Datalogisk Institut
KØBENHAVNS UNIVERSITET

A Branch-and-Cut Algorithm for the Elementary Shortest Path Problem with a Capacity Constraint

Mads Kehlet Jepsen, Bjørn Petersen, Simon Spoorendonk

Technical Report no. 08/01
ISSN: 0107-8283

Dept. of Computer Science
University of Copenhagen • Universitetsparken 1
DK-2100 Copenhagen • Denmark

A Branch-and-Cut Algorithm for the Elementary Shortest Path Problem with a Capacity Constraint

Mads Kehlet Jepsen · Bjørn Petersen · Simon Spoorendonk
mitzi@diku.dk bjorn@diku.dk spooren@diku.dk
DIKU Department of Computer Science, University of Copenhagen
Universitetsparken 1, DK-2100 Copenhagen Ø, Denmark

March 7, 2008

Abstract

This paper introduces a branch-and-cut algorithm for the elementary shortest path problem with a capacity constraint which appears as a subproblem in several column generation based algorithms, e.g., in the classical Dantzig-Wolfe decomposition of the capacitated vehicle routing problem. A mathematical model and valid inequalities are presented. Furthermore, a new family of inequalities denoted the generalized capacity inequalities are introduced. Experimental results are performed on a set of benchmark instances generated from well known benchmark instances for the capacitated vehicle routing problem. Until now, label algorithms have been the dominant solution method for this problem but experimental results show that the branch-and-cut algorithm clearly outperforms on all the generated instances. Secondly, it can be concluded that although the generalized capacity inequalities drastically improves the lower bound the high separation time makes their usefulness questionable in their current form.

Keywords: Branch-and-Cut, Elementary Shortest Path Problem with Resource Constraints, Capacitated Vehicle Routing Problem

1 Introduction

The elementary shortest path problem with a capacity constraint (ESPPCC) can be stated as: Given an undirected graph $G(V,E)$ with nodes V and edges E, a cost c_e associated to each edge $e \in E$, a load d_i associated to each node $i \in V$, an upper limit on the amount of accumulated load Q, a source node $s \in V$, and a target node $t \in V$; find a path between s and t with minimum cost satisfying that the sum of the loads from the visited nodes is not more than Q.

The ESPPCC is a special case of the elementary shortest path problem with resource constraints (ESPPRC) and is known to be strongly $\mathcal{NP}$-Hard, see Dror (1994). To our knowledge the first solution method proposed for the ESPPRC was a lagrangian relaxation method by Beasley and Christofides (1989). Later Carlyle et al. (2005) used the same approach and obtained good results when edge costs are non-negative.

A label algorithm has been developed by Dumitrescu (2002) and Feillet et al. (2004). Based on the idea of Dijkstra's bi-directional shortest path algorithm the label algorithm was improved by Righini and Salani (2006) by expanding a path both forward and backward from the depot and connecting them in the middle thereby potentially reducing the running time. Furthermore Boland et al. (2006)

and Righini and Salani (2007) have independently developed a label algorithm that solves the ESPPRC by iteratively solving the non-elementary version and increasingly constraining the number of visits to the customers. In the latter paper this method is referred to as a decremental state-space relaxation. The non-elementary problem has pseudo-polynomial complexity and can be solved with label algorithms, see Desrosiers et al. (1984), Desrochers et al. (1992), Irnich and Villeneuve (2006). The state-space relaxation has also been used to calculate lower bounds in a branch-and-bound algorithm by Righini and Salani (2007).

The ESPPRC appears as a subproblem in many column generation based algorithms, particularly within routing problems. Feillet et al. (2004) were the first to use a label algorithm for ESPPRC in a vehicle routing context, more precisely for the vehicle routing problem with time windows (VRPTW). Later label algorithms for the ESPPRC have been used by Chabrier (2005), Danna and Pape (2005), Salani (2005), Jepsen et al. (2007) to successfully solve until then unsolved VRPTW instances from the benchmarks by Solomon (1987). However, label algorithms tend to have shortcomings when it is possible to produce long paths in the number of visited nodes, e.g., when resources are not particularly constraining. This usually results in a large amount of labels that have to be processed and can become very time consuming, see Jepsen et al. (2007). This motivates alternative solution methods for solving ESPPRC, e.g., lagrangian relaxation or branch-and-cut (BAC) algorithms. An advantage of a BAC algorithm would be the possibility to efficiently handle *decreasing* extension functions which can be difficult to handle in label algorithms that traditionally relies on *increasing* functions when extending labels, see Desaulniers et al. (1998), Irnich (2007) for further details on the concept of extension functions.

Bauer et al. (2002) suggest to solve the ESPPCC by a BAC algorithm, but to our knowledge nothing further has been published in the literature although several BAC algorithms exist for problems related to the ESPPCC. Bauer et al. (2002) consider the knapsack constrained circuit problem (KCCP) where a minimal capacitated cycle in a graph is sought. This is equivalent to the ESPPCC if one node is fixed in the KCCP, since this node can be spilt into a source and a destination node in the ESPPCC. They implemented a BAC algorithm to solve a variant of the KCCP where the demand of the nodes were given with unit weights and denoted it the cardinality constraint circuit problem.

A related and well studied problem is the traveling salesman problem (TSP), see Padberg and Rinaldi (1990), Naddef and Thilnel (2002a,b). The main difference between the ESPPCC and the TSP is that no capacity limit is present in the TSP and that all the nodes must be visited on the path.

The prize collecting traveling salesman problem (PCTSP) presented by Balas (1989, 1995) is another related problem. In this TSP variant a prize is collected at each visited node and a minimum amount of accumulated prizes must be collected on the tour. The difference with this variant of the TSP and the ESPPRC is that in the PCTSP a minimum amount of prizes (load) needs to be collected, whereas for the ESPPCC a maximum amount of load can be collected.

