Gutenberg School of Management and Economics

Discussion Paper Series

Solving Elementary Shortest-Path Problems as Mixed-Integer Programs

Michael Drexl and Stefan Irnich

Januar 2012

Discussion paper number 1201

Johannes Gutenberg University Mainz
Gutenberg School of Management and Economics
Jakob-Welder-Weg 9
55128 Mainz
Germany
wiwi.uni-mainz.de

Contact details

Michael Drexl
Chair of Logistics Management
Johannes Gutenberg University Mainz
Jakob-Welder-Weg 9
55128 Mainz
Germany
drexl@uni-mainz.de

Stefan Irnich
Chair of Logistics Management
Gutenberg School of Management and Economics
Johannes Gutenberg University Mainz
Jakob-Welder-Weg 9
55128 Mainz
Germany

irnich@uni-mainz.de

Solving Elementary Shortest-Path Problems as Mixed-Integer Programs

Technical Report LM-2012-01

Michael Drexl, Stefan Irnich
Chair of Logistics Management, Gutenberg School of Management and Economics
Johannes Gutenberg University, Mainz
E-mail: {drexl,irnich}@uni-mainz.de

11th January 2012

Abstract

Ibrahim, Maculan, and Minoux (International Transactions in Operational Research, vol. 16, 2009, pp. 361–369) presented and analyzed two integer programming formulations for the elementary shortest-path problem (ESPP), which is known to be NP-hard if the underlying digraph contains negative cycles. In fact, the authors showed that a formulation based on commodity flows possesses a significantly stronger LP-relaxation than a formulation based on arc flow variables. Since the ESPP is essentially an integer problem, the contribution of our paper lies in extending this research by comparing the formulations with regard to the computation time and memory requirements required for their integer solution. Moreover, we assess the quality of the lower bounds provided by an integer relaxation of the commodity flow formulation.
Keywords: Elementary shortest-path problem, Negative cycles, Mixed-integer programming

1 Introduction

The elementary shortest-path problem (ESPP) is to determine a shortest path between two vertices of a graph so that each vertex of the graph is visited at most once. For graphs without negative cycles, strongly polynomial algorithms for solving the ESPP exist (Ahuja et al. [1]). By contrast, the computation of shortest elementary paths in graphs with negative cycles is NP-hard (ib.). Ibrahim et al. [7] have studied two integer programming formulations for the ESPP and have compared these with regard to the strength of the respective linear relaxations. The contribution of the present paper is to compare the integer versions of the formulations with regard to the computation time and memory requirements, and to assess the quality of the lower bounds provided by an integer relaxation of the second formulation. Our research is motivated by the fact that (resource-constrained) ESPPs in graphs with negative cycles appear as subproblems in column-generation solution approaches for vehicle-routing problems (VRPs) (Toth and Vigo [11], Golden et al. [5]). The traditional method for solving these shortest-path subproblems is a labelling algorithm based on dynamic programming (Irnich and Desaulniers [8]). However, there are variants of VRPs where such labelling algorithms do not work well or cannot be applied at all (Desaulniers et al. [3], Crainic et al. [2], Drexl [4]). Therefore, the research in this paper also constitutes a step toward finding out if and how such subproblems can be solved by branch-and-cut.

2 Mathematical models

We assume a directed graph $D = (V, A)$ with vertex set V and arc set A. Without loss of generality, D is assumed to contain neither loops nor parallel arcs, so that an arc from a vertex $i \in V$ to a vertex $j \in V$ can unequivocally be referred to as $(i, j) \in A$ with cost $c_{ij} \in \mathbb{Q}$. A *path* from s to t in D (an *s-t-path*) is a sequence $p = i_1, i_2, \ldots, i_{n-1}, i_n$ with $i_1 = s$, $i_n = t$, $i_k \in V$ for $k = 1, \ldots, n$, and $(i_k, i_{k+1}) \in A$ for $k = 1, \ldots, n-1$. The *cost* $c(p)$ of such a path p is $\sum_{k=1}^{n-1} c_{i_k i_{k+1}}$. D may contain *negative cycles*, that is, paths p with $i_1 = i_n$ and $c(p) < 0$. A path is *elementary* if it fulfils $i_k \neq i_l$ for all $1 \leq k < l \leq n$.
A *(weak) component* of D is a digraph $D' = (V', A')$ with $V' \subseteq V$, $A' = \{(i, j) \in A : i, j \in V'\}$ and the property that for any two vertices $i, j \in V'$, there is a sequence $i_1, i_2, \ldots, i_n$ 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 $1 \leq k < n$.
In the following, we use the standard notation for the forward star $\delta^+(S) := \{(i, j) \in A : i \in S \not\ni j\}$, the backward star $\delta^-(S) := \{(j, i) \in A : i \in S \not\ni j\}$, and the inner arcs $A(S) := \{(i, j) \in A : i, j \in S\}$ for all $S \subseteq V$. For simplicity, we define the shortcuts $\delta^+(i) := \delta^+(\{i\})$ and $\delta^-(i) := \delta^-(\{i\})$. Without loss of generality, we assume that $\delta^-(s) = \delta^+(t) = \emptyset$. Finally, for any subset $B \subseteq A$ and any vector $w \in \mathbb{Q}^{|B|}$, we define $w(B) := \sum_{(i,j) \in B} w_{ij}$.
We seek a shortest elementary path from a specified start vertex $s \in V$ to a specified target vertex $t \in V$. (When negative cycles are present, no shortest non-elementary path exists.)

