Discrete Optimization 7 (2010) 32–47

Contents lists available at ScienceDirect

Discrete Optimization

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

# Algorithmic expedients for the Prize Collecting Steiner Tree Problem

Mohamed Haouari [a,d,*], Safa Bhar Layeb [b], Hanif D. Sherali [c]

[a] Department of Industrial Engineering, Ozyegin University, Istanbul, Turkey

[b] Combinatorial Optimization Research Group - ROI, Ecole Polytechnique de Tunisie, BP 743, 2078, La Marsa, Tunisia

[c] Grado Department of Industrial and Systems Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061, USA

[d] Princess Fatimah Alnijris Research Chair for AMT, College of Engineering, King Saud University, Riyadh, Saudi Arabia

ARTICLE INFO

*Article history:*
Received 30 May 2007
Received in revised form 19 November 2009
Accepted 3 January 2010
Available online 11 March 2010

*Keywords:*
Steiner tree
Reduction techniques
Worst-case analysis
Valid inequalities

ABSTRACT

This paper investigates the Prize Collecting Steiner Tree Problem (PCSTP) on a graph, which is a generalization of the well-known Steiner tree problem. Given a root node, edge costs, node prizes and penalties, as well as a preset quota, the PCSTP seeks to find a subtree that includes the root node and collects a total prize not smaller than the specified quota, while minimizing the sum of the total edge costs of the tree plus the penalties associated with the nodes that are not included in the subtree. For this challenging network design problem that arises in telecommunication settings, we present two valid 0-1 programming formulations and use them to develop preprocessing procedures for reducing the graph size. Also, we design an optimization-based heuristic that requires solving a PCSTP on a specific tree-subgraph. Although, this latter special case is shown to be $\mathcal{NP}$-hard, it is effectively solvable in pseudo-polynomial time. The worst-case performance of the proposed heuristic is also investigated. In addition, we describe new valid inequalities for the PCSTP and embed all the aforementioned constructs in an exact row-generation approach. Our computational study reveals that the proposed approach can solve relatively large-scale PCSTP instances having up to 1000 nodes to optimality.



## 1. Introduction

The Steiner tree problem is one of the most actively investigated problems in graph theory and combinatorial optimization. This core problem poses significant algorithmic challenges and arises in several applications where it serves as a building block for many complex network design problems. Given a connected undirected graph $G = (V, E)$, where $V$ denotes the set of nodes and $E$ the set of edges, along with a weight $c_e$ associated with each edge $e \in E$, the Steiner tree problem seeks a minimum-weight subtree of $G$ that spans a specified subset $N \subset V$ of *terminal* nodes, optionally using the subset $\bar{N} = V \setminus N$ of *Steiner* nodes. The Steiner tree problem is $\mathcal{NP}$-hard for most relevant classes of graphs [1].

In this paper, we consider the following generalization of the Steiner tree problem. We are given a connected, undirected graph $G = (V, E)$, where $V = \{1, \ldots, n\}$ is the node set with node 1 being a specified root node, and $E$ is the edge set, along with a nonnegative edge weight $c_e$ associated with each edge $e \in E$, a nonnegative prize $p_j$ and a penalty $\gamma_j$ associated with each node $j \in V \setminus \{1\}$, and a preset prize quota $Q$. The problem consists of finding a subset $S \subseteq V$ that includes the root node and has a total prize sum of at least $Q$, along with a corresponding subtree $T(S) = (S, E(S))$ of $G$ that minimizes the sum of the weights of the edges in the tree plus the sum of the penalties of those nodes that are not covered by the tree. That is, the problem is to find a subtree $T(S) = (S, E(S))$ of $G$, where $1 \in S$ and $\sum_{j \in S} p_j \geq Q$, which minimizes $\sum_{e \in E(S)} c_e + \sum_{j \in V \setminus S} \gamma_j$.

* Corresponding author at: Department of Industrial Engineering, Ozyegin University, Istanbul, Turkey.
*E-mail address:* mohamed.haouari@ozyegin.edu.tr (M. Haouari).

1572-5286/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.disopt.2010.01.001

We refer to this problem as the *Prize Collecting Steiner Tree Problem* (**PCSTP**). The Steiner tree problem is a special case of the PCSTP with the root node being any terminal node, $p_j = 1$ for all $j \in N \setminus \{\text{root node}\}$, $p_j = 0$ for all $j \in \bar{N}$, $Q = |N| - 1$, and $\gamma_j = 0$ for all $j \in V$.

The PCSTP is a challenging network design problem that arises in many application contexts. For instance, in a telecommunication setting, we may want to design a fiber-optic backhaul network for a fixed wireless system. Here, the root node represents a specified location of a gate (or hub), while the non-root nodes represent the possible antenna sites. For each potential latter site, we are given an estimate of the prize (e.g., revenue) to be obtained from the customers who are reachable from this antenna site, as well as a penalty (opportunity cost) for not placing an antenna at this location. The weight of an edge connecting any pair of nodes corresponds to the cost of laying a fiber-optic cable between the corresponding locations. The problem then, is to design a minimum cost tree network that connects a selected subset of locations such that the total collected revenue is not less than a preset goal. Similar applications arise in cable television and local access networks as well. It is noteworthy that there are some further practical advantages of having both prizes and penalties in the model. First, the units of the prizes and penalties might not be the same. For example, we might have a minimal quota on the number of nodes to be selected (or a quota on the minimal number of nodes to be selected from a specified subset $R \subseteq V \setminus \{1\}$ of nodes), in which case, we would simply have $p_j = 1, \forall j \in V \setminus \{1\}$ (or $p_j = 1, \forall j \in R$, and 0 otherwise). Second, consider the variant of the PCSTP, initially introduced by Klau et al. [2], where the problem is to find a subtree of $G$ that contains the root, spans a node subset $S$, and that maximizes the revenue-to-cost ratio (viz., $\frac{\sum_{j\in S} p_j}{\sum_{e\in E(S)} c_e + \sum_{j\in V\setminus S} \gamma_j}$). This ratio optimization problem is of great practical relevance, as investors are often interested in maximizing the value of the net return on investment. Clearly, a possible alternative to solve this challenging fractional integer programming problem is to iteratively solve a sequence of PCSTPs that are similar to the model addressed in the present paper, where at each iteration a tentative quota on the return (prize) is enforced with the objective of minimizing the total cost.

To the best of our knowledge, the PCSTP, in the foregoing general form, has not been addressed in the literature. This PCSTP variant follows the original definition of the prize-collecting traveling salesman problem studied in the seminal paper by Balas [3]. Moreover, it provides a unifying framework for two special PCSTP variants that have received particular interest from many authors. Indeed, several authors have investigated the *zero-quota* variant. In this case, the problem amounts to finding a subtree that minimizes the cost of the edges in the tree plus the penalties of the nodes not included in the tree. This simplified PCSTP was first introduced by Bienstock et al. [4] who proposed a 3-approximation algorithm. Goemans and Williamson [5] presented an $O(n^3 \log n)$-time primal-dual approximation algorithm with a performance guarantee of $2 - 1/(n-1)$. Subsequently, Johnson et al. [6] proposed an improved version of this primal-dual approximation algorithm having an $O(n^2 \log n)$ complexity. Moreover, Canuto et al. [7] developed a multi-start local search heuristic with perturbation. The different restarts were conducted using a GRASP approach and the initial solutions were constructed by invoking a randomized variant of the Goemans and Williamson's algorithm with modified node profits. In addition to these approximation and heuristic procedures, Lucena and Resende [8] presented a lower bounding method for the zero-quota problem variant, which is based on a 0-1 formulation having an exponential number of subtour-elimination constraints. Lower bounds were derived by solving the LP relaxation of the proposed formulation using a cutting plane approach. The separation algorithm for identifying violated subtour-elimination constraints was refined to generate orthogonal cuts (i.e., subtour-elimination cuts having no common nodes). The authors tested their approach on 114 instances with up to 1000 vertices and 25,000 edges and found that the bounds directly verified optimality in 96 of these test cases. Recently, da Cunha et al. [9] proposed tight primal lower bounding algorithms based on a Lagrangian non-delayed relax-and-cut approach as well as an effective local search method. Furthermore, Ljubić et al. [10] presented an effective branch-and-cut algorithm for the same problem variant. Their approach is based on an integer programming formulation using connectivity inequalities. They report the exact solution of large instances having up to 2500 vertices and 62,500 edges. A second special variant of the PCSTP, often referred to as the *quota problem*, where the node penalties are set to zero has been addressed by Johnson et al. [6] and Haouari and Chaouachi [11]. Johnson et al. [6] showed that there exists a 3-approximation polynomial-time algorithm for this problem by using an approximation algorithm for the $k$-minimum spanning tree problem ($k$-MST). On the other hand, Haouari and Chaouachi [11] developed a Lagrangian decomposition-based lower bounding procedure and designed a genetic algorithm that incorporates primal as well as dual information produced by the Lagrangian decomposition.

Recently, Haouari et al. [12] introduced and investigated a new generalized version of the PCSTP where the node set is partitioned into $k$ clusters and a quota is set for each cluster. For this problem, the authors proposed and analyzed several Lagrangian relaxation-based lower bounds. Moreover, several nondifferentiable optimization algorithms were tested and compared with an exact stabilized constraint generation procedure for solving its Lagrangian dual.

In this paper, we present an exact approach for solving the PCSTP. We provide evidence that combining preprocessing procedures, effective heuristics, and tight 0-1 programming formulations make it possible to solve to optimality large PCSTP instances.

The remainder of this paper is organized as follows. In Section 2, we develop two valid 0-1 programming formulations for the PCSTP, and in Section 3, we describe Lagrangian- as well as LP-based preprocessing procedures that permit us to significantly reduce the problem size. For deriving upper-bounding solutions, we design an effective optimization-based heuristic and analyze its worst-case performance in Section 4. These constructs are then embedded along with a separation

routine for generating members of a certain class of valid inequalities in a row-generation approach for solving the PCSTP in Section 5, and we present extensive computational results in Section 6. Finally, Section 7 concludes the paper.

## 2. 0-1 programming formulations

In this section, we describe two valid 0-1 programming formulations for the PCSTP. Both these formulations will be subsequently used for developing preprocessing tests.

