Column Generation embedding Carousel Greedy for the Maximum Network Lifetime Problem with Interference Constraints

Francesco Carrabs, Carmine Cerrone, Ciriaco D'Ambrosio and Andrea Raiconi

Abstract We aim to maximize the operational time of a network of sensors, whose task is to monitor a predefined set of target locations. The classical approach proposed in the literature consists in individuating subsets of sensors (*covers*) that can individually monitor the targets, and in assigning appropriate activation times to each cover. Indeed, since sensors may belong to multiple covers, it is important to make sure that their overall battery capacities are not violated. We consider additional constraints that prohibit certain sensors to appear in the same cover, since they would interfere with each other. We propose a Column Generation approach, in which the separation problem is solved either exactly or heuristically by means of a recently introduced technique to enhance basic greedy algorithms, known as Carousel Greedy. Our experiments show the effectiveness of this approach.

1 Introduction

The Maximum Lifetime Problem (MLP) and its variants have been the focus of many studies in the last years. Given a geographical region in which some important target locations (or simply *targets*) have been individuated, the aim is to use a network of sensors for as long as possible to keep these locations under observation. A key concept is the one of *cover*; with this term, we refer to a subset of sensors that is independently able to monitor all targets. A common approach for facing MLP problems is the following: find a collection of covers, and activate them one at a time. By activating a cover, we mean that all of its sensors are switched to

Francesco Carrabs, Ciriaco D'Ambrosio, Andrea Raiconi
Department of Mathematics, University of Salerno, Via Giovanni Paolo II 132, 84084 Fisciano (SA), Italy e-mail: {fcarrabs,cdambrosio,araiconi}@unisa.it

Carmine Cerrone
Department of Biosciences and Territory, Contrada Fonte Lappone, 86090 Pesche (IS), Italy e-mail: carmine.cerrone@unimol.it

their sensing mode, while all other sensors are kept in an idle, energy-saving state. Clearly, if a sensor belongs to more than a cover, the overall activation times of these covers cannot exceed the maximum activation time imposed by the sensor battery.

Among the first works dealing with the MLP problem, we recall [3]. In this work, it is first shown that allowing non-disjoint covers can bring noticeable improvements in terms of network lifetime. The authors also demonstrate that the problem is NP-complete, and present LP-based and greedy heuristics. A resolution approach to solve MLP based on Column Generation was presented in [15]. Several MLP variants have been proposed and studied. In some works, sensors are allowed to enlarge their sensing radii at the expense of additional energy consumption ([13],[14],[19]). Other lines of research consider the case in which it is allowed to leave some targets uncovered ([8],[16],[20]), sensors belong to different types ([2],[5]), or connectivity issues are taken into account ([1],[10],[18],[6],[9]). In the context of MLP problem variants, fewer research efforts have considered interference issues. Concurrent transmission of sensors that are too close may cause data collision, which in turn is responsible for data loss and additional energy expense; see for instance [17]. In [7], the authors present the Maximum Lifetime Problem with Interference Constraints (MLIC). In this problem, a collection of pairs of *conflicting* sensors are considered. For each of these pairs, at most one sensor can belong to any given cover. The authors present a Column Generation algorithm, whose separation problem is solved either exactly or heuristically by means of a greedy heuristic. In this work, we modify the algorithm presented in [7] by facing the heuristic resolution of the separation problem through Carousel Greedy, a novel paradigm for enhancing greedy heuristics, originally proposed in [12]. As will be shown in our computational tests, improving this component of the Column Generation scheme brings noticeable improvements in the efficiency of the whole algorithm. The next sections of the paper illustrate the general Column Generation approach for MLIC (Section 2), the proposed Carousel Greedy procedure for the separation problem (Section 3), the results of our computational tests (Section 4) and our final remarks (Section 5).

