On optimally solving sub-tree scheduling for wireless sensor networks with partial coverage: a branch-and-cut algorithm

Nicola Bianchessi[a,*]

[a]*Dipartimento di Informatica, Università degli Studi di Milano, Via Celoria 18, I-20133 Milan, Italy.*

Abstract

Given a wireless sensor network, we consider the problem to minimize its total energy consumption over consecutive time slots with respect to communication activities. Non-empty and disjoint subsets of nodes are required to be active and connected under a tree topology configuration in the different time slots, and each network node must be active in a unique time slot. Moreover, the power required by the same pair of network nodes to communicate on the associated direct channel may vary in the different time slots. The problem has been recently introduced in the literature under the name Sub-Tree Scheduling for Wireless Sensor Networks with Partial Coverage. We focus on the exact solution of the problem. We present a branch-and-cut (BC) algorithm based on a novel Integer Linear Programming formulation which allows avoiding the introduction of symmetries in the solution space. In particular, the algorithm relies on an efficient and non-typical separation algorithm for known valid inequalities, and on an easy-to-implement primal bound heuristic. The effectiveness of the BC algorithm is empirically shown through an extensive experimental analysis involving 300 newly generated benchmark instances with up to 200 network nodes and 8 time slots. Moreover, the experimental results show that the BC algorithm represents a valid computational tool to benchmark the performance of heuristics addressing the problem, and can be used in practice, as an heuristic solver, to tackle problem instances that are not too large.

Keywords: Wireless sensor network, Consecutive time slots, Sub-tree scheduling, Partial coverage, Energy consumption minimization, Branch-and-cut algorithm

1. Introduction

A wireless sensor network (WSN) consists of sensor devices deployed across a geographic area to sense the environment by measuring physical parameters such as temperature, motion, etc. Energy consumption is among the major issues concerning the design/management of WSNs (Akyildiz *et al.*, 2002; Yardibi and Karasan, 2010). In fact, WNSs are usually deployed in remote areas where the replacement of batteries in sensors is difficult, if not impossible. Thus, by reducing energy consumption in WNSs it is possible to increase their lifetime.

Each sensor node in a WSN performs three basic activities: sensing, computation (processing), and communication. It is usually assumed that in each network node energy is primarily consumed due to communication activities (sending/receiving data), and that the energy to perform sensing and computation is negligible compared to the former (Raghunathan *et al.*, 2002; Rost and Fettweis, 2010; Bulut and Korpeoglu, 2011).

Kulshrestha and Mishra (2021) classify the approaches aimed at reducing energy consumption in WSNs into two groups: *energy efficiency* (Anastasi *et al.*, 2009) and *energy balancing* (Ishmanov *et al.*, 2011) approaches. In the taxonomy proposed, the approaches belonging to the subclass *energy balancing/duty cycling/sleep scheduling* are based on the idea that network nodes may be alternatively put in active and sleep mode for a certain time interval, so as to reduce their energy consumptions in sleep mode (mode in which sensing and communication activities are interrupted). To this subclass of approaches belongs those discussed in (Wang *et al.*, 2017; Bidoki *et al.*, 2018; Adasme, 2019; Hawbani *et al.*, 2019; Kumar and Kim, 2019; Natarajan and Subramanian, 2019; Guruprakash *et al.*, 2020; Shallahuddin *et al.*, 2020; Zhang *et al.*,

*Corresponding author.
Email address: `bianchessi@di.unimi.it` (Nicola Bianchessi)

2020). We focus on the the optimization problem introduced by Adasme (2019) under the name Sub-Tree Scheduling for Wireless Sensor Networks with Partial Coverage (STSWSN-PC).

The STSWSN-PC seeks for the minimization of the total energy consumption of a WSN over consecutive time slots with respect to communication activities (e.g., of the total connectivity power costs associated with the use of a WSN over consecutive time slots). Requirements defining the problem are that non-empty and disjoint subsets of nodes have to be active and connected under a tree topology configuration in the different time slots (while the remaining nodes are temporary put into sleep mode), and that each network node must be active in a unique time slot. The operating scenario/policy assumed is the following. Sensors are deployed in the geographic area that has to be monitored. The deployment of the sensors is defined a priori according to the specific practical application at hand. Given a planning horizon $T = \{1, \ldots, |T|\}$ of $|T|$ time slots, the nodes that are active in time slot $t \in T$ perform sensing and communication activities, while the remaining nodes are put in sleep mode to save energy. In an additional round at time slot $|T| + 1$, all nodes become active and may communicate among themselves so that all the necessary information concerning the monitored area can be collected and elaborated. Then, the whole process is repeated iteratively for the entire lifetime of the network. In particular, the operating policy does not guarantee that the coverage level of sensors is, continuously over time, 100% of the geographic area. This is the concept of partial coverage. However, requiring that a non-empty subset of nodes has to be active in each time slot ensures a minimum level of monitoring/coverage of the area over time. Furthermore, note that the operating policy implies to set the duration of the time slots such that the information collected in time slot 1 do not become obsolete before the additional round in time slot $|T| + 1$. (A similar operating policy has been considered also in (Slijepcevic and Potkonjak, 2001) to the aim of maximizing the coverage of the sensed area for each subset of nodes, but without considering the connectivity of the active nodes.) Finally, as for energy consumption, the power required by the same pair of network nodes to communicate on the associated direct channel may vary in the different time slots. Adasme (2019) assumed that connectivity power costs may be estimated with effective prediction methods (taking into account all necessary parameters) for relatively long planning horizons (see, i.e., (Anderson, 2015)), so that power measurements do not become invalid as soon as they are determined. For a more in-depth discussion about the motivations behind the study of the problem, and an extensive description of its technicalities, we refer the interested readers to (Adasme, 2019).

Besides introducing the problem, an additional major contribution of Adasme (2019) consists of presenting two effective meta-heuristics, based on variable neighbour search (Mladenović and Hansen, 1997; Hansen and Mladenović, 2003) and simulating annealing (Kirkpatrick *et al.*, 1983) paradigm, respectively, and making use of the Kruskal's algorithm (Kruskal, 1956). The heuristics are tested on large problem instances, defined by considering up to 400 network nodes and 8 time slots, with a time limit of 2 hours. According to the results obtained, the main finding of the author is that the heuristic based on variable neighbour search has to be preferred when relatively high-quality solutions (w.r.t the total (connectivity) power costs) must be computed quickly. Whereas, preference goes to the heuristic based on simulated annealing when minimizing the total power costs is the major issue. Moreover, in order to optimally solve the STSWSN-PC, and/or provide computational tools to benchmark the performance of the heuristics proposed, the author presented Mixed Integer Quadratic/Linear Programming formulations for the problem, to be solved by means of state-of-the-art commercial solvers. The best results are obtained by solving a Mixed Integer Linear Programming (MILP) formulation by means of the state-of-the-art commercial MILP solver at hand.

In this study we focus on the exact solution of the STSWSN-PC. Our overall/main contribution is the presentation of a branch-and-cut (BC) algorithm which improves the state-of-the-art w.r.t. existing exact algorithms addressing the problem. Particular contributions are:

- We model the problem by means of a novel ILP formulation which allows avoiding the introduction of symmetries in the solution space. Even if the binary variables we consider are polynomial in number, the formulation is defined by an exponential number of constraints. This excludes the possibility to explicitly define the formulation and solve it directly through the MILP solver at hand, and in particular to use the solver as a black-box optimization tool. However, the formulation allows the design of a BC algorithm in which the constraints exponential in number are temporarily ignored at the beginning of the solution process, and then dynamically inserted into the formulation when violated by computed (fractional/integer) solutions.
- The constraints exponential in number in the novel ILP formulation are the generalized version of the subtour elimination constraints (GSECs), introduced by Dantzig *et al.* (1954). By exploiting the specific features of the STSWSN-PC, we devise an efficient and non-typical separation algorithm for the GSECs that relies only on the computation of connected components.
- During the solution process, state-of-the-art commercial MILP solvers usually run internal (primal bound)

heuristics in order to compute, as quickly as possible, tight primal bounds and, consequently, speed up the convergence of the whole solution process. When non-redundant constraints are removed from the formulation, like in our case, the solutions computed by the internal heuristics may be infeasible. In order to make the most of the MILP solver at hand, we design an easy-to-implement primal bound heuristic and embed it into the BC algorithm. The heuristic receives in input an integer solution which is infeasible to the original problem, and makes it feasible in polynomial time. The value of the solution computed may eventually improve the current primal bound and restrict the search space.

