A Framework for Multi-Robot Node Coverage in Sensor Networks

Andrea Gasparri [*]
Dip. di Informatica e Automazione
Università "Roma Tre"
Via della Vasca Navale, 79, 00146 Roma, Italy
Phone: +39-06-55173206 Fax: +39-06-5573030
gasparri@dia.uniroma3.it

Bhaskar Krishnamachari
Department of Electrical Engineering
University of Southern California
Los Angeles, CA, 90007 USA
Phone: +1-213-821-2528 Fax: +1-213-821-1109
bkrishna@usc.edu

Gaurav S. Sukhatme
Robotic Embedded Systems Laboratory
University of Southern California
Los Angeles, CA, 90089 USA
Phone: +1 213-740-0218 Fax: +1 213-821-5696
gaurav@usc.edu

Abstract

Area coverage is a well-known problem in robotics. Extensive research has been conducted for the single robot coverage problem in the past decades. More recently, the research community has focused its attention on formulations where multiple robots are considered. In this paper, a new formulation of the multi-robot coverage problem

[*] *Corresponding Author*

is proposed. The novelty of this work is the introduction of a sensor network, which cooperates with the team of robots in order to provide coordination. The sensor network, taking advantage of its distributed nature, is responsible for both the construction of the path and for guiding the robots. The coverage of the environment is achieved by guaranteeing the reachability of the sensor nodes by the robots. Two distributed algorithms for path construction are discussed. The first aims to speed up the construction process exploiting a concurrent approach. The second aims to provide an underlying structure for the paths by building a Hamiltonian path and then partitioning it. A statistical analysis has been performed to show the effectiveness of the proposed algorithms. In particular, three different indexes of quality, namely completeness, fairness, and robustness, have been studied.

Keywords: Multi-Robot Systems, Sensor Networks, Area Coverage

1 Introduction

Area coverage has been investigated by the robotics research community through the years. Indeed, this problem lends itself to several applications in different fields, from industrial, such as lawn-mowing [1] or vacuum-cleaning [2], to military such as de-mining [3], or humanitarian, such as search and rescue operations [4].

The robot area coverage problem is the problem of determining a path that must be followed by a robot in order to completely cover the environment. Several approaches, ranging from grid decompositions of the environment to the development of heuristics, have been proposed. More recently, formulations which extend the problem to the multi-robot context have been introduced. The idea is to take advantage of the cooperation among the robots to provide higher robustness as well as to lower the time required to complete the task.

Indeed, three major issues define the area coverage problem, according to [5]: the capability to generate paths that are able to completely cover the environment; the time required to complete the coverage operations; and finally, the availability of a priori information of the environment. Additionally, a fourth issue derives from the presence (or absence) of obstacles.

In this paper, a new formulation for the Multi-Robot Coverage problem is proposed. The novelty of this work is the introduction of a sensor network, which cooperates with the team of robots in order to provide coordination to it. This paper investigates how the dynamics of the area coverage problem are modified when introducing a sensor network into the system. The sensor network, taking advantage of its distributed nature, is responsible for both the construction of the path and for guiding of the robots. Here, we focus on the distributed construction of paths. Specifically, two distributed algorithms have been provided. The first one aims to speed up the construction process exploiting a concurrent approach. The second aims to provide an underlying structure by building a Hamiltonian path and then partitioning it. A statistical analysis has been performed to show the effectiveness of the proposed solutions. In particular, three different indexes of quality, namely completeness, fairness, and robustness, have been studied.

The rest of the paper is organized as follows. In section 2 the state of the art of both the Robot Coverage problem and of the Hamiltonian Path problem is given. In section 3 the formulation of the problem is given and some indexes of quality are proposed. In section 4 an overview of the two proposed distributed approaches is given. In section 5 the most relevant aspects of these approaches are deeper investigated. In section 6 the effectiveness of these approaches with respect to the indexes of quality introduced in section 3 is investigated. In section 7 the performance, in terms of computational load and communication load is discussed. Finally, in section 8 conclusions are drawn.

2 Related Work

2.1 Robot Area Coverage

The (area) coverage problem was shown to be related to the covering salesman problem in [6]. Specifically, the covering salesman problem is a variant of the traveling salesman problem where, instead of visiting each city, an agent must visit a neighborhood of each city that minimizes the travel length for the agent. In [6], this problem was proven to be NP-hard making use of the reduction from the (NP-hard) problem “Hamiltonian Circuit in Planar Bipartite Graphs with Maximum Degree 3” to the problem “Hamiltonian Circuit in Grid Graphs”.

In [7], two different formulations of the coverage problem are described: *offline* and *online*. The offline formulation assumes robots to be equipped with a map of the work area, [8],[9],[10]. The online formulation does not assume any prior information about the environment to be available for the robots, [11], [12], [13]. Moreover, coverage algorithms can be classified into deterministic and non-deterministics according to [14]. Deterministic approaches ([15],[16],[17],[18],[19]) guarantee the complete coverage of the environment, while non-deterministic approaches ([20],[21]) cannot.

In [8], the problem of covering a continuous planar area by a square-shaped tool attached to a mobile robot is addressed. Here, the authors suggest to subdivide the work-area into disjoint cells, whose size is related to the tool carried by the robot. Successively, a spanning tree of the graph induced by the cells is performed, while covering every point precisely once. Hence, the coverage of the environment is simply obtained by letting the robot circumnavigate the tree.

In [9], an offline formulation of the multi-robot coverage problem is addressed. A path-planning algorithm which extends the idea proposed in [8] to the multi-robot scenario is described. The main contribution of this work is the introduction of the Multi Spanning Tree Coverage (MSTC) problem formulation. The idea is to optimally divide the spanning-tree previously computed in such a way that each robot covers an equal portion of the tree. An argumentation about the robustness and the efficiency of the proposed algorithm is provided as well.

In [10], an extension of the previous approach is suggested. This work is motivated by the observation that the coverage time can be significantly influenced by the structure of the spanning trees. As a result of their investigation, the authors claim that an optimal time coverage algorithm for a system with k robots will result (at least theoretically) in total coverage time of $\lceil N/k \rceil$, where N is the number of cells. In addition, they prove that robots should be spread out as uniformly as possible along the spanning tree in order to achieve this goal. In fact, in this way the k paths will result in almost the same length.

In [11], the authors propose an algorithmic approach to deal with the distributed complete coverage problem. This work represents an extension to the single robot sensor-based coverage of unknown environments proposed in [22]. By assuming that global communication is available among robots, each robot is assigned an area of the unknown environment to cover. This area is decomposed into cells and it is described by an adjacency graph which is incrementally built. In detail, the Morse decomposition based on the Boustrophedon approach, first proposed in [23], is exploited by each robot. Robots can achieve a global picture of the environment by integrating each graph with information coming from other robots.

In [12], an algorithm for the complete multi-robot coverage of a connected space with unknown obstacles is presented. This algorithm operates by maintaining, as far as possible, small uncovered regions between covered areas and obstacles. In addition, the authors show that repeated coverage can occur only around regions where the paths between obstacles are less than twice the width of the robots coverage range. This property holds even when the robots have no *a priori* knowledge of the environment, and therefore helps to prevent unnecessary wastage of time and resources.

In [13], the authors introduce a novel strategy for the exploration of an unknown environment with a multi-robot system. Communication among robots is restricted to line-of-sight and to a maximum interdistance between robots. The proposed strategy produces a spring-like formation which scan the area in strips. Additionally, in the presence of obstacles the formation is deformed and split in two in order to circumvent the obstacle and to adapt to the varying width of the free space. Moreover, the authors provide an upper bound for the amount of repeated coverage. This is limited to the areas where paths between obstacles are too narrow to allow robots to enter and leave the area on non-overlapping paths.

In [15], an algorithm for efficient cooperative search is described. The idea is to divide a work area into small pieces in order to make multiple mobile robots cooperate efficiently. The searching motion of mobile robots is represented by a queue of paths and the division of the work areas is achieved by allocating appropriate paths to each robot respectively. In this way, the cost of the searching can be shared among the multiple mobile robots.

In [16], the problem of cooperative area sweeping by multiple mobile robots is addressed. The authors propose a non path-following approach called On-Line Goal Selection (OGS) algorithm for robot motion planning in autonomous behavior. Cooperation is achieved by organizing the robots as a decentralized market-like structure and task-sharing is addressed by exploiting a negotiation mechanism.

