SyWaP: Synchronized Wavefront Propagation for multi-robot assignment of spatially-situated tasks

Antoine Bautin, Olivier Simonin*, François Charpillet
Inria Nancy Grand Est, MAIA team
Université de Lorraine, LORIA (UMR 7503)
CNRS, LORIA (UMR 7503)
firstname.lastname@inria.fr

***Abstract*—Task assignment is paramount for the efficiency of multi-robot missions. In the context of multi-robot exploration of unknown environments, tasks can be identified as targets to reach in order to expand knowledge of the environment. These targets can be frontiers between known reachable area and unknown areas or a best view configuration to observe each frontiers. Once these tasks are identified, they are assigned to robots. Standard approaches work in a two step process where the costs of reaching targets are computed before running a task allocation algorithm. In this article, we combine these two steps to propose an efficient task allocation algorithm. This algorithm can take into account different assignment criteria such as the rank of a robot in a list of robots ordered by cost.**

I. Introduction

In this article we focus on multi-robot exploration but the task allocation algorithm may be used for other Spatially-Situated Tasks (for example, pick-up and deliver an object, escort a visitor).

One advantage of multi-robot exploration over mono-robot is to require less time to fully discover an unknown environment [1]. An exploration strategy can aim at minimizing time or at minimizing uncertainty in the map and on robot localization. In both cases task identification and assignment are the key to efficiency.

Our exploration strategy aims at minimizing exploration time to fully explore an unknown environment (time to build a complete map of the environment). For this, each robot is assigned a target in order to distribute the robots in the environment and explore simultaneously different areas.

Using Gerkey et al. taxonomy [2], we consider the target assignment problem in multi-robot exploration as Single Tasks, Single Robot, Instantaneous Assignment (ST-SR-IA):

- ST because robots can only execute one task at a time i.e. a robot can only explore one frontier at a time,
- SR because only one robot is needed to perform a task i.e. one robot is enough to explore a frontier,
- IA because tasks have unknown duration and during exploration new tasks (targets) are discovered.

We assume the robot team to be homogeneous i.e. no robot is more fit for a task than another. Also, tasks are not prioritized (in the exploration context, the utility of exploring each frontier is equal). For exploration, we consider that the cost of a task is equal to the travel distance to the target which is itself proportional to the travel time. We therefore compute the path distances in between robot and targets.

*Now at INSA Lyon - CITI-Inria Lab.

Task assignment can then be summarized in two steps:

- identify the tasks, such as targets or configurations to reach in order to increase knowledge of the environment,
- assign a target to each robot based on different criteria including the cost of each task (for exploration, path distance between robot-target pairs).

The evaluation of costs is computationally very expensive. We address this issue using a wavefront propagation algorithm because it allows to compute multiple distance with one propagation (comparable to a multi-objective A* algorithm). The proposed SyWaP approach consists in synchronizing the propagation of multiple wavefronts based on propagating distance and position of the other robots in the environment. This enables us to stop the wavefronts propagations quickly therefore saving computation time. SyWaP works with different assignment criteria. In order to demonstrate the validity of this approach we show results of its application with two different assignment criteria.

The main contribution of this paper is an algorithm to efficiently compute assignments of spatially situated tasks in ST-SR-IA problems with homogeneous robots and tasks of equal priorities.

II. Previous work

This section presents the previous work in multi-robot target assignment for exploration.

A. Target identification

For exploration, a task is either a target or a configuration to reach that will allow the robot team to discover new areas.

Frontiers [3] are commonly used as targets for exploration. They are the border between known and unknown zones. A robot going towards a frontier discovers new areas thus creating new frontiers. Repeating this process until no frontiers are available guarantees a complete exploration. For mono-robot, this strategy is very efficient (see [4]).

Another approach consists in the sampling of configurations near the frontier [5] to find the next best view (observation configuration). This technique is used in mono-robot but is less efficient than frontier assignment.

Gossage et al. [6] and Wurm et al. [7] use a Voronoï discretization. The nodes of the built topological map are associated with the Voronoï cells. Robots are assigned to unexplored cells. These methods are similar to the utility concept of the clustering of frontiers cells because they avoid assignment of multiple robots to nearby frontiers.

Grouping of frontiers allows the identification of a number of groups equal to the number of robots. In Faigl et al. [8], clustering is done using a K-means variant algorithm. One group is then assigned to each frontier. Initialization of the K-means algorithm is done by positioning a cluster center on each robot. This approach allows a good distribution of robots because it creates groups of frontiers depending on their respective distance. However, using this approach, group size can be inhomogeneous and observation of a target usually leads to the creation of a nearby target (frontiers are pushed back). The robot which explored the previous target is best suited to explore the new one and will therefore be assigned to it making the plan to explore multiple targets obsolete.

B. Cost computation

Task assignment algorithm usually uses a cost matrix (distance for each robot-target pair). To build it, path distances between robots and frontiers needs to be computed. The A* algorithm is optimal to compute a path using an adequate heuristic. However, in order to compute multiple paths it has to be adapted. An A* algorithm with an heuristic equal to zero is equivalent to a Dijkstra. On regular grids this algorithm is often called a wavefront propagation [9]. We compare both approaches in section IV.

