	1995.1	269.9
	STCP + ACA	169.2	380.3	0.0	0.0	2081.0	281.6
	ACA + CAA + AAC	0.0	459.6	278.0	265.4	1958.8	266.8
	ALL	186.9	369.8	184.6	253.7	2011.5	269.5
G7	STCP	5.2	0.0	0.0	0.0	3513.1	465.3
	ACA	0.0	341.6	0.0	0.0	3591.4	486.7
	STCP + ACA	4.9	336.8	0.0	0.0	3576.4	487.6
	ACA + CAA + AAC	0.0	336.1	158.2	223.0	3539.1	489.7
	ALL	5.1	345.0	168.5	229.0	3553.8	493.8
G8	STCP	0.1	0.0	0.0	0.0	3299.8	392.3
	ACA	0.0	16.7	0.0	0.0	3352.1	417.6
	STCP + ACA	0.1	16.7	0.0	0.0	3347.1	418.6
	ACA + CAA + AAC	0.0	15.2	4.7	10.7	3333.1	440.8
	ALL	0.1	14.9	4.5	10.7	3330.5	441.3

Table 5: Results with $s-t$ cut based cutting planes on 1500-node instances.

Class	Settings	Inequalities				BBN	Time
		stcp	aca	caa	aac		
G9	STCP	192.6	0.0	0.0	0.0	1541.0	385.4
	ACA	0.0	676.1	0.0	0.0	1421.4	373.7
	STCP + ACA	201.0	468.8	0.0	0.0	1474.5	377.8
	ACA + CAA + AAC	0.0	509.4	272.9	299.8	1381.7	369.6
	ALL	163.3	346.1	171.3	229.5	1445.3	374.9
G10	STCP	318.0	0.0	0.0	0.0	2007.5	520.8
	ACA	0.0	408.5	0.0	0.0	1845.7	487.4
	STCP + ACA	297.0	394.5	0.0	0.0	1971.7	515.6
	ACA + CAA + AAC	0.0	342.9	175.8	228.7	1824.0	483.3
	ALL	290.1	319.5	178.5	199.5	1930.4	509.5
G11	STCP	14.7	0.0	0.0	0.0	3972.9	1078.2
	ACA	0.0	623.6	0.0	0.0	3739.9	1060.8
	STCP + ACA	15.5	644.6	0.0	0.0	3753.4	1056.4
	ACA + CAA + AAC	0.0	537.7	232.6	389.6	3810.8	1076.5
	ALL	16.1	559.9	206.1	434.2	3886.8	1077.3
G12	STCP	0.3	0.0	0.0	0.0	4326.4	1037.3
	ACA	0.0	46.2	0.0	0.0	4341.6	1093.0
	STCP + ACA	0.3	46.2	0.0	0.0	4341.6	1089.7
	ACA + CAA + AAC	0.0	44.9	14.9	35.5	4350.6	1117.8
	ALL	0.4	44.9	14.8	35.6	4349.0	1121.5

plane of his own to the problem (i.e., `XPRS_CUTSTRATEGY` was set to 0). Moreover, in these experiments we gave the optimal solution value to the solver (i.e., `XPRS_MIPABSCUTOFF` was set to the optimal value).

The summary of the experiments can be found in Table 6, 7 and 8, respectively; whereas the detailed computations are provided in Table A6 of the Supplementary Material. In Tables 6, 7 and 8 we only indicated the number of the investigated branch-and-bound nodes (BBN), the total running time (Time) of the branch-and-cut procedure in seconds, and the number of generated cuts (Inequalities/spp and Inequalities/sr).

The SPP setting refers to the use of the strengthened subpath precedence inequalities (11) and (12). We generated these inequalities for an arc e only if the solution value on the arc was positive (i.e., $x_e^* > 0$). The SR setting refers to the use of the infeasible subpath based inequalities (19) and (20). The SPP + SR setting refer to the simultaneous use of all the inequalities in the same experiment. Cuts were generated in each node with depth at most 8 in one round, except the root node where we separate inequalities in 20 rounds.

