European Journal of Operational Research 229 (2013) 595–598

Contents lists available at SciVerse ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

A note on the separation of subtour elimination constraints in elementary shortest path problems

Michael Drexl [*]

Chair of Logistics Management, Gutenberg School of Management and Economics, Johannes Gutenberg University, Mainz, Germany
Fraunhofer Centre for Applied Research on Supply Chain Services SCS, Nuremberg, Germany

ARTICLE INFO

Article history:
Received 18 July 2012
Accepted 10 March 2013
Available online 21 March 2013

Keywords:
Integer programming
Branch-and-cut
Separation
Subtour elimination constraints
Strong components

ABSTRACT

This note proposes an alternative procedure for identifying violated subtour elimination constraints (SECs) in branch-and-cut algorithms for elementary shortest path problems. The procedure is also applicable to other routing problems, such as variants of travelling salesman or shortest Hamiltonian path problems, on directed graphs. The proposed procedure is based on computing the strong components of the support graph. The procedure possesses a better worst-case time complexity than the standard way of separating SECs, which uses maximum flow algorithms, and is easier to implement.

1. Introduction

In many well-known routing problems such as travelling salesman (TSP, Gutin and Punnen, 2002) or elementary shortest path problems (ESPP, Ibrahim et al., 2009), a feasible solution must be a connected graph. In integer programming formulations for such problems, subtour elimination constraints (SECs) are used to ensure connectedness of solutions. There are different types of SECs, and for some of these types, the number of SECs is exponential in the size of the considered problem instance. When formulations using such a type of SEC are solved by branch-and-cut algorithms, violated SECs must be identified dynamically in the course of the algorithm. This identification process is commonly referred to as separation (cf. Wolsey, 1998). The separation of violated SECs is usually performed by means of maximum flow algorithms which yield the minimum cut as a by-product, for problems on undirected (Jepsen et al., 2011) as well as directed graphs (Fischetti and Toth, 1997). In this note, an alternative procedure for problems defined on directed graphs is described. The procedure is based on computing the strong components of the support digraph.

The note is structured as follows. After introducing some indispensable notation, the separation of SECs by maximum flows as well as by strong components is described for a standard formulation of a concrete problem, the ESPP. Then, the complexity of SEC separation and the applicability of separation by strong components to other routing problems are discussed. After that, computational results with a branch-and-cut algorithm for the ESPP, comparing the behaviour of both SEC separation approaches, are presented. The note ends with a brief conclusion.

2. Notation and definitions

Let $D=(V,A,c)$ be a simple, directed graph (digraph) with vertex set $V=V(D)$, arc set $A=A(D)$, and a cost $c_{ij}\in\mathbb{Q}$ associated with each $(i,j)\in A$. (A digraph is *simple* if it contains no arc starting and ending at the same vertex and no two arcs with the same start and the same end vertex.) A *subdigraph* of D is a digraph $D'=(V',A')$ with $V'\subseteq V$ and $A'=\{(i,j)\in A{:}i,j\in V'\}$. A *path* from $s\in V$ to $t\in V$ in D (an *s–t-path*) is a sequence $p=(i_1,i_2,\dots,i_{n-1},i_n)$ with $i_1=s$, $i_n=t$, $i_k\in V$ for $k=1,\dots,n$, and $(i_k,i_{k+1})\in A$ for $k=1,\dots,n-1$. The *cost* $c(p)$ of such a path p is $\sum_{k=1}^{n-1}c_{i_ki_{k+1}}$. A *cycle* is a path p with $i_1=i_n$. A cycle p is called *negative cycle* if $c(p)<0$. A path is called *elementary* if it fulfils $i_k\neq i_l$ for all $1\leqslant k<l\leqslant n$, that is, if it contains no cycles. A cycle is called elementary if it fulfils $i_k\neq i_l$ for all $1\leqslant k<l\leqslant n-1$.

D is *strongly connected* if it contains an i–j-path and a j–i-path for each pair of vertices $i,j\in V$. A strongly connected subdigraph $D'=(V',A')$ of D is called a *strong component* of D if there is no vertex $i\in V'$ and no vertex $j\in V\backslash V'$ for which D contains an i–j-path and a j–i-path. A *weak component* of D is a subdigraph $D'=(V',A')$ with the property that for any two vertices $i,j\in V'$, there is a sequence $(i_1,i_2,\dots,i_n)$ of vertices in V' with $i=i_1$, $j=i_n$, and either $(i_k,i_{k+1})\in A$ or $(i_{k+1},i_k)\in A$ or both for all $1\leqslant k<n$. A strong or weak component is called *non-trivial* if it consists of more than one vertex.