In the orienteering problem presented by Fischetti et al. (1998) the profit of visiting the nodes is maximized and the length of the tour is bounded by a maximum length. The difference here being that the load is on the edges instead of the nodes.

The main contribution of this paper is the introduction of a BAC algorithm for solving the ESPPCC. This includes a 2-index mathematical model and a presentation of valid inequalities with emphasis on the introduction of the generalized capacity inequalities. Thorough computational experiments are performed to verify the usefulness of a the BAC algorithm, e.g., by comparing running times with those of a label algorithm.

The paper is outlined as follows: Section 2 presents a formal integer programming model of the

ESPPCC, Section 3 presents the cutting planes used in the BAC algorithm, the computational results are found in Section 4, and Section 5 holds concluding remarks and suggestions for further research.

2 Mathematical Models

This section presents a flow model for the ESPPCC in an undirected graph $G(E,V)$. In the following, variable y_i indicate the use of node $i \in V \setminus \{s,t\}$, and variable x_e indicate the use of edge $e \in E$ where $e(i,j)$ denotes the end nodes i and j of e. When describing the model some shorthand notation will be used. For the set of edges T let

$$x(T) = \sum_{e \in T} x_e$$

Furthermore, for a set of nodes $S \subseteq V$ let the set of edges $\delta(S) = \{e(i,j) : i \in S \wedge j \in V \setminus S\}$ denote the edges between S and $V \setminus S$ with $\delta(\{i\}) = \delta(i)$ for a single node $i \in V$. Also, for a set of nodes S let

$$y(S) = \sum_{i \in S} y(i)$$

and let $E(S) = \{e(i,j) : i \in S \wedge j \in S\}$ be the set of edges between the nodes in S.

The mathematical model of ESPPCC is then:

$$\min \sum_{e \in E} c_e x_e \tag{1}$$

$$\text{s.t. } x(\delta(s)) = 1 \tag{2}$$

$$x(\delta(t)) = 1 \tag{3}$$

$$\sum_{e \in \delta(i)} x_e = 2y_i \qquad \forall i \in V \setminus \{s,t\} \tag{4}$$

$$\sum_{i \in V} d_i y_i \leq Q \tag{5}$$

$$x(E(S)) \leq y(S) - y_i \qquad \forall i \in S, \forall S \subseteq V, |S| \geq 2 \tag{6}$$

$$x_e \in \{0,1\} \qquad \forall e \in E \tag{7}$$

$$y_i \in \{0,1\} \qquad \forall i \in V \setminus \{s,t\} \tag{8}$$

The objective function (1) minimizes the overall edge cost. Constraints (2) and (3) ensure that the path starts in the source node and and ends in the target. Constraints (4) map the x and y variables. Constraint (5) imposes the capacity. Constraints (6) impose connectivity and subtour elimination. Finally, constraints (7) and (8) bounds the variables indicating the use of edges and nodes.

This model has $|E| + |V-2|$ variables and an exponential number of constraints due to (6). In a BAC algorithm these constraints will be disregarded and separated when violated to ensure feasibility.

3 Cutting Planes

This section presents the inequalities used in the BAC algorithm: the generalized subtour elimination constraints (6) which are included in the model presented in Section 2 but can be used as cutting planes when relaxed from the model, the 0-1 knapsack cover inequalities, and the generalized capacity inequalities for the ESPPCC.

3.1 Generalized Subtour Elimination Constraints

These constraints are a generalization of the subtour elimination constraints known from TSP, which are also valid for ESPPCC on the form:

$$x(E(S)) \leq |S| - 1 \qquad \forall S \subseteq V \tag{9}$$

Restricting the constants on the right-hand-side to reflect the actual node flow provides a tighter inequality since $y_i \leq 1$ for all $i \in V \setminus \{s,t\}$. The generalized subtour elimination constraints can be written on either of the forms:

$$x(E(S)) \leq y(S) - y_i \qquad \forall i \in S, \forall S \subseteq V, |S| \geq 2 \tag{10}$$

$$x(\delta(S)) \geq 2y_i \qquad \forall i \in S, \forall S \subseteq V \setminus \{s,t\} \tag{11}$$

Separation of (10) and (11) can be done by solving a $s-t$-minimum cut from each node $i \in V \setminus \{s,t\}$ to the target node t on the induced graph of the LP solution $(x^\star, y^\star)$ with edge weights w_e given as:

$$w_e = \begin{cases} x_e^\star & e \in E \setminus \{e(s,t)\} \\ M & e = (s,t) \end{cases}$$

where M is a sufficiently large constant to ensure that s and t are on the same side of the cut, see Wolsey (1998).

3.2 0-1 Knapsack Cover Inequalities

A 0-1 knapsack cover inequality for a set of nodes $S \subseteq V$ where $\sum_{i \in S} d_i > Q$ are given as:

$$y(S) \leq |S| - 1 \tag{12}$$

The inequality state that if a set of nodes violates the capacity then not all nodes in the set can be visited by the path. The 0-1 knapsack cover inequality (12) can be rewritten as

$$\sum_{i \in S} (1 - y_i) \geq 1 \tag{13}$$

Given the LP solution (x^*, y^*) the separation problem becomes finding a cover S, i.e, a set $S \subseteq V$ satisfying $\sum_{i \in S} d_i > Q$, where

$$\sum_{i \in S} (1 - y_i^*) < 1 \tag{14}$$

in which case the corresponding 0-1 knapsack cover inequality (12) is violated. The most violating (12) is identified by minimizing the left-hand side of (14), i.e., by solving:

$$\zeta = \min_{S \subseteq V} \left\{ \sum_{i \in S} (1 - y_i^*) z_i : \sum_{i \in S} d_i z_i > Q, z \in \{0,1\}^{|V|} \right\}$$

If $\zeta \geq 1$ no violated (12) exists. This problem is a minimization version of the well known 0-1 knapsack problem, see Kellerer et al. (2004), Wolsey (1998).

