The Multi-period Prize-collecting Steiner Tree Problem with Budget Constraints

Larissa F. T. Faria[a,b,e,*], Jean-François Cordeau[c,e], Sanjay Dominik Jena[d,e], Hélio Lopes[a], David Sotelo[b]

[a]*Department of Computer Science, PUC-Rio, Brazil*
[b]*Operations Research and Data Science, Petrobras, Brazil*
[c]*Department of Logistics and Operations Management, HEC Montréal, Canada*
[d]*Department of Analytics, Operations and Information Technology, School of Management, ESG UQAM, Montréal, Canada*
[e]*Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation, Montréal, Canada*

Abstract

The classical Prize-collecting Steiner Tree Problem aims at finding a connected subgraph that maximizes the revenues collected from connected vertices minus the costs to utilize the connecting edges. We consider a multi-period variant in which, additionally: (a) vertices are allowed to be added to the tree at different time periods; (b) a predefined budget is imposed on edges selected over specified sets of time periods; and (c) the total length of the edges that can be added over a time period is limited. We propose a branch-and-cut algorithm that satisfactorily solves in reasonable time benchmark instances from the literature, adapted to a multi-period setting, with up to 3300 vertices and 300 terminal vertices.

Keywords: Prize-collecting Steiner Tree, Multi-Period, Branch-and-Cut, Network Design

1. Introduction

The Steiner Tree Problem (STP), also called Steiner Tree Problem in Graphs (Lucena and Beasley, 1998; Lucena and Resende, 2004; Rosseti et al., 2003), consists of building a tree of minimal weight from an undirected graph with non-negative edge weights and a subset of its vertices (named terminals). A feasible solution must include all terminal vertices, but may also contain additional non-terminal vertices.

Given a graph G which associates a prize to each vertex and a cost to each edge, the Prize-collecting Steiner Tree Problem (PCST) aims to build a connected subgraph of G, maximizing the sum of the selected vertices' prizes reduced by the sum of the selected edges' costs (Ljubić et al., 2005). The PCST is NP-hard (Karp, 1972), but several exact methods have been proposed to solve large instances to optimality (Canuto et al., 2001; Da Cunha et al., 2009; Feofiloff et al., 2007; Fischetti et al., 2017; Gamrath et al., 2017; Johnson et al.,

*Corresponding author
Email addresses: lfaria@inf.puc-rio.br (Larissa F. T. Faria), jean-francois.cordeau@hec.ca (Jean-François Cordeau), jena.sanjay-dominik@uqam.ca (Sanjay Dominik Jena), lopes@inf.puc-rio.br (Hélio Lopes), david@petrobras.com.br (David Sotelo)

2000; Klau et al., 2004; Ljubić et al., 2005, 2006; Lucena and Resende, 2004). Furthermore, real-world problems related to network expansion can be modeled based on the PCST, leading to variants of this classical problem (Arulselvan et al., 2011; Costa et al., 2006, 2009; Gollowitzer and Ljubić, 2011; Suhl and Hilbert, 1998). In this paper, we propose an exact method for a variant of the PCST, denoted as the Multi-period Prize-collecting Steiner Tree Problem with Budget Constraints (MPCSTB). This problem takes into account three additional elements when compared to the classical PCST:

1. Vertices and edges can be added to the solution in different time periods, chosen from a discrete set forming the time horizon. The prize associated to the insertion of a vertex may depend on the time period in which it is added.
2. Budgets may be defined for different sets of time periods, limiting the sum of the costs of the edges that can be added during specific periods of the planning horizon.
3. The total length of edges added to the subgraph may be limited over each time period.

These elements are important to consider when attempting to model a network design problem, for example, for the natural gas industry. In this context, a physical network is composed by pairs of cities which are joined by pipeline stretches. The distribution center can be modeled as the root of the pipeline network. Therefore, a city is said to be connected to the pipeline network if there is a pipeline path from the distribution center leading to this city. Natural gas flows in both directions of a pipeline stretch, leading to the pipeline network representation as an undirected graph. Naturally, various cities can be connected at different time periods, as long as a path is available due to the construction of pipeline stretches to guarantee the connection. The period in which a city is incorporated into the network results in different profits, usually related to the demand of that city, from the corresponding period to the end of the time horizon. These profits are represented as prizes, which are associated to the vertices of the undirected graph (i.e., the cities they represent). Similarly, building a new pipeline stretch involves costs that are associated to the corresponding edge of the undirected graph. At each set of time periods, budget constraints may restrict the maximum amount that can be spent on building pipeline stretches. Further, physical and logistical restrictions may require a distance limit on the maximum stretch length that can be built at each time period. Overall, the problem at hand can thus be used to model the expansion of a gas network throughout a number of future periods, maximizing the difference between the sum of the profits of the incorporated cities and the cost of the new pipeline stretches built.

Problem definition: We now formally define the problem considered in this paper.

Definition 1 (Multi-period Prize-collecting Steiner Tree Problem with Budget Constraints, MPCSTB)**.** Let $T = \{1, \ldots, |T|\}$ be a *time horizon* over which is defined a function *distanceLimit* $: T \to \mathbb{Q}^+$. Let $\hat{T}_B = \{T_B\}$, where $T_B \subseteq T$ is a subset of the time horizon over which is defined a function *budgetLimit* $: \hat{T}_B \to \mathbb{Q}^+$. Let $G = (V, E)$ be an undirected graph with a *revenue function* $r : (V, T) \to \mathbb{Q}^+$ defined on its vertices, a *cost function* $c : (E, T) \to \mathbb{Q}^+$ defined on its edges and a *distance function* $d : E \to \mathbb{Q}^+$ defined on its edges. There is a specially identified root vertex $v_0 \in V$ that represents all vertices that are already connected to the network. Furthermore, let $Z = (V_Z, E_Z)$ be a subgraph of G with functions $\alpha : V_Z \to T$ and $\beta : E_Z \to T$ mapping its vertices and edges to the time horizon, respectively. Finally, let $Z_t = (V_{Z_t}, E_{Z_t})$ be the subgraph of Z where $V_{Z_t} = \{v \in V_Z \mid \alpha(v) \leq t\}$

and $E_{Z_t} = \{e \in E_Z \mid \beta(e) \leq t\}$. The MPCSTB consists of finding a subgraph Z and the corresponding functions α and β, which maximize:

$$profit(Z) = \sum_{v \in V_Z} r(v, \alpha(v)) - \sum_{e \in E_Z} c(e, \beta(e)) \tag{1}$$

subject to:

$$\sum_{e \in E_{Z_t}} c(e, \beta(e)) \leq budgetLimit_{T_B},\ \forall T_B \in \hat{T}_B \tag{2}$$

$$\sum_{e \in E_{Z_t} \setminus E_{Z_{t-1}}} d(e) \leq distanceLimit_t,\ \forall t \in T \tag{3}$$

and Z_t is connected.

The set of all vertices V can be divided into terminal vertices, representing vertices that have profit greater than zero, and Steiner vertices, that represent vertices that have profit equal to zero. Figure 1 illustrates an example of a MPCSTB instance (based on a PCST instance given in Ljubić et al. (2006)) with three time periods. Each edge has fixed costs and a length (in kilometers), hollow circles represent terminal vertices and filled circles represent Steiner vertices. Each time period has a distance limit of 11 kilometers. The three-period time horizon has a budget limit of 100 units. Figure 2 shows a feasible state of the network at the first period of the planning horizon, Figure 3 shows a feasible network state at the second period and Figure 4 shows the final feasible, but not optimal, solution for all three periods.

Academic contributions: A slightly simpler problem than the one considered here has been introduced by Suhl and Hilbert (1998), who use a branch-and-cut algorithm to solve an integer programming formulation based exclusively on edge decision variables. We extend the problem by incorporating budget constraints that can be imposed for different subsets of time periods, which are common in projects such as those related to transportation and energy infrastructure expansion. We provide an intuitive mathematical formulation using both vertex and edge variables, and propose a branch-and-cut algorithm to solve the model. In contrast to undirected generalized subtour elimination constraints (as used by Suhl and Hilbert (1998)), our mathematical formulation considers cut constraints. Our branch-and-cut algorithm uses two separation procedures, an integer and a fractional procedure, whereas Suhl and Hilbert (1998)'s work has no mention of a fractional separation procedure. We also present a primal heuristic that considerably improves the upper bounds. We have reimplemented Suhl and Hilbert (1998)'s model to compare results obtained on two different sets of instances. One of these was artificially generated while the other one is based on real-world instances from a gas company. Our algorithm outperforms Suhl and Hilbert (1998)'s results for the artificial instances, where we solve instances of size up to 3300 vertices and 18073 edges. On the instance set inspired by real-world data, both models present similar performance. In summary, our exact method is able to solve reasonably large instances for a natural extension of the PCST, finding solutions of high quality for realistic problem sizes in a reasonable amount of computing time. The computational experiments focus on evaluating the performance of our model and on exploring the impact of different budget limitations.

