Multi-cycle Coverage for Multi-robot Patrolling - application to data collection in WSNs -

Mihai-Ioan Popescu, Hervé Rivano, Olivier Simonin

University of Lyon, INSA Lyon, Inria, CITI, F-69621 Villeurbanne, France
`firstname.lastname@insa-lyon.fr`

Résumé : Patrolling is mainly used in situations where the need of repeatedly visiting certain places is critical. In this paper, we consider a deployment of a wireless sensor network (WSN) that cannot be fully meshed because of the distance or obstacles. Several robots are then in charge of getting close enough to the nodes in order to connect to them, and perform a patrol to collect all the data in time. We discuss the problem of multi-robot patrolling within the constrained wireless networking settings. We show that this is fundamentally a problem of vertex coverage with bounded simple cycles (CBSC). We offer a formalization of the CBSC problem and prove it is NP-hard and at least as hard as the Traveling Salesman Problem (TSP). Then, we provide and analyze heuristics relying on clusterings and geometric techniques. The performances of our solutions are assessed in regards to networking parameters, robot energy, but also to random and particular graph models.

1 INTRODUCTION

Multi-agent patrolling consists in organizing the continuous coverage of an area by several agents, e.g. drones, autonomous vehicles, etc.. Patrolling is predominately used in tasks and applications which require to repeatedly visit certain places. This is the case of coverage, surveillance (e.g. intruder and anomaly detection), data collection, or rescue operations (e.g. after natural disasters) (Rescue, 2001) (Daniel *et al.*, 2009). In this regard, each agent follows its own trajectory. The main objective in patrolling optimization is to minimize the time between two consecutive visits of the same place, called *idleness*. In the multi-agent case, the problem is known to be NP-hard, and for most of the scenarios, efficient solutions are periodic and the trajectories are cycles (Chevaleyre, 2004) (Glad *et al.*, 2008).

In this paper, we consider the problem of multi-robot patrolling constrained by maximum required idleness, with application to collection of data produced by a wireless sensor network (WSN), as in Figure 1. We assume that the WSN cannot be fully meshed because of the inter-node distance or because of obstacles. The challenge is therefore to provide solutions to the general problem of patrolling while taking into account performance metrics induced by the networking constraints.

Having agents patrolling a wireless sensor networks in order to collect the data has been extensively studied in the networking literature, agents being denoted *mobile sinks* (Tunca *et al.*, 2014). These solutions have been introduced for coping with poor connectivity among fixed sensors in difficult radio conditions. The main objective is to improve the network lifetime, which is related to the energy consumed by the nodes for communicating. While most of the literature focuses on networking issues such as address allocation and routing protocols, the mobility of a set of sinks (Marta & Cardei, 2009) and the optimization of their trajectories (Basagni *et al.*, 2008) have also yielded some interest. There is a consensus on the efficiency of using mobile sinks. In our work, we therefore abstract the networking issues into metrics and objectives for the multi-agent patrols with communication capabilities.

In this work, we express the problem of multi-agent patrolling in WSNs with respect to networking requirements and real life resource limits. We take into account issues like limited sensor memory, sensor throughput, data delivery time, limited robot energy and memory, and the finite number of robots. Our main contribution consists in offering, analyzing and testing solutions for the multi-robot patrolling, which respect the requirements of the network as well as robots' limitations, while trying to minimize the number

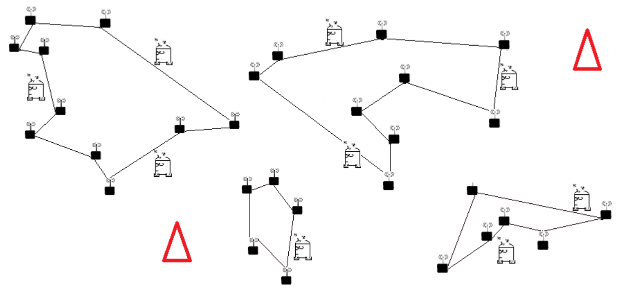
FIGURE 1 – Multi-agent patrolling in a wireless sensor network, for collecting the sensor data. Delivery of the data is made at the red end stations.

of robots to use. We consider that robots are drones, which means they can fly directly between two sensors, so we use Euclidean distances in the sensor graph. First, we prove the necessity of partitioning the area, such that each robot will patrol its own region of the sensor graph. Then, we apply a cyclic patrolling strategy for each region of the graph partition and we explain this choice. For this, we define the CBSC problem (*Covering with Bounded Simple Cycles*) which assumes the covering of the graph nodes with a minimum number of bounded Hamiltonian cycles. We prove this problem is NP-hard and at least as hard as the Traveling Salesman Problem (TSP), which is NP-hard in turn (Papadimitriou, 1977). We also address the following questions : how to partition (or how to cover) the nodes of the graph, to improve the solution for the CBSC ? How to tell whether or not a set of nodes can be covered by an upper bounded Hamiltonian cycle (using a certain TSP heuristic) ? The heuristics that we analyze are state-of-the-art clusterings and our proposed *North-Eastern Neighbor* heuristic, based on geometric computations.

In Section 2, we present the problem setting and related notions, together with the work motivation. After introducing related work on patrolling, graph partitioning and covering in Section 3, we introduce the formalization of the CBSC problem and give insights on its complexity in Section 4. Section 5 presents the adaptation of classic clusterings for CBSC and introduces the North-Eastern Neighbor heuristic. In Section 6, we analyze the performances of these algorithms based on the TSP cycles computed by Christofides heuristics (Christofides, 1976). The results for CBSC on an extensive set of instances are analyzed with respect to the practical networking parameters and different graph models.