We can observe that in almost all cases SR or SPP+SR proved to be the most effective method, except on the instances in class G12, where SPP was the winner both in computation time and number of branch-and-bound nodes explored. The reason for this is that the SR cuts could be generated in a much greater number than the SPP cuts. Notice also that when both types of cuts are generated, then the total number of SR and SPP cuts is about the number

Table 6: Results with infeasible subpath based cutting planes on 500-node instances.

		Inequalities			
Class	Settings	spp	sr	BBN	Time
G1	SPP	591.9	0.0	1249.2	46.7
	SR	0.0	1031.8	1094.2	39.2
	SPP + SR	561.4	819.6	1261.4	47.1
G2	SPP	697.1	0.0	1537.2	65.2
	SR	0.0	1140.3	1480.3	62.9
	SPP + SR	712.5	979.1	1539.9	61.5
G3	SPP	970.7	0.0	2845.1	111.3
	SR	0.0	3212.3	2917.2	107.8
	SPP + SR	908.6	2704.1	2809.3	102.7
G4	SPP	951.6	0.0	2650.2	95.3
	SR	0.0	3828.5	2428.4	87.7
	SPP + SR	809.5	3607.9	2694.3	96.7

Table 7: Results with infeasible subpath based cutting planes on 1000-node instances.

		Inequalities			
Class	Settings	spp	sr	BBN	Time
G5	SPP	759.8	0.0	1040.2	128.4
	SR	0.0	1545.0	1001.3	125.1
	SPP + SR	715.0	1113.6	1024.5	122.3
G6	SPP	725.8	0.0	2261.6	288.7
	SR	0.0	1134.2	2019.2	272.5
	SPP + SR	697.0	729.9	2188.0	283.0
G7	SPP	1032.2	0.0	3568.1	481.8
	SR	0.0	3968.5	3320.7	444.0
	SPP + SR	983.3	2808.7	3483.2	459.7
G8	SPP	1146.3	0.0	3248.2	399.9
	SR	0.0	5138.4	3123.7	378.9
	SPP + SR	1058.7	4513.1	3237.6	390.9

of SR cuts in the pure SR case.

7.6. Combined experiments

In the experiments presented below we combined the various components to find the best way of using them together for solving hard instances. We report only on the most successful combinations.

The detailed results of the experiments can be found in Table A7 of the Supplementary Material, and are summarized in Table 9, which contains the average results of our tests on each instance class. In these tables we only indicate the number of the investigated branch-and-bound nodes (BBN) and the total running time (Time) of the branch-and-cut procedure in seconds. The XPRS setting refers to the pure use of Xpress with settings `XPRS_CUTSTRATEGY`

Table 8: Results with infeasible subpath based cutting planes on 1500-node instances.

		Inequalities			
Class	Settings	spp	sr	BBN	Time
G9	SPP	711.7	0.0	1458.2	374.8
	SR	0.0	1839.8	1420.2	374.1
	SPP + SR	702.4	1197.1	1474.9	370.4
G10	SPP	759.6	0.0	1977.7	506.4
	SR	0.0	977.8	1815.8	487.0
	SPP + SR	705.6	658.2	1940.8	497.5
G11	SPP	1084.5	0.0	3907.9	1046.4
	SR	0.0	4436.8	3673.2	1000.0
	SPP + SR	1104.5	3024.8	3848.7	1040.7
G12	SPP	1379.1	0.0	4392.6	1055.8
	SR	0.0	5964.9	4403.2	1076.3
	SPP + SR	1263.6	5715.6	4439.7	1071.6