Experimental results support the validity of the contributions and are briefly illustrated in the following. We tested the BC algorithm on 300 instances generated as described in (Adasme, 2019), with up to 200 network nodes and 8 time slots. In particular, we allowed only single-thread runs, so as to get deterministic results, and set the time for each run to 1 hour. (i) All the instances with up to 60 (80) network nodes are solved to optimality with a maximum solution time of 44 (510) seconds. All but 2 instances with 100 network nodes are optimally solved. (For the 2 non-solved instances, the percentage optimality gap is about 2%.) (ii) Notably enough, the BC algorithm was always able to compute valid upper (primal) and lower (dual) bounds (even with a time limit of 150 seconds). In Adasme (2019), with a time limit of 2 hours, (i) the largest instances optimally solved consider up to 40 network nodes and 2 time slots, and (ii) feasible solutions are computed only for instances with up 70 network nodes and 4 time slots. Moreover, as discussed in Section 4.2, the experimental results show that the BC algorithm is a valid computational tool to benchmark the performance of heuristics addressing the problem, and can be used in practice, as an heuristic solver, to tackle problem instances that are not too large.

The remainder of the paper is organized as follows. In Section 2, we formally define the STSWSN-PC and provide the mathematical formulation that leads the basis for the branch-and-cut (BC) algorithm, which is discussed in Section 3. Experimental results are presented in Section 4, before final conclusions are drawn in Section 5.

2. Problem definition and formulation

2.1. Problem definition

Let $T = \{1, \dots, |T|\}$ be a set of consecutive time slots of arbitrary duration. The STSWSN-PC can be defined over a sequence of undirected, weighted, complete graphs $G^t = (N, E^t)$. All the graphs share a common node set $N = \{1, \dots, n\}$ representing the nodes of the WSN. Then, each graph G^t is defined by its own set of edges $E = \{(i, j)|i < j, i, j \in N\}$, where edge (i, j) represents the possibility to use a direct communication channel between nodes i and j, in time slot t, at a connectivity power cost p^t_{ij}. A solution s to the problem is defined by subsets of active nodes $S^t \subset N$, $S^t \neq \varnothing$, $t \in T$, such that (i) in each S^t nodes are connected under a tree topology, (ii) $S^{t'} \cap S^{t''} = \varnothing$, $t' \neq t''$, $t', t'' \in T$, and (iii) $\cup_{t \in T} S^t = N$. Given a feasible solution $\bar{s}$ to the problem, let $\bar{E}^t \subseteq E^t$ be the subset of edges selected in the solution w.r.t. time slot $t \in T$, and thus corresponding to the direct communication channels ensuring the connection under a tree topology for the active nodes in S^t, $t \in T$. The aim is to find a solution $\bar{s}$ associated with the minimum total power costs $p(\bar{s}) = \sum_{t \in T} \sum_{(i,j) \in \bar{E}^t} p^t_{ij}$. (It is worth mentioning that the formulation presented in Section 2.2, and the branch-and-cut algorithm presented in Section 3, apply also for sparse graphs $G^t = (N, E^t)$. In fact, as pointed out in (Adasme, 2019), complete graphs may represent sparse ones by simply penalizing the edges not to be considered in optimal solutions.) A sample instance of the problem is illustrated in Figure 1, together with a corresponding optimal solution.

2.2. Mathematical formulation

We model the problem by means of a compact formulation that uses two sets of variables. Binary variables z^t_i, which are equal to 1 if node i is scheduled to be active in time slot $t \in T$, 0 otherwise, and binary variables x^t_{ij}, which model whether a direct communication channel is used between nodes i and j in time slot $t \in T$, $x^t_{ij} = 1$, or not, $x^t_{ij} = 0$. Then, define $E^t(S)$ as the subset of edges of E^t having both endpoints in $S \subseteq N$. The formulation is as follows.

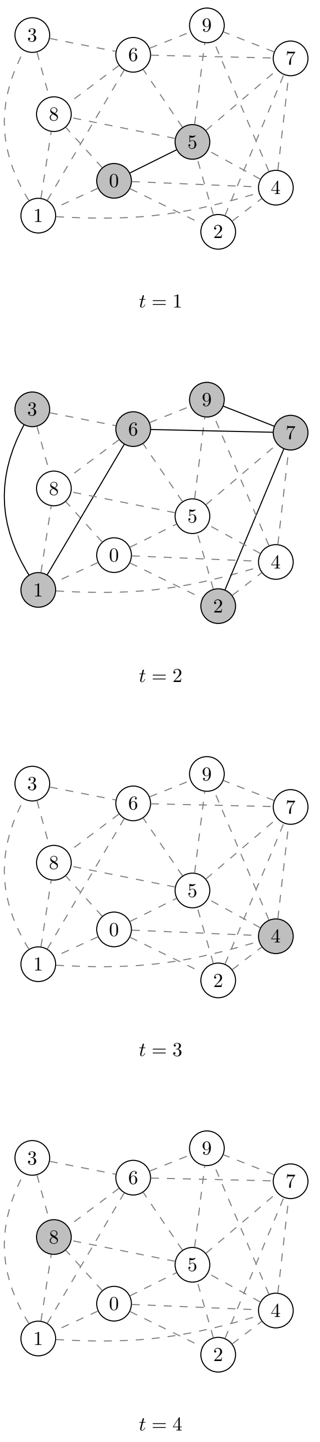
$$\min \sum_{t \in T} \sum_{(i,j) \in E^t} p^t_{ij} x^t_{ij} \tag{1a}$$

$$s.t. \sum_{t \in T} z^t_i = 1 \qquad i \in N \tag{1b}$$

Figure 1: A STSWSN-PC instance for $N = 10$ and $|T| = 4$ (a) and a corresponding optimal solution (b).

		j									
t	i	0	1	2	3	4	5	6	7	8	9
1	0	-	2	2	$+\infty$	7	2	$+\infty$	$+\infty$	8	$+\infty$
	1	-	-	$+\infty$	5	$+\infty$	$+\infty$	8	$+\infty$	5	$+\infty$
	2	-	-	-	$+\infty$	5	7	$+\infty$	6	$+\infty$	$+\infty$
	3	-	-	-	-	$+\infty$	$+\infty$	3	$+\infty$	9	$+\infty$
	4	-	-	-	-	-	4	$+\infty$	9	$+\infty$	8
	5	-	-	-	-	-	-	1	3	2	7
	6	-	-	-	-	-	-	-	2	5	9
	7	-	-	-	-	-	-	-	-	$+\infty$	4
	8	-	-	-	-	-	-	-	-	-	$+\infty$
	9	-	-	-	-	-	-	-	-	-	-
2	0	-	5	9	$+\infty$	2	8	$+\infty$	$+\infty$	5	$+\infty$
	1	-	-	$+\infty$	1	$+\infty$	$+\infty$	1	$+\infty$	3	$+\infty$
	2	-	-	-	$+\infty$	4	4	$+\infty$	1	$+\infty$	$+\infty$
	3	-	-	-	-	$+\infty$	$+\infty$	8	$+\infty$	9	$+\infty$
	4	-	-	-	-	-	9	$+\infty$	4	$+\infty$	5
	5	-	-	-	-	-	-	8	2	6	6
	6	-	-	-	-	-	-	-	2	6	7
	7	-	-	-	-	-	-	-	-	$+\infty$	2
	8	-	-	-	-	-	-	-	-	-	$+\infty$
	9	-	-	-	-	-	-	-	-	-	-
3	0	-	5	6	$+\infty$	6	9	$+\infty$	$+\infty$	8	$+\infty$
	1	-	-	$+\infty$	9	$+\infty$	$+\infty$	3	$+\infty$	6	$+\infty$
	2	-	-	-	$+\infty$	5	9	$+\infty$	7	$+\infty$	$+\infty$
	3	-	-	-	-	$+\infty$	$+\infty$	4	$+\infty$	10	$+\infty$
	4	-	-	-	-	-	8	$+\infty$	3	$+\infty$	4
	5	-	-	-	-	-	-	4	7	8	5
	6	-	-	-	-	-	-	-	9	5	3
	7	-	-	-	-	-	-	-	-	$+\infty$	3
	8	-	-	-	-	-	-	-	-	-	$+\infty$
	9	-	-	-	-	-	-	-	-	-	-
4	0	-	6	9	$+\infty$	9	6	$+\infty$	$+\infty$	3	$+\infty$
	1	-	-	$+\infty$	6	$+\infty$	$+\infty$	9	$+\infty$	9	$+\infty$
	2	-	-	-	$+\infty$	2	8	$+\infty$	6	$+\infty$	$+\infty$
	3	-	-	-	-	$+\infty$	$+\infty$	5	$+\infty$	5	$+\infty$
	4	-	-	-	-	-	9	$+\infty$	3	$+\infty$	7
	5	-	-	-	-	-	-	4	7	6	7
	6	-	-	-	-	-	-	-	6	8	10
	7	-	-	-	-	-	-	-	-	$+\infty$	9
	8	-	-	-	-	-	-	-	-	-	$+\infty$
	9	-	-	-	-	-	-	-	-	-	-

(a) p_{ij}^t values.