2 PROBLEM STATEMENT

In our work, we focus on proposing and analyzing solutions for multi-agent patrolling under WSN constraints, while trying to minimize the number of robots to use. The studied system assumes a Wireless Sensor Network (WSN), with fixed sensors, whose positions are known (though approximate positions are sufficient, within the limit of the radio range). We let $G = (V, E)$ be the complete graph of sensors, where a sensor $s \in V$ is represented by its (x, y) coordinates in the $2D$ plane, and E contains the $\frac{n\times(n-1)}{2}$ edges. The WSN produces data in real time, and some robots are in charge of performing a patrol to collect it in time, while taking into account the limited storage and energy they have. The robots possess a limited wireless communication range, so they can exchange data if it is the case. We assume the robots being drones, which can easily fly directly between any two sensors. Hence, we consider Euclidean distances between any two nodes of G, which is therefore metric. The collected data has to be delivered to an end station as soon as possible (see Fig. 1). Throughout the paper, we use the terminology of *Hamiltonian cycle* instead of *simple cycle*, when it does not cause any confusion. From now on, we use the definitions of (Chevaleyre, 2004).

Definition 1. *Each agent that performs a patrol follows a strategy. An agent strategy is defined as a function* $f : \mathbb{N} \rightarrow V$*, where each value* $f(i)$ *represents the* i^{th} *node visited by the agent. A multi-agent strategy is simply defined as a set of single-agent strategies.*

Definition 2. *Assuming a patrolling strategy on* G*, the idleness of a node represents the time elapsed since the last visit of an agent. For a sensor* $s \in V$*, its maximum idleness is denoted by* $MaxIdl(s)$ *(over all performed visits). The worst idleness is the maximum idleness recorded during the patrolling, among all* *the graph nodes, and it is denoted $WI(G)$. The average idleness of a graph is the average of its nodes' maximum idlenesses, and it is denoted $AvgIdl(G)$.*

There exist multiple types of idleness that we can consider as evaluation metric (Machado *et al.*, 2003) (Crites & Barto, 1996) (e.g. average idleness, polynomial idleness), but for our sensor graph, we will use the criteria of the worst idleness, introduced in (Machado *et al.*, 2003). Now we state a formal definition of the patrolling problem.

Definition 3. *For a connected graph $G = (V, E)$, the patrolling problem consists in finding a multi-agent strategy for visiting the nodes in V, which minimizes $WI(G)$.*

We now discuss the requirements and the constraints of the multi-robot patrolling in a WSN, and explain how we can solve the related issues. We argue why defining and addressing the CBSC problem is needed.

In practice, sensors have limited storage capacity (usually several kilobytes), so by producing data, its memory gets full at some point. Hence, the sensor storage capacity (denoted S_{mem}) and the throughput of data production by a sensor (denoted S_{thr}) give us an upper bound on the maximum idleness that the patrol should respect, in order to avoid memory overflow and not lose packets : $\frac{S_{mem}}{S_{thr}} \geq MaxIdl(s)$ (the sensor throughput is measured in bytes/sec). Having the bound for $MaxIdl(s)$ and the robot speed (denoted R_{speed}), we can compute the maximum length of the path the robots should walk between two consecutive visits of the same sensor : $\frac{S_{mem}}{S_{thr}} \times R_{speed}$. We could also consider the time spent by an agent for collecting the sensor data, but the sensor radio range permits the robot to collect the data while passing by the sensor and not stopping. Otherwise, it could just be a constant added to each edge (or node) in the graph, which can be neglected therefore.

Partitioning motivation

Two constraints which affect the patrolling are related to data delivery time and to robot storage capacity. We want the data to be delivered as soon as possible to the end station. When dealing with large WSNs, the delay would be high if delivered after patrolling the majority of the sensors. Hence, it is convenient to perform this after patrolling a region of the area. Moreover, robots have limited storage capacity, so they have to discharge the data at some point (through other robots or directly at the end station). This constraint places a bound on the number of sensors a robot should visit, and together with the delivery time constraint, it pushes towards the partitioning of the sensor graph. Since we do not want agents to leave their patrolling region, the data has to be passed between regions, to finally reach the end station. This can be done via inter-region agent rendezvous, which represent the trade-off when partitioning the area (see Fig. 2) : having too many rendezvous does not offer any advantage for data delivering, since they require time also. Hence, we aim at minimizing the number of regions, which in turn minimizes the number of inter-region rendezvous.

Energy consumption is another resource constraint which impacts the patrolling. Robots have limited energy and they need to recharge periodically. On the other hand, we want them to be able to perform at least one patrolling iteration of the sensor graph, before they recharge (i.e. to pass at least one time through each sensor). This constraint may be as well requiring to partition the graph, since the energy may not be sufficient enough for one iteration of the whole sensor graph. We could also consider energy consumption in the communication with the sensor, but it can be a constant added to each edge (or node) of the graph, and therefore can be neglected.

The above arguments express the need of partitioning the sensor graph into regions where the agent patrolling can respect the imposed constraints. For this, we first state a definition of partitioning :

Definition 4. *By partitioning the graph G, we mean covering the set of vertices V with a family of disjoint sets $\{P_i\}_{i=\overline{1,k}}$, where $P_i \subseteq V$, and $\bigcup_{i=1}^{k} P_i = V$. Here, P is called partition for the graph G (see Fig. 2).*

In our setting, we do not benefit of an infinite number of agents. Hence, we minimize resource usage by requiring a single agent for each region of the graph. Our objective is to use as few robots as possible for the patrol. A proof of the following result can be found in (Chevaleyre, 2005).