In [17], a distributed cooperative coverage algorithm named DC_R is described. It represents an extension of an earlier complete single-robot algorithm called CC_R [24]. In detail, DC_R executes independently on each robot in a team where the individual robots do not know the initial locations of their peers and applies to systems of robots operating in a rectilinear environment that use only intrinsic contact sensing to determine the boundaries of the environment.

In [18], a polynomial-time multi-robot coverage heuristic named the Multi-Robot Forest Coverage (MFC) is proposed. This approach relies on an algorithm for finding a tree cover with trees of balanced weights (one for each robot). In addition, the authors provide an analysis which shows the cover time to be at most eight times larger than the optimal one.

In [19], the authors introduce a distributed algorithm exploiting the indirect form of communication adopted by ants. Robots leave chemical odor traces which evaporate over time and evaluate the strength of smell at every point they reach, with some measurement error. The effectiveness of this communication scheme to perform the task of cleaning the floor of an unmapped building, or more broadly any other task requiring the traversal of an unknown region is investigated.

In [20], two algorithms for solving the 2D coverage problem using a team of robots are described. Specifically, these algorithms rely on the observation that local dispersion is a natural way to achieve global coverage. Therefore, both algorithms are based on local, mutually dispersive interaction between robots when they are within sensing range of each other. In fact, “spreading out” robots throughout the environment lowers the risk of having robots too close to each other, which would result in poor coverage as overlap would be experienced.

In [21], a biologically inspired neural network approach to autonomous cooperative coverage path planning of multiple cleaning robots is proposed. Paths are generated on-line by using a neural network, where the dynamic of each neuron is characterized by a shunting neural equation. Robots see each other as moving obstacles and cooperate to achieve a common sweeping goal.

In this work, a slightly different formulation is proposed. The classic scenario is extended by a sensor network which provides coordination to the robots by exploiting its distributed nature. The sensor network is responsible for both the construction of the path and for guiding of the robots. In particular, this work investigates how the dynamics of the coverage problem are modified by the introduction of such a sensor network.

2.2 Hamiltonian Path

In the mathematical field of graph theory, a Hamiltonian path is a path in an undirected graph which visits each vertex exactly once. A Hamiltonian cycle (or Hamiltonian circuit) is a cycle in an undirected graph which visits each vertex exactly once and also returns to the starting vertex. Determining whether such paths and cycles exist in graphs is the Hamiltonian path problem which is NP-complete [25].

The Hamiltonian path problem for graph G is equivalent to the Hamiltonian cycle problem in a graph H obtained from G by adding a new vertex and connecting it to all vertexes of G. The Hamiltonian cycle problem is a special case of the traveling salesman problem (TSP), obtained by setting the distance between two cities to a finite constant if they are adjacent or to infinity otherwise.

Several centralized approaches providing either exact or approximate solutions for the TSP have been proposed in literature. Exact solutions are generally obtained exploiting branch and bound techniques [26], linear programming formulations [27], or integer programming formulations [28]. Approximate solutions are achieved using heuristic local search methods. Heuristics are largely applied to solve large instances of the TSP as they can compute near optimal solutions in a relatively short time. Starting from the classical tour construction heuristics such as nearest neighbor heuristics [29], insertion heuristics [30], heuristics based on spanning trees [31], savings heuristics [32], and 3-opt [33], further heuristic approaches have been proposed like simulated annealing [34], genetic algorithms [35], neural networks [36] or ant-colony [37].

Distributed implementation methods can be found in literature as well. For instance, a distributed implementation of simulated annealing is described in [38], while a distributed implementation of a parallel genetic algorithm on a cluster of workstations is given in [39]. Another distributed approach, which provides a parallelization of a branch-and-bound algorithm, is proposed in [40].

Many variations of the classical TSP formulation have been proposed over the years. Among them, the Multiple Traveling Salesman Problem (MTSP) is the closest to the Multi-Robot Coverage Problem. An interesting extension of the MTSP is the balanced version for which all paths are required to be of equal length. An overview of the MTSP formulation and solution procedures can be found in [41].

3 Problem Formulation

Given a sensor network described by a graph $G = (N, L)$ where N is the set of nodes (sensors) with cardinality n and L is the set of edges (connectivity matrix) and, given a set K of robots with cardinality k, the following assumptions are made:

- The sensor network is deployed within an open environment without obstacles. (The deployment might be either manual or automatic by a robot. If by robot, the algorithm proposed in [42] or [43] might be used),
- The *quasi-uniform* deployment is obtained by placing each sensor node at exactly one vertex of a noisy grid built over the whole environment,
- Each node knows its location in respect to a global frame (the localization can be achieved for instance using the algorithm proposed in [44]),
- There is a link layer that provides the abstraction of reliable communication (for instance using re-transmission),

- The cardinality of the robots set is strictly lower than the cardinality of the sensors set: $k << n$.

According to this scenario, the coverage problem consists of constructing k paths, where a path is defined as an open succession of 2-connected nodes, so that:

- each node belongs to a path
- any pair of paths is disjointed
- all paths are of equal length (optimality condition)

Hence, the coverage of an environmet is achieved by guaranteeing the reachability of the nodes by the robots. Note that, the introduction of a sensor network allows some assumptions to be relaxed. Specifically, the following significant differences can be pointed out with respect to the "off-line coverage" formulation:

- No prior knowledge of the environment is required for the robots
- The number of robots, either joining or leaving the network, can be assumed to be dynamic

At the same time, this formulation leads to new interesting questions:

- How to provide a fully distributed framework able to build paths as close to optimality as possible?
- How to handle the eventuality that a robot might join (respectively leave) the network. This would imply the ability of the network to re-organize the pre-existing paths in order to keep the "optimality" condition.

This formulation can be described by the Integer Linear Programming formulation proposed in Appendix A.1 if the distance between adjacent nodes is considered approximately constant, as a consequence of the *quasi-uniform* deployment. In this case, any feasible solution is inherently optimal. However, it is legitimate to ask how close to optimality can a solution be if a distributed formulation is considered. Therefore, the following properties, particularly the *fairness* of the solution, are investigated in order to evaluate the effectiveness of the proposed solution:

- *Completeness* in terms of network coverage, which means having each node belonging to a path

- *Fairness* in terms of equal distribution of the duty among the robots, which means having paths in average of the same length
- *Robustness* in terms of ability to re-cover from the situation in which a robot might get broken or a new one might join the network

Indeed, these indexes were already adopted in [9]. Although some similarities might be pointed out, this paper is mainly focused on investigating how the dynamics of the problem are modified by the deployment of a sensor network within the environment.

4 The Proposed Solution

Two different distributed approaches are proposed. Both of them have strengths and weaknesses that will be discussed in detail. However, due to the NP-hard nature of the problem, none of them is able to guarantee optimality in a deterministic way. For this reason, statistical analysis is provided in order to validate their effectiveness. Following the formulation given in Section 3, it can be observed that a fair distribution of the duty is achieved if the number of nodes belonging to any path is approximately $\lceil N/k \rceil$. Indeed, this is in agreement with the condition of optimality proposed in [10], for which an optimal algorithm should lead to a total time coverage of $\lceil N/k \rceil$. In fact, due to the *quasi-uniform* deployment of the sensor network, the time required for a robot to travel between two neighbors can be approximatively considered a constant c in average. Therefore, a path can be traversed in $c \cdot \lceil N/k \rceil$.

4.1 Algorithm I: Overview

The first algorithm constructs paths in a concurrent way. The idea is to use the distributed nature of the sensor network to speed up the construction process. There are tree steps:

- Heads Selection
- Coarse Paths Construction
- Orphan Recovery Policy

4.1.1 Heads Selection

When the presence of the robots is sensed, this information is spread out across the network, and the heads selection process is triggered. The idea is to select heads, i.e., nodes where the path construction process is started from, so that a natural partition of the environment is achieved. Intuitively, this can be explained by the fact that, as the algorithm is distributed and the construction of the paths is concurrent, minimizing the interaction improves the performance. More details will be provided in the next section.

4.1.2 Coarse path construction

This process is started as soon as the heads have been chosen. As previously stated, the idea is to exploit the distributed nature of the network to speed up the path construction process. Several policies to make the more effective local decision have been investigated and their peculiarities will be shown in the next section. The pseudo-code in Algorithm (1) shows a possible distributed implementation of the proposed solution, where v and p are respectively the current tail of the path and its predecessor, $\mathcal{N}(i)$ is the set of neighbors of node i, $\mathcal{N}_{av}(i) \subset \mathcal{N}(i)$ is the subset of available neighbors describing the possible candidates to be added and $d(i, j)$ represents a distance between node i and node j. Fig. 1 shows a typical result of the coarse path construction procedure.

