Integer Programming Formulations for the Elementary Shortest Path Problem

Article *in* European Journal of Operational Research · January 2016

DOI: 10.1016/j.ejor.2016.01.003

1 author:

Leonardo Taccari

11 PUBLICATIONS **47** CITATIONS

SEE PROFILE

Integer programming formulations for the elementary shortest path problem

Leonardo Taccari

Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Italy

Abstract

Given a directed graph $G = (V, A)$ with arbitrary arc costs, the Elementary Shortest Path Problem (ESPP) consists of finding a minimum-cost path between two nodes s and t such that each node of G is visited at most once. If negative costs are allowed, the problem is $\mathcal{NP}$-hard. In this paper, several integer programming formulations for the ESPP are compared. We present analytical results based on a polyhedral study of the formulations, and computational experiments where we compare their linear programming relaxation bounds and their behavior within a branch-and-cut framework. The computational results show that a formulation with dynamically generated cutset inequalities is the most effective.

Keywords: integer programming, elementary shortest path, branch-and-cut, extended formulations, subtour elimination constraints, generalized cutset inequalities

1. Introduction

Given a directed graph $G = (V, A)$ and arc costs c_{ij} for each $(i, j) \in A$, the shortest path problem consists of finding a minimum-cost path between two nodes s and t.

Often, an implicit assumption is that such path has to be elementary. A path is elementary if it does not visit any node more than once, i.e., if it does not contain subtours. When the costs c_{ij} induce no negative cycles on G, the problem can be solved efficiently via ad-hoc polynomial time algorithms, like Bellman-Ford's or Dijikstra's algorithm (if $c_{ij} \geq 0$). However, if negative cycles do arise, subtours must be explicitly prevented, leading to the so-called Elementary Shortest Path Problem (ESPP).

The ESPP is clearly $\mathcal{NP}$-hard due to a simple reduction from the Hamiltonian path problem. Its equivalent maximization counterpart, where one seeks a longest path over a graph with positive cycles, has been vastly discussed in the

Email address: leonardo.taccari@polimi.it (Leonardo Taccari)

literature, usually referred to as the Longest Path Problem (LPP). Björklund et al. [7] prove that the LPP is hard to approximate on unweighted directed graphs within a $n^{1-\epsilon}$ for any ϵ, unless $\mathcal{P} = \mathcal{NP}$. For undirected graphs, approximation algorithms are described in, e.g., [27] and [17].

Related to the ESPP are variants of the Travelling Salesman Problem (TSP) with profits, such as the Prize Collecting TSP [3], the Orienteering Problem [36] and the Capacitated Profitable Tour Problem [26], that involve prizes on each node, which can be visited at most once.

While an interesting problem in its own right, the ESPP also arises in the pricing subproblems of branch-and-price algorithms [1]. Often, the pricing phase in Vehicle Routing Problems (VRP) involves resource-constrained variants of the elementary shortest path problem (ESPPRC) [23]. These problems are usually solved with fast dynamic programming-based label algorithms, e.g., see [32, 15, 8]. It is possible to adapt this kind of approach to the unconstrained ESPP by considering an artificial resource for each node which is consumed when the node is visited, and imposing that no more than one unit of each resource is used, as already proposed in [6, 8]. However, this is a rather weak constraint, in the sense that it allows for very long paths, so that approaches based on label algorithms become very time consuming, as noted by Drexl and Irnich [14]. This limitation is already highlighted for the ESPP with a capacity constraint by Jepsen et al. [25], that propose a branch-and-cut algorithm that significantly outperforms label algorithms. In the context of a branch-and-price algorithm, where the ESPP is solved repeatedly in the pricing phase, another desirable feature of an integer programming approach is its flexibility, that allows one to easily incorporate general branching decisions or valid inequalities (e.g., the subset-row inequalities in [24]) that would change the structure of the pricing subproblem. These reasons motivate the study of integer programming techniques for the ESPP.

Several branch-and-cut approaches can be found in the literature for related problems [5, 19, 16, 26]. However, not much previous work has appeared on integer programming approaches tailored specifically for the ESPP. Ibrahim et al. [22] provide computational results on the linear programming (LP) bounds of a flow-based extended formulation, but give no details on its behavior in a branch-and-bound algorithm. Another extended formulation is proposed by Haouari et al. [21]. Drexl and Irnich [14] describe a branch-and-cut approach and compare its efficiency with the extended formulation in [22], while Drexl [13] studies the efficient separation of subtour elimination constraints for the ESPP.

In the context of integer programming, the choice of a formulation is crucial for the effectiveness of methods based on branch-and-cut. In this article we present a thorough comparison between different formulations for the ESPP, which are described in detail in Section 2. We include integer programming formulations with exponentially many subtour elimination constraints, and mixed-integer programming extended formulations with a polynomial number of variables and constraints. In Section 3 we provide some analytical results, including a proof of equivalence between the polyhedra described by the two strongest formulations. Section 4 reports computational experiments where we compare the

LP relaxation bounds and branch-and-cut results. Finally, in Section 5, we give some concluding remarks and discuss further research topics.

2. Integer programming formulations

Let us consider a directed graph $G = (V, A)$ with the set of nodes V and the set of arcs A. Let n and m be the cardinality of V and A, respectively. A path is a sequence of nodes $v_1, \ldots, v_k$, and is said to be elementary if no node appears in the path more than once. A cycle, or tour, is a path with $v_1 = v_k$. We denote by $\delta^+(i)$ and $\delta^-(i)$ the set of outgoing and incoming arcs of node i, by $\delta^+(S)$ and $\delta^-(S)$ the arcs leaving/entering the set $S \subseteq V$, and by $A(S)$ the set of arcs with both ends in $S \subseteq V$. Let us also define $V_i := V \setminus \{i\}$, $V_{ij} := V \setminus \{i, j\}$, and, for any arc set $B \subseteq A$, we define $x(B) := \sum_{b \in B} x_b$. In all the formulations, it is assumed w.l.o.g. that $|\delta^-(s)| = |\delta^+(t)| = 0$.

A standard integer programming formulation to determine a shortest path from node s to node t is the following:

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

$$\sum_{(i,j) \in \delta^+(i)} x_{ij} - \sum_{(j,i) \in \delta^-(i)} x_{ji} = \begin{cases} 1 & \text{if } i = s \\ -1 & \text{if } i = t \\ 0 & \text{else} \end{cases} \qquad \forall i \in V \tag{2}$$

$$\sum_{(i,j) \in \delta^+(i)} x_{ij} \leq 1 \qquad \forall i \in V \tag{3}$$

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

where $c_{ij} \in \mathbb{R}$ are the arc costs, and x_{ij} are binary arc variables that take value 1 if the arc (i, j) belongs to the path. Constraints (2) are flow conservation constraints, while Constraints (3) ensure that the outgoing degree of each node is at most one. When the costs c_{ij} induce negative cycles on G, i.e., there is a subtour such that the total cost of its arcs is negative, this system of inequalities is not sufficient to guarantee the elementarity of the path. Thus, additional constraints (and possibly variables) are necessary to prevent subtours.

Notice that the crucial difference with respect to problems in which the path has to be Hamiltonian is the absence of the degree constraints:

$$\sum_{(i,j) \in \delta^+(i)} x_{ij} = \sum_{(j,i) \in \delta^-(i)} x_{ji} = 1 \qquad \forall i \in V_{st}.$$

We now describe different sets of constraints and variables that can be added to Formulation (1)–(4) to obtain a valid integer programming formulation for the ESPP.