Inspired by the generalization of the generalized subtour elimination constraints Jepsen and Spoorendonk (2008) suggested to exploit the fact that since $y_i \leq 1$ for all $i \in V \setminus \{s,t,\}$ then the flow through

a set of nodes S can be less than 2 in an LP solution. That is, scaling the right-hand-side of (12) with half the flow $x(\delta(S))$ yields

$$y(S) \leq \frac{1}{2}(|S|-1)x(\delta(S)) \tag{15}$$

When $x(\delta(S)) < 2$ there are cases where the inequality (15) is violated and the normal 0-1 knapsack cover inequality (12) is not. Jepsen and Spoorendonk (2008) suggested an enumeration scheme to separate the inequalities. Their results indicated that (15) had a negative effect on the convergence of the BAC algorithm, therefor this family of inequalities are not pursued further in this paper.

3.3 Generalized Capacity Inequalities

This subsection introduces a family of inequalities inspired by the fractional capacity inequalities of the capacitated vehicle routing problem (CVRP), see Toth and Vigo (2001). The generalized capacity inequality for a set of nodes $S \subseteq V \setminus \{s,t\}$ introduced in this paper are given as:

$$\frac{1}{2}Qx(\delta(S)) \geq \sum_{i \in S} d_i y_i \tag{16}$$

The inequality ensures that a set S of nodes are visited according to their demand, e.g., if two third the capacity is consumed in S then the set must be visited at least two third time. An example of a violated (16) can be seen in the example given in Figure 3.3 on the following page.

The validity of (16) is proved in the following proposition:

Proposition 1. *The generalized capacity inequality* (16) *is valid for ESPPCC.*

Proof. If $y(S) = 0$ it is implied that $x(\delta(S)) = 0$, therefor both the left-hand side and the right hand side evaluate to 0.

If $y(S) \geq 1$ it is implied that $x(\delta(S)) \geq 2$ and due to the capacity constraint (5) the right-hand side can never evaluate to more than Q which will be the minimal value of the left-hand side, i.e., in this case the capacity constraint (5) dominates the generalized capacity inequality (16). □

Given an LP solution $(x^\star, y^\star)$ the separation problem of (16) becomes the problem of finding a set $S \subseteq V \setminus \{s,t\}$ where

$$\begin{aligned}
&\frac{1}{2}Qx^\star(\delta(S)) < \sum_{i \in S} d_i y_i^\star \\
\Leftrightarrow\ &\frac{1}{2}Qx^\star(\delta(S)) - \sum_{i \in S} d_i y_i^\star + \sum_{i \in V} d_i < \sum_{i \in V} d_i \\
\Leftrightarrow\ &\frac{1}{2}Qx^\star(\delta(S)) + \sum_{i \in S} d_i (1 - y_i^\star) + \sum_{i \in V \setminus S} d_i < \sum_{i \in V} d_i
\end{aligned}$$

Separating (16) can be done by solving $|V|-2$ different $s-t$-minimum cut problems one from each node $h \in V \setminus \{s,t\}$ to the target node t. The problems are solved as maxflow problems using the same procedure as for separating (10) and (11). The maxflow problem for each h is solved on a directed

Consider the fractional solution given by the graph to the right with different fractional edge values indicated by the dotted and punctured lines. The nodes are numbered $0,\ldots,5$ where a path is sought from node 0 to 0. The demand is given as $d = \{0,2,2,2,2,1\}$ and the capacity Q is 5.

Consider a generalized capacity inequality (16) covering the node set $S = \{1,2,3\}$ resulting in a fractional flow $x^\star(\delta(S)) = x^\star_{01} + x^\star_{03} = \frac{4}{3}$ through the node set. The corresponding (16) is violated since

$$\frac{1}{2}Qx^\star(\delta(S)) = \frac{10}{3} \not\geq \sum_{i\in S} d_i y^\star_i = \frac{12}{3}$$

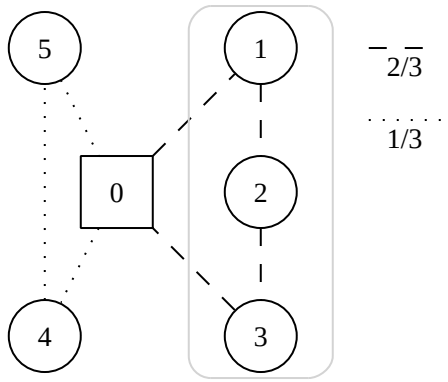

Figure 1: A violated generalized capacity inequality (16).

graph induced from the LP solution $(x^\star, y^\star)$, i.e., edges are split into opposite directed arcs, and the arcs into h are disregarded. The edge weights e_{ij} are given as:

$$w_{ij} = \begin{cases} \frac{1}{2}Qx^\star_{hj} + d_j & i = h,\ j \in V\setminus\{h,t\} \\ \frac{1}{2}Qx^\star_{it} + d_i(1-y^\star_i) & i \in V\setminus\{s,t\},\ j = t \\ \frac{1}{2}Qx^\star_{ij} & i \in V\setminus\{h,t\},\ j \in V\setminus\{h,t\} \\ M & i = s, j = t \end{cases}$$

where M is a sufficiently large constant to ensure that s and t are on the same side of the cut.

This graph is more dense than the induced graph used for separating (10) and (11), therefor the separation of (16) is expected to be slower.