(b) Non-empty and disjoint subsets of active nodes connected under a tree topology configuration in the different time slots.

$$\sum_{(i,j)\in E^t} x_{ij}^t = \sum_{i\in N} z_i^t - 1 \qquad t \in T \tag{1c}$$

$$\sum_{j\in N|(i,j)\in E^t} x_{ij}^t \leq (|N| - 1 - (|T| - 1))z_i^t \qquad i \in N, t \in T \tag{1d}$$

$$\sum_{(i,j)\in E^t(S)} x_{ij}^t \leq \sum_{i\in S} z_i^t - z_h^t \qquad S \subset N, |S| \geq 2, h \in S, t \in T \tag{1e}$$

$$z_i^t \in \{0,1\} \qquad i \in N, t \in T \tag{1f}$$

$$x_{ij}^t \in \{0,1\} \qquad (i,j) \in E^t, t \in T \tag{1g}$$

The objective function (1a) minimizes the total power costs. Constraints (1b) impose that each network node has to be active in a unique time slot. Constraints (1c) ensure that the total number of edges selected for each time slot $t \in T$ equals the total number of active nodes minus one. This is the necessary condition to obtain a subtree in each time slot. Consistency among the x_{ij}^t and z_i^t variables are imposed in constraints (1d) for each $i \in N$ and $t \in T$. Coefficient $(|N| - 1 - (|T| - 1))$ comes from the facts that an active node can be connected to at most $|N| - 1$ additional active nodes in each time slot, and that at least one node has to be active in each time slot. Then, constraints (1e), the generalized version of the subtour elimination constraints (GSECs) introduced by Dantzig *et al.* (1954), impose that any connected component resulting from an assignment of binary values to variables z_i^t and x_{ij}^t, $t \in T$, does not contain cycles. Thus, for any assignment of binary values to variables z_i^t and x_{ij}^t, $t \in T$, constraints (1e), together with constraints (1c) and (1d), guarantee the definition of subsets of active nodes $S^t \subset N$, $S^t \neq \varnothing$, connected under a tree topology in each time slot $t \in T$. In particular, $S^t \neq \varnothing$ is implied by (1c). Finally, constraints (1f) and (1g) define the domains of the variables.

Modelling the problem directly on graphs $G^t = (N, E^t)$, $t \in T$, avoids introducing symmetries in the solution space. Nevertheless, the number of constraints (1e) is exponential in the cardinality of N. This excludes the possibility to explicitly define model (1) and solve it directly through a commercial Mixed-Integer Linear Programming (MILP) solver, even for relatively small instances of the problem. Thus, we designed a branch-and-cut (BC) algorithm, based on model (1), in which constraints (1e) are dynamically inserted into the formulation when violated by fractional/integer solutions.

3. Branch-and-cut algorithm

In this section, we present the main components of the branch-and-cut (BC) algorithm we devised for solving the STSWSN-PC.

The BC algorithm considers the initial Linear Programming (LP) relaxation of model (1) defined by (1a), (1b)-(1d), and the linear relaxation of the integrality constraints (1f) and (1g). For the presentation of the components, we denote by $\bar{s} = (\bar{\mathbf{z}}, \bar{\mathbf{x}})$ the (fractional) solution to current LP relaxation (the initial LP relaxation, eventually augmented by branching and cutting constraints). Then, we define $\bar{E}^t \subseteq E^t$ as the subset of edges corresponding to the direct communication channels to use in time slot $t \in T$ according to solution $\bar{s}$, i.e., edges $(i,j) \in E^t$ such that $0 < \bar{x}_{ij}^t \leq 1$. Finally, we define $\bar{H}^t = (\bar{N}^t \subset N, \bar{E}^t)$ as the graph induced by edges in $\bar{E}^t$.

3.1. Static valid inequalities

In order to strengthen the initial LP relaxation, we include in it constraints (2a) and (2b), bounding respectively the minimum and maximum cardinality of the subsets of active nodes S^t, $t \in T$.

$$\sum_{i\in N} z_i^t \geq 1 \qquad t \in T \tag{2a}$$

$$\sum_{i\in N} z_i^t \leq |N| - (|T| - 1) \qquad t \in T \tag{2b}$$

Similarly, we impose constraints bounding the maximum number of direct communication channels that can be established per time slot, (3a), and in total, (3b).

$$\sum_{(i,j)\in E^t} x_{ij}^t \leq (|N| - 1 - (|T| - 1)) \qquad t \in T \tag{3a}$$

$$\sum_{t\in T} \sum_{(i,j)\in E^t} x_{ij}^t \leq |N| - 1 \tag{3b}$$

3.2. Separation algorithm for the GSECs

The separation algorithm we devised is exact for integer solutions, in the sense that at least one violated GSEC is found if one exists, and heuristic when the solution to the current LP relaxation is fractional. In this latter case, when the algorithm does not compute any violated GSEC, branching is performed to remove the current fractional solution from the search space.

The main steps of the algorithm are discussed in the following.

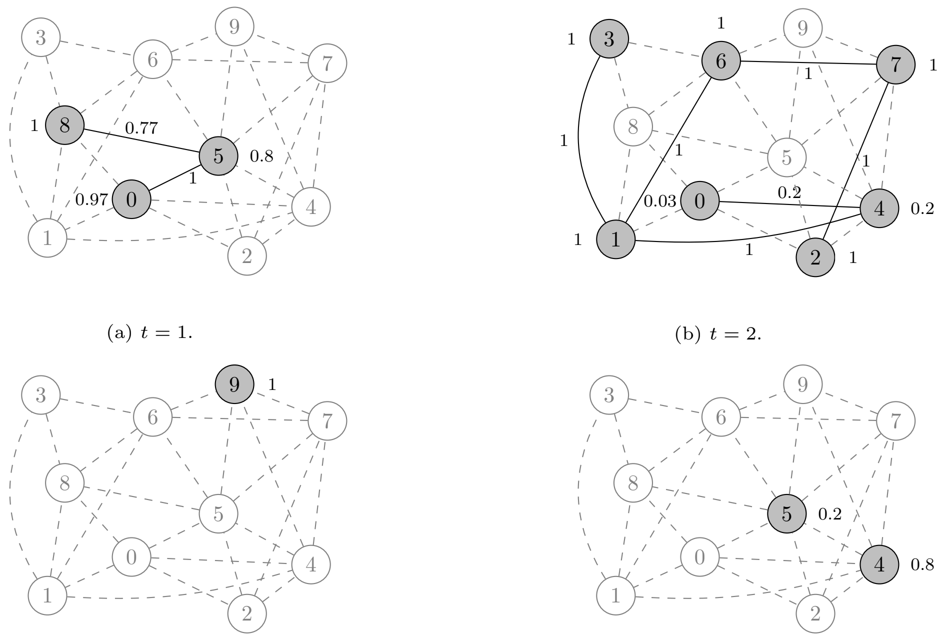
Step 1. By applying the algorithm described in (Goodrich and Tamassia, 2014, p. 222)), connected components (CCs) are computed for each graph $\bar{H}^t = (\bar{N}^t, \bar{E}^t)$, $t \in T$.

Step 2. When the solution is integer and the number of CCs in each graph $\bar{H}^t = (\bar{N}^t, \bar{E}^t)$, $t \in T$, is equal to 1, no violated GSEC exists. This is due to constraints (1c), which ensure that a unique CC in a graph $\bar{H}^t$ is a tree. Thus, the solution is feasible to (1). The separation algorithm terminates. Otherwise, the algorithm performs the following step.

