Prolonging Lifetime in Wireless Sensor Networks with Interference Constraints

Francesco Carrabs, Raffaele Cerulli, Ciriaco D'Ambrosio, and Andrea Raiconi

Department of Mathematics, University of Salerno
Via Giovanni Paolo II 132, 84084 Fisciano (SA), Italy
{fcarrabs,raffaele,cdambrosio,araiconi}@unisa.it

Abstract. In this work, we consider a scenario in which we have to monitor some locations of interest in a geographical area by means of a wireless sensor network. Our aim is to keep the network operational for as long as possible, while preventing certain sensors from being active simultaneously, since they would interfere with one another causing data loss, need for retransmissions and overall affecting the throughput and efficiency of the network. We propose an exact approach based on column generation, as well as a heuristic algorithm to solve its separation problem. Computational tests prove our approach to be effective, and that the introduction of our heuristic in the Column Generation framework allows significant gains in terms of required computational effort.

Keywords: Wireless Sensor Network, Column Generation, Maximum Lifetime, Interference Constraints, Greedy Algorithm

1 Introduction

Wireless Sensor Networks (WSNs) represent nowadays an ubiquitous technology. The rapid advancements in the technologies involved in the production of cost-effective embedded hardware greatly contributed to this trend, enabling the development of new related concepts such as the Internet of Things [1] and flying drones [9], and also raising concerns about the security of wireless communications ([15],[16],[17]). Among their many applications, monitoring and surveillance are very relevant in several contexts, with applications varying from natural disasters forecasting to personal safety and city management ([5],[18],[20]). One may consider, for instance, monitoring of environmental parameters such as temperature, sound, soil movements or pollution levels, or surveillance and motion detection in urban, military or domestic environments.

One of the main drawbacks is represented by the batteries installed in the individual sensor devices, which due to cost and size reasons have often a limited capacity. Hence, finding means to efficiently coordinate their activity to prolong the amount of time over which the network can remain operational is a crucial issue. To this end, many scientific efforts have been focused on proposing solutions for the *Maximum Network Lifetime Problem* (MLP) and variants. The aim of the problem is to individuate multiple and possibly overlapping sensor subsets

(*covers*) that can individually provide coverage for all the considered points of interest (*targets*), as well as an appropriate amount of activation time for each of them. Since each cover is able to provide the necessary coverage, all sensors that do not belong to it can be kept in a battery-preserving idle state while it is active. If a sensor belongs to more than one cover, the sum of their chosen activation times should not exceed the battery lifetime of the sensor. Under these assumptions, a continuous monitoring of the region of interest can be performed by activating covers one by one, and the overall network lifetime is given by the sum of the individual activation times. MLP was proven to be NP-Complete in [7], in which the authors also show that considering overlapping covers can greatly improve the optimal solution with respect to disjoint ones.

The main issue to be faced to solve a Maximum Network Lifetime problem is given by the number covers, which can be exponential with respect to the size of the considered instance (theoretically, any subset of the set of sensors may be a cover). The Column Generation (CG) technique has proved to be an effective tool to tackle this issue. This technique decomposes the problem in two components, a so-called *master problem* that only considers a subset of covers and assigns appropriate lifetime durations to each of them, and a *separation problem* whose aim is to produce new promising covers to be used by the master problem. A CG approach for the classical version of the problem was proposed by [21]. Since the separation problem is a hard problem itself, the author also proposed a heuristic algorithm to face it.

As mentioned, several variants of the problem have been proposed as well, to adapt it to different application contexts. For instance, there are works that consider cases in which not all targets must be continuously covered, but rather a given percentage of them ([12],[23],[30]), sensors that can adjust their radii to find optimal trade-offs between coverage and energy consumption ([8],[19],[22], [28]), heterogeneous networks with multiple sensor types ([3],[10]), and works considering connectivity issues in order to transmit the collected information to a central processing unit ([2],[6],[14],[24],[27],[31],[11],[13]).

Another relevant issue in the WSNs literature is related to interferences among the sensors of the network. The sudden increase in the number of connected devices has raised concern on this topic, due for instance to the coexistence of different wireless technologies operating in the same frequency band [4]. Even within sensors of the same type, concurrent transmission of sensors that are close enough to interfere may limit the network throughput and cause additional energy expense due to collisions, which cause data loss and hence require retransmissions [26]. Some works attempt to produce a connected topology that minimizes the interferences ([25],[26],[29]).