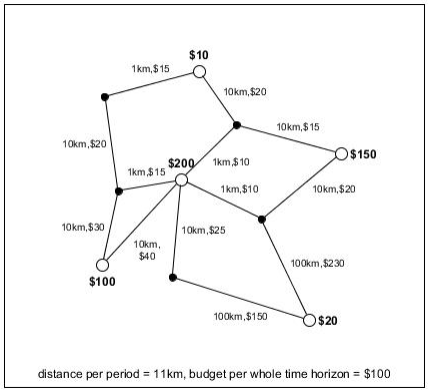

Figure 1: Example of a MPCSTB instance.

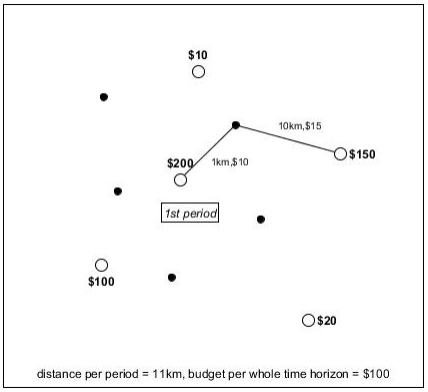

Figure 2: First period feasible network.

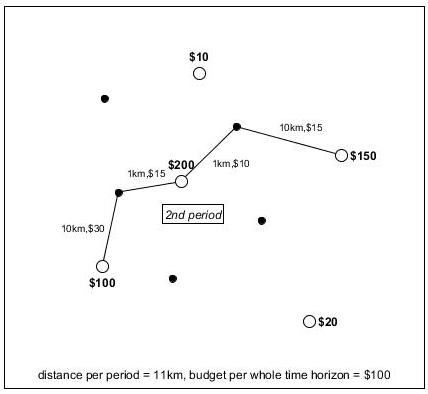

Figure 3: Second period feasible network.

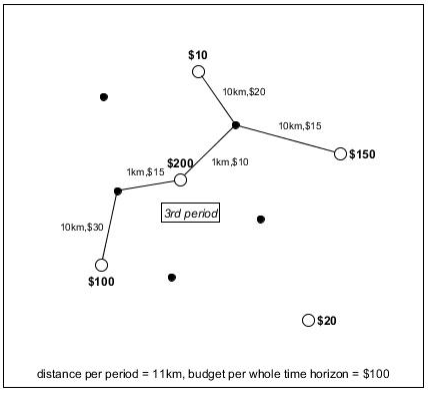

Figure 4: Feasible, but not optimal network.

Paper outline: The remainder of the article is organized as follows: Section 2 provides a review on the literature related to the problem. Section 3 introduces a mixed-integer linear programming formulation while Section 4 provides a branch-and-cut algorithm for the problem. Computational results are reported in Section 5. Finally, Section 6 draws conclusions and indicates possible future work directions.

2. Related Literature

2.1. Network design problems

The MPCSTB generalizes the PCST, and therefore extends classical network design problems. Network design problems aim at identifying a subset of edges in a graph that have minimum total cost while satisfying all constraints, for example, those required to connect pairs of origin and destination vertices in order to route commodities. Solution algorithms for these problems can be classified into exact and heuristic approaches. Examples of common exact techniques used include cutting planes (Poss, 2012) and branch-and-bound (Humpola and Fügenschuh, 2015). Additionally, when modeling a network design problem, its operation, its expansion or both may be of interest. Depending on the application context,

flow-related constraints may be added. Borraz-Sánchez et al. (2016), for example, are interested in finding the optimal solution that would capture physical, operational or even contractual constraints. Their work focuses on adjusting the network's parameters, including regular pipelines, valves, short pipes, control valves, compressor stations, and regulators, instead of focusing solely on the network expansion, as we do here.

2.2. PCST

Our problem, the MPCSTB, is an extension of the PCST. Lucena and Resende (2004) propose an integer programming formulation of the PCST problem and describe an algorithm based on polyhedral cutting planes to obtain lower bounds for the problem. Later, Ljubić et al. (2005) introduced a branch-and-cut algorithm based on a directed graph model where they efficiently separate sets of violated inequalities using a maximum flow algorithm. Ljubić et al. (2006) aim to solve large and difficult instances of the PCST to optimality in reasonable computing times using a branch-and-cut algorithm that adds violated cut constraints between an artificial root and the selected customer vertices.

Uchoa (2006) suggests applying redefined reduction tests, proven to be effective on the Steiner Tree Problem in Graphs. Da Cunha et al. (2009) generate primal and dual bounds to the problem, making use of a Lagrangian Non Delayed Relax and Cut algorithm. Finally, even if the PCST has proven to be a challenging NP-hard problem, Fischetti et al. (2017) present a simple solution method and obtain high quality solutions for hard instances from the literature. They use a model that exclusively uses node variables, which proves to be successful for instances where all edges have the same cost.

Additionally, metaheuristics have been developed to find quality solutions to the PCST. Canuto et al. (2001) develop a multi-start local-search-based algorithm with perturbations. Klau et al. (2004) propose a memetic algorithm with incorporated local improvement. For a general overview of the methods developed to solve the PCST, we refer to Costa et al. (2006).

2.3. PCST with budget constraints

Several works have proposed to add budget constraints to the PCST. Johnson et al. (2000) define the so-called quota version of the PCST by searching for the tree with minimum total edge costs that contains vertices whose total prize is at least a given quota. Additionally, Johnson et al. (2000) consider the PCST with budget constraints, i.e., finding the tree with maximum prize, given that the total edge cost is within a given budget. The authors define the quota problem as a generalization of the k-Minimum Spanning Tree (k-MST) problem and propose to extend constant-factor approximation algorithms to solve it. The k-MST problem looks for a tree of minimum cost that has exactly k vertices, consequently forming a subgraph of a larger graph. For the (unrooted) budget problem, Johnson et al. (2000) propose to incorporate it into a practical heuristic, involving the performance of multiple runs of the Goemans-Williamson algorithm (Goemans and Williamson, 1997) and the use of an increasing sequence of prize multipliers.

Costa et al. (2009) define the Steiner tree problem with revenues, budget and hop constraints. This problem is a variant of the PCST problem with additional budget and hop constraints. Budget constraints impose limits on the total cost of the network, whereas hop constraints impose limits on the number of edges between each vertex and the root. For the

PCST with budget constraints, the authors show that branch-and-cut algorithms using cut constraints (instead of generalized subtour elimination constraints) obtain the best results to date. Also, for several variants of the Steiner Tree problem, directed models are proven to be easier to solve than their undirected counterpart (see, e.g., Chopra and Rao (1994a), Chopra and Rao (1994b), Feremans et al. (2002), Ljubić et al. (2005) and Magnanti and Raghavan (2005)).

2.4. Multi-period PCSTs with budget constraints

To the best of our knowledge, Suhl and Hilbert (1998) constitute the only attempt to solve the MPCSTB. A gas network is represented by an undirected graph. Vertex profits are represented by negative edge weights, allowing the authors to use a formulation exclusively based on edge variables. A part of the graph may already have been connected in previous periods and at every subsequent period, the solution must be a tree. The task is to maximize the profit obtained by connecting vertices to the network over a multi-period study horizon. Furthermore, budget and distance constraints restrict the number of node connections per time period. The authors make use of an integer programming formulation leading to a branch-and-cut algorithm, along with an optimization software system (called MOPS (Suhl, 1994)) for solving the large-scale problem.

In this paper, we attempt to solve the MPCSTB with more general budget constraints using branch-and-cut with two separation procedures: one to separate integer solutions and another to separate fractional ones. In addition, we apply an algorithm to tighten the upper bound of the problem, helping to prove optimality. The mathematical formulation is given in Section 3 and the solution method is described in Section 4.

3. Mathematical Formulation

In this section, we present an integer programming formulation for the MPCSTB. Even though the MPCSTB assumes that edges are undirected, we provide a formulation that is based on directed arcs, given that those formulations have been shown to provide stronger linear programming (LP) relaxation bounds (see, e.g., Chopra and Rao (1994a,b); Feremans et al. (2002); Fischetti (1991); Goemans and Myung (1993); Ljubić et al. (2005); Magnanti and Raghavan (2005); Magnanti and Wolsey (1995)).

We denote by T the planning horizon (for example, 2021 to 2025), composed of time periods $t \in T$ (which may, for example, represent one year each). We also denote by $\hat{T}_B = \{T_B\}$ the set of all subsets of consecutive time periods from T. Let $G = (V, A)$ be a directed graph with vertex set $V = \{0, ..., n\}$ and arc set $A = \{a = (i, j) : i, j \in V\}$, where each arc $a \in A$ has an associated construction cost c_a^t, depending on the time period t when it is constructed. For $W \subseteq V$, define A_W as the set of arcs with both endpoints in W. We assume that there is a root vertex, denoted as $v_0 \in V$ that represents all vertices that are already connected to the network at the beginning of the planning horizon. If the instance does not have an actual root vertex, an artificial one will be created. This root vertex v_0 is assumed to be available throughout all time periods. In order to guarantee that the final network is connected, one needs to ensure that all selected vertices are connected to the root vertex v_0. To this end, let $\delta^-(W) := \{(i, j) \in A | i \notin W, j \in W\}, \forall W \subseteq V$, which denotes the set of arcs that have their origin vertex outside W, and their destination vertex within W.