2.1. Dantzig-Fulkerson-Johnson (DFJ)

For the TSP, success has been achieved with strong formulations with exponentially many constraints. It is possible to write a formulation for the ESPP based on the classical Dantzig-Fulkerson-Johnson subtour elimination constraints [10], adding to the basic formulation (1)–(4) the following inequalities:

$$\sum_{(i,j)\in A(S)} x_{ij} \leq |S| - 1 \qquad \forall S \subseteq V_{st}, |S| \geq 2. \tag{5}$$

In each subset S, subtours are prevented ensuring that the number of arcs in S which are selected is smaller than the number of nodes in S. This formulation includes $O(m)$ variables and $O(2^n)$ constraints.

Observation. For Hamiltonian path problems, due to the degree constraints, the DFJ subtour eliminations constraints can be equivalently written in the cutset form:

$$\sum_{(i,j)\in\delta^+(S)} x_{ij} \geq 1 \qquad \forall S \subset V, |S| \geq 2. \tag{6}$$

For the ESPP, Constraints (6) are valid only for subsets S with $s \in S$ and $t \notin S$. Moreover, they are *not* sufficient to prevent all subtours.

Example 1. Consider the solution depicted in Figure 1, assuming it is a complete graph and that only the arcs with $x_{ij} = 1$ are drawn. The solution does not violate any inequality (6) for any set S containing s, since $x(\delta^+(S)) = 1$ for any such S, although it contains a (disconnected) subtour. On the other hand, notice that the inequality (6) is not valid for S', although it does not contain subtours, due to $x(\delta^+(S')) = 0$.

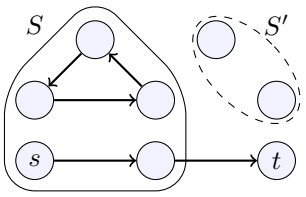

Figure 1: An example where Constraints (6) are not sufficient to prevent subtours.

2.2. Generalized cutset inequalities (GCS)

DFJ Constraints (6) can be adapted to the ESPP by replacing the constant right-hand side with a variable expression. This approach is used for a symmetric version of ESPPRC by Jepsen et al. [25] and applied to the asymmetric ESPP by Drexl and Irnich [14]. Similar subtour elimination constraints are also

used in branch-and-cut algorithms for the VRP [29] or variants of the TSP with profits [16, 26]. We refer to them as *generalized cutset inequalities* (*GCS*):

$$\sum_{(i,j)\in\delta^+(S)} x_{ij} \geq \sum_{(k,j)\in\delta^+(k)} x_{kj} \qquad \begin{array}{r}\forall k \in S, \forall S \subseteq V_{st},\\ |S| \geq 2.\end{array} \tag{7}$$

Constraints (7) prevent subtours by ensuring that, for each subset S, the number of selected arcs leaving S is not smaller than the number of selected arcs outgoing from any node in S. In an integer solution, this means that the cut induced by S must contain at least one arc if at least one node in S belongs to the s-t path, while, if S does not contain any node in the s-t path, the constraint is the trivial inequality. The number of variables in the formulation is $O(m)$, while the number of constraints is $O(n2^n)$.

Constraints (7) can be shown to be equivalent to a strengthened version of (5).

Proposition 2. *Constraints* (7) *can be rewritten as:*

$$\sum_{(i,j)\in A(S)} x_{ij} \leq \sum_{i\in S\setminus\{k\}} \sum_{(i,j)\in\delta^+(i)} x_{ij} \qquad \begin{array}{r}\forall k \in S, \forall S \subseteq V_{st},\\ |S| \geq 2.\end{array} \tag{8}$$

Proof. For all $k \in S$, $x(\delta^+(k)) \leq x(\delta^+(S)) = x(\delta^+(S)) + x(A(S)) - x(A(S)) = \sum_{i\in S} x(\delta^+(i)) - x(A(S))$. □

These inequalities can be interpreted as imposing that the number of selected arcs in a subset S is strictly smaller than the number of nodes in S that belong to the s-t path.

2.3. Sequential formulation (MTZ)

To derive an extended formulation à la Miller, Tucker and Zemlin [28] (hereafter MTZ) it is enough to introduce, for each node, an auxiliary variable that can be viewed as the position of the node along the path and a constraint for each arc:

$$t_j \geq t_i + 1 + (n-1)(x_{ij} - 1) \qquad \begin{array}{r}\forall (i,j) \in A,\\ i \neq s, j \neq t.\end{array} \tag{9}$$

For the Asymmetric TSP (ATSP), this formulation is well-known to give poor linear relaxation bounds. However, it is very compact, as it requires only $O(m)$ additional constraints and $O(n)$ auxiliary variables.

2.4. Reformulation-linearization based formulation (RLT)

From the following nonlinear reformulation of the MTZ formulation:

$$t_j x_{ij} = (t_i + 1)x_{ij} \qquad \forall (i,j) \in A, i \neq s \tag{10}$$

$$t_j x_{sj} = x_{sj} \qquad \forall (s,j) \in \delta^+(s) \tag{11}$$

$$1 \leq t_i \leq n-1, \qquad i \in V_s \tag{12}$$

Haouari et al. [21] apply a partial Sherali-Adams *reformulation-linearization technique* [33] to obtain the following stronger formulation for the ESPP:

$$\alpha_{ij} = \beta_{ij} + x_{ij} \qquad \forall (i,j) \in A \tag{13}$$

$$x_{sj} + \sum_{\substack{(i,j)\in\delta^-(j)\\ i\neq s}} \alpha_{ij} - \sum_{(j,i)\in\delta^+(j)} \beta_{ji} = 0 \qquad \begin{array}{r}\forall j \in V_t,\\ (s,j) \in \delta^+(s)\end{array} \tag{14}$$

$$\sum_{(i,j)\in\delta^-(j)} \alpha_{ij} - \sum_{(j,i)\in\delta^+(j)} \beta_{ji} = 0 \qquad \begin{array}{r}\forall j \in V_{st},\\ (s,j) \notin \delta^+(s)\end{array} \tag{15}$$

$$\alpha_{ij} \leq (n-1)x_{ij} \qquad \forall (i,j) \in A, i \neq s \tag{16}$$

$$x_{ij} \leq \beta_{ij} \qquad \forall (i,j) \in A, i \neq s \tag{17}$$

$$\alpha_{ij} \geq 0, \beta_{ij} \geq 0 \qquad \forall (i,j) \in A, \tag{18}$$

where the bilinear terms are linearized introducing the variables $\alpha_{ij} := t_j x_{ij}$ and $\beta_{ij} := t_i x_{ij}$ and Constraints (16)–(17). On a given selected arc $(i,j) \in A$, the variables β_{ij} and α_{ij} can be interpreted respectively as the position of the nodes i and j along the path. This extended formulation requires $O(m)$ constraints and $O(m)$ auxiliary variables.