### 2.1. A minimum spanning tree-based formulation (MSTF)

It is interesting to view a Steiner tree problem as a *minimum spanning tree problem with side-constraints*. Beasley [13] was the first to propose such a reformulation for the standard Steiner tree problem, and Lucena and Resende [8] and da Cunha et al. [9] extended it to the zero-quota PCSTP. This reformulation was later tailored by Haouari et al. [12] to a very general variant of the PCSTP. Here, it is briefly reproduced for the sake of completeness. To present this model, let us first modify the original graph $G$ by adding a dummy node 0 as well as dummy edges of the form $\{0, j\}$, $\forall j \in V$, where a weight $c_{\{0,j\}} \equiv \gamma_j$ is associated with each edge $\{0, j\}$, $\forall j \in V \setminus \{1\}$, and a zero weight is assigned to the edge $\{0, 1\}$, i.e., $c_{\{0,1\}} \equiv 0$. Let $\bar{G} = (\bar{V}, \bar{E})$ denote the resulting graph.

Now, consider a spanning tree $T = (\bar{V}, \bar{E}(T))$ of $\bar{G}$ having the following properties:

(P1) the sum of the prizes of the nodes of $V \setminus \{1\}$ that are adjacent to 0 in $T$ is not larger than $\bar{Q} = \sum_{j \in V \setminus \{1\}} p_j - Q$;
(P2) every node $j \in V \setminus \{1\}$ that is adjacent to 0 in $T$ has degree 1;
(P3) $\{0, 1\} \in \bar{E}(T)$.

Haouari et al. [12] show that there is a one-to-one correspondence between each spanning tree of $\bar{G}$ that satisfies properties (P1)–(P3) and each PCSTP solution, with both solutions having the same total weight or cost. Consequently, a valid 0-1 programming formulation of the PCSTP can be constructed as follows:

$$\textbf{MSTF:} \quad \text{Minimize} \sum_{\{i,j\} \in \bar{E}} c_{\{i,j\}} x_{\{i,j\}} \tag{1}$$

subject to:

$$\sum_{j \in V \setminus \{1\}} p_j x_{\{0,j\}} \leq \bar{Q}, \tag{2}$$

$$x_{\{0,j\}} + x_{\{i,j\}} \leq 1, \quad \forall j \in V \setminus \{1\},\ \{i, j\} \in E, \tag{3}$$

$$x_{\{0,1\}} = 1, \tag{4}$$

$$\sum_{\{i,j\} \in \bar{E}} x_{\{i,j\}} = \left|\bar{V}\right| - 1, \tag{5}$$

$$\sum_{\{i,j\} \in \bar{E}: i,j \in S} x_{\{i,j\}} \leq |S| - 1, \quad \forall S \subset \bar{V},\ 3 \leq |S| \leq \left|\bar{V}\right| - 2, \tag{6}$$

$$x_{\{i,j\}} \in \{0, 1\}, \quad \forall \{i, j\} \in \bar{E}, \tag{7}$$

where the binary decision variable $x_{\{i,j\}}$ equals 1 if edge $\{i, j\} \in \bar{E}$ is in the optimal tree, and 0 otherwise. Constraints (2)–(4) ensure that the solution satisfies (P1)–(P3), respectively. Constraints (5)–(6) guarantee that the solution is the incidence vector of a spanning tree, where Constraint (5) asserts that we must select exactly $\left|\bar{V}\right| - 1$ edges, and Constraint (6) ensures that the set of chosen edges contains no cycles. It is easy to check that if $(x_e)_{e \in \bar{E}}$ is the incidence vector of a feasible solution to (2)–(7), then $(x_e)_{e \in E}$ is the incidence vector of the corresponding feasible PCSTP solution.

### 2.2. A directed cut-based formulation (DCF)

Instead of formulating the PCSTP on the undirected graph $G$, we consider a directed cut-based formulation that is defined on the bi-directed graph $B = (V, A)$ that is obtained from $G$ by replacing each edge $e = \{i, j\} \in E$ with two directed arcs $(i, j)$ and $(j, i)$ (with corresponding weights $c_{ij} = c_{ji} = c_e$). Thus, we can pose the PCSTP as requiring to find an optimal arborescence in $B$ that is rooted at node 1 as follows.

Denote by $z_{ij}$, $(i, j) \in A$, the binary variable that takes the value 1 if arc $(i, j)$ belongs to the arborescence and 0 otherwise. We also define a binary variable $y_j, j \in V \setminus \{1\}$, which takes on a value of 1 if node $j$ belongs to the arborescence and 0

otherwise. Using these definitions, Problem PCSTP can be formulated as follows:

$$\textbf{DCF}: \quad \text{Minimize} \sum_{(i,j)\in A} c_{ij} z_{ij} + \sum_{j\in V\setminus\{1\}} \gamma_j (1 - y_j) \tag{8}$$

subject to:

$$\sum_{j\in V\setminus\{1\}} p_j y_j \geq Q, \tag{9}$$

$$\sum_{(i,j)\in\delta^+(S)} z_{ij} \geq y_k, \quad \forall S \subset V,\ 1 \in S,\ k \in V \setminus S, \tag{10}$$

$$z_{ij} \in \{0, 1\}, \quad \forall (i, j) \in A, \tag{11}$$

$$y_j \in \{0, 1\}, \quad \forall j \in V \setminus \{1\}, \tag{12}$$

where for any subset $S \subset V$, we define $\delta^+(S) = \{(i, j) \in A : i \in S \text{ and } j \in V \setminus S\}$. The objective function (8) is to minimize the sum of the weights of the arcs that belong to the arborescence plus the penalties of the nodes that are not covered. Constraint (9) requires that the total prize should not be less than the specified quota. Constraints (10) enforce the connectivity of the solution by essentially requiring the existence of a dipath between the root node and each covered node. Finally, Constraints (11) and (12) state that the decision variables are binary-valued. Section 2.2.1 presents other valid inequalities that can be utilized to further tighten this formulation.

Actually, it is interesting to view the PCSTP as a *Node Weighted Steiner Tree Problem* (NWSTP) with an additional side-constraint (9). The NWSTP, first introduced by Segev [14], requires finding a subtree of a node-weighted graph $G = (V, E)$ that spans a subset $N$ of terminals and possibly includes some nodes from the subset $N' = V \setminus N$ such that the sum of its edge weights minus the sum of the weights on the nodes that are not spanned by the tree is minimized. Obviously, if the node weights are zero, then the NWSTP reduces to the Steiner tree problem. In order to reformulate the PCSTP as an NSP with an additional side-constraint (9), it suffices to set $N = \{1\}$ and define for each node $j \in V \setminus \{1\}$ a weight $w_j = -\gamma_j$. In so doing, we observe that Formulation (8)–(12) is a generalization of the so-called *Directed Cut-Based Node Variable* (DCBNV) formulation that has been previously proposed for the NSP. The DCBNV formulation is particularly interesting since it is shown to provide a very strong LP-relaxation for the Steiner tree problem (see [15]). In Section 5, we shall describe an exact algorithm for solving the PCSTP using the directed cut-based formulation (8)–(12).

**Remark 1.** A similar cut-based formulation has been proposed by Ljubić et al. [10] for the zero-quota PCSTP variant. However, their model differs from the one investigated in this paper not only because of the existence of the quota constraint, but also because they assume the presence of zero-penalty nodes. Hence, some of the valid constraints that are described in their paper, including the so-called *flow-balance constraints*, are not valid for our problem (and *vice-versa*). Moreover, while they implemented a sophisticated branch-and-cut algorithm, our goal is to show that a combination of preprocessing strategies and valid inequalities enable the optimal solution of large instances using an easy-to-code row-generation algorithm.

### 2.2.1. Strengthening the LP-relaxation of model DCF

It is well known that the tightness of the LP-relaxation is of crucial importance for the effective solution of an integer program. In this section, we describe certain classes of valid inequalities for strengthening the LP-relaxation of Model DCF. In the sequel, $V$ and $A$ refer to the set of nodes and arcs that are obtained after invoking any preprocessing of the type described subsequently in Section 3 to reduce the size of the problem.

*Node-induced inequalities.* First, we provide two simple valid inequalities that were previously proposed by Ljubić et al. [10]. Obviously, in any feasible binary solution, we have that $z_{ij} + z_{ji} \leq 1, \forall (i, j) \in A$. Also, for any $i \in V \setminus \{1\}, y_i = 0 \Rightarrow z_{ij} = z_{ji} = 0, \forall (i, j) \in A$. We can combine these constraints as follows:

$$z_{ij} + z_{ji} \leq y_j, \quad \forall j \in V \setminus \{1\},\ (i, j) \in A. \tag{13}$$

Moreover, since the arc weights are nonnegative, then there exists an optimal arborescence where each covered node $j \in V \setminus \{1\}$ has exactly one incident arc (coming into it). Thus, we have

$$\sum_{(i,j)\in\delta^-(j)} z_{ij} = y_j, \quad \forall j \in V \setminus \{1\}, \tag{14}$$

where $\delta^-(j)$ denotes the set of arcs that are incident to node $j \in V \setminus \{1\}$.

**Remark 2.** In a strict sense, Constraints (13) are *not* strengthening inequalities for the model DCF in the presence of (14). Indeed, as observed in Fischetti [16], the connectivity constraints (10) together with Constraints (14) imply the so-called generalized subtour-elimination constraints (GSEC):

$$\sum_{(i,j)\in E(S)} z_{ij} \leq \sum_{j\in S} y_j - y_k, \quad \forall S \subset V, k \in V,$$

where $E(S)$ denotes the set of edges having both ends in $S$. Clearly, setting $S = \{i, j\}$ yields (13). However, as we shall see in Section 5, the model will be solved using a row-generation procedure, where the cut constraints (10) are initially relaxed and then dynamically appended to the relaxed model. Therefore, Constraints (13) are indeed strengthening inequalities for the relaxed master program.

*A flow-based inequality.* This inequality has been previously proposed for the Steiner tree problem (for example in [17]). It requires that node $j \in V \setminus \{1\}$ has no outgoing arc if it has no incident arc. Thus, we have

$$\sum_{(i,j)\in\delta^-(j)} z_{ij} \geq z_{jk}, \quad \forall j \in V \setminus \{1\},\ (j,k) \in A. \tag{15}$$

*Cover inequalities.* We now introduce a new set of inequalities that are specific to Steiner tree problems having a quota constraint. For each node $j \in V \setminus \{1\}$, denote by $\mu_j$ the minimal number of arcs in a dipath in $B$ from the root node to $j$. Also, for each integer $h$, define the node subset $V_h = \{j \in V : \mu_j = h\}$. These subsets are detected by running the following Breadth-First-Search (BFS).