We introduce binary variables $y_i^t \in \{0,1\}, \forall i \in V, \forall t \in T$, which take value 1 if vertex i is connected for the first time in time period t, and 0 otherwise. We also use binary arc variables $x_{ij}^t \in \{0,1\}, \forall (i,j) \in A, \forall t \in T$, which take value 1 if arc (i,j) is constructed (for the first time) in time period t, and 0 otherwise. Constant c_{ij}^t denotes the cost to install arc (i,j) in the beginning of time period t. The arc installation costs need to be payed only once: at the time period when they are built. However, note that the arc installation at an early time period may involve maintenance costs for the following time periods. Constants r_i^t represent the revenues collected when connecting vertex i to the existing network in period t. Note that the revenue constant may also contain revenues from subsequent time periods. Constants d_{ij} denote the distances between vertex i and vertex j.

In the Steiner tree literature, the profit function (1) given in Section 1 is known as the *Net Worth Maximization* function (Johnson et al., 2000). Equivalently, the so-called *Goemans and Williamson Minimization* function (Goemans and Williamson, 1997) aims at finding a subtree that minimizes the objective function $\sum_T (\sum_{v \notin V_Z} r_t(v) + \sum_{e \in A_Z} c_t(a))$. In this work, we use the *Goemans and Williamson Minimization* function, given by Equation (4), as it is common in the literature (see, e.g., Canuto et al. (2001), Goemans and Williamson (1997), Ljubić et al. (2005) and Ljubić et al. (2006)). The MPCSTB problem can be formulated as follows:

$$Min \sum_{t \in T} \Big(\sum_{i \in V} r_i^t (1 - y_i^t) + \sum_{(i,j) \in A} c_{ij}^t x_{ij}^t \Big) \tag{4}$$

Subject to:

Cut constraints

$$\sum_{t'=1}^{t} \sum_{(u,v) \in \delta^-(W)} x_{uv}^{t'} \geqslant \sum_{t'=1}^{t} y_i^{t'}, \ \forall W \subseteq V \setminus \{v_0\}, i \in W, t \in T \tag{5}$$

Multi-period constraints

$$\sum_{t \in T} x_{ij}^t \leqslant 1, \ \forall (i,j) \in A \tag{6}$$

$$\sum_{t \in T} y_i^t \leqslant 1, \ \forall i \in V \tag{7}$$

Side constraints

$$\sum_{t \in T_B} \sum_{(i,j) \in A} c_{ij}^t x_{ij}^t \leqslant budgetLimit^{T_B}, \ \forall T_B \in \hat{T}_B \tag{8}$$

$$\sum_{(i,j) \in A} d_{ij} x_{ij}^t \leqslant distanceLimit^t, \ \forall t \in T \tag{9}$$

Connectivity constraints

$$\sum_{t'=1}^{t} \sum_{j \in V} x_{ji}^{t'} \geqslant \sum_{t'=1}^{t} y_i^{t'}, \ \forall i \in V \setminus \{v_0\}, \forall t \in T \tag{10}$$

Variable domains

$$x_{ij}^t \in \{0,1\}, \forall (i,j) \in A, t \in T \tag{11}$$

$$y_i^t \in \{0,1\}, \forall i \in V, t \in T. \tag{12}$$

The exponentially large set of cut constraints (5) ensures that, for each time period, all selected network vertices are also connected to the root vertex. A subset W of vertices i that does not contain the root vertex v_0 is created. While attempting to connect i to the network, the set of cut constraints forces that an arc coming from $\delta^-(W)$ is also connected, i.e., an arc that will connect the set where v_0 is present to the complementary set W where v_0 is not present. Inequalities (6) and (7) ensure that each vertex and arc can be selected at most once throughout the planning horizon. Constraints (8) express the maximum budget allowed for specified subsets of time periods and constraints (9) limit the total length of edges that can be added over each time period. The constraint set (10) guarantees that every selected vertex has exactly one predecessor on its path from the root. This connectivity constraint set is commonly seen in PCST formulations (Costa et al., 2009; Ljubić et al., 2005, 2006).

To improve the efficiency of the model, the following valid inequalities are added to the formulation above:

Valid inequalities

$$\sum_{t'=1}^{t} y_i^{t'} \geqslant \sum_{t'=1}^{t} x_{ij}^{t'}, \ \forall i \in V \setminus \{v_0\}, \forall (i,j) \in A, \forall t \in T \tag{13}$$

$$x_{ij}^t + x_{ji}^t \leqslant 1, \ \forall i \in V \setminus \{v_0\}, \forall (i,j) \in A, \forall t \in T. \tag{14}$$

The valid inequalities (13) act as connectivity constraints: if arc (i,j) is available at time period t, vertex i has to be connected to the network at that time period or before. Constraints (14) show that every arc adjacent to a vertex in the solution tree can be selected only in one direction. These inequalities are commonly included in PCST formulations (see, e.g., Ljubić et al. (2005, 2006)) and are a special case of the cut constraints (5), written in their equivalent GSEC form (Ljubić et al., 2006). Adding these inequalities, specially all at once, may enlarge the LP. However, as they do not have to be separated implicitly during the branch-and-cut algorithm, they present a speed-up that outweighs the enlargement of the LP.

Since an artificial root vertex may be used to represent the connected network, some related constraints may be added to the above formulation:

Artificial root vertex constraints

$$\sum_{j \in V} x_{v_0 j}^{t=1} = 1 \tag{15}$$

$$y_{v_0}^{t=1} = 1 \tag{16}$$

Symmetry constraints

$$x_{v_0 j}^{t=1} + y_i^{t=1} \leqslant 1, \ \forall (i,j) \in A \mid j > i. \tag{17}$$

Constraints (15) guarantee that only one arc is chosen among the artificial root and any other vertex. This artificial arc has zero cost and does not alter the objective function value. Likewise, the artificial root vertex has no revenue and therefore has no effect on the objective function value of the model. In the same manner, constraints (16) ensure that the artificial root vertex enters the connected network at time period 1, i.e., the first time period of the planning horizon, since the root vertex represents the previously connected network. Finally, constraints (17) impose that the vertex adjacent to the root is the one with the smallest index. These constraints aim at excluding a plethora of symmetric solutions, therefore considerably reducing the solution time in a branch-and-bound framework (Gamrath et al., 2017).

The generation of the entire set of cut constraints (5) constitutes a critical issue for the integer programming (IP) model. Depending on the size of the graph, the number of cut constraints can be too large and render the corresponding IP model intractable. The number of cut constraints (5) actually violated by an integer solution obtained from a reduced model (without constraints (5)) is typically rather small compared to the total number of potential constraints (5). It is therefore common to add only those constraints that are violated during the branch-and-bound algorithm.

It is important to note that two separation procedures are used in this model: one to separate integer infeasible solutions and another to separate fractional solutions, both of them based on the cut constraint set (5). The separation procedures exploit the fact that constraints (5) imply the connectivity of the root to all other selected vertices (Costa et al., 2009). During the separation phase applied at each node of the branch-and-bound tree, we add constraints of type (5) that are violated by the current LP-relaxation solution (Ljubić et al., 2005). Further information concerning the separation procedures is provided in Section 4.1.

3.1. A note on Generalized Subtour Elimination Constraints vs. Cut Constraints

The classical generalized subtour elimination constraints (GSECs) are used in the Dantzig-Fulkerson-Johnson formulation and were introduced by Dantzig et al. (1954) for the Travelling Salesman Problem.

In our formulation, we use cut constraints (5) to guarantee connectivity in the network. They ensure that, if there is a $W \subseteq V$ that includes a vertex i, but not the root vertex v_0 and the vertex i is connected, one of the arcs in the set of all incoming arcs in W must be chosen to be in the solution. Note that disconnectivity would imply the existence of a cut separating v_0 and i, which would clearly violate the corresponding cut constraint (Arulselvan et al., 2011).

Even though the lower bounding procedure presented in Lucena and Resende (2004) is based on undirected GSECs, Chopra and Rao (1994a) were able to show for the STP that directed GSECs prevail over directed counterparts of several other facet defining inequalities of the undirected (GSEC) formulation. Directed GSEC formulations are therefore preferable in practice (Ljubić et al., 2006), as directed models have demonstrated better results than the undirected counterparts for several variants of the STP (Chopra and Rao, 1994a,b;

Costa et al., 2009; Feremans et al., 2002; Ljubić et al., 2005; Magnanti and Raghavan, 2005). Moreover, Fischetti (1991) shows that the cut constraints (5) can be rewritten as a directed version of the undirected GSECs. Finally, the model chosen in this work (using cut constraints (5)) is also less dense than the equivalent model based on directed GSECs, which is computationally preferable within a branch-and-cut framework (Ljubić et al., 2005). A computational comparison of one model with the model using the undirected GSECs can be found in Section 5.