C. Target assignment

An extension to multi-robot of the original frontier algorithm was proposed in [10]. Each robot is assigned to its nearest target and is called *MinDist* hereafter. This system is said to feature implicit cooperation because robot cooperation is achieved only by cooperatively building a map. However, in some situation robots tend to explore the same areas.

Most task exploration strategies are based on a greedy assignment of target to robots ([11], [12] [13]). It aims at minimizing the total cost of the assignments while guaranteeing a balanced distribution of robots among frontiers. At each iteration the robot-target pair with the lowest cost is assigned. This algorithm is usually applied in a centralized manner but different decentralized systems have been proposed. It can be decentralized [13] using, for example, a biding algorithm where each robot computes its cost towards every available target and broadcasts it. The robot with the highest bid is assigned to the target. If all costs are available then, another form of decentralization is for each robot to execute the assignment algorithm separately.

The Hungarian method [14] is a combinatorial optimization algorithm. It is executed on the full cost matrix and solves optimally the sum of cost minimization. Its complexity is in $O(n^3)$ where n is the maximum number of robots and targets. It is adapted for multi-robot exploration and has often been used [7] [15]. Yet it requires the full cost matrix computation and is usually used in a centralized system.

The utility concept introduced by [12] allows to combine the assignment process with the identification of targets that will be assigned. Utility is a difference or a ratio between an estimated information gain and a cost. Frontier-robot pairs are greedily assigned based on the highest utility knowing the already assigned robots. For example, Burgard et al . [11] sets the estimated information gain of each target to 1 and at each assignment the estimated information gain of visible frontier is reduced in a inversely proportional manner to their inter-distance.

The *MinPos* algorithm is an alternative to assignment based on distance and information gain [16]. It uses the concept of rank of a robot towards a frontier equal to the number of other robot closer to that frontier. *MinPos* assigns robots to the target where they have the lowest rank. If the robot has the same rank towards multiple frontiers it is assigned to the nearest of them. This assignment favors a good distribution of robots in the environment and performance for exploration has been shown to be better or competitive with Greedy assignment based strategies.

The assignment process can take place using different communication strategies. A centralized strategy allow an efficient cooperation but is not scalable mainly because it requires communication between the robots and the central agent. Explicit communication allows an efficient cooperation but induces a communication overhead.

Implicit coordination, as proposed in [10], is a good compromise between cooperation and communication but its performance for exploration is poor compared to the *MinPos*, Greedy or Hungarian assignments. We have used implicit coordination with *MinPos* and Greedy algorithm in [16]. The strategy used here is based on implicit coordination and allows each robot to compute efficiently its next target based on a global map and other robot locations.

III. Framework

In this paper, the robot team is assumed to be homogeneous i.e. robots are identical. It is a simplifying assumption that brings robustness: every robot can replace another one. Also, with heterogeneous teams, task allocation is different as it needs to take into account individual properties of each robot (kinematic and dynamical properties as well as observational capacities).

Robot are autonomous and as communication in between robots can be unreliable we have chosen to use a decentralized decision making system. To this end, as the global task in common (exploration of the whole environment) each robot needs to know which areas have been explored to avoid redundancies. Robots cooperate to build a global map by broadcasting their individual maps periodically in order to share their individual observations. Each robot then decides autonomously its next target based on the information available to it at this point, but no explicit cooperation communication is used. Communication is important for a good cooperation.

All communication overhead in the cooperation strategy will impact scalability and robustness. Besides the collaborative global map building, robots communicate frequently their locations which is very lightweight information. This framework is inherently asynchronous : robots do not wait for other robots to reach their target, they compute a new target every time a threshold distance has been traveled or as soon as the previous target has been reached.

Figure 1 illustrates the pipeline of an individual robot in a multi-robot system. This article focuses on the leftmost step (in yellow) the target identification and assignment step. Once the robot has found its assigned target is computes a path or a trajectory to reach it. Path or trajectory planning is not in scope of this paper but note that using the *MinPos* and Greedy assignment, there are few conflicts in between robots due to the distribution of robot in the environment. The planned trajectory is then followed until the target is reached, an unexpected event happens or a threshold distance has been traveled. During the execution of the trajectory, the robot build a map of the newly discovered areas. The next step after traveling, is to communicate to build a similar map on every robot. This is described in the next section.

A. Multi-robot mapping

Each robot executes its SLAM (Simultaneous Localization and Mapping) algorithm separately. We use a the SLAM algorithm from Lucidarme et al. [17] where mapping is done using an occupancy grid [18]. From this probabilistic occupancy grid a coarse deterministic grid is computed containing exploration and mapping information (states of cells are free, occupied or unknown). This grid is sent and received between robots. Robots receiving a map merges it with its map to compute its assignment and trajectory. Due to the lower resolution of exchanged grids, the SLAM is accurate enough to merge individual robot maps without correction.