2.5. Single-flow formulation (SF)

A formulation similar to the single-flow ATSP formulation of Gavish and Graves [18] can be obtained introducing an auxiliary flow q to be delivered to the nodes belonging to the s-t path. In addition, variables z_k are added to the formulation:

$$q_{ij} \leq (n-1)x_{ij} \qquad \forall (i,j) \in A \tag{19}$$

$$\sum_{(s,j)\in\delta^+(s)} q_{sj} = \sum_{k\in V_s} z_k \tag{20}$$

$$\sum_{(i,k)\in\delta^-(k)} q_{ik} - \sum_{(k,j)\in\delta^+(k)} q_{kj} = z_k \qquad \forall k \in V_s \tag{21}$$

$$\sum_{(i,k)\in\delta^-(k)} x_{ik} = z_k \qquad \forall k \in V_s \tag{22}$$

$$q_{ij} \geq 0 \qquad \forall (i,j) \in A \tag{23}$$

$$z_k \in \{0,1\} \qquad \forall k \in V_s. \tag{24}$$

Constraints (19) impose that the auxiliary flow is positive only over the arcs where $x_{ij} = 1$. The auxiliary flow leaving from the node s has value equal to the number of nodes that are reached by the s-t path. Constraints (21) ensure that the balance of the auxiliary flow on each node is equivalent to z_k, which, according to Constraint (22), is either 1, if node k is in the s-t path, or 0 otherwise.

2.6. Multicommodity-flow formulation (MCF)

An extension of the single-flow formulation is obtained by disaggregating the auxiliary flow into $n-1$ unitary flows. Subtours are prevented by enforcing one unit of a distinct auxiliary flow from s to each node that belongs to the s-t path:

$$q_{ij}^k \leq x_{ij} \qquad \forall k \in V_s, \ (i,j) \in A \tag{25}$$

$$\sum_{(i,j)\in\delta^+(i)} q_{ij}^k - \sum_{(j,i)\in\delta^-(i)} q_{ji}^k = \begin{cases} z_k & \text{if } i = s \\ -z_k & \text{if } i = k \\ 0 & \text{else} \end{cases} \qquad \forall i \in V, \ \forall k \in V_s \tag{26}$$

$$\sum_{(i,k)\in\delta^-(k)} x_{ik} = z_k \qquad \forall k \in V_s \tag{27}$$

$$\sum_{(s,j)\in\delta^+(s)} x_{sj} = 1 \tag{28}$$

$$\sum_{(i,t)\in\delta^-(t)} x_{it} = 1 \tag{29}$$

$$q_{ij}^k \geq 0 \qquad \forall k \in V_s, \ (i,j) \in A \tag{30}$$

$$z_k \in \{0,1\} \qquad \forall k \in V_s. \tag{31}$$

The formulation includes $O(nm)$ additional variables and constraints. This extended formulation is introduced, for the ESPP, by Ibrahim et al. [22], and it is very similar to classic multi-commodity flow formulations for the ATSP proposed by Wong [37] and Claus [9].

2.7. Overview

In Table 1 we summarize the presented formulations. To the best of our knowledge, formulations *MTZ* and *SF* have not been previously considered for the ESPP, although similar ones are well known for TSP or VRP variants.

Table 1: A summary of the considered formulations.

	number of variables	number of constraints	description
DFJ	$O(m)$	$O(2^n)$	Dantzig-Fulkerson-Johnson (5)
GCS	$O(m)$	$O(n2^n)$	generalized cutsets (7)
MTZ	$O(m)$	$O(m)$	Miller-Tucker-Zemlin (9)
RLT	$O(m)$	$O(m)$	reformulation-linearization (13)–(18)
SF	$O(m)$	$O(m)$	single-flow (19)-(24)
MCF	$O(nm)$	$O(nm)$	multi-commodity flow (25)-(31)

3. Polyhedral results

Let us describe some analytical results for the considered ESPP formulations.

Proposition 3. *Formulation* MCF *is stronger than formulation* SF*.*

Proof. Constraints (19)–(21) can be obtained from *MCF* simply aggregating Constraints (25)–(26) over $k \in V_s$, and then substituting $\sum_{k \in V_s} q_{ij}^k$ with q_{ij}. The example in Figure 2 shows that the inclusion is strict. □

Proposition 4. *Formulation* GCS *is stronger than formulation* DFJ*.*

Proof. The result follows by considering *GCS* as stated in (8), whose right-hand side is obviously smaller or equal to $|S| - 1$, right-hand side in (5), and the inclusion is strict by the example in Figure 2. □

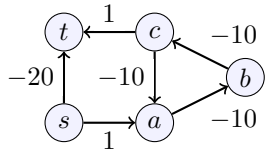

Figure 2: Example proving strict inclusion for Proposition 3 and 4. With *GCS* and *MCF*, the LP optimal solution is the one with $x_{st} = 1$ and optimal value -20. With *SF*, the optimal solution has value -26, with $x_{st} = x_{ca} = \frac{1}{4}$, $x_{sa} = x_{ct} = \frac{3}{4}$ and $x_{ab} = x_{bc} = 1$. With *DFJ*, the solution has value -40, with $x_{st} = 1$ and a disconnected subtour with $x_{ab} = x_{bc} = x_{ca} = \frac{2}{3}$.

Showing that formulation *MCF* is as tight as *GCS* requires to calculate the projection of the *MCF* extended formulation into the space of the x variables. We will make use of a strong result presented by Padberg and Sung in [31], and follow a similar approach to the equivalence proofs therein.

Theorem 5. *The projection of the* MCF*-polytope onto the x-space is equivalent to the* GCS*-polytope.*

Proof. Recall that $|\delta^-(s)| = |\delta^+(t)| = 0$. The variables z_k can be projected out

of (25)–(31) so that MCF can be rewritten as:

$$q_{ij}^k \leq x_{ij} \qquad \forall k \in V_s, \; \forall (i,j) \in A \tag{32}$$

$$\sum_{(i,j)\in\delta^+(i)} q_{ij}^k - \sum_{(j,i)\in\delta^-(i)} q_{ji}^k = 0 \qquad \forall i \in V_s, i \neq k, \; \forall k \in V_s \tag{33}$$

$$\sum_{(s,j)\in\delta^+(s)} q_{sj}^k = \sum_{(i,k)\in\delta^-(k)} x_{ik} \qquad \forall k \in V_s \tag{34}$$

$$\sum_{(j,k)\in\delta^-(k)} q_{jk}^k = \sum_{(i,k)\in\delta^-(k)} x_{ik} \qquad \forall k \in V_s \tag{35}$$

$$\sum_{(i,k)\in\delta^+(k)} x_{ik} = \sum_{(i,k)\in\delta^-(k)} x_{ik} \qquad \forall k \in V_{st} \tag{36}$$

$$\sum_{(s,j)\in\delta^+(s)} x_{sj} = 1 \tag{37}$$

$$\sum_{(i,t)\in\delta^-(t)} x_{it} = 1 \tag{38}$$

$$q_{ij}^k \geq 0, x_{ij} \geq 0 \qquad \forall (i,j) \in A, k \in V_s. \tag{39}$$

