Solving resource constrained shortest path problems with Branch-and-Cut

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) a shortest path satisfying additional resource bounds is sought. Each arc has a length and a resource usage vector specifying the requirements from each resource of a finite set of resources. The resource bounds limit the total usage along feasible paths for each resource type. In this paper we introduce new cutting planes, separation procedures, variable fixing methods, and a primal heuristic for solving RCSPP to optimality. We provide detailed computational experiments, and a comparison to cutting planes from 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. Given a directed graph $G = (V, E)$, where V is a finite set of nodes and $E \subseteq V \times V$ is a finite set of directed arcs, a cost function $c : E \to \mathbb{Z}$ on the arcs (negative values are allowed), two special nodes $s, t \in V$ with $s \neq t$ and such that there is a directed path from s to t in G, that is, a subgraph $P = (V_P, E_P)$ of G with $V_P = \{v_0, \ldots, v_k\} \subseteq V$, $s = v_0$, $t = v_k$, and $E_P = \bigcup_{i=1,\ldots,k}\{e_i\}$, where e_i is some arc of G directed from node v_{i-1} to node v_i. The path is *elementary* if all nodes v_0 through v_k are distinct. In the *(elementary) shortest path problem* an (elementary) $s-t$ path of smallest total cost is sought, i.e., a directed elementary path from s to t such that the total cost of the arcs on the path is the smallest, i.e.,

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

[*]Corresponding author, Tel.: +36 1 2796156

Email addresses: `marko.horvath@sztaki.mta.hu` (Markó Horváth), `tamas.kis@sztaki.mta.hu` (Tamás Kis) where $\mathcal{P}_{st}$ is the set of all directed $s-t$ paths. Now, suppose that in addition to the above problem data we also have weights assigned to the arcs, that is, to each arc the function $w : E \to \mathbb{Z}^m$ assigns an m-dimensional vector, 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 *(elementary) resource constrained shortest path problem* a directed (elementary) 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], but it is a standard assumption that the graph admits neither a cycle of negative total cost, nor one of negative total resource consumption, where a *cycle* is defined like a path except that the first and the last nodes are the same.

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.

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

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

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

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

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 usage of the paths are bounded by (5). If we need elementary paths, and the graph G contains cycles, then additional means are needed to eliminate cycles in the solution. For instance, the inequalities

$$\sum_{e \in \gamma(S)} x_e \leq |S| - 1, \quad \forall\, S \subset V, \tag{7}$$

eliminate all the cycles, where $\gamma(S)$ is the set of arcs spanned by the subset of nodes $S \subset V$, i.e., $\gamma(S) := \{e \in E \mid \mathit{head}(e), \mathit{tail}(e) \in S\}$.

The elementary RCSPP has been shown NP-hard in the strong sense for graphs containing cycles by Dror [5]. However, the problem remains NP-hard even if the graph is acyclic 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]. 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, 15]. *Node labeling* type methods are based on dynamic programming, where the nodes of the graph are labeled with the length 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 search-tree 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 generalized subtour elimination inequalities are proposed to ensure elementary paths. The authors also suggest generalized capacity inequalities, but computational experiments show that they are not effective in practice. 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.

In this paper we will pursue a polyhedral approach, and use a Branch-and-Cut method to solve the RCSPP.

Our results. We generalize some of the valid inequalities of Garcia [8] to strengthen the LP relaxation of the RCSPP. We will focus on two kinds of inequalities: $s-t$ cut precedence, and subpath precedence. We will consider various generalizations of these inequalities, and answer an open problem in [8], that is, we provide new complexity results of separating subpath precedence as well as infeasible subpath based inequalities. We will also propose a new primal heuristic for finding feasible solutions, as well as variable fixing techniques to be applied in the course of Branch-and-Cut. Our methods have been evaluated in computational experiments, and our main finding is that our generalization of the subpath precedence inequalities combined with the variable fixing and primal heuristic methods is really effective in practice for solving large-scale RCSPP instances.

Structure of the paper. The previous work which was the starting point of our research is summarized in Section 2. New valid inequalities for RCSPP along with the corresponding separation procedures are proposed in Section 3. New variable fixing methods and primal heuristics are described in Section 4. Detailed computational evaluation is provided in Section 5. We conclude the paper in Section 6.

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 (recall the definition of a directed path in the first paragraph of the Introduction). 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 $y : E \to \mathbb{R}$ of the arcs, $y(A) := \sum_{e \in A} y(e)$ is the sum of weights of those arcs in A. For a pair of nodes i, j, let σ^r_{ij} denote the length 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*. The *RCSPP polytope* is the set of binary vectors satisfying (4)-(7), noting that if the underlying graph contains no directed cycles (*acyclic* in short), then the inequalities (7) are superfluous.