= -1 (generation of built-in cutting planes), `XPRS_HEURSTRATEGY` = -1 (use built-in heuristics), `XPRS_PRESOLVE` = 0 and `XPRS_MIPPRESOLVE` = 0 (no presolves). The GC and NC settings refer to the use of our branch-and-cut method without the cutting planes of Xpress (`XPRS_CUTSTRATEGY` = 0), without Xpress heuristics (`XPRS_HEURSTRATEGY` = 0), and without presolves (`XPRS_PRESOLVE` = 0, `XPRS_MIPPRESOLVE` = 0). The difference between these settings is that in case GC we use inequalities STCP and SPP, however in case NC we use inequalities STCP, AAC, ACA, CAA, SR. Finally, the NC+XPRS setting refers to our branch-and-cut method with the same inequalities as in NC, along with Xpress cutting planes (`XPRS_CUTSTRATEGY` = -1), heuristics (`XPRS_HEURSTRATEGY` = -1), but without presolves (`XPRS_PRESOLVE` = 0, `XPRS_MIPPRESOLVE` = 0).

We can observe that for all problem sizes, and all types of resource limits, we obtained the best results (in term of solving time) either by the GC or by the NC setting.

7.7. Conclusion

The above tests suggest that our primal heuristic and mainly our variable fixing method can significantly reduce the execution time of a branch-and-bound procedure (see Section 7.3). We can also see that both of our cutting planes and the cutting planes from literature (Section 7.4 and 7.5) can reduce the computation times and the number of branch-and-bound nodes of a plain branch-and-bound procedure (cf. PLAIN method in Table 2). However, if we compare the results of the combined experiments (see Section 7.6) with the results of the heuristic and variable fixing experiments (see Section 7.3), we can conclude that adding cutting planes on top of heuristic and variable fixing methods does not improve, and in most cases degrades the overall performance.

Table 9: Summary of combined experiments.

Setting	Class	BBN	Time	Class	BBN	Time	Class	BBN	Time
XPRS	G1	1744.3	78.0	G5	1450.4	226.7	G9	2215.9	653.4
GC		990.1	22.6		904.3	55.7		1142.3	104.3
NC		816.0	19.8		886.2	56.5		991.3	96.3
NC+XPRS		1194.9	28.5		1040.9	76.4		1337.5	137.6
XPRS	G2	2241.1	106.2	G6	2720.8	447.2	G10	2664.7	826.7
GC		1370.6	34.4		1262.0	71.1		994.5	84.6
NC		1283.7	33.6		1087.8	65.7		921.8	85.1
NC+XPRS		1617.7	45.6		1489.8	89.2		1112.9	108.6
XPRS	G3	4578.3	177.8	G7	4961.6	768.3	G11	5508.8	1515.8
GC		3587.6	81.5		3575.7	219.9		4290.4	430.3
NC		3362.5	83.5		3623.9	238.8		4156.6	415.5
NC+XPRS		4317.0	109.3		4345.0	326.6		4502.6	541.0
XPRS	G4	4341.8	169.0	G8	5021.0	670.9	G12	5777.3	1689.0
GC		3133.3	66.2		3859.2	207.4		4830.0	429.2
NC		3124.3	82.7		3771.6	239.7		5372.3	530.2
NC+XPRS		4172.3	117.3		4697.7	320.2		5280.8	620.1

8. Comparison with state-of-the-art methods

In this section we compare our branch-and-cut approach with other approaches from the literature, namely the Reference Point Method of Pugliese and Guerriero [18], and the Pulse Algorithm of Lozano and Medaglia [15].

The solution method of Pugliese and Guerriero [18] have been implemented in Java programming language and tested by using a PC with Intel Core i7-620M, 2.67 GHz CPU, under Windows 7; while the solution approach of Lozano and Medaglia [15] have been implemented in Java programming language and tested by using a PC with Intel Core 2 Duo P8600, 2.4 GHz CPU, under Windows XP. Since these environments differ from each other and from ours (as we describe it next), we do not intend to directly compare the running times of the different procedures, but we want to investigate how they behave on different sets of instances.