2.1 A classical formulation

The first formulation for the ESPP considered here uses only one type of variable, x_{ij}, indicating whether or not arc $(i, j) \in A$ is traversed (cf. Ibrahim et al. [7], Jepsen et al. [9]):

$$\sum_{(i,j) \in A} c_{ij} x_{ij} \to \min \text{ subject to} \tag{1a}$$

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

$$x(\delta^+(S)) - x(\delta^+(i)) \geq 0 \quad \forall\, i \in S \subsetneq V, |S| \geq 2 \tag{1c}$$

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

The objective function, (1a), is the sum of the costs of the arcs in the path. Constraints (1b) ensure flow conservation, and constraints (1c), of which there are exponentially many (their number is exponential in the number of vertices of the graph), are the subtour-elimination constraints (SECs), that is, they exclude cycles and thus ensure elementarity of the solution paths.
Compared to the formulation given by Ibrahim et al., the following modification is made on formulation (1): Ibrahim et al. use constraints

$$x(A(S)) \leq |S| - 1 \quad \forall\, S \subseteq V, |S| \geq 2, \tag{2}$$

to eliminate subtours. Instead, we use constraints (1c), since we did not have an efficient procedure for separating constraints (2); moreover, (1c) are stronger than (2).

2.2 A formulation based on commodity flows

The second formulation studied by Ibrahim et al. uses three types of variable: As before, x_{ij} indicates whether or not arc $(i, j) \in A$ is traversed. Moreover, y_i indicates, for all $i \in V$, whether or not vertex i is visited. Finally, variables $z_{ij}^k \geq 0$ measure the flow, through arc $(i, j) \in A$, from the source vertex s to a vertex $k \in V \setminus \{s\}$. Defining commodities $K := V \setminus \{s, t\}$, the model can be stated as follows:

$$\sum_{(i,j)\in A} c_{ij} x_{ij} \to \min \text{ subject to} \tag{3a}$$

$$z_{ij}^k \leq x_{ij} \quad \forall\, k \in K, (i,j) \in A, i \neq k, s \neq j \neq t \tag{3b}$$

$$z^k(\delta^+(s)) = y_k \quad \forall\, k \in K \tag{3c}$$

$$z^k(\delta^+(i)) - z^k(\delta^-(i)) = 0 \quad \forall\, k \in K, i \in V \setminus \{s, k, t\} \tag{3d}$$

$$z^k(\delta^-(k)) = y_k \quad \forall\, k \in K \tag{3e}$$

$$x(\delta^+(i)) = y_i \quad \forall\, i \in V \setminus \{t\} \tag{3f}$$

$$x(\delta^-(i)) = y_i \quad \forall\, i \in V \setminus \{s\} \tag{3g}$$

$$x(\delta^+(s)) = 1 \tag{3h}$$

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

$$x_{ij} \in \{0, 1\} \quad \forall\, (i,j) \in A \tag{3j}$$

$$y_i \in \{0, 1\} \quad \forall\, i \in V \tag{3k}$$

$$z_{ij}^k \geq 0 \quad \forall\, k \in K, (i,j) \in A, i \neq k, s \neq j \neq t \tag{3l}$$