Theorem 1. *The optimal patrolling strategy in terms of worst idleness, for the single-agent case, is the strategy based on the shortest Hamiltonian cycle of the complete graph.*

Given this theorem, in the context of our setting, the optimal patrolling strategy to apply inside each graph region is the shortest Hamiltonian cycle of the subgraph. So we try to cover the sensor graph with cycles

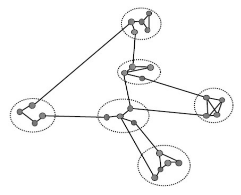
FIGURE 2 – Partition of a graph, with possible rendezvous paths between its regions (Chevaleyre, 2004).

for patrolling, cycles which respect networking and robot constraints. The bound for $MaxIdl(s)$ and the limited robot energy impose a bound on the length of the patrolling cycle. This translates into a problem of vertex covering with bounded simple cycles (CBSC), which we formalize in Section 4.

Before moving forward to related work and problem formalization, we briefly state our contributions : we address the problem of multi-agent patrolling in WSN by defining the problem of vertex Covering with Bounded Simple Cycles (CBSC). We prove it is NP-hard, and as a consequence we use heuristics to partition the graph and to cover it with bounded cycles. In this sense, we consider the *Hierarchical Agglomerative Clustering* and show a way it can be adequately used for specific requirements yielded by the networking metrics and robot limitations. We propose another heuristic which exploits the geometric structure of the graph, the *North-Eastern Neighbour* heuristic. We assess the performance of the partitioning algorithms with respect to the networking metrics, robot energy, and different graph models.

3 RELATED WORK

As presented in the literature (Chevaleyre, 2004) (Portugal & Rocha, 2011), the patrolling of each agent can be performed over the whole area, or with respect to some regions of the area (as in Fig. 3). Cycling strategies and partition based strategies have been studied by Chevaleyre (Chevaleyre, 2004). The experimental results showed that the ones based on cycles performed better than other strategies (such as Cognitive Coordination, or Reinforcement Learning). In addition, the solution improved when the cycling strategies were combined with the partitioning of the longest edges. We make use of this observation in our solution of Section 5. However, no constraints were considered on the patrolling cycles and no bounds on the node idleness.

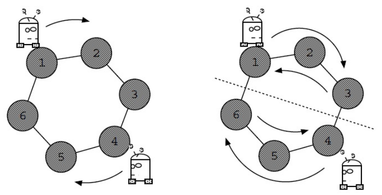

FIGURE 3 – Agents patrolling across the fixed sensors, in two different scenarios : patrolling of the whole graph, patrolling of individual regions (Chevaleyre, 2004).

To the best of our knowledge, there is no preferred partitioning algorithm for the multi-agent patrolling problem, as it is also pointed out in (Chevaleyre, 2004). For particular graphs like grids, Glad et al. (Glad *et al.*, 2008) used their ant-based EVAW algorithm to prove that when the agents self-organize in cycles (i.e. each agent patrols a cycle on the grid), these cycles are necessarily of the same length. Area patrolling under frequency constraints has been studied in (Elmaliach *et al.*, 2009), where all the agents follow one Hamiltonian cycle on the grid. The frequency constraint is respected by placing the agents at equidistant positions on the cycle, covering the whole environment. These papers do not consider a partitioned area, nor a weighted area graph. Partitioning of the area for agent patrolling has been proposed in (Portugal & Rocha, 2010) using the so-called Multilevel Subgraph Patrolling (MSP) algorithm. The authors use the Multilevel Graph Partitioning (Karypis & Kumar, 1998) to section the area graph and offer a decisional scheme which gradually proposes patrolling strategies in the order of their performance (e.g. Eulerian cycle, Hamiltonian cycle, etc.). Here, the partitioning decisions consider the total edge cost of the subgraphs, while trying to preserve their dimensions. Hence, the approach is not suitable when trying to avoid merging isolated regions of the graph (and long edges), like we initially prefer to do.

The computation of the shortest Hamiltonian cycle of a graph is a hard combinatorial optimization problem, known as the Travelling Salesman Problem (TSP). It is proven to be NP-hard (NP-complete for integer distances (Papadimitriou, 1977)). For this, Christofides (Christofides, 1976) offered a heuristic for a 1.5-approximation of the TSP solution. Vertex covering with a minimum number of bounded cycles has been studied for undirected unweighted graphs (Bekkai *et al.*, 2009). The authors provided an upper bound as a function of graph's order and graph's independence number. The problem becomes much harder, since our setting fundamentally assumes weighted links between nodes (i.e. Euclidean distances).

4 FORMALIZATION (CBSC PROBLEM)

In this section we formalize the CBSC problem and we address it in order to offer an efficient solution for multi-agent patrolling under WSN constraints. We discuss the main elements of the problem, give a hardness proof and point out another two variants of CBSC (minimum-weight CBSC and bounded-size CBSC).