In order to compare *MCF* and *GCS*, we need to project out also the q-variables of the MCF formulation. Let us define the sets:

$$\begin{aligned}
&X = \{\underline{x} \in \mathbb{R}^m \mid \underline{x} \text{ satisfies (36), (37) and (38)}\},\\
&P_{GCS} = \{\underline{x} \in X \mid \underline{x} \text{ satisfies (7)}\},\\
&P_{MCF} = \{(\underline{x}, \underline{q}) \in \mathbb{R}^{mn} \mid (\underline{x}, \underline{q}) \text{ satisfies (32)–(39)}\},\\
&Proj_x(P_{MCF}) = \{\underline{x} \in X \mid \exists\ \underline{q} \text{ s.t. } (\underline{x}, \underline{q}) \in P_{MCF}\},
\end{aligned}$$

where P_{GCS} is the *GCS*-polytope, P_{MCF} is the *MCF*-polytope and $Proj_x(P_{MCF})$ is its projection onto the x-space. It is convenient to rewrite Constraints (32)–(35) in matrix form as follows:

$$B\underline{x} + M\underline{q} = \underline{0} \tag{40}$$

$$-D\underline{x} + I\underline{q} \leq \underline{0} \tag{41}$$

$$\underline{x}, \underline{q} \geq \underline{0}. \tag{42}$$

Equation (40) corresponds to (33)–(35), while (41) corresponds to (32). The matrices B, M, D and I can be decomposed according to the index k. Each block M_k represents the node-arc incidence matrix of the graph G. I_k are identity matrices of dimension $m \times m$. Each submatrix B_k has zeros everywhere, except for the row corresponding to node s, with entries of value -1 for each arc in $\delta^-(k)$, and the row corresponding to node k, with $+1$ entries for each arc in $\delta^-(k)$. Each row of D_k corresponds to a variable q_{ij}^k and has zeros everywhere, except for a $+1$ in the column associated with variable x_{ij}.

The projection onto the x-space of the polytope P_{MCF} defined by (40)–(41) can be obtained as follows (see, e.g., [4]):

$$Proj_x(P_{MCF}) = \{\underline{x} \in X \mid (\underline{u}B - \underline{v}D - \underline{w})\underline{x} \leq 0 \quad \forall (\underline{u}, \underline{v}, \underline{w}) \in C\}, \tag{43}$$

where C is the cone defined as:

$$C = \{(\underline{u}, \underline{v}, \underline{w}) \mid \underline{u}M + \underline{v}I \geq 0, \underline{v} \geq 0, \underline{w} \geq 0\}.$$

The result allows us to carry out the comparison between P_{GCS} and $Proj_x(P_{MCF})$ simply by finding a system of generators for the cone C.

From the inequalities $\underline{w} \geq \underline{0}$ we obtain extreme rays of the form $\underline{u} = \underline{0}$, $\underline{v} = \underline{0}$, $\underline{w} = \underline{e}_i$, where $\underline{e}_i$ is the i-th standard basis vector of $\mathbb{R}^m$, that yield the nonnegativity constraints

$$x_{ij} \geq 0 \qquad \forall (i,j) \in A. \tag{44}$$

This allows us to restrict our following study to the cone C' defined as:

$$C' = \{(\underline{u}, \underline{v}) \mid \underline{u}M + \underline{v}I \geq 0, \underline{v} \geq 0\}.$$

Exploiting the decomposition of M, we can work on the even smaller cones:

$$C_k = \{(\underline{u}^k, \underline{v}^k) \mid \underline{u}^k M_k + \underline{v}^k \geq 0, \underline{v}^k \geq 0\}. \tag{45}$$

Due to (43), once we have the system of generators $(\underline{u}^k, \underline{v}^k)$ for each cone C_k, the constraints in the x-space are obtained by calculating $(\underline{u}^k B_k - \underline{v}^k D_k)\underline{x} \leq \underline{0}$ for each $k \in V_s$.

According to Proposition 6 in [31], a full system of generators of a cone C_k defined as in (45), where M_k is a node-arc incidence matrix of a digraph, is given by:
- a basis of its lineality space, of the form:

$$\underline{u}^k = \pm \underline{e}, \ \underline{v}^k = \underline{0},$$

where $\underline{e}$ is the all-ones vector, that in our case translate to the trivial equality $\underline{0} = \underline{0}$, and
- the extreme rays, given by all the positive multiples of the vector $(\underline{u}^k, \underline{v}^k)$ such that:

$$(i) \ \ u_i^k = 0 \quad \forall i \in V, \qquad v_{ij}^k = \begin{cases} 1 & \text{for one } (i,j) \in A \\ 0 & \text{otherwise} \end{cases}$$

$$(ii) \ u_i^k = \begin{cases} 1 & \forall i \in S, \\ 0 & \text{otherwise} \end{cases} \qquad v_{ij}^k = \begin{cases} 1 & \forall \ i \in \bar{S}, j \in S, \\ 0 & \text{otherwise} \end{cases}$$

$$(iii) \ u_i^k = \begin{cases} -1 & \forall i \in S, \\ 0 & \text{otherwise} \end{cases} \qquad v_{ij}^k = \begin{cases} 1 & \forall \ i \in S, j \in \bar{S}, \\ 0 & \text{otherwise} \end{cases}$$

for any $S \subseteq V$, where $\bar{S} = V \setminus S$. The extreme rays of the form (i) give rise to nonnegativity constraints.

From the extreme rays given by (ii) and (iii) we obtain the inequalities:

$$-x(\delta^-(S)) + u_k x(\delta^-(k)) - u_s x(\delta^-(k)) \leq 0 \tag{46}$$

$$-x(\delta^+(S)) - u_k x(\delta^-(k)) + u_s x(\delta^-(k)) \leq 0 \tag{47}$$

where $u_i = 1$ if $i \in S$, and 0 otherwise. For both (46) and (47), we can distinguish four cases depending on whether s and k are in S, thus whether u_s, u_k are 0 or 1. If both s and k are in S, or neither of them is, the inequality is implied by the nonnegativity constraints (44). If only the coefficient with negative sign is nonzero, the corresponding inequality is, again, redundant. Therefore, the only meaningful cases are the following:

$$x(\delta^-(S)) \geq x(\delta^-(k)) \qquad \forall S \subseteq V, s \notin S, \forall k \in S \tag{48}$$

$$x(\delta^+(S)) \geq x(\delta^-(k)) \qquad \forall S \subseteq V, s \in S, \forall k \notin S. \tag{49}$$

We have established so far that $Proj_x(P_{MCF})$ is fully described by the nonnegativity constraints and Constraints (48)–(49). This set of inequalities can be shown to be equivalent to:

$$x(\delta^+(S)) \geq x(\delta^+(k)) \qquad \forall S \subseteq V_{st}, k \in S. \tag{50}$$

Constraints (48) and (49) are equivalent, due to the fact that $x(\delta^+(S)) = x(\delta^-(\bar{S}))$. Let us then consider only (48). For $k = t$, the inequality is trivially satisfied by all $x \in X$, thus redundant. For $k \neq t$ and $t \notin S$, we obtain exactly the inequalities in (50), since by (36)–(38), we have that $x(\delta^-(k)) = x(\delta^+(k))$ and $x(\delta^-(S)) = x(\delta^+(S))$ for any S containing neither s nor t. If $k \neq t$ and $t \in S$, it suffices to observe that, since $\delta^+(t) = 0$, the inequality $x(\delta^-(S)) \geq x(\delta^-(k))$ is implied by $x(\delta^-(S \setminus \{t\})) \geq x(\delta^-(k))$, which, again, can be rewritten in the form (50).

