Homecoming: A multi-robot exploration method for conjunct environments with a systematic return procedure

Shervin Ghasemlou[1], Ali Mohades[2], Taher Abbas Shangari[1], Mohammadreza Tavassoli[1]

[1]Amirkabir Robotics Center, Amirkabir University of Technology, Tehran, Iran
Shervin.ghasemloo@aut.ac.ir,taher.abbasi@aut.ac.ir,tavassoli.mreza@aut.ac.ir
[2]Faculty of mathematics and computer science, Amirkabir University of Technology, Tehran, Iran
mohades@aut.ac.ir

Keywords: multi-robot exploration, conjunct environment, task allocation

Abstract. The present work proposes a multi-robot exploration method for conjunct environments, based on one of the state-of-the-art algorithms. In many exploration missions, after the subject is found, it is beneficial if the discoverer robot returns back to the base station, in order to report, delivery or recharge. In addition, the exploration might need a long time to be finished or has to be done over and over. Returning back to the base station enables robots to get recharged, fixed, or even substituted with other robots. Furthermore, the equilibrium in task allocation to robots is this work's other concern. The presented algorithm also reduces the maximum energy consumption of robots, as a good side effect. The efficiency of the proposed algorithm is demonstrated by providing simulation results for a variety of obstacle densities and different number of robots.

1 Introduction

In the multi-robot exploration field of research, the most important objective is to provide an efficient method to explore unknown environments using a team of robots. The method should lead robots to visit every accessible region in the area, usually under connectivity constraints. The application fields of the algorithms vary from demining [1] to rescue [2] and mapping [3]. The communication type is usually dynamic and robots are equipped with Bluetooth [4], Wi-Fi [5] or ZigBee [6] technologies in order to communicate with each other.

There are two research fields in the literature which have very similar concepts to the multi-robot exploration study field: terrain coverage and foraging. For the terrain coverage studies, two main differences distinguish them from multi-robot exploration researches. Firstly, in terrain coverage studies it is not essential to have permanent com-

adfa, p. 1, 2011.

munication between robots. The main concern is to visit every corner of the environment at least once. Since usually there is no communication range limitations, the configuration space can be visited completely with-out any worry about breaking the connection. In terrain coverage problems, robots also don't have to report their situation to the base center or other robots during the process. Secondly, since the map of the environment is not available in multi-robot exploration problems, these algorithms should work online. On the other hand, in terrain coverage studies robots usually have access to the map of the environment, therefore every move can be computed before beginning of the process. In a survey paper from Choset [7], all researches in the field of terrain coverage are classified as heuristic algorithms, approximate algorithms, partial-approximate algorithms, and exact cellular decomposition ones. To show the in-tractability of this problem, in a paper from Zheng et al. [8], authors have shown those coverage problem versions which try to minimize the coverage time, are NP-hard problems. Authors also have provided a polynomial time algorithm based on another work [9] and claimed that the coverage time of their algorithm is close to the optimal solution. In several papers authors also consider sensor-based cover-age [10, 11, 12, 13], where a robot doesn't have to move into a cell to add that cell to set of visited ones, it is sufficient to observe that cell via sensors.

In the foraging problems, the number of employed robots are comparatively much more than both multi-robot exploration and terrain coverage fields of study. Another distinguishing characteristic of foraging researches is that the employed robots are often equipped with very limited tools. In foraging problems providing an effective task allocation scheme, usually based on swarm intelligence algorithms, is the main purpose. Similar to multi-robot exploration studies, robots communicate with each other, but in a different manner. In most of the studies communication is not dynamic and permanent. Based on the way ants communicate with each other by means of a chemical called "pheromone", robots communicate through virtual pheromone with each other [14]. In a paper by Couceiro et.al [15] the authors combine a biology inspired algorithm with a potential field method to explore the area. In other work [16], two distributed exploration algorithms are provided, gradient and sweeper algorithms. The first one is able to quickly return robots back to the base station, while the sweeper algorithm has the ability to find food in farther distances, albeit with a slower speed.

In spite of having different properties mentioned above, there is no exact boundary to separate these three categories from each other. For instance, in [17] the authors provide a swarm navigating method in which robots are in contact with each other by means of a wireless connection. The presented algorithm in this paper, as a multi-robot exploration algorithm, has common properties with both terrain coverage and foraging algorithms. In the next section, we investigate the multi-robot exploration literature more thoroughly.

2 Previous works

In an early work [18] a method was proposed in which a center makes decisions for all robots. In [19] authors presented a centralized method too, which considers the cost of

reaching a particular point along with the efficiency of the same point. This algorithm always tries to assign a point to a robot as its next position, if reaching to that point makes the best possible balance between cost and efficiency. In this study efficiency is based on the probability of visibility of the assigned target point to a robot, from the assigned target points to the other robots.

Although centralized methods are able to provide complete solutions, they act slowly. On the other hand decentralized algorithms are faster, but lack completeness and optimality. In [20] a decentralized method is proposed which con-siders range constraints. In this study, robots are able to decide whether to keep persistency of the communication with other robots or to avoid obstacles. The lack of prior knowledge about the environment makes it hard to provide a robust method, or to guarantee the performance. In [21] a decentralized multi-robot exploration method is provided which guarantees the performance of the algorithm under some assumptions. In another work [22] as a market-based one, a hierarchical task allocation method is provided that uses coalitions. The authors claim that such methods act better than greedy or coalition-free ones. In another work [23] two strategies for task allocation is provided. In the first strategy as a decentralized method, while the performance is good, the energy consumption is more than the second strategy, which is a centralized one. In [24] the authors have studied two different kinds of explorations where in both cases robots have to be in touch with each other. In one case, robots have to be connected with a fixed base station also, which delimitates exploration area, similar to the present work.