2. Preliminaries and related work

In this section we recapitulate previous results that we will extend in Sections 3 and 4. The inequalities presented in this section are valid for the RCSPP polytope.

2.1. 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 some arc $e' \in \delta^{out}(head(e))$ with $\sigma^r_{s,tail(e)} + w^r_e + w^r_{e'} + \sigma^r_{head(e'),t} \leq W^r$ for each resource r, as shown by Garcia [8]. That is, 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

$F^{out}_{e,r} := \{e' \in \phi^{out}_{e,r} \mid \mathit{head}(e') \neq \mathit{tail}(e)\}$. Then the *node precedence inequality* for arc e with respect to resource r is

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

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 $\phi^{in}_{e,r} := \{e' \in \delta^{in}(\mathit{tail}(e)) \mid \sigma^r_{s,\mathit{tail}(e')} + w^r_{e'} + w^r_e + \sigma^r_{\mathit{head}(e),t} \leq W^r\}$, and $F^{in}_{e,r} := \{e' \in \phi^{in}_{e,r} \mid \mathit{tail}(e') \neq \mathit{head}(e)\}$. The resulting *reverse node precedence inequality* for arc e with respect to resource r is

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

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

2.2. $s-t$ 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 $\mathit{head}(e) \in S \setminus \{s\}$. Then any resource feasible $s-t$ path π through arc e must cross the $s-t$ cut $\delta^{out}(S)$ on some arc in $\gamma(\rho^{out}(\mathit{head}(e)))$ (the set of arcs $e' \in E$ such that $\mathit{tail}(e')$ is reachable from node $\mathit{head}(e)$ on a directed path in G). In addition, π must pass through some arc $e' \in \delta^{out}(S) \cap \gamma(\rho^{out}(\mathit{head}(e)))$ with $\sigma^r_{s,\mathit{tail}(e)} + w^r_e + \sigma^r_{\mathit{head}(e),\mathit{tail}(e')} + w^r_{e'} + \sigma^r_{\mathit{head}(e'),t} \leq W^r$ for each resource r. Again, Garcia [8] defined the set $\Phi^{out}_{e,r} = \{e' \in E \mid \sigma^r_{s,\mathit{tail}(e)} + w^r_e + \sigma^r_{\mathit{head}(e),\mathit{tail}(e')} + w^r_{e'} + \sigma^r_{\mathit{head}(e'),t} \leq W^r\}$ for each resource r. For any $S \subset V$ with $s \in S$, $t \notin S$, and $e \in E$ with $\mathit{head}(e) \in S \setminus \{s\}$, $F^{out}_{e,r}(S) = \{e' \in \delta^{out}(S) \mid e' \in \Phi^{out}_{e,r}, \mathit{tail}(e') \neq \mathit{tail}(e), \mathit{head}(e') \neq \mathit{tail}(e)\}$. Then the *$s-t$ cut precedence inequality* for resource r is

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

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 $\mathit{head}(e)-t$ cut in a graph derived from G, for details, see [8].

For $e \in E$ one can define analogously the set $\Phi^{in}_{e,r} = \{e' \in E \mid \sigma^r_{s,\mathit{tail}(e')} + w^r_{e'} + \sigma^r_{\mathit{head}(e'),\mathit{tail}(e)} + w^r_e + \sigma^r_{\mathit{head}(e),t} \leq W^r\}$, and for any $S \subset V$ with $s \in S$, $t \notin S$, and $\mathit{tail}(e) \in \overline{S} \setminus \{t\}$, $F^{in}_{e,r}(S) = \{e' \in \delta^{out}(S) \mid e' \in \Phi^{in}_{e,r}, \mathit{tail}(e') \neq \mathit{head}(e), \mathit{head}(e') \neq \mathit{head}(e)\}$. Then the *reverse $s-t$ cut precedence inequality* for resource r is