8.1. Test environment and implementation

All of our computations were performed on a notebook with Intel Core i7-4710MQ, 2.5 GHz CPU, under Windows 7. Our procedure has been implemented in C++ programming language using the FICO Xpress Optimization Suite (version 28.01.04) [7] (Xpress) as a branch-and-cut framework; and the LEMON C++ library (version 1.3.1) [14] (Lemon) to handle graphs and to perform graph algorithms.

8.2. Instances

For these experiments we used well-known instance sets from the literature, see Table 10. The columns of the table depict (in this order) the name of the instance set, the number of resources, the minimum and the maximum number

Table 10: Properties of instance sets

	Instance set	Res	Nodes Min	Nodes Max	Arcs Min	Arcs Max
D1	Dumitrescu and Boland [6]	1	10 002	135 002	29 900	404 850
D2	Dumitrescu and Boland [6]	1	625	40 000	2 400	159 200
S	Santos et al. [19]	1	10 000	40 000	15 000	800 000

Table 11: Average results on instance sets D1, D2 and S

		Preprocessing		Xpress	
Set	Instances	Time	Solved	BBN	Time
D1	8	0.38	0	1.0	179.28
D2	56	0.12	1	3.0	22.96
S	880	0.12	878	2.0	160.71

of nodes, and the minimum and maximum number of arcs, respectively, over the instances in the set.

8.3. Experiments

In these experiments, first, we applied DB-preprocessing. If the preprocessing found an optimal solution or proved that problem is infeasible we stopped, otherwise we invoked Xpress to solve the preprocessed problem instance.

The results on all these instances are summarized in Table 11, which contains one row per instance class. The columns depict (in this order), the name of the instance set, the number of instances in the set, the average time (in seconds) of preprocessing over all instances in the set, and the number of instances solved optimally by just applying preprocessing and no Xpress. Further on, for those instances not solved to optimality by preprocessing, the columns BBN and Time provide the number of branch-and-bound nodes and computation time (in seconds) needed by Xpress to find optimal solutions.

8.3.1. Experiments on instance sets D1 and D2

Instance sets D1 and D2 were developed by Dumitrescu and Boland [6] and also used by Pugliese and Guerriero [18]. The DB-preprocessing procedure could solve only one of these instances, but the preprocessed problems were not difficult to solve with a branch-and-cut method. We considered two scenarios. In the first case (Method A) we allowed Xpress to perform presolves (i.e., `XPRS_PRESOLVE` and `XPRS_MIPPRESOLVE` were set to -1, however `XPRS_ROOTPRESOLVE` was set to 0) while in the second case (Method B) we forbade Xpress to perform presolves (i.e., the previous parameters were set to 0). In both cases we allowed Xpress's cuts and heuristics (i.e., `XPRS_CUTSTRATEGY`

Table 12: Computation times (in seconds) on instance set D1

	RPM[1]		Method A [2]		Method B [2]	
Instance	LC	ISSA	PP	BNC	PP	BNC
D1-L\1	0.02	2.50	0.02	0.48	0.02	1.38
D1-L\2	3.68	2 324.59	0.23	64.34	0.23	69.55
D1-L\3	3.62	1 330.74	0.42	65.40	0.42	71.50
D1-L\4	21.89	21 926.07	1.31	719.84	1.31	707.29
D1-M\1	3.60	11.31	0.01	2.04	0.01	3.90
D1-M\2	96.92	564.33	0.13	47.66	0.13	51.54
D1-M\3	377.97	1 180.05	0.25	100.77	0.25	109.78
D1-M\4	1 849.89	15 635.17	0.70	403.16	0.70	490.15

[1]: tested with Intel Core i7-620M, 2.67 GHz CPU, under Windows 7

[2]: tested with Intel Core i7-4710MQ, 2.5 GHz CPU, under Windows 7

and `XPRS_HEURSTRATEGY` were set to -1), and we applied DB-preprocessing procedure as a variable fixing method in each branch-and-bound tree node.