Hence, the projection of P_{MCF} onto the x-space is given by

$$Proj_x(P_{MCF}) = \{\underline{x} \in X \mid \underline{x} \text{ satisfies (36)–(38) and (50)}\},$$

and it follows that $Proj_x(P_{MCF}) = P_{GCS}$. □

From this result and Proposition 4, it also follows that formulation *MCF* is stronger than formulation *DFJ*.

4. Computational comparison

Let us now compare the described formulations with respect to their LP relaxation bounds and their behavior within an exact branch-and-cut framework. Formulation *DFJ* is not included in the tests, as it is clearly dominated by *GCS*.

Four types of instances are considered in the tests. The first benchmark set consists of instances from the pricing phase of the unsplittable flow problem

in [1, 34] on small-sized networks from the SNDlib [30], namely, the topologies `atlanta`, `france`, `geant`, `germany` and `nobel-us`.

The second one is a set of small to medium-sized random-cost graphs, either sparse (`rnd-s`) or dense (`rnd-d`). The graphs for the instances in `rnd-s` are generated by building a connected component including all the nodes, and then randomly adding arcs until the desired density is reached. The instances in `rnd-d` are the dense instances in [13], with random arc costs on a complete graph.

The third benchmark set (`prc`) contains the pricing instances in [13]. It consists of small and medium-size pricing problems from a column generation algorithm for the asymmetric m-salesmen TSP at the first (`f`), penultimate (`p`) and last (`l`) pricing iterations.

The fourth set (`rome99`) contains a part of the directed road network of the city of Rome, Italy, used in the 9th DIMACS Implementation Challenge on Shortest Paths [11]. Since all the arcs have a positive cost, representing the distance in meters, we flip their sign (i.e., we solve a longest path problem over the original graph). To generate distinct instances over the same graph, we sample randomly 30 (s, t) pairs from V.

Table 2 summarizes the features of the test instances. For each subset, we have 30 instances, for a total of 690.

Table 2: Description of the instances.

	n	m	range of arc costs	number of instances
`nobel-us`	14	42	[−10000,10000]	30
`atlanta`	15	44	[−10000,10000]	30
`geant`	22	72	[−10000,10000]	30
`france`	25	90	[−10000,10000]	30
`germany`	50	176	[−10000,10000]	30
`rnd-s`	50/100/200	164/660/2654	[−1000,1000]	30/30/30
`rnd-s`	500/1000	16634/66601	[−1000,1000]	30/30
`rnd-d`	25/50/100	600/2450/9900	[−1000,1000]	30/30/30
`prc-f`	27/52/102	702/2652/10302	$[-10^8, -9.5 \cdot 10^7]$	30/30/30
`prc-p`	27/52/102	702/2652/10302	$[-4 \cdot 10^4, 5.2 \cdot 10^6]$	30/30/30
`prc-l`	27/52/102	702/2652/10302	$[-4 \cdot 10^4, 5.2 \cdot 10^6]$	30/30/30
`rome99`	3353	8870	$[-13000, -1]$	30

4.1. Linear programming relaxation bounds

The LP relaxation bounds are computed constructing the complete model for the extended formulations *MTZ*, *RLT* and *SF*. For formulations *GCS* and *MCF* we use a Min Cut-based separation procedure (its implementation details are left to the next section). Note that we use a delayed row-generation algorithm also to solve *MCF* since its size is rather large, although polynomial, and preliminary experiments indicated this is an effective strategy. The tests are carried out with IBM Ilog Cplex 12.6 on an Intel Xeon E5645 @2.40GHz.

Table 3 reports the average gap of the linear relaxation bounds with respect to the optimal integer values, computed as $100\frac{|LB-Opt|}{|LB|}$, the number of optimal integer solutions, and the average computing time. The results confirm that the LP bounds of *MCF* and *GCS* are equivalent, and show that they are by far the tightest formulations. *MCF* and *GCS* have gap 0 on 40% of the instances and find an optimal integer solution on almost 30% of the instances. The remaining formulations have a relaxation with gap 0 in less than 10% of the considered instances, and find an optimal integer solution in less than 2% of the cases. Formulations *RLT* and *SF* provide similar bounds. Considering the computing time, *MTZ* is solved more quickly than all the other formulations, while *RLT* and *SF* are challenging for large networks. On the largest instances, the LP relaxation of the *MCF* formulation could not be solved within the time limit of 1200 seconds.

It is worthwhile to point out that, even in the cases where the bounds are very good with all the formulations (e.g., on the `prc-f` instances), weaker formulations provide solutions with many more fractional values, as reflected by the smaller number of optimal integer solutions.

Table 3: Average LP relaxation gaps (%) and number of optimal integer solutions. Missing values are due to time limit. `rome99` instances are not included, since optima are not available.

	GCS			*MTZ*			*RLT*			*SF*			*MCF*		
	%gap	opt	time	%gap	opt	time	%gap	opt	time	%gap	opt	time	%gap	opt	time
`nobel-us`	26.5	19	0.0	72.0	0	0.0	69.1	0	0.0	69.7	0	0.0	26.5	19	0.0
`atlanta`	9.9	23	0.0	42.8	0	0.0	36.5	0	0.0	37.5	0	0.0	9.9	23	0.0
`geant`	5.5	14	0.0	32.0	0	0.0	30.9	0	0.0	31.1	0	0.0	5.5	14	0.0
`france`	4.3	22	0.0	64.5	0	0.0	59.9	0	0.0	60.4	0	0.0	4.3	22	0.0
`germany`	2.0	2	0.4	31.5	0	0.0	31.3	0	0.0	31.3	0	0.0	2.0	2	0.4
`rnd-s-50`	1.7	4	0.1	13.1	0	0.0	12.5	0	0.0	12.6	0	0.0	1.7	4	0.5
`rnd-s-100`	0.2	2	0.2	1.5	0	0.0	1.5	0	0.0	1.5	0	0.0	0.2	2	7.1
`rnd-s-200`	0.0	0	0.5	0.4	0	0.0	0.4	0	0.5	0.4	0	0.4	0.0	0	345.5
`rnd-s-500`	0.0	0	5.6	0.1	0	0.8	0.1	0	68.5	0.1	0	29.7	–	–	–
`rnd-s-1000`	0.0	2	51.9	0.0	0	37.4	0.0	0	792.5	0.0	0	319.8	–	–	–
`rnd-d-100`	0.0	5	0.2	0.0	3	0.1	0.0	3	4.1	0.0	3	3.5	0.0	5	84.2
`rnd-d-25`	0.2	15	0.0	0.6	3	0.0	0.5	4	0.0	0.5	3	0.0	0.2	15	0.0
`rnd-d-50`	0.0	6	0.0	0.1	3	0.0	0.1	3	0.1	0.1	3	0.2	0.0	6	2.4
`prc-f-25`	0.0	13	0.0	0.0	0	0.0	0.0	0	0.0	0.0	0	0.0	0.0	13	0.7
`prc-f-50`	0.0	3	0.2	0.0	0	0.0	0.0	0	0.2	0.0	0	0.2	0.0	3	44.9
`prc-f-100`	0.0	0	2.5	0.0	0	0.0	0.0	0	11.0	0.0	0	4.1	–	–	–
`prc-p-25`	10.1	15	0.1	91.5	0	0.0	87.1	0	0.0	87.4	0	0.0	10.1	15	2.6
`prc-p-50`	8.2	6	1.1	80.6	0	0.0	77.5	0	0.2	77.6	0	0.2	8.2	6	89.3
`prc-p-100`	1.8	0	14.0	60.3	0	0.0	42.1	0	11.6	42.3	0	4.1	–	–	–
`prc-l-25`	1.4	24	0.2	86.5	0	0.0	80.8	0	0.0	81.1	0	0.0	1.4	24	2.8
`prc-l-50`	2.3	8	2.1	55.6	0	0.0	49.8	0	0.2	49.9	0	0.2	2.3	8	53.6
`prc-l-100`	2.2	0	12.1	60.4	0	0.0	42.1	0	11.0	42.4	0	4.0	–	–	–

4.2. Branch-and-cut

Since we aim at integer solutions, let us compare the behavior of a state-of-the-art MIP solver using the considered formulations. The formulations and the

separation procedures are implemented in C++ with IBM Ilog Cplex/Concert 12.6, using default settings for the branch-and-cut.