2 Column Generation approach

Let $S = \{s_1, \dots, s_m\}$ be the set of the sensors, and $T = \{t_1, \dots, t_m\}$ be the target points. A cover C_k for the MLIC problem is a subset of S, such that each target $t_j \in T$ is within the sensing area of at least one sensor $s_i \in C_k$ (it is *monitored*, or *covered* by it), and such that every couple of sensors $(s_i, s_j) \in C_k \times C_k$ is not a conflicting pair.

The overall number of covers can be exponential in size. Hence, in [7] a Column Generation (ColGen) approach was proposed in order to implicitly discard most of them.

Let $\mathscr{C} = \{C_1, \dots, C_z\}$ be a set composed by some feasible covers for MLIC. In [7], the master problem of the ColGen approach for the MLIC problem is formulated as follows:

$$\textbf{[MP]} \max \sum_{C_k \in \mathscr{C}} w_k \tag{1}$$

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

$$w_k \ge 0 \qquad \forall C_k \in \mathscr{C} \tag{3}$$

Variables w_k represent for how long each cover $C_k \in \mathscr{C}$ is kept in active state in the solution, while parameter τ_i $\forall s_i \in S$ represents the maximum amount of activation time available for the sensor, given its battery capacity. The column of each variable w_k in the coefficient matrix contains a 1 in the i-th position if $s_i \in C_k$, 0 otherwise. It is then clear that the optimal **[MP]** solution represents the maximum lifetime that can be obtained by considering the subset of covers contained in $\mathscr{C}$, while respecting the battery duration constraints. In order to evaluate whether this is also the optimal solution for the whole problem, we need to solve a separation problem to identify the nonbasic variable with minimum reduced cost. The separation problem formulation, also presented in [7], is the following:

$$\textbf{[SP]} \min \sum_{s_i \in S} \pi_i y_i \tag{4}$$

$$\sum_{s_i \in S: \delta_{ij} = 1} y_i \ge 1 \qquad \forall t_j \in T \tag{5}$$

$$y_i + y_j \le 1 \qquad \forall (s_i, s_j) \in S \times S : \gamma_{ij} = 1, i < j \tag{6}$$

$$y_i \in \{0, 1\} \qquad \forall s_i \in S \tag{7}$$

The separation problem needs to build the column associated to a cover, hence each variable y_i $\forall s_i \in S$ will be equal to 1 if the sensor is chosen to belong to it, and 0 otherwise. Parameters π_i, weighting the y_i variables in the objective function, represent the dual prices associated to the sensors after solving **[MP]**. Each parameter δ_{ij} is equal to 1 is sensor $s_i \in S$ can monitor target $t_j \in T$ and 0 otherwise. Finally, each parameter γ_{ij} $\forall (i, j) \in S \times S$ is equal to 1 if (s_i, s_j) is a conflicting pair and 0 otherwise. Hence constraints (5) and (6) make sure that the chosen sensors define a cover, while the objective function minimizes its reduced cost. In particular, if $\sum_{s_i \in C} \pi_i - 1 \ge 0$ for the newly built cover C, then the solution found by **[MP]** was optimal for the MLIC problem; otherwise, C could potentially be used to find a better solution (it is defined to be an *attractive* cover). Hence, C is added to the set $\mathscr{C}$ and the ColGen algorithm iterates.

The main drawback of this approach is that the separation problem is NP-Hard itself. Hence, in [7], a greedy heuristic for the separation problem is presented. In this work we enhance this heuristic by trasforming it into a Carousel Greedy, which is presented in next section.

3 Carousel Greedy approach for the separation problem

The Carousel Greedy is a generalized method to enhance greedy algorithms, originally proposed in [12]. The aim is to obtain a procedure which is almost as fast and simple as the greedy procedure on which it is based, while achieving accuracy levels similar to those of a metaheuristic. The authors show the effectiveness of their proposal for several classical combinatorial optimization problems. The main observation underlying Carousel Greedy is that during the execution of a constructive heuristic, the later decisions are likely to be more informed and valid than the earlier ones. Indeed, wrong decisions taken in the first stages may be the cause of poor overall performances. Given this observation, a Carousel Greedy procedure increases the solution space visited by a basic greedy, operating in three main steps:

- In the first step a partial (unfeasible) solution is built. The first step ends when the partial solution size reaches a given percentage of the size of a complete (feasible) solution.
- In the second step, the partial solution is modified by iteratively removing from it the oldest choices and making new ones. The second step ends after a pre-defined number of iterations.
- In the final step, the partial solution is completed to produce a feasible solution.