Definition 5. *Let $G = (V, E)$ be a complete undirected graph. A real positive value (of finite precision) is assigned to each edge $e \in E$, representing its length L_e. The graph G is assumed to be metric*[1]*. The problem of Covering with Bounded Simple Cycles requires to cover the set of graph vertices V, with a minimum number of simple cycles, whose lengths are less then or equal to a bound B.*

There is no point in covering the same vertex twice, so we assume the cycles to be vertex-disjoint. Since we want to minimize the number of robots needed for the patrolling and the number of rendezvous between regions, we try to minimize the number of cycles. We evaluate the patrolling efficiency in terms of sensor idleness (concretely, the average graph idleness). In this regard, minimizing the idleness corresponds to the *minimum-weight* version of CBSC (MW-CBSC), where, among all the solutions of the CBSC, we search for the covering with the minimum total length. Related to the finite storage of the robot, avoiding the overflow corresponds to what we call the *bounded-size* CBSC (BS-CBSC). This version of the problem imposes a second bound on the size of each cycle, i.e. the number of sensors (or edges). This constraint is not making the general problem easier, since the cycle length does not depend on it, but the problem becomes trivial if we do not consider anymore the length bound B (or it is big enough to be ignored) : the solution of the BS-CBSC would be $\left\lceil \frac{|V|}{k} \right\rceil$, where $k \in \mathbb{N}$ is the size bound.

TSP and CBSC complexity

The problem of finding the shortest Hamiltonian cycle in a complete weighted graph is NP-hard, and is known as a combinatorial optimization problem, named *Traveling Salesman Problem* (TSP). The Euclidean TSP (i.e. TSP with Euclidean distances between points in the plane) is proven to be NP-complete (Papadimitriou, 1977). It corresponds to our setting (which we could call Euclidean CBSC), since we are aware of the sensors' positions. To denote the TSP solution length, we use the Opt_{TSP} notation.

TSP being NP-hard, in practice, people use certain heuristics to finally obtain desired running times (polynomial time algorithms). Christofides heuristic (Christofides, 1976) runs in time $O(|V|^3)$ and it is proven to deliver Hamiltonian cycles with length no more than $\frac{3}{2} \times Opt_{TSP}$. This heuristic will be used throughout the experimental process.

We now state a theorem regarding the hardness of CBSC. The proof can be found in Annex A, and it is based on the reduction of TSP to CBSC. Its basic idea lies on a "binary search" for the TSP solution. The decision at every step of the search is made with respect to a value of the bound B. The search time is $O(\log Opt_{TSP})$ (linear in the precision of the input). Since the reduction time is polynomial in the size of the input and Euclidean TSP is NP-complete, CBSC is therefore NP-hard.

Theorem 2. *CBSC is NP-hard, being at least as hard as the Traveling Salesman Problem (TSP).*

Corollary 1. *The BS-CBSC and MW-CBSC problems are at least as hard as the CBSC.*

1. graph whose edges respect the triangle inequality ($L_{ij} + L_{jk} \geq L_{ik}$, for all $i, j, k \in V$)

The proof of Corollary 1 can be made easily, since the reduction is straightforward in both cases.

A phase in the solution of CBSC requires that we cover a set of nodes with a bounded simple cycle. For the computation of a simple cycle on the set, we use the Christofides heuristic with polynomial complexity. But because of the hardness of CBSC (Theorem 2), the question is how to partition the graph in a reasonable amount of time (e.g. polynomial time), and consequently, how to tell if a set of nodes can be covered with a bounded simple cycle. We address the questions through heuristics, in the next section.

5 HEURISTICS FOR CBSC PROBLEM

We now present heuristics for the CBSC problem, which we test and compare in Section 6. We focus on the core issue of the problem : the partitioning of the sensor graph for cycle coverage. State-of-the-art heuristics for vertex partitioning include clustering algorithms, such as the hierarchical clusterings. We describe a heuristic of this type and show a way it can be adapted to our needs. Then, we present also our proposed N-EN heuristic.

5.1 Clustering Heuristics

For the partitioning of the sensor graph we consider clustering algorithms, such as the ones based on hierarchies, densities or centroids (e.g. k-Means clustering). Here, we choose the HAC heuristic because of the quadratic running time and the linkage properties it assumes. It does not assume predefined cluster centers (like the k-Means clustering), nor taking exponential time to run (like the divisive hierarchical clustering).

Hierarchical Agglomerative Clustering (HAC)

One popular clustering strategy is the hierarchical one (Jain *et al.*, 1999), which forms the cluster hierarchy either using the top-down (divisive) approach, or the bottom-up (agglomerative) one. HAC uses a proximity matrix and forms clusters incrementally based on the inter-node and inter-cluster distances. Hence, the clusters are formed according to the node connectivity (linkage), which is appropriate when trying to cover as many nodes as possible with an upper bounded cycle. HAC is known to be an efficient clustering algorithm, whose complexity is $O(|V|^2 \log |V|)$. We now show a possible way of using this heuristic for bounded cycle coverage.