The objective function, (3a), is identical to the one for model (1). In any feasible solution to (3), constraints (3b) ensure that constraints (3c)–(3e) provide flow conservation in the z variables for all visited vertices. This means that (3c)–(3e) ensure that there is a path from s to each visited vertex, including t. Constraints (3f) and (3g) ensure that each visited vertex i other than s and t is reached and left exactly once, in other words, that there is exactly one arc entering i and one arc leaving i. Constraints (3h) and (3i) require that the source vertex s be left and that the sink vertex t be reached exactly once. Now, since each visited vertex other than s and t is reached and left exactly once, all these vertices must lie on the unique s-t-path, and, moreover, this path must be elementary: A subtour containing a vertex i which lies on a path from s to t is impossible since this would imply that i is reached more than once. An isolated subtour not connected to s is impossible since there is a path from s to each visited vertex. In this way, the elimination of subtours is ensured by the interplay of all constraints.
Compared to the formulation given by Ibrahim et al., the following modifications are made on formulation (3): Ibrahim et al. introduce z_{ij}^k variables for all $k \in V \setminus \{s\}$ and $(i,j) \in A$, they formulate constraints (3b) for all $k \in V \setminus \{s\}$ and $(i,j) \in A$, constraints (3c) and (3e) for all $k \in V \setminus \{s\}$, and constraints (3d) for all $k \in V \setminus \{s\}$ and $i \in V \setminus \{s, k\}$. Moreover, they formulate constraints (3f) also for $i = s$ and constraints (3g) also for $i = t$. Thus, formulation (3) uses fewer variables and constraints, which is possible since $\delta^-(s) = \delta^+(t) = \emptyset$ is assumed.

2.3 T-family relaxations

Ibrahim et al. study also a third formulation, called T-family relaxation. Such a relaxation results from (3) by replacing $k \in K$ by $k \in T$ with T being a subset of K. Of particular interest is the case where $T = \emptyset$. In this case, there are no z variables, and constraints (3b)–(3e) vanish.
Note that, if the set T is a proper subset of K, subtours may occur in the optimal solution to the LP-relaxation as well as in the optimal integer solution.

2.4 Structural properties of the formulations

The subtour-elimination constraints are necessary in both formulations; they are non-redundant inequalities for the formulation, that is, disregarding the SECs may lead to false solutions. The difference between formulations (1) on the one hand and (3) on the other is that the former has an exponential number ($O(2^{|V|})$) of constraints overall. This is due to the exponential number of subtour-elimination constraints (1c), which must be separated dynamically for larger instances if an exact solution is to be computed. By contrast, the number of variables and constraints in the latter formulation is $O(|V||A|)$ and allows their explicit specification. (Nevertheless, for larger instances, it is recommendable to also dynamically separate constraints (3b)–(3e)). On the downside, the number of variables and variable types is larger in the latter formulation. The effects of these structural properties on the computational behaviour of the formulations are unclear and must be tested empirically. This was done in our computational experiments, which are described next.

3 Computational experiments

Ibrahim et al. used random test instances small enough to allow explicit specification of all constraints in both formulations. We decided to create larger instances which require the separation of SECs for the classical formulation. Moreover, we extracted pricing subproblems from a heuristic column-generation algorithm for the asymmetric m-salesmen TSP (cf. Gutin and Punnen [6], Chapter 1) to see how these compare with purely random instances. The pricing problem in such an algorithm is an ESPP on a graph with negative cycles, due to the dual prices of the master-problem constraints. To be precise, Table 1 specifies the 15 classes of the 420 test instances that were generated. For the random instances, the arc cost values were created from a uniform distribution within the indicated ranges. For the pricing-problem instances, for each underlying m-salesmen TSP instance, the first, penultimate, and last pricing problem created by the column-generation algorithm were used. The very negative values for the 'first' and 'penultimate' instances are due to Big-M values for artificial variables. All instances contain at least one negative cycle.

Class name	Type	No. instances	No. vertices	No. arcs	Arc cost range	Arc cost type
R_sparse_25	Random	20	26	300	$[-10; +10]$	Integer
R_sparse_50	Random	20	51	1,225	$[-10; +10]$	Integer
R_sparse_100	Random	20	101	4,950	$[-10; +10]$	Integer
R_dense_25	Random	30	26	553	$[-1{,}000.0; +1{,}000.0]$	Double
R_dense_50	Random	30	51	2,353	$[-1{,}000.0; +1{,}000.0]$	Double
R_dense_100	Random	30	101	9,703	$[-1{,}000.0; +1{,}000.0]$	Double
P_first_25	Pricing	30	28	651	$[-10^8; -9.48 \cdot 10^7]$	Double
P_penultimate_25	Pricing	30	28	651	$[-10^7; +30{,}000]$	Double
P_last_25	Pricing	30	28	651	$[-30{,}000; +30{,}000]$	Double
P_first_50	Pricing	30	53	2,551	$[-10^8; -9.48 \cdot 10^7]$	Double
P_penultimate_50	Pricing	30	53	2,551	$[-10^7; +30{,}000]$	Double
P_last_50	Pricing	30	53	2,551	$[-30{,}000; +30{,}000]$	Double
P_first_100	Pricing	30	103	10,101	$[-10^8; -9.48 \cdot 10^7]$	Double
P_penultimate_100	Pricing	30	103	10,101	$[-10^7; +30{,}000]$	Double
P_last_100	Pricing	30	103	10,101	$[-30{,}000; +30{,}000]$	Double