Communication: In order to achieve a good cooperation i.e. to avoid re-visiting places already explored by another robot, robots collaboratively build a global map by sending their individual maps periodically (at least every time they reached their target). In addition to that, robots communicate their position.

B. Target identification

From our point of view, frontiers are tasks of unknown duration because the time to fully explore a frontier (push it back until it disappears) is unknown We therefore think it is not interesting to assign multiple targets to one robot.

To reduce the computation cost, robots are assumed to have a circular shape and to be holonomic. A configuration space grid $\mathcal{C}$ is then simply computed by enlarging obstacles by the size of the robot resulting in a grid composed of $\mathcal{C}_{free}, \mathcal{C}_{obs}, \mathcal{C}_{unkown}$. Frontiers cell are then identified as $\mathcal{C}_{free}$ cells neighboring a $\mathcal{C}_{unkown}$ cell. In this work, we group contiguous frontier cells and define the target for this frontier as the cell in the group which is nearest to group's center of mass. Note that observation locations or observation configurations can be used as targets without modifying the algorithms proposed. This is similar to the utility approach because it prevents assignment of multiple robots to nearby frontiers although it does not consider the visibility in between non-contiguous frontiers.

Once the robot has identified these targets, the next step is to find the one it is assigned to.

IV. Synchronized wavefront propagation

Cost evaluation is computationally expensive, in the approach proposed hereafter only the cost between the assigned robot-frontier pair is computed.

A. Wavefront propagation algorithm

The wavefront propagation algorithm [9] builds an artificial potential field with only one minimum: the source of propagation. It consists of incrementally computing the set of cells located at the same distance [1] from the source on the configuration grid. This set is the wavefront. The source is initialized at 0. Let $\mathcal{W}_i$ be the set of cells located at distance i and $\mathcal{X}_G$ the source/target cells of potential 0. Algorithm 1 specifies this wavefront computation. This algorithm complexity is in $O(n)$ where n is the number of reachable cells from the source.

Algorithm 1 Wavefront propagation

```
1: Init $\mathcal{W}_0 = \mathcal{X}_G; i = 0$
2: Init $\mathcal{W}_{i+1} = \varnothing$
3: for all $x \in \mathcal{W}_i$ do
4:     $\phi(x) = i$
5:     Insert all neighboring cells $y \in \mathcal{C}_{free}$ of $x$ in $\mathcal{W}_{i+1}$
6: end for
7: if $\mathcal{W}_{i+1} = \varnothing$ Terminate else i=i+1 et go to step 2
```

To compare the wavefront propagation algorithm, we compute the cost between a target and a varying number of robots. Figure 2 illustrates computation time of this comparison.

- a single target A* algorithm which computes a path (and its cost) from a frontier to a robot and redoing this for every robot.
- a multi-target A* uses the heuristic equal to the minimum euclidean among all robots until it is reached and removed from the list.

The A* algorithm efficiency is linked to the chosen heuristic and to the size of the environment and the number and shape of the obstacles (for example, the euclidean heuristic is further from the path distance in maze environments). For the comparison, we have chosen a typical office map composed of 14 rooms and a million pixels.

B. Synchronized wavefront propagation

We have developed a wavefront propagation algorithm allowing to propagate wavefronts in steps. Iterating over the wavefronts and propagating each one of a step wavefront propagate at the same speed. When one wavefront reaches a robot location two cases are considered:

- if the encountered robot is the robot computing its assignment, computation is finished. The robot is assigned to the frontier associated with the wavefront.

[1] distance of the path to reach the source

Fig. 1: Exploration pipeline for a robot in our multi-robot system

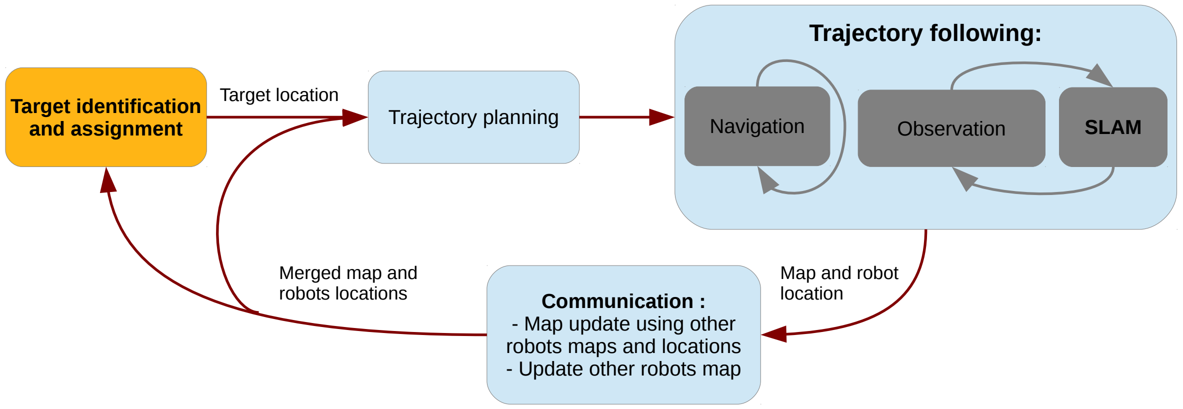
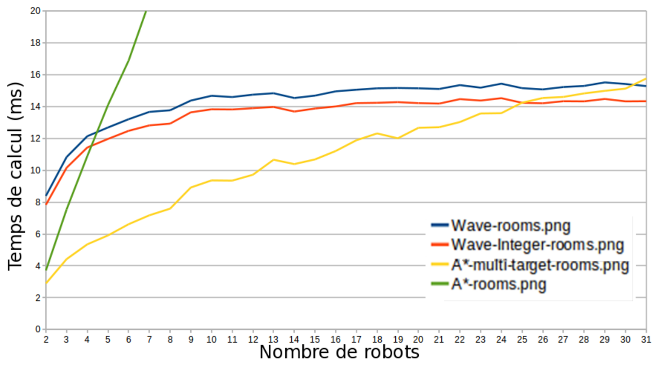