* Address: Chair of Logistics Management, Johannes Gutenberg University, Jakob-Welder-Weg 9 55128 Mainz, Germany. Tel.: +49 61313922085.
E-mail address: drexl@uni-mainz.de

0377-2217/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.ejor.2013.03.009

The *support digraph* $D_S = D_S(x)$ of a solution $x \in \mathbb{Q}_+^{|A|}$ to a relaxation of an integer program defined on D is the digraph that contains all arcs $(i,j) \in A$ with positive flow $x_{ij} > 0$ and only the vertices incident to these arcs. A *subtour* in a support digraph D_S is a strongly connected subdigraph of D_S that does not contain one or more vertices that must be visited in any feasible solution. That is, a subtour is always a subdigraph of exactly one non-trivial strong component. For simplicity, let $\delta^+(i) := \delta^+(\{i\})$ and $\delta^-(i) := \delta^-(\{i\})$. Finally, for any $B \subseteq A$ and $x \in \mathbb{Q}^{|B|}$, let $x(B) := \sum_{(i,j)\in B} x_{ij}$.

3. Separating SECs

Consider the following formulation (1)–(4) for the ESPP on a simple digraph $D = (V,A,c)$. The formulation seeks a shortest elementary path from a specified start vertex $s \in V$ to a specified target vertex $t \in V$ and uses binary variables x_{ij} indicating whether or not arc $(i,j) \in A$ is traversed. Without loss of generality, it is assumed that $\delta^-(s) = \delta^+(t) = \emptyset$.

$$\text{minimize} \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{1}$$

subject to

$$x(\delta^+(i)) - x(\delta^-(i)) = \begin{cases} 1, & i = s \\ -1, & i = t \\ 0, & i \in V \setminus \{s,t\} \end{cases} \tag{2}$$

$$x(\delta^+(S)) \geqslant x(\delta^+(i)) \quad \forall\, i \in S \subseteq (V \setminus \{s,t\}), |S| \geqslant 2 \tag{3}$$

$$x_{ij} \in \{0,1\} \quad \forall\, (i,j) \in A \tag{4}$$

Constraints (2) are the flow conservation constraints, and (3) are the SECs, which are only needed for the ESPP if the digraph may contain a negative cycle. In the following, an SEC is unequivocally referred to by (S,i).

Note that in constraints (3), S must not contain s or t. This is because a solution containing only the arc (s,t) is obviously feasible, but if $S = \{s,t\}$, then $x(\delta^+(S)) = 0 < 1 = x(\delta^+(s))$. To see that it is sufficient to consider sets S not containing s, note that, if (S,i) is a violated SEC with $s \in S$, then $(S\backslash\{s\},i)$ is also a violated SEC. This is because $\delta^-(s) = 0$, and hence $x(\delta^+(S\backslash\{s\})) \leqslant x(\delta^+(S)) < x(\delta^+(i))$.

In branch-and-cut algorithms for ESPPs and similar problems, on directed as well as undirected graphs, violated SECs are usually identified by optimally solving a maximum flow problem from s to i in the support (di)graph for each vertex $i \in V\backslash\{t\}$, using the x_{ij} values as arc capacities. If, for a vertex i, the maximum flow is less than $x(\delta^+(i))$, a violated SEC has been found. S is then the set of vertices that are on the same side of the i–t-cut as i, cf. Wolsey (1998, p. 154) ff., Jepsen et al. (2011), Fischetti and Toth (1997).

An alternative procedure for separating violated SECs (3) is based on the following observations.

Observation 1. If the support digraph $D_S = (V_S,A_S)$ of an integral solution $x \in \{0,1\}^{|A|}$ satisfying the flow conservation constraints (2) contains a violated SEC (S,i), then i belongs to a non-trivial strong component $C = (V_C,A_C)$ in D_S, and (V_C,i) is a violated SEC.

Observation 2. If the support digraph $D_S = (V_S,A_S)$ of an integral solution $x \in \{0,1\}^{|A|}$ satisfying the flow conservation constraints (2) contains a non-trivial strong component $C = (V_C,A_C)$, then the subtour elimination constraint $x(\delta^+(V_C)) \geqslant x(\delta^+(i))$ is violated for some $i \in V_C$.