In this work we address the problem of maximizing the network lifetime on a wireless sensor network with interference constraints. In this variant of the problem, we are given a collection of pairs of *conflicting* sensors. Two sensors are said to be conflicting if each of them should be in idle mode while the other is in sensing mode, since they would generate interference if used together. It follows that by allowing covers to contain at most a sensor for each conflicting

pair, we are able to find the maximum network lifetime that can be obtained without interferences.

To solve the problem, we propose a Column Generation exact approach. An appropriately designed heuristic algorithm to speed up the separation problem resolution is also proposed.

It should be noted that our definition of conflicting sensors is perfectly fit to model the case in which sensors interfere with each other because they are too close; in this case, there will be a conflicting pair for each couple of sensors that are contained in each other's sensing or transmission range (in the following, this conflict-defining range will be called *conflict range*) . This will also be our assumption in the computational tests carried out for this work. However, it is also general enough to be adaptable to other interference definitions, or any other cases in which specific sensor subsets should not be used together.

The rest of the work is organized as follows. In Section 2 we formally define the problem. Section 3 describes in detail the proposed algorithm. Section 4 summarizes the results of our computational tests. Finally, Section 5 contains some final remarks.

2 Problem definition

Let $W = \{S, T, \tau, \delta, \gamma\}$ be a wireless sensor network, where:

- $S = \{s_1, \dots s_n\}$ is the set of sensors (let $|S| = n$);
- $T = \{t_1, \dots t_m\}$ is the set of targets (let $|T| = m$);
- $\tau : S \to \mathbb{R}^+$ is a function assigning to each sensor the maximum amount of time for which it can be kept in active (sensing) mode before exhausting its battery. In the following, we will use the notation τ_i to refer to $\tau(s_i)$;
- $\delta : S \times T \to \{0, 1\}$ is a function such that $\delta(s_i, t_j)$ is equal to 1 if sensor t_j is located within the sensing range of sensor s_i and can therefore be monitored by it (in which case we say that s_i *covers* t_j), and 0 otherwise. In the following, we will use the notation δ_{ij} to refer to $\delta(s_i, t_j)$;
- $\gamma : S \times S \to \{0, 1\}$ is a function such that $\gamma(s_i, s_j)$ is equal to 1 if sensors s_i and s_j are in conflict, and therefore at most one of them can be active in any given time instant, and 0 otherwise. In the following, we will use the notation γ_{ij} to refer to $\gamma(s_i, s_j)$. By definition, $\gamma_{ij} = \gamma_{ji}\ \forall (s_i, s_j) \in S \times S$.

A cover $C \subseteq S$ is a subset of sensors satisfying the following two properties:

1. $\sum_{s_i \in C} \delta_{ij} \geq 1\ \forall t_j \in T$. That is, each target of the network can be monitored by at least a sensor belonging to the cover;
2. $\gamma_{ij} = 0\ \forall (s_i, s_j) \in C \times C$. That is, the cover contains at most one sensor for each conflicting pair.

It follows that, by activating all sensors in C while keeping those in $S \setminus C$ in idle mode, the whole set of targets can be monitored without generating interference.

The Maximum Lifetime problem with Interferences Constraints (MLIC) requires to find a collection of pairs $(C_1, w_1), \ldots, (C_z, w_z)$, where each C_k is a cover and the related $w_k \geq 0$ is a value representing the *activation time* assigned to it, such that:

1. $\sum_{C_k : s_i \in C_k} w_k \leq \tau_i \; \forall s_i \in S$. That is, each sensor is not scheduled to be active for an amount of time that exceeds its battery capacity;
2. $\sum_{k=1}^{z} w_k$ (that is, the overall network lifetime) is maximized.

Assuming to be able to identify and enumerate all covers $C_1, \ldots, C_\ell$, the problem is described by the following linear programming mathematical model:

$$[\mathbf{P}] \max \sum_{k=1}^{\ell} w_k \tag{1}$$

$$\sum_{C_k : s_i \in C_k} w_k \leq \tau_i \qquad \forall s_i \in S \tag{2}$$

$$w_k \geq 0 \qquad \forall k = 1, \ldots, \ell \tag{3}$$

The model contains one constraint (row) for each sensor of the network, and one variable (column) for each cover; each variable w_k has coefficient 1 in the ith constraint if $s_i \in C_k$, and is not in the constraint otherwise.

It is easy to see that the objective function (1) and constraints (2) correspond to the lifetime maximization and to the enforcement of battery capacity constraints, respectively.

However, as mentioned in Section 1, the number of covers (and hence, the number of variables) is potentially exponential, therefore using an optimization software such as IBM ILOG CPLEX to find the optimal solution of this model is not generally possible in practice. For this reason, we developed the Column Generation approach presented in next section.