Table 1: Test instances

For formulation (3), the following three approaches were examined: (i) Solve with all SECs added ex ante. (ii) Solve with SECs as lazy constraints. This means that all SECs are added ex ante to a pool. Initially, the model consists only of constraints (3f)–(3l). The LP-relaxation is solved, and when an integer feasible solution is found, the lazy constraints are checked for violation. Any

violated lazy constraints are then added, and the LP-relaxation of the model is re-optimized.
(iii) Solve with dynamic separation of SECs.
To dynamically separate the subtour-elimination constraints of formulations (1) and (3), that is, constraints (1c) and (3b)–(3e) respectively, a two-stage approach is used. First, the support graph is checked for isolated components not connected to s and t. For formulation (1), for one vertex of each isolated component found, an SEC is added. For formulation (3), for one vertex i of each isolated component found, the corresponding set of SECs for $k = i$ is added. Second, if the support graph consists of only one component, a maximum i-t-flow/minimum i-t-cut problem is solved for each vertex $i \in V \setminus \{t\}$, using the x_{ij} values as arc capacities. A maximum flow which is less than the absolute outflow from i, that is, less than $x(\delta^+(i))$, indicates a violated SEC. In model (1), S is then the set of vertices which are on the same side of the i-t-cut as i. For one such i, an SEC is added in formulation (1); in formulation (3), the corresponding set of SECs for $k = i$ is added. Basically, it is sufficient to check for violated SECs whenever a feasible integer solution to the current formulation containing only a part of all SECs is found. However, it turned out useful to also add violated SECs after solving the LP-relaxation at each node of the branch-and-bound tree.
To solve the test instances, the formulations described above were implemented in C++, using IBM Ilog Cplex Concert Technology, version 12.2. The standard Cplex cuts were automatically added. Where SECs were dynamically separated, the isolated components were identified with a union-find data structure as described by Wayne [12]. The max-flow problems were solved using a code written by Skorobohatyj [10]. All computations were performed in single-thread mode on a PC with an Intel Core i7-2600 CPU, 3.40 GHz, and 16 GB main memory running Windows 7 64-bit. A time limit of 1,200 seconds of CPU time for each instance was set.
The computational results are indicated in the tables on the subsequent pages. The columns in the tables have the following meaning:
Instance class: Class of test instance as described in Table 1
Solution approach: Formulation and solution approach used
No. variables: Number of variables in respective formulation
No. constraints: Number of constraints in respective formulation without dynamically added SECs, that is, for solution of formulation (3) with all SECs added ex ante, overall number of constraints including (3b)–(3e)
% optimal: Percentage of instances solved to optimality; for the exact approaches, an instance is only counted if optimization terminated before time limit was reached
B & B nodes: Number of nodes in the branch-and-bound tree
No. separated SECs: Number of SECs which were separated dynamically, or, for the approach with a static lazy constraint pool, were identified as violated and were moved from the pool to the formulation
CPU time: Overall CPU time in seconds
For the rightmost three columns, '(min. / avg. / max.)' means the minimum, average, and maximum value respectively.
The computational experiments yielded the following essential results:

- The classical formulation (1) clearly outperforms the commodity flow formulation (3). Comparing (1) instance by instance with the respective best exact solution approach for (3) shows that:
  - (1) uses less computation time than (3) for 94 % of all 420 test instances and is faster by at least a factor of 10 for 66 % of all 280 instances with 50 or more vertices.
  - (1) is more than one second slower than (3) for only one instance (4.91 seconds).
  - (1) yields an optimal solution within the time limit for 98 % of all instances, compared to 74 % for (3).
  - (3) solves no instance to optimality which (1) does not also solve optimally.