3.2. Comparison to existing MPCSTB formulation

To the best of our knowledge, Suhl and Hilbert (1998) constitutes the only work that models the MPCSTB. The authors use a formulation exclusively based on arc variables, transforming all vertex variables into arc variables. Such a transformation replaces each vertex variable y by an arc variable x, which means that their formulation has the exact same number of variables than ours. The authors use the undirected GSEC inequalities to exclude cycles at each time period, which, as discussed in Section 3.1, is a methodology corroborated by the literature (Chopra and Rao, 1994a,b; Costa et al., 2009; Feremans et al., 2002; Fischetti, 1991; Ljubić et al., 2005, 2006; Magnanti and Raghavan, 2005) to be weaker than the one we propose. Moreover, the authors use branch-and-cut and insert violated subtour elimination constraints if the IP solution presents disconnected cycles. However, they do not mention any fractional separation algorithm to dynamically identify the constraints that have to be added to the model. Further, given that their work does not propose a primal heuristic to produce upper bounds, the instances solved are small in comparison to those that are solved to optimality in our work, as shown in Section 5. Finally, our side constraints that consider the budget limit (8) are broader than those considered in Suhl and Hilbert (1998) due to the flexibility of choosing subsets that may contain multiple time periods.

4. Branch and Cut Algorithm

4.1. Separation algorithms

An efficient separation of violated inequalities is crucial to tackle complex problem instances. We separate the constraints of type (5) during the optimization process using the separation procedures described in Sections 4.1.1 and 4.1.2.

4.1.1. Integer infeasible solutions

Our branch-and-cut approach includes cutting off infeasible integer points as well as infeasible fractional ones. We describe the algorithm used to separate the integer infeasible solutions. Such solutions may have been enumerated during the branching procedure or may even have been detected by the heuristics of the MIP solver, since the set of constraints (5) is not provided and therefore the solver is not given the complete structure of the problem.

In order to separate integer infeasible solutions, we find the connected components of a selected vertex. If these do not include the root vertex, we add the corresponding cut. The algorithm (see Algorithm 1) that separates the integer infeasible solutions has a complexity of $O(n+m)$, where n is the number of vertices and m is the number of arcs. It is important to mention that we employ a Breadth-first search (BFS) (Cormen et al., 2009) to identify all

connected nodes, starting at the tree root and exploring all neighbor nodes at the present depth prior to moving on to nodes at the next depth level.

Algorithm 1 Separation procedure at integer nodes

Input: The connected components of the current integer infeasible solution at time period t, found by a BFS.
Output: A set of violated inequalities incorporated into the current LP.
while !(Exist only one connected component including the root node) **do**
  for Each connected component that does not include the root node **do**
    Create set W that does not include the root node and contains the connected component.
    Create set $\overline{W} = V \setminus W$, complementary to set W, containing the root and all other vertices.
    for $w \in W$ **do**
      Add the violated cut $\sum_{t' \leq t} \sum_{v \in \overline{W}} x_{vw}^{t'} \geq \sum_{t' \leq t} y_w^{t'}$ to the LP.
    end for
  end for
end while

4.1.2. Fractional infeasible solutions

For fractional solutions, cut constraints are separated by calculating the maximum flow value (Fischetti et al., 2017). Maximum flow problems aim to find the maximum possible flow rate through a network. The max-flow min-cut theorem (see, e.g., Conforti et al. (2014)) states that the maximum value of a flow from source s to sink t is equal to the minimum capacity of an $s - t$ cut in a network.

For $W \subseteq V$, define A_W as the set of arcs with both endpoints in W. Denote $(\hat{x}, \hat{y})$ the corresponding LP relaxation solution. Then, a support graph $G_W = (W, A_W, \hat{x})$ is built as a result of the LP solution: the arc capacities are defined as $\hat{x}_{ij}^t$ for all $(i, j, t) \in A_W$ and the support graph vertices are those where $\hat{y}_i^t \neq 0$. Subsequently, the maximum flow is calculated from the root node v_0 to each vertex $i \in W$ that has $\hat{y}_i^t > 0$. A violated inequality is added to the LP for each maximum flow value smaller than $\hat{y}_i^t$. Such violated inequality is induced by the corresponding min-cut in the graph G_W (see, e.g., Gollowitzer and Ljubić (2011)).

The outline of the separation procedure is given in Algorithm 2. Ljubić et al. (2005) originally presented the procedure for a single time period. The separation algorithm is executed independently for every time period of the planning horizon.

The input of the algorithm is a support graph of the form $G_W = (W, A_W, \hat{x})$ that is built from the set of vertices $W \subseteq V$, the set of arcs A_W and the relaxed solution $(\hat{x}, \hat{y})$. We compute the maximum flow on the support graph for all (v_0, i) pairs of vertices, where $i \in W$ and $\hat{y}_i^t > 0$. Goldberg's implementation (Cherkassky and Goldberg, 1995) of the push-relabel maximum flow algorithm returns the maximum flow value $f = MaxFlow(G_W, \hat{x}, v_0, i, W_{v_0}, W_i)$, as well as sets $W_{v_0}, v_0 \in W_{v_0}$ and $W_i, i \in W_i$ that together define the minimum cut with value f (see, e.g., Ljubić et al. (2005)). Subset $W_{v_0} \subset W$ contains root vertex v_0 and induces a minimum cut closest to v_0. Therefore, as established by the max-flow min-cut theorem, $x(\delta^+(W_{v_0})) = f$. At the same time, subset $W_i \subset W$

Algorithm 2 Separation procedure at fractional nodes

Input: A support graph $G_W = (W, A_W, \hat{x})$.
Output: A set of violated inequalities incorporated into the current LP.
for $i \in W \mid \hat{y}_i^t > 0$ **do**
  $f = MaxFlow(G_W, \hat{x}, v_0, i, W_{v_0}, W_i)$;
  Detect the cut $\delta^+(W_{v_0})$ such that $\hat{x}(\delta^+(W_{v_0})) = f, v_0 \in W_{v_0}$;
  if $f < \hat{y}_i^t$ **then**
    Insert the violated cut $x(\delta^+(W_{v_0})) \geq y_i^t$ into the LP;
  end if
end for

contains vertex i and induces a minimum cut closest to i, i.e., $x(\delta^-(W_i)) = f$. Finally, if $f < \hat{y}_i^t$, we add the violated cut $x(\delta^+(W_{v_0})) \geq y_i^t$ to the model.

4.2. Primal heuristic

We use a primal heuristic to improve the upper bound of the problem (i.e., the best known integer feasible solution). Our heuristic is only called at the root node of the branch-and-bound tree, before the branching is performed, once the linear program is solved and no more violated inequalities are found. It is based on Ljubić et al. (2005)'s work and expanded to the multi-period setting with budget constraints.

The general idea of the heuristic is to pick the most promising vertices based on the LP relaxation solution and, using Kruskal's minimum spanning tree heuristic (Cormen et al., 2009), select a set of promising edges to connect these vertices. In order to respect the budget and distance constraints, a second step of the heuristic consists in deciding in which time period each of the selected vertices and arcs are built. The first step taken by the algorithm is the selection of a set of vertices S from graph $G = (V, A, c)$ that will be part of the heuristic solution. To select the most promising vertices, we use the information of the fractional values of the y variables in the LP relaxation solution of the current node in the branch-and-cut tree. For each vertex, we sum over the y values of all time periods. If the sum is greater than 0.5, the vertex is selected and added to set S.

Next, a distance network G_S is calculated for S, where $G_S = (S, S \times S, d_S)$. We define the length d_S of an edge in G_S as the length of the shortest path connecting the two corresponding vertices of the edge in G. The shortest path matrix is calculated using the Floyd-Warshall algorithm (see, e.g., Cormen et al. (2009)) based on modified edge length from G. Specifically, we determine the length of an edge connecting two vertices in S as the value 1 minus the solution value of that edge, which is defined next. The solution value of an edge is the maximum value between the solution values of the x variables in the LP solution in the two arcs that define that edge. To be precise, we assign to each edge (i, j, t) the cost $(1 - max\{\hat{x}_{ij}^t, \hat{x}_{ji}^t\})$ where $\hat{x}_{ij}^t$ is the value of the corresponding x variable in the fractional solution of the current branch-and-bound node. The shortest path is therefore based on the x variables that have high fractional values in the LP relaxation solution.

We then compute Kruskal's minimum spanning tree $Z = (S, A_Z)$ (Cormen et al., 2009) on G_S. Naturally, there are vertices on the shortest paths that correspond to arcs in A_Z. Therefore, we define the set S' of vertices in G as the union of S and the set of all vertices

that lie on the shortest paths. Consequently, $G_H = (S', A_H, c)$ is defined as the subgraph of G induced by the vertex set S'. Notice that in G_H, the cost of each arc is equivalent to the original cost in the problem instance. G_H is clearly connected, allowing us to compute Kruskal's minimum spanning tree (MST) $Z' = (S', A_{Z'})$. Given that this heuristic produces a single-period solution, we need to manually separate it into several time periods.

The second step therefore separates the solution given by the MST Z' into several time periods such that both budget and distance constraints are satisfied. We use a greedy algorithm based on the fractional solution values of the y variables for each vertex in Z'. The initial vertex is set as v_0, no matter whether the instance has an actual root node or if that root node is artificially created. According to the structure of the minimum spanning tree, there may be several vertices that can now be connected to i. If there is only one vertex, then this vertex is connected. If there are several vertices, the greedy algorithm will select the vertex i with the highest fractional y solution value. If at the end of the planning horizon there are still nodes from Z' to be inserted in the multi-period solution Z'', these nodes are discarded in order to respect the budget and distance limits constraints.