Proof. Let $C = (V_C,A_C)$ be a non-trivial strong component in an integral solution $x \in \{0,1\}^{|A|}$ satisfying (2).

If there is no arc leaving C, then, because C is a strong component, all vertices $i \in V_C$ fulfil $x(\delta^+(i)) \geqslant 1 > 0 = x(\delta^+(V_C))$.

If there is an arc (i,j) leaving C, that is, $i \in V_C \not\ni j$, it must lie on a path from i to t, since otherwise, because of flow conservation, the flow on (i,j) would have to return to i and thus j would be in the same strong component as i. As the solution is integral, (i,j) is the only arc leaving C, because $x(\delta^-(t)) = 1$. Consequently, $x(\delta^+(i)) \geqslant 2 > 1 = x(\delta^+(V_C))$. □

Observations 1 and 2 imply the following observation.

Observation 3. An integral solution $x \in \{0,1\}^{|A|}$ satisfying constraints (2) is feasible, that is, violates no SEC, if and only if it does not contain a non-trivial strong component.

Observation 4. There are fractional solutions that contain a violated SEC but that do not contain a non-trivial strong component whose vertex set defines a violated SEC. An example is depicted in Fig. 1. In the support digraph depicted in the figure, the SEC $(\{i,j\},i)$ is violated, but i and j belong to the strong component $C = (\{i,j,k\}, \{(i,j), (j,i), (j,k), (k,j)\})$, and for no vertex $h \in V(C)$ is $(V(C),h)$ a violated SEC.

Observation 5. In fractional solutions consisting of only one weak component, there may be non-trivial strong components although no SEC is violated (Fig. 2, left) as well as strong components whose vertex set defines a violated SEC (Fig. 2, right).

The above observations show the following: SEC separation by maximum flows will find a violated SEC, if one exists, for fractional as well as integral solutions. On the other hand, separation by strong components only guarantees this for integral solutions. Therefore, when separation by strong components is to be used instead of separation by maximum flows, either the separation routine is only called when an integral solution has been found, or, for fractional solutions, each SEC resulting from a strong component is checked for violation using the current values of the x variables.

The decisive point is that Observations 1–5 show that a branch-and-cut algorithm where violated SECs (3) are identified only by computing the strong components of the support digraph will, if sufficient time and memory are available, eventually determine an optimal solution to any instance of (1)–(4) that has a feasible solution.

4. Complexity of SEC separation

Solving a maximum flow problem on a digraph $D = (V,A)$ takes $\mathcal{O}\left(|V|^2\sqrt{|A|}\right)$ time (Ahuja et al., 1993, p. 240). To compute the maximum flows from all $i \in V\backslash\{t\}$ to t, no more than $|V| - 1$ maximum flow problems have to be solved, leading to an overall worst-case complexity of $\mathcal{O}\left(|V|^3\sqrt{|A|}\right)$ for separation by maximum flows, which, if D is simple, equals $\mathcal{O}(|V|^4)$.

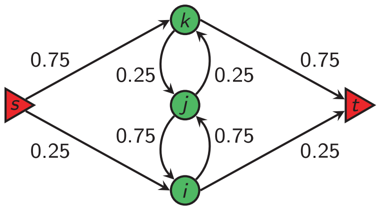

Fig. 1. Violated SEC $(\{i,j\},i)$, but no strong component violating an SEC.

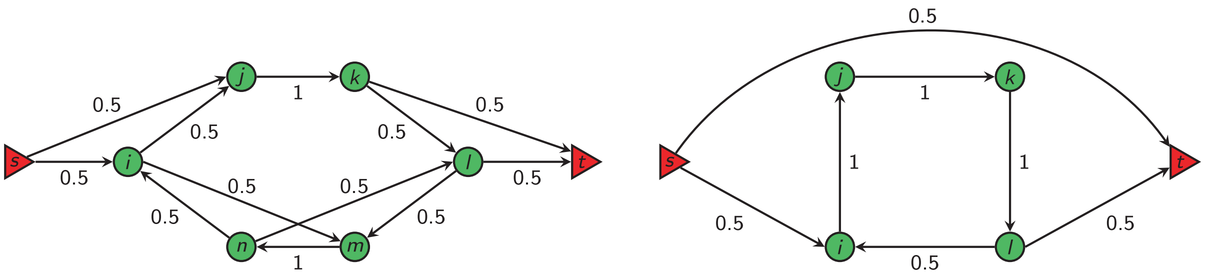

Fig. 2. Strong components without and with violated SEC.