Instance class	Solution approach	No. variables	No. constraints	% optimal	B & B nodes (min. / avg. / max.)	No. separated SECs (min. / avg. / max.)	CPU time (min. / avg. / max.)
R_sparse_25	Classical	299	26	100	1 / 2 / 10	25 / 39 / 64	0.00 / 0.06 / 0.13
	Commodity flow complete	6,908	7,235	100	1 / 2 / 12	n.a.	0.05 / 0.59 / 1.17
	Commodity flow, lazy constraint pool	6,908	52	100	1 / 3 / 15	670 / 3,527 / 4,869	0.05 / 0.52 / 1.23
	Commodity flow, dynamic SEC separation	6,908	52	100	1 / 3 / 24	0 / 760 / 1,777	0.00 / 0.09 / 0.31
	T-family relaxation, $T = \emptyset$	325	52	20	1 / 1 / 1	n.a.	0.00 / 0.01 / 0.03
R_sparse_50	Classical	1,225	51	100	1 / 4 / 10	52 / 82 / 134	0.31 / 0.58 / 1.03
	Commodity flow complete	58,937	60,213	100	1 / 3 / 17	n.a.	36.89 / 146.22 / 680.31
	Commodity flow, lazy constraint pool	58,937	102	100	2 / 16 / 70	33,762 / 41,761 / 51,706	49.52 / 168.83 / 635.08
	Commodity flow, dynamic SEC separation	58,937	102	100	1 / 8 / 31	0 / 5,697 / 11,051	0.19 / 6.86 / 37.35
	T-family relaxation, $T = \emptyset$	1,276	102	35	1 / 1 / 1	n.a.	0.00 / 0.01 / 0.05
R_sparse_100	Classical	4,950	101	100	1 / 6 / 17	61 / 146 / 236	3.79 / 9.98 / 15.18
	Commodity flow complete	485,105	490,157	5	1 / 1 / 1	n.a.	756.84 / 1,178.15 / $>$ 1,200
	Commodity flow, lazy constraint pool	485,105	202	0	5 / 16 / 32	94,087 / 159,726 / 171,934	$>$ 1,200 / $>$ 1,200 / $>$ 1,200
	Commodity flow, dynamic SEC separation	485,105	202	70	1 / 12 / 32	0 / 30,414 / 49,574	17.35 / 723.14 / $>$ 1,200
	T-family relaxation, $T = \emptyset$	5,051	202	70	1 / 1 / 1	n.a.	0.02 / 0.04 / 0.08
R_dense_25	Classical	553	26	100	1 / 7 / 38	25 / 38 / 61	0.02 / 0.08 / 0.19
	Commodity flow complete	12,769	12,842	100	1 / 5 / 31	n.a.	0.48 / 1.35 / 5.04
	Commodity flow, lazy constraint pool	12,769	52	100	1 / 7 / 44	3,926 / 6,389 / 9,862	0.39 / 1.70 / 5.91
	Commodity flow, dynamic SEC separation	12,769	52	100	1 / 7 / 46	0 / 1,381 / 2,655	0.02 / 0.32 / 1.33
	T-family relaxation, $T = \emptyset$	579	52	20	1 / 1 / 1	n.a.	0.00 / 0.01 / 0.05
R_dense_50	Classical	2,353	51	100	1 / 13 / 70	50 / 76 / 105	0.94 / 1.36 / 2.14
	Commodity flow complete	113,044	113,192	97	1 / 12 / 95	n.a.	24.23 / 304.59 / $>$ 1,200
	Commodity flow, lazy constraint pool	113,044	102	87	1 / 17 / 67	56,047 / 78,768 / 100,221	80.86 / 465.52 / $>$ 1,200
	Commodity flow, dynamic SEC separation	113,044	102	100	1 / 15 / 91	0 / 10,608 / 20,754	0.55 / 25.77 / 90.70
	T-family relaxation, $T = \emptyset$	2,404	102	15	1 / 1 / 1	n.a.	0.00 / 0.02 / 0.05
R_dense_100	Classical	9,703	101	100	1 / 15 / 132	98 / 121 / 161	30.44 / 34.86 / 55.18
	Commodity flow complete	951,094	951,392	0	1 / 1 / 1	n.a.	$>$ 1,200 / $>$ 1,200 / $>$ 1,200
	Commodity flow, lazy constraint pool	951,094	202	0	1 / 1 / 1	186,285 / 211,832 / 231,129	$>$ 1,200 / $>$ 1,200 / $>$ 1,200
	Commodity flow, dynamic SEC separation	951,094	202	70	1 / 7 / 15	0 / 46,109 / 76,848	47.80 / 853.84 / 1,246.70
	T-family relaxation, $T = \emptyset$	9,804	202	15	1 / 1 / 1	n.a.	0.02 / 0.05 / 0.08