**1.** **Set** $V_0 = \{1\}, W = V \setminus \{1\}, h = 0$
**2.** **While** $(W \neq \emptyset)$
**Begin**
**2.1** $h = h + 1$
**2.2** $V_h = \{j \in W : \exists i \in V_{h-1}$ and $(i, j) \in A\}$
**2.3** $W = W \setminus V_h$
**End** (While)

This algorithm has a linear-time complexity $O(|A|)$ in the number of arcs of $B$.
Let $h^*$ denote the smallest integer satisfying

$$\sum_{h=1}^{h^*} \sum_{j\in V_h} p_j \geq Q.$$

Clearly, in any feasible solution, there is necessarily a path that includes a succession of nodes respectively belonging to $V_1, \ldots V_{h^*}$. Thus, we have

$$\sum_{(i,j)\in\delta(V_{h-1},V_h)} z_{ij} \geq 1, \quad \forall\, h = 1, \ldots, h^*, \tag{16}$$

where $\delta(V_{h-1}, V_h) \equiv \{(i, j) \in A : i \in V_{h-1}$ and $j \in V_h\}$. Because of the manner in which (16) enforces selections from subsets of arcs based on the special connectivity structure of the underlying graph $B$, these covering-type valid inequalities turn out to be quite effective in our computations.

In the sequel, we shall refer to the enhanced version of Model DCF that is augmented by the valid inequalities (13)–(16) as EDCF.

## 3. Graph reduction

An important feature of the proposed solution strategy is the development of effective preprocessing procedures for reducing the problem size by discarding some unnecessary edges and nodes prior to solving the problem. In this section, we describe, in turn, two such Lagrangian- and LP-based reduction techniques.

### 3.1. Lagrangian relaxation-based preprocessing

Following the analysis in Haouari et al. [12], we can derive a lower bound for the PCSTP by applying Lagrangian decomposition to Model MSTF. To that aim, we define for each variable $x_{\{0,j\}}, j \in V \setminus \{1\}$, two copies denoted by $u_j$ and $v_j$, respectively. Also, we introduce for each variable $x_{\{i,j\}}, \{i, j\} \in E$, a companion copy denoted by $w_{\{i,j\}}$. This yields the extended formulation given below:

$$\text{Minimize} \sum_{\{i,j\}\in\bar{E}} c_{\{i,j\}} x_{\{i,j\}} \tag{17}$$

subject to:

$$\sum_{j\in V\setminus\{1\}} p_j u_j \leq \bar{Q}, \tag{18}$$

$$u_j \in \{0, 1\}, \quad \forall j \in V \setminus \{1\}, \tag{19}$$

$$v_j + w_{\{i,j\}} \leq 1, \quad \forall j \in V \setminus \{1\}, \{i, j\} \in E, \tag{20}$$

$$v_j, w_{\{i,j\}} \in \{0, 1\}, \quad \forall j \in V \setminus \{1\}, \{i, j\} \in E, \tag{21}$$

$$x_{\{0,1\}} = 1, \tag{22}$$

$$\sum_{\{i,j\}\in\bar{E}} x_{\{i,j\}} = \left|\bar{V}\right| - 1, \tag{23}$$

$$\sum_{\{i,j\}\in\bar{E}:i,j\in S} x_{\{i,j\}} \leq |S| - 1, \quad \forall S \subset \bar{V}, 3 \leq |S| \leq \left|\bar{V}\right| - 2, \tag{24}$$

$$x_{\{i,j\}} \in \{0, 1\}, \quad \forall \{i, j\} \in \bar{E}, \tag{25}$$

$$x_{\{0,j\}} - u_j = 0, \quad \forall j \in V \setminus \{1\}, \tag{26}$$

$$x_{\{0,j\}} - v_j = 0, \quad \forall j \in V \setminus \{1\}, \tag{27}$$

$$x_{\{i,j\}} - w_{\{i,j\}} = 0, \quad \forall \{i, j\} \in E. \tag{28}$$

Note that $x_{\{0,j\}} = 1$, or $u_j = 1$, or $v_j = 1$ means that node $j \in V \setminus \{1\}$ is *not* covered by the PCSTP solution. Also, $x_{\{i,j\}} = 1$ or $w_{\{i,j\}} = 1$ means that edge $\{i, j\} \in E$ is included in the PCSTP solution.

Define $\alpha_j, \forall j \in V \setminus \{1\}$, $\beta_j, \forall j \in V \setminus \{1\}$, and $\delta_{\{i,j\}}, \forall \{i, j\} \in E$, as the Lagrange multipliers associated with the constraints (26), (27), and (28), respectively. The corresponding Lagrangian function is given by:

$$\theta(\alpha, \beta, \delta) = \text{Min} \sum_{\{i,j\}\in\bar{E}} \tilde{c}_{\{i,j\}} x_{\{i,j\}} - \sum_{j\in V\setminus\{1\}} \alpha_j u_j - \left( \sum_{j\in V\setminus\{1\}} \beta_j v_j + \sum_{\{i,j\}\in E} \delta_{\{i,j\}} w_{\{i,j\}} \right) \tag{29}$$

subject to: (18)–(25),

where,

$$\tilde{c}_{\{i,j\}} = c_{\{i,j\}} + \delta_{\{i,j\}}, \quad \forall \{i, j\} \in E, \tag{30}$$

$$\tilde{c}_{\{0,j\}} = \alpha_j + \beta_j + \gamma_j, \quad \forall j \in V \setminus \{1\}. \tag{31}$$

We see that the computation of $\theta(\alpha, \beta, \delta)$ requires solving three well-known combinatorial optimization problems:

- a binary knapsack (maximization) problem (KP) defined by (18) and (19), which is solvable in pseudo-polynomial time $O(n\bar{Q})$ where $n = |V|$;
- a maximum-weight stable set problem (MWSP) defined by (20) and (21). This MWSP is defined on the bipartite graph $H = (B_1 \cup B_2, F)$ that is obtained as follows. With each edge $\{0, j\}, j \in V \setminus \{1\}$, we associate a node $a_j \in B_1$, and with each edge $\{i, j\} \in E$, we associate a node $b_{\{i,j\}} \in B_2$. An edge $(a_j, b_{\{i,k\}}) \in F$ exists if and only if $j \in \{i, k\}$. The weights of the nodes $a_j \in B_1$ and $b_{\{i,j\}} \in B_2$ are taken as $\beta_j$ and $\delta_{\{i,j\}}$, respectively. The MWSP in bipartite graphs is solvable in polynomial time using linear programming [18];
- a minimum spanning tree problem (MST) defined by (22)–(25). It is well known that this problem can be solved in polynomial time of complexity $O(|\bar{E}| \log |\bar{E}|)$ using a greedy algorithm.

Accordingly, we solve the Lagrangian dual problem defined by

$$\textbf{LD} : \text{Maximize } \theta(\alpha, \beta, \delta), \tag{32}$$

to derive a lower bound on the PCSTP. In our implementation, LD is solved approximately using a deflected subgradient algorithm called the *Average Direction Strategy* (*ADS*) (see [19]). Let $(\alpha^*, \beta^*, \delta^*)$ denote the vector of Lagrange multipliers that is obtained upon termination of the ADS algorithm. Also, let $(x^*, u^*, v^*, w^*)$ denote the corresponding primal solution that evaluates $\theta(\alpha^*, \beta^*, \delta^*)$ in (29). Thus, $x^*$ is the incidence vector of a spanning tree $\mathcal{T}$ of $\bar{G}$, $u^*$ is the incidence vector of a feasible knapsack solution, and $(v^*, w^*)$ is the incidence vector of a stable set of $H$. Denote the corresponding objective values by $T^*$, $K^*$, and $S^*$, respectively, so that we have

$$\theta(\alpha^*, \beta^*, \delta^*) = T^* - K^* - S^*. \tag{33}$$

For each edge $e = \{i, j\} \in E$, let $K_e$ denote the value of the optimal solution of the knapsack problem:

$$\text{Maximize} \sum_{k\in V\setminus\{1,i,j\}} \alpha_k^* u_k \tag{34}$$

subject to:

$$\sum_{k\in V\setminus\{1,i,j\}} p_k u_k \leq \bar{Q}, \tag{35}$$

$$u_k \in \{0, 1\}, \quad \forall k \in V \setminus \{1, i, j\}. \tag{36}$$

Obviously, if $u_i^* = u_j^* = 0$ then $K_e = K^*$.

Next, consider the MST subproblem. Assume that edge $e \in E$ is not contained in $\mathcal{T}$. We know that adding $e$ to $\mathcal{T}$ would create exactly one cycle $\mathcal{C}_e$. Let $f \in \mathcal{C}_e \setminus \{e\}$ denote an edge of $\mathcal{C}_e$ having the largest reduced cost (i.e., $f \in \arg\max_{\{i,j\} \in \mathcal{C}_e \setminus \{e\}} \tilde{c}^*_{\{i,j\}}$ where $\tilde{c}^*_{\{i,j\}} = c_{\{i,j\}} + \delta^*_{\{i,j\}}, \forall \{i,j\} \in E$). Now, for each $e = \{i,j\} \in E$, define

- $T_e = T^*$ if $x^*_e = 1$, and $T_e = T^* + \tilde{c}^*_e - \tilde{c}^*_f$ otherwise;
- $S_e = S^*$ if $w^*_e = 1$, and $S_e = S^* + \bar{\delta}_e$ otherwise, where $\bar{\delta}_e$ is the reduced cost of $e$ that is obtained after solving the MWSP (for evaluating $\theta(\alpha^*, \beta^*, \delta^*)$) by linear programming.

Let *UB* denote a known (best) upper bound on the PCSTP, and consider the following result.

**Proposition 1.** *An edge $e \in E$ cannot be included in an optimal solution if*

$$T_e - K_e - S_e > UB. \tag{37}$$

**Proof.** It suffices to observe that if edge $e = \{i,j\}$ is enforced to belong to the solution, then we would set the constraints $x_e = 1$, $u_i = u_j = 0$, and $w_e = 1$ within (17)–(28). Examining the resulting Lagrangian dual value for the same dual solution $(\alpha^*, \beta^*, \delta^*)$, and solving the corresponding subproblems KP, MST, and deriving an upper bound on the corresponding subproblem MWSP, we see that a valid resulting lower bound on the PCSTP that includes edge $e$ is $T_e - K_e - S_e$. The result follows immediately. ■

This result offers a practical way of discarding sub-optimal edges from the graph prior to solving the PCSTP.