The strong components of D can be computed in $\mathcal{O}(|V| + |A|)$ time ($\mathcal{O}(|V|^2)$ if D is simple) using the algorithm proposed by Tarjan (1972), which is based on depth-first search.

Since each vertex belongs to at most one non-trivial strong component, there are at most $|V| - 2$ SECs to be checked for violation after each solution to a relaxation of (1)–(4). This can be done in $\mathcal{O}(|A|)$ time, because each arc must be considered at most once. Thus, the worst-case time complexity of separation by strong components is $\mathcal{O}(|V| + |A| + |A|)$, which, if D is simple, equals $\mathcal{O}(|V|^2)$. This means that, for simple digraphs, separation by strong components has a worst-case time complexity that is two orders of magnitude better than separation by maximum flows.

5. Other routing problems

The proposed procedure can be used in branch-and-cut algorithms for many routing problems with specified start and end vertex on digraphs, such as the orienteering problem (Vansteenwegen et al., 2011) and the asymmetric Hamiltonian s–t-path problem (s–t-AHPP). A formulation for the latter problem is obtained from (1)–(4) by adding the constraint $x(\delta^+(i)) = 1$ for all $i \in V \backslash \{s,t\}$, which implies that SECs (3) can be replaced by

$$x(\delta^+(S)) \geqslant 1 \quad \forall\, S \subseteq (V \setminus \{s,t\}), \quad |S| \geqslant 2, \tag{5}$$

or, equivalently,

$$x(A(S)) \leqslant |S| - 1 \quad \forall\, S \subseteq (V \setminus \{s,t\}), \quad |S| \geqslant 2. \tag{6}$$

The above observations carry over directly to these SECs.

Branch-and-cut algorithms for asymmetric TSPs (ATSPs), that is, TSPs defined on digraphs can also use SEC separation by strong components. Since any feasible ATSP solution consists of exactly one strong component, this works only as long as the support digraph contains more than one strong component. However, a transformation of an ATSP instance into an s–t-AHPP instance is very easy: It is sufficient to replace one arbitrary vertex i by two vertices s and t, and to let all arcs originally emanating from i emanate from s and all arcs originally leading to i lead to t. Any feasible solution to the resulting AHPP then corresponds to a feasible ATSP solution with the same costs. This preprocessing can be done in $\mathcal{O}(|V|)$ time.

6. Computational experiments

Theoretical worst-case complexity and practical average-case time consumption are two different matters, as the simplex algorithm shows. Therefore, in the following, a computational study comparing the empirical behaviour of separation by maximum flows and separation by strong components is described.

In an earlier project, the author had implemented a branch-and-cut algorithm for the ESPP, using SEC separation by maximum flows, and had created a set of 360 test instances. The set contains 90 random instances with rational arc costs uniformly distributed within $[-1000; +1000]$, and 270 pricing subproblems from a heuristic column generation algorithm for the asymmetric m-salesmen TSP (cf. Gutin and Punnen, 2002, Chapter 1). The pricing problem in such an algorithm is an ESPP on a digraph with negative cycles, due to the dual prices of the master problem constraints. The rational arc cost values for these instances lie in $[-10^8; 10^8]$. All instances contain at least one negative cycle.

To solve the test instances, formulation (1)–(4) had been implemented in C++, using IBM Ilog Cplex Concert Technology, version 12.2. The standard Cplex cuts are automatically added. Separation by maximum flows is performed using a code written by Skorobohatyj (2012). In each call to a separation routine, before separating by maximum flows, the non-trivial weak components of the support digraph were identified with a union-find data structure as described by Wayne (2012). This is necessary because the maximum flow code does not work without modification for digraphs consisting of more than one weak component.

To test separation by strong components, a routine was added to the above implementation. This routine also first identifies the non-trivial weak components with union-find, and if there is only

Table 1
Computational results.