$$x_e \leq x(F^{in}_{e,r}(S)). \tag{11}$$

2.3. Subpath precedence inequalities

Let $\pi = \Big(\{i_1, \ldots, i_p\}, \bigcup_{j=1,\ldots,p-1}\{e_j\}\Big)$ be a path from node i_1 to node i_p in G such that $i_1 \neq t$, $i_p \neq s$, $i_2, \ldots, i_{p-1} \notin \{s,t\}$, and e_j being a directed arc of G from node i_j to i_{j+1}. We say that π *is an infeasible subpath with respect to resource* r if $\sigma^r_{s,i_1} + \sum_{k=1}^{p-1} w^r_{e_k} + \sigma^r_{i_p,t} > W^r$. 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 $i_2, \ldots, i_{p-1}$. Suppose π^* leaves π on the arc e' directed from node i_k of π to some node $j \neq i_{k+1}$. Since π^* is feasible for resource r, the condition $\sigma^r_{s,i_1} + \sum_{\ell=1}^{k-1} w^r_{e_\ell} + w^r_{e'} + \sigma^r_{head(e'),t} \leq W^r$ is satisfied. Using this observation, Garcia [8] has defined the sets $\phi^{out}_{\pi,r}(k) = \{e' \in \delta^{out}(i_k) \mid \sigma^r_{s,i_1} + \sum_{\ell=1}^{k-1} w^r_{e_\ell} + w^r_{e'} + \sigma^r_{head(e'),t} \leq W^r\}$, and $F^{out}_{\pi,r}(k) = \{e' \in \phi^{out}_{\pi,r}(k) \mid head(e') \neq i_{k+1}, i_1, \ldots, i_{k-1}\}$ for $k = 2, \ldots, p-1$. 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

$$x_{e_1} \leq x(F^{out}_{\pi,r}). \tag{12}$$

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 2.1.

One can define analogously the sets $\phi^{in}_{\pi,r}(k) = \{e' \in \delta^{in}(i_k) \mid \sigma^r_{s,tail(e')} + w^r_{e'} + \sum_{\ell=k}^{p-1} w^r_{e_\ell} + \sigma^r_{i_p t} \leq W^r\}$, and $F^{in}_{\pi,r}(k) = \{e' \in \phi^{in}_{\pi,r}(k) \mid tail(e') \neq i_{k-1}, i_{k+1}, \ldots, i_p\}$ for $k = 2, \ldots, p-1$. 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

$$x_{e_p} \leq x(F^{in}_{\pi,r}). \tag{13}$$

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.

2.4. Strengthening the inequalities

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

$$x_e \leq x(F^{out}_{e,r}), \tag{14}$$

or

$$x_e \leq x(F^{in}_{e,r}), \tag{15}$$

where $F^{out}_{e,r}$ is a set of arcs that lie on a directed $head(e) - t$ path, $F^{in}_{e,r}$ 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 (14). Let $B_{e,r} = \{e' \in \delta^{in}(head(e)) \mid \sigma^r_{s,tail(e')} + w^r_{e'} \geq \sigma^r_{s,tail(e)} + w^r_e\}$. Then (14) can be strengthened:

Proposition 1. *If (14) is valid for the RCSPP polytope, then so is*

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

A similar strengthening method applies to inequalities in the class (15). Define the set $A_{e,r} = \{e' \in \delta^{out}(tail(e)) \mid w^r_{e'} + \sigma^r_{head(e'),t} \geq w^r_e + \sigma^r_{head(e),t}\}$.

Proposition 2. *If (15) is valid for the RCSPP polytope, then so is*

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

Using a simple observation in case of multiple resources, one can strengthen inequalities of the forms (14) and (15) in the following way:

Proposition 3. *If (14) is valid for the RCSPP polytope for all resource $r = 1, \ldots, m$, then so is*

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

Proposition 4. *If (15) is valid for the RCSPP polytope for all resource $r = 1, \ldots, m$, then so is*

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

2.5. 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 length of the shortest $s-i$ path and length 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 search-tree 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 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.

3. New valid inequalities and separation procedures

In this section we generalize the valid inequalities of Section 2.

3.1. 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

$$\sigma^r_{s,i_1} + w^r_{e_1} + \sigma^r_{j_1,i_2} + w^r_{e_2} + \sigma^r_{j_2,t} \leq W^r \tag{16}$$

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

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

Proposition 5. *If G contains no directed path from j_2 to i_1, then the inequality (17) 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 (17) 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 (17) 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

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

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

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

And one can also define analogously the set

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

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

$$x_{e_1} + x_{e_2} - 1 \le x(F_{(e_1,e_2,\cdot),r}(S)). \tag{19}$$

Proposition 6. *If G contains no directed path from j_2 to i_1, then the inequalities (18) and (19) are valid for the RCSPP polytope.*

Now we give an example showing that a member of this new class of inequalities is violated, while all $s-t$ cut precedence inequalities (10), (11) are satisfied.

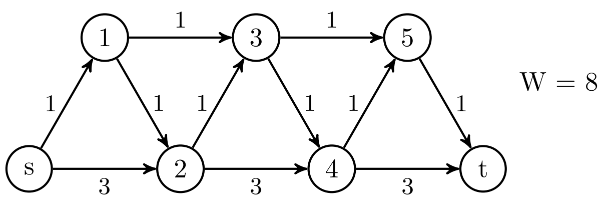

Figure 1: Network of Example 1.