### 3.2. LP-based and cost-based preprocessing

Consider the LP-relaxation of the formulation defined by (8)–(16). Suppose that this LP yields an optimal objective value *LR* along with subsets $\xi_0$ and $\xi_1$ of variables that are equal to 0 and 1, respectively (the details of the solution procedure are provided in Section 5). It is well known that if a node- or an arc-variable $\theta \in \xi_0$ ($\xi_1$) has a reduced cost that is strictly larger (smaller) than $UB - LR$ ($LR - UB$), then $\theta = 0$ ($\theta = 1$) for any optimal integer solution. Hence, further *a priori* variable-settings might be achieved through invoking such LP-based preprocessing tests.

It is worth noting that in case where an arc $(i,j) \in A$ should be included in an optimal solution, nodes $i$ and $j$ are merged into a single node $v$ satisfying $p_v = p_i + p_j$ and $\gamma_v = \gamma_i + \gamma_j$, where we now set $y_v = 1$. Moreover, each arc $(k,i) \in A$ is replaced with an arc $(k,v)$ having the same cost and each arc outgoing from node $i$ or $j$ is replaced with an arc outgoing from node $v$ and having the same cost (obviously if after this transformation there are two arcs going from node $v$ to some node $k$, then we discard the one having the larger cost).

Interestingly, we can possibly fix further variables by observing that if for some arc $(i,j) \in A$ the inequality $c_{ij} \leq \gamma_j$ holds, then we have that $y_i = 1 \Rightarrow y_j = 1$ is valid at optimality. Thus, in this case, we can impose the inequality $y_i \leq y_j$. Consequently, if for node $j \in V \setminus \{1\}$, the inequality $c_{1j} \leq \gamma_j$ holds, then we set $y_j = 1$. Also, as previously noted by Duin and Volgenant [20], if for $(i,j) \in A$ we have $c_{ij} \leq \min\{\gamma_i, \gamma_j\}$, then the equality $y_i = y_j$ is valid. We refer to these preprocessing rules as *cost-based tests*.

In our implementation, the above-described preprocessing tests are invoked in the following manner. We first use Proposition 1 to achieve a graph reduction, and then subsequently, we use the LP-based preprocessing as well as the cost-based tests to discard additional arcs and nodes. Moreover, after achieving each graph reduction, all isolated nodes are also discarded. In addition, all nodes having no incident arcs are removed as well. Next, we check the connectedness of the reduced graph. Indeed, the elimination of edges/arcs during the preprocessing phase may produce a reduced graph that is disconnected. In this case, all of the disconnected components that do not contain the root node are discarded. Finally, it is worth emphasizing that each time a new node-variable is set to a fixed value, we propagate this setting to the adjacent nodes.

## 4. A span-and-prune heuristic procedure

### 4.1. $\mathcal{NP}$-hardness result

The PCSTP is known to be $\mathcal{NP}$-hard. We provide a stronger complexity result.

**Proposition 2.** *The PCSTP on trees is weakly $\mathcal{NP}$-hard.*

**Proof.** The proof is based upon reduction from the $\mathcal{NP}$-hard binary knapsack problem (BKP), stated below in a minimization form:

$$\text{Minimize} \left\{ \sum_{j=1}^{n} \pi_j x_j : \sum_{j=1}^{n} a_j x_j \geq b, x_j \in \{0,1\}, \ \forall j = 1, \ldots, n \right\} \tag{38}$$

where $\pi_j \geq 0$ and $a_j \geq 0, \forall j = 1, \ldots, n$.

Now, we construct a tree $G^* = (V^*, E^*)$ having the topology of a star graph with the center node 0 and the leaf nodes $1, 2, \ldots, n$. Thus, the edge set is $E^* = \{\{0, 1\}, \{0, 2\}, \ldots, \{0, n\}\}$. We define a PCSTP instance on $G^*$ as follows. We take node 0 as the root node and ascribe a cost of $\pi_j$ for each edge $\{0, j\} \in E^*$. For each leaf node $j \in V^* \setminus \{0\}$, we define a prize $a_j$ and a zero penalty and set the quota to $b$. Clearly, finding an optimal BKP solution amounts to solving a PCSTP on the tree $G^*$ and vice versa. ■

### 4.2. A pseudo-polynomial time algorithm for the PCSTP on trees

In this section, we show that it is still possible to solve a PCSTP on a tree quite effectively using a pseudo-polynomial time algorithm. Assume that the minimum-cost spanning tree of $G$ that is produced in the first step is denoted by $T = (V, E(T))$ and that the cost of this particular solution of the PCSTP is $c(T)$. For convenience, we convert $T$ into an arborescence $\vec{T}$ that is rooted at node 1 and is obtained as follows. For each node $j \in V \setminus \{1\}$, let $\{i, j\} \in E(T)$ denotes the edge that is incident to $j$ in the path $\mathcal{P}_j$ in $T$ between the root node and $j$. Then, edge $\{i, j\}$ is replaced with arc $\delta_j^- \equiv (i, j)$.

Consider the following notation:

- $\sigma_j =$ set of all nodes of the sub-arborescence of $\vec{T}$ rooted at node $j$, $\forall j \in V \setminus \{1\}$;
- $a_j = p_j + \sum_{k \in \sigma_j} p_k$, $\forall j \in V \setminus \{1\}$;
- $\omega_j = \gamma_j + \sum_{k \in \sigma_j} \gamma_k$, $\forall j \in V \setminus \{1\}$;
- $\pi_j =$ total edge weight of the sub-arborescence of $\vec{T}$ rooted at node $j$ plus the weight of the arc $\delta_j^-$, $\forall j \in V \setminus \{1\}$.

We observe that if an arc $\delta_j^-$ ($j \in V \setminus \{1\}$) is discarded from $\vec{T}$ (along with the sub-arborescence rooted at node $j$), then we get a PCSTP solution having a total cost $c(T) - \pi_j + \omega_j$. Moreover, this solution is feasible if $a_j \leq \bar{Q}$. Hence, solving the PCSTP on $T$ amounts to solving the following precedence-constrained knapsack problem:

$$\text{Maximize} \sum_{j \in V \setminus \{1\}} (\pi_j - \omega_j) x_j \tag{39}$$

subject to:

$$\sum_{j=1}^{n} a_j x_j \leq \bar{Q}, \tag{40}$$

$$x_i + x_j \leq 1, \quad \forall i \in \sigma_j,\ j \in V \setminus \{1\}, \tag{41}$$

$$x_j \in \{0, 1\}, \quad j \in V \setminus \{1\}, \tag{42}$$

where $x_j = 1$ if $\vec{T}$ is pruned by snipping (discarding) arc $\delta_j^-$ ($j \in V \setminus \{1\}$), and 0 otherwise. The objective (39) is to maximize the total contribution of the discarded arcs (note that the cost of the final resulting PCSTP solution is $c(T) - \sum_{j \in V \setminus \{1\}} (\pi_j - \omega_j) x_j$). Constraint (40) guarantees that the resulting solution is feasible to the prize quota restriction. The packing constraints (41) express that if $i$ is a successor of $j$ in $\vec{T}$ then we cannot select to discard both $\delta_i^-$ and $\delta_j^-$ and, hence, this avoids multiple counting of edge and node weights. Constraints (42) require that the decision variables are binary-valued.

**Proposition 3.** *The problem defined by (39)–(42) can be solved in $O(n\bar{Q})$-time.*

**Proof.** First, we associate to $\vec{T}$ a permutation $\eta = (\eta(1), \eta(2), \ldots, \eta(n))$ of the node set that exhibits the preorder arrangement property in that for each node $j$, its successor nodes are arranged to appear immediately following it (see [21]). Hence, $\eta(1) = 1$. The permutation $\eta$ can be constructed in $O(n)$-time using a *depth-first search* algorithm [21, p. 73–76]. In the sequel, we replace the index of each node $j$ by its corresponding ordering $\eta(j)$. Now, we define an acyclic digraph $D = (\hat{V}, \hat{A})$ as follows. The set of nodes is $\hat{V} = V \cup \{n + 1\}$. The arc set $\hat{A}$ is constructed as follows:

- There is an arc $(1, j)$ for $j = 2, \ldots, n + 1$;
- There is an arc $(j, n + 1)$ for $j = 2, \ldots, n$;
- There is an arc $(i, j)$ for $i < j$ and node $j$ is not a successor of node $i$, $\forall i, j \in V \setminus \{1\}$.

For each arc $(i, j) \in \hat{A}$, we define two numbers: a length $c_{ij}$ and a demand $d_{ij}$. The length of any arc that is incident to node $j$ ($j \in V \setminus \{1\}$) is $\pi_j - \omega_j$ and its corresponding demand is $a_j$. The length and demand of each arc incident to the dummy node $n + 1$ are null. Let $\mathcal{P} = (1, j_1, \ldots, j_q, n + 1)$ denote a dipath in $D$ from node 1 to node $n + 1$. This dipath has a total length $l(\mathcal{P}) = \sum_{h=1}^{q} (\pi_{j_h} - \omega_{j_h})$ and a total demand $d(\mathcal{P}) = \sum_{h=1}^{q} a_{j_h}$. Thus, we associate to $\mathcal{P}$ the solution of the system (40)–(42), which is defined by setting $x_j = 1$ for $j \in \{j_1, \ldots, j_q\}$ and 0 otherwise. One can readily check that if $d(\mathcal{P}) \leq \bar{Q}$ then this latter solution is necessarily feasible and has a total cost equal to $l(\mathcal{P})$. Conversely, given a feasible solution to (40)–(42), we can associate a dipath between nodes 1 and $n+1$ having the same cost and a total demand not larger than $\bar{Q}$. Consequently, we can solve the problem defined by (39)–(42) by equivalently solving a longest path problem with a knapsack constraint in an acyclic digraph. This latter problem can be solved in $O(n\bar{Q})$-time using dynamic programming [22]. ■

### 4.3. A span-and-prune heuristic

In this section, we describe a two-phase heuristic for computing an upper bound *UB* that might prove useful for enhancing the effectiveness of the proposed preprocessing procedures.