4 Computational Results

A new set of benchmarks based on the CVRP instances from `http://www.branchandcut.org` are used for the computational experiments. The instances are divided in series A, B, E, G, M, and P according to the authors. The ESPPCC instances are pricing problems gathered from solving the CVRP with column generation, see Fukasawa et al. (2006), Jepsen et al. (2007), i.e., dual values have been subtracted from the edge costs resulting in instances with negative edge weights. The ESPPCC instances are based on late column generation iterations of the harder (time $>$ 500 sec.) CVRP instances solved by Fukasawa et al. (2006). The ESPPCC instances are gathered in the SPPRCLIB available at `http://www.diku.dk/~spooren/spprclib.htm`.

The experiments begins with an investigation of the impact of the parameter setting for cut generation of the generalized subtour elimination constraints (10) which is part of the model for the ESPPCC given by (1)-(8). Next, the impact of the generalized capacity inequalities (16) are investigated. For the parameter test we consider 10 of the harder problems in the test library, two from each series except the G series. This is followed by a lower bound comparison using different separation strate-

gies. Last a comparison of the BAC algorithm and a state-of-the-art label algorithm derived from the algorithm used in Jepsen et al. (2007).

All experiments are performed on a 2.66 GHz Intel(R) Xeon(R) X5355 machine with 8 GB memory using CPLEX 10.2. The BAC algorithm is implemented using callback functions for cut generation which is available in the CPLEX callable library. The tests are performed using the default CPLEX parameters. This includes the generation of cuts for general mixed-integer programs such as Chvátal-Gomory, MIR, and disjunctive cuts. Also, the 0-1 knapsack covers are included in the CPLEX default settings and preliminary tests indicated that neither the separation time or the change in lower bounds were much affected by the cuts. Therefor, we have not done any further tests of the 0-1 knapsack covers but rely on the CPLEX default settings.

4.1 Parameters Impact for the Generalized Subtour Elimination Constraints

The setting of the parameters for generation of violated generalized subtour elimination constraints (6) can have a huge influence on the computational time for the BAC algorithm. A low threshold on violation will result in good lower bounds and fewer branch nodes but a slower convergence in each node, while the opposite is true for a high threshold. Also the number of violated cuts added in each iteration can influence the convergence and the time spent re-optimizing the LP-problem.

Figure 2 on the next page shows a plot with two axes given as the violation threshold and number of cuts to add per iteration. The requirement of violation is ranging from 0.1 to 1 in steps of 0.1 and the number of cuts to add is ranging from 1 then 10 to 100 in steps of 10. The third vertical axis indicate the average time spent. The time for each instance is scaled to the interval $[0,1]$ where 0 is the minimum time and 1 a maximum time given for all parameter settings for that instance.

From Figure 2 on the following page it is observed that the best parameter setting appears to be to add 1 cut per iteration that is violated by at least 0.4. This indicates that the cut separation time is insignificant compared to solving the LPs.

4.2 Investigating the Generalized Capacity Inequalities

Note, that the generalized capacity inequalities (16) can substitute the generalized subtour elimination constraints (10) in model (1)-(8), since any infeasible integer solution will be violated by some generalized capacity inequality. However, preliminary test showed that due to slow separation times (16) are not competitive compared to (10) in a BAC algorithm. Therefor, a cut policy was chosen such that (16) are only separated (and possible added) whenever no violated (10) are separated (using the default parameters found above).

Again preliminary tests indicated that due to the slow separation routine (16) were not worth the effort. A slow separation was expected since the max-flow calculations is done on very dense graphs compared to the very sparse graph used in the separation of (10). However, we believe that it is relevant to investigate if (16) can ever become useful, e.g., with a faster heuristic separation routine.

Figure 3 on page 9 shows, as before, a plot of the violation threshold, number of cuts to add per iteration, and average time. The time is calculated without the separation time of (16), hence it only indicates if the convergence of the BAC is improved or not when (16) are added.

Figure 3 on page 9 indicates that a large violation threshold (≥ 0.8) is preferred for (16). This further indicates that the convergence of the BAC algorithm is faster when few of (16) are added leading to a smaller LP but a worse lower bound. Figure 4 substantiate this result as it can be seen that almost no cuts are added with violation thresholds 0.8 and higher.