Target identification and assignment
Target location
Trajectory planning
Trajectory following:
Navigation
Observation
SLAM
Merged map and robots locations
Map and robot location
Communication :
- Map update using other robots maps and locations
- Update other robots map

Fig. 2: A* vs wavefront for a increasing number of robots



- otherwise, propagation of this wave is paused.
  If all wavefront are paused, they are all restarted sequentially, starting with the one that has covered the lowest distance. So that the covered distance by each wavefront is equal.
  This way, the first wavefront that reaches the robot is the one satisfying the assignment criteria.

$\mathbf{Pot}(\mathcal{W}^j)$ is the distance reached by the wavefront associated with frontier j after the last propagation step,

Algorithm 2 Propagate function

```
1: Input : Wavefront $\mathcal{W}$
2: $x = POP(\mathcal{W})$
3: $startPot = \phi(x)$
4: **while** $\phi(x) - startPot < step \ \& \ \mathcal{W} \neq \varnothing$ **do**
5:     **for all** Neighbors $y$ of $x \in \mathcal{C}_{free}$ **do**
6:         $\phi(y) = NeighborhoodCostFunction(x, y)$
7:         $PUSH(y, W)$
8:     **end for**
9:     $x = POP(\mathcal{W})$
10: **end while**
```

In order to synchronize wavefronts on the distance reached, each wavefront (satisfying the rank criteria) is sequentially propagated of a step. The *Propagate* function (algorithm 2) used in the algorithm 3 and 4 is used to do this. In this algorithm, $NeighborhoodCostFunction(x, y)$ computes the transition cost between x and y. Propagation is done with a neighborhood function that uses a $\sqrt{2}$ approximation. and adds neighboring cells in a priority queue to process the nearest cell first (POP extracts that cell, and PUSH adds a new cell in the queue). The size of the step depends on the required precision (see figure 3).

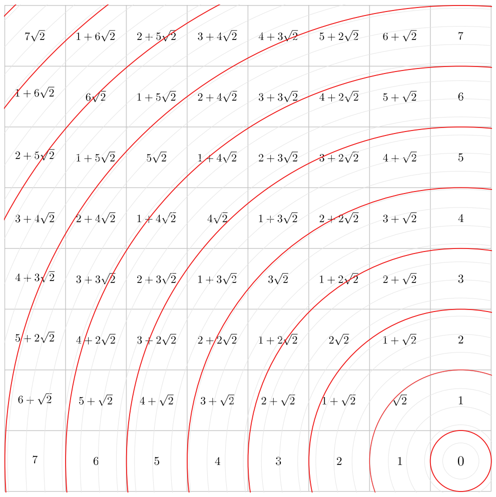

$7\sqrt{2}$ $1+6\sqrt{2}$ $2+5\sqrt{2}$ $3+4\sqrt{2}$ $4+3\sqrt{2}$ $5+2\sqrt{2}$ $6+\sqrt{2}$ 7
$1+6\sqrt{2}$ $6\sqrt{2}$ $1+5\sqrt{2}$ $2+4\sqrt{2}$ $3+3\sqrt{2}$ $4+2\sqrt{2}$ $5+\sqrt{2}$ 6
$2+5\sqrt{2}$ $1+5\sqrt{2}$ $5\sqrt{2}$ $1+4\sqrt{2}$ $2+3\sqrt{2}$ $3+2\sqrt{2}$ $4+\sqrt{2}$ 5
$3+4\sqrt{2}$ $2+4\sqrt{2}$ $1+4\sqrt{2}$ $4\sqrt{2}$ $1+3\sqrt{2}$ $2+2\sqrt{2}$ $3+\sqrt{2}$ 4
$4+3\sqrt{2}$ $3+3\sqrt{2}$ $2+3\sqrt{2}$ $1+3\sqrt{2}$ $3\sqrt{2}$ $1+2\sqrt{2}$ $2+\sqrt{2}$ 3
$5+2\sqrt{2}$ $4+2\sqrt{2}$ $3+2\sqrt{2}$ $2+2\sqrt{2}$ $1+2\sqrt{2}$ $2\sqrt{2}$ $1+\sqrt{2}$ 2
$6+\sqrt{2}$ $5+\sqrt{2}$ $4+\sqrt{2}$ $3+\sqrt{2}$ $2+\sqrt{2}$ $1+\sqrt{2}$ $\sqrt{2}$ 1
7 6 5 4 3 2 1 0