The basic idea is to construct a heuristic solution by successively solving to optimality two well-studied combinatorial optimization problems. In a first step, a minimum-cost spanning tree on $G$ is computed using the edge weights $c_e$, $\forall e \in E$. Obviously, this tree is a feasible solution to the PCSTP. However, a possibly improved solution may be obtained after pruning some unnecessary branches. Thus, the second step requires solving a PCSTP on the derived spanning tree. In the sequel, we refer to this approach as the *span-and-prune heuristic* (or **SPH**, for short).

As shown in Section 4.2, the PCSTP on a tree can be solved in pseudo-polynomial time. Consequently, the SPH procedure delivers a heuristic solution to the PCSTP in pseudo-polynomial time.

#### 4.3.1. Worst-case performance

We next address the worst-case performance of SPH.

**Proposition 4.** *The worst-case relative performance of SPH is unbounded.*

**Proof.** Examine the following geometric instance of the PCSTP. Consider the vertices $1, 2, \ldots, n$ of a regular $n$-polygon. The length of each edge is 1. Now, distort this polygon slightly by relocating vertex $n$ at a distance $1 + \epsilon$ ($\epsilon > 0$) from its two neighboring vertices (namely, 1 and $n-1$), so that the distance between vertex $n$ and each vertex $j$ ($j = 2, \ldots, n-2$) is strictly larger than $1 + \epsilon$. We set a prize $p_j = 1$ for each $j$ ($j = 2, \ldots, n-1$), and $p_n = n - 2$. The penalties are zero and the quota is $n - 2$. The first step of SPH yields a minimum spanning tree which is the path $(1, 2, \ldots, n)$ having a total length $Z_{MST} = n - 1 + \epsilon$. The next phase of SPH prunes this tree by discarding the arc $(n-1, n)$ and thus yields a feasible PCSTP solution having a total cost $Z_{SPH} = n - 2$. However, the optimal solution includes the single edge $\{1, n\}$, collects a total profit $n - 2$ and has a cost $Z^* = 1 + \epsilon$. We see that in this case, $\lim_{n\to\infty} \frac{Z_{SPH}}{Z^*} = +\infty$. ■

#### 4.3.2. Iterated SPH

Because the final solution produced by SPH depends strongly on the input spanning tree, and in light of Proposition 4, we found it useful to reiterate SPH several times with different input spanning trees. More precisely, we have implemented the following iterated strategy. During the computation of the Lagrangian bounds via (32), a minimal spanning tree is computed at each iteration of the ADS deflected subgradient algorithm. Thus, every fifty iterations, the computed spanning tree is considered as a new input to SPH and a PCSTP is solved on this tree. Accordingly, we retain the best solution thus found. We shall see in Section 6 where the results of the computational study are discussed that this iterated SPH delivers excellent solutions in most cases.

## 5. A row-generation solution approach

In this section, we describe an exact solution method for Model EDCF that was presented in Sections 2.2 and 2.2.1. Since this model contains an exponential number of constraints, we shall solve it using a row-generation procedure, where the cut constraints (10) are not all included *a priori* and only violated constraints are iteratively generated on the fly and appended to the model. More precisely, we first apply the Lagrangian relaxation-based preprocessing strategy of Section 3.1 using the heuristic solution generated as in Section 4 in order to reduce the given graph $G$, and then construct an initial Model EDCF that includes all the constraints except (10). We next solve the LP relaxation of this model using a general-purpose solver (such as CPLEX) to yield an optimal solution $(\bar{y}, \bar{z})$, say. In order to check whether there exist one or more violated cut constraints (10), we need to solve the following separation problem for each $k \in V \setminus \{1\}$ such that $\bar{y}_k > 0$:

$$\textit{Find } W_k \subset V \textit{ satisfying } 1 \in W_k \textit{ and } k \in V \setminus W_k, \textit{ such that } \sum_{(i,j)\in\delta^+(W_k)} \bar{z}_{ij} < \bar{y}_k,$$
$$\textit{or verify that no such subset exists.} \tag{43}$$

This problem amounts to finding a minimum cut that separates the root node 1 and node $k$ on the digraph $B$ of Section 2.2, where the capacity of each arc $(i, j) \in A$ is $\bar{z}_{ij}$. Let $\phi_k$ denote the value of the maximum flow between 1 and $k$ using these arc capacities. By the max-flow-min-cut theorem (see [21]), if $\phi_k < \bar{y}_k$, then the node subset corresponding to the minimal cut solves (43) and yields a violated cut (10). Thus, if one or more violated cut constraints are found, then the model is augmented by these cuts, and the process is reiterated until a feasible solution is found. Next, the graph is reduced using the LP-based and the cost-based preprocessing strategies discussed in Section 3.2. Finally, using the resulting graph, we next continue the row/cut-generation strategy discussed above, except that now, each relaxed model is solved as a 0-1 program.

A synthesis of the overall proposed approach is given below:

**Step 0:** Input a PCSTP instance defined on an undirected graph $G$.

**Table 1**
Performance on Class B test cases.

| *Inst.* | *n* | *m* | $Z_{ISPH}$ | $Z^*$ | *%Gap ISPH* | *Time ISPH* | *Total time* | Termination test |
|---|---|---|---|---|---|---|---|---|
| B.01 | 50 | 63 | 142 | 140 | 1.43 | 0.20 | 0.22 | – |
| B.02 | 50 | 63 | 153 | 153 | 0.00 | 0.25 | 0.25 | 3 |
| B.03 | 50 | 63 | 128 | 128 | 0.00 | 0.20 | 0.22 | 3 |
| B.04 | 50 | 100 | 121 | 120 | 0.83 | 0.27 | 0.42 | – |
| B.05 | 50 | 100 | 107 | 107 | 0.00 | 0.27 | 0.27 | 3 |
| B.06 | 50 | 100 | 121 | 121 | 0.00 | 0.42 | 0.44 | 3 |
| B.07 | 75 | 94 | 204 | 204 | 0.00 | 0.78 | 0.83 | 3 |
| B.08 | 75 | 94 | 204 | 204 | 0.00 | 0.33 | 0.33 | 3 |
| B.09 | 75 | 94 | 204 | 204 | 0.00 | 0.50 | 0.58 | – |
| B.10 | 75 | 150 | 191 | 191 | 0.00 | 0.55 | 0.55 | 3 |
| B.11 | 75 | 150 | 183 | 183 | 0.00 | 0.25 | 0.25 | 1 |
| B.12 | 75 | 150 | 176 | 176 | 0.00 | 0.33 | 0.33 | 3 |
| B.13 | 100 | 125 | 306 | 300 | 2.00 | 0.69 | 0.96 | – |
| B.14 | 100 | 125 | 278 | 278 | 0.00 | 0.62 | 0.73 | 3 |
| B.15 | 100 | 125 | 282 | 282 | 0.00 | 0.78 | 0.83 | 3 |
| B.16 | 100 | 200 | 219 | 219 | 0.00 | 0.52 | 0.52 | 3 |
| B.17 | 100 | 200 | 198 | 198 | 0.00 | 0.45 | 0.45 | 3 |
| B.18 | 100 | 200 | 226 | 224 | 0.89 | 0.53 | 1.28 | – |
| Average | | | | | 0.29 | 0.44 | 0.52 | |

**Step 1:** Solve the Lagrangian dual defined by (32) in order to derive a lower bound $LB_{LD}$, and compute an upper bound *UB* using the iterated span-and-prune heuristic routine.
**Step 2:** Termination test 1: if $UB = LB_{LD}$, then go to Step 7.
**Step 3:** Use the Lagrangian relaxation-based preprocessing test of Section 3.1 (see Proposition 1) for reducing the graph *G*, and construct the associated bi-directed graph *B*. Invoke the iterated span-and-prune heuristic on *B*. If an improved solution is discovered then update *UB*. Termination test 2: if $UB = LB_{LD}$, then go to Step 7.
**Step 4:** Solve the linear relaxation of Model EDCF using the foregoing row-generation strategy. Let $LB_{LP}$ denote the value of the optimal solution. Termination test 3: if $UB = LB_{LP}$, then go to Step 7.
**Step 5:** Invoke the LP-based and the cost-based preprocessing strategy of Section 3.2 for reducing *B*.
**Step 6:** Use a general-purpose MIP solver to optimize Model EDCF that includes the cut constraints (10) that have been generated in Step 4, and then subsequently, generate any violated Constraints (10) at the resulting binary optimum and reiterate Step 6. If no such violated constraints are found, go to Step 7.
**Step 7:** Output the incumbent solution as an optimum.

It is worth noting that if the instance has integer costs and penalties, then we can replace *LB* by $\lceil LB \rceil$ above.

## 6. Computational results

We have coded the proposed solution approach in Microsoft Visual C++ (Version 6.0) and have implemented it on a Pentium IV 3.2 GHz PC with 3.0 GB RAM. We set the maximum CPU time limit to 3600 s. The longest path problem with a knapsack side-constraint that arises in the prune-and-span heuristic (see Proposition 3) has been solved using the Lagrangian relaxation-based approach described in Handler and Zang [23]. For solving the various LPs and IPs we have used CPLEX (version 9.0).

*Performance on SteinLib instances*

The test-bed used consists of three problem classes (denoted B, C, and D) that are differentiated based on their relative sizes and are composed of 58 PCSTP instances that were obtained after appending prizes and penalties for benchmark instances from *SteinLib*, a library for the Steiner tree problem [24]. These problem instances tend to have sparse undirected graphs, and have numbers of vertices and edges ranging from 50–1000 and 63–25,000, respectively. The node prizes were randomly drawn from the discrete uniform distribution $\mathbf{U}[1, 10]$, and for each node $j \in V \setminus \{1\}$, the corresponding penalty was set to $\gamma_j = \lceil 0.5 p_j \rceil$. Finally, for each instance, the quota was set to $Q = \left\lceil 0.5 \sum_{j \in V \setminus \{1\}} p_j \right\rceil$.

The results obtained are displayed in Tables 1–3. The column headings are as follows: *Inst.* = name of the instance; $n$ = number of nodes; $m$ = number of edges; $Z_{ISPH}$ = value obtained with the iterated span-and-prune heuristic; $Z^*$ = value of the optimal solution, $\%Gap\ ISPH = 100 \times \frac{Z_{ISPH} - \hat{Z}}{\hat{Z}}$ (where $\hat{Z} \equiv Z^*$ if the optimal solution is known, and $\hat{Z} \equiv \max\{LB_{LD}, LB_{LP}\}$, otherwise), *Time ISPH* = CPU time (s) required by the iterated span-and-prune heuristic; *Time* = total CPU time (s) required; *Termination Test* = Termination test used to stop the procedure (if any).