Table 2: Computational results for random instances

Instance class	Solution approach	No. variables	No. constraints	% optimal	B & B nodes (min. / avg. / max.)	No. separated SECs (min. / avg. / max.)	CPU time (min. / avg. / max.)
P_first_25	Classical	651	28	100	1 / 2 / 8	36 / 46 / 53	0.05 / 0.10 / 0.17
	Commodity flow complete	16,329	16,408	100	1 / 1 / 1	n.a.	1.09 / 2.75 / 6.88
	Commodity flow, lazy constraint pool	16,329	56	100	1 / 2 / 8	7,191 / 9,210 / 11,811	1.17 / 3.37 / 7.35
	Commodity flow, dynamic SEC separation	16,329	56	100	1 / 1 / 10	4,389 / 6,625 / 8,151	0.23 / 1.02 / 3.60
	T-family relaxation, $T = \emptyset$	679	56	0	1 / 1 / 1	n.a.	0.00 / 0.01 / 0.03
P_first_50	Classical	2,551	53	100	1 / 9 / 29	83 / 101 / 133	1.14 / 1.48 / 1.81
	Commodity flow complete	127,654	127,808	25	1 / 6 / 17	n.a.	98.45 / 636.18 / $>$ 1,200
	Commodity flow, lazy constraint pool	127,654	106	17	7 / 29 / 61	75,128 / 97,699 / 107,764	301.10 / 917.93 / $>$ 1,200
	Commodity flow, dynamic SEC separation	127,654	106	100	1 / 8 / 33	50,040 / 55,795 / 60,048	60.72 / 191.39 / 588.89
	T-family relaxation, $T = \emptyset$	2,604	106	0	1 / 1 / 1	n.a.	0.00 / 0.02 / 0.03
P_first_100	Classical	10,101	103	100	6 / 41 / 86	175 / 207 / 239	30.69 / 38.29 / 48.49
	Commodity flow complete	1,010,304	1,010,608	0	1 / 1 / 1	n.a.	$>$ 1,200 / $>$ 1,200 / $>$ 1,200
	Commodity flow, lazy constraint pool	1,010,304	206	0	24 / 36 / 61	167,309 / 177,753 / 190,606	$>$ 1,200 / $>$ 1,200 / $>$ 1,200
	Commodity flow, dynamic SEC separation	1,010,304	206	0	1 / 1 / 1	420,084 / 446,756 / 470,094	$>$ 1,200 / $>$ 1,200 / $>$ 1,200
	T-family relaxation, $T = \emptyset$	10,204	206	0	1 / 1 / 1	n.a.	0.02 / 0.05 / 0.08
P_penultimate_25	Classical	651	28	100	1 / 12 / 88	55 / 80 / 126	0.17 / 0.31 / 0.55
	Commodity flow complete	16,329	16,408	100	1 / 3 / 17	n.a.	0.91 / 2.87 / 20.83
	Commodity flow, lazy constraint pool	16,329	56	100	1 / 26 / 211	3,172 / 8,195 / 11,049	0.81 / 3.91 / 12.03
	Commodity flow, dynamic SEC separation	16,329	56	100	1 / 17 / 121	5,643 / 8,235 / 10,659	0.53 / 3.48 / 12.28
	T-family relaxation, $T = \emptyset$	679	56	0	1 / 1 / 1	n.a.	0.00 / 0.01 / 0.03
P_penultimate_50	Classical	2,261	53	100	2 / 38 / 228	104 / 202 / 585	1.39 / 4.06 / 17.68
	Commodity flow complete	113,041	113,485	100	1 / 8 / 35	n.a.	42.35 / 212.50 / 853.00
	Commodity flow, lazy constraint pool	113,041	106	77	42 / 134 / 251	60,592 / 77,714 / 89,954	319.55 / 732.08 / $>$ 1,200
	Commodity flow, dynamic SEC separation	113,041	106	97	3 / 44 / 219	22,735 / 43,356 / 61,156	42.07 / 271.58 / $>$ 1,200
	T-family relaxation, $T = \emptyset$	2,314	106	0	1 / 1 / 1	n.a.	0.00 / 0.02 / 0.06
P_penultimate_100	Classical	10,101	103	93	27 / 1,704 / 6,556	321 / 470 / 735	74.40 / 344.10 / $>$ 1,200
	Commodity flow complete	1,010,304	1,010,608	0	1 / 1 / 1	n.a.	$>$ 1,200 / $>$ 1,200 / $>$ 1,200
	Commodity flow, lazy constraint pool	1,010,304	206	0	30 / 53 / 114	152,511 / 172,616 / 199,425	$>$ 1,200 / $>$ 1,200 / $>$ 1,200
	Commodity flow, dynamic SEC separation	1,010,304	206	0	1 / 1 / 1	390,078 / 415,416 / 440,088	$>$ 1,200 / $>$ 1,200 / $>$ 1,200
	T-family relaxation, $T = \emptyset$	10,204	206	0	1 / 1 / 1	n.a.	0.02 / 0.05 / 0.08