HAC for CBSC problem

In order to use HAC clustering for the CBSC problem, we need to tell when a cluster is pertinent to form a cycle on it. So we introduce a condition for the evaluation of clustering : when a set of nodes is evaluated as sufficient enough for obtaining a Hamiltonian cycle with length close to the bound B, then we "freeze" that set (we do not continue to enlarge it). Consequently, it will be considered as a region of the final graph partition. The heuristic used for the evaluation of a set of nodes is a constant time function, which computes on the fly a Hamiltonian cycle on the nodes that are sequentially added to the set : if i and j are the first, respectively the last node added to the cluster, and the current cluster is evaluated to $s \in \mathbb{R}$, then for every new node k, the heuristic will test if $s + L_{jk} + L_{ki} \leq B$. In case the inequality is true, the set of nodes is considered adequate to be covered by a Hamiltonian cycle upper bounded by B, and s is set to $s + L_{jk}$. We use the same technique to evaluate a set of nodes in the case of the N-EN partitioning heuristic.
Cluster evaluation : For the realization of this cluster evaluation, we perform a depth-first search (DFS) of the cluster tree containing the hierarchy. Before we backtrack, we evaluate the cluster lying at that node of the tree. If it is considered adequate by the evaluation heuristic, then we move to its parent node (which represents its merging with another cluster). If the parent node is evaluated as inadequate, then we freeze the previous visited cluster (its child node) and move on to the unvisited children of the parent node. Since the search is a DFS, the algorithm stops when reaching the root of the cluster tree.

5.2 N-EN Heuristic Proposal

In the case of HAC, the clusters are formed in parallel based on the proximity matrix, which can result in too small or too big clusters (before a merging step, and respectively after). In addition, a part of the graph nodes do not get aggregated in any cluster until late in the clustering process. If before they do, the cluster is frozen by the partitioning heuristic, then the nodes will remain isolated, resulting in one-node regions. If we want to avoid these events, we need another type of heuristic. For this, we propose the *North-Eastern Neighbor* (N-EN) heuristic.

The N-EN heuristic assumes another parameter (L_{max}) which represents the maximum length allowed of any edge in the graph, hence defining the neighbors in the heuristic. We propose the following algorithm (Alg. 1) that consists of the main phases used in computation :

Algorithm 1: The proposed approach for CBSC

```
1 Do not consider edges $e \in E$, with $e > L_{max}$, when forming the neighbor lists
2 Obtain the connected components of the graph $G$
3 for every connected component of $G$ do
4     Run the N-EN heuristic for vertex partitioning
5     Run Christofides heuristic on every region of the partition
```

In the first phase of the algorithm, obtaining the neighbor lists according to L_{max} is done in quadratic time $O(|V|^2)$, which makes the proposed heuristic have the same time complexity as HAC. To obtain the connected components of G, we use the Tarjan SCC algorithm (Tarjan, 1971), which runs in linear time $O(|V|)$. After this, the nodes will be partitioned according to the N-EN heuristic, so that we finally obtain Hamiltonian cycles within our constraints. What follows regards the lines 4 and 5 of the Algorithm 1.

Algorithm 2: North-Eastern Neighbor Heuristic (+ Christofides execution)

```
1  for all connected components (C) of the graph do
2      Sort the nodes in C by their $(x + y)$ sum
3      $S = \emptyset$
4      while there are unvisited nodes inside C do
5          $v$ = choose the most North-Eastern node from C (not visited), which is neighbor to $S$
6                (mark it as visited)
7          $S = S \cup v$ ;
8          for all the unvisited neighbors $n$ of $v$ do
9              if $eval(S \cup \{n\}) \leq B$ then
10                 $S = S \cup n$ (mark $n$ as visited)
11         // $n$ = last neighbor of $v$
12         if $eval(S \cup \{n\}) > B$ then
13             run Christofides heuristic on $S$, and store the resulted cycle
14             $S = \emptyset$
15     run Christofides heuristic on $S$, and store the resulted cycle
```

The choice of starting from the north-eastern part of the area and processing it towards the south-west is due to the incremental coverage that we want to perform, in a rectangular environment. The diagonal coverage allows us to incrementally form cycles that do not stop growing once they hit the borders of the area (this is likely to happen in the case of north departure, when the coverage extends to western or eastern extremities). Concretely, that is minimizing the risk of covering the nodes at the edges of area with small cycles ($<< B$), when they could actually be avoided. The north-east ordering criteria simply considers the sum of the (x,y) coordinates of every sensor in the area. In Algorithm 2, every time the heuristic freezes a set of nodes or finishes visiting the neighbors of a node, it will choose the most north-eastern node to continue. Moreover, in the latter case, it will continue to form the set S starting with the most north-eastern node (from the connected component) that has a neighbor inside S. In this case, we start a new simple cycle on the newly added nodes. Its length is added to the evaluation. This is why we may obtain cycles that exceed B. Otherwise, the heuristic will deliver too many small cycles, while the good ones are the closest to B (to maximize the ratio %BC / TC, as in Section 6). The evaluation function $eval(S)$ is the same function as for HAC, used to evaluate a set of nodes according to the bound B. Sorting the nodes can be done in

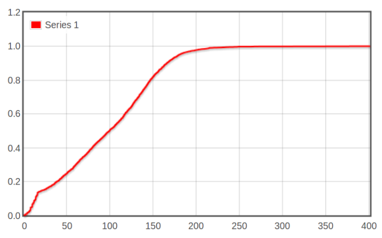

FIGURE 4 – The cumulative distribution function (CDF) of the cycle lengths, obtained for $B = 150$ and 500 random graphs of 100 nodes, using the N-EN heuristic. X-axis : the lengths of the cycles.

	N-EN Heuristic	HAC clustering
%BC	82.27	92.87
#BC | TC	2679 | 3256	3571 | 3845
%BC / TC $\times 10$	0.25	0.24
worst idleness $WI(G)$	401	343
avg idleness $AvgIdl(G)$	96.2	80.1
avg # of iterations	1.5	1.8

TABLE 1 – The results obtained with HAC and N-EN heuristics, on an extensive set of 500 random graphs of 100 nodes, for $B = 150$; %BC = the percentage of bounded cycles (over TC) ; TC = the total number of cycles.