For the polynomial-size extended formulations *MTZ*, *RLT* and *SF*, the full model is built.

For *GCS*, we report results obtained with two different separation routines. In the approach denoted by *GCS-StrongComp*, the separation is carried out, on both fractional and integer solutions, identifying the strongly connected components in the support graph induced by the variables x_{ij}. This can be done in a $O(n+m)$ running time with Tarjan's algorithm [35]. Once a strong component S has been found, it is enough to check if Constraint (7) is violated for any of the nodes in S. This separation procedure is efficient, but not guaranteed to find all the violated inequalities on fractional solutions. Correctness is preserved by the fact that the procedure is exact for integer solutions. In the approach denoted by *GCS-MinCut*, the separation is carried out on fractional solutions by solving a sequence of Min Cut (or Max Flow) problems between each node and t. Solving $n-1$ Min Cut problems yields an overall worst-case complexity of $O(n^3\sqrt{m})$ using Goldberg-Tarjan's highest-label preflow-push algorithm [20]. This way, all violated inequalities are identified, although with a higher computational cost. Note that, on integer solutions, the faster strong component-based procedure is sufficient, and, on fractional solutions, it is computationally convenient to try the strong components procedure first, and resort to the Min-Cut separation only if the heuristic finds no violated inequality. The same separation procedures are also used for *MCF*: when a violation is found for a node k, we add the full set of Constraints (25)–(27) corresponding to that node.

To solve the Min Cut problems and identify the strongly connected components, we use the efficient implementations in the open-source LEMON Graph Library 1.3 [12]. We refer the interested reader to [13] for additional considerations on the separation of subtour elimination constraints for the ESPP.

A remark is in order. In a branch-and-cut algorithm, the generation of the cutting planes must be balanced with respect to the branching: adding too many inequalities may hinder the solution of the LPs in the nodes, although better bounds result in fewer explored nodes. We use two parameters to control the trade-off between the quality of the lower bounds and the computing time to solve the LPs. Specifically, given a solution $\underline{x}$, we only consider the inequalities with a violation not smaller than ε (correctness is preserved on integer solutions for $\varepsilon < 1$), and we add at most ν of them (in particular, we select the first ν maximally violated). In Tables 4 and 5 we summarize a tuning procedure that is carried out on a subset of 120 medium-size instances to identify the best parameters for *GCS-StrongComp* and *GCS-MinCut*. We report the geometric mean of the time to optimality, the number of nodes and the number of added cuts. The values are normalized, for each instance, with respect to the results with $\varepsilon = 0.001$ and $\nu = 1$. The tables indicate that, in both cases, it is convenient to add all the inequalities that are violated by the given tolerance ε. According to these results, in the following experiments, we use $\varepsilon = 0.8$ and $m = \texttt{all}$ for *GCS-MinCut*, and $\varepsilon = 0.2, m = \texttt{all}$ for *GCS-StrongComp*.

Concerning the row generation for *MCF*, recall that for every violation we

Table 4: Tuning of ε and ν for *GCS-StrongComp*.

ε	0.001	0.1	0.2	0.4	0.8
ν	time/nodes/cuts	time/nodes/cuts	time/nodes/cuts	time/nodes/cuts	time/nodes/cuts
1	1.00/1.00/1.00	1.00/1.05/1.00	0.99/1.17/0.97	0.98/1.22/0.92	1.00/1.70/0.90
5	0.90/0.68/1.11	0.90/0.73/1.11	0.87/0.81/1.10	0.89/0.94/1.08	0.86/1.20/1.06
10	0.83/0.60/1.06	0.79/0.61/1.06	0.79/0.66/1.05	0.80/0.74/1.02	0.79/1.01/1.01
20	0.85/0.58/1.08	0.78/0.60/1.07	0.77/0.66/1.05	0.77/0.74/1.04	0.79/0.95/1.02
`all`	0.77/0.62/1.08	0.75/0.63/1.06	0.73/0.65/1.05	0.77/0.76/1.05	0.75/0.96/1.00

Table 5: Tuning of ε and ν for *GCS-MinCut*.

ε	0.001	0.1	0.2	0.4	0.8
ν	time/nodes/cuts	time/nodes/cuts	time/nodes/cuts	time/nodes/cuts	time/nodes/cuts
1	1.00/1.00/1.00	0.94/1.32/1.01	0.89/1.61/0.98	0.83/1.80/0.90	0.80/2.93/0.85
5	0.82/0.76/1.19	0.78/1.00/1.19	0.73/1.22/1.13	0.70/1.43/1.04	0.68/2.02/1.02
10	0.79/0.75/1.21	0.71/0.90/1.20	0.67/1.04/1.15	0.64/1.24/1.02	0.61/1.68/0.95
20	0.76/0.76/1.16	0.63/0.88/1.10	0.63/1.03/1.11	0.60/1.20/1.01	0.59/1.58/0.95
`all`	0.72/0.78/1.20	0.65/0.97/1.20	0.60/0.93/1.11	0.62/1.23/1.05	0.56/1.60/0.94

need to add a set of $O(m)$ constraints. For this reason, we decide to add only the inequalities corresponding to the node with maximum violation. The choice of the separation algorithm and of ε does not have a significant impact; in the following experiments, we set $\varepsilon = 0.2$ and use the strong component-based separation procedure.

In Table 6 we summarize the results over the whole set of test instances. Column "opt" reports the number of instances that are solved to proven optimality within the time limit of 1200 seconds. Column "time" reports the average computing time. Column "nodes" reports the average number of explored nodes in the branch-and-bound tree. Column "cuts" reports the average number of *GCS* inequalities added by the separation procedures.

First of all, we can observe that the small-size pricing instances from the SNDlib are easy for all the formulations. It is also worth noting that, for graphs of similar size, the `rnd` instances are typically much easier than the `prc` ones. For the TSP, instances with random costs are known to be rather easy to solve [2], and a similar effect may take place for the ESPP. None of the approaches is able to solve to proven optimality any of the instances in `rome99`.

GCS-StrongComp solves to optimality all the instances in the set, except for those in `rome99`, and proves to be by far the best choice for the largest instances. The *GCS-MinCut* approach has difficulties on large-size instances, where most of the computing time is spent in the separation phase. In particular, optimality cannot be proven on 4 instances of `rand-1000`, and 2 instances of `prc-l-100`.