For detailed results we refer to Appendix D of the Supplementary Material, however, we summarize our results in Tables 12 and 13 where we indicate execution times in seconds. Columns under RPM refer to the Reference Point Method, where the total running time is the sum of two parts: the Label Correcting method (LC) and the Interactive Search Strategy Algorithm (ISSA). The columns under Method A and Method B refer to our branch-and-cut methods, consisting of two parts: preprocessing (PP) and solving (if needed) by branch-and-cut (BNC).

We recall that the solution approaches have been tested on different platforms, thus we do not recommend a direct comparison of running times. However, we can observe that our branch-and-cut procedure behaves in a more stable way than the Reference Point Method, since for the latter the running time grows more rapidly with the size of the instances.

For example, for set D1-L the running time of the RMP on the largest instance is more than 8700 times larger than the execution time on the smallest instance, however, in case of our branch-and-cut procedures the difference is about 1450 (Method A) and 500 (Method B), respectively. For set D1-M these values are 1172.7 (RPM) against 197 (Method A) and 125.5 (Method B), respectively.

8.3.2. Experiments on instance set S

Instance set S was developed by Santos et al. [19] and also used by Pugliese and Guerriero [18], and Lozano and Medaglia [15]. Originally, this problem set contains 900 instances, however, we have got only 880 of them. All of the instances have a single resource constraint and can be classified into 18 classes according to their sizes (S1-S18) or into 5 groups according to their resource types (group 1-group 5). For details we refer to [19].

Table 13: Computation times (in seconds) on instance set D2

	RPM[1]		Method A [2]		Method B [2]	
Set	LC	ISSA	PP	BNC	PP	BNC
D2-L-1	0.00	0.00	0.00	0.01	0.00	0.01
D2-L-2	0.02	0.04	0.01	0.24	0.01	0.73
D2-L-3	0.04	0.17	0.02	0.73	0.02	1.60
D2-L-4	0.33	1.36	0.07	3.10	0.07	4.34
D2-L-5	0.72	2.96	0.15	8.98	0.15	24.23
D2-L-6	1.01	10.16	0.20	32.07	0.20	43.82
D2-L-7	2.09	52.60	0.36	59.65	0.36	79.10
D2-M-1	0.01	0.04	0.00	0.04	0.00	0.08
D2-M-2	0.18	0.27	0.01	0.33	0.01	0.57
D2-M-3	0.62	4.65	0.02	2.99	0.02	4.15
D2-M-4	9.30	16.27	0.09	11.82	0.09	15.51
D2-M-5	36.60	89.46	0.14	19.42	0.14	28.17
D2-M-6	63.26	475.64	0.19	42.95	0.19	46.15
D2-M-7	243.91	487.75	0.28	104.79	0.28	93.55

[1]: tested with Intel Core i7-620M, 2.67 GHz CPU, under Windows 7

[2]: tested with Intel Core i7-4710MQ, 2.5 GHz CPU, under Windows 7

We found that all but two instances can be solved to optimality by just applying preprocessing in a split of second. For detailed results we refer to Appendix S. In Table 14 we summarize the results of mentioned approaches, namely the Reference Point Method (RPM), the Pulse Algorithm (PA) and our branch-and-cut procedure (BNC). In each cell we indicate the average execution time in seconds on the 10 instances of the corresponding group in the corresponding set. Again, we recall that the solution approaches have been tested on different platforms, thus we do not recommend to compare the corresponding execution times with each other. However, we can consider that the pulse algorithm and the branch-and-cut procedure behave more stable way then the reference point method, that is execution times increase less by the increase of the input size. For example, for group 1 in sets S1-S6 the execution times are in the same order of magnitude in case of PA and BNC, however, in case of RPM the execution time increases more by 18 times between set S1 and set S6. For group 5 between sets S13 and S18 the running time of RPM increases more by 40 times.
We remark that for BNC we have two salient results, namely for group 2 in set S6 and set S18. These two sets contain the two instances not solved in the preprocessing phase. We note that Pugliese and Guerriero [18] also confirmed the efficiency of the DB-preprocessing procedure on these instances, but our computational times are orders of magnitude better than theirs. A possible reason for their high computation times is that they implemented the methods in Java (while we used C++), and on the other hand we used very efficient graph structures and shortest path algorithms provided by LEMON.