3 Column Generation approach

Our CG approach starts by finding an optimal solution for a variant of the **[P]** formulation that only considers a subset of all covers, called the master problem or **[MP]** from now on. Ideally, in order to understand if this solution is optimal for the original problem as well, one should compute all reduced costs for the variables that were not taken into account by **[MP]**.

In order to avoid to explicitly consider all these variables, we instead define and solve another problem (called separation problem) whose aim is to build the column (representing, in our case, a cover) of the variable with minimum reduced cost. If this reduced cost is negative, the related variable is then introduced into **[MP]**, since it may improve the current solution (we define such a variable and the related cover to be *attractive*), and **[MP]** is solved again. This procedure is iterated until the optimal solution for the separation problem corresponds to

Algorithm 1: GreedySP

Input: Wireless network, dual prices vector $\underline{\pi}$;
Output: Attractive cover C or failure;

```
C ← ∅;
S_c ← S;
T_u ← T;
ω_c ← updateWeights(S_c);
while |T_u| > 0 do
    if S_c == ∅ then
        return failure;
    s ← ChooseBest(S_c, ω_c);
    C ← C ∪ {s};
    T_u ← RemoveCovered(s);
    S_c ← RemoveConflicts(s);
    S_c ← RemoveUseless(T_u);
    ω_c ← updateWeights(S_c);
if Attractive(C) then
    return C;
else
    return failure;
```

a cover with a nonnegative reduced cost. Indeed, in this case, the last optimal solution found for [**MP**] is also optimal for the original problem [**P**].

Let π_i, $\forall s_i \in S$, be the dual prices of the sensors associated to the optimal solution of an instance of [**MP**]. The separation problem requires to find the cover C_k that minimizes the quantity $\sum_{s_i \in C_k} \pi_i - c_k = \sum_{s_i \in C_k} \pi_i - 1$, since the coefficient c_k of each variable w_k in the objective function (1) of [**P**] is 1. Being a set cover problem, the separation problem is NP-Hard. Moreover, it can be noted that any attractive cover (not necessarily the optimal one) can improve the current solution. For these reasons, we propose a greedy heuristic (called GreedySP) to solve this problem, which is described in Section 3.1.

However, whenever GreedySP fails (meaning that it either does not find a cover, or does not find an attractive one) there is no guarantee that the optimality condition has been reached. Therefore, we also propose an integer programming formulation called [**SP**] to solve the separation problem to optimality. In more detail, after solving [**MP**], in each CG iteration the procedure first solves the separation problem heuristically using our greedy algorithm. If it succeeds in finding an attractive cover, the cover is added to the set considered by [**MP**], and the current CG iteration ends. Otherwise, [**SP**] is used to solve the separation problem to optimality. The [**SP**] model is presented in Section 3.2.

As will be shown in Section 4, our algorithm is an effective method to solve the MLIC problem. On the one hand, it allows to solve the problem optimally; on the other hand, the introduction of GreedySP allows to significantly reduce the required computational effort.

3.1 Heuristic for the separation problem

The pseudocode for GreedySP is reported in Algorithm 1. The algorithm takes as input the wireless sensor network and the dual prices vector $\underline{\pi}$. In the first step, the algorithm initializes the sets C, S_c, T_u and the vector $\underline{\omega_c}$. The C set represents the cover that GreedySP is attempting to build, and therefore it is initialized with the empty set. The S_c set contains the candidate sensors, that is, sensors that do not belong to C but are not in conflict with any element of C, and is initialized with the S set. The T_u set contains the uncovered targets, meaning that they are not within the sensing range of any sensor in C; therefore, when C is empty, T_u is equal to T. The *updateWeights* function attributes a weight to each element of S_c. The weights are computed as follows:

$$\omega_c(s_i) = \frac{\pi_i}{\sum_{t_j \in T_u} \delta_{ij}} \qquad \forall s_i \in S_c \tag{4}$$

The weighting function balances the dual price of each candidate sensor and the number of uncovered targets that are covered by it. Since we are interested in minimizing the sum of the dual prices belonging to the cover, and selecting sensors that cover many targets may lead to needing less of them in C, it follows that the smaller is $\omega_c(s_i)$, the better is the sensor evaluation. Note that the weight is undefined if s_i does not cover any target in T_u; we assume this to never be the case in the initialization phase, since it would mean that the sensor is useless (no target in T is within its sensing range) and therefore it should be removed in preprocessing.