Compared to the extended formulations, *GCS-StrongComp* is a clear winner on all benchmark sets, where it is often more than one order of magnitude faster. Despite the typically better linear relaxation bounds, formulation *RLT* is usually not faster than *SF*, that allows the solver to reach proven optimality

for a larger number of instances. Interestingly, while rather ineffective overall, formulation *MTZ* yields good results on the `rnd` instances. This might be due to the fact that, on the `rnd` set, all the extended formulations have similarly good linear relaxation bounds, and it probably pays off to have an LP of smaller size.

Formulation *MCF*, despite the tight linear relaxation, appears to be too heavy to be of practical interest. On small-sized instances, very few B&B nodes are necessary. However, this is not enough to overcome the computational load required by solving the linear relaxation: even with a row-generation approach, the size of the LP grows quickly. On graphs with 100 or more nodes, only a small fraction of the instances can be solved within the time limit. On the largest graphs, the instances in `rnd-s-1000` and `rome99`, the solver quickly reaches the memory limit of 16 GB.

Figure 3 summarizes the computational experiments with a performance profile. On the y-axis, we report the fraction of all the instances that are solved to optimality within the time on the x-axis (in logarithmic scale).

GCS-StrongComp is the topmost curve, solving more than 85% of the instances within 10 seconds. *GCS-MinCut* is not far behind, although it is generally slower. Both solve around 95% of the instances within 1200 seconds. Formulations *RLT*, *SF* and *MTZ* yield similar results on the easiest instances, although, overall, only less than 70% of the instances are solved to optimality with *MTZ*, while *SF* and *RLT* reach, respectively, 90% and 87%. With the *MCF* approach (bottom curve), Cplex is already significantly slower on the easiest instances, and solves the smallest fraction of the instances (around 65%).

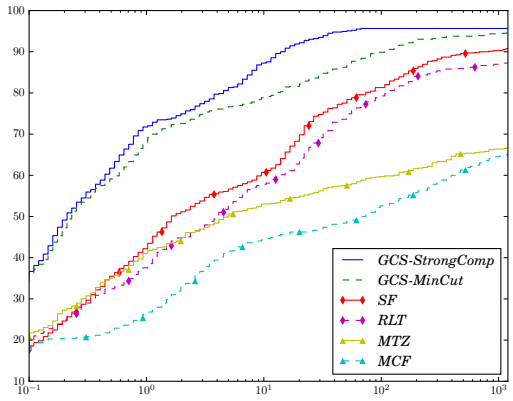

Figure 3: Fraction of instances solved to optimality within a given time (seconds). The x-axis is in logarithmic scale.

5. Conclusions

We have analyzed integer programming formulations for the ESPP, including formulations not yet appeared in the literature.

The polyhedral results in Section 3 provide a partial hierarchy among the ESPP formulations, and prove that the strong extended formulation *MCF* has a projection on the space of the arc variables which is equivalent to the polytope of the *GCS* formulation, that has exponentially many subtour elimination constraints.

It is also important to understand how effective the formulations are from a computational point of view. In this regard, we report a set of extensive computational experiments, suggesting that the extended formulations are inferior for all practical purposes, and the dynamic separation of subtour elimination constraints appears to be the best option when tackling the ESPP as an integer program. The *GCS* approach with the strong component-based separation procedure is able to solve small-sized instances in a few seconds and medium-sized instances, with up to 1000 nodes, within a minute, but it is still not sufficient to solve large-scale problems. When the implementation of a separation procedure is not possible or convenient, formulation *SF* is probably the best choice.

It seems likely that the development of good primal heuristics and additional strong valid inequalities, possibly extended from of the ATSP (e.g., 2-matching or comb inequalities), might further speed up the computing times and allow the solution of larger-sized instances. It might also be useful to borrow techniques from the typical approaches used for the ESPPRC, such as a preprocessing phase with the aim of reducing the search space.

Acknowledgments

The author would like to thank the two anonymous referees for the useful comments that helped improve the quality of the manuscript.

References

[1] Amaldi, E., Coniglio, S., Taccari, L., 2014. Maximum throughput network routing subject to fair flow allocation. In: ISCO. Vol. 8596 of Lecture Notes in Computer Science. Springer, pp. 1–12.

[2] Applegate, D. L., Bixby, R. E., Cook, W. J., Chvátal, V., 2006. The traveling salesman problem: a computational study. Princeton University Press.

[3] Balas, E., 1989. The prize collecting traveling salesman problem. Networks 19 (6), 621–636.

[4] Balas, E., 2005. Projection, lifting and extended formulation in integer and combinatorial optimization. Annals of Operations Research 140 (1), 125–161.

[5] Bauer, P., Linderoth, J. T., Savelsbergh, M. W., 2002. A branch and cut approach to the cardinality constrained circuit problem. Mathematical Programming 91 (2), 307–348.

[6] Beasley, J., Christofides, N., 1989. An algorithm for the resource constrained shortest path problem. Networks 19 (4), 379–394.

[7] Björklund, A., Husfeldt, T., Khanna, S., 2004. Approximating longest directed paths and cycles. In: Automata, Languages and Programming. Springer, pp. 222–233.

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

[9] Claus, A., 1984. A new formulation for the travelling salesman problem. SIAM Journal on Algebraic Discrete Methods 5 (1), 21–25.

[10] Dantzig, G., Fulkerson, R., Johnson, S., 1954. Solution of a large-scale traveling-salesman problem. Operations Research 2 (4), 393–410.

[11] Demetrescu, C., Goldberg, A. V., Johnson, D. S., 2009. The Shortest Path Problem: Ninth DIMACS Implementation Challenge. Vol. 74. AMS.

[12] Dezső, B., Jüttner, A., Kovács, P., 2011. LEMON – an open source C++ graph template library. Electronic Notes in Theoretical Computer Science 264 (5), 23–45.

[13] Drexl, M., 2013. A note on the separation of subtour elimination constraints in elementary shortest path problems. European Journal of Operational Research 229 (3), 595–598.

[14] Drexl, M., Irnich, S., 2014. Solving elementary shortest-path problems as mixed-integer programs. OR Spectrum 36 (2), 281–296.

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

[16] Fischetti, M., Gonzalez, J. J. S., Toth, P., 1998. Solving the orienteering problem through branch-and-cut. INFORMS Journal on Computing 10 (2), 133–148.

[17] Gabow, H. N., 2007. Finding paths and cycles of superpolylogarithmic length. SIAM Journal on Computing 36 (6), 1648–1671.

[18] Gavish, B., Graves, S. C., 1978. The travelling salesman problem and related problems. Working Paper GR-078-78, Massachusetts Institute of Technology.

[19] Gendreau, M., Laporte, G., 1998. A branch-and-cut algorithm for the undirected selective traveling salesman problem. Networks 32, 263–273.

[20] Goldberg, A. V., Tarjan, R. E., 1988. A new approach to the maximum-flow problem. Journal of the ACM (JACM) 35 (4), 921–940.

[21] Haouari, M., Maculan, N., Mrad, M., 2013. Enhanced compact models for the connected subgraph problem and for the shortest path problem in digraphs with negative cycles. Computers & Operations Research 40 (10), 2485–2492.

[22] 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 (3), 361–369.

[23] Irnich, S., Desaulniers, G., 2005. Shortest Path Problems with Resource Constraints. Springer, Ch. 2, pp. 33–65.