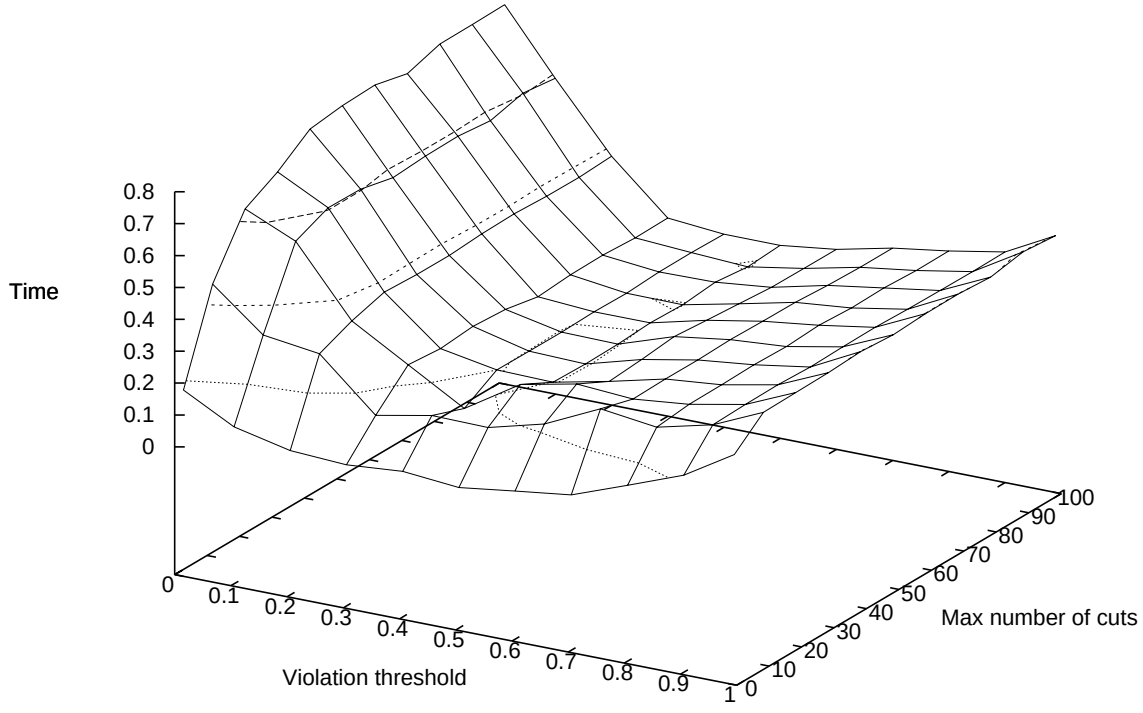

Figure 2: Parameter test for the generalized subtour elimination constraints (10). Above is a plot of the average time given the violation threshold and the number of cuts to add.

Although the generalized capacity inequalities (16) are a theoretically nice set of inequalities our tests have shown that in their current form and with the proposed exact separation routine the inequalities do not appear to be computationally competitive.

4.3 Lower Bound Comparison

Table 1 on page 11 sums up the root lower bounds (root) and the number of branch nodes (B&B) for three different cut separation parameter settings. A '-' entry in the branch node columns indicate that the BAC algorithm timed out at 600 seconds. The three parameter setting tested are:

- `GSEC` is the BAC algorithm where at most 1 violated generalized subtour elimination constraint (10) with a minimum violation of 0.01 is added per iteration.
- `GCI` is the BAC algorithm with the `GSEC` parameter setting and when no violated (10) are found then at most 1 violated generalized capacity inequality (16) with a minimum violation of 0.01 is added.
- `default` is the BAC algorithm where at most 1 violated generalized subtour elimination constraint (10) with a minimum violation of 0.4 is added per iteration.

The optimal solution is given in the right most column.

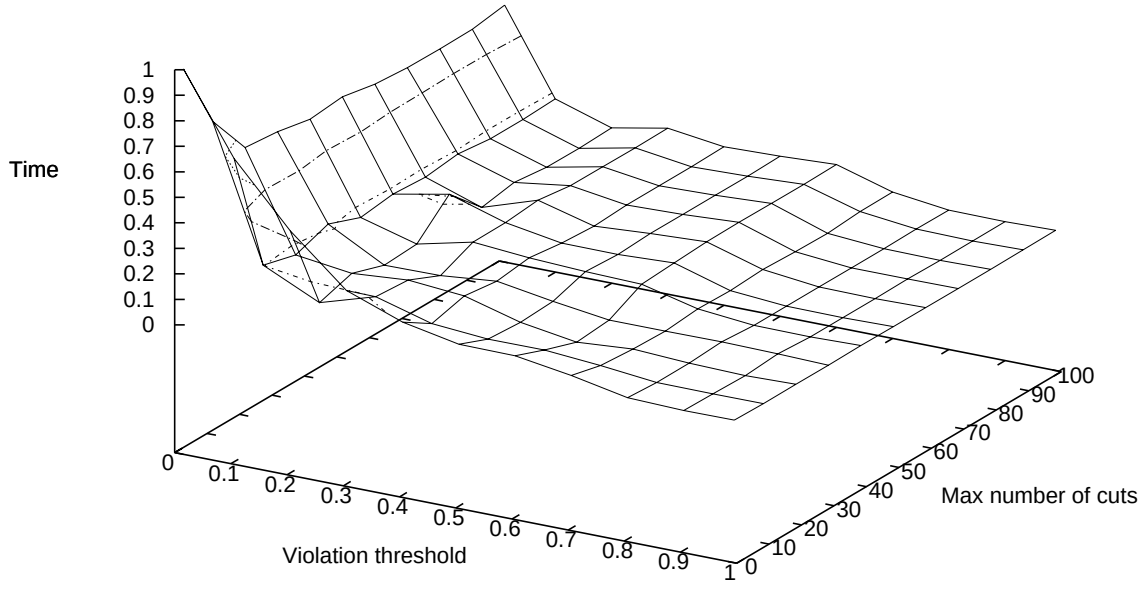

Figure 3: Parameter test for the generalized capacity inequalities (16). Above is a plot of the average time given the violation threshold and the number of cuts to add.

When comparing the parameter settings `GSEC` and `GCI`, it is obvious that the generalized capacity inequalities (16) improves the lower bounds considerably. The average gap is decreases by 63% when comparing the two settings, this includes the instances that timed out and potentially could have improved the lower bound further. Surprisingly the number of branch nodes does not decrease proportionally with the size of the gap. That is, for the instances that did not time out the average gap is closed by 76% but with only 7% fewer branch nodes. In several cases the number of branch nodes actually increases considerable. This indicates that (16) complicates the branch decisions.

The behavior observed between parameter setting `GSEC` and `default` is more as expected. A worse lower bound with the `default` setting leads to more branch nodes. However, the previous test for the generalized subtour elimination (10) constraints showed that this setting was the fastest on average.

4.4 Comparison with a Label Algorithm

Table 2 on page 12 shows the running time of the BAC algorithm with default parameters compared to the running time of a label algorithm.