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 (18)*

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

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 (17), (18), and (19) in turn. Suppose we want to find violated (17) 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 $\mathit{tail}(e) \notin \{j_1, i_2, j_2, t\}$ and $\mathit{head}(e) \notin \{j_1, i_2, j_2, t\}$, 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 (17), (18), and (19) can be separated in polynomial time.

Finally notice that the strengthening methods of Section 2.4 can be applied to (17), (18), and (19) as well.

3.2. 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 2.3), with $p \geq 4$. We will 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

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

Proposition 7. *If G contains no directed path from i_p to i_1, then the inequality (20) 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 (20) 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

$$x_{e_1} + x_{e_{p-1}} - 1 \leq x(\tilde{F}^{in}_{\pi,r}). \tag{21}$$

Proposition 8. *If G contains no directed path from i_p to i_1, then the inequality (21) is valid for the RCSPP polytope.*

The following two examples show that neither the subpath precedence (defined in Section 2.3), 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 (12) and the infeasible subpath inequalities (20):

$$\begin{aligned} \textit{subpath precedence inequality for } \pi_1 &: \; x_{s,2} \le x_{2,3} + x_{2,4} + x_{5,6} && (22) \\ \textit{subpath precedence inequality for } \pi_2 &: \; x_{s,2} \le x_{2,4} + x_{2,5} + x_{5,6} && (23) \\ \textit{infeasible subpath inequality for } \pi_1, \pi_2 &: \; x_{s,2} + x_{5,t} \le x_{2,4} + 1 && (24) \end{aligned}$$

Condition (24) excludes both π_1 and π_2, but (22) excludes only π_1 and (23) excludes only π_2.

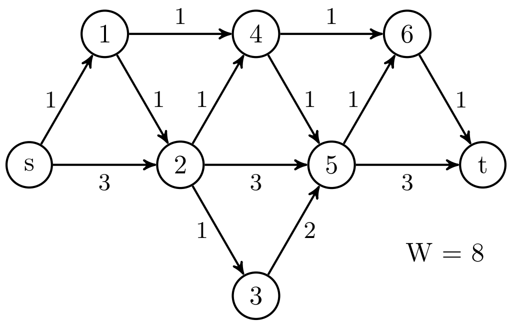

Figure 2: Network for Example 2.

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 (12) and the infeasible subpath inequalities (20):*

$$\begin{aligned} \textit{subpath precedence inequality for } \pi_1, \pi_2, \pi_3 &: \; x_{s,2} \le x_{2,3} + x_{4,5} && (25) \\ \textit{infeasible subpath inequality for } \pi_1 &: \; x_{s,2} + x_{4,t} \le x_{2,3} + 1 && (26) \\ \textit{infeasible subpath inequality for } \pi_2 &: \; x_{s,2} + x_{4,6} \le x_{2,3} + 1 && (27) \\ \textit{infeasible subpath inequality for } \pi_3 &: \; x_{s,2} + x_{6,t} \le x_{2,3} + 1 && (28) \end{aligned}$$

Condition (25) excludes all of π_1, π_2 and π_3, but (26), (27) and (28) only exclude π_1, π_2 and π_3, respectively.

The separation of the inequalities (20) and (21) is not an easy problem. Firstly, we prove that the separation of the subpath precedence inequalities (12) and (13) 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 $W \in \mathbb{R}$.

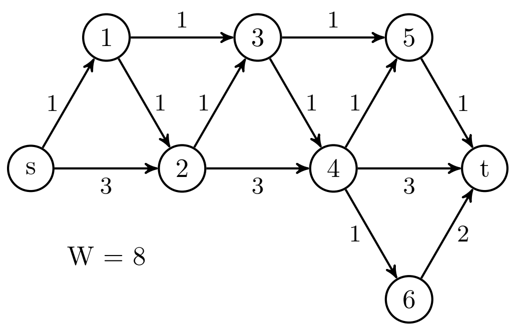

Figure 3: Network for Example 3.