Table 14: Computation times (in seconds) on instance set S

	group 1			group 2			group 3			group 4			group 5		
Set	RPM[1]	PA[2]	BNC[3]	RPM	PA	BNC	RPM	PA	BNC	RPM	PA	BNC	RPM	PA	BNC
S1	0.88	0.01	0.01	0.81	0.01	0.01	0.81	0.01	0.01	0.80	0.01	0.01	0.84	0.01	0.02
S2	1.61	0.02	0.01	1.56	0.02	0.02	1.59	0.02	0.02	1.67	0.02	0.02	1.85	0.02	0.02
S3	4.20	0.02	0.02	3.57	0.02	0.02	3.76	0.02	0.02	4.51	0.02	0.03	5.31	0.02	0.03
S4	7.73	0.04	0.04	6.96	0.05	0.05	8.81	0.04	0.05	11.18	0.05	0.05	13.06	0.04	0.03
S5	11.38	0.07	0.06	13.47	0.06	0.09	17.49	0.07	0.09	21.10	0.07	0.06	24.48	0.06	0.04
S6	16.61	0.07	0.08	20.05	0.14	0.77	25.26	0.16	0.12	30.04	0.10	0.10	33.82	0.08	0.07
S7	3.00	0.02	0.03	3.00	0.02	0.03	3.00	0.02	0.03	2.97	0.02	0.03	2.97	0.02	0.03
S8	5.73	0.03	0.04	5.66	0.03	0.04	5.69	0.03	0.04	5.92	0.03	0.04	6.33	0.03	0.03
S9	13.45	0.06	0.04	12.39	0.06	0.04	13.83	0.06	0.04	16.92	0.06	0.04	20.22	0.06	0.04
S10	28.84	0.11	0.09	28.68	0.09	0.11	33.79	0.09	0.10	42.92	0.09	0.09	50.11	0.12	0.08
S11	43.66	0.17	0.14	49.47	0.14	0.19	65.49	0.15	0.18	78.18	0.15	0.15	89.40	0.19	0.14
S12	64.85	0.17	0.22	77.41	0.22	0.34	96.36	0.28	0.24	111.99	0.23	0.20	127.56	0.18	0.16
S13	13.39	0.05	0.06	12.09	0.05	0.05	12.50	0.05	0.05	12.81	0.05	0.07	12.99	0.05	0.06
S14	22.05	0.08	0.08	22.06	0.08	0.08	23.02	0.08	0.08	25.26	0.08	0.11	27.46	0.08	0.11
S15	53.39	0.12	0.15	49.39	0.12	0.13	57.26	0.12	0.13	74.26	0.12	0.18	88.16	0.12	0.18
S16	112.92	0.21	0.22	108.43	0.22	0.23	142.34	0.26	0.25	180.23	0.22	0.20	215.63	0.21	0.21
S17	167.53	0.29	0.36	201.34	0.34	0.49	270.36	0.31	0.38	331.30	0.31	0.39	435.46	0.31	0.36
S18	248.18	0.39	0.49	321.77	0.47	32.20	420.77	0.52	0.56	504.70	0.55	0.37	567.61	0.43	0.33

[1]: tested with Intel Core i7-620M, 2.67 GHz CPU, under Windows 7
[2]: tested with Intel Core 2 Duo P8600, 2.4 GHz CPU, under Windows XP
[3]: tested with Intel Core i7-4710MQ, 2.5 GHz CPU, under Windows 7

8.3.3. Conclusions

As we described before, we summarize our results in Table 11. From the table we can see that in instance set S, 878 out of 880 instances can be solved in a split of second using only the DB-preprocessing procedure, while in the set D, preprocessing techniques (both prior to forming the MIP, and in the course of branch-and-bound) have a major role in reducing the computation times.