Instance class	Vertices (min./avg./max.)	Arcs (min./avg./max.)	Solution approach	B & B nodes (avg.)	Separated SECs (avg.)	CPU time (seconds) (avg.)
Random (90)	26/59/101	553/4203/9703	MF, all	12.9	281.6	2.4
			MF, one	11.6	78.2	12.2
			SC, all	13.6	289.9	1.2
			SC, one	12.8	92.6	0.9
Pricing (270)	28/61/103	651/4370/10,101	MF, all	224.4	1119.1	45.9
			MF, one	475.0	234.7	95.6
			SC, all	271.2	1521.4	17.7
			SC, one	532.4	213.7	15.1
All (360)	26/61/103	553/4328/10,101	MF, all	171.6	909.7	35.0
			MF, one	359.2	195.6	74.7
			SC, all	206.8	1213.5	13.6
			SC, one	402.5	183.4	11.6

one non-trivial weak component, the non-trivial strong components are identified using the Boost Graph library (Boost, 2012).

The results are shown in Table 1. Twelve different algorithmic set-ups were tried for both separation by maximum flows ('MF' in the table) and separation by strong components ('SC'). With respect to running time, the best results for separation by maximum flows were obtained when all violated SECs found were added in each iteration ('all' in the table). For separation by strong components, it was better to add only one most violated SEC in each iteration ('one'). In both cases, it was much better to separate SECs at each node of the branch-and-bound tree, that is, also for fractional solutions of the LP relaxation, instead of separating SECs only at nodes where the solution of the LP relaxation is integral. Therefore, results for the latter case are omitted.

With respect to running times, Table 1 shows that the fastest set-up for separation by strong components ('SC, one') is more than three times faster than the fastest set-up for separation by maximum flows ('MF, all'). The best set-up for separation by strong components also separated fewer SECs than the best set-up for separation by maximum flows.

7. Conclusion

This note has described that and how violated subtour elimination constraints in branch-and-cut algorithms for several asymmetric routing problems can be identified by computing the strong components of the support digraph.

It has been shown that separation by strong components has a worst-case time complexity that is two orders of magnitude better than the usual separation by maximum flows. Computational experiments have confirmed the principal effectiveness of separation by strong components.

It is evident that the empirical behaviour of an algorithm depends, to a large degree, on the quality of the implementation. Therefore, it is difficult to give a general statement in favour of one separation approach or the other. Nevertheless, inspection of the source code of several implementations of maximum flow as well as strong component algorithms (Boost, 2012; Sedgewick and Wayne, 2012; Skorobohatyj, 2012) shows that, whereas it is non-trivial to implement an efficient maximum flow algorithm, a highly efficient implementation of a procedure for determining the strong components of a digraph is relatively easy. Implementations of algorithms for computing strong components as well as for computing maximum flows are freely available on the Internet for various common programming languages. Thus, it is easy to incorporate both approaches in an implementation of a branch-and-cut algorithm and compare them for different instance types.

References

Ahuja, R., Magnanti, T., Orlin, J., 1993. Network Flows. Prentice-Hall, Upper Saddle River.

Boost. Boost Graph Library. <http://boost.org> (accessed 14.07.12).

Fischetti, M., Toth, P., 1997. A polyhedral approach to the asymmetric traveling salesman problem. Management Science 43, 1520–1536.

Gutin, G., Punnen, A. (Eds.), 2002. The Traveling Salesman Problem and its Variations. Kluwer, Dordrecht.

Ibrahim, M., Maculan, N., Minoux, M., 2009. A strong flow-based formulation for the shortest path problem in digraphs with negative cycles. International Transactions in Operational Research 16, 361–369.

Jepsen, M., Petersen, B., Spoorendonk, S., 2011. A branch-and-cut algorithm for the capacitated profitable tour problem. Technical report, DTU Management Engineering, Technical University of Denmark.

Sedgewick, R., Wayne, K., 2012. Algorithms. <http://algs4.cs.princeton.edu/42directed> (accessed 14.07.12).

Skorobohatyj, G., 2012. Finding a Minimum Cut between all Pairs of Nodes in an Undirected Graph. <http://elib.zib.de/pub/Packages/mathprog/mincut/all-pairs/index.html> (accessed 14.07.12).

Tarjan, R., 1972. Depth-first search and linear graph algorithms. SIAM Journal on Computing 1, 146–160.

Vansteenwegen, P., Souffriau, W., Van Oudheusden, D., 2011. The orienteering problem: a survey. European Journal of Operational Research 209, 1–10.

Wayne, K., 2012. Union-Find Algorithms. <http://www.cs.princeton.edu/~rs/AlgsDS07/01UnionFind.pdf> (accessed 14.07.12).

Wolsey, L., 1998. Integer Programming. Wiley, New York.