Our proposed Carousel-SP algorithm enhances Greedy-SP, a constructive heuristic presented in [7] to solve the MLIC separation problem. Carousel-SP works as follows:

- The first step starts from an empty set C, and iteratively adds sensors to it. The sensors are chosen from a set of candidates S_c, initialized with S. At each iteration, the algorithm uses a greedy criterion to select the next sensor to be added to C. In more detail, it selects the sensor $s_i \in S_c$ that minimizes the quantity $\omega_i = \frac{\pi_i}{|T_i|}$, where π_i is the dual price of the sensor and $|T_i|$ the amount of additional targets that would be monitored by C by adding s_i to it. The greedy criterion is designed to favor sensors with low dual prices that cover many targets. After adding s_i to C, the elements of S_c that would form a conflicting pair with s_i (as well as s_i itself) are removed from S_c. Indeed, adding one of these sensors to C would violate constraints (6). The first step ends as soon as the number of uncovered targets is equal to or lower than $\beta|T|$, with $\beta \in [0,1]$. If S_c becomes empty first, Carousel-SP ends in failure.
- In each iteration of the second step, the sensor in C corresponding to the oldest choice is removed from it, and is replaced with a new one. After the removal of a sensor s_i from C, the set S_c is updated to contain s_i itself and any sensor that was removed because it was in conflict with s_i. Since some new targets may become uncovered after the s_i removal, the ω_i values are updated as well. Each new sensor added to C is selected according to the same greedy criterion used for the first step. Eventually, after one or more iterations of the second step, C may become a feasible solution (that is, a cover); if this is the case, the feasible solution with the lowest objective function value ($\sum_{s_i \in C} \pi_i$) is stored as incumbent

solution C'. Furthermore, after having found a cover, two sensors instead than one are removed from C in the following iteration, to avoid cycling on the same solution. The second step is iterated αh times, with $\alpha \geq 1$ and $h = |C|$ at the end of the first step. After the last iteration, if an incumbent solution C' exists, it is returned and Carousel-SP ends its execution.

- The third step operates similarly to the first one, with two differences; it starts from the C set returned by the second step, and it iterates until all targets are covered. As soon as C becomes a feasible solution, it is returned, and Carousel-SP ends. If S_c becomes empty first, Carousel-SP ends in failure.

We now discuss how Carousel-SP is integrated within the ColGen framework. In each iteration, after solving **[MP]**, we attempt to solve the separation problem heuristically using Carousel-SP. If it returns an attractive cover C, it is added to $\mathscr{C}$ and the current ColGen iteration ends. Otherwise, if Carousel-SP fails during the first or the third step or if returns an unattractive cover, we need to solve the separation problem exactly using **[SP]**. Again, if an attractive cover is found, it is added to $\mathscr{C}$ and a new ColGen iteration begins. Otherwise, the optimality for the MLIC problem of the last solution found by **[MP]** has been proven.