4.1.3 Orphan Recovery Policy

Due to the distributed nature of the system as well as the locality of the decision process, a second step is required in order to fully cover the environment. Once the first step is terminated, there might be several incomplete paths with different lengths. For this reason, a local shortest-path-first-served mechanism is provided in order to balance the final length of the paths.

Nodes which have not been added to any path, after a period of latency of attachment request put themselves in the "orphan" status and start this process. Note that, particular attention should be paid to avoid the formation of crosslinks when adding orphans to the paths. For this reason, an ad-hoc policy to avoid this issue has been devised and will be discussed in the next section. The pseudo-code in Algorithm (2) shows a possible distributed implementation of this step, where v is an orphan, $M(i, j)$ is the crossing links check function, $Att(i, j)$ is the checks . Fig. 2 depicts how the paths are modified after the Orphan Recovery Policy is run.

Algorithm 1: Coarse path construction

```
$v \leftarrow$ Tail
$p \leftarrow$ Tail's predecessor
$v$ broadcasts request of availability to $\mathcal{N}(v)$
$\mathcal{N}_{av}(v) \leftarrow v$ gains availability responses
$d \leftarrow \infty$
$n \leftarrow 0$
for $j \in \mathcal{N}_{av}(v)$ do
    $v$ computes $d(v,j)$
    if $d(v,j) < d$ then
        $d = d(v,j)$
        $n = j$
    end
end
if $n = 0$ then
    $v$ notifies to $p$ the end of construction process
else
    $v$ broadcasts the request of attachment of $n$
    $v$ receives the reply $r$ from $n$
    if $r = success$ then
        $v$ notifies to $n$ to start the path construction process
    else
        $N_{av}(v) = N_{av}(v) \setminus \{n\}$
        Go to 5
    end
end
```

Algorithm 2: Orphan Recovery Policy

```
1  $v \leftarrow$ Orphan
2  $v$ broadcasts request of attachment to $\mathcal{N}(v)$
3  $\mathcal{N}_{av}(v) \leftarrow v$ gains availability responses
4  $\mathcal{N}_{sav}(v) \leftarrow v$ sorts $\mathcal{N}_{av}(v)$ w.r.t. ascending length of path
5  $M,\ M_B \leftarrow \infty$
6  $B,\ p,\ p_B \leftarrow 0$
7  for $j \in \mathcal{N}_{sav}(v)$ do
8      $Att(v,j) \leftarrow v$ computes attachment condition
9      if $Att(v,j) = Branch$ then
10         $v$ computes $M'(v,j)$
11         if $M'(v,j) < M_B$ then
12             $M_B = M'(v,j)$
13             $p_B = j$
14         end
15     else
16         $v$ computes $M(v,j)$
17         if $M(v,j) < M$ then
18             $M = M(v,j)$
19             $p = j$
20         end
21     end
22 end
23 if $p \neq 0$ then
24     $v$ sends request of attachment to $p$
25     $v$ receives the reply $r$ from $p$
26     if $r = success$ then
27         $v$ sets its status from “orphan” to “visited”
28     else
29         Go to 2
30     end
31 else if $p_B \neq 0$ then
32     $v$ sends request of attachment to $p$
33     $v$ receives the reply $r$ from $p$
34     if $r = success$ then
35         $v$ sets its status from “orphan” to “visited”
36     else
37         Go to 2
38     end
39 end
```

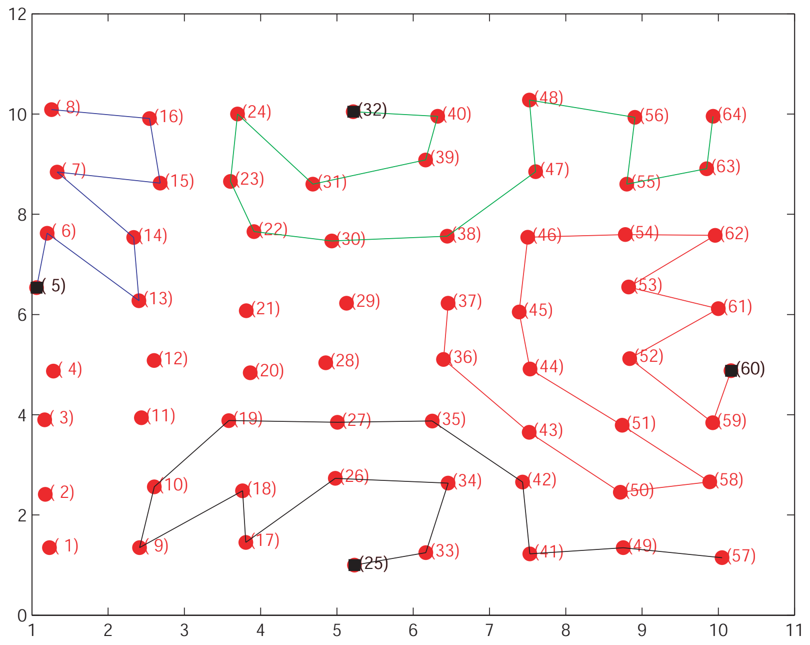
Figure 1: Coarse path construction

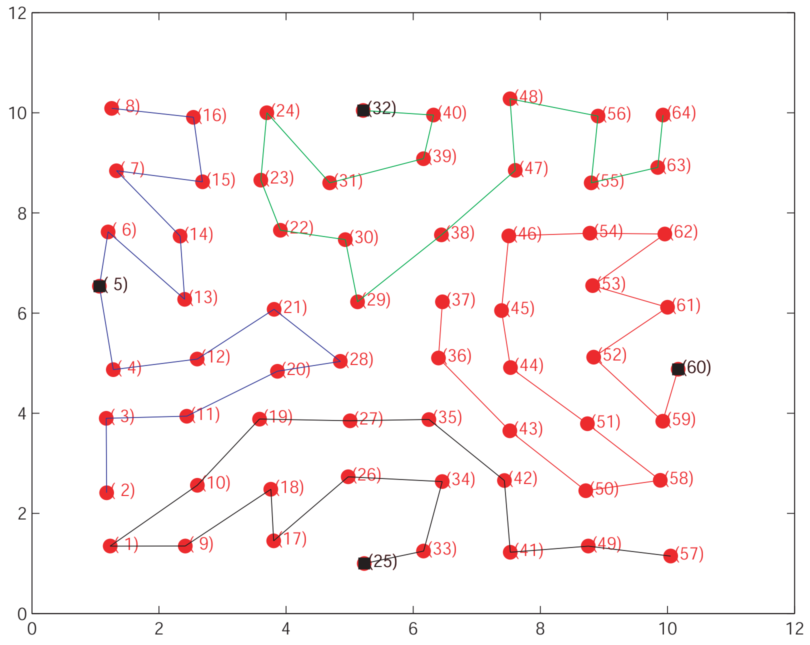
Figure 2: Final Paths

4.2 Algorithm II: Overview

The second algorithm involves the construction of an approximate Hamiltonian path in a distributed way. The idea is to first build a continuous "backbone" and successively provide an optimal partition of it. The proposed solution involves two steps:

- Approximate Hamiltonian Path Construction
- Path Partitioning

This algorithm, similar to the "Route First-Cluster Second" approach proposed in [45] for transportation scheduling, has the great advantage of providing an underlying structure for the obtained paths. This property turns out to be very interesting when investigating the robustness of the algorithm(considering a robot either joining or leaving the scene). In fact, the availability of a continuos "backbone" allows for an easy reconfiguration, i.e., modification of paths, by exploiting only local information.

4.2.1 Approximate Hamiltonian Path Construction

When the presence of the robots is sensed, the approximate Hamiltonian path construction process is triggered. A fully distributed algorithm which makes local decisions based on a heuristic approach is provided. Moreover, an additional step, namely the "Path Refining" process, is given as well in order to add nodes which have not been included into the path after the first step is run. Note that, the proposed algorithm builds an approximate Hamiltonian path, since the final path might not include all nodes due to the locality of the decision process. However, these nodes can be simply added to the paths as branches. This situation does not significantly influence the performance, as in practice it happens very rarely and involves a negligible percentage of nodes (less than 1%). The pseudo-code in Algorithm (3) shows a possible distributed implementation of the first step of the proposed approach, while the path refining process is the same as in Algorithm (2) without sorting the neighbors. Fig 3 shows a typical result of the Hamiltonian path construction procedure after the first step, while Fig 4 depicts the final path once the refinement has been run.