Table 3: Computational results for pricing instances

Instance class	Solution approach	No. variables	No. constraints	% optimal	B & B nodes (min. / avg. / max.)	No. separated SECs (min. / avg. / max.)	CPU time (min. / avg. / max.)
P_last_25	Classical	651	28	100	1 / 29 / 198	64 / 95 / 168	0.19 / 0.41 / 0.83
	Commodity flow complete	16,329	16,408	100	1 / 2 / 38	n.a.	1.36 / 2.82 / 12.92
	Commodity flow, lazy constraint pool	16,329	56	100	1 / 15 / 86	7,154 / 9,184 / 10,828	1.83 / 4.70 / 10.61
	Commodity flow, dynamic SEC separation	16,329	56	100	1 / 22 / 405	5,016 / 8,172 / 12,540	1.05 / 4.88 / 48.27
	T-family relaxation, $T = \emptyset$	679	56	0	1 / 1 / 1	n.a.	0.00 / 0.01 / 0.03
P_last_50	Classical	2,261	53	100	1 / 59 / 327	204 / 417 / 1,185	3.68 / 11.77 / 61.90
	Commodity flow complete	113,041	113,485	100	1 / 14 / 107	n.a.	30.86 / 227.49 / 1,115.77
	Commodity flow, lazy constraint pool	113,041	106	22	19 / 152 / 381	40,052 / 68,440 / 82,460	69.25 / 840.34 / > 1,200
	Commodity flow, dynamic SEC separation	113,041	106	100	1 / 53 / 360	20,384 / 40,102 / 56,615	23.06 / 237.92 / 884.04
	T-family relaxation, $T = \emptyset$	2,314	106	0	1 / 1 / 1	n.a.	0.00 / 0.02 / 0.03
P_last_100	Classical	10,101	103	25	96 / 2,347 / 6,510	327 / 493 / 625	92.82 / 470.14 / > 1,200
	Commodity flow complete	1,010,304	1,010,608	0	1 / 1 / 1	n.a.	> 1,200 / > 1,200 / > 1,200
	Commodity flow, lazy constraint pool	1,010,304	206	0	31 / 61 / 96	155,923 / 176,796 / 198,197	> 1,200 / > 1,200 / > 1,200
	Commodity flow, dynamic SEC separation	1,010,304	206	0	1 / 1 / 1	390,078 / 416,750 / 450,090	> 1,200 / > 1,200 / > 1,200
	T-family relaxation, $T = \emptyset$	10,204	206	0	1 / 1 / 1	n.a.	0.02 / 0.04 / 0.08

Table 4: Computational results for pricing instances (continued)

Solution approach	Avg. no. variables	Avg. no. constraints	% optimal	B & B nodes (min. / avg. / max.)	No. separated SECs (min. / avg. / max.)	CPU time (min. / avg. / max.)
Classical	4,018	61	98	1 / 306 / 6,556	25 / 180 / 1,185	0.00 / 65.29 / > 1,200
Commodity flow complete	348,417	348,928	65	1 / 4 / 107	n.a.	0.05 / 505.47 / > 1,200
Commodity flow, lazy constraint pool	348,417	121	60	1 / 40 / 381	670 / 87,948 / 231,129	0.05 / 620.28 / > 1,200
Commodity flow, dynamic SEC separation	348,417	121	74	1 / 14 / 405	0 / 108,849 / 470,094	0.00 / 405.49 / > 1,200
T-family relaxation, $T = \emptyset$	4,079	121	8	1 / 1 / 1	n.a.	0.00 / 0.02 / 0.08
All approaches	n.a.	n.a.	46	1 / 61 / 6,556	0 / 19,698 / 470.094	0.00 / 319.31 / > 1,200