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, we multiply the values $p_1, \ldots, p_n$, and P by a suitable integer to ensure that

$$\bar{c} \leq (P + p_{sum} + 1)/(P + p_{sum} + 2). \tag{29}$$

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 (29) 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 $(i, i+1)$, $i = 1, \dots, n+1$, 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 $1, \dots, n$: $e_{0,i} = (0, i)$ for $i = 1, \dots, n$, 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:

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

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 (29). 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 $\pi = (i_1, \dots, i_p)$ 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 $i = 0, \dots, n+2$, and $\sum\limits_{i=1}^{n} \max\{w(e^+_{i,i+1}), w(e^0_{i,i+1})\} = p_{sum}$, 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 $e_{1,n+2} \notin F^{out}_\pi$, 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 $i = 1, \dots, n+1$, then F^{out}_π comprises the arc $e^0_{i,i+1}$, since $w(\pi[0, i]) \leq 2p_{sum} + 1$, $w(e^0_{i,i+1}) = 0$, and $\sigma_{i+1,n+2} = 0$. On the other hand, if π contains an arc $e^0_{i,i+1}$ for some $i = 1, \dots, n$, then F^{out}_π 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 $J := \{i \in \{1, \dots, n\} \ : \ e^+_{i,i+1} \in E_\pi\}$, we have proved that

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

Thus, if π is a solution for SPP-SEP, i.e., π is an infeasible subpath and x^* violates (12), i.e., $x^*(F^{out}_\pi) < x^*(e^+_{0,1}) = \bar{c}$, then J is a solution for KP, since $p(J) = w(\pi) - w(e^+_{0,1}) - w(e^+_{n+1,n+2}) > P$, and $\bar{a}(J) = x^*(F^{out}_\pi) < \bar{c}$ if and only if $a(J) < c$.

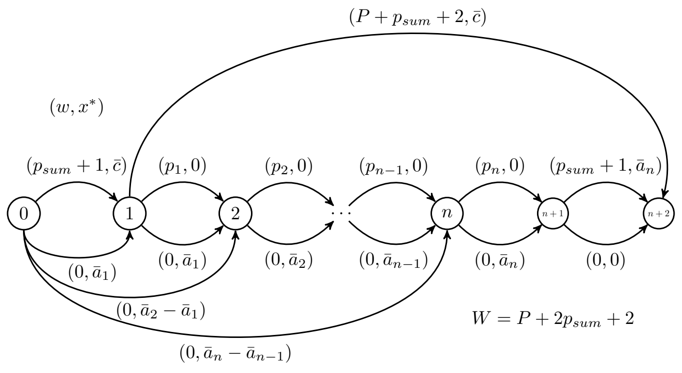

Figure 4: The constructed graph

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

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

It is easy to verify that $F^{out}_\pi = \left(\bigcup_{i \in J} \{e^0_{i,i+1}\}\right) \cup \left(\bigcup_{i \notin J} \{e^+_{i,i+1}\}\right) \cup \{e^0_{n+1,n+2}\}$, and the inequality (12) is violated by x^*. □

Now we turn to the separation problem for the inequalities (20) and (21).

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 $\mathit{head}(e_{n+2,n+3}) = n + 3$ and $\mathit{tail}(e_{n+2,n+3}) = n + 2$. 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 (20), 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 actual 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 (20) 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).

Although one can devise a similar method to separate inequalities (21) by modifying the procedure EVAL_OUT, however, it is better to combine the separation of inequalities (20) and (21), 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 (20) and (21), respectively.

4. Variable fixing and heuristics

In this section we present our primal heuristic and our variable fixing method.

4.1. DFS based feasible solution search heuristic

The aim of our primal heuristic is to find an $s-t$ path which is feasible for all resource constraints when a fractional solution x^* to the LP relaxation is available. 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. The basis of the heuristic is the observation that a fractional solution x^* to the LP relaxation is a convex combination of some $s-t$ paths, and since it respects the resource constraints, at least one of these $s-t$ paths respects as well. Thus, 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.

Algorithm 1 Heuristic for separating inequalities (20) for $x^* \in [0,1]^E$

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

Algorithm 2 DFS based feasible solution search heuristic

```
1: $U \leftarrow$ best upper bound
2: $\vartheta_s^r(s) \leftarrow 0$ and $\vartheta_t^r(t) \leftarrow 0$ for all $r$
3: $\vartheta_s^r(u) \leftarrow \infty$ for all $u \in V - s$ and $\vartheta_t^r(u) \leftarrow \infty$ for all $u \in V - t$ for all $r$
4: $cost_s(s) \leftarrow 0, cost_s(u) \leftarrow \infty$ for all $u \in V - s$
5: $cost_t(t) \leftarrow 0, cost_t(u) \leftarrow \infty$ for all $u \in V - t$
6: $processed(u) \leftarrow false$ for all $u \in V$
7: $path(u) \leftarrow false$ for all $u \in V$
8: insert $s$ into an empty stack $S$
9: while $S \neq \emptyset$ do
10:   $u \leftarrow S.top()$
11:   if $processed(u) = true$ then
12:     update $U$
13:     $path(u) \leftarrow false$
14:     $S.pop()$
15:   else
16:     $path(u) \leftarrow true$
17:     for $e \in \delta^{out}(u) \cap E^*$ do
18:       $v \leftarrow head(e)$
19:       if $path(v) = false$ then
20:         if $processed(v) = false$ then
21:           $cost_s(v) = cost_s(u) + c(e)$
22:           $\vartheta_s^r(v) \leftarrow \vartheta_s^r(u) + w_e^r$ for all $r$
23:           insert $v$ into $S$
24:         else
25:           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
26:             $cost_t(u) \leftarrow c(e) + cost_t(v)$
27:             $\vartheta_t^r(u) \leftarrow w_e^r + \vartheta_t^r(v)$ for all $r$
28:           end if
29:           update $U$
30:         end if
31:       end if
32:     end for
33:     $processed(u) \leftarrow true$
34:   end if
35: end while
```