Step 3. This step is reached if (i) the solution is fractional (independently of the number of CCs in each graph $\bar{H}^t = (\bar{N}^t, \bar{E}^t)$, $t \in T$) or (ii) when the solution is integer and the number of CCs is greater than 1 for at least one graph $\bar{H}^t$, $t \in T$. Let κ be an integer parameter, equal to 2 if the solution is integer, and 1 otherwise. Then, let $\bar{\mathcal{C}}^t = \{\bar{C}_1^t, \ldots, \bar{C}_{|\bar{\mathcal{C}}^t|}^t\}$ represent the set of the CCs in time slot $t \in T$. For each $t \in T$ such that $|\bar{\mathcal{C}}^t| \geq \kappa$, the algorithm considers each connected component $\bar{C}_i^t \in \bar{\mathcal{C}}^t$ and checks whether $\sum_{(i,j)\in \bar{E}^t(\bar{C}_i^t)} \bar{x}_{ij}^t > \sum_{i\in \bar{C}^t} \bar{z}_i^t - \bar{z}_h^t$ for some $h \in \bar{C}^t$ or not. More precisely, when $\sum_{(i,j)\in \bar{E}^t(\bar{C}_i^t)} \bar{x}_{ij}^t - (\sum_{i\in \bar{C}^t} \bar{z}_i^t - \bar{z}_h^t)$ is greater than 0.05 for some $h \in \bar{C}^t$, the violated GSEC associated with time slot t, subset of nodes $\bar{C}^t$, and node $h^* = argmax_{h\in \bar{C}^t}\{\sum_{(i,j)\in \bar{E}^t(\bar{C}^t)} \bar{x}_{ij}^t - (\sum_{i\in \bar{C}^t} \bar{z}_i^t - \bar{z}_h^t)\}$ is inserted into the current LP relaxation. (For a given graph $\bar{H}^t$, it may happen that GSECs for several connected components are added at the same time to the current LP relaxation.)
In case (i), the fractional values $(\bar{\mathbf{z}}, \bar{\mathbf{x}})$ characterizing solution $\bar{s}$ may allow the definition of CCs (eventually containing cycles) that do not violate the corresponding GSECs (independently of the number of CCs in each graph $\bar{H}^t = (\bar{N}^t, \bar{E}^t)$, $t \in T$). This is illustrated in Figure 2, in which the fractional solution represented is relative to the sample instance described in Figure 1. Moreover, values $(\bar{\mathbf{z}}, \bar{\mathbf{x}})$ may allow the definition of CCs (eventually not containing cycles) that do violate the corresponding GSECs. This is shown in Figure 3, in which, always w.r.t. the instance described in Figure 1, another fractional solution is depicted.
On the contrary, in case (ii), as the number of CCs is greater than 1 for at least one graph $\bar{H}^t$, $t \in T$, the integer solution is infeasible and, in particular, there must exist a connected component $\bar{C}^{t^*} \subset \bar{N}^{t^*}$, $t^* \in T$, for which edges in $\bar{E}^{t^*}(\bar{C}^{t^*})$ define a cycle and, therefore, for which the corresponding GSEC is violated. In fact, let $\bar{H}^{t'}$ be one of the graphs for which the number of associated CCs is greater than 1. Each connected component $\bar{C}_i^{t'} \in \bar{\mathcal{C}}^{t'}$ would not include cycles if $|\bar{E}^{t'}(\bar{C}_i^{t'})| = |\bar{C}_i^{t'}| - 1$. This means that the total number of edges associated with variables $\bar{x}_{ij}^{t'}$ equal to 1 should be $\sum_{\bar{C}_i^{t'}\in \bar{\mathcal{C}}^{t'}} |\bar{C}_i^{t'}| - 1 = \sum_{i\in N} \bar{z}_i^{t'} - |\bar{\mathcal{C}}^{t'}|$. However, constraints (1c), imposes $\sum_{(i,j)\in E^{t'}} \bar{x}_{ij}^{t'} = \sum_{i\in N} \bar{z}_i^{t'} - 1$. Thus, in time slot t', further $|\bar{\mathcal{C}}^{t'}| - 1 \geq 1$ edges are associated with variables $\bar{x}_{ij}^{t'}$ equal to 1. Each of these edges connect two nodes already (not directly) connected in some $\bar{C}_i^{t'} \in \bar{\mathcal{C}}^{t'}$, defining thus a cycle. In case (ii), by applying Step 3, the algorithm is always able to compute the connected component $\bar{C}^{t^*}$. An infeasible integer solution to the instance described in Figure 1, including a connected component whose edges define a cycle, is illustrated in Figure 4.

Notably enough, by exploiting the specific features of the STSWSN-PC, the separation algorithm relies only on the computation of the CCs for each graph $\bar{H}^t = (\bar{N}^t, \bar{E}^t)$, $t \in T$.

Figure 2: A fractional solution for the instance described in Figure 1. Next to the nodes (edges) are reported the non-zero values of the corresponding $\bar{z}^t$ ($\bar{x}^t_{ij}$) variables. The solution does not violate any GSEC that can be defined on the basis of the connected components appearing in the different time slots.



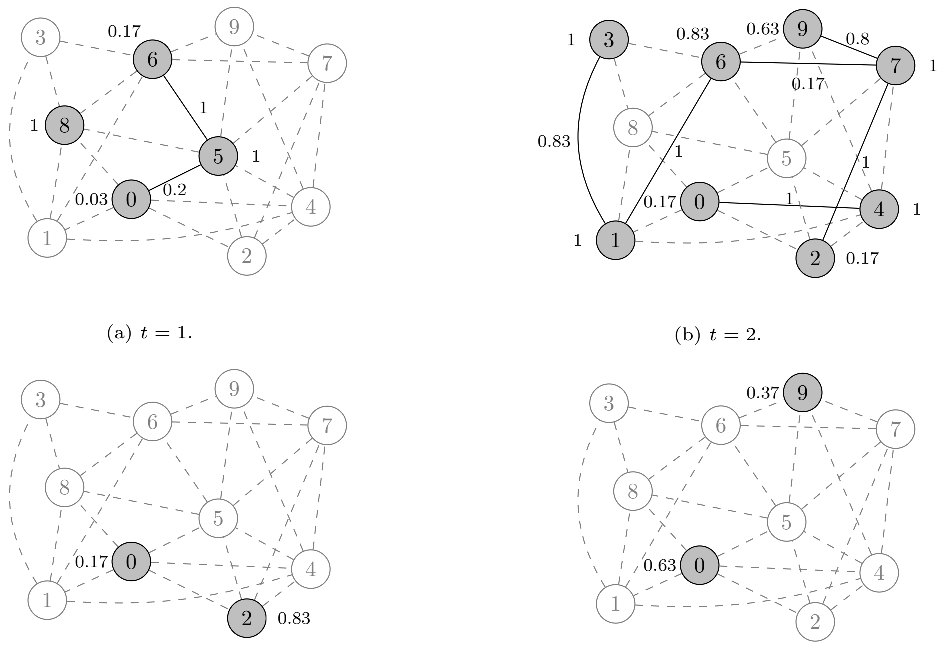
(a) $t = 1$.

(b) $t = 2$.

(c) $t = 3$.

(d) $t = 4$.

Figure 3: A fractional solution for the instance described in Figure 1. Next to the nodes (edges) are reported the non-zero values of the corresponding $\bar{z}^t$ ($\bar{x}^t_{ij}$) variables. The connected components associated with violated GSECs are: $\{0, 5, 6\}$ in time slot 1, and $\{0, 4\}$ and $\{1, 2, 3, 6, 7, 9\}$ in time slot 2.



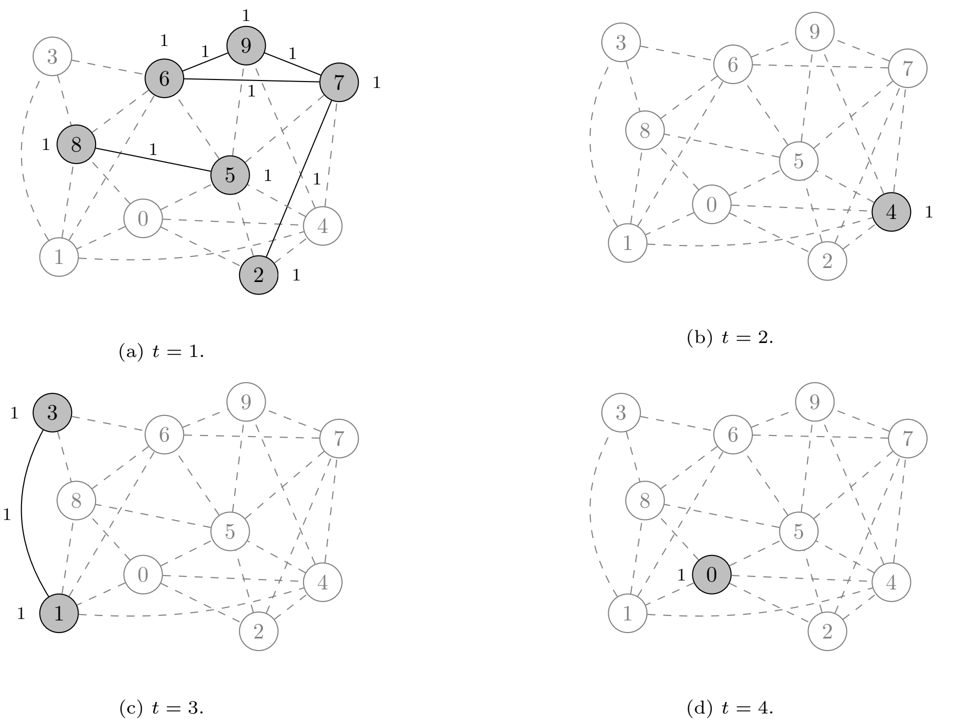
(a) $t = 1$.

(b) $t = 2$.

(c) $t = 3$.