The BAC algorithm clearly outperforms the label algorithm. That is, in all 45 instances. However, it is worth noting that when the solution is near 0 (which is and upper bound for all instances since they are generated as pricing problems in a column generation algorithm) then the label algorithm

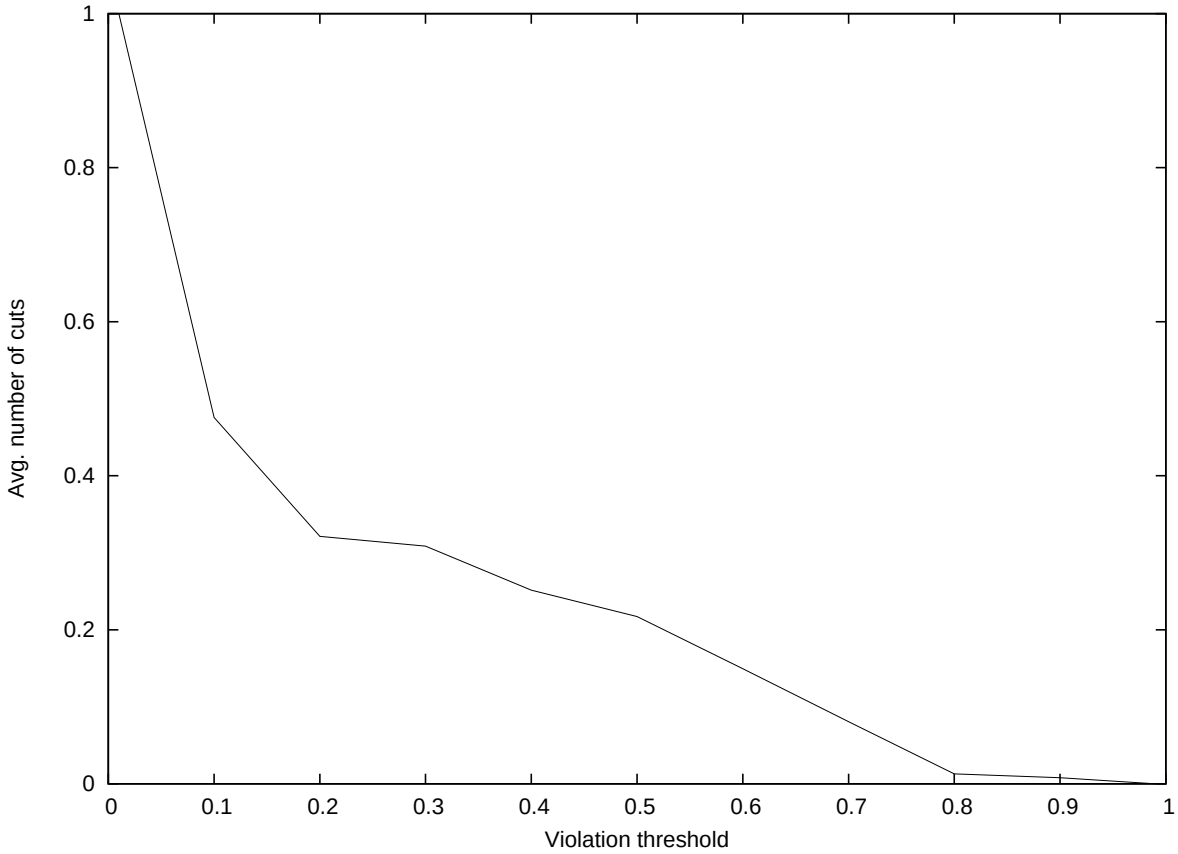

Figure 4: Parameter test for the generalized capacity inequalities (16). Above is given the average scaled number of generalized capacity inequalities added with different violation thresholds when solving the instances, i.e., with a violation threshold of 0.1 the number of cuts are decreased by about 50 % compared to the setting with a violation threshold of 0.01.

performs much better than on the instances that contains much negativity. That is, the label algorithm is faster when there are less negativity in the problem whereas the BAC algorithm appears to be more robust.

	GSEC		GCI		default		
Name	B&B	root	B&B	root	B&B	root	solution
A-n54-k7-149	231	-90877	-	-41213	280	-109018	-12492
A-n60-k9-57	1641	-98206	-	-64557	3071	-118437	-1000
A-n61-k9-80	205	-63534	92	-41032	462	-73397	-23549
A-n62-k8-99	133	-103839	-	-47340	301	-122973	-35969
A-n63-k9-157	122	-63082	492	-38929	113	-78190	-24189
A-n63-k10-44	127	-76475	149	-51035	280	-80765	-32561
A-n64-k9-45	358	-92812	157	-65209	425	-104686	-50550
A-n65-k9-10	152	-93117	129	-58526	189	-103936	-42835
A-n69-k9-42	72	-56453	-	-53299	179	-60410	-43290
A-n80-k10-14	84	-121510	45	-112483	120	-128508	-105283
B-n45-k6-54	277	-95588	497	-88761	502	-103214	-74278
B-n50-k8-40	166	-105497	-	-41212	237	-128488	-12832
B-n52-k7-15	25	-85997	22	-79129	59	-90278	-74998
B-n57-k7-20	12	-876421	19	-876421	328	-882924	-867154
B-n66-k9-50	239	-81006	28	-38097	1195	-94120	-26520
B-n67-k10-26	184	-55180	178	-26808	343	-63086	-21924
B-n68-k9-65	150	-88375	-	-55175	342	-99383	-31001
B-n78-k10-70	344	-91021	-	-54330	480	-101516	-44333
E-n76-k7-44	117	-30127	115	-25885	338	-32038	-22214
E-n76-k10-72	239	-36569	164	-31404	138	-38613	-25241
E-n76-k14-102	3163	-28126	-	-16153	3992	-31018	-1
E-n76-k15-40	3747	-25752	-	-17526	4993	-28675	-1
E-n101-k8-291	48	-8296	-	-7398	197	-9472	-4266
E-n101-k14-158	1468	-25748	-	-22729	1350	-30882	-3590
G-n262-k25-316	669	-1434843	-	-1434843	1510	-1434883	-1426535
M-n101-k10-97	40	-35323	37	-34758	76	-37825	-32628
M-n121-k7-260	89	-162680	-	-161424	147	-164742	-160097
M-n151-k12-15	338	-87899	-	-85488	822	-92880	-79996
M-n200-k16-143	6	-199411	4	-199411	118	-201772	-198792
M-n200-k17-12	4	-121506	1	-121210	7	-121506	-121210
P-n50-k7-92	950	-18594	1152	-12245	1319	-21516	-2
P-n50-k8-19	160	-89868	40	-89848	207	-90606	-83307
P-n50-k10-24	197	-19811	608	-11971	443	-21975	-2965
P-n51-k10-30	1028	-23812	-	-18488	1588	-27061	-2
P-n55-k7-116	84	-27065	36	-22945	105	-28094	-17824
P-n55-k8-260	101	-18839	145	-11377	167	-22237	-3573
P-n55-k10-44	913	-21448	2197	-11798	1192	-25131	-1090
P-n55-k15-88	5971	-26723	-	-20135	4781	-28128	-2
P-n60-k10-24	242	-26948	137	-21183	301	-29289	-15001
P-n60-k15-8	1495	-21889	1889	-13812	1507	-24674	-534
P-n65-k10-102	2390	-18923	-	-10975	2532	-21424	-3
P-n70-k10-12	2	-72264	1	-70317	21	-73460	-70317
P-n76-k4-41	1	-88276	1	-88276	1	-88276	-88276
P-n76-k5-16	6	-108884	10	-108884	24	-108884	-107633
P-n101-k4-174	174	-19656	165	-19041	395	-19887	-17702