[24] Jepsen, M., Petersen, B., Spoorendonk, S., Pisinger, D., 2008. Subset-row inequalities applied to the vehicle-routing problem with time windows. Operations Research 56 (2), 497–511.

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

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

[27] Karger, D., Motwani, R., Ramkumar, G., 1997. On approximating the longest path in a graph. Algorithmica 18 (1), 82–98.

[28] Miller, C. E., Tucker, A. W., Zemlin, R. A., 1960. Integer programming formulation of traveling salesman problems. Journal of the ACM (JACM) 7 (4), 326–329.

[29] Naddef, D., Rinaldi, G., 2002. Branch-and-cut algorithms for the capacitated VRP. SIAM Monographs on Discrete Mathematics and Applications Philadelphia, PA, Ch. 3, pp. 53–84.

[30] Orlowski, S., Wessäly, R., Pióro, M., Tomaszewski, A., 2010. SNDlib 1.0 - survivable network design library. Networks 55 (3), 276–286.

[31] Padberg, M., Sung, T.-Y., 1991. An analytical comparison of different formulations of the travelling salesman problem. Mathematical Programming 52 (1-3), 315–357.

[32] Righini, G., Salani, M., 2006. Symmetry helps: bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. Discrete Optimization 3 (3), 255–273.

[33] Sherali, H. D., Adams, W. P., 1990. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. SIAM Journal on Discrete Mathematics 3 (3), 411–430.

[34] Taccari, L., 2015. Mixed-integer programming models and methods for bilevel fair network optimization and energy cogeneration planning. Ph.D. thesis, Politecnico di Milano, Milano.

[35] Tarjan, R., 1972. Depth-first search and linear graph algorithms. SIAM Journal on Computing 1 (2), 146–160.

[36] Vansteenwegen, P., Souffriau, W., Oudheusden, D. V., 2011. The orienteering problem: A survey. European Journal of Operational Research 209 (1), 1–10.

[37] Wong, R., 1980. Integer programming formulations of the travelling salesman problem. In: Proceedings of the IEEE international conference on circuits and computers. pp. 149–152.

Table 6: Branch-and-cut results. Missing values are due to the memory limit being reached.

	GCS-StrongComp				*GCS-MinCut*				*MTZ*			*RLT*			*SF*			*MCF*			
	opt	time	nodes	cuts	opt	time	nodes	cuts	opt	time	nodes	opt	time	nodes	opt	time	nodes	opt	time	nodes	cuts
nobel-us	30	0.0	0.6	10.5	30	0.0	3.1	9.6	30	0.0	26.2	30	0.0	19.1	30	0.0	9.0	30	0.0	1.0	397.3
atlanta	30	0.0	0.8	10.9	30	0.0	1.1	10.7	30	0.0	22.0	30	0.0	7.4	30	0.0	1.8	30	0.0	0.6	415.4
geant	30	0.0	1.1	12.8	30	0.0	2.5	11.5	29	0.0	20.9	30	0.0	9.6	30	0.0	4.7	30	0.0	1.1	633.7
france	30	0.0	0.7	26.4	30	0.0	2.8	24.7	30	0.1	344.3	30	0.1	51.1	30	0.1	21.1	30	0.1	1.0	1549.7
germany	30	0.1	25.6	89.9	30	0.1	40.0	81.4	30	3.5	5947.7	30	0.4	352.5	30	0.7	403.0	30	1.6	10.6	4595.3
rnd-s-50	30	0.1	22.3	46.2	30	0.1	33.1	43.4	30	0.2	82.9	30	0.2	79.8	30	0.2	59.0	30	3.2	18.3	6611.3
rnd-s-100	30	0.4	37.3	56.1	30	0.5	40.2	56.7	30	0.9	189.5	30	4.3	403.6	30	2.0	151.0	24	557.9	72.3	32582.7
rnd-s-200	30	0.8	22.1	66.7	30	1.8	25.7	75.0	30	9.0	308.8	30	38.5	322.1	30	19.7	218.1	1	1193.8	1.9	87376.9
rnd-s-500	30	4.2	27.6	77.4	30	32.3	29.4	77.1	30	218.1	496.5	6	1104.1	217.3	26	557.1	373.6	0	1200	0.0	$2.5 \cdot 10^5$
rnd-s-1000	30	21.3	29.6	139.2	26	352.4	25.6	142.4	3	1158.3	51.8	0	1200	0.0	0	1200	0.0	0	–	–	–
rnd-d-25	30	0.0	2.2	6.5	30	0.0	2.4	6.6	30	0.1	31.6	30	0.1	13.6	30	0.1	10.5	30	0.5	4.6	2365.5
rnd-d-50	30	0.1	8.0	14.7	30	0.1	8.3	15.3	30	1.7	460.9	30	1.3	39.7	30	1.0	18.2	30	69.0	14.6	18769.7
rnd-d-100	30	0.4	14.0	24.2	30	0.8	16.1	24.8	30	3.5	70.0	30	30.0	113.8	30	22.1	107.7	6	996.0	2.5	67352.1
prc-f-25	30	0.0	4.0	25.0	30	0.0	3.9	25.7	30	2.8	3059.3	30	1.0	317.1	30	0.4	49.6	30	3.6	13.9	7978.8
prc-f-50	30	0.2	19.9	68.1	30	0.3	16.3	70.8	23	332.8	$1.3 \cdot 10^5$	30	29.9	1346.4	30	16.4	1047.8	28	459.5	34.2	64155.9
prc-f-100	30	11.0	2068.9	246.8	30	40.5	1649.6	236.6	3	1142.7	30600.6	27	328.0	3465.5	30	174.6	1633.7	0	1200	0.0	$4.6 \cdot 10^5$
prc-p-25	30	0.1	7.4	54.5	30	0.2	30.8	49.2	16	688.2	$8.1 \cdot 10^5$	30	1.4	1455.7	30	0.9	426.6	30	5.4	4.1	11687.6
prc-p-50	30	0.6	26.8	103.6	30	0.8	112.4	103.3	11	872.3	$4.5 \cdot 10^5$	30	5.5	1224.6	30	13.4	1113.0	30	288.4	11.8	65648.2
prc-p-100	30	12.7	1243.0	325.3	30	88.1	3003.6	428.2	0	1200	90700.8	29	165.2	19837.3	30	75.6	3432.7	0	1200	0.0	$4.3 \cdot 10^5$
prc-l-25	30	0.2	12.3	64.3	30	0.2	50.9	59.5	10	808.5	$1.4 \cdot 10^6$	30	5.3	1552.5	30	1.3	175.6	30	5.2	4.1	12078.0
prc-l-50	30	0.9	77.3	169.7	30	1.2	188.4	164.0	5	1027.5	$5.1 \cdot 10^5$	30	24.2	1068.5	30	14.8	764.6	30	248.5	23.4	59021.8
prc-l-100	30	14.5	1597.2	340.4	28	209.8	6144.9	444.6	0	1200	88407.3	30	134.7	15209.0	30	84.2	4188.2	0	1200	0.0	$4.3 \cdot 10^5$
rome99	0	1200	16944.9	5668.8	0	1200	728.0	13271.4	0	1200	3126.2	0	1200	780.4	0	1200	447.8	0	–	–	–