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 $cost_s(u) + cost_t(u)$.

4.2. Cost based variable fixing

The basic idea of the method is that if all $s - t$ paths through an arc e have bigger total cost than the actual upper bound on the optimal path length, then e cannot appear in an optimal $s - t$ path, hence can be eliminated from the graph. This simple observation has also appeared in the Dumitrescu-Boland preprocessing scheme (c.f. Section 2.5), but it can be applied during the Branch-and-Cut procedure.

Once a fractional solution x^* is available at the current Branch-and-Cut node, we create a subgraph $\bar{G}$ of G as follows. In the beginning $\bar{G}$ is identical to G. If an arc e (i.e., the corresponding variable x_e) is previously fixed to 0, we delete e from $\bar{G}$. If an arc e (i.e., the corresponding variable x_e) is previously fixed to 1, we delete all arcs e' from $\bar{G}$ such that there is no $s - t$ path consisting of both e and e'. Finally, for all nodes u we calculate the length of the shortest $s - u$ and of the shortest $u - t$ path in $\bar{G}$ according to the cost function c. We denote these values $\bar{\sigma}^c_{su}$ and $\bar{\sigma}^c_{ut}$, respectively. If for an arc e $\bar{\sigma}^c_{s,tail(e)} + c(e) + \bar{\sigma}^c_{head(e),t} > U$ holds – where U is the best upper bound on the optimum found so far – we fix the arc e (i.e., the corresponding variable x_e) to 0.

5. Computational results

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, and to

- to find the best combination of the various techniques for solving hard instances.

In the following sections we will address the above points in turn, but before we sketch our computational framework.

5.1. 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. All experiments were run using a single thread only.

Our Branch-and-Cut solver has been implemented in C++ programming language. To handle graphs and to perform some graph algorithms we used the LEMON C++ library (version 1.3.1) [14]. We also used the well-known FICO XPRESS [7] (version 7.4) callable library as a Branch-and-Cut framework.

In all experiments we applied the preprocessing scheme of Dumitrescu and Boland, and the preprocessing scheme of Aneja et al. on graph $G([i])$ for each node i. Moreover, once the Dumitrescu and Boland preprocessing returned an upper bound on the optimum, we used that as an upper bound in the Branch-and-Cut method.

5.2. Instances

Our RCSPP instances were randomly generated. To construct their underlying directed graph we used a method similar to that in [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 $j = i+1, \ldots, \min\{n, i + \lfloor n/4 \rfloor\}$ 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 Garcia [8].

We generated 20 graphs for each $n \in \{500, 1000, 1500\}$ with 10 resource functions and corresponding resource limits in every instance. For each graph we determined a resource function, 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.

Table 1: Summary of the problem instances.

Class	n	m	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 ($W = 40$)

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 (n), the number of resource functions (m), and the resource limit generating method (*Method*). Tables A1, A2 and A3 of the Supplementary Material contain more informations about the instances.

5.3. Heuristic and variable fixing experiments

In this section we present the results of the experiments of heuristic and variable fixing methods described in Section 4. 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 DEFAULT settings.

As described in Section 4 we used the variable fixing method (VARFIX) in every Branch-and-Bound node before solving the LP relaxation. The heuristic solution search method (HEURSOL) were used in every search-tree node, however, the variable fixing method of Garcia (ARCFIX) were used only in the root node. Both of them were applied after an optimal solution for the LP relaxation had been found. On each instance every method was tested separately (the corresponding rows of the following tables are DEFAULT, ARCFIX, VARFIX, and HEURSOL), and all together, which is the ALL method.