Our primal heuristic for the MPCSTB runs in linear time and is depicted in Algorithm 3. The heuristic solution is integer feasible, ensuring that the vertices in S' are connected (as guaranteed by the Floyd-Warshall algorithm). Moreover, they are connected in an efficient way through a minimum spanning tree, that, by definition, connects all vertices in S' without cycles and with the minimum possible total arc weight. We also guarantee to respect the side constraints, separating the resulting minimum spanning tree Z' by time periods, applying a greedy algorithm. This algorithm, while respecting the budget and limit constraints, chooses to insert in the heuristic solution the maximum number of vertices of Z' possible in the first period, the maximum number of remaining vertices of Z' in the second period and so forth. In sum, all constraints in the problem are respected, guaranteeing a viable heuristic solution.

Algorithm 3 Primal Heuristic

Input: Solution of LP relaxation $(\hat{x}, \hat{y})$.
Output: A heuristic solution Z''.
if $\sum_{t \in T} \hat{y}_i^t \geq 0.5$ **then**
    $S \leftarrow S \cup \{i\}$;
end if
Calculate $\hat{x}_{ij} = max\{\hat{x}_{ij}^t\} \mid \forall i, j \in S$;
Calculate $l_S = (1 - max\{\hat{x}_{ij}, \hat{x}_{ji}\}) \mid \forall i, j \in S$;
Calculate d_S = Floyd-Warshall(l_S);
Compute distance network $G_S = (S, S \times S, d_S)$;
Compute Kruskal's $Z = (S, A_Z)$ in G_S;
Define S_{sp} as the set of all vertices on the shortest paths in A_Z;
Define $S' = S \cup S_{sp}$;
Define $G_H = (S', A_H, c)$;
Compute Kruskal's $Z' = (S', A_{Z'})$;
Separate single-period Z' into multi-period Z'' by greedy algorithm.

5. Computational Results

We will now computationally evaluate the performance of our proposed algorithm. Our computational experiments have been performed on two different sets of instances:

- The "PUCNU"[1] dataset (Fischetti et al., 2017), whose instances are based on the PUC series (Rosseti et al., 2003) for the classical STP. These instances were designed for a single time period and, for the purpose of testing our model, they were transformed into multi-period instances with a number of periods equal to 2, 3, 5 or 8.

- The incomplete graph instances "IG instances"[2], which we have generated by randomly selecting points in a defined Cartesian plan. These are instances with incomplete graphs, that is, edges are defined for only a subset of all vertex pairs with a cost equal to the Euclidean distance between the vertices. Note that the graphs are incomplete, but connected. Multi-period instances are created with a number of periods equal to 2, 3, 5, 8, 10 or 15.

Computational experiments were carried out on a Linux Mint 18.3 Cinnamon 64-bit operating system, version 3.6.7, using a single CPU with 3.40 GHz Intel processor and 16 GB of RAM. All runs had a time limit of 1 hour. The algorithm is written in the Java programming language. The commercial packages ILOG CPLEX and ILOG Concert Technology, version 12.7.1 (IBM, 2017) were used to solve the ILP. The cut constraints (5) are not part of the initial formulation, but dynamically added. Integer infeasible points are cut off by means of a **LazyConstraintCallback**, executing our separation algorithm, detailed in Section 4.1.1. Fractional infeasible points are cut off by means of a **UserCutCallback**, executing the separation procedure explained in Section 4.1.2. It is important to note that the **UserCutCallback** is used within the cut loop that CPLEX calls at each node of the branch-and-cut algorithm, once CPLEX has ended its own cut generation. After that, CPLEX calls a **HeuristicCallback**, executing our primal heuristic, as described in Section 4.2. The algorithm terminates after proving optimality or after reaching the given time limit.

5.1. Distance and budget limits

It is important to mention that all sets of instances make use of the artificial root constraints (15), (16) and (17) and compel us to define the distance and budget limits in a way that challenges the algorithm to solve the problem. To define a tight distance limit per period, we use the average distance of the edges added throughout the entire planning horizon, refered to as $\overline{d}$, divided by the number of periods in the planning horizon $|T|$, given in equation (18):

$$distanceLimit^t = \frac{\overline{d}}{|T|},\ \forall t \in T. \tag{18}$$

Equation (19) defines the total average distance $\overline{d}$ as the product between the average distance D_a of building an edge and the number of terminals T_n that would maximize the number of ways to combine k terminals from a set of nT terminals.

[1] Publicly available at http://dimacs11.zib.de/instances/PCSPG-PUCNU.zip .
[2] Publicly available at https://github.com/larissaftf/IG-instances .

$$\overline{d} = D_a \times T_n \tag{19}$$

In order to define a tight distance limit, we consider the k-combination of a set of terminals T_L, which is defined as a subset of k distinct elements of T_L. If the set has nT elements, the number of k-combinations is equal to the binomial coefficient, as in equation (20):

$$\binom{nT}{k} = \frac{nT!}{k!(nT-k)!}. \tag{20}$$

The value of k that maximizes the number of ways to combine k terminals from a set of nT terminals is at $nT/2$. Hence, the distance limit for each time period is calculated as the number of terminals (nT) times the average distance D_a divided by 2 times the number of total periods, as in equation (21):

$$distanceLimit^t = \frac{nT \times D_a}{(2 \times |T|)}, \ \forall t \in T. \tag{21}$$

A tight budget limit is defined in a similar way, but considering the number of time periods in the subset for which it is defined, i.e., $|\hat{T}_B|$. C_a is the average cost of building an edge. The percentage rate p_r has the purpose of making the budget limit even tighter, as shown in equation (22):

$$budgetLimit^{T_B} = \frac{nT \times C_a \times p_r \times |\hat{T}_B|}{(2 \times |T|)}, \ \forall T_B \in \hat{T}_B. \tag{22}$$

5.2. Modified PUCNU instances

The test instances known as "PUC series" were introduced by Rosseti et al. (2003) with the purpose of evaluating and comparing existing and newly developed algorithms for the Steiner problem in graphs. Such instances are not amenable to reductions proposed by preprocessing techniques. They also presented large integrality gaps between the optimal integer solution and that of the LP relaxation. Moreover, they are prone to a lot of symmetry, which made them difficult to solve to optimality for both exact methods and heuristics.

The PUCNU instances presented in Fischetti et al. (2017) have been generated based on the "PUC series" instances. They contain incomplete graphs with edge costs equal to 1 and vertex profits varying from 0 to 2. It is important to note that a vertex with profit 0 corresponds to a Steiner node and a vertex with profit 1 or 2 corresponds to a terminal node. The edge costs are the same for all time periods and the profit accumulates over time: if a vertex is selected at a certain time period, its profit is accounted for that period and for the following periods. In these instances, costs and profits are close to each other. This allows for many feasible solutions of high quality but make it difficult to prove optimality. Given that the PUCNU instances were designed for a single time period, we transform them into multi-period instances to test our algorithm.

5.2.1. Instance characteristics

Table 1 reports the characteristics of the PUCNU dataset. For each instance, it summarizes the instance name, the number of vertices, the number of edges and the number of terminal nodes ("nT").

Table 1: Modified PUCNU instances

name	$\|V\|$	$\|E\|$	**nT**
bip42nu	1200	3982	200
bip52nu	2200	7997	200
bip62nu	1200	10002	200
bipa2nu	3300	18073	300
bipe2nu	550	5013	50
cc10-2nu	1024	5120	135
cc11-2nu	2048	11263	244
cc12-2nu	4096	24574	473
cc3-10nu	1000	13500	50
cc3-11nu	1331	19965	61
cc3-12nu	1728	28512	74
cc3-4nu	64	288	8
cc3-5nu	125	750	13
cc5-3nu	243	1215	27
cc6-2nu	64	192	12
cc6-3nu	729	4368	76
cc7-3nu	2187	15308	222
cc9-2nu	512	2304	64

5.2.2. Results

We here show the results for instances with 8 time periods based on the PUCNU instances, with two different settings: one with one budget limit for the entire time horizon (see Table 2) and another with a separate budget limit for time periods 1 to 4 and another budget limit for time periods 5 to 8 (see Table 3).

Tables 2 and 3 show the best lower ("LB") and upper bound ("UB") found, the final optimality gap, the number of nodes explored in the branch-and-bound tree, the number of total cuts added by the model and the computing time. A time limit of one hour was used for all experiments. However, if the time limit has been exceeded while a callback was running, the callback has been finished before stopping the algorithm. Therefore, some time markers may have values greater than 3600 seconds.