$O(|V| \log |V|)$ and choosing the most North-Eastern node, neighbor to set S, takes at most $O(|V|^2)$ steps for the whole execution of the algorithm. Visiting the neighbors of each node takes also $O(|V|^2)$ time, so the worst case performance of the algorithm is $O(|V|^2)$.

6 PERFORMANCE ANALYSIS

In the experiments, we used the heuristics presented in the previous section (HAC and N-EN), for the partitioning of the sensor graph. For the computation of a Hamiltonian cycle in each region of the graph partition, we used the Christofides heuristic. We also implemented the procedures necessary to generate random positions for the graph vertices in the $2D$ plane of any size or to consider any particular graph as input. The vertex positions have been generated using the random uniform distribution. Moreover, all the experiments have been conducted for an area size of 100×100 meters. For HAC, we used the criteria of complete-linkage clustering (the distance between two clusters $C1$ and $C2$ is the maximum distance between any node in $C1$ and any node in $C2$). We consider that all lengths are measured in meters.

Table 1 presents the results obtained with HAC and N-EN on an extensive set of random graph instances. The results are compared with respect to the percentage of cycles with length less than B (denoted BC), with respect to the number of cycles in the coverage (TC), the worst idleness of the patrolling ($WI(G)$), the average graph idleness ($AvgIdl(G)$) and the average number of iterations a robot would have to patrol (which is actually $B/AvgIdl(G)$). We also evaluate the ratio %BC / TC (since the desired %BC would be 100, while minimizing TC).

While HAC delivers more cycles respecting the bound B, it also delivers a greater number of cycles than the N-EN heuristic. In this case, $\approx 93\%$ of the cycles respect the maximum idleness and energy, but their number is much higher than in the case of N-EN. Hence, we could argue that N-EN performs better in this sense, but if computing the ratio of these metrics (%BC / TC), we observe that the heuristics actually have almost equal performances. The $WI(G)$ and $AvgIdl(G)$ of the N-EN are bigger than the ones of HAC, which may be explained by the fact that N-EN delivered a smaller number of cycles, but with higher lengths. The average number of iterations a robot would patrol is higher in the case of HAC, which implies a lower amount of energy consumption (if recharging after 1 iteration), so HAC performs better here. Yet, regarding the minimization of the number of robots, one would like to obtain a value closer to 1 (but not less), since this means getting cycles close to B (but not higher), and for this, N-EN behaves better (see

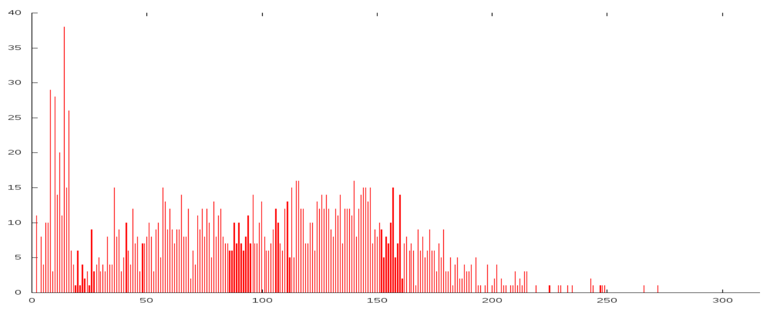

FIGURE 5 – X-axis : the cycle length. Y-axis : the frequency of the cycle length (in terms of number of cycles). The frequency of the cycle lengths, obtained for $B = 150$ and 250 random graphs of 100 nodes, using the N-EN heuristic.

"avg # of iterations" in Table 1). Overall, though it is convenient to obtain more patrolling cycles which respect the data collecting frequency ($\frac{S_{mem}}{S_{thr}}$) and the energy constraint, obtaining a higher number of such cycles means a higher number of graph regions, robots to use and inter-region rendezvous, which does not optimize the objective of CBSC.

In the CDF of Figure 4, one can see the probability of a cycle to have length (C_l) less than the bound B : $P(C_l < B = 150) \approx 80\%$, which corresponds to the data in the Table 1. The CDF increases linearly uniform up to a threshold around the bound $B = 150$. This means that the cycle lengths are nearly uniformly distributed across the interval of $[0, 150]$ (as it can be also seen in Fig. 5). A peak of the CDF is observed for cycles of small lengths, this being a cause of the random graphs, which often contain groups of isolated node.

We now discuss in detail the behavior of the N-EN heuristic. We describe the observations with respect to parameters B and L_{max}.

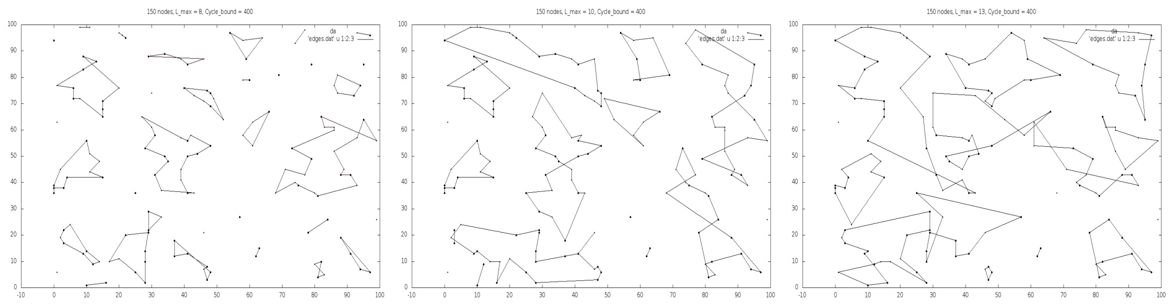

FIGURE 6 – The evolution of the cycle coverage, as L_{max} parameter increases (8, 10 and 13). Scenario for 150 nodes and B fixed to 400.