4.2.2 Path Partitioning

This second step aims to fairly distribute the duty among the robots so that each of them has to travel in average the same distance. The availability of a Hamiltonian path along with the knowledge of the number of robots

Algorithm 3: Approximate Hamiltonian path construction

```
$v \leftarrow$ Tail
$p \leftarrow$ Tail's predecessor
$v$ broadcasts request of availability to $\mathcal{N}(v)$
$\mathcal{N}_{av}(v) \leftarrow v$ gains availability responses
$d \leftarrow \infty$
$n \leftarrow 0$
for $j \in \mathcal{N}_{av}(v)$ do
    $v$ computes $d(i, j)$
    if $d(v, j) < d$ then
        $d = d(v, j)$
        $n = j$
    end
end
if $n = 0$ then
    $v$ notifies to $p$ the end of the path construction
else
    $v$ broadcasts the notification of attachment of $n$
    $v$ receives the acknowledgments of notification by $n$
end
```

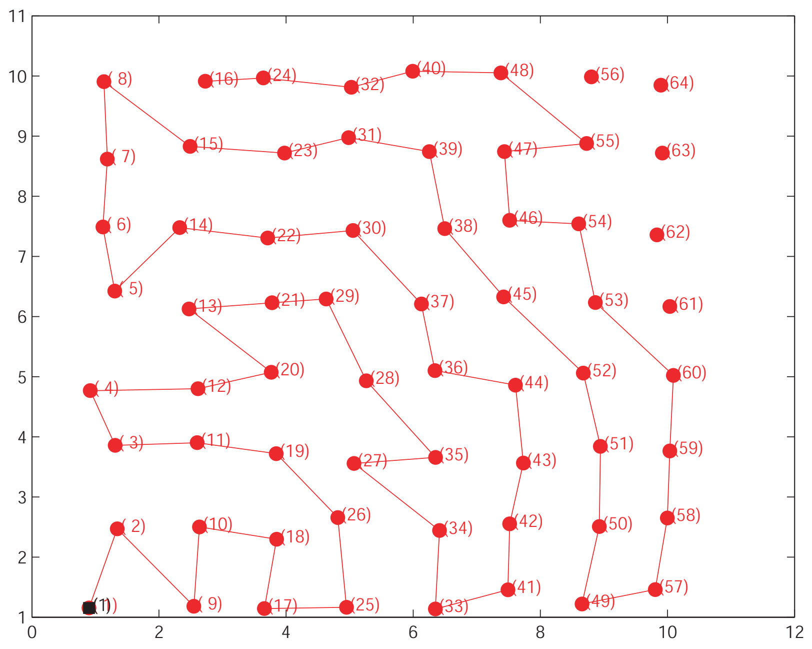
Figure 3: Approximate Hamiltonian Path construction

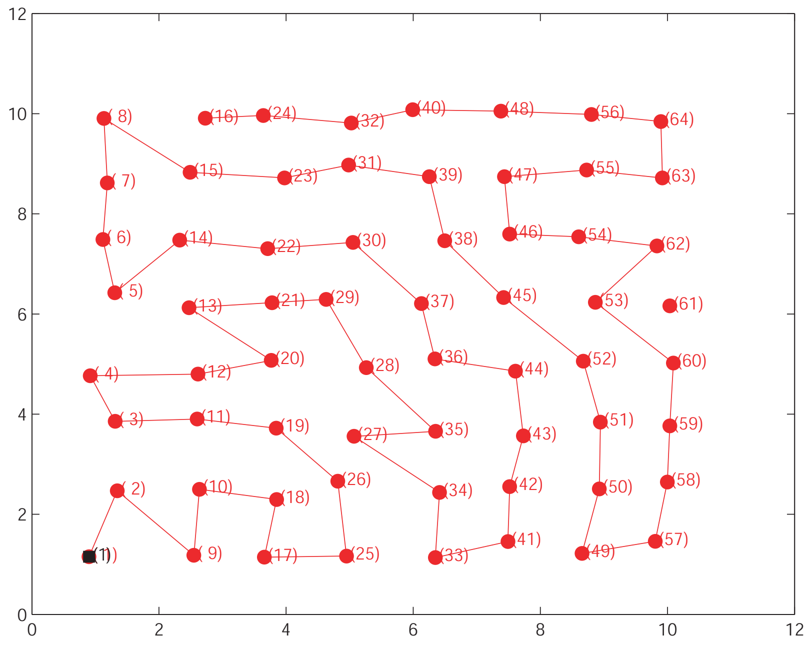
Figure 4: Refining Step

within the network make it possible to solve this process locally without the requirement of additional communication among nodes. In detail, the *i*-th node performs the following operation to find out which path it belongs to:

$$P_i = \begin{cases} \left\lceil \frac{h_i - N(\mathrm{mod}\ k)\cdot\left\lceil\frac{N}{k}\right\rceil}{\left\lfloor\frac{N}{K}\right\rfloor} \right\rceil + N(\mathrm{mod}\ k) & if \quad h_i > N(\mathrm{mod}\ k)\cdot\left\lceil\frac{N}{k}\right\rceil \\ \\ \left\lceil \frac{h_i}{\left\lceil\frac{N}{k}\right\rceil} \right\rceil & \text{otherwise} \end{cases} \tag{1}$$

where, P_i is an integer representing the path number, N is the number of nodes, k is the number of robots (paths), h_i is the position of the i-th node in the Hamiltonian path and, $\lceil x \rceil$ is the ceiling function which converts the argument x to the smallest integer not less than x, while $\lfloor x \rfloor$ is the floor function which converts the argument x to the highest integer less than or equal to x. The equation (1) is a consequence of the following observation about the division operator:

$$a = b \cdot c + d \tag{2}$$

$$a = (b - d) \cdot c + d \cdot (c + 1) \tag{3}$$

which simply says that a set of a nodes can be partitioned in d subsets of cardinality $(c + 1)$ and $(b - d)$ subsets of cardinality c. In this way, a simple but effective way to guarantee fairness when partitioning the Hamiltonian path is achieved.

5 Algorithmic Analysis

The proposed algorithms are made up by several steps. In this section, a deeper investigation for the most relevant aspects is provided.

5.1 Heads Selection

As the system is fully distributed and the paths are built concurrently, the selection of the heads turns out to be crucial for the performance of the algorithm. Note that, the effectiveness of a selection cannot be validated independently from the other steps. Indeed, it is strictly related to the local policy that has been adopted for the path construction. This underlines the complexity that arises when dealing with distributed algorithms.

Several approaches have been investigated, the following two turned out to be the most prominent:

- Selecting the heads roughly at the center of mass of the k regions in which the environment is supposed to be partitioned
- Selecting the heads equidistantly along the border of the network

Both strategies confirm the intuition for which having a natural partition of the environment allows each path to "grow" independently. In fact, a minimization of the interactions leads to a minimization of the packets that need to be sent over the network. This is explained by the fact that the majority of the traffic over the network is due to the negotiations of a node among different paths. Simulations have been performed to prove the effectiveness of both approaches. According to the results, the second technique turns out to scale better with the size of the network. Indeed, it was quite expected, as it was developed purposely to overcome the limitations of the first technique.

5.2 Local Policies for Path Construction

The area coverage problem has been proved to be NP-hard [6]. The distributed nature of the system does not help to make it easier. Therefore, the assumption of considering heuristics to solve the problem is reasonable. Here, the idea is to provide a local mechanism to effectively explore the environment.

At the beginning the following simple policies have been investigated:

- Naive Random Policy
- Closest-Neighbor Policy
- Closest-to-the-Head Policy
- Closest-to-the-Barycenter Policy

As expected, the random policy does not provide good results, though it might be considered as a lower bound for the performance of the system. Unfortunately, the policy based on the choice of the closest neighbor does not produce satisfying results either. This can be explained by the fact that, none of them provides any kind of cooperation. Note that, cooperation is meant in an "implicit" way. In other words, it should be achieved as an emerging phenomenon rather than by an explicit action. This motivation inspired the development of the last two policies, which showed to perform significantly better. In fact, as paths become "attractive", interactions get implicitly minimized (emerging cooperation).

However, the best performances have been achieved when considering a *hybrid* policy. More clearly, starting from the following considerations:

- Both the Closest-To-The-Head Policy and the Closest-To-The-Barycenter Policy produce paths which do not move too far away from the head. However, the paths tend to be a little bit too jagged

- The Closest-Neighbor Policy produces paths which are very straight. However, they tend to interfere with each other.