The summary of the experiments can be found in Tables 2, 3 and 4, respectively; whereas the detailed computations are provided in Table A4 of the Supplementary Material. In Tables 2, 3 and 4 we only indicate the average number of the investigated Branch-and-Bound nodes (*#BnBn*) and the average running time (t_{total}) of the entire Branch-and-Bound procedure in seconds.

We can observe that for all problem sizes, and all types of resource limits, we obtained the best results by the ALL method, that is, when we combined all

Table 2: Summary of fixing and heuristic experiments for 500 nodes instances.

Class	Settings	#BnBn	t_{total}
G1	DEFAULT	1438.5	60.1
	ARCFIX	1220.8	53.4
	VARFIX	1517.2	48.4
	HEURSOL	1259.7	51.0
	ALL	1142.6	38.8
G2	DEFAULT	1883.6	87.8
	ARCFIX	1982.6	90.8
	VARFIX	2057.7	75.6
	HEURSOL	1769.3	79.4
	ALL	1760.1	64.5
G3	DEFAULT	4275.3	180.9
	ARCFIX	4275.3	181.5
	VARFIX	3977.0	121.3
	HEURSOL	3302.3	132.0
	ALL	3212.3	96.3
G4	DEFAULT	3699.0	139.6
	ARCFIX	3699.0	139.9
	VARFIX	3553.3	97.0
	HEURSOL	2740.8	99.4
	ALL	2818.8	77.3

our variable fixing, arc fixing, and heuristic search method. It is also important to note that VARFIX is the second best in almost all cases.

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

In this section we summarize the experiments with inequalities described in Section 2.2 and Section 3.1. 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 5, 6 and 7, respectively; whereas the detailed computations are provided in Table A5 of the Supplementary Material. In Tables 5, 6 and 7 we only indicated the average number of the investigated Branch-and-Bound nodes (*#BnBn*), the average total running time (t_{total}) of the Branch-and-Cut procedure in seconds, and the average number of generated cuts (*#stcp*, *#aca*, *#aac* and *#caa*)

The STCP setting refers to the use of the strengthened $s-t$ cut precedence inequalities (10) and (11). 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 (17), (18) and (19), 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

Table 3: Summary of fixing and heuristic experiments for 1000 nodes instances.

Class	Settings	#BnBn	t_{total}
G5	DEFAULT	1499.2	225.9
	ARCFIX	1328.2	204.9
	VARFIX	1502.3	174.0
	HEURSOL	1224.0	169.4
	ALL	1155.6	133.5
G6	DEFAULT	2621.3	379.5
	ARCFIX	2464.7	366.6
	VARFIX	2649.5	302.0
	HEURSOL	2309.4	325.7
	ALL	2140.4	234.0
G7	DEFAULT	4877.5	709.4
	ARCFIX	4877.5	709.7
	VARFIX	4924.7	546.9
	HEURSOL	4411.2	608.3
	ALL	3961.2	419.6
G8	DEFAULT	4222.5	571.9
	ARCFIX	4222.5	570.4
	VARFIX	4512.2	448.0
	HEURSOL	3656.1	457.8
	ALL	3833.7	362.9

Table 4: Summary of fixing and heuristic experiments for 1500 nodes instances.

Class	Settings	#BnBn	t_{total}
G9	DEFAULT	2165.9	634.6
	ARCFIX	1884.0	547.9
	VARFIX	2091.6	450.4
	HEURSOL	1841.0	493.8
	ALL	1870.4	379.8
G10	DEFAULT	2308.4	693.3
	ARCFIX	1932.0	631.7
	VARFIX	2437.3	564.0
	HEURSOL	2091.9	589.7
	ALL	1545.9	358.6
G11	DEFAULT	5333.7	1654.7
	ARCFIX	5333.7	1656.1
	VARFIX	5917.3	1351.7
	HEURSOL	4637.1	1334.1
	ALL	4481.0	1000.0
G12	DEFAULT	6044.6	1671.8
	ARCFIX	6044.6	1675.8
	VARFIX	6091.1	1223.2
	HEURSOL	5087.6	1283.5
	ALL	4893.4	938.2

Table 5: Results with $s - t$ cut based cutting planes on 500 nodes instances.

Class	Settings	#stcp	#aca	#aac	#caa	#BnBn	t_{total}
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

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 (17), (18) and (19) is more expensive in total than the generation of the $s - t$ cut precedence inequalities.

5.5. 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 2.3 and Section 3.2. 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 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 8, 9 and 10, respectively; whereas the detailed computations are provided in Table A6 of the

Table 6: Results with $s-t$ cut based cutting planes on 1000 nodes instances.