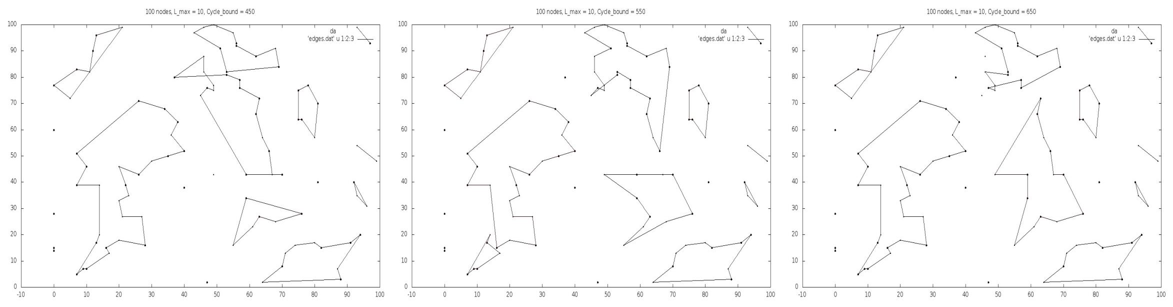

FIGURE 7 – The stabilization of the cycle coverage, as B increases (450, 550 and 650). Scenario for 100 nodes and L_{max} fixed to 10.

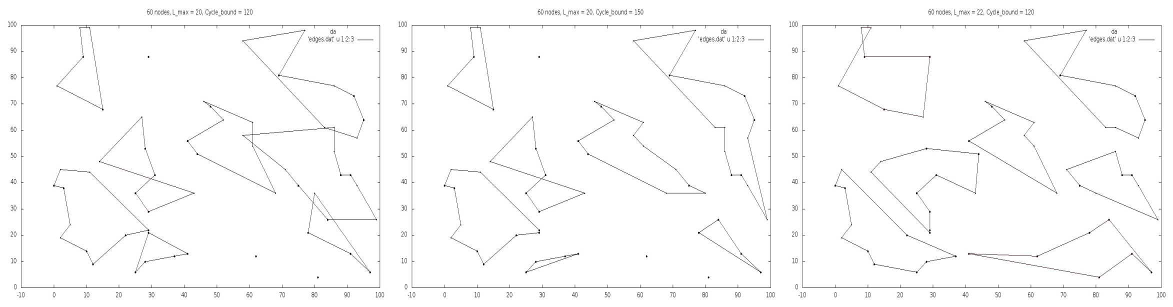

FIGURE 8 – The improvement of the cycle coverage, as B and L_{max} are modified together ($120-20$, $150-20$ and $120-22$). Scenario for 100 nodes.

***L_{max}* parameter variation**

In the graphics of Figure 6, one can observe how the solution changes with respect to the modification of L_{max}. Throughout the experiments, we observed that when L_{max} is inside a specific interval, cycles do not overlap. In the random graph model it is difficult to say which interval is the best. When L_{max} increases above a specific limit, the cycles start to overlap. The complementary situation is true : when the parameter decreases below a specific value, the cycles start to be smaller or to disappear. The motivation behind this aspect lies in the N-EN heuristic, where, if L_{max} increases, the nodes neighborhood gets larger and the number of neighbors increases.

***B* parameter variation**

In another scenario, we fix the L_{max}. Then, if only the B parameter modifies, the resulted cycles will get larger and larger on the graph, but at some time they will stabilize (see Figure 7). That is because the L_{max} parameter fixes the set of neighbors for a node, and a set of the partition cannot get larger, even if the bound B allows this. We can say that the solution stabilizes, as no big differences occur from that point on.

***B* and *L_{max}* variation**

Another thing observed during the experiments with N-EN heuristic is that the variation of B within a threshold, can lead to better coverage if several values for L_{max} are tried. So, by adjusting the two parameters together, B and L_{max}, one may find a set of cycles that do not overlap and that cover all the sensors (i.e. containing no isolated sensors ; see Figure 8). However, how to properly choose the parameters remains an open question.

One problem that still persists with the N-EN heuristic is that sometimes a cycle of the coverage is consistently bigger than all the other ones. From the experimental observations, this exception occurs only in some graphs, and it may happen because of the way we form the set of nodes inside the heuristic. In some cases, a post-covering optimization could be done, to redefine the shape of cycles whose edges overlap. In this sense, other TSP heuristics could be tried as well.

7 CONCLUSION

In the conducted study, we started from the general problem of multi-agent patrolling under the constraints of wireless sensor networks, problem which has not been studied before in the context that we do. The novelty that the problem setting brings is the consideration of networking aspects and constraints in the strategies used for multi-agent patrolling.

This paper addressed the problem of multi-agent patrolling in WSNs by defining and formalizing the problem of vertex Covering with Bounded Simple Cycles (CBSC). We proved it is NP-hard, and consequently considered polynomial-time algorithms to offer solutions for CBSC. Our algorithms, adapted HAC and the proposed N-EN heuristic, were evaluated with respect to networking metrics, robot limitations, and different graph models. Our results show that HAC performs better when it comes of cycles that respect the networking idleness ($WI(G)$ or $AvgIdl(G)$), but N-EN tends to form fewer cycles in each coverage. The ratio between these metrics finally shows that overall the heuristics have almost equal performance, with a slight advantage for N-EN (due to the number of cycles it delivers).