A solution, which introduces the concept of *virtual distance*, has been devised. Specifically, a virtual distance is defined as the linear combination of two factors:

$$Vd(x_1, x_2, x_3) = \alpha \cdot \|x_1 - x_2\| + (1 - \alpha) \cdot \|x_1 - x_3\| \tag{4}$$

where, $\{x_1, x_2, x_3\} \in \mathcal{R}^2$ represent three location, $\|x\|$ is the Euclidean metric and α is a tunable parameter. At this point, a flexible way of taking into account two different aspects when making local decisions is provided.

5.3 Crossing Links Formation

The crosslink formation has been thoroughly investigated for several problems, such as geographical routing ([46],[47]). Here, a simple solution, which turns out to be very effective for the particular structure of the problem, has been adopted. In order to explain it, consider the situation depicted in Fig. 5. According to it, nodes $\{A, B, C\}$ belong to the same path, while node D

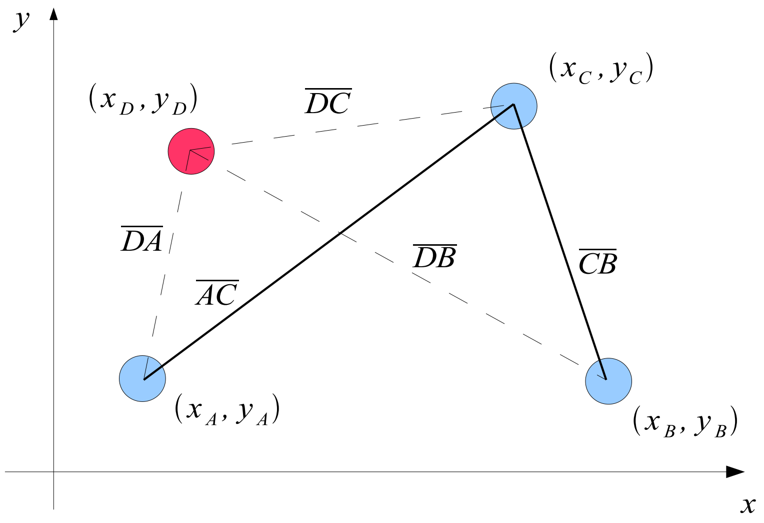

Figure 5: Crosslink situation

is an orphan. Moreover, the pair of segments $\overline{AC}, \overline{CB}$ represent the actual paths, while the pair of segments $\overline{DA}, \overline{DC}$ and $\overline{DC}, \overline{DB}$ describe the links that would be added if the orphan D were attached respectively to the pair of nodes $\{A, C\}$ or $\{B, C\}$. It is easy to verify that only one of the two options is "safe", namely attaching D to the pair $\{A, C\}$. The proposed solution leads to the safe decision, computing the difference between the lengths of the additional pair of links that would be added and the link that would be removed. The intuition behind this method is that the removal of long links coupled with the addition of short ones should result in lowering the chances to create a crosslink, at least from a probabilistic standpoint. In practice, in the case of the example it would be:

$$Sol = \underset{\{1,2\}}{\operatorname{argmin}} \{M_1, M_2\} \tag{5}$$

where

$$\begin{aligned} M_1 &= \overline{DA} + \overline{DC} - \overline{AC} \\ M_2 &= \overline{DB} + \overline{DC} - \overline{CB} \end{aligned} \tag{6}$$

represent the difference respectively for the first and second pair of nodes. Fig. 6 shows the correct attachment for the orphan D to the pair $\{A, C\}$.

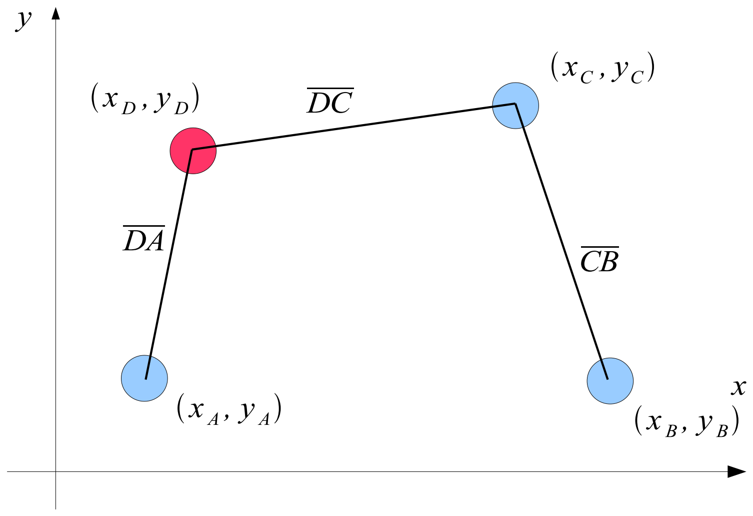

Figure 6: Crosslink solution

6 Analysis of Completeness, Fairness and Robustness

Some properties, namely completeness, fairness and robustness, have been introduced in Section 3, as indexes of quality to evaluate the effectiveness of an approach. Here, an investigation w.r.t. these indexes for both algorithms is provided.

6.1 Test Case

In order to evaluate the performance of the proposed algorithms, a statistical analysis was performed. A simulation environment developed under Matlab by the authors has been exploited. The analysis involved several configurations with an increasing number of nodes, $(64, 132, 256)$. Each configuration were tested considering an increasing number of robots as well, $(4, 6, 8)$. The connectivity graph was obtained according to the following parameters: percentage of connected node 90%, range of connectivity 15% of the longest distance between two nodes. Finally, for each configuration, 100 trials have been run.

6.2 Algorithm I

This algorithm allows to build paths concurrently exploiting the distributed nature of the sensor network.

6.2.1 Completeness

According to the simulation results, the algorithm is always able to guarantee *completeness* as long as there are no isolated nodes. Tab. 1. shows the percentage of "covered" nodes achieved with the first step, namely the Coarse Path Construction, and the second one, namely the Orphan Recovery Policy. Although, the algorithm cannot guarantee the full coverage of the

Table 1: Completeness: Success Percentage

Number of Nodes	Number of Robots	First Step [% Success]	Secont Step [% Success]
64	{4, 6, 8}	{82.06, 88.83, 88.75}	{100, 100, 100}
132	{4, 6, 8}	{81.44, 81.54, 81.43}	{100, 100, 100}
256	{4, 6, 8}	{86.00, 86.55, 88.10}	{100, 100, 100}

environment with only one step, the statistical analysis proved the network to be fully covered after the second step is run.

6.2.2 Fairness

The algorithm has been devised with the aim of balancing the duty among the robots. Although optimality cannot be guaranteed (in a deterministic way), statistical results, shown in Tab. 2, proves the effectiveness of the proposed solution. Note that, the ratio $\frac{Std}{Mean\,Value}$ is percent.

Table 2: Fairness

Number of Nodes	Number of Robots	Path Length Mean Value	$\frac{\text{Std}}{\text{Mean Value}}$
64	{4, 6, 8}	{16, 10.6, 8}	{10.83, 8.84, 16.54}
132	{4, 6, 8}	{33, 22, 16.5}	{6.94, 4.90, 5.25}
256	{4, 6, 8}	{64, 42.67, 32}	{5.41, 2.92, 3.49}

6.2.3 Robustness

This algorithm speeds up the construction of the paths by means of a concurrent approach. However, this is achieved exploiting locally a heuristic procedure. Consequently, the resulting paths do not have a common baseline. Therefore, providing a generic approach to properly modify them whether a robot leaves or joins the network, is far from being easy to realize. In practice, simulations proved that any time a robot joins or leaves the network, it is much more convenient to re-build from scratch the paths rather than trying to modify the pre-existing ones. Indeed, this is the main limitation of this algorithm, which is only partially reduced by the fact that the time required to build the k paths is significantly lowered by the concurrent approach.

6.3 Algorithm II

This algorithm first builds an approximate Hamiltonian path, and successively performs an optimal partition of it.

6.3.1 Completeness

As the algorithm builds a Hamiltonian path before to assign a duty to each robot, the completeness is structurally guaranteed as long as there are no isolated nodes. Specifically, Tab. 3 shows the percentage of nodes belonging to the approximate Hamiltonian path after the first and second step. In addition, the percentage of nodes attached as branches is provided as well. According to the statistical results, shown in Tab. 3, although the algorithm cannot be guaranteed to always provide the exact Hamiltonian Path, in practice it performs very well.