(d) $t = 4$.

Figure 4: An infeasible integer solution for the instance described in Figure 1. Next to the nodes (edges) are reported the non-zero values of the corresponding $\bar{z}^t$ ($\bar{x}^t_{ij}$) variables. In time slot 1, the edges of connected components $\{2, 6, 7, 9\}$ define a cycle: the GSEC associated with the connected component is violated.



3.3. Primal bound heuristic

In order to speed up the BC algorithm, we embedded into the algorithm a primal bound heuristic. The heuristic is triggered whenever an integer solution violating the GSECs is found.

Again, we start from the CCs in each graph $\bar{H}^t = (\bar{N}^t, \bar{E}^t)$, $t \in T$.

For a given $t \in T$, when the number of CCs is greater than 1, we consider the set of CCs sorted in non-increasing order according to the cardinality of the components themselves, i.e., the set $\bar{\mathcal{C}}^t = \{\bar{C}^t_1, \ldots, \bar{C}^t_{|\bar{\mathcal{C}}^t|}\}$ such that $|\bar{C}^t_i| \geq |\bar{C}^t_{i+1}|$, $1 \leq i \leq |\bar{\mathcal{C}}^t| - 1$. Then, for each $\bar{C}^t_i \in \bar{\mathcal{C}}^t$, we define a minimum cost spanning tree w.r.t. nodes in $\bar{C}^t_i$ and edges in $\bar{E}^t(\bar{C}^t_i)$. To this aim, we apply an implementation of the Kruskal's algorithm (Kruskal, 1956). Finally, we iteratively merge all the trees associated with the CCs. More precisely, at iteration $i < |\bar{\mathcal{C}}^t|$, we merge the tree associated with the connected component $\bar{C}^t_{i+1}$ with that resulting from the merge of the trees associated with connected components $\bar{C}^t_k$, $k \leq i$. The merge is performed by considering the cheapest edge in E^t joining 1-degree nodes of the two trees. The merge is always possible as each tree with $n > 1$ nodes has at least two 1-degree nodes.

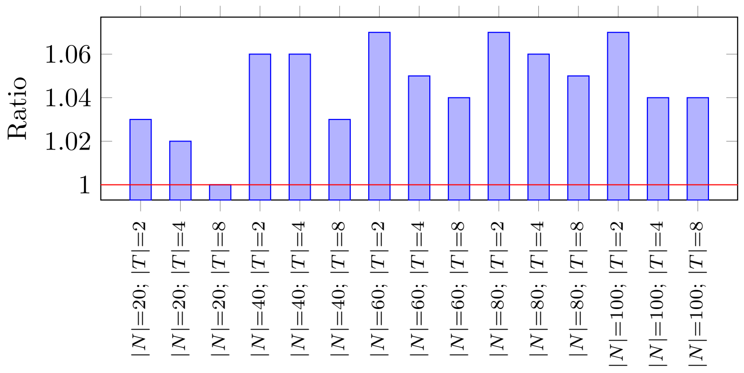
When a single tree spanning all the nodes in $\bar{N}^t$ has been defined for each $t \in T$, a feasible solution $\bar{s}'$ to (1) has been computed. The value $p(\bar{s}')$ may eventually improve the current primal bound and restrict the search space.

4. Experimental analysis

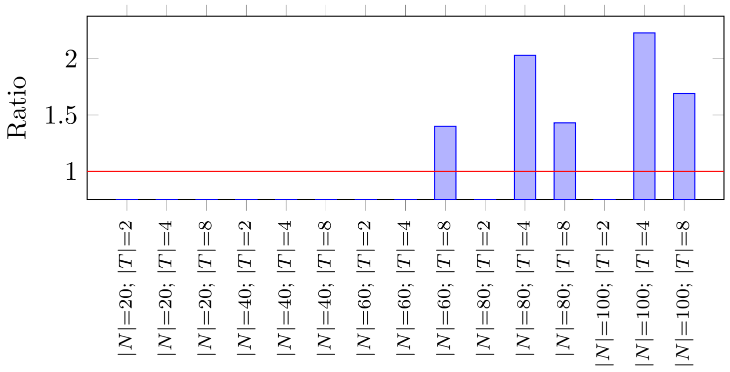
We tested the BC algorithm on instances generated as described in (Adasme, 2019). Problem instances are completely defined by undirected, weighted, complete graphs $G = (N, E^t)$, $t \in T$ (Adasme, 2019, Section 1). Thus, for each instance, we generated complete undirected graphs $G = (N, E^t)$ by associating with each edge $(i, j) \in E^t$ an integer power connectivity cost p^t_{ij} randomly drawn form the interval (0; 1000] μWs. The magnitude of the interval allows to fairly compare the performance of the presented BC algorithm with those of the exact algorithms proposed in (Adasme, 2019). We considered 10 instances for each $|N| \in \{20, 40, 60, 80, 100, 120, 140, 160, 180, 200\}$ and $|T| \in \{2, 4, 8\}$, for a total of 300 instances.

The BC algorithm has been implemented in C++, by using CPLEX 20.1 Concert Technology, and compiled in release mode with MS Visual Studio Community 2022. The experiments have been carried out on a

Figure 5: Solution of the linear relaxation of (1) plus constraints (2) and (3) - Root node of the branch-and-bound tree - Subsets of instances with $|N| \leq 100$



(a) Geometric means of ratios of lower bounds $LB^{\texttt{GSECs}}/LB^{\texttt{NO-GSECs}}$



(b) Geometric means of ratios of solution times $Sol.\ time^{\texttt{GSECs}}/Sol.\ time^{\texttt{NO-GSECs}}$

64-bit Windows PC, with the Intel Xeon processor W-1250P, 4.10 GHz, and 32 GB of RAM. CPLEX built-in cuts have been used in all experiments. Due to numerical instability we set `IloCplex::NumericalEmphasis = CPX ON`. Moreover, we set `IloCplex::ParallelMode` to 1 in order to force CPLEX to always use deterministic algorithms, and parameter `IloCplex::Threads` has been set to 1 in order to get deterministic results. In fact, due to threads synchronization, by embedding user defined components into CPLEX, its behaviour become non-deterministic. For all the other CPLEX's parameters we kept their default values.

In order to assess the impact of the main components of the BC algorithm, and finalize its design, we have initially run some preliminary experiments on a subset of the generated instances. Then, the performance of the final version of the BC algorithm have been assessed on the full set of instances. The analysis of the results obtained is reported thereafter. Instances and instance-wise detailed results concerning the experiments discussed in Section 4.2 are publicly available (Bianchessi, 2023).

4.1. Preliminary experiments

We first compared the performance of the BC algorithm by allowing or not the separation of fractional solutions by means of the algorithm described in Section 3.2, i.e. the algorithm for the separation of the GSECs. We limited the performance comparison to what concerns the solution of the LP relaxation at the root node of the branch-and-bound tree. To this aim, only for this specific subset of preliminary experiments, we temporarily set `IloCplex::HeurFreq = -1`, preventing CPLEX to run its internal primal bound heuristics.

For the subsets of instances considered, we report in Figure 5-(a) the geometric means of the ratios of lower bounds taken over the 10 instances of each subset. For a given subset, we computed instance by instance the ratio $LB^{\texttt{GSECs}}/LB^{\texttt{NO-GSECs}}$, and provided the graphical representation of the geometric mean over the 10 instances of the subset. For example, the fourth bar associated with an average ratio $LB^{\texttt{GSECs}}/LB^{\texttt{NO-GSECs}}$ of about 1.06, above 1, means that, for the subset of instances $|N| = 40$; $|T| = 2$, lower bounds computed by applying the separation of the GSECs are consistently greater (by a factor 1.06 on average) than those computed without separating the GSECs. The results reported show that it is beneficial applying the separation of the GSECs for all the subsets but subset $|N| = 20$; $|T| = 2$. In particular, the impact of GSECs on the tightening of the lower bounds seems to decrease with the increase of $|T|$, allowing however to improve the bounds by about a factor of at least 1.04, on average, for $|N| \geq 60$. Then, in Figure 5-(b) we show the corresponding geometric means of the ratios of solution times needed to solve the linear relaxation of model (1) plus constraints (2) and (3). The ratios are reported only for subset of instances for which the BC algorithm, running under at least one of the two settings, required more than 1 second, on average, to solve the instances in the subset. The results depicted show that the average solution time may double by applying the separation of the GSECs. However, it is worth mentioning that the average solution time remains less than 7.5 seconds for the instances of subset $|N| = 100$; $|T| = 8$, i.e, the instances associated with the greatest solution times.