The while loop adds new sensors to C one by one, and iterates until there are no uncovered targets, meaning that C is indeed a cover. Inside the loop, we first check whether S_c is empty, in which case C cannot be completed and the algorithm ends, reporting a failure. If S_c is not empty, the *ChooseBest* function selects its minimum-weight element s and removes it from S_c. The new sensor is added to C, and all targets that are covered by it are removed by T_u. The *RemoveConflicts* and *RemoveUseless* functions update S_c by removing all sensors that cannot be chosen in future iterations since they are either in conflict with the new sensor, or do not cover uncovered targets any more. Finally, since the T_u set was updated, the $\underline{\omega_c}$ weights are recomputed for the remaining elements of S_c.

At the end of the while loop, as mentioned, the set C constitutes a cover. However, in order to understand if it is an attractive one, we check whether $\sum_{s_i \in C} \pi_i < 1$. If C is attractive, it is retuned by GreedySP and the current CG iteration ends, otherwise the heuristic reports a failure.

3.2 Integer Programming Formulation for the separation problem

The following formulation describes the separation problem:

$$[\mathbf{SP}] \min \sum_{s_i \in S} \pi_i y_i \tag{5}$$

$$\sum_{s_i \in S : \delta_{ij} = 1} y_i \geq 1 \qquad \forall t_j \in T \tag{6}$$

$$y_i + y_j \leq 1 \qquad \forall (s_i, s_j) \in S \times S : \gamma_{ij} = 1, i < j \tag{7}$$

$$y_i \in \{0, 1\} \qquad \forall s_i \in S \tag{8}$$

For each sensor $s_i \in S$, the binary variable y_i will be equal to 1 if the sensor is chosen to be part of the cover, 0 otherwise. The objective function (5) minimizes the sum of the dual prices of the chosen sensors. Constraints (6) impose for each target t_j the selection of at least a sensor among those that can cover it, while constraints (7) make sure that at most a sensor is selected for each conflicting pair.

As mentioned, **[SP]** is used within the CG framework to solve the subproblem to optimality whenever GreedySP fails. If the objective function value is greater than or equal to 1, the optimality condition has been reached and CG ends, otherwise the cover found by the model is attractive and CG must be iterated again.

3.3 Initializing the CG method

We need a procedure to build the set of covers considered by **[MP]** in the first iteration of the CG method. In our algorithm, we initialize this set with a single cover, produced by a variant of GreedySP. Since in this case we do not have dual prices associated to the sensors, the weights in this variant are computed as follows:

$$\omega_c(s_i) = \frac{\sum_{s_k \in S} \gamma_{ki}}{\sum_{t_j \in T_u} \delta_{ij}} \qquad \forall s_i \in S_c \tag{9}$$

That is, in this case the weighting function balances the number of conflicts of the sensor in the network and the number of uncovered targets that are covered by it. Note that, since we do not have dual prices, the concept of attractive cover is not defined for this initialization step, therefore if GreedySP produces a cover, it is always returned and used to initialize **[MP]**. If GreedySP fails to build a cover, we use the **[SP]** formulation with random dual price values to initialize **[MP]**.

4 Computational Results

In this section, in order to evaluate the effectiveness of our CG approach that uses GreedySP (called CG+GSP from now on), we compare it with a version that

always solves the separation problem exactly by means of the [**SP**] formulation. In the following, we will refer to this second version with the name SimpleCG.

Our algorithms were coded in C++, and the tests were executed on a Linux platform with an Intel Xeon E5-2650 CPU running at 2.30GHz and 128 GB of RAM. The mathematical formulations embedded in the CG framework were solved using the Concert library of IBM ILOG CPLEX 12.6.1. All tests were run in single thread mode, and we fixed a time limit equal to 1 hour for each test. When a certified optimal solution is not found within this amount of time, the best solution found is returned and used for our comparisons. In these cases, a "*" symbol is added near the related values in Tables 1 and 2.

We considered two different groups of instances. The first group was proposed in [14] for a different variant of the maximum lifetime problem. In this work, the authors proposed instances with a number of sensors varying in the set $\{100, 200, 300, 400, 500\}$, and 15 or 30 targets. However, here we only considered instances with a number of sensors greater than or equal to 300, as the smaller ones resulted to be too easily solvable for the MLIC problem. All sensors have the same battery duration, normalized to 1 time unit. Sensors and targets are disposed in a square area with size 500×500. The sensing range (RS) is equal to either 100 or 125, meaning that any sensor with an euclidean distance within this value from a given sensor is considered to be covered by it. Additionally, for these instances we consider a conflict range (RC) equal to either 125 or 175, meaning that two sensors that are within this distance from each other form a conflicting pair. There are 4 different instances for each combination of parameters, leading to 96 instances.