Table 5: Aggregated computational results over all 420 instances

  - (1) separates fewer SECs than (3) for more than 94 % of all 420 test instances, although the overall number of SECs in the former formulation is much larger than in the latter.
- For the commodity flow formulation (3), dynamic separation of SECs is by far better than adding all SECs ex ante. Using a lazy constraint pool for the SECs is still worse. This is demonstrated by the fact that with dynamic separation, 74 % of all test instances are solved to optimality, compared to 65 and 60 % with ex ante adding of SECs and a static lazy constraint pool respectively. Moreover, dynamic separation is faster than the other two approaches for 72 % of all instances, and uses 25 and 53 % less overall computation time respectively.
- The T-family relaxation with $T = \emptyset$ yields very bad lower bounds. On average over all instances solved to optimality, the objective function values obtained with the T-family relaxation are 197 % below those of the optimal solutions.
- It is easy to see that the solutions obtained with the T-family relaxation with $T = \emptyset$ consist of an elementary s-t-path and zero or more cycles not connected to s and t. Removing all such isolated components yields a feasible solution, and, hence, an upper bound for the ESPP. The upper bounds obtained by this procedure, however, are also very bad, lying on average more than 70 % above the optimal solution values.
- A correlation analysis between formulations (1) and (3) with dynamic separation of SECs regarding the CPU time and the number of separated SECs showed only rather weak positive correlations between the formulations: The values of the sample correlation coefficient r were 0.703 and 0.718 respectively. This means that if an instance is relatively difficult to solve with the one formulation, this instance tends to be difficult to solve with the other formulation as well, although the relationship is not very pronounced.
- The instances generated from pricing problems are significantly more difficult than the random instances: On average, a pricing-problem instance required 30 % more computation time and the separation of 530 % more SECs compared to a random instance. No significant difference exists between the computation times needed and the number of SECs separated for the instances generated from the first, penultimate, and last pricing problems.

4 Conclusion

The central result of the computational study described in this paper is that, unfortunately, the results obtained by Ibrahim et al. for the LP-relaxations of the presented formulations do not carry over to the MIP solution. The classical formulation with only arc variables and exponentially many SECs is by far superior to the commodity flow formulation.
A challenging direction for future research would be a thorough polyhedral study of both formulations. It is however unlikely that valid inequalities for the commodity flow formulation will be found which are separable and strong enough to change the results in favour of the latter and close the huge performance gap to the classical formulation.

Acknowledgement. This research was funded by the Deutsche Forschungsgemeinschaft (DFG) under grant no. IR 122/5-1.

References

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

[2] T. Crainic, N. Ricciardi, and G. Storchi.
Models for evaluating and planning city logistics systems.
Transportation Science, 43:432–454, 2009.

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

[4] M. Drexl.
Synchronization in vehicle routing—a survey of VRPs with multiple synchronization constraints.
Technical Report 1103, Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, 2011.
Accepted for publication in Transportation Science.

[5] B. Golden, S. Raghavan, and E. Wasil, editors.
The Vehicle Routing Problem: Latest Advances and New Challenges, volume 43 of *Operations Research/Computer Science Interfaces Series.*
Springer, Berlin, 2008.

[6] G. Gutin and A. Punnen, editors.
The Traveling Salesman Problem and its Variations.
Kluwer, Dordrecht, 2002.

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

[8] S. Irnich and G. Desaulniers.
Shortest path problems with resource constraints.
In G. Desaulniers, J. Desrosiers, and M. Solomon, editors, *Column Generation*, pages 33–65. Springer, New York, 2005.

[9] M. Jepsen, B. Petersen, and S. Spoorendonk.
A branch-and-cut algorithm for the elementary shortest path problem with a capacity constraint.
Technical Report 08/01, Department of Computer Science, University of Copenhagen, 2008.

[10] G. Skorobohatyj.
Finding a Minimum Cut between all Pairs of Nodes in an Undirected Graph.
Available at `http://elib.zib.de/pub/Packages/mathprog/mincut/all-pairs/index.html`.
Accessed 5th January 2012.

[11] P. Toth and D. Vigo, editors.
The Vehicle Routing Problem.
SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, 2002.

[12] K. Wayne.
Union-Find Algorithms.
Available at `http://www.cs.princeton.edu/~rs/AlgsDS07/01UnionFind.pdf`.
Accessed 5th January 2012.