Table 2: Modified PUCNU instances with 8 time periods — one budget limit for entire planning horizon

name	**LB**	**UB**	**gap(%)**	**# nodes**	**# cuts**	**time(s)**
bip42nu	1813.01	1831.00	0.98	34	1764	3806.37
bip52nu	1765.67	1781.00	0.86	4	1523	3650.91
bip62nu	1717.40	1733.00	0.90	1	781	3618.07
bipa2nu	-	-	-	-	-	Memout

Continued on next page

Table 2 – *Continued from previous page*

name	LB	UB	gap(%)	# nodes	# cuts	time(s)
bipe2nu	373.37	385.00	3.02	20	305	3605.16
cc10-2nu	1206.26	1285.00	6.13	4	2833	3612.95
cc11-2nu	2249.05	2354.00	4.46	1	606	3645.99
cc12-2nu	-	-	-	-	-	Memout
cc3-10nu	367.81	417.00	11.80	6	1504	3614.96
cc3-11nu	573.67	618.00	7.17	3	827	3624.34
cc3-12nu	685.50	746.00	8.11	1	361	3639.49
cc3-4nu	51.99	65.00	20.01	21	1097	3671.80
cc3-5nu	111.27	118.00	5.70	113	2180	3705.38
cc5-3nu	232.20	257.00	9.65	16	2331	3894.27
cc6-2nu	88.59	99.00	10.51	21	1022	3707.83
cc6-3nu	684.50	730.00	6.23	7	2419	3607.58
cc7-3nu	2059.48	2178.00	5.44	1	1053	3653.51
cc9-2nu	580.28	603.00	3.77	14	5506	3603.97

Table 3: Modified PUCNU instances with 8 time periods — One budget limit for periods 1 to 4 and another for periods 5 to 8

name	LB	UB	gap(%)	# nodes	# cuts	time(s)
bip42nu	1879.99	1884.00	0.21	99	3101	3630.98
bip52nu	1833.40	1850.00	0.90	2	810	3651.62
bip62nu	1790.91	1822.00	1.71	3	820	3605.00
bipa2nu	-	-	-	-	-	Memout
bipe2nu	390.73	402.00	2.80	7	738	3605.25
cc10-2nu	1245.15	1323.00	5.88	6	4091	3612.79
cc11-2nu	2324.57	2423.00	4.06	0	910	3646.51
cc12-2nu	-	-	-	-	-	Memout
cc3-10nu	381.10	434.00	12.19	5	1324	3615.03
cc3-11nu	588.27	640.00	8.08	3	859	3623.98
cc3-12nu	711.00	744.00	4.44	2	608	3639.60
cc3-4nu	55.37	64.00	13.48	23	1042	3686.08
cc3-5nu	115.54	122.00	5.29	53	1462	3614.99
cc5-3nu	239.53	262.00	8.58	25	2508	3654.69
cc6-2nu	92.03	102.00	9.78	29	1259	3603.03
cc6-3nu	708.01	758.00	6.59	10	3049	3607.21
cc7-3nu	2124.20	2244.00	5.34	1	1461	3653.22
cc9-2nu	592.74	615.00	3.62	9	3364	3754.10

The large instances (**cc12-2nu** and **bipa2nu**) have not been solved due to a lack of memory, i.e., CPLEX reached its memory limit and could not build the model. Instance

cc11-2nu is solved at the root node of the branch-and-bound tree (# **nodes** equals 0). The gaps observed for both settings of budget limits are similar in magnitude, no matter whether the budget limit is defined for the entire planning horizon or for the sets of periods (periods 1 to 4, and 5 to 8).

To analyze the impact of the different side constraints, we can compare the structure of the best integer feasible solutions found (not necessarily optimal) for each setting. This information is shown in Table 4 for the case where there is one budget limit for the full planning horizon and for the case with one budget limit for time periods 1 to 4 and another one for time periods 5 to 8. The table shows the number of terminals that are present in the integer feasible solution found, the budget limit value and distance limit value calculated for those instances, the total revenue obtained by the solutions and the total amount spent. It can be seen that a budget limit for the full planning horizon allows the terminal vertices to enter the network at an earlier time period than they would for a budget limit per subset of time periods. That conclusion is reached due to the value of the total revenue obtained for the different integer feasible solutions. Because of a rounding procedure at the calculation of the budget limit, the budget limit may be one unit larger for the case where there is one budget limit for a subset of time periods. Hence, extra terminals may be able to be selected. However, the integer feasible solution found has a lower net worth than the net worth for the case where there is only one budget limit per time horizon. This illustrates that, while budget constraints on subsets of time periods are important to model real world applications, such restrictions may lead to significantly less revenue in practice.

Table 4: Modified PUCNU instances with 8 time periods — Solution structure

				One total budget		**Two budgets**	
name	**termInSol**	**budgetLimit**	**distLimit**	**totalRev**	**totalSpent**	**totalRev**	**totalSpent**
bip42nu	61	73	13	690	73	638	74
bip52nu	62	73	13	708	73	640	74
bip62nu	67	73	13	756	73	668	74
bipa2nu	1	107	19	16	0	16	0
bipe2nu	21	23	4	238	23	222	24
cc10-2nu	34	51	9	366	51	323	46
cc11-2nu	63	90	16	688	90	619	90
cc12-2nu	1	168	30	16	0	16	0
cc3-10nu	13	23	4	150	23	134	24
cc3-11nu	14	23	4	164	22	144	24
cc3-12nu	17	28	5	194	28	196	28
cc3-4nu	3	6	1	42	3	46	6
cc3-5nu	5	6	1	56	6	52	6
cc5-3nu	6	12	2	74	11	66	8
cc6-2nu	4	6	1	42	5	39	5
cc6-3nu	21	28	5	210	28	182	28
cc7-3nu	55	79	14	597	79	532	80

Continued on next page

Table 4 – *Continued from previous page*

name	termInSol	budgetLimit	distLimit	totalRev	totalSpent	totalRev	totalSpent
cc9-2nu	15	23	4	164	23	153	24

Comparison with formulation from Suhl and Hilbert (1998)

We re-implemented Suhl and Hilbert (1998)'s model to be able to compare our results to theirs. Tables 5 and 6 show the computational results for instances with 5 time periods, where each time period has its individual budget. As our problem is a minimization problem and Suhl and Hilbert (1998)'s is a maximization problem, lower bounds and upper bounds differ in value and in meaning. For a minimization problem, the upper bound is the best integer feasible solution found whereas the lower bound is the relaxed solution. For a maximization problem, the upper bound is the relaxed solution and the lower bound is the incumbent solution. To facilitate the comparison between the models, we have transformed the lower and upper bounds of Suhl and Hilbert (1998)'s model as if their objective function was of the *Goemans and Williamson Minimization* Problem. It can be seen that Suhl and Hilbert (1998)'s model may have difficulties finding good integer feasible solutions. Our model outperforms Suhl and Hilbert (1998)'s for all instances, except for **cc3-5nu**, **cc5-3nu** and **cc6-2nu**.

Table 5: Modified PUCNU instances — Number of periods: 5 — One budget limit per time period — MPCSTB

name	LB	UB	gap(%)	# nodes	# cuts	time(s)
bip42nu	1237.60	1240.00	0.19	20	1900	3741.92
bip52nu	1209.43	1217.00	0.62	34	1665	3662.16
bip62nu	1186.12	1200.00	1.16	10	1203	3611.56
bipa2nu	1726.99	1763.00	2.04	0	0	3681.66
bipe2nu	276.34	282.00	2.01	207	1617	3630.32
cc10-2nu	815.89	849.00	3.90	5	3464	3608.50
cc11-2nu	1509.45	1567.00	3.67	2	2029	3630.05
cc12-2nu	-	-	-	-	-	Memout
cc3-10nu	264.39	284.00	6.91	9	1888	3609.96
cc3-11nu	377.78	410.00	7.86	6	1336	3615.97
cc3-12nu	455.00	480.00	5.21	5	1014	3625.99
cc3-4nu	44.00	44.00	0.00	27	862	3352.47
cc3-5nu	63.48	70.00	9.32	31	1830	3627.00
cc5-3nu	148.45	159.00	6.63	25	3242	3601.30
cc6-2nu	51.67	58.00	10.91	30	1260	3651.13
cc6-3nu	454.00	486.00	6.58	9	3173	3605.35
cc7-3nu	1359.74	1427.00	4.71	2	1225	3635.84
cc9-2nu	380.46	392.00	2.94	6	2351	3603.07

Table 6: Modified PUCNU instances — Number of periods: 5 — One budget limit per time period — SUHL AND HILBERT

name	LB	UB	gap(%)	# nodes	# cuts	time(s)
bip42nu	1205.63	1252.00	3.70	5501	6	3603.57
bip52nu	1182.78	1232.00	4.00	718	0	3600.47
bip62nu	1167.23	1212.00	3.69	398	3	3635.43
bipa2nu	1693.39	2180.00	22.32	0	1	3772.37
bipe2nu	269.93	285.00	5.29	1674	1	3601.10
cc10-2nu	778.85	851.00	8.48	23918	65	3600.37
cc11-2nu	1443.74	1706.00	15.37	6300	34	3603.86
cc12-2nu	-	-	-	-	-	Memout
cc3-10nu	254.29	287.00	11.40	6436	45	3600.16
cc3-11nu	361.43	434.00	16.72	2100	12	3612.27
cc3-12nu	436.81	476.00	8.23	801	3	3816.92
cc3-4nu	44.00	44.00	0.00	221	0	1.55
cc3-5nu	69.00	69.00	0.00	9731	0	61.88
cc5-3nu	148.57	157.00	5.37	200901	44	3600.86
cc6-2nu	55.00	55.00	0.00	4320	0	10.96
cc6-3nu	438.60	481.00	8.81	40801	86	3602.24
cc7-3nu	1306.80	-	-	6679	51	3600.35
cc9-2nu	366.92	392.00	6.40	98784	26	3600.10

Visualization of a solution

Figure 5 visualizes the best integer feasible solution found by our model for the 5-periods instance **cc6-2nu**. The figure indicates the nodes connected by a thick solid line in the first time period, by a dashed line in the second time period, by a dotted line in the third time period, by a dash-dot line in the fourth time period and finally by a thin solid line in the fifth time period. The lower bound, upper bound, optimality gap, the number of nodes and cuts, and the required computing time are as reported in Table 5.