In order to test our approach on more challenging instances, we propose a second group, generated similarly to the ones in [14]. The new set of instances have a number of sensors equal to either 750, 1000, or 1250, while all other parameters are the same as the previous group. Hence, the second group is composed of 96 instances as well.

The results of the comparison on the first group are reported in Table 1. The table contains 12 rows divided in two groups of six rows each, associated to the sensing ranges $RS = 100$ and $RS = 125$, respectively. The first two columns show the number of sensors ($|S|$) and targets ($|T|$) in the scenarios. The next 10 columns are divided in two groups associated to the conflict ranges $RC = 125$ and $RC = 175$, respectively. Under these columns, we report in each entry an average value related to a scenario composed of four instances with the same characteristics. In more detail, we report for each algorithm the lifetime (LF) rounded to the second decimal digit, the computational time ($Time$) in seconds and the percentage gap between these computational times (Gap), evaluated as $\frac{Time(SimpleCG)-Time(CG+GSP)}{Time(SimpleCG)}$. When the time gap between the two algorithms is lower then 5 seconds, we do not report its percentage value, since we consider it negligible.

On the scenarios with $RS = 100$ and $RC = 125$, CG+GSP is always faster than SimpleCG with a gap value that ranges from about 43% to 92%. However, both algorithms are very effective on these scenarios, since they are solved to

			RC=125					RC=175				
			CG+GSP		SimpleCG			CG+GSP		SimpleCG		
	$\|S\|$	$\|T\|$	LF	Time	LF	Time	Gap	LF	Time	LF	Time	Gap
RS=100	300	15	16.25	0.98	16.25	4.05		13.75	4.57	13.75	6.56	
	300	30	13.25	1.38	13.25	4.58		9.16	9.35	9.16	9.93	
	400	15	20.75	0.73	20.75	9.44	92.27%	18.25	12.11	18.25	19.58	38.15%
	400	30	19.00	5.41	19.00	13.35	59.48%	14.25	34.84	14.25	36.19	
	500	15	28.75	4.72	28.75	26.54	82.22%	25.50	42.36	25.50	55.12	23.15%
	500	30	26.00	20.49	26.00	35.92	42.96%	19.00	118.05	19.00	122.02	
RS=125	300	15	23.25	0.19	23.25	5.36	96.46%	23.25	3.89	23.25	11.40	65.88%
	300	30	18.25	0.36	18.25	5.39	93.32%	18.25	7.17	18.25	10.80	
	400	15	32.50	0.24	32.50	14.97	98.40%	32.50	5.59	32.50	36.80	84.81%
	400	30	27.25	0.31	27.25	16.77	98.15%	26.75	30.48	26.75	40.76	25.22%
	500	15	41.25	0.55	41.25	38.28	98.56%	41.25	35.52	41.25	90.10	60.58%
	500	30	38.00	3.63	38.00	40.78	91.10%	38.00	85.67	38.00	107.35	20.20%

Table 1. Comparison between CG+GSP and SimpleCG on the smaller instances.

optimality in less than a minute. By increasing the conflict range RC to 175, it can be observed that computational times increase for both algorithms. Indeed, while all scenarios are still optimally solved by them, the requested computational time grows up to about two minutes. On these instances, CG+GSP is again always faster than SimpleCG; however, percentage gaps are smaller, and we report them in just 2 out of 6 cases, in which this value ranges from 23% to 38%. This is due to GreedySP becoming less effective, due to the additional conflicts on these instances.

By considering the instances with sensing range $RS = 125$, the computational times increase for SimpleCG and decrease for CG+GSP. Indeed, on the scenarios with $RS = 125$ and $RC = 125$, the gap values are always higher than 90%, proving that GreedySP is highly effective in finding attractive covers in these denser instances. By increasing the conflict range to 175, we observe a reduction of the gap values with respect to the scenarios with $RC = 125$. However, they remain generally significant, up to 84.81%, and the difference between the computational times of the two algorithms is under 5 seconds only in 1 case out of 6.

These results suggest that the main factor affecting the complexity of the problem is the size of the conflict range (and hence the number of conflicts), and that CG+GSP appears to relevantly outperform SimpleCG. However, this dataset resulted to be too simple to draw definitive conclusions, since all scenarios were solved within 123 seconds. For these reasons, we generated the second group of instances with up to 1250 sensors.