From Tables 1–3, we observe that the proposed approach exhibits a very good performance, being able to solve to optimality large-sized PCSTP instances having up to 1000 nodes within a reasonable CPU time effort. Indeed, a proven optimal solution has been obtained for 53 of the 58 instances. For the five unsolved instances, the percentage optimality gaps

**Table 2**
Performance on Class C test cases.

| *Inst.* | *n* | *m* | $Z_{ISPH}$ | $Z^*$ | *%Gap ISPH* | *Time ISPH* | *Total time* | Termination test |
|---|---|---|---|---|---|---|---|---|
| C.01 | 500 | 625 | 1381 | 1379 | 0.15 | 19.48 | 166.67 | – |
| C.02 | 500 | 625 | 1418 | 1415 | 0.21 | 28.31 | 51.57 | – |
| C.03 | 500 | 625 | 1390 | 1385 | 0.36 | 46.37 | 520.58 | – |
| C.04 | 500 | 625 | 1415 | 1415 | 0.00 | 58.34 | 71.8 | 3 |
| C.05 | 500 | 625 | 1363 | 1358 | 0.37 | 25.91 | 40.86 | – |
| C.06 | 500 | 1000 | 1138 | 1138 | 0.00 | 10.92 | 12.04 | 3 |
| C.07 | 500 | 1000 | 1216 | 1213 | 0.25 | 14.00 | 16.95 | – |
| C.08 | 500 | 1000 | 1154 | 1154 | 0.00 | 15.73 | 20.87 | 3 |
| C.09 | 500 | 1000 | 1156 | 1156 | 0.00 | 14.49 | 15.49 | 1 |
| C.10 | 500 | 1000 | 1152 | 1152 | 0.00 | 16.09 | 17.31 | 3 |
| C.11 | 500 | 2500 | 748 | 748 | 0.00 | 19.58 | 20.58 | 1 |
| C.12 | 500 | 2500 | 784 | 784 | 0.00 | 16.92 | 17.22 | 1 |
| C.13 | 500 | 2500 | 782 | 781 | 0.13 | 15.08 | 28.8 | – |
| C.14 | 500 | 2500 | 767 | 767 | 0.00 | 11.05 | 12.18 | 3 |
| C.15 | 500 | 2500 | 768 | 768 | 0.00 | 19.89 | 20.7 | 1 |
| C.16 | 500 | 12500 | 503 | 503 | 0.00 | 18.31 | 20.56 | 1 |
| C.17 | 500 | 12500 | 499 | 499 | 0.00 | 16.89 | 19.12 | 1 |
| C.18 | 500 | 12500 | 502 | 502 | 0.00 | 16.38 | 18.62 | 1 |
| C.19 | 500 | 12500 | 504 | 504 | 0.00 | 6.58 | 8.96 | 1 |
| C.20 | 500 | 12500 | 504 | 504 | 0.00 | 21.70 | 23.93 | 1 |
| Average | | | | | 0.07 | 20.60 | 56.24 | |

**Table 3**
Performance on Class D test cases.

| *Inst.* | *n* | *m* | $Z_{ISPH}$ | *%Gap ISPH* | $Z^*$ | *Time ISPH* | *Total time* | Termination test |
|---|---|---|---|---|---|---|---|---|
| D.01* | 1000 | 1250 | 2681 | 0.11 | – | 466.89 | – | – |
| D.02 | 1000 | 1250 | 2717 | 0.18 | 2712 | 204.52 | 1517.09 | – |
| D.03 | 1000 | 1250 | 2714 | 0.00 | 2714 | 91.47 | 267.57 | 3 |
| D.04 | 1000 | 1250 | 2708 | 0.26 | 2701 | 145.02 | 228.37 | – |
| D.05 | 1000 | 1250 | 2734 | 0.26 | 2727 | 121.48 | 281.83 | – |
| D.06* | 1000 | 2000 | 2313 | 0.17 | – | 103.14 | – | – |
| D.07 | 1000 | 2000 | 2331 | 0.09 | 2329 | 59.46 | 99.38 | 3 |
| D.08 | 1000 | 2000 | 2314 | 0.04 | 2313 | 68.13 | 72.82 | 3 |
| D.09 | 1000 | 2000 | 2327 | 0.00 | 2327 | 87.88 | 95.14 | 3 |
| D.10 | 1000 | 2000 | 2254 | 0.13 | 2251 | 81.23 | 91.31 | 3 |
| D.11 | 1000 | 5000 | 1543 | 0.06 | 1542 | 302.39 | 605.48 | – |
| D.12* | 1000 | 5000 | 1603 | 0.25 | – | 2295.31 | – | – |
| D.13* | 1000 | 5000 | 1579 | 0.19 | – | 149.34 | – | – |
| D.14* | 1000 | 5000 | 1538 | 0.20 | – | 107.48 | – | – |
| D.15 | 1000 | 5000 | 1558 | 0.06 | 1557 | 73.80 | 188.95 | 3 |
| D.16 | 1000 | 25000 | 1005 | 0.00 | 1005 | 54.86 | 72.81 | 3 |
| D.17 | 1000 | 25000 | 1005 | 0.00 | 1005 | 72.24 | 95.00 | 3 |
| D.18 | 1000 | 25000 | 1005 | 0.00 | 1005 | 212.05 | 226.53 | 3 |
| D.19 | 1000 | 25000 | 1005 | 0.00 | 1005 | 215.13 | 233.09 | 3 |
| D.20 | 1000 | 25000 | 1004 | 0.00 | 1004 | 50.58 | 70.92 | 3 |
| Average | | | | 0.10 | | 248.12 | 276.42 | |

* Remained unsolved after reaching the maximum time limit of 3600 s.

at termination were less than 0.25%. Interestingly we observe that 38 instances were optimally solved without resorting to the row-generation procedure. Indeed, for 10 instances, the Lagrangian bound yielded a zero gap (Termination Test 1), while the LP bound yielded a zero gap for 28 instances (Termination Test 3). However, Termination Test 2 was never invoked. Moreover, we see that the iterated span-and-prune heuristic performs extremely well and produces average percentage optimality gaps for Classes B, C, and D of 0.29%, 0.07%, and 0.10%, respectively, consuming at an average 0.44, 20.60, and 248.12 CPU s.

In Table 4, we report for the particular instances for which $UB > LB \equiv \max\{LB_{LR}, LB_{LP}\}$ (note that for this subset of instances, all of the above-described preprocessing strategies were invoked), the percentage of discarded arcs (*%DA*); the percentage of discarded nodes (*%DN*); and the percentage of node variables set to 1 (*%FN*). (We do not report the percentage of arc variables set to 1 because this percentage is zero for all instances, except for (C13), for which two arcs were set to 1.) We observe from this table that very often, the problem size is significantly reduced.

*Impact of the valid inequalities*

In order to investigate the impact of the valid inequalities (13)–(16), we dropped them from the formulation and ran the proposed algorithm on the solved instances for which $UB > LB$ at the root node. The results are displayed in Table 5.

**Table 4**
Performance of the preprocessing routines for the instances having $LB < UB$.

| *Inst.* | *n* | *m* | *%DA* | *%DN* | *%FN* |
|---|---|---|---|---|---|
| B.01 | 50 | 63 | 29.37 | 20.00 | 26.00 |
| B.04 | 50 | 100 | 40.00 | 24.00 | 32.00 |
| B.09 | 75 | 94 | 43.09 | 22.67 | 26.67 |
| B.13 | 100 | 125 | 7.60 | 5.00 | 9.00 |
| B.18 | 100 | 200 | 25.50 | 11.00 | 20.00 |
| C.01 | 500 | 625 | 33.68 | 24.00 | 20.60 |
| C.02 | 500 | 625 | 20.48 | 14.60 | 22.20 |
| C.03 | 500 | 625 | 7.04 | 4.00 | 11.20 |
| C.05 | 500 | 625 | 2.48 | 2.40 | 13.80 |
| C.07 | 500 | 1000 | 23.75 | 6.80 | 27.20 |
| C.13 | 500 | 2500 | 70.28 | 4.40 | 83.00 |
| D.01 | 1000 | 1250 | 13.12 | 11.20 | 26.90 |
| D.02 | 1000 | 1250 | 6.08 | 5.40 | 16.50 |
| D.04 | 1000 | 1250 | 2.04 | 1.40 | 14.50 |
| D.05 | 1000 | 1250 | 2.36 | 1.80 | 14.90 |
| D.06 | 1000 | 2000 | 30.88 | 10.30 | 32.00 |
| D.11 | 1000 | 5000 | 67.52 | 2.60 | 85.00 |
| D.12 | 1000 | 5000 | 46.45 | 0.00 | 82.50 |
| D.13 | 1000 | 5000 | 44.74 | 0.10 | 83.60 |
| D.14 | 1000 | 5000 | 45.28 | 0.50 | 83.90 |
| Average | | | 28.09 | 9.74 | 36.57 |

**Table 5**
Impact of the valid inequalities for the solved instances having $LB < UB$.

| *Inst.* | *n* | *m* | *Time* | *Time ratio* |
|---|---|---|---|---|
| B.01 | 50 | 63 | 0.28 | 1.28 |
| B.04 | 50 | 100 | 0.47 | 1.11 |
| B.09 | 75 | 94 | 0.83 | 1.43 |
| B.13 | 100 | 125 | 4.61 | 4.82 |
| B.18 | 100 | 200 | 2.78 | 2.16 |
| C.01 | 500 | 625 | 172.98 | 1.03 |
| C.02 | 500 | 625 | 75.01 | 1.45 |
| C.03 | 500 | 625 | 521.51 | 1.00 |
| C.05 | 500 | 625 | 80.38 | 1.97 |
| C.07 | 500 | 1000 | 390.23 | 23.02 |
| C.13 | 500 | 2500 | 81.35 | 2.82 |
| D.05 | 1000 | 1250 | 435.61 | 1.55 |
| D.11 | 1000 | 5000 | 804.86 | 1.32 |
| Average | | | | 3.45 |

In this table, the column "Time ratio" reports the ratio of the CPU time obtained with the simplified formulation to that for the enhanced one. We observe from this table that for several solved instances, the CPU time increased dramatically when the proposed valid inequalities were omitted and that seven instances remained unsolved after reaching the maximum CPU time limit (3600 s).