Fig. 3: Wavefront propagation steps necessary for synchronization (a step is the execution of the *Propagate* function)

Before detailing the SyWaP approach for the *MinPos* and Greedy algorithm, we here give an example for the *MinDist* algorithm to explain the main idea behind SyWaP. Recall that *MinDist* assigns the robot to its nearest frontier. To find this frontier, a single wavefront can be propagated from the robot position and stopped as soon it encounters a target location ; the encountered target is the nearest and therefore the robot assignment (this is how we implemented it).

Wavefronts can also be propagated from every target until they reach the robot location. At this point distances to every target can compared to determine the nearest. The SyWaP idea is to propagate all the wavefronts at the same speed (synchronized on propagation distance). Then, the first wavefront that reaches the robot is the one originating from the nearest frontier. All wavefront propagation can then be stopped because the robot has computed its assignment. Only one wavefront has been propagated to the robot location thus saving computation time.

Figure 4 illustrates the propagation of 3 waves stopped when the first one reaches the robot location (big white disc). The wavefront originating from the orange frontier the robot is assigned to the frontier cell representing the frontier (small black disc).

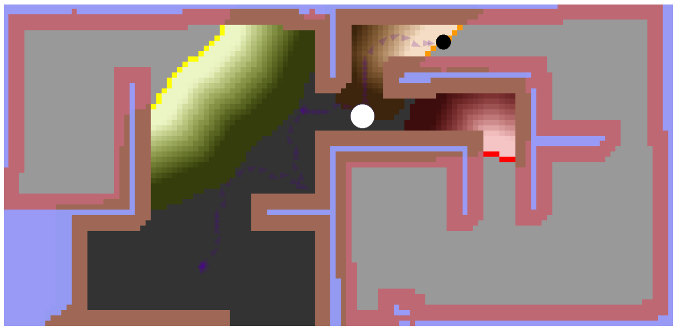


Fig. 4: Group of frontier cell and wavefront propagation

We apply the same idea to the *MinPos* algorithm, by not only synchronizing wavefronts on propagation distance but on the uppermost criteria of the number of robot encountered by the wavefront (thus computing its rank towards the frontier). This new algorithm is detailed in the next section.

C. MinPos SyWaP algorithm

Recall that *MinPos* assigns the robot (computing its assignment) to the target for which there is the lowest rank i.e. the number of robots closer to that frontier. If the robot has the same rank towards multiple frontiers it is assigned to the nearest of them.

For the *MinPos*, it is not necessary to propagate all wavefronts until they reach the robot computing its assignment. Indeed, the SyWaP approach allows to first synchronize the wavefronts on the number of robots encountered then on the distance covered.

Figure 5 illustrates the computation of synchronized wavefront propagation (fig. 5 right) compared to the propagation of wavefronts until each one has reached the robot computing its assignment (fig. 5 left). Wavefronts are represented by the color of its associated frontier. Luminosity indicates the distance from the frontier. We can observe that propagation distances using SyWaP are small compared to sequential propagation.

Algorithm 3 consists of creating the set $\mathcal{W}_{mr}$ (*Min Rank Wavefronts*) containing the wavefronts which have encountered the minimum number of robots. From this set, the wavefront which has covered the less distance (which have the minimum

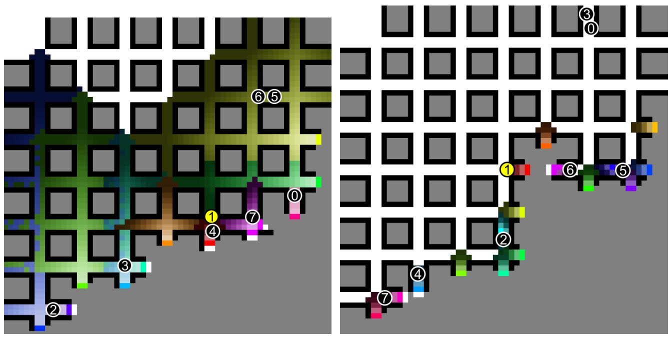


Fig. 5: Comparison of propagated waves using *MinPos* algorithm when stopping propagation on the robot computing its assignment or using SyWaP.

potential) is propagated of a step. This operation is repeated until the robot computing its assignment is reached by a wavefront or until no wavefront can be propagated (all cells reachable from the frontier have been covered). In this last case, the robot has no possible assignment, all the frontiers are inaccessible and exploration is finished.