The results on these new instances are reported in Table 2. As expected, the computational times are higher than the ones required to solve previous instances, and there are scenarios that are not solved within the time limit. In more detail, on the scenarios with $RS = 100$ and $RC = 125$, CG+GSP requires

			RC=125					RC=175				
			CG+GSP		SimpleCG			CG+GSP		SimpleCG		
	$\|S\|$	$\|T\|$	LF	Time	LF	Time	Gap	LF	Time	LF	Time	Gap
RS=100	750	15	70.25	37.46	70.25	429.89	91.29%	68.50	671.88	68.50	977.29	31.25%
	750	30	43.50	59.64	43.50	208.32	71.37%	43.00	719.42	43.00	729.30	1.35%
	1000	15	60.25	2.53	60.25	814.45	99.69%	60.25	962.01	60.25	1773.15	45.75%
	1000	30	52.50	26.90	52.50	668.10	95.97%	49.00	1704.59	49.00	1757.12	2.99%
	1250	15	89.75	15.95	89.75	1935.21	99.18%	85.91*	2274.69	79.84*	3400.97	33.12%
	1250	30	57.00	342.85	57.00	1307.43	73.78%	52.75	2365.43	52.75	2500.29	5.39%
RS=125	750	15	72.25	1.51	72.25	279.57	99.46%	72.25	117.82	72.25	701.84	83.21%
	750	30	56.50	2.53	56.50	257.09	99.02%	55.50	419.09	55.50	660.50	36.55%
	1000	15	96.75	3.85	96.75	940.66	99.59%	96.75	796.88	96.75	2338.06	65.92%
	1000	30	79.25	4.21	79.25	733.95	99.43%	79.00	1360.21	79.00	1958.42	30.55%
	1250	15	127.75	8.88	127.75	2393.13	99.63%	127.75	843.91	95.45*	3523.17	76.05%
	1250	30	68.25	4.99	68.25	1378.17	99.64%	68.25	2035.12	68.25	2569.43	20.79%

Table 2. Comparison between CG+GSP and SimpleCG on the larger instances

up to 342 seconds to solve the problem, while SimpleCG requires 1307 seconds in the worst case. Percentage gaps range from 71% to over 99%. By increasing the conflict range to 175, similarly to the results of Table 1, the gap values decrease and range from 1.35% to 45.75%. In the scenario with 1250 sensors and 15 targets, both algorithms fail to find all optimal solutions within the time limit. In more detail, this happens for a single instance of the scenario for CG+GSP and for 3 out of 4 instances for SimpleCG. As a consequence, the average solution found by CG+GSP (85.91) is significantly better than the one of SimpleCG (79.84).

On the scenarios with $RS = 125$ and $RC = 125$ the percentage gaps are always higher than 99%. Indeed, CG+GSP is able to solve these scenarios within 10 seconds, while SimpleCG requires from 257 to 2393 seconds. Overall, for both the groups of instances, the combination of parameters $RS = 125, RC = 125$ appears to be the one where CG+GSP has the largest advantage over SimpleCG, since the smaller conflict ranges and higher sensing ranges allow GreedySP to find more attractive covers. On the the scenarios with $RS = 125$ and $RC = 175$, percentage gaps range from 20.79% and 83.21%. Again SimpleCG fails to optimally solve 3 instances in the scenario with 1250 sensors and 15 targets, while CG+GSP solves to optimality all of them. The average solution returned by SimpleCG for this scenario is equal to 95.45, while the optimal value provided by CG+GSP is significantly better, being equal to 127.75.

Overall, the results reported in Table 2 show that CG+GSP is often one order of magnitude faster than SimpleCG, with percentage time gaps higher than 65% for 15 out of 24 scenarios. Moreover, CG+GSP is more effective, with a single failure out of 96 instances, as opposed to the 6 failures of SimpleCG. Finally, in the single scenario in which both algorithms failed, CG+GSP returned a significantly better solution.