Table 3: Completeness: Success Percentage

Number of Nodes	First Step [% Success]	Second Step [% Success]	Isolated Branches [% Nodes]
64	85.19	100	0.11
132	85.23	100	0.08
256	82.04	100	0.07

Note that, the problem of computing a Hamiltonian path is hard, whether the nature of the system is distributed or not. Indeed, the formulation based on the Integer Linear Programming, provided in Appendix A.1, gives an evidence of this complexity. In fact, focusing the attention on the constraint (30), known in literature as "Sub-tour elimination constraint" [48], it can be easily recognized that the complexity of the problem grows exponentially, in terms of number of constraints added to the problem, w.r.t. the number of nodes.

Here, although simplicity and reduced computational complexity have been taken into account when devising this algorithm, the performance achieved is satisfactory. This is due to the particular nature of the adopted connectivity graph. Nevertheless, for the context in analysis, this assumption is reasonable, as the desired connectivity graph can always be achieved by means of an appropriate tuning of the communication range of the nodes. In practice, the real requirement is merely the availability of local connectivity between the neighbor as depicted in Fig. (7).

6.3.2 Fairness

The algorithm has been devised with the aim of equally distributing the duty among the robots. Moreover, unlike the previous approach, here the availability of an exact Hamiltonian path guarantees optimality in a deterministic way. However, as shown in Table 3, the Hamiltonian path, due to the distributed nature of the computation, might not be optimal.

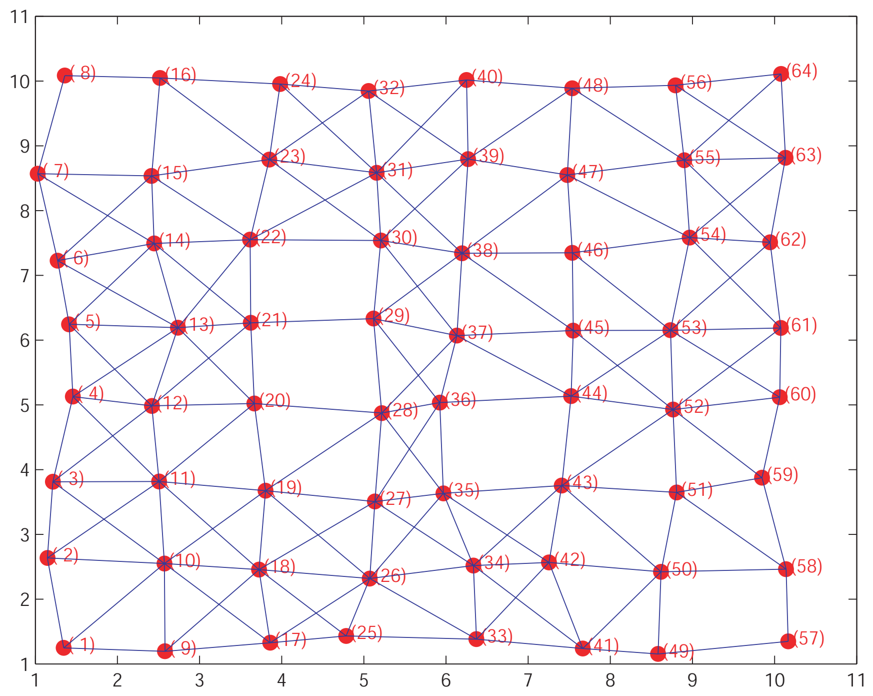

Figure 7: Connectivity

6.3.3 Robustness

The eventuality that a robot might either get broken or join the network can be handled in a systematic way. This is due to the fact that the availability of a Hamiltonian path provides a common baseline for the assignment of the paths. Moreover,both adding a path or deleting a pre-existing one are simply a specialization of the general rule provided in 4.2.2. Note that, another interesting aspect can be pointed out: the *reactivity* of the network, i.e. the time required for the network to re-organize itself anytime a robot gets broken or a new one joins the network. In fact, as soon as one of these events is recognized, the time required for the network to re-organize itself is not related to the re-construction of the paths, which is instantaneously given by the rule (1), but it is only related to the notification time, i.e., the time required to spread out the information across the network. This is indeed an interesting property of the algorithm.

7 Performance Analysis

In this section, an analysis about the the complexity in terms of computational and communication load, of both algorithms is provided.

7.1 Algorithm I

7.1.1 Computational Load

During the "Coarse Path Construction", according to the pseudo-code proposed in Algorithm (1), the dominant operation is the loop (lines: 7 – 13) in which the candidate to be added is chosen. The worst case is achieved when the current tail has to sent p times the request of attachment, where p is the degree of a node, i.e., the number of its neighbors. The related complexity is

$$\begin{aligned} C_1(p) &= O(p) + O(p-1) + \cdots + O(1) && (7) \\ C_1(p) &= O(p^2) && (8) \end{aligned}$$

where p is the max number of neighbors of each node. Note that, the pseudo-code provided in Algorithm (1) does not reflect the most efficient implementation. In fact, a priority queue might be introduced to easily lower the complexity to $O(p)$. However, this version has been preferred for sake of clarity.

During the "Orphan Recovery" step, according to the pseudo-code proposed in Algorithm (2), the dominant operation is the loop (lines: 7 – 22) in which the orphan selects the path to be added to. The related complexity is equal to $O(p)$ where p is the max degree of any node. The worst case is achieved when the request of attachment for the orphan has to be sent p times. Thus, the complexity turns out to be

$$C_2(p) = O(p^2). \tag{9}$$

Therefore, the overall computational complexity for the construction of each path is equal to:

$$\begin{aligned} C(N', p) &= N' \cdot [\alpha \cdot C_1(p) + (1-\alpha) \cdot C_2(p)] && (10) \\ C(N', p) &= O(N' \cdot p^2) && (11) \end{aligned}$$

where, $N' = \frac{N}{k}$ is the average number of nodes of a path, with N the number of nodes and k the number of robots, and α is the percentage of nodes covered at the first step, which can be retrieved by the statistical analysis.

7.1.2 Communication Load

During the "Coarse Path Construction", according to the pseudo-code proposed in Algorithm (1), the number of exchanged packets is determined by the number of neighbors p of a node (line 4). In detail:

$$P_1(p) = O(p). \tag{12}$$

During the "Orphan Recovery" step, according to the pseudo-code proposed in Algorithm (2), the number of exchanged packates is $O(p)$ for each attempt of attachment (line 3). In the worst case, p attempts are required in order to successfully attach an orphan (line 29). Consequently, the number of packets exchanged is:

$$P_2(p) = O(p^2). \tag{13}$$

Therefore, the overall number of packets required by the algorithm to build each path is given by:

$$\begin{aligned} P(N', p) &= N' \cdot [\alpha \cdot P_1(p) + (1 - \alpha) \cdot P_2(p)] && (14) \\ P(N', p) &= O(N' \cdot p^2) && (15) \end{aligned}$$

where, $N' = \frac{N}{k}$ is the average number of nodes of a path.

7.2 Algorithm II

7.2.1 Computational Load

During the "Approximate Hamiltonian Path Construction", according to the pseudo-code proposed in Algorithm (3), the dominant operation is the loop (lines: 7 – 13) in which the candidate to be added is chosen. The related complexity is

$$C_1(p) = O(p) \tag{16}$$

where p is the max number of neighbors of each node.

During the "Path Refining process", according to the pseudo-code proposed in Algorithm (2), the dominant operation is the loop (lines: 7 – 22) in which the orphan selects the path to be added to. The related complexity is equal to $O(p)$ where p is the max number of neighbors of each node. The worst case is achieved when the request of attachment for the orphan has to be sent p times. Thus, the complexity turns out to be

$$C_2(p) = O(p^2). \tag{17}$$

Therefore, the overall computational complexity is equal to:

$$\begin{aligned} C(N,p) &= N \cdot [\alpha \cdot C_1(p) + (1-\alpha) \cdot C_2(p)] \quad (18) \\ C(N,p) &= O(N \cdot p^2) \quad (19) \end{aligned}$$

where, N is the number of nodes and α is the percentage of nodes covered at the first step, which can be retrieved by the statistical analysis.

7.2.2 Communication Load

During the "Approximate Hamiltonian Path Construction", according to the pseudo-code proposed in Algorithm (3), the number of exchanged packets is determined by the number of neighbors p of a node (line 4). In detail:

$$P_1(p) = O(p). \quad (20)$$

During the "Path Refining process", according to the pseudo-code proposed in Algorithm (2), the number of exchanged packates is $O(p)$ for each attempt of attachment (line 3), where p is the number of neighbors of a node. In the worst case, p attempts are required in order to successfully attach an orphan (line 29). Consequently, the number of packets exchanged is:

$$P_2(p) = O(p^2). \quad (21)$$

Therefore, the overall number of packets required by the algorithm in order to build an approximate Hamiltonian path is given by:

$$\begin{aligned} P(N,p) &= N \cdot [\alpha \cdot P_1(p) + (1-\alpha) \cdot P_2(p)] \quad (22) \\ P(N,p) &= O(N \cdot p^2) \quad (23) \end{aligned}$$

where N is the number of nodes.