4 Computational Results

In this section we compare our approach (CG+C), that embeds Carousel-SP, with the ColGen algorithm (CG+G) proposed in [7], which uses Greedy-SP. Both CG+G and CG+C were coded in C++. All tests were run on a Linux machine with an Intel Xeon E5-2650 CPU running at 2.30GHz and 128 GB of RAM. For both the approaches, the Concert library of IBM ILOG CPLEX 12.6.1 was used to solve the mathematical formulations. Tests were run in single thread mode, with a time limit of 1 hour for each test. We considered the same set of instances used in [7], with a number of sensors varying in the set $\{300, 400, 500, 750, 1000, 1250\}$, and either 15 or 30 targets. All sensors have the a battery duration normalized to 1 time unit. Sensors and targets are disposed in a square area with size 500×500. Each sensor has a *sensing* (RS) and a *conflict* (RC) range. RS is equal to either 100 or 125; targets with an euclidean distance within this value from a sensor are covered by it. RC is equal to 175; two sensors within this distance from each other form a conflicting pair. There are 4 different instances for each combination of parameters, for a total of 96 instances. Note that the computational test carried out in [7] also considered the case $RC = 125$. However, the authors have shown that these instances are usually very easily solved; in particular, for $RS = 125$, the number of separation problems solved to optimality is often equal to 1. In this context, it is pointless to apply Carousel-SP, since it is more expensive than Greedy-SP and there are not margins to speed up the convergence of the ColGen algorithm. Therefore, we report in the following the computational results only for $RC = 175$. Regarding the Carousel-SP parameters, after a preliminary test phase values $\alpha = 3$ and $\beta = 0.2$ were chosen.

	$\lvert S\rvert$	$\lvert T\rvert$	CG+G LF	CG+G Time	CG+G SubInv	CG+C LF	CG+C Time	CG+C SubInv	Gap(%)
RS=100	300	15	13.75	4.57	75.50	13.75	4.66	74.25	
	300	30	9.16	9.35	72.25	9.16	10.10	74.25	
	400	15	18.25	12.11	84.75	18.25	12.71	89.25	
	400	30	14.25	34.84	127.25	14.25	33.32	123.50	4.36%
	500	15	25.50	42.36	177.75	25.50	39.20	164.25	7.46%
	500	30	19.00	118.05	204.25	19.00	121.56	200.75	-2.97%
RS=125	300	15	23.25	3.89	54.00	23.25	3.29	39.25	
	300	30	18.25	7.17	83.25	18.25	7.25	76.25	
	400	15	32.50	5.59	32.75	32.50	6.83	29.50	-22.18%
	400	30	26.75	30.48	150.50	26.75	28.95	138.75	5.02%
	500	15	41.25	35.52	122.00	41.25	29.23	87.50	17.71%
	500	30	38.00	85.67	228.25	38.00	88.24	226.25	-3.00%

Table 1 Computational results on the small instances with RC=175.

The results of the comparison between the two algorithms on the smaller instances ($|S| \leq 500$) are reported in Table 1, containing 12 rows split in 2 groups of 6 rows each, associated to $RS = 100$ and $RS = 125$, respectively. The first two columns show the number of sensors ($|S|$) and targets ($|T|$) in the scenarios. The next 6 columns report, for each algorithm, the lifetime rounded to 2 decimal digits (LF), the computational time in seconds ($Time$) and the number of separation problems solved to optimality ($SubInv$), respectively. Each entry in these columns is an average value for the 4 instances of a given scenario. Finally, the last column shows the percentage gap (Gap) between the computational times, evaluated as $\frac{Time(CG+G)-Time(CG+C)}{Time(CG+G)}$. When the gap is lower than 1 second, we consider it negligible and therefore we do not report its percentage value. All the small scenarios are solved to optimality by both algorithms, and hence we only focus on performances. On 5 scenarios, the time gap is negligible. On the remaining scenarios, CG+C is faster than CG+G 4 times, with a percentage gap that ranges from 4.36% to 17.71%, and is slower 3 times, with a percentage gap ranging from 2.97% to 22.18%. Regardless of percentage gaps, the performances of the two algorithms are very close for all these scenarios, since the time gap is always lower than 7 seconds.

The results obtained on the larger scenarios are more interesting. These results are reported in Table 2. As expected, the computational times of CG+C and CG+G are higher than the ones required to solve small scenarios, and there is a scenario (marked with a "*" symbol) that is not solved within the time limit by CG+G ($|S| = 1250$, $|T| = 15$ and $RS = 100$). The scenario is, instead, solved to optimality by CG+C in around half an hour. CG+C results only once slower than CG+G, on the scenario with $|S| = 1000$, $|T| = 30$ and $RS = 125$, with a percentage gap equal to 0.51%, corresponding to about 7 seconds. On all the remaining scenarios CG+C is