Pushing our analysis a step further, we investigated the impact of dropping a single type of valid inequality at a time. A summary of the results is displayed in Table 6. In this table, the column entitled "(vi)_ratio" reports the ratio of the CPU time obtained with the formulation that does *not* include constraints (vi) (vi $= 13, \ldots, 16$) to that which includes all the proposed valid inequalities.

Interestingly, we see from Table 6 that each single valid inequality type positively impacts the overall efficacy of the proposed approach. Indeed, notwithstanding the fact that three instances required shorter CPU times, the average CPU times increased for *all* the four simplified formulations by 7%–77%. Moreover, we see that one instance (D02) could only be solved when all the valid inequalities are jointly included in the formulation. Surprisingly, we see that dropping the simple two-node GSEC inequalities (13) resulted in the highest increase of the average CPU time (77%) and also prevented three additional instances from being solved. Moreover, we observe that the cover inequalities (16) play a significant role, since dropping them increased the average CPU time by 41% and prevented two additional instances from being optimally solved.

*Impact of the preprocessing routines*

To assess the efficacy of the proposed preprocessing routines, we ran a simplified version of the exact approach that includes the valid inequalities (13)–(16) but where the graph reduction routines are skipped. A summary of the results is displayed in Table 7. Not surprisingly, the CPU time decreased for all the (*small-sized*) instances of Class B, except for one

**Table 6**
Analysis of the impact of each class of valid inequalities for the solved instances having $LB < UB$.

| *Inst.* | *n* | *m* | (13)_ratio | (14)_ratio | (15)_ratio | (16)_ratio |
|---|---|---|---|---|---|---|
| B.01 | 50 | 63 | 1.42 | 1.02 | 1.01 | 1.06 |
| B.04 | 50 | 100 | 0.87 | 0.80 | 0.91 | 0.86 |
| B.09 | 75 | 94 | 0.94 | 0.94 | 0.96 | 1.02 |
| B.13 | 100 | 125 | 1.20 | 0.85 | 0.92 | 1.05 |
| B.18 | 100 | 200 | 2.94 | 2.49 | 1.09 | 1.46 |
| C.01 | 500 | 625 | 1.65 | 0.99 | 0.91 | – |
| C.02 | 500 | 625 | 2.90 | 0.69 | 0.72 | 1.26 |
| C.03 | 500 | 625 | – | 0.17 | 0.88 | 2.24 |
| C.05 | 500 | 625 | 0.92 | 0.84 | 0.76 | 0.85 |
| C.07 | 500 | 1000 | 0.93 | 0.89 | 0.86 | 2.86 |
| C.13 | 500 | 2500 | 5.73 | 2.69 | 0.57 | 0.65 |
| D.02 | 1000 | 1250 | – | – | – | – |
| D.04 | 1000 | 1250 | – | – | 2.40 | 3.34 |
| D.05 | 1000 | 1250 | 1.20 | 0.83 | 2.49 | 1.10 |
| D.11 | 1000 | 5000 | 0.51 | 0.98 | 0.54 | 0.53 |
| Average | | | 1.77 | 1.09 | 1.07 | 1.41 |

(–) means that the instance remained unsolved after reaching the maximum time limit of 3600 s.

**Table 7**
Impact of the preprocessing routines.

| Class B | | Class C | | Class D | |
|---|---|---|---|---|---|
| *Inst.* | Time ratio | *Inst.* | Time ratio | *Inst.* | Time ratio |
| B.01 | 0.15 | C.01 | **1.22** | D.01 | **unsolved** |
| B.02 | 0.06 | C.02 | 0.16 | D.02 | **unsolved** |
| B.03 | 0.00 | C.03 | 0.81 | D.03 | **unsolved** |
| B.04 | 0.30 | C.04 | 0.10 | D.04 | **unsolved** |
| B.05 | 0.06 | C.05 | 0.12 | D.05 | 0.43 |
| B.06 | 0.11 | C.06 | 0.12 | D.06 | **unsolved** |
| B.07 | 0.11 | C.07 | 0.04 | D.07 | **7.96** |
| B.08 | 0.05 | C.08 | **2.19** | D.08 | 0.09 |
| B.09 | 0.11 | C.09 | 0.02 | D.09 | **1.26** |
| B.10 | 0.06 | C.10 | 0.07 | D.10 | **unsolved** |
| B.11 | **1.38** | C.11 | 0.12 | D.11 | 0.37 |
| B.12 | 0.05 | C.12 | 0.06 | D.12 | **unsolved** |
| B.13 | 0.08 | C.13 | 0.42 | D.13 | **unsolved** |
| B.14 | 0.11 | C.14 | **2.39** | D.14 | **unsolved** |
| B.15 | 0.09 | C.15 | **unsolved** | D.15 | 0.03 |
| B.16 | 0.06 | C.16 | **unsolved** | D.16 | **unsolved** |
| B.17 | 0.07 | C.17 | 0.40 | D.17 | **3.49** |
| B.18 | 0.29 | C.18 | **3.49** | D.18 | 0.14 |
| | | C.19 | **unsolved** | D.19 | **unsolved** |
| | | C.20 | **unsolved** | D.20 | **unsolved** |

instance. However, for the instances of Class C, we observe that the CPU times increased for four instances and that four instances remained unsolved. Finally, we observe a significant degradation of the algorithmic performance for the instances of Class D. Indeed, we see that the CPU times significantly increased for three instances and, more importantly, twelve instances remained unsolved after reaching the maximum CPU time limit. This portends the utility of the preprocessing routines for solving large sized instances of PCSTP.

*Performance on the quota problem*

We also tested our approach on an important special case, namely the quota problem (Q-PCSTP) where no penalty is incurred for noncovered nodes. It is worth noting that even though this problem was first introduced a decade ago, we are not aware of any exact algorithm for solving it. We considered a test-bed of 15 Q-PCSTP Euclidean instances that were randomly generated as described in Haouari and Chaouachi [11]. Each instance is characterized by a number of nodes $n$ (ranging from 100 to 500), and a number of edges $m$ taken equal to $3n$, $6n$, and $10n$, respectively. Prizes were derived from a discrete uniform distribution $U[1, 30]$. For each instance, the corresponding quota was set to $Q = \left\lceil \alpha \sum_{j \in V \setminus \{1\}} p_j \right\rceil$, where the parameter $\alpha$ was fixed to 0.4, 0.6, and 0.8, respectively. Thus, we obtained three different problem sets, each including 15 instances and being characterized by a distinct value of $\alpha$. The results are displayed in Table 8 (the column headings are similar to those of Table 1).

Interestingly, we observe from this table, that the proposed approach exhibits a good performance on Q-PCSTP instances, being able to solve exactly relatively large-size instances within a reasonable CPU effort. Indeed, 42 instances (out of 45)

**Table 8**
Performance on the Quota problem.

| | | $\alpha = 0.4$ | | | | | $\alpha = 0.6$ | | | | | $\alpha = 0.8$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Inst.* | *n:m* | $Z_{ISPH}$ | *%Gap ISPH* | $Z^*$ | *Time* | *Time ISPH* | $Z_{ISPH}$ | *%Gap ISPH* | $Z^*$ | *Time* | *Time ISPH* | $Z_{ISPH}$ | *%Gap ISPH* | $Z^*$ | *Time* | *Time ISPH* |
| Q.01 | 100:300 | 6787 | 1.54 | 6684 | 6.58 | 0.67 | 11645 | 0.93 | 11538 | 2.34 | 0.17 | 17728 | 0.59 | 17624 | 1.94 | 0.16 |
| Q.02 | 100:600 | 6068 | 3.39 | 5869 | 7.65 | 0.26 | 9975 | 4.05 | 9587 | 9.87 | 0.45 | 14100 | 0.06 | 14092 | 5.86 | 0.33 |
| Q.03 | 100:1000 | 4882 | 8.49 | 4500 | 13.62 | 0.55 | 7593 | 6.43 | 7134 | 22.51 | 0.36 | 10351 | 1.92 | 10156 | 9.42 | 0.49 |
| Q.04 | 200:600 | 14409 | 1.12 | 14250 | 15.04 | 1.34 | 23691 | 0.38 | 23602 | 41.78 | 1.94 | 36292 | 0.89 | 35973 | 14.51 | 1.58 |
| Q.05 | 200:1200 | 10360 | 1.06 | 10251 | 59.76 | 1.05 | 17841 | 1.32 | 17609 | 165.74 | 1.87 | 27213 | 0.50 | 27078 | 91.40 | 11.67 |
| Q.06 | 200:2000 | 8791 | 2.72 | 8558 | 25.21 | 3.75 | 14312 | 1.12 | 14154 | 17.04 | 1.22 | 21494 | 0.71 | 21342 | 21.09 | 2.24 |
| Q.07 | 300:900 | 20173 | 0.54 | 20065 | 26.76 | 13.88 | 34983 | 0.28 | 34886 | 26.26 | 7.66 | 53712 | 0.07 | 53677 | 42.51 | 5.87 |
| Q.08 | 300:1800 | 12611 | 1.10 | 12474 | 41.11 | 10.31 | 22012 | 0.44 | 21915 | 46.46 | 10.78 | 34491 | 0.00 | 34491 | 25.17 | 3.19 |
| Q.09 | 300:3000 | 10708 | 1.63 | 10536 | 115.43 | 11.95 | 18123 | 0.62 | 18011 | 63.23 | 8.20 | 28630 | 0.15 | 28588 | 44.43 | 11.48 |
| Q.10 | 400:1200 | 26234 | 0.41 | 26127 | 34.95 | 11.21 | 44839 | 0.29 | 44710 | 34.61 | 13.80 | 69395 | 0.21 | 69252 | 31.84 | 18.48 |
| Q.11 | 400:2400 | 19803 | 2.01 | 19413 | 82.23 | 29.13 | 33296 | 0.36 | 33175 | 112.45 | 32.64 | 51322 | 0.29 | 51176 | 52.43 | 14.00 |
| Q.12 | 400:4000 | 16494 | 2.12 | 16151 | 174.70 | 45.33 | 26449 | 0.27 | 26377 | 100.78 | 29.08 | 39663 | 0.15 | 39605 | 70.15 | 22.03 |
| Q.13 | 500:1500 | 33676 | 0.72 | 33435 | 352.45 | 30.09 | 57550 | 0.12 | 57482 | 74.31 | 43.14 | 89817 | 0.06 | 89766 | 732.89 | 26.02 |
| Q.14 | 500:3000 | 24698 | 0.13 | – | – | 40.70 | 41968 | 0.58 | – | – | 58.17 | 63465 | 0.37 | – | – | 35.55 |
| Q.15 | 500:5000 | 18948 | 3.16 | 18368 | 200.83 | 44.91 | 31967 | 0.53 | 31797 | 145.32 | 31.22 | 48814 | 0.26 | 48688 | 148.528 | 54.92 |
| Avg | | | 2.01 | | 82.59 | 16.34 | | 1.18 | | 61.62 | 16.05 | | 0.41 | | 92.30 | 13.87 |

were solved within the maximum time limit. Actually, all the three unsolved instances were derived from instance Q.14 and therefore have the same underlying graph and prizes. Moreover, we see that the performance of the iterated span-and-prune heuristic tends to deteriorate as the parameter $\alpha$ decreases. This is consistent with the observation made in Haouari and Chaouachi [11] for the performance of the hybrid Lagrangian genetic algorithm that is described therein.