Class	Settings	#stcp	#aca	#aac	#caa	#BnBn	t_{total}
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 7: Results with $s-t$ cut based cutting planes on 1500 nodes instances.

Class	Settings	#stcp	#aca	#aac	#caa	#BnBn	t_{total}
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

Table 8: Results with infeasible subpath based cutting planes on 500 nodes instances.

Class	Settings	#spp	#sr	#BnBn	t_{total}
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 9: Results with infeasible subpath based cutting planes on 1000 nodes instances.

Class	Settings	#spp	#sr	#BnBn	t_{total}
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

Supplementary Material. In Tables 8, 9 and 10 we only indicated the number of the investigated Branch-and-Bound nodes (*#BnBn*), the total running time (t_{total}) of the Branch-and-Cut procedure in seconds, and the number of generated cuts (*#spp* and *#sr*).

The SPP setting refers to the use of the strengthened subpath precedence inequalities (12) and (13). 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 (20) and (21). 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 the most effective method, except on the instances in class G12, where SPP was the winner both in computation time and number of search-tree 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 of SR cuts in the pure SR case.

Table 10: Results with infeasible subpath based cutting planes on 1500 nodes instances.

Class	Settings	#spp	#sr	#BnBn	t_{total}
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

5.6. 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 11, Table 12 and Table 13, respectively. Tables 11, 12 and 13 contain the average results of our tests corresponding to the instance classes. In these tables we only indicated the number of the investigated Branch-and-Bound nodes (*#BnBn*) and the total running time (t_{total}) of the Branch-and-Cut procedure in seconds. The XPRS setting refers to the pure use of XPRESS with settings `XPRS_CUTSTRATEGY` = -1, `XPRS_HEURSTRATEGY` = -1, `XPRS_PRESOLVE` = 0, `XPRS_MIPPRESOLVE` = 0. The OUR refers to our Branch-and-Cut method with inequalities STCP, AAC, ACA, CAA, SR, and without the cutting planes of XPRESS (`XPRS_CUTSTRATEGY` = 0), no XPRESS heuristics (`XPRS_HEURSTRATEGY` = 0), and presolve (`XPRS_PRESOLVE` = 0, `XPRS_MIPPRESOLVE` = 0). The OUR + XPRS refers to our Branch-and-Cut method with the same inequalities as in OUR, along with XPRESS cutting planes (`XPRS_CUTSTRATEGY` = -1), heuristics (`XPRS_HEURSTRATEGY` = -1), and no presolve (`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 by the OUR method, and for all cases the OUR + XPRS method yielded the second best results.

6. 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.

Table 11: Summary of combined experiments for 500 nodes instances

Class	Settings	#BnBn	t_{total}
G1	XPRS	1842.4	81.2
	OUR	983.0	38.0
	OUR + XPRS	1322.0	57.3
G2	XPRS	2241.1	104.0
	OUR	1648.5	62.7
	OUR + XPRS	2159.0	83.2
G3	XPRS	4578.3	175.7
	OUR	3424.6	114.8
	OUR + XPRS	4749.5	146.0
G4	XPRS	4341.8	166.4
	OUR	2943.2	110.2
	OUR + XPRS	4163.1	150.1

Table 12: Summary of combined experiments for 1000 nodes instances

Class	Settings	#BnBn	t_{total}
G5	XPRS	1450.4	223.2
	OUR	1053.8	128.8
	OUR + XPRS	1206.1	161.1
G6	XPRS	2720.8	439.3
	OUR	2056.1	240.8
	OUR + XPRS	2384.5	266.7
G7	XPRS	4961.6	755.3
	OUR	3641.4	415.8
	OUR + XPRS	4849.7	592.1
G8	XPRS	5021.0	657.0
	OUR	3814.8	410.8
	OUR + XPRS	4669.6	516.3

Table 13: Summary of combined experiments for 1500 nodes instances

Class	Settings	#BnBn	t_{total}
G9	XPRS	2215.9	637.5
	OUR	1503.5	323.1
	OUR + XPRS	1601.8	412.0
G10	XPRS	2664.7	812.7
	OUR	1208.6	334.0
	OUR + XPRS	1222.7	356.3
G11	XPRS	5508.8	1701.6
	OUR	4513.8	1011.9
	OUR + XPRS	4973.1	1160.8
G12	XPRS	5777.3	1655.3
	OUR	4976.2	1035.2
	OUR + XPRS	5640.3	1291.2

7. 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 Bólyai research grant BO/00412/12/3 of the Hungarian Academy of Sciences.

References

[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., A Branch-and-Cut Algorithm for the Elementary Shortest Path Problem with a Capacity Constraint, Technical report, DIKU, 2008.

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