	$\lvert S\rvert$	$\lvert T\rvert$	CG+G LF	CG+G Time	CG+G SubInv	CG+C LF	CG+C Time	CG+C SubInv	Gap(%)
RS=100	750	15	68.50	671.88	454.50	68.50	582.77	426.25	13.26%
	750	30	43.00	719.42	421.25	43.00	696.33	419.50	3.21%
	1000	15	60.25	962.01	257.25	60.25	487.45	120.00	49.33%
	1000	30	49.00	1704.59	499.50	49.00	1610.24	481.25	5.54%
	1250	15	85.91*	2274.69	453.50	86.75	2014.62	425.25	11.43%
	1250	30	52.75	2365.43	406.50	52.75	2145.40	389.75	9.30%
RS=125	750	15	72.25	117.82	79.00	72.25	98.99	55.75	15.98%
	750	30	55.50	419.09	237.75	55.50	391.86	232.25	6.50%
	1000	15	96.75	796.88	239.75	96.75	660.31	181.25	17.14%
	1000	30	79.00	1360.21	406.50	79.00	1367.19	403.50	-0.51%
	1250	15	127.75	843.91	147.50	127.75	562.74	95.25	33.32%
	1250	30	68.25	2035.12	357.00	68.25	1707.32	321.75	16.11%

Table 2 Computational results on the large instances with RC=175.

faster than CG+G, with a percentage gap that ranges from 3.21% to 49.33% and a time gap up to about 500 seconds. In particular, for 7 out of 12 scenarios, CG+C results at least 10% faster than CG+G. It can be noted that the computational time is heavily affected by theSubInv value. For instance, on the scenario with $|S| = 1000$, $|T| = 15$ and $RS = 100$, the SubInv value and the computational time for CG+G are about twice greater than the values for CG+C. Similar observations can be done for the other scenarios. By providing better columns to the master problem, Carousel-SP reduces the SubInv value and speeds up the convergence of the ColGen approach.

5 Conclusions

We proposed a column generation algorithm to solve the Maximum Lifetime Problem with Interference Constraints. We improve a previous algorithm by introducing a new method to solve heuristically the separation problem, based on the Carousel Greedy paradigm. Computational tests show the effectiveness of our proposal, in particular for larger test instances. Further research will be focused on improving the Carousel Greedy procedure through hybridization with metaheuristic approaches, such as Tabu Search and Genetic ([4], [11]).

References

1. 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.

2. 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.
3. 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.
4. F. Carrabs, C. Cerrone, and R. Cerulli. A tabu search approach for the circle packing problem. In *Network-Based Information Systems (NBiS), 2014 17th International Conference on*, pages 165–171, 2014.
5. 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.
6. 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.
7. F. Carrabs, R. Cerulli, C. D'Ambrosio, and A. Raiconi. Prolonging lifetime in wireless sensor networks with interference constraints. Submitted to GPC 2017 - 12th International Conference on Green, Pervasive and Cloud Computing.
8. 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.
9. 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.
10. 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.
11. C. Cerrone, R. Cerulli, and M. Gaudioso. Omega one multi ethnic genetic approach. *Optimization Letters*, 10(2):309–324, 2016.
12. C. Cerrone, R. Cerulli, and B. Golden. Carousel greedy: A generalized greedy algorithm with applications in optimization. Submitted to Computers & Operations Research.
13. R. Cerulli, R. De Donato, and A. Raiconi. Exact and heuristic methods to maximize network lifetime in wireless sensor networks with adjustable sensing ranges. *European Journal of Operational Research*, 220(1):58–66, 2012.
14. R. Cerulli, M. Gentili, and A. Raiconi. Maximizing lifetime and handling reliability in wireless sensor networks. *Networks*, 64(4):321–338, 2014.
15. 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.
16. M. Gentili and A. Raiconi. $\alpha-$coverage to extend network lifetime on wireless sensor networks. *Optimization Letters*, 7(1):157–172, 2013.
17. 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.
18. 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.
19. 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.
20. 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.