We here introduce the notations used in the algorithms $\mathcal{W}$ the set of all wavefront,
$\mathcal{W}_0^j$ the set of cells belonging to frontier $\mathcal{F}_j$,
$\mathcal{W}_d^j$ the set of cells in the wavefront associated with frontier $\mathcal{F}_j$ at distance d,
$\mathbf{Rank}(\mathcal{W}^j)$ the number of robot encountered by the wavefront associated with frontier j after the last propagation step,
$\mathbf{Pot}(\mathcal{W}^j)$ the distance reached by the wavefront associated with frontier j after the last propagation step,
$\mathbf{Cell}(\mathcal{R}_i)$ the cell occupied by robot i.
$\mathbf{A}(\mathcal{R}_i)$ assignment of robot i, $\mathbf{A}(\mathcal{R}_i) \in \{\varnothing \cup \mathcal{F}\}$,

D. Greedy SyWaP algorithm

The Greedy algorithm using SyWaP works in the same manner as *MinPos* SyWaP. Instead of building a set of wavefronts having encountered the least robots, the algorithm builds $\mathcal{W}_{ma}$ the set of robot having encountered the least unassigned robot. An unassigned robot becomes assigned when a wavefront reaches its position. Wavefronts are synchronized on the number of unassigned robot encountered. We therefore add a counter to each wavefront counting the number of robots assigned to its associated target.
$\mathbf{Assigned}(\mathcal{W}^j)$ is the number of assigned robot to frontier $\mathcal{F}_j$. Algorithm 4 details the Greedy SyWaP algorithm.

V. Results

To validate and measure performance of the SyWaP approach we have developed a simulator using the framework described in section III. The algorithm compared are the *MinDist* algorithm, *MinPos* and Greedy algorithm with SyWaP, and *MinPos* and Greedy using the whole cost matrix computed by propagating wavefronts on the whole explored environment.

In order to validate the developed algorithm we compared their performance for the exploration. Figure 6 illustrates the

Algorithm 3 *MinPos* SyWaP algorithm

```
1:  Input: $\mathcal{F}$, $\mathcal{R}$, $\mathcal{CG}_{rid}$
2:  Output: Assignment of robot $\mathcal{R}_a$
3:  $\mathcal{W} = \mathcal{F}$
4:  while $\mathbf{A}(\mathcal{R}_a) = \varnothing$ & $\mathcal{W} \neq \varnothing$ do
5:      $\mathcal{W}_{mr} = \operatorname*{argmin}_{\mathcal{W}^k \in \mathcal{W}} \mathbf{Rank}(\mathcal{W}^k)$
6:      $\mathcal{W}_{mrmd} = \operatorname*{argmin}_{\mathcal{W}^k \in \mathcal{W}_{mr}} \mathbf{Pot}(\mathcal{W}^k)$
7:      for all $\mathcal{W}^j \in \mathcal{W}_{mrmd}$ do
8:          $d = \mathbf{Pot}(\mathcal{W}^j)$
9:          $\mathcal{W}^j_{d+1} = Propagate(\mathcal{W}^j_d)$
10:         if $\mathbf{Cell}(\mathcal{R}_a) \in \mathcal{W}^j_{d+1}$ then $\mathbf{A}(\mathcal{R}_a) = \mathcal{F}_j$
11:         if $\exists \mathcal{R}_i \in \mathcal{R} | \mathbf{Cell}(\mathcal{R}_i) \in \mathcal{W}^j_{d+1}$
12:         then
13:             $\mathbf{Rank}(\mathcal{W}^j) = \mathbf{Rank}(\mathcal{W}^j) + 1$
14:         end if
15:         if $\mathcal{W}^j_{d+1} = \varnothing$ then $\mathcal{W} \setminus \mathcal{W}^j$
16:     end for
17: end while
```

Algorithm 4 Greedy and synchronized wavefront propagation

```
1:  Input: $\mathcal{F}$, $\mathcal{R}$, $\mathcal{CG}_{rid}$
2:  Output: Assignment of robot $\mathcal{R}_a$
3:  $\mathcal{W} = \mathcal{F}$
4:  while $\mathbf{A}(\mathcal{R}_a) = \varnothing$ & $\mathcal{W} \neq \varnothing$ do
5:      $\mathcal{W}_{ma} = \operatorname*{argmin}_{\mathcal{W}^k \in \mathcal{W}} \mathbf{Assigned}(\mathcal{W}^k)$
6:      $\mathcal{W}_{mamd} = \operatorname*{argmin}_{\mathcal{W}^k \in \mathcal{W}_{ma}} \mathbf{Pot}(\mathcal{W}^k)$
7:      for all $\mathcal{W}^j \in \mathcal{W}_{mrmd}$ do
8:          $d = \mathbf{Pot}(\mathcal{W}^j)$
9:          $\mathcal{W}^j_{d+1} = Propagate(\mathcal{W}^j_d)$
10:         if $\mathbf{Cell}(\mathcal{R}_a) \in \mathcal{W}^j_{d+1}$ then $\mathbf{A}(\mathcal{R}_a) = \mathcal{F}_j$
11:         if $\exists\ \mathcal{R}_i \in \mathcal{R} | \mathbf{Cell}(\mathcal{R}_i) \in \mathcal{W}^j_{d+1}$ & $\mathbf{A}(\mathcal{R}_i) = \varnothing$
12:         then
13:             $\mathbf{Assigned}(\mathcal{W}^j) = \mathbf{Assigned}(\mathcal{W}^j) + 1$
14:             $\mathbf{A}(\mathcal{R}_i) = \mathcal{F}_j$
15:         end if
16:         if $\mathcal{W}^j_{d+1} = \varnothing$ then $\mathcal{W} \setminus \mathcal{W}^j$
17:     end for
18: end while
```