Annex A

Proof (Theorem 2). We show that TSP can be reduced in polynomial time to CBSC. Let $G = (V, E)$ be the graph for which we want to solve the TSP, where E contains the $\frac{n\times(n-1)}{2}$ edge lengths in finite precision. For simplicity, we round the values to integers (i.e. multiplication by $10^{|s|}$, where $|s|$ is the numeric scale of the lengths). First, let H be the length of a Hamiltonian cycle on G (obtained with Christofides heuristic or any other algorithm). This gives us an upper bound on the Opt_{TSP}. Then, we set the interval for the binary search : $[Left, Right] \leftarrow [0, H]$ (or $[\frac{2}{3} \times H, H]$ suffices in the case of Christofides heuristic). We start the search by setting the bound for CBSC to the middle of the interval : $B \leftarrow \lfloor \frac{Left+Right}{2} \rfloor$. We check the solution of the CBSC, to see if the minimum number of cycles is 1 or greater than 1. We set $Right \leftarrow B$ in the first case, or $Left \leftarrow B$ in the second. The search is performed until $L = R$, which will represent the actual Opt_{TSP}. The running time of Christofides heuristic is $O(|V|^3)$, while the binary search is performed in $O(\log H)$. Hence, the time of the reduction is polynomial in the size of the input. □

Acknowledgments

This work was partly supported by Region Rhône-Alpes (France).

Références

BASAGNI S., CAROSI A., MELACHRINOUDIS E., PETRIOLI C. & WANG Z. M. (2008). Controlled sink mobility for prolonging wireless sensor networks lifetime. *Wirel. Netw.*, **14**(6), 831–858.

BEKKAI S., FORGE D. & KOUIDER M. (2009). Covering the vertices of a graph with cycles of bounded length. *Discrete Mathematics*, **309**(8), 1963 – 1966.

CHEVALEYRE Y. (2004). Theoretical analysis of the multi-agent patrolling problem. In *Intelligent Agent Technology, 2004. (IAT 2004). Proceedings. IEEE/WIC/ACM International Conference on*, p. 302–308.

CHEVALEYRE Y. (2005). *The patrolling problem.* Technical report, Univ. Paris 9. Available in french at `http://www.lamsade.dauphine.fr/~chevaley/papers/anna_patro.pdf`.

CHRISTOFIDES N. (1976). *Worst-case analysis of a new heuristic for the travelling salesman problem.* Technical Report 388, Graduate School of Industrial Administration, Carnegie Mellon University.

CRITES R. & BARTO A. (1996). Improving elevator performance using reinforcement learning. In *Advances in Neural Information Processing Systems 8*, p. 1017–1023 : MIT Press.

DANIEL K., DUSZA B., LEWANDOWSKI A. & WIETFELD C. (2009). Airshield : A system-of-systems muav remote sensing architecture for disaster response. In *Systems Conference, 2009 3rd Annual IEEE*, p. 196–200.

ELMALIACH Y., AGMON N. & KAMINKA G. A. (2009). Multi-robot area patrol under frequency constraints. *Annals of Mathematics and Artificial Intelligence*, **57**(3-4), 293–320.

GLAD A., SIMONIN O., BUFFET O. & CHARPILLET F. (2008). Theoretical study of ant-based algorithms for multi-agent patrolling. In *Proceedings of the 2008 Conference on ECAI 2008 : 18th European Conference on Artificial Intelligence*, p. 626–630.

JAIN A. K., MURTY M. N. & FLYNN P. J. (1999). Data clustering : A review. *ACM Comput. Surv.*, **31**(3), 264–323.

KARYPIS G. & KUMAR V. (1998). A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, **20**(1), 359–392.

MACHADO A., RAMALHO G., ZUCKER J.-D. & DROGOUL A. (2003). Multi-agent patrolling : An empirical analysis of alternative architectures. In *Proceedings of the 3rd International Conference on Multi-agent-based Simulation II*, MABS'02, p. 155–170.

MARTA M. & CARDEI M. (2009). Improved sensor network lifetime with multiple mobile sinks. *Pervasive and Mobile Computing*, **5**(5), 542 – 555.

PAPADIMITRIOU C. H. (1977). The euclidean travelling salesman problem is np-complete. *Theoretical Computer Science*, **4**(3), 237 – 244.

PORTUGAL D. & ROCHA R. (2010). Msp algorithm : Multi-robot patrolling based on territory allocation using balanced graph partitioning. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, p. 1271–1276.

PORTUGAL D. & ROCHA R. (2011). A survey on multi-robot patrolling algorithms. In L. CAMARINHA-MATOS, Ed., *Technological Innovation for Sustainability*, volume 349 of *IFIP Advances in Information and Communication Technology*, p. 139–146. Springer Berlin Heidelberg.

RESCUE R. (2001). home page : `http://www.robocuprescue.org`.

TARJAN R. (1971). Depth-first search and linear graph algorithms. In *Switching and Automata Theory, 1971., 12th Annual Symposium on*, p. 114–121.

TUNCA C., ISIK S., DONMEZ M. & ERSOY C. (2014). Distributed mobile sink routing for wireless sensor networks : A survey. *Communications Surveys Tutorials, IEEE*, **16**(2), 877–897.