*Impact of the penalty/prize ratio*

As a final experiment, we investigated the impact of the penalty/prize ratios on the overall performance of our approach. To that aim, we considered a restricted subset of the five first instances of Class C. All these instances have 500 nodes and 625 edges. For each node $j \in V \setminus \{1\}$, we set the corresponding penalty as $\gamma_j = \lceil \beta p_j \rceil$ where the parameter $\beta$ was fixed to 0, 0.5, 1, and 2, respectively. The results are displayed in Table 9.

Not surprisingly, we observe that as the parameter $\beta$ increases the problem becomes easier to solve (as the solution tends to be a minimum spanning tree). It is noteworthy that the largest CPU times were achieved with $\beta = 0.5$, which was the parameter setting used in all the aforementioned experiments.

## 7. Conclusion

In this paper, we have addressed the Prize Collecting Steiner Tree Problem (PCSTP). We have presented two 0-1 programming formulations and have developed two preprocessing procedures based on Lagrangian decomposition and LP relaxations for reducing the size of the underlying graph. A second principal contribution of this paper is the development of an optimization-based heuristic that requires solving a PCSTP on a tree. This latter special case was shown to be $\mathcal{NP}$-hard, but solvable in pseudo-polynomial time after reformulating it as a longest path problem with an additional knapsack side-constraint over a specified acyclic digraph. Although, the worst-case performance of the proposed heuristic was shown to be unbounded, it was demonstrated that the iterated variant yields excellent results in practice for most test instances. Finally, we have described an exact row-generation approach for solving a directed cut-based formulation of PCSTP. This latter formulation was augmented with several classes of valid inequalities that serve to tighten its LP relaxation. Our numerical experiments provide strong empirical evidence that the efficient combination of Lagrangian decomposition, graph reduction procedures, heuristic solution, valid inequalities, and row-generation-based MIP solution strategies enable the optimal solution of large-scale PCSTP instances having up to 1000 nodes and 25,000 edges. Moreover, we have reported on the solution of Euclidean Q-PCSTP instances having up to 500 nodes and 5000 edges.

As a topic for future research, we recommend the derivation of additional classes of valid (or facet-defining) inequalities that might prove useful for accelerating the convergence of the algorithm.

## Acknowledgements

This research has been supported in part by the *National Science Foundation* under grant CMMI-0552676. The authors also thank two anonymous referees for a number of careful remarks and insightful comments that have helped to improve

**Table 9**
Impact of the penalty/prize ratio.

| | $\beta = 0$ | | | | | $\beta = 0.5$ | | | | | $\beta = 1$ | | | | | $\beta = 2$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Inst.* | $Z_{ISPH}$ | *%Gap ISPH* | $Z^*$ | *Time* | *Time ISPH* | $Z_{ISPH}$ | *%Gap ISPH* | $Z^*$ | *Time* | *Time ISPH* | $Z_{ISPH}$ | *%Gap ISPH* | $Z^*$ | *Time* | *Time ISPH* | $Z_{ISPH}$ | *%Gap ISPH* | $Z^*$ | *Time* | *Time ISPH* |
| C.01 | 695 | 0.58 | 691 | 34.44 | 25.50 | 1381 | 0.15 | 1379 | 166.67 | 19.48 | 2026 | 0.00 | 2026 | 17.00 | 2.97 | 2329 | 0.00 | 2329 | 7.74 | 2.17 |
| C.02 | 719 | 0.98 | 712 | 132.08 | 123.72 | 1418 | 0.21 | 1415 | 51.57 | 28.31 | 2046 | 0.00 | 2046 | 9.22 | 3.22 | 2263 | 0.00 | 2263 | 5.80 | 1.58 |
| C.03 | 738 | 0.68 | 733 | 39.38 | 30.38 | 1390 | 0.36 | 1385 | 520.58 | 46.37 | 2007 | 0.05 | 2006 | 9.19 | 3.31 | 2259 | 0.00 | 2259 | 6.64 | 1.69 |
| C.04 | 714 | 0.42 | 711 | 39.33 | 16.53 | 1415 | 0.00 | 1415 | 71.80 | 58.34 | 2052 | 0.00 | 2052 | 28.30 | 3.24 | 2319 | 0.00 | 2319 | 23.30 | 2.05 |
| C.05 | 677 | 0.30 | 675 | 35.08 | 26.36 | 1363 | 0.37 | 1358 | 40.86 | 25.91 | 2017 | 0.00 | 2017 | 5.54 | 3.30 | 2295 | 0.00 | 2295 | 6.54 | 1.69 |
| Average | | 0.59 | | 56.06 | 44.50 | | 0.22 | | 170.30 | 35.68 | | 0.01 | | 13.85 | 3.21 | | 0.00 | | 10.00 | 1.84 |

the presentation in this paper. Dr. Mohamed Haouari would like to thank Fatimah Alnijris Research Chair for Advanced Manufacturing Technology for the financial support provided for this research.

## References

[1] D.S Johnson, The NP-completeness column: An ongoing guide, Journal of Algorithms 6 (1985) 434–451.
[2] G.W. Klau, I. Ljubić, P. Mutzel, U. Pferschy, R. Weiskircher, The fractional prize-collecting Steiner tree problem on trees, Extended Abstract, ESA 2003, 2003, pp. 691–702.
[3] E. Balas, The prize-collecting traveling salesman problem, Networks 19 (1989) 621–636.
[4] D. Bienstock, M. Goemans, D. Simchi-Levi, D. Williamson, A note on the prize collecting traveling salesman problem, Mathematical Programming 59 (1993) 413–420.
[5] M.X. Goemans, D.P. Williamson, The primal-dual method for approximation algorithms and its application to network design problems, in: D.S. Hochbaum (Ed.), Approximation Algorithms for NP-hard Problems, PWS Publishing Company, Boston, 1997, pp. 144–191.
[6] D.S. Johnson, M. Minkoff, S. Philips, The prize collecting Steiner tree problem: Theory and practice, in: Proceedings of the 11th ACM-SIAM Symposium on Discrete Mathematics, San Fransisco, CA, 2000, pp. 760–769.
[7] S.A. Canuto, M.G.C. Resende, C.C. Ribeiro, Local search with perturbations for the prize-collecting Steiner tree problem in graphs, Networks 38 (2001) 50–58.
[8] A. Lucena, M.G.C. Resende, Strong lower bounds for the prize collecting Steiner problem in graphs, Discrete Applied Mathematics 141 (2004) 277–294.
[9] A. da Cunha, A. Lucena, N. Maculan, M. Resende, A relax-and-cut algorithm for the prize-collecting Steiner problem in graphs, Discrete Applied Mathematics 157 (2009) 1198–1217.
[10] I. Ljubić, R. Weiskircher, U. Pferschy, G. Klau, P. Mutzel, M. Fischetti, An algorithmic framework for the exact solution of the prize-collecting Steiner tree problem, Mathematical Programming 105 (2006) 427–449.
[11] M. Haouari, J. Chaouachi, A hybrid Lagrangian genetic algorithm for the prize collecting Steiner tree problem, Computers and Operations Research 33 (2006) 1274–1288.
[12] M. Haouari, S. Layeb, H.D. Sherali, The prize collecting Steiner tree problem: Models and Lagrangian dual optimization approaches, Computational Optimization and Applications 40 (2008) 13–39.
[13] J.E. Beasley, An SST-based algorithm for the Steiner problem in graphs, Networks 19 (1989) 1–16.
[14] A. Segev, The node-weighted Steiner tree problem, Networks 17 (1987) 1–17.
[15] S. Chopra, C.Y. Tsai, Polyhedral approaches for the Steiner tree problem on graphs, in: X. Cheng, D.-Z. Du (Eds.), Steiner Trees in Industry, in: Combinatorial Optimization, vol. 11, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2001, pp. 175–202.
[16] M. Fischetti, Facets of two Steiner arborescence polyhedra, Mathematical Programming 51 (1991) 401–419.
[17] T. Polzin, S. Vahdati Daneshmand, A comparison of Steiner tree relaxations, Discrete Applied Mathematics 112 (2001) 241–261.
[18] Y. Ikura, G.L. Nemhauser, An efficient primal simplex algorithm for maximum weighted vertex packing on bipartite graphs, Annals of Discrete Mathematics 16 (1982) 149–168.
[19] H.D. Sherali, O. Ulular, Conjugate gradient methods using quasi-Newton updates with inexact line searches, Journal of Mathematical Analysis and Applications 150 (1990) 359–377.
[20] C.W. Duin, A. Volgenant, Some generalizations of the Steiner problem in graphs, Networks 17 (1987) 353–364.
[21] R.K. Ahuja, T.L. Magnanti, J.B. Orlin, Network Flows: Theory, Algorithms, and Applications, Prentice Hall, Upper Saddle River, New Jersey, 1993.
[22] H.C. Joksch, The shortest route problem with constraints, Journal of Mathematical and Analytical Applications 14 (1966) 191–197.
[23] G.Y. Handler, I. Zang, A dual algorithm for the constrained shortest path problem, Networks 10 (1980) 293–310.
[24] T. Koch, A. Martin, S. Voss, SteinLib: An updated library on Steiner tree problems in graphs, Technical Report ZIB-Report 00-37, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Takustr. 7, Berlin, 2000.