results on a 200×200pixels regular grid (filled with 8 pixels square obstacles and 3 pixels wide corridors). Robots field of view of 10 pixels over 360°. Robots start from a corner of the environment. We can see that exploration using the *MinDist* assignment takes on average 20% more time than the other algorithms. On this environment, exploration performance between *MinPos* and the Greedy is not significative. *MinPos* and Greedy perform equaly than their not SyWaP counterpart thus validating our approach.

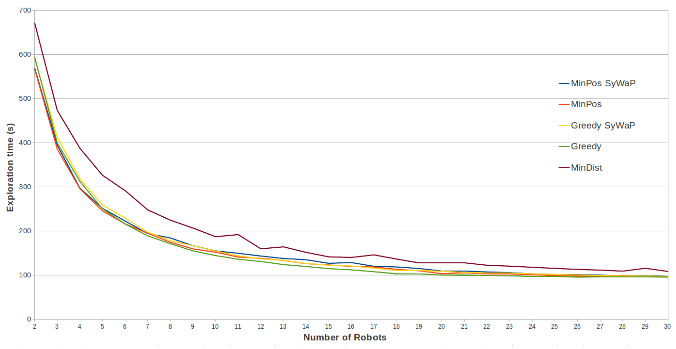


Fig. 6: Comparison of exploration performance between assignment algorithms with and without SyWaP.

However a clear difference appears in the computation times.

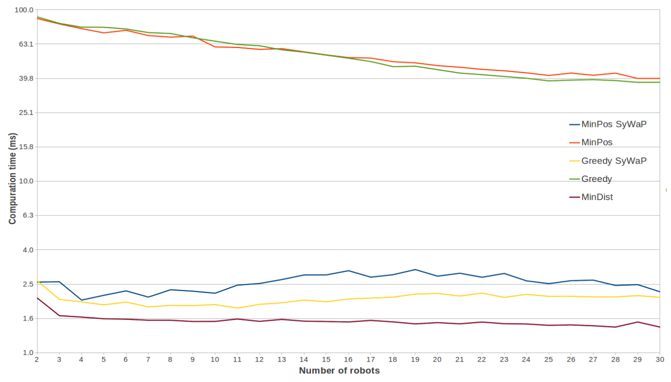


Fig. 7: Comparison of computation time between assignment algorithms with and without SyWaP.

Figure 7 shows the average computation times for each robot assignment during the same exploration. We can observe that, as expected, computation time of *MinDist* is the lowest. Close to it are the algorithms using *MinPos* and Greedy using SyWaP ; while their non-SyWaP counterpart takes a lot more computation time. Note the logarithmic scale of this chart. SyWaP algorithms reduces computation by a factor of 10.

We can also observe that the *MinPos* algorithm is slightly slower than the Greedy algorithm. We believe this due to the fact that with the *MinPos* SyWaP wavefronts are paused whenever they encounter a robot thus forcing the algorithm to propagate more waves. With Greedy, wavefronts are paused only paused when encountering an unassigned robot.

The framework and presented in section III *MinPos* algorithm have been validated with 5 MiniRex robots(illustrated figure 8 for the exploration of a unknown $120m^2$ environment during robotic challenge CAROTTE that our team (Cart-O-Matic 5lair.free.fr/Projects/Cartomatic/ won in 2012. The 2D map built with the trajectory of each robot is illustrated figure 9. The simulator used for the experiments presented here uses large part of the code used in the robots.

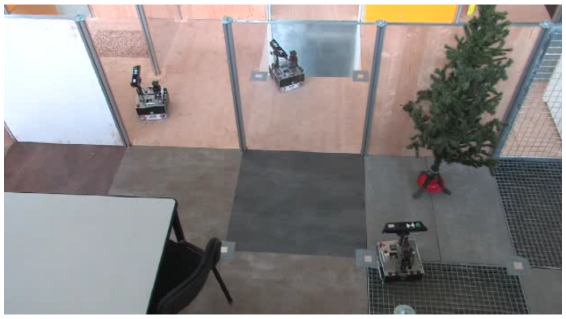


Fig. 8: 3 MiniRex robots exploring a $120m^2$ environment

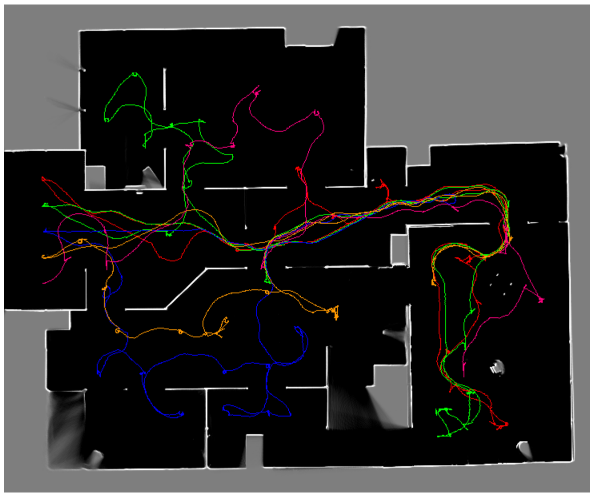


Fig. 9: Trajectories of each robot (initial position are on the left side)