9. Conclusions

In this paper we have extended previous work by Garcia [8] for solving RCSPP by branch-and-cut. We have introduced new cutting planes, new separation-, and variable fixing procedures, as well as a primal heuristic. We have thoroughly tested each of the components in separate, as well as in combined experiments. The experiments show that the new techniques can improve, sometimes significantly, the performance of a branch-and-cut type method.

10. Acknowledgments

This work has been supported by the OTKA grant K112881, and by the NFÜ grant ED_13-2-2013-0002. The research of Tamás Kis has been supported by the János Bolyai research grant BO/00412/12/3 of the Hungarian Academy of Sciences.

[1] Aneja, Y. P., Aggarwal, V. and Nair, K. P. K., Shortest chain subject to side constraints. Networks, 13 (1983) 295–302.

[2] Avella, P., Boccia, M., and Sforza, A., Resource constrained shortest path problems in path planning for fleet management, Journal of Mathematical Modeling and Algorithms, 3 (2004) 1–17.

[3] Beasley, J. and Christofides, N., An algorithm for the resource constrained shortest path problem, Networks, 19 (1989) 379–394.

[4] Desrochers, M. and Soumis, F., A generalized permanent labeling algorithm for the shortest path problem with time windows, INFOR, 26 (1988) 191–212.

[5] Dror, M., Note on the complexity of the shortest path models for column generation in VRPTW. Operations Research, 42 (1994) 977–978.

[6] Dumitrescu, I. and Boland, N., Improved preprocessing, labeling and scaling algorithms for the weight-constrained shortest path problem, Networks, 42 (2003) 135–153.

[7] FICO Xpress Optimization Suite, http://www.fico.com/en/products/fico-xpress-optimization-suite/, 2014

[8] Garcia, R., Resource Constrained Shortest Paths and Extensions, PhD thesis, Georgia Institute of Technology, 2009.

[9] Garey, M.R., Johnson, D.S., Computers and Intractability: A Guide to the Theory of NP-Completeness, New York: W.H. Freeman and Company, 1979.

[10] Handler, G. Y. and Zang, I., A dual algorithm for the constrained shortest path problem, Networks, 10 (1980) 293–309.

[11] Irnich, S. and Desaulniers, G., Shortest path problems with resource constraints, In: Desaulniers, G., Desrosiers, J., Solomon, M.M., Column Generation, Springer US, 2005.

[12] Jepsen, M., Petersen, B., Spoorendonk, S., Pisinger, D., A branch-and-cut algorithm for the capacitated profitable tour problem, Discrete Optimization, 14 (2014) 78–96.

[13] Joksch, H.C., The Shortest Route Problem with Constraints. Journal of Mathematical Analysis and Application, 14 (1966) 191–197.

[14] LEMON — Library for Efficient Modeling and Optimization in Networks, http://lemon.cs.elte.hu/, 2014.

[15] Lozano, L., Medaglia, L., On an exact method for the constrained shortest path problem, Computers and Operations Research, 40 (2013) 378–384.

[16] Mehlhorn, K. and Ziegelmann, M., Resource constrained shortest paths, in 7th Annual European Symposium on Algorithms (ESA2000), LNCS 1879, pp. 326–337, 2000.

[17] Pugliese, L., Guerriero, F., A survey of resource constrained shortest path problems: Exact solution approaches, Networks, 62 (2013) 183–200.

[18] Pugliese, L., Guerriero, F., A Reference Point Approach for the Resource Constrained Shortest Path Problems, Transportation Science, 47 (2013) 247–265.

[19] Santos, L., Coutinho-Rodrigues, J., Current, J. R., An improved solution algorithm for the constrained shortest path problem. Transportation Research Part B: Methodological, 41(7) (2007) 756–771.