Table 1: Comparison of the number of branch nodes and lower bounds.

Name	BAC time (s)	label time (s)	speed up
A-n54-k7-149	6.96	1735.23	249.3
A-n60-k9-57	36.55	242.64	6.6
A-n61-k9-80	4.44	>7200.00	∞
A-n62-k8-99	17.94	>7200.00	∞
A-n63-k9-157	3.16	>7200.00	∞
A-n63-k10-44	2.12	693.80	327.3
A-n64-k9-45	14.57	>7200.00	∞
A-n65-k9-10	4.43	>7200.00	∞
A-n69-k9-42	1.76	3246.72	1844.7
A-n80-k10-14	12.14	>7200.00	∞
B-n45-k6-54	1.32	>7200.00	∞
B-n50-k8-40	11.01	>7200.00	∞
B-n52-k7-15	1.00	>7200.00	∞
B-n57-k7-20	1.74	>7200.00	∞
B-n66-k9-50	66.93	>7200.00	∞
B-n67-k10-26	4.62	>7200.00	∞
B-n68-k9-65	11.88	>7200.00	∞
B-n78-k10-70	24.30	>7200.00	∞
E-n76-k7-44	6.02	>7200.00	∞
E-n76-k10-72	1.19	>7200.00	∞
E-n76-k14-102	14.77	45.19	3.1
E-n76-k15-40	19.59	151.59	7.7
E-n101-k8-291	8.08	>7200.00	∞
E-n101-k14-158	37.84	>7200.00	∞
G-n262-k25-316	53.00	>7200.00	∞
M-n101-k10-97	3.12	>7200.00	∞
M-n121-k7-260	34.46	>7200.00	∞
M-n151-k12-15	78.03	>7200.00	∞
M-n200-k16-143	3.18	>7200.00	∞
M-n200-k17-12	17.75	>7200.00	∞
P-n50-k7-92	2.42	104.22	43.1
P-n50-k8-19	0.36	>7200.00	∞
P-n50-k10-24	0.72	2.91	4.0
P-n51-k10-30	2.18	4.06	1.9
P-n55-k7-116	0.58	2275.07	3922.5
P-n55-k8-260	1.20	133.45	111.2
P-n55-k10-44	2.14	14.69	6.9
P-n55-k15-88	3.97	44.73	11.3
P-n60-k10-24	1.04	110.20	106.0
P-n60-k15-8	1.95	2.50	1.3
P-n65-k10-102	6.65	163.48	24.6
P-n70-k10-12	0.24	>7200.00	∞
P-n76-k4-41	1.85	>7200.00	∞
P-n76-k5-16	0.57	>7200.00	∞
P-n101-k4-174	11.25	>7200.00	∞
Best	45	0	

Table 2: Time comparison of the BAC and the label algorithm.

5 Concluding Remarks

This paper introduced a BAC algorithm for solving the ESPPCC. The algorithm clearly outperformed a label algorithm for the considered instances. Label algorithms have been the preferred solution approach up until now, but the experimental results presented in this paper suggest otherwise. However, it should be noted that the label algorithm is almost competitive when the instances contain little negativity. This is especially interesting when using column generation algorithms where subproblems with little negativity must be solved to optimality.

Furthermore, the generalized capacity inequalities were introduced as a set of valid inequalities for the ESPPCC. On the bright side, it can be concluded that the inequalities improve the lower bounds significantly. However, this comes at a cost of complicating the branch decision leading to a large amount of branch nodes. Also, the exact separation routine takes a considerable amount of time. This is due to solving a maxflow problem on a near complete graph. That is, the generalized capacity inequalities improves the lower bound but leads to overall slower running times.

Future research could include the adaption of more valid inequalities known from related problems such as the TSP and the CVRP, e.g., two-matching inequalities, comb inequalities, and infeasible path inequalities. Another interesting direction is the conditional cuts by Fischetti et al. (1998). Such a cut resembles a specialized branch rule as it cuts off some of the branch tree after solving a subproblem for the subtree.

Also, it would be interesting to extend the BAC algorithm to solve the more general ESPPRC. Extending the mathematical formulation to cover a directed graph is fairly easy. This will result in a doubling of the number of variables. This does not favor the BAC algorithm where compact formulations are preferred but it may be manageable. However, a much more delicate matter is handling resources with bounds imposed at the nodes or edges (compared to a capacity resource where the bound is global). Such resources includes the well known time window constraints which are often modeled using ”big-M” constraints that are known to be notoriously computationally unstable. An alternative for handling time window-like constraints are the use of valid inequalities such as infeasible path inequalities. However, there are an exponential number of these inequalities which are $\mathcal{NP}$-hard to separate (compared to the polynomial separation time of the generalized subtour elimination constrains) so these cuts are bound to have a significant impact on the running time.