Videos of the SyWaP approach for *MinPos* and Greedy assignments of targets during exploration simulations and videos of experiments with MiniRex robots using the *MinPos* algorithm can be seen at www.loria.fr/~bautin/videos.html.

VI. Conclusion

We addressed the problem of task assignment of spatially-situated tasks in homogeneous multi-robot teams. A novel approach combining cost computation with the assignment algorithm has been proposed. It is based on the synchronization of wavefronts propagations started from each target. The performances for exploration are similar. The improvement lies in the computation cost which is divided by 10. The approach has been validated within a framework developed for robotic exploration of unknown environment with multiple identical autonomous robots. To cooperate, the robots share their location and local maps. Each robot then decides autonomously which target it will explore next. As perspective, we plan to extend the SyWaP algorithm to hybrid metric/topological map representation to further reduce this computation cost.

References

[1] C. Stachniss, "Exploration and mapping with mobile robots," Ph.D. dissertation, University of Freiburg, Department of Computer Science, April 2006.

[2] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *The International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004. [Online]. Available: http://ijr.sagepub.com/content/23/9/939.abstract

[3] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., IEEE International Symposium on*. Washington, DC, USA: IEEE Computer Society, jul 1997, pp. 146 –151.

[4] D. Holz, N. Basilico, F. Amigoni, and S. Behnke, "Evaluating the efficiency of frontier-based exploration strategies," in *Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK)*, 2010, pp. 1–8.

[5] H. H. González-Banos and J.-C. Latombe, "Navigation strategies for exploring indoor environments," *The International Journal of Robotics Research*, vol. 21, no. 10-11, pp. 829–848, 2002.

[6] M. Gossage, A. P. New, and C. K. Cheng, "Frontier-graph exploration for multi-robot systems in an unknown indoor environment," *Distributed Autonomous Robotic Systems 7*, pp. 51–60, 2006.

[7] K. M. Wurm, C. Stachniss, and W. Burgard, "Coordinated multi-robot exploration using a segmentation of the environment," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE, 2008, pp. 1160–1165.

[8] J. Faigl, M. Kulich, and L. Preucil, "Goal assignment using distance cost in multi-robot exploration," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, 2012, pp. 3741–3746.

[9] J. Barraquand, B. Langlois, and J.-C. Latombe, "Numerical potential field techniques for robot path planning," in *Advanced Robotics, 1991. 'Robots in Unstructured Environments', 91 ICAR., Fifth International Conference on*, jun 1991, pp. 1012 –1017 vol.2.

[10] B. Yamauchi, "Frontier-based exploration using multiple robots," in *AGENTS '98: Proceedings of the second international conference on Autonomous agents*. New York, NY, USA: ACM, 1998, pp. 47–53.

[11] W. Burgard, M. Moors, C. Stachniss, and F. Schneider, "Coordinated multi-robot exploration," *Robotics, IEEE Transactions on*, vol. 21, no. 3, pp. 376 – 386, june 2005.

[12] R. Simmons, D. Apfelbaum, W. Burgard, M. Fox, D. an Moors, S. Thrun, and H. Younes, "Coordination for multi-robot exploration and mapping," in *Proceedings of the AAAI National Conference on Artificial Intelligence*. Austin, TX: AAAI, 2000.

[13] R. Zlot, A. Stentz, M. Dias, and S. Thayer, "Multi-robot exploration controlled by a market economy," in *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, vol. 3, 2002, pp. 3016 –3023.

[14] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[15] J. Ko, B. Stewart, D. Fox, K. Konolige, and B. Limketkai, "A practical, decision-theoretic approach to multi-robot mapping and exploration," in *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 4. IEEE, 2003, pp. 3232–3238.

[16] A. Bautin, O. Simonin, and F. Charpillet, "Minpos : A novel frontier allocation algorithm for multi-robot exploration," in *ICIRA'12 Proceedings of the 5th international conference on Intelligent Robotics and Applications*, ser. Lecture Notes in Computer Science, vol. 7507. Springer Berlin Heidelberg, Oct. 2012, pp. 496–508. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-33515-0_49

[17] P. Lucidarme and S. Lagrange, "Slam-o-matic : Slam algorithm based on global search of local minima. Patent num. FR1155625," Patent, June, 2011.

[18] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, Jun 1989.