Having empirically shown the usefulness of applying the separation algorithm of the GSECs to cut off fractional solutions, in order to finalize the design of the BC algorithm we proceeded as follows. We derived four different variants from the BC algorithm, `BC-S-H`, BC-$\overline{\texttt{S}}$-H, BC-S-$\overline{\texttt{H}}$, and BC-$\overline{\texttt{S}}$-$\overline{\texttt{H}}$, by switching off:

- the possibility to apply the separation algorithm of the GSECs to cut off fractional solutions at non-root nodes of the branch-and-bound tree ($\overline{\texttt{S}}$);
- the possibility to apply the primal bound heuristic described in Section 3.3 ($\overline{\texttt{H}}$).

For every variant, we solved the 150 instances with $|N| \leq 100$ nodes. The time limit (`TL`) for each run was set to 300 seconds. Each algorithmic variant was able to solve all the instances with $|N| = 20$ nodes by exploring in most of the cases only the root node of the tree, and with an average computing time less than 1 second. The average results for the remaining instances with $40 \leq |N| \leq 100$ nodes are reported in Table 1.

For each combination of number of nodes ($|N|$) and time slots ($|T|$), and each algorithmic variant, we report the average solution time (*Time*) in seconds, the average number of branch-and-bound nodes explored (*Nodes*), and the number of feasible instances solved to the optimality (*Opt*). Then, if any, for instances not optimally solved for which valid lower/dual and upper/primal bounds have been found, we report the average percentage optimality gap (*Gap (%)*) and, in brackets, the number of instances over which the average gap is computed. In particular, the gap is computed according to the formula $100 * (UB^* - LB^*)/UB^*$, where UB^* (LB^*) is the best upper (lower) bound value computed during the search. Note that each row of the table reports the aggregate results for the subset of 10 instances associated with the same values of $|N|$ and $|T|$. Additionally, note that the average number of nodes explored in column '*Nodes*' is reported in thousands.

Looking at the results reported for instances with $|N| \geq 60$, especially at the average solution times and number of instances solved to optimality, we can observe that the problem becomes more difficult to solve at the increase of the number of nodes in the network ($|N|$) and the number of time slots ($|T|$). The average number of instances solved to optimality decreases with the increase of $|N|$ and/or $|T|$. Accordingly, the average solution time grows steadily. The average percentage optimality gap follows a trend similar to that of the average solution time. These considerations apply to all the algorithmic variants.

Comparing the average number of nodes explored for BC-$\overline{\texttt{S}}$-H and BC-$\overline{\texttt{S}}$-$\overline{\texttt{H}}$ against the corresponding values for `BC-S-H` and BC-S-$\overline{\texttt{H}}$, respectively, we can see that, by always applying the separation of the GSECs, the time required to solve each node of the branch-and-bound tree increases substantially. In fact, the average number of nodes explored decreases by about one order of magnitude. This makes BC-$\overline{\texttt{S}}$-H and BC-$\overline{\texttt{S}}$-$\overline{\texttt{H}}$ more effective than the respective counterparts (`BC-S-H` and BC-S-$\overline{\texttt{H}}$), especially w.r.t. the number of optimal solutions found. This latter consideration seems to be slightly in contrast with the average results for specific subsets of instances. For example, for instances of subset $|N| = 80$; $|T| = 2$, BC-$\overline{\texttt{S}}$-H and `BC-S-H` compute respectively 9 and 10 optimal solutions. Similarly, BC-S-$\overline{\texttt{H}}$ computes one more optimal solution than BC-$\overline{\texttt{S}}$-$\overline{\texttt{H}}$ for instances of subset $|N| = 80$; $|T| = 8$ and $|N| = 100$; $|T| = 2$. Additionally, w.r.t. subset $|N| = 80$; $|T| = 8$, `BC-S-H` is slightly better than BC-$\overline{\texttt{S}}$-H as for both the average solution time and average percentage optimality gap (for the same number of instances not solved). Nevertheless, these small deviations from the general trend substantially depend on the different branch-and-bound trees explored by the algorithms. The overall results support the greater efficacy of BC-$\overline{\texttt{S}}$-H and BC-$\overline{\texttt{S}}$-$\overline{\texttt{H}}$ over `BC-S-H` and BC-S-$\overline{\texttt{H}}$, respectively. Furthermore, thanks to the possibility of applying the primal bound heuristic, BC-$\overline{\texttt{S}}$-H is slightly better than BC-$\overline{\texttt{S}}$-$\overline{\texttt{H}}$ with respect to both the number of instances optimally solved and the average solution time, halving in addition the average percentage gap for the instances not solved to optimality. To some extent, when the size of the problem instances is not too large, this makes BC-$\overline{\texttt{S}}$-H also suitable to be used in practice as an heuristic. We elaborate more on this in the next section.

Table 1: Computational results on instances with $40 \leq |N| \leq 100$ - TL = 300 sec.

Subset		BC-S-H				BC-$\overline{\text{S}}$-H				BC-S-$\overline{\text{H}}$				BC-$\overline{\text{S}}$-$\overline{\text{H}}$			
$\|N\|$	$\|T\|$	Time	Nodes	Opt	Gap (%)	Time	Nodes	Opt	Gap (%)	Time	Nodes	Opt	Gap (%)	Time	Nodes	Opt	Gap (%)
40	2	0.3	0.1	10		0.3	0.1	10		0.3	0.1	10		0.3	0.2	10	
	4	1.4	0.2	10		1.2	0.4	10		1.3	0.2	10		1.0	0.3	10	
	8	2.5	0.2	10		3.1	0.6	10		2.3	0.2	10		2.4	0.4	10	
Av./Tot.		1.4	0.2	30		1.5	0.4	30		1.3	0.2	30		1.2	0.3	30	
60	2	1.1	0.2	10		1.0	0.5	10		1.1	0.2	10		1.4	0.7	10	
	4	10.2	0.8	10		6.6	1.8	10		8.2	0.6	10		7.0	2.3	10	
	8	14.4	0.7	10		17.6	2.8	10		14.3	0.7	10		16.2	2.7	10	
Av./Tot.		8.6	0.6	30		8.4	1.7	30		7.9	0.5	30		8.2	1.9	30	
80	2	36.5	1.5	10		38.7	14.2	9	0.5 (1)	35.9	1.5	9	2.0 (1)	46.3	20.2	9	0.6 (1)
	4	79.0	1.9	9	10.8 (1)	45.0	9.6	10		107.9	2.4	9	8.6 (1)	53.8	10.9	9	1.3 (1)
	8	154.4	3.0	7	5.3 (3)	162.5	17.3	7	7.5 (3)	163.4	3.2	8	10.6 (2)	150.1	16.5	7	19.5 (3)
Av./Tot.		90.0	2.2	26	6.7 (4)	82.0	13.7	26	5.8 (4)	102.4	2.4	26	7.9 (4)	83.4	15.9	25	12.1 (5)
100	2	184.2	3.1	6	4.5 (4)	149.5	41.0	7	2.3 (3)	167.9	3.1	6	3.3 (4)	205.4	52.8	5	2.7 (5)
	4	293.6	3.0	1	9.5 (9)	175.4	25.1	7	3.8 (3)	TL	3.2	0	10.4 (10)	178.1	25.8	7	4.9 (3)
	8	TL	2.2	0	26.5 (10)	TL	17.1	0	19.4 (10)	TL	2.3	0	28.1 (10)	TL	16.5	0	44.6 (10)
Av./Tot.		259.3	2.8	7	16.0 (23)	208.3	27.8	14	13.3 (16)	256.0	2.9	6	16.6 (24)	227.9	31.7	12	26.3 (18)
Av./Tot.		89.8	1.4	93	14.6 (27)	75.1	10.9	100	11.8 (20)	91.9	1.5	92	15.3 (28)	80.2	12.4	97	23.2 (23)

We conclude our preliminary analysis briefly reporting on the average results that BC-$\overline{\text{S}}$-H computes when constraints (2) and (3) are not included into the model. By considering again the instances with $40 \leq |N| \leq 100$ nodes, and a time limit of 300 seconds for each run, we get the following overall results:

- 102 instances solved to optimality;
- an average solution time of 74.4 seconds;
- an average percentage gap for the instances not solved to optimality of 17.4%.

This large value of the average percentage gap empirically shows the usefulness of including the static valid inequalities (2) and (3) into the model.

As a consequence of all the analysis outlined in this section, we run the final experiments by using BC $\equiv$ BC-$\overline{\text{S}}$-H as final version of the BC algorithm.

4.2. Global assessment

We run BC to solve the full set of generated instances with a TL of 1 hour.

Results are reported in Table 2. The structure of Table 2 is similar to that of Table 1, but now the aggregate results for each combination of number of nodes ($|N|$) and time slots ($|T|$) are reported for the unique algorithmic variant BC. In particular, the average percentage optimality gap (*Gap (%)*) is computed over all the instances in each subset. Moreover, column (*Root* | *Gap (%)*) and (*Root* | *Time*) report the average percentage optimality gap and the average computing time, respectively, after the solution of the linear relaxation of model (1), plus constraints (2) and (3), at the root node of the branch-and-bound tree. Values in column (*Root* | *Gap (%)*) are computed according to the formula $100 * (UB^* - LB_{root})/UB^*$.