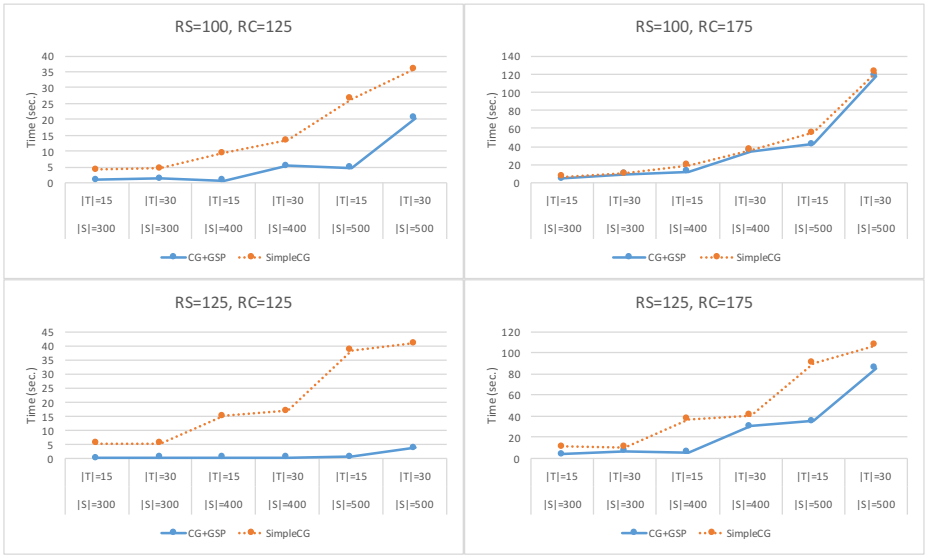

Fig. 1. Computational time for CG+GSP and SimpleCG, on instances with $|S| \leq 500$

Fig. 2. Computational time for CG+GSP and SimpleCG, on instances with $|S| \geq 750$

Figures 1 and 2 give a graphical representation of the computational time required by the two algorithms. Again, it can be noticed that the continuous line (representing CG+GSP) is always below the dotted line (SimpleCG), and that the gaps are particularly significant in the scenarios with RC=125.

5 Conclusions

In this work we addressed a novel variant of the maximum lifetime problem, that takes into account the interference issue. We proposed a column generation-based exact resolution approach, and developed an appropriately designed greedy heuristic to speed up the resolution of the separation problem. Computational results show that our approach is significantly better than a straightforward column generation implementation that does not take advantage of the heuristic, resulting faster on all the performed tests, often by a significant margin, and finding a larger number of optimal solutions.

References

1. A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash. Internet of things: A survey on enabling technologies, protocols and applications. *IEEE Communications Surveys & Tutorials*, 17(4), 2015.
2. A. Alfieri, A. Bianco, P. Brandimarte, and C. F. Chiasserini. Maximizing system lifetime in wireless sensor networks. *European Journal of Operational Research*, 181(1):390–402, 2007.
3. W. Awada and M. Cardei. Energy-efficient data gathering in heterogeneous wireless sensor networks. In *Proceedings of the IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, pages 53–60, 2006.
4. N. Azmi, L. Kamarudin, M. Mahmuddin, A. Zakaria, A. Shakaff, S. Khatun, K. Kamarudin, and M. Morshed. Interference issues and mitigation method in wsn 2.4ghz ism band: A survey. In *Electronic Design (ICED), 2nd International Conference on*, pages 403–408, 2014.
5. L. Bianco, C. Cerrone, R. Cerulli, and M. Gentili. Locating sensors to observe network arc flows: Exact and heuristic approaches. *Computers & Operations Research*, 46:12–22, 2014.
6. I. Cardei and M. Cardei. Energy-efficient connected-coverage in wireless sensor networks. *International Journal of Sensor Networks*, 3(3):201–210, 2008.
7. M. Cardei, M. T. Thai, Y. Li, and W. Wu. Energy-efficient target coverage in wireless sensor networks. In *Proceedings of the 24th conference of the IEEE Communications Society*, volume 3, pages 1976–1984, 2005.
8. M. Cardei, J. Wu, and M. Lu. Improving network lifetime using sensors with adjustable sensing ranges. *International Journal of Sensor Networks*, 1(1-2):41–49, 2006.
9. F. Carrabs, C. Cerrone, R. Cerulli, and M. Gaudioso. A novel discretization scheme for the close enough traveling salesman problem. *Computers & Operations Research*, 78:163–171, 2017.
10. F. Carrabs, R. Cerulli, C. D'Ambrosio, M. Gentili, and A. Raiconi. Maximizing lifetime in wireless sensor networks with multiple sensor families. *Computers & Operations Research*, 60:121–137, 2015.