References

Balas, E. 1989. The prize collecting traveling salesman problem. *Networks* **19** 621–636.

Balas, E. 1995. The Prize Collecting Traveling Salesman Problem: II. Polyhedral Results. *Networks 25* **25** 199–216.

Bauer, P., J. T. Linderoth, M. W. P. Savelsbergh. 2002. A branch and cut approach to the cardinality constrained circuit problem. *Mathematical Programming* **91** 307–348.

Beasley, J.E., N. Christofides. 1989. An algorithm for the resource constrained shortest path problem. *Networks* **19** 379–394.

Boland, N., J. Dethridge, I. Dumitrescu. 2006. Accelerated label setting algorithms for the elementary resource constrained shortest path problem. *Operation Research Letters* **34**(1) 58–68.

Carlyle, W. M., J. O. Roset, R. K. Wood. 2005. Lagrangian relaxation and enumeration for solving constrained shortest-path problems. Operations Research Department, Naval Postgraduate School, Monterey, California, USA.

Chabrier, A. 2005. Vehicle routing problem with elementary shortest path based column generation. *Computers & Operations Research* **23**(10) 2972–2990.

Danna, E., C. Le Pape. 2005. Accelerating branch-and-price with local search: A case study on the vehicle routing problem with time windows. G. Desaulniers, J. Desrosiers, , M. M. Solomon, eds., *Column Generation*, chap. 3. Springer, 30–130.

Desaulniers, G., J. Desrosiers, J. Ioachim, I. M. Solomon, F. Soumis, D. Villeneuve. 1998. A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. T. G. Crainic, G. Laporte, eds., *Fleet Management and Logistics*. Kluwer, 57–93.

Desrochers, M., J. Desrosiers, M. Solomon. 1992. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research* **40** 342–354.

Desrosiers, J., F. Soumis, M. Desrochers. 1984. Routing with time windows by column generation. *Networks 14* **14** 545–565.

Dror, M. 1994. Note on the complexity of the shortest path models for column generation in vrptw. *Operations Research* **42** 977–979.

Dumitrescu, I. 2002. Constrained path and cycle problems. Ph.D. thesis, Department of Mathematics and Statistics, University of Melbourne, Australia.

Feillet, D., P. Dejax, M. Gendreau, C. Gueguen. 2004. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks* **44**(3) 216–229.

Fischetti, M., J. Jos e Salazar Gonzãles, P. Toth. 1998. Solving the orienteering problem through branch-and-cut. *INFORMS Journal on Computing* **10**(2) 133–148.

Fukasawa, R., H. Longo, J. Lysgaard, M. Poggi de Aragão, M. Reis, E. Uchoa, R.F. Werneck. 2006. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming* **106**(3) 491–511.

Irnich, S. 2007. Resource extension functions: Properties, inversion, and generalization to segments. *OR Spectrum* doi:10.1007/s00291-007-0083-6.

Irnich, S., D. Villeneuve. 2006. The shortest path problem with resource constraints and k-cycle elimination for $k \geq 3$. *INFORMS Journal on Computing* **18**(3) 391–406. doi:10.1287/ijoc.1040.0117.

Jepsen, M., B. Petersen, S. Spoorendonk, D. Pisinger. 2007. Subset-row inequalities applied to the vehicle routing problem with time windows. *Operations Research* doi:10.1287/opre.1070.0449.

Jepsen, M., S. Spoorendonk. 2008. A note on the flow extended 0-1 knapsack cover inequalities for the elementary shortest path problem with a capacity constraint. Tech. Rep. 08-02, Department of Computer Science (DIKU),University of Copenhagen, Denmark, Universitetsparken 1, DK-2100 Copenhagen Ø, Denmark.

Kellerer, H., U. Pferschy, D. Pisinger. 2004. *Knapsack Problems*. Springer, Berlin, Germany.

Naddef, D., S. Thilnel. 2002a. Efficient Separation Routines for the Symmetric Traveling Salesman Problem I: General Tools and Comb Separation. *Mathematical Programming* **92** 237–255.

Naddef, D., S. Thilnel. 2002b. Efficient Separation Routines for the Symmetric Traveling Salesman Problem II: Separating Multi Handle Inequalities. *Mathematical Programming* **92** 257–285.

Padberg, M., G. Rinaldi. 1990. An efficient algorithm for the minimum capacity cut problem. *Mathematical Programming* **47** 19–36.

Righini, G., M. Salani. 2006. Symmetry helps: bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization* **3**(3) 255–273. doi: 10.1016/j.disopt.2006.05.007.

Righini, G., M. Salani. 2007. New dynamic programming algorithms for the resource constrained shortest path problem. *Networks* doi:10.1002/net.20212.

Salani, M. 2005. Branch-and-price algorithms for vehicle routing problems. Ph.D. thesis, Universitá degli studi di Milano, Facolta di Scienze Matematiche, Fisiche e Naturali, Dipartimento di Tecnologie dell'Informazione, Milano, Italy.

Solomon, M. M. 1987. Algorithms for the vehicle routing and scheduling problem with time window constraints. *Operations Research* **35** 234–265.

Toth, P., D. Vigo. 2001. An overview of vehicle routing problems. P. Toth, D. Vigo, eds., *The Vehicle Routing Problem, SIAM Monographs on Discrete Mathematics and Applications*, vol. 9, chap. 1. SIAM, 1–26.

Wolsey, L. A. 1998. *Integer Programming*. John Wiley & Sons, Inc.