The most striking result is that BC is able to optimally solve 60% of the instances (182 over 300). In particular:

- all but 2 instances with $|N| \leq 100$ have been optimally solved;
- all the instances with $|N| \leq 60$ ($|N| \leq 80$) have been solved to optimality with and average solution time less than 20 (200) seconds (see subset of instances $|N| = 60$; $|T| = 8$ ($|N| = 80$; $|T| = 8$));
- for $|T|$ equal to 2, 4, and 8, instances are optimally solved up to $|N|$ equal to 200, 200, and 140, respectively.

As for the average percentage optimality gap (i.e., a measure of quality concerning the primal/dual bounds computed):

- for $|T| = 2$, it is slightly above 5% only for instances with $|N| = 180$ nodes;
- for $|T| = 4$, it is above 10% only for instances with $|N| = 200$; equal to 15.73% for subset of instances $|N| = 200$; $|T| = 4$;
- for $|T| = 8$, it is less than 6.5% (19%) for all subset instances with $|N| \leq 120$ ($|N| \leq 140$).

Table 2: Computational results on the full set of instances - `TL` = 1 h.

Subset		BC			Root	
$\|N\|$	$\|T\|$	Time	Opt	Gap (%)	Gap (%)	Time
20	2	0.1	10	0.00	0.00	0.1
	4	0.1	10	0.00	0.48	0.1
	8	0.3	10	0.00	1.16	0.2
Av./Tot.		0.2	30	0.00	0.55	0.1
40	2	0.4	10	0.00	1.97	0.1
	4	1.2	10	0.00	1.84	0.3
	8	3	10	0.00	4.24	0.7
Av./Tot.		1.5	30	0.00	2.68	0.4
60	2	1	10	0.00	2.50	0.2
	4	6.7	10	0.00	3.58	0.9
	8	17.8	10	0.00	5.40	2
Av./Tot.		8.5	30	0.00	3.83	1
80	2	53.5	10	0.00	2.73	0.4
	4	48.5	10	0.00	3.56	1.8
	8	195.2	10	0.00	7.35	3.9
Av./Tot.		99	30	0.00	4.55	2
100	2	585.1	9	0.18	4.12	0.7
	4	246.1	10	0.00	5.45	3.3
	8	1293.3	9	0.20	8.31	7.2
Av./Tot.		708.2	28	0.13	5.96	3.7

(a) $20 \leq |N| \leq 100$

Subset		BC			Root	
$\|N\|$	$\|T\|$	Time	Opt	Gap (%)	Gap (%)	Time
120	2	1607.7	7	0.67	3.32	1.1
	4	1069.9	8	1.63	5.64	4.4
	8	3480.4	1	6.24	12.28	11.8
Av./Tot.		2052.6	16	2.84	7.08	5.8
140	2	2753.3	3	1.95	4.39	2.6
	4	2371.2	4	1.61	5.61	5.9
	8	3106.1	3	18.72	22.31	16.3
Av./Tot.		2743.5	10	7.43	10.77	8.3
160	2	3221.2	2	4.85	6.68	3.7
	4	2887.1	3	6.88	10.09	9.4
	8	TL	0	36.53	38.73	23.1
Av./Tot.		3236.1	5	16.08	18.50	12
180	2	TL	0	5.88	7.12	5.2
	4	3320.1	1	9.70	12.09	11.6
	8	TL	0	49.50	50.84	30.5
Av./Tot.		3506.8	1	21.69	23.35	15.8
200	2	3272.8	1	4.21	5.34	7.8
	4	3437.6	1	15.73	17.66	24.2
	8	TL	0	55.06	56.06	44.2
Av./Tot.		3436.8	2	25.00	26.35	25.4

(b) $120 \leq |N| \leq 200$

Thus, `BC` outperforms the exact algorithms proposed in Adasme (2019) that, with a time limit of 2 hours, were able to optimally solve only instances with up to 40 network nodes and 2 time slots, and to compute feasible solutions only for instances with up 70 network nodes and 4 time slots.

The average computing time required to solve the root node of the branch-and-bound tree is less than 44.2 seconds (see subset of instances $|N| = 200$; $|T| = 8$). Already after the solution of the root node, the average percentage optimality gap is less than 5.5% (8.5%) for instances solved to optimality with up to 100 network nodes and 4 (8) time slots. With respect to all the instances optimally solved, the average and maximum percentage optimality gap (after the solution of the root node) are respectively 3.44% and 10.47%. Therefore, `BC`, besides being an effective exact solution algorithm, represents a valid computational tool to benchmark the performance of heuristics addressing the problem.

Results concerning average percentage optimality gap are further elaborated in Figure 6 for instances with $|N| \geq 80$ (for each subset of instances with $|N| \leq 80$, the average percentage optimality gap is always less than 0.7% already with a `TL` of 300 seconds). For each value of $|T| \in \{2, 4, 8\}$, and each subset of instances with $|N| \geq 80$, we illustrate the variation of the average percentage optimality gap at the increase of the available computing time. Even by setting the time limit to 150 seconds, `BC` is always able to compute valid primal and dual bounds. Already with a `TL` of 900 seconds, by considering all the subsets of instances with $|N| \leq 100$, the average percentage optimality gap is above 0.2% only for subset $|N| = 100$; $|T| = 8$, with a value of 3.5%. Always by considering a `TL` of 900 seconds, as for the average percentage optimality gap concerning all the instances:

- for $|T| = 2$, it is slightly above 7% only for instances with $|N| = 180$ nodes;
- for $|T| = 4$, it is above 5% only for instances with $|N|$ equal to 160, 180, and 200, respectively equal to 12.25%, 15.99%, and 15.73%;
- for $|T| = 8$, it is less than 3.5% (24%) for all instances with $|N| \leq 100$ ($|N| \leq 120$).

This makes `BC` also suitable to be used in practice, as an heuristic solver, to address problem instances that are not too large.

5. Conclusions

In this study, we considered the problem to minimize the total connectivity power costs associated with the use of a wireless sensor network over consecutive time slots. Non-empty and disjoint subsets of nodes are required to be active and connected under a tree topology configuration in the different time slots, and each network node must be active in a unique time slot. Moreover, the power required by the same pair of network nodes to communicate on the associated direct channel may vary in the different time slots. The problem has been recently introduced in the literature to the aim of extending the lifetime of WSNs. We focused on the exact solution of the problem. We presented a branch-and-cut (BC) algorithm which improves the state-of-the-art w.r.t. existing exact solution algorithms, as shown by means of extensive computational experiments on 300 newly generated benchmark instances with up to 200 network nodes and 8 time slots. The BC algorithm is based on a novel Integer Linear Programming formulation which allows avoiding the introduction of symmetries in the solution space, on an efficient and non-typical separation algorithm for the generalized subtour elimination constraints, and on an easy-to-implement primal bound heuristic to restrict the search space. To the best of our knowledge, the largest problem instances optimally solved so far in the literature (with a time limit of 2 hours) consider up to 40 network nodes and 2 time slots. By forcing single-thread runs, the presented BC algorithm solves to optimality instances with up to 60 (80) network nodes and 8 time slots with a maximum solution time of 44 (510) seconds. With a time limit of 1 hour, the algorithm is able to optimally solve instances with up to 200, 200, and 140, network nodes for a number of time slots respectively equal to 2, 4, and 8. Additionally, being able to compute tight dual bounds, the BC algorithm may represent a valid computational tool to benchmark the performance of heuristics addressing the problem. Finally, even by setting the time limit to 150 seconds, the BC algorithm is always able to compute valid primal and dual bounds. With a time limit of 900 seconds, the average optimality gap is less than 3.5% for instances with up 100 network nodes and 8 time slots. In particular, when the number of time slots is 2 (4), the average percentage optimality gap is always less than 7.5% (16%). When the number of time slots is 8, the gap is less than 3.5% (24%) for instances with up to 100 (120) network nodes. This makes the BC algorithm also suitable to be used in practice, as an heuristic solver, to address problem instances which are not too large.

References

Adasme, P. (2019). Optimal sub-tree scheduling for wireless sensor networks with partial coverage. *Computer Standards & Interfaces*, **61**, 20–35.

Akyildiz, I., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002). Wireless sensor networks: a survey. *Computer Networks*, **38**(4), 393–422.

Anastasi, G., Conti, M., Di Francesco, M., and Passarella, A. (2009). Energy conservation in wireless sensor networks: A survey. *Ad Hoc Networks*, **7**(3), 537–568.