11. F. Carrabs, R. Cerulli, C. D'Ambrosio, and A. Raiconi. An exact algorithm to extend lifetime through roles allocation in sensor networks with connectivity constraints. To appear in Optimization Letters. DOI: 10.1007/s11590-016-1072-y.
12. F. Carrabs, R. Cerulli, C. D'Ambrosio, and A. Raiconi. A hybrid exact approach for maximizing lifetime in sensor networks with complete and partial coverage constraints. *Journal of Network and Computer Applications*, 58:12–22, 2015.
13. F. Carrabs, R. Cerulli, C. D'Ambrosio, and A. Raiconi. Extending lifetime through partial coverage and roles allocation in connectivity-constrained sensor networks. *IFAC-PapersOnline*, 49(12):973–978, 2016.
14. F. Castaño, A. Rossi, M. Sevaux, and N. Velasco. A column generation approach to extend lifetime in wireless sensor networks with coverage and connectivity constraints. *Computers & Operations Research*, 52(B):220–230, 2014.
15. A. Castiglione, P. D'Arco, A. De Santis, and R. Russo. Secure group communication schemes for dynamic heterogeneous distributed computing. To appear in Future Generation Computer Systems. http://dx.doi.org/10.1016/j.future.2015.11.026.
16. A. Castiglione, F. Palmieri, U. Fiore, A. Castiglione, and A. De Santis. Modeling energy-efficient secure communications in multi-mode wireless mobile devices. *Journal of Computer and System Sciences*, 81(8), 2015.
17. A. Castiglione, A. De Santis, A. Castiglione, F. Palmieri, and U. Fiore. An energy-aware framework for reliable and secure end-to-end ubiquitous data communications. In *Intelligent Networking and Collaborative Systems (INCoS), 5th International Conference on*, pages 157–165, 2013.
18. C. Cerrone, R. Cerulli, and M. Gentili. Vehicle-id sensor location for route flow recognition: Models and algorithms. *European Journal of Operational Research*, 247(2):618–629, 2015.
19. R. Cerulli, M. Gentili, and A. Raiconi. Maximizing lifetime and handling reliability in wireless sensor networks. *Networks*, 64(4):321–338, 2014.
20. D. Chen, Z. Liu, L. Wang, M. Dou, J. Chen, and H. Li. Natural disaster monitoring with wireless sensor networks: A case study of data-intensive applications upon low-cost scalable systems. *Mobile Networks and Applications*, 18(5):651–663, 2013.
21. K. Deschinkel. A column generation based heuristic for maximum lifetime coverage in wireless sensor networks. In *SENSORCOMM 11, 5th Int. Conf. on Sensor Technologies and Applications*, volume 4, pages 209 – 214, 2011.
22. A. Dhawan, C. T. Vu, A. Zelikovsky, Y. Li, and S. K. Prasad. Maximum lifetime of sensor networks with adjustable sensing range. In *Proceedings of the Seventh ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, pages 285 – 289, 2006.
23. M. Gentili and A. Raiconi. $\alpha-$coverage to extend network lifetime on wireless sensor networks. *Optimization Letters*, 7(1):157–172, 2013.
24. Y. Gu, Y. Ji, and B. Zhao. Maximize lifetime of heterogeneous wireless sensor networks with joint coverage and connectivity requirement. In *EmbeddedCom-09, 8th International Conference on Embedded Computing*, pages 226–231, 2009.
25. R.E.N. Moraes, C.C. Ribeiro, and G.M. Ribeiro. Exact formulations for the minimum interference problem in k-connected ad hoc wireless networks. *International Transactions in Operational Research*, 23(6):1113–1139, 2016.
26. T. Moscibroda and R. Wattenhofer. Minimizing interference in ad hoc and sensor networks. In *2nd ACM SIGACT/SIGMOBILE International Workshop on Foundations of Mobile Computing (DIALM-POMC)*, pages 24–33, 2005.

27. A. Raiconi and M. Gentili. Exact and metaheuristic approaches to extend lifetime and maintain connectivity in wireless sensors networks. In J. Pahl, T. Reiners, and S. Voss, editors, *Network Optimization*, volume 6701 of *Lecture Notes in Computer Science*, pages 607–619. Springer, Berlin/Heidelberg, 2011.
28. A. Rossi, A. Singh, and M. Sevaux. An exact approach for maximizing the lifetime of sensor networks with adjustable sensing ranges. *Computers & Operations Research*, 39(12):3166–3176, 2012.
29. H. Tan, T. Lou, Y. Wang, Q.-S. Hua, and F.C.M. Lau. Exact algorithms to minimize interference in wireless sensor networks. *Theoretical Computer Science*, 412(50):6913–6925, 2011.
30. C. Wang, M. T. Thai, Y. Li, F. Wang, and W. Wu. Minimum coverage breach and maximum network lifetime in wireless sensor networks. In *Proceedings of the IEEE Global Telecommunications Conference*, pages 1118–1123, 2007.
31. Q. Zhao and M. Gurusamy. Lifetime maximization for connected target coverage in wireless sensor networks. *IEEE/ACM Transactions on Networking*, 16(6):1378–1391, 2008.