8 Conclusion

The robot area coverage problem has been investigated by the robotics research community through the years. At the beginning, formulations involving a single robot were proposed, while more recently teams of robots have been taken into account.

In this paper, a new formulation for the Multi-Robot Coverage problem is proposed. The novelty is the introduction of a sensor network, which cooperates with the robots in order to provide coordination to them. This is achieved taking advantage of the distributed nature of the sensor network. The sensor network is responsible for both the construction of the path and for guiding the robots.

This paper focuses the attention on the distributed construction of paths. Two different algorithms have been provided. The first one speeds up the construction process exploiting a concurrent approach. While the second guarantees an underlying structure for the paths building a Hamiltonian path and then partitioning it.

A statistical analysis involving several configurations with different numbers of nodes and robots has been provided to validate the effectiveness of the proposed approaches. Three indexes of quality, namely completeness, fairness and robustness, have been exploited to investigate the performances.

According to the statistical results, both approaches proved to perform well. Moreover, they were shown to be complementary in the sense that the strength of one approach was the weakness of the other and vice-versa. In fact, the first algorithm, although it speeds up the the path construction process, turns out not to be optimal in case whether a robot leaves or joins the network. Conversely, the second approach, although it takes more time for the Hamiltonian path construction, provides *reactivity* in case a robot either joins or leaves the network.

Several challenges still remain for future work. From a theoretical standpoint, an analytical investigation for the bounds of performances should be considered. Moreover, a wider definition of robustness involving the failure of the sensor network, in terms of both communication and functioning, should be taken into account. Finally, a real implementation of these approaches in order to validate their effectiveness in a real context should be provided.

9 Acknowledgment

The authors want to thank the anonymous reviewers for the helpful suggestions provided during the review process. In addition, Andrea Gasparri wants to thank Amitabha Ghosh for his suggestions at the beginning of the work.

References

[1] E. Arkin, S. Fekete, and J. Mitchell, “The lawnmower problem,” in *5th Canadian Conf. on Computational Geometry*, Waterloo, Ontario,, August 1993.

[2] J. Colegrave and A. Branch, "A case study of autonomous household vacuum cleaner," In AIAA/NASA CIRFFSS, 1994.

[3] D. W. Gage, "Randomized search strategies with imperfect sensors," in *Proc. SPIE Vol. 2058, p. 270-279, Mobile Robots VIII, Wendell H. Chun; William J. Wolfe; Eds.*, ser. Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference, W. H. Chun and W. J. Wolfe, Eds., vol. 2058, Feb. 1994, pp. 270–279.

[4] A. L. Pearce, P. E. Rybski, S. A. Stoeter, and N. Papanikolopoulos, "Dispersion behaviors for a team of multiple miniature robots," in *International Conference on Robotics and Automation, ICRA-2003*, Taipei, Taiwan, September 2003, pp. 1158 – 1163.

[5] H. Choset, "Coverage for robotics - a survey of recent results," *Annals of Mathematics and Artificial Intelligence*, vol. 31, no. 1, pp. 113 – 126, October 2001.

[6] E. M. Arkin, S. P. Fekete, and J. S. B. Mitchell, "Approximation algorithms for lawn mowing and milling," *Computational Geometry*, vol. 17, no. 1-2, pp. 25–50, 2000.

[7] H. Choset, "Coverage for robotics - a survey of recent results," *Annals of Mathematics and Artificial Intelligence*, vol. 31, pp. 113 – 126, 2001.

[8] Y. Gabriely and E. Rimon, "Spanning-tree based coverage of continuous areas by a mobile robot," *Annals of Mathematics and Artificial Intelligence*, vol. 31, no. 1-4, pp. 77–98, 2001.

[9] N. Hazon and G. A. Kaminka, "Redundancy, efficiency, and robustness in multi-robot coverage," in *ICRA 2005*, 2005.

[10] N. Agmon, N. Hazon, and G. A. Kaminka, "Constructing spanning trees for efficient multi-robot coverage," in *ICRA 2006*, 2006, p. In press.

[11] C. S. Kong, N. A. Peng, and I. Rekleitis, "Distributed coverage with multi-robot system," in *IEEE International Conference on Robotics and Automation. ICRA-2006.*, May 2006, pp. 2423–2429.

[12] S. G. Shuzhi and C. Fua, "Complete multi-robot coverage of unknown environments with minimum repeated coverage," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 2005, pp. 715 – 720.

[13] J. Rogge and D. Aeyels, "A novel strategy for exploration with multiple robots," in *Proceedings of the 4th International Conference On Informatics in Control, Automation and Robotics*, Angers, France, 2007.

[14] I. Rekleitis, V. Lee-Shue, A. P. New, and H. Choset, "Limited communication, multi-robot team based coverage," in *International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE*, vol. 4, 2004, pp. 3462–3468.

[15] D. Kurabayashi, J. Ota, and E. Yoshida, "An algorithm of dividing a work area to multiple mobile robots," in *IROS '95: Proceedings of the International Conference on Intelligent Robots and Systems-Volume 2*. Washington, DC, USA: IEEE Computer Society, 1995, p. 2286.

[16] T. W. Min and H. K. Yin, "A decentralized approach for cooperative sweeping by multiple mobile robots," in *International Conference on Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ*, 1998, pp. 380 –385.

[17] Z. Butler, A. Rizzi, and R. Hollis, "Distributed coverage of rectilinear environments," in *Proc. of the Workshop on the Algorithmic Foundations of Robotics*. A. K. Peters, January 2001.

[18] X. Zheng, S. Jain, S. Koenig, and D. Kempe, "Multi-robot forest coverage," in *IROS 2005*, 2005.

[19] I. Wagner, L. M, and A. Bruckstein, "Distributed covering by ant-robots using evaporating traces," *Robotics and Automation, IEEE Transactions on*, vol. 15, no. 5, pp. 918–933, October 1999.

[20] M. Batalin and G. Sukhatme, "Spreading out: A local approach to multi-robot coverage," in *6th International Symposium on Distributed Autonomous Robotic Systems*, 2002, pp. 373–382.

[21] L. Chaomin and S. Yang, "A real-time cooperative sweeping strategy for multiple cleaning robots," in *Proceedings of the 2002 IEEE International Symposium on Intelligent Control*, 2002, pp. 660 – 665.

[22] E. U. Acar and H. Choset, "Sensor-based coverage of unknown environments: Incremental construction of morse decompositions," *The International Journal of Robotics Research*, vol. 21, no. 4, pp. 345–366, April 2002.

[23] H. Choset and P. Pignon, “Coverage path planning: The boustrophedon cellular decomposition,” in *International Conference on Field and Service Robotics*, Canberra, Australia, 1997.

[24] Z. Butler, A. Rizzi, and R. Hollis, “Contact sensor-based coverage of rectilinear environments,” in *IEEE Int’l Symposium on Intelligent Control*, September 1999, pp. 266 – 271.

[25] J. Bang-Jensen and G. Gutin, Eds., *Digraphs: Theory, Algorithms and Applications.* Springer-Verlag, London Springer Monographs in Mathematics, 2000.

[26] T. Volgenant and R. Jonker, “A branch and bound algorithm for the symmetric traveling salesman problem based on the 1-tree relaxation,” *European Journal of Operations Research*, vol. 9, pp. 83 – 89, 1982.

[27] M. Diaby, “The traveling salesman problem: A linear programming formulation,” *WSEAS Transactions on Mathematics*, vol. 6, no. 6, pp. 745–754, June 2007.

[28] C. E. Miller, A. W. Tucker, and R. A. Zemlin, “Integer programming formulation of traveling salesman problems,” *J. ACM*, vol. 7, no. 4, pp. 326–329, 1960.

[29] L. L. Karg and G. L. Thompson, “A heuristic approach to traveling salesman problems,” *Management Sci.*, vol. 10, pp. 225 – 248, 1964.

[30] D. Rosenkrantz, R. Stearns, and P. L. I, “An analysis of several heuristics for the traveling salesman problem,” *SIAM Journal on Computing*, vol. 6, no. 3, pp. 563 – 581, 1977.