Anderson, A. J. (2015). *Channel prediction in wireless communications.* Phd thesis, The University of Edinburgh. http://hdl.handle.net/1842/16188.

Bianchessi, N. (2023). Sub-Tree Scheduling for Wireless Sensor Networks with Partial Coverage (STSWSN-PC) Dataset. UNIMI Dataverse, https://doi.org/10.13130/RD_UNIMI/IHTWC0.

Bidoki, N. H., Baghdadabad, M. B., Sukthankar, G., and Turgut, D. (2018). Joint value of information and energy aware sleep scheduling in wireless sensor networks: A linear programming approach. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–6.

Bulut, E. and Korpeoglu, I. (2011). Sleep scheduling with expected common coverage in wireless sensor networks. *Wireless Networks*, **17**, 19–40.

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.

Goodrich, M. T. and Tamassia, R. (2014). *Algorithm Design and Applications.* Wiley.

Guruprakash, B., Balasubramanian, C., and Sukumar, R. (2020). An approach by adopting multi-objective clustering and data collection along with node sleep scheduling for energy efficient and delay aware WSN. *Peer-to-Peer Networking and Applications*, **13**(1), 304–319.

Hansen, P. and Mladenović, N. (2003). *Variable Neighborhood Search*, pages 145–184. Springer US, Boston, MA.

Hawbani, A., Wang, X., Sharabi, Y., Ghannami, A., Kuhlani, H., and Karmoshi, S. (2019). LORA: Load-Balanced opportunistic routing for asynchronous Duty-Cycled WSN. *IEEE Transactions on Mobile Computing*, **18**(7), 1601–1615.

Ishmanov, F., Malik, A. S., and Kim, S. W. (2011). Energy consumption balancing (ecb) issues and mechanisms in wireless sensor networks (wsns): a comprehensive overview. *European Transactions on Telecommunications*, **22**(4), 151–167.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, **220**(4598), 671–680.

Kruskal, J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, **7**(1), 48–50.

Kulshrestha, J. and Mishra, M. K. (2021). Energy balanced data gathering approaches, issues and research directions. *Telecommunication Systems*, **76**(2), 299–327.

Kumar, S. and Kim, H. (2019). Energy efficient scheduling in wireless sensor networks for periodic data gathering. *IEEE Access*, **7**, 11410–11426.

Mladenović, N. and Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, **24**(11), 1097–1100.

Natarajan, M. and Subramanian, S. (2019). A cross-layer design: energy efficient multilevel dynamic feedback scheduling in wireless sensor networks using deadline aware active time quantum for environmental monitoring. *International Journal of Electronics*, **106**(1), 87–108.

Raghunathan, V., Schurgers, C., Park, S., and Srivastava, M. (2002). Energy-aware wireless microsensor networks. *IEEE Signal Processing Magazine*, **19**(2), 40–50.

Rost, P. and Fettweis, G. P. (2010). On the transmission-computation-energy tradeoff in wireless and fixed networks. *2010 IEEE Globecom Workshops*, pages 1394–1399.

Shallahuddin, A. A., Kadir, M. F. A., Mohamed, M. A., Abidin@Bharun, A. F. A., Usop, N. S. M., and Zakaria, Z. A. (2020). An enhanced adaptive duty cycle scheme for optimum data transmission in wireless sensor network. In K. J. Kim and H.-Y. Kim, editors, *Information Science and Applications*, pages 33–40, Singapore. Springer Singapore.

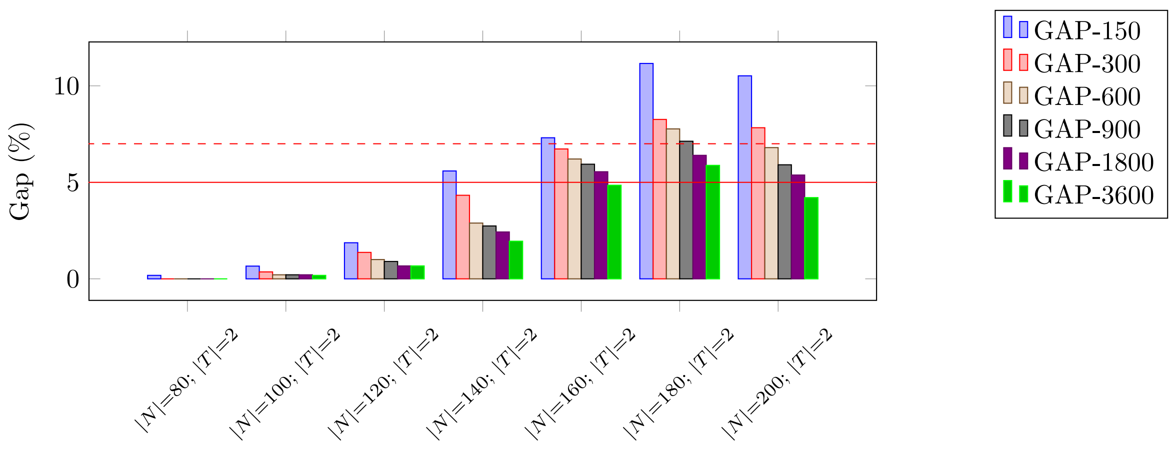
Slijepcevic, S. and Potkonjak, M. (2001). Power efficient organization of wireless sensor networks. In *ICC 2001. IEEE International Conference on Communications. Conference Record (Cat. No.01CH37240)*, volume 2, pages 472–476 vol.2.

Wang, X., Wu, X., and Zhang, X. (2017). Optimizing opportunistic routing in asynchronous wireless sensor networks. *IEEE Communications Letters*, **21**(10), 2302–2305.

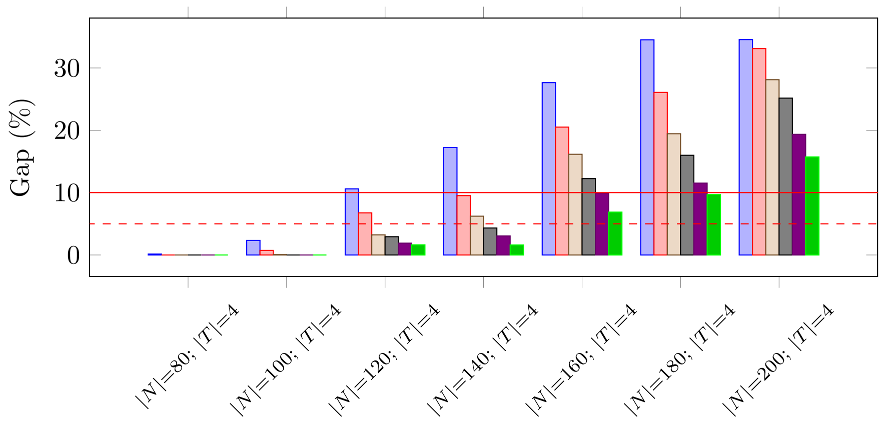
Yardibi, T. and Karasan, E. (2010). A distributed activity scheduling algorithm for wireless sensor networks with partial coverage. *Wireless Networks*, **16**(1), 213–225.

Zhang, X., Tao, L., Yan, F., and Sung, D. K. (2020). Shortest-Latency opportunistic routing in asynchronous wireless sensor networks with independent Duty-Cycling. *IEEE Transactions on Mobile Computing*, **19**(3), 711–723.

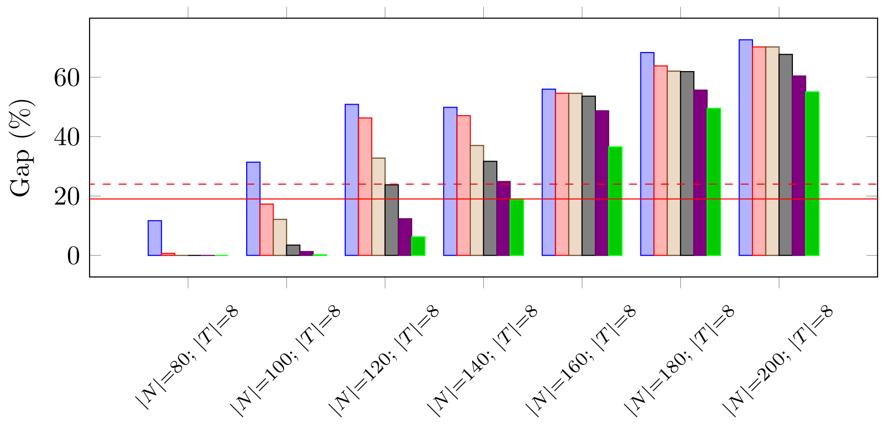
Figure 6: Average percentage optimality gap at computing time equal to 150, 300, 600, 900, 1800, or 3600 sec.



(a) $|T| = 2$



(b) $|T| = 4$



(c) $|T| = 8$