5.3. Randomly generated instances of incomplete graphs

The randomly generated instances are inspired by real-world Brazilian gas network expansion instances, which could not be directly used in this paper due to confidentiality agreements. We will use these instances to test the scalability of our proposed approach. The so-called "IG instances" define 95% of the nodes as Steiner nodes (with zero profit) and 5% of nodes have random fractional value profits. We first randomly selected the edges that should be included in the graph. Then, to ensure that the graph is connected, we apply a connected components algorithm, guaranteeing that all nodes in the graph are in the same connected component. Moreover, we chose the generated instances of incomplete graphs with respective densities as low as possible, without losing connectivity. The cost of the selected edges are fractional and randomly drawn from a uniform distribution, proportional to

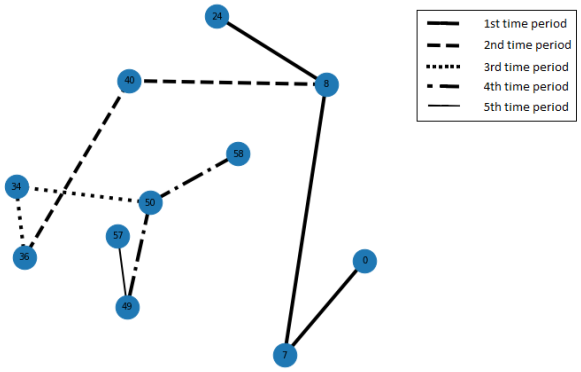

Figure 5: Integer feasible solution found for instance **cc6-2nu** for a 5-periods run.

the Euclidean distance of the vertices, drawn randomly from a Cartesian plan. Since these instances are made of incomplete graphs, they have fewer variables and fewer constraints than complete graph instances. At first glance, incomplete graph instances seem to be easier to solve due to the smaller number of variables. However, the opposite may be the case, as the path to a profitable node may include several zero revenue ones along the way.

5.3.1. Instance characteristics

We have generated instances with different numbers of vertices $|V| \in \{50, 100, 150, 200, 250, 300\}$. For each $|V|$, we randomly selected 5 instances. Table 7 reports the instance characteristics of the IG dataset: the instance name, the number of vertices, the number of edges, the number of terminals ("nT") and the percentage of edges that have been retained in the instance from the complete graph.

Table 7: IG instance characteristics

name	$\|V\|$	$\|E\|$	**nT**	**perc(%)**
50_1	50	117	3	9.55
50_2	50	106	3	8.65
50_3	50	137	3	11.18
50_4	50	139	3	11.35
50_5	50	120	3	9.80
100_1	100	245	5	4.95
100_2	100	260	5	5.25
100_3	100	304	5	6.14
100_4	100	261	5	5.27
100_5	100	248	5	5.01
150_1	150	413	7	3.70
150_2	150	409	7	3.66
150_3	150	428	7	3.83
150_4	150	423	7	3.79
150_5	150	435	7	3.89

Continued on next page

Table 7 – *Continued from previous page*

name	$\|V\|$	$\|E\|$	**nT**	**perc(%)**
200_1	200	565	10	2.84
200_2	200	559	10	2.81
200_3	200	552	10	2.77
200_4	200	541	10	2.72
200_5	200	544	10	2.73
250_1	250	837	12	2.69
250_2	250	826	12	2.65
250_3	250	850	12	2.73
250_4	250	828	12	2.66
250_5	250	823	12	2.64
300_1	300	1215	15	2.71
300_2	300	1359	15	3.03
300_3	300	1234	15	2.75
300_4	300	1277	15	2.85
300_5	300	1275	15	2.84

5.3.2. Results

As before, we will compare between two different settings of budget limits. In all following experiments, instances have two time periods. Table 8 shows the average results for the setting of one single budget limit over the 2 time periods and Table 9 summarizes the results for the setting of one budget limit for each of the 2 time periods.

Table 8: Average results for IG instances with 2 time periods — One budget limit over both periods

$\|V\|$	$\overline{\textbf{LB}}$	$\overline{\textbf{UB}}$	$\overline{\textbf{gap}}(\%)$	$\overline{\textbf{\# nodes}}$	$\overline{\textbf{\# cuts}}$	$\overline{\textbf{time}}(s)$
50	12.19	12.19	0.00	9	74	11.56
100	17.08	17.08	0.00	46	869	302.79
150	21.05	22.59	6.07	138	3273	2395.95
200	27.98	29.24	4.27	150	4869	2204.92
250	32.41	35.02	7.47	248	6977	3735.59
300	36.57	45.96	20.47	67	9207	3698.31

Table 9: Average results for IG instances with 2 time periods — One budget limit for each time period

$\|V\|$	$\overline{\textbf{LB}}$	$\overline{\textbf{UB}}$	$\overline{\textbf{gap}}(\%)$	$\overline{\textbf{\# nodes}}$	$\overline{\textbf{\# cuts}}$	$\overline{\textbf{time}}(s)$
50	12.19	12.19	0.00	8	75	6.49
100	17.61	17,61	0.00	53	995	510.38

Continued on next page

Table 9 – *Continued from previous page*

$\|V\|$	$\overline{\textbf{LB}}$	$\overline{\textbf{UB}}$	$\overline{\textbf{gap}}(\%)$	$\overline{\textbf{\# nodes}}$	$\overline{\textbf{\# cuts}}$	$\overline{\textbf{time}}(s)$
150	22.01	22.57	2.19	153	2965	2386.11
200	29.10	29.55	1.47	142	4390	2416.38
250	31.20	35.45	11.66	161	6918	3152.42
300	36.30	46.67	22.17	65	6959	3626.73

We now analyze the structure of the best found feasible integer solution. Table 10 indicates the average number of terminals that were able to enter the network, the budget and distance limit values, the total revenue and the total amount spent for that run. A single budget limit for the entire horizon tends to increase the total revenue when compared to a budget limit for each time period, because the nodes are allowed to enter the network at an earlier time period.

Table 10: Average results for IG instances with 2 time periods — Solution structure

				One total budget		**Two budgets**	
$\|V\|$	**termInSol**	**budgetLimit**	**distLimit**	**totalRev**	**totalSpent**	**totalRev**	**totalSpent**
50	2	2.00	1.60	10.90	0.27	10.90	0.27
100	2	2.00	2.00	11.03	0.66	10.50	0.66
150	3	2.00	2.00	10.18	0.65	10.18	0.63
200	3	2.00	2.00	11.53	1.04	11.13	0.95
250	4	2.00	2.00	12.53	0.85	12.14	0.89
300	5	3.00	3.00	14.90	2.14	15.43	2.07

Comparison with formulation from Suhl and Hilbert (1998)

Next, we compare our model results with the one of Suhl and Hilbert (1998)'s. Table 11 presents the results the latter obtains using one budget limit per time period. Comparing their results to ours (see Table 9 for this particular setting), it seems that both models achieve similar performance on average for the IG dataset.

Table 11: Average results for IG instances with 2 time periods — One budget limit for each time period — SUHL AND HILBERT

$\|V\|$	$\overline{\textbf{LB}}$	$\overline{\textbf{UB}}$	$\overline{\textbf{gap}}(\%)$	$\overline{\textbf{\# nodes}}$	$\overline{\textbf{\# cuts}}$	$\overline{\textbf{time}}(s)$
50	12.19	12.19	0.00	59	2	0.17
100	17.61	17.61	0.00	8061	28	7.85
150	22.57	22.57	0.00	214702	168	420.04
200	29.55	29.55	0.00	369445	198	971.94

Continued on next page

Table 11 – *Continued from previous page*

$\|V\|$	**LB**	**UB**	**gap**(%)	**# nodes**	**# cuts**	**time**(s)
250	31.10	34.67	10.26	1007466	416	3600.16
300	31.69	43.95	27.98	1062307	940	3600.15

Visualization of a solution

Figure 6 gives an illustrative example of the optimal solution found by our model for a 5-periods run of instance **100_4**.

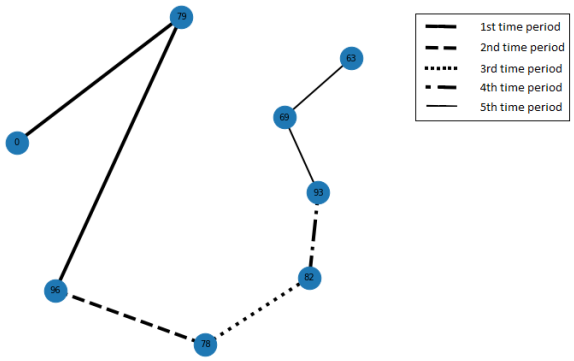

Figure 6: Optimal solution found for instance **100_4** for a 5-periods run.