[31] N. Christofides, “Worst-case analysis of a new heuristic for the travelling salesman problem,” Graduate School of Industrial Administration, CMU, Tech. Rep. 388, 1976.

[32] G. Clarke and G. Wright, “Scheduling of vehicles from a central depot to number of delivery points,” *Operations Research*, vol. 12, pp. 568–581, 1964.

[33] F. Bock, “An algorithm for solving traveling-salesman and related network optimization problems,” 1965, unpublished manuscript associated with talk presented at the 14th ORSA National Meeting.

[34] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, “Optimization by simulated annealing,” *Science*, vol. 220, no. 4598, pp. 671–680, May 1983.

[35] H. Muhlenbein, M. Georges-Schleuter, and O. Kramer, “Evolution algorithms in combinatorial optimization,” *Parallel Computing*, vol. 7, 1988.

[36] M. Budinich, “Neural networks for the travelling salesman problem,” *J. Artif. Neural Netw.*, vol. 2, no. 4, pp. 431–435, 1995.

[37] M. Dorigo and L. M. Gambardella, “Ant colony system: A cooperative learning approach to the traveling salesman problem,” *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, April 1997.

[38] J. Allwright and D. Carpenter, “A distributed implementation of simulated annealing for the traveling salesman problem,” *Parallel Computing*, vol. 10, pp. 335 – 338, 1989.

[39] G. A. Sena, D. Megherbi, and G. Isern, “Implementation of a parallel genetic algorithm on a cluster of workstations: traveling salesman problem, a case study,” *Future Gener. Comput. Syst.*, vol. 17, no. 4, pp. 477–488, 2001.

[40] S. Tschöke, R. Lüling, and B. Monien, “Solving the traveling salesman problem with a distributed branch-and-bound algorithm on a 1024 processor network,” in *IPPS ’95: Proceedings of the 9th International Symposium on Parallel Processing*. Washington, DC, USA: IEEE Computer Society, 1995, pp. 182–189.

[41] T. Bektas, “The multiple traveling salesman problem: an overview of formulations and solution procedures,” *Omega: The International Journal of Management Science*, vol. 34, no. 3, pp. 209–129, 2006.

[42] M. Batalin and G. S. Sukhatme, “Coverage, exploration and deployment by a mobile robot and communication network,” *Telecommunication Systems Journal*, vol. 26, no. 2, pp. 181 – 196, 2004, special Issue on Wireless Sensor Networks.

[43] ——, “The design and analysis of an efficient local algorithm for coverage and exploration based on sensor network deployment,” *IEEE Transactions on Robotics*, vol. 23, no. 4, pp. 661 – 675, 2007.

[44] A. Gasparri, S. Panzieri, F. Pascucci, and G. Ulivi, “An interlaced kalm filter for sensors networks localization,” *International Journal of Sensor Networks (IJSNet)*, 2007, to appear in the Special Issue on Interdisciplinary Design of Algorithms and Protocols in Wireless Sensor Networks.

[45] J. Beasley, “Route first - cluster second methods for vehicle routing,” *Omega*, vol. 11, no. 4, pp. 403 – 408, 1983.

[46] Y. Kim, R. Govindan, B. Karp, and S. Shenker, “Geographic routing made practical,” in *NSDI’05: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*. Berkeley, CA, USA: USENIX Association, 2005, pp. 16–16.

[47] Y. J. Kim, R. Govindan, B. Karp, and S. Shenker, “Lazy Cross-Link Removal for Geographic Routing,” in *Proceedings of the ACM Conference on Embedded Networked Sensor Systems (Sensys)*, Boulder, Colorado, USA, November 2006.

[48] G. Laporte, “Generalized subtour elimination constraints and connectivity constraints,” *The Journal of the Operational Research Society*, vol. 37, no. 5, pp. 509–514, May 1986.

A Centralize Formulation as ILP

A.1 Simple Partitioning

Given a graph $G = (N, L)$ with N the set of nodes (with cardinality n) and L the connectivity matrix (with cardinality l), $H \in N$ the set of sinks (with cardinality k) and R the set of partition’s indexes (always with cardinality k). The simple partitioning problem can be formulated as follows:

$$
\begin{aligned}
&\mathit{min} && 1 = 1 \\
&s.t. && \sum_k \sum_i p_{ij}^h = 1 \qquad \forall j \in N \setminus H && (24)\\
& && \sum_k \sum_i p_{ij}^h = 0 \qquad \forall j \in H && (25)\\
& && \sum_k \sum_i p_{ji}^h = 1 \qquad \forall j \in H && (26)\\
& && \sum_{i \in H} \sum_{j \in \mathcal{N}(i)} p_{ij}^h = 1 \quad \forall h \in R && (27)\\
& && \sum_{h \in \mathcal{N}(j)} p_{jh}^h \leq \sum_{i \in \mathcal{N}(j)} p_{ij}^h \qquad \forall j \in N \setminus H, \quad \forall h \in R && (28)\\
& && \sum_k p_{ij}^h + p_{ji}^h \leq 1 \qquad \forall (i,j) \in L && (29)\\
& && \sum_{(i,j) \in L \cap V} p_{ij}^h \leq |V| - 1 \qquad \forall V \subset N \setminus H, V \neq \emptyset, \forall h \in R && (30)\\
& && \left\lfloor \frac{N}{k} \right\rfloor - 1 \leq \sum_{(i,j) \in L} p_{ij}^h \leq \left\lceil \frac{N}{k} \right\rceil - 1 \qquad \forall h \in R && (31)
\end{aligned}
$$

where,

$$
p_{(ij)}^h = \begin{cases} 1 & \text{If the link } (i,h) \text{ belong to the path } h, \\ 0 & \text{otherwise.} \end{cases} \tag{32}
$$

In detail, the (24) constrains each node to belong only to one path, the (25) and (26) are boundary conditions for the heads. The (27) forces the heads to belong to different paths, the (28) is similar to the *flow conservation* constraint of network flows problem. Together with constraint (24), it forces each node $j \in N \setminus H$ to have at most one outgoing link (note that sources will have only an incoming link). The (29) avoids the paths to go backward, the 30, known in literature as *Sub-tour elimination constraint* avoids to have isolated cycles. Finally the (31) implies that all paths will have either length $\lfloor \frac{N}{k} \rfloor$ or $\lfloor \frac{N}{k} \rfloor + 1$.

The bogus objective function points out that any feasible solution is inherently optimal for this formulation. If the distance between adjacent nodes is not considered constant, this formulation can be slightly modified to account the minimization of the overall length of the paths. Additionally, note that this problem might not be feasible if the connectivity matrix is not carefully chosen. However, for a sensor network this problem can be easily overcome by properly tuning the range of connectivity among nodes.

A.2 Shortest Paths Partitioning

In order to minimize the overall length of the paths, the simple partitioning formulation can be modified by introducing a distance matrix D of Euclidean distances between pairs of nodes along with a real objective function.

$$
\begin{aligned}
min \quad & \sum_{k}^{R} \sum_{(i,j)\in L} d_{ij} \cdot p_{ij}^{k} \\
s.t. \quad & \sum_{k} \sum_{i} p_{ij}^{k} = 1 \qquad \forall j \in N \setminus H && (33)\\
& \sum_{k} \sum_{i} p_{ij}^{k} = 0 \qquad \forall j \in H && (34)\\
& \sum_{k} \sum_{i} p_{ji}^{k} = 1 \qquad \forall j \in H && (35)\\
& \sum_{i\in H} \sum_{j\in \mathcal{N}(i)} p_{ij}^{k} = 1 \qquad \forall k \in R && (36)\\
& \sum_{h\in \mathcal{N}(j)} p_{jh}^{k} \leq \sum_{i\in \mathcal{N}(j)} p_{ij}^{k} \qquad \forall j \in N \setminus H, \quad \forall k \in R && (37)\\
& \sum_{k} p_{ij}^{k} + p_{ji}^{k} \leq 1 \qquad \forall (i,j) \in L && (38)\\
& \sum_{(i,j)\in L\cap V} p_{ij}^{k} \leq |V| - 1 \qquad \forall V \subset N, V \neq \emptyset,\ \forall k \in R && (39)\\
& \left\lfloor \frac{N}{k} \right\rfloor - 1 \leq \sum_{(i,j)\in L} p_{ij}^{k} \leq \left\lceil \frac{N}{k} \right\rceil - 1 \qquad \forall k \in R && (40)
\end{aligned}
$$

In detail, the cost of a path is defined as the sum of the Euclidian distances between successive nodes belonging to this path.