Impact of number of time periods

As a final analysis, we now explore how the number of time periods impacts the difficulty of solving the problem. Table 12 presents results for instance **100_2** with different numbers of time periods ranging from 2 to 15. We observe that the difficulty of solving the problem does not grow exponentially with the increase in the number of periods. We therefore expect that the problem can be reasonably well solved even if the number of time periods is large.

Table 12: IG instances — Instance 100_2

# per	**LB**	**UB**	**gap(%)**	**# nodes**	**# cuts**	**time(s)**
2	20.04	20.04	0.00	71	1059	326.21
3	33.18	33.18	0.00	55	797	244.60
5	51.09	51.09	0.00	100	1623	762.57
8	75.96	75.96	0.00	234	3666	1052.68
10	88.49	88.49	0.00	264	3551	987.62
15	116.23	116.24	0.01	542	7142	2094.73

6. Conclusions

The Multi-period Prize-collecting Steiner Tree problem with Budget constraints is a generalization of the classical Prize-collecting Steiner Tree problem. Customers are selected and connected by a network of minimum cost, along different time periods and respecting a predefined budget and a predefined distance limit. Therefore, the problem involves planning the expansion of a network throughout multiple time periods, respecting the distance limit given per period and the budget limit given per subset of periods. The objective is to maximize the sum of the profits of the incorporated vertices reduced by the cost of the new edges.

The objective of this paper is to provide an algorithm that is capable of finding solutions of high-quality to realistically sized problems in reasonable computing times. We propose a branch-and-cut approach where the connectivity cut constraints are dynamically generated only when they are violated. We use two separation procedures and a primal heuristic that are vital for the performance of the algorithm. Benchmark instances from the literature, adapted to a multi-period setting, with up to 3300 vertices and 300 terminals, are satisfactorily solved with our approach. Randomly generated incomplete graph instances with up to 300 vertices, where approximately 15 are terminals, are also satisfactorily solved. We note that these problems are generally hard to solve. Using a 60 minutes time limit, some of the instances still presented large optimality gaps. Given the strategic nature of the problem, we suspect that the allowance of more computing time will further help to close those gaps. Our approach was compared to Suhl and Hilbert (1998)'s model and outperformed this benchmark on the first data set. On the second data set, both models performed similarly well.

Possible future work includes (a) the use of different budget and distance limits for subset-sets of time periods; and (b) introducing stochasticity to the model, given that several parameters (such as profits) may not be subject to uncertainty.

Acknowledgements

The authors gratefully acknowledge the support of the Coordination for the Improvement of Higher Education Personnel (CAPES) - Finance Code 001. The work of the third author was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada under grant 2017-05224.

References

Arulselvan, A., Bley, A., Gollowitzer, S., Ljubić, I., and Maurer, O. (2011). MIP modeling of incremental connected facility location. In *Network Optimization*, pages 490–502. Springer.

Borraz-Sánchez, C., Bent, R., Backhaus, S., Hijazi, H., and Van Hentenryck, P. (2016). Convex relaxations for gas expansion planning. *INFORMS Journal on Computing*, 28(4):645–656.

Canuto, S. A., Resende, M. G., and Ribeiro, C. C. (2001). Local search with perturbations for the prize-collecting Steiner tree problem in graphs. *Networks: An International Journal*, 38(1):50–58.

Cherkassky, B. V. and Goldberg, A. V. (1995). On implementing push-relabel method for the maximum flow problem. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 157–171. Springer.

Chopra, S. and Rao, M. R. (1994a). The Steiner tree problem I: Formulations, compositions and extension of facets. *Mathematical Programming*, 64(1-3):209–229.

Chopra, S. and Rao, M. R. (1994b). The Steiner tree problem II: Properties and classes of facets. *Mathematical Programming*, 64(1-3):231–246.

Conforti, M., Cornuéjols, G., and Zambelli, G. (2014). *Integer Programming*. Springer.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms*. MIT press.

Costa, A. M., Cordeau, J.-F., and Laporte, G. (2006). Steiner tree problems with profits. *INFOR: Information Systems and Operational Research*, 44(2):99–115.

Costa, A. M., Cordeau, J.-F., and Laporte, G. (2009). Models and branch-and-cut algorithms for the Steiner tree problem with revenues, budget and hop constraints. *Networks: An International Journal*, 53(2):141–159.

Da Cunha, A. S., Lucena, A., Maculan, N., and Resende, M. G. (2009). A relax-and-cut algorithm for the prize-collecting Steiner problem in graphs. *Discrete Applied Mathematics*, 157(6):1198–1217.

Dantzig, G., Fulkerson, R., and Johnson, S. (1954). Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society of America*, 2(4):393–410.

Feofiloff, P., Fernandes, C. G., Ferreira, C. E., and de Pina, J. C. (2007). Primal-dual approximation algorithms for the prize-collecting Steiner tree problem. *Information Processing Letters*, 103(5):195–202.

Feremans, C., Labbé, M., and Laporte, G. (2002). A comparative analysis of several formulations for the generalized minimum spanning tree problem. *Networks: An International Journal*, 39(1):29–34.

Fischetti, M. (1991). Facets of two Steiner arborescence polyhedra. *Mathematical Programming*, 51(1-3):401–419.

Fischetti, M., Leitner, M., Ljubić, I., Luipersbeck, M., Monaci, M., Resch, M., Salvagnin, D., and Sinnl, M. (2017). Thinning out Steiner trees: a node-based model for uniform edge costs. *Mathematical Programming Computation*, 9(2):203–229.

Gamrath, G., Koch, T., Maher, S. J., Rehfeldt, D., and Shinano, Y. (2017). SCIP-Jack- a solver for STP and variants with parallelization extensions. *Mathematical Programming Computation*, 9(2):231–296.

Goemans, M. X. and Myung, Y.-S. (1993). A catalog of Steiner tree formulations. *Networks*, 23(1):19–28.

Goemans, M. X. and Williamson, D. P. (1997). The primal-dual method for approximation algorithms and its application to network design problems. *Approximation algorithms for NP-hard problems*, pages 144–191.

Gollowitzer, S. and Ljubić, I. (2011). MIP models for connected facility location: A theoretical and computational study. *Computers & Operations Research*, 38(2):435–449.

Humpola, J. and Fügenschuh, A. (2015). Convex reformulations for solving a nonlinear network design problem. *Computational Optimization and Applications*, 62(3):717–759.

IBM (2017). ILOG CPLEX 12.7.1. *CPLEX Users Manual*.

Johnson, D. S., Minkoff, M., and Phillips, S. (2000). The prize collecting Steiner tree problem: theory and practice. In *SODA*, volume 10, page 4. Citeseer.

Karp, R. M. (1972). Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Springer.

Klau, G. W., Ljubić, I., Moser, A., Mutzel, P., Neuner, P., Pferschy, U., Raidl, G., and Weiskircher, R. (2004). Combining a memetic algorithm with integer programming to solve the prize-collecting Steiner tree problem. In *Genetic and Evolutionary Computation Conference*, pages 1304–1315. Springer.

Ljubić, I., Weiskircher, R., Pferschy, U., Klau, G. W., Mutzel, P., and Fischetti, M. (2005). Solving the prize-collecting Steiner tree problem to optimality. In *ALENEX/ANALCO*, pages 68–76.

Ljubić, I., Weiskircher, R., Pferschy, U., Klau, G. W., Mutzel, P., and Fischetti, M. (2006). An algorithmic framework for the exact solution of the prize-collecting Steiner tree problem. *Mathematical Programming*, 105(2-3):427–449.

Lucena, A. and Beasley, J. (1998). A branch and cut algorithm for the Steiner problem in graphs. *Networks: An International Journal*, 31(1):39–59.

Lucena, A. and Resende, M. G. (2004). Strong lower bounds for the prize collecting Steiner problem in graphs. *Discrete Applied Mathematics*, 141(1-3):277–294.

Magnanti, T. L. and Raghavan, S. (2005). Strong formulations for network design problems with connectivity requirements. *Networks: An International Journal*, 45(2):61–79.

Magnanti, T. L. and Wolsey, L. A. (1995). Optimal trees. *Handbooks in Operations Research and Management Science*, 7:503–615.

Poss, M. (2012). Models and algorithms for network design problems. *4OR*, 10(2):215–216.

Rosseti, I., De Aragão, M. P., Ribeiro, C. C., Uchoa, E., and Werneck, R. F. (2003). New benchmark instances for the Steiner problem in graphs. In *Metaheuristics: Computer Decision-Making*, pages 601–614. Springer.

Suhl, U. H. (1994). MOPS–mathematical optimization system. *European Journal of Operational Research*, 72(2):312–322.

Suhl, U. H. and Hilbert, H. (1998). A branch-and-cut algorithm for solving generalized multiperiod Steiner problems in graphs. *Networks: An International Journal*, 31(4):273–282.

Uchoa, E. (2006). Reduction tests for the prize-collecting Steiner problem. *Operations Research Letters*, 34(4):437–444.