Performance measures vary widely. Different papers use total path lengths [25], balance in workload distribution [25], total steps of the algorithm [26], algorithm's overload [27, 28], energy consumption [28, 29] and time complexity [8, 23] to measure the efficiency of provided algorithms. In our work, distribution of exploration task, maximum energy consumption, traveled distance and number of step-moves for each robot are considered.

In the literature, there are several studies which have considered energy consumption [10, 27, 28, 30]. None of these works provide a scheme to recharge robots. If the exploration process is ongoing, as in a surveillance system [31], robots have to be charged or exchanged with other charged ones to continue the exploration task. In a paper from Koveos et al. [32], authors have studied multi-robot exploration in space missions. In their work robots are in touch with the base station and the base station is responsible for tracking and recharging them. In their study robots are getting recharged by radiations emitted from a laser. The distance of robots from the base station, environmental conditions and also the amount of energy needed for robots can affect the efficiency of recharge procedure in their work.

In a paper by Kovacs, Pásztor and Istenes [33] authors have proposed an algorithm to explore the environment under connectivity constraints. In this work robots have to be connected to each other and a fixed base station using a Blue-tooth communication system. The algorithm guarantees the communication persistency during exploration. The authors have shown that in obstacle-free environments, their algorithm works optimally, in the terms of the number of step-moves. The way we define conjunct environments in this paper, enables the presented algorithm to see conjunct environments

similar to obstacle free environments, from a global planner point of view. This property is the result of leaving the obstacle avoidance duty to the local planner. The optimality of the provided algorithm in [33], makes it suitable to be used as a basis for our global planner.

Three major contributions of this paper to the field are as follows. First, a systematic procedure for returning robots back to the base station have been provided. The return procedure executes simultaneous with the exploration task and doesn't interrupt it. The provided method also keeps the communication persistency as long as robots explore the environment. Second, the proposed method makes the system balanced in assigning the exploration task to robots and also reduces the maximum energy consumption. Simulation results show that the presented algorithm works close to the optimal solution for reducing maximum energy consumption and balancing the system. Third, the provided global planner works independent from local planners of each robot. Therefore the provided algorithm has the ability to be used in any conjunct environment with any type of robot.

3 Preliminaries

Similar to most studies, we use a grid to divide the environment into square cells. Two cells are considered neighbors if they share a common edge. In this part our aim is to define conjunct environments in a way that, regardless of where obstacles lie, each robot is able to move to any neighbor cell, or swap its position with neighbor robots, only through the common edge.

Conjunct environment: If in the environment E, the length of the edge of cells is A, and we have:

$$d_o + 2d_r + 4\mu \leq A \tag{1}$$

$$\forall \text{ Obstacle O: Distance } (O, O_c) \geq 2d_r + 4\mu \tag{2}$$

then we call E a conjunct environment.

In this definition d_o is the maximum diameter of excircles of obstacles, d_r is the diameter of robots, O_c is the closest obstacle to obstacle O and $\mu \geq 0$ is the prudential margin. The Distance function returns the closest distance between two obstacles by the Euclidean metric.

Fig.1.left illustrates the worst possible case for migration of a robot from one cell to a neighbor cell in a conjunct environment. Two robots R_1 and R_2 are in two neighbor cells and an obstacle lies in the middle of the common edge. Even in this case, there is enough room on both sides of the obstacle for each robot to pass through and go to the other cell. The grey tube indicates the obstacle-free area between the obstacle and the closest possible obstacle.

For flying robots when the altitude of flight is constant, it is usually easy to classify the workspace as conjunct. It is almost the same for marine robots too, because in both cases most of the times there is actually no obstacle in the work space. But the definition

of conjunct environment also covers obstructed areas. For instance, in a forest the maximum diameter of trees is available. The density of trees in the area is also available for

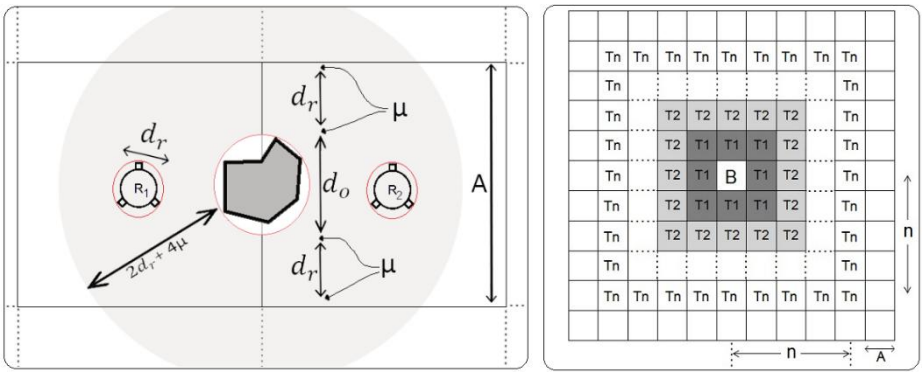

Fig. 1. Left: two neighbor cells of the environment and position of robots and obstacles in one possible worst case. Red circles are indicating excircles of robots and the obstacle. Right: the exploration Environment, the base station's cell (in the center) and Toruses.

many forests (see [34]). This makes it possible to estimate the closest possible distance between trees. Our definition of conjunct environment also covers some extra-terrestrial environments. There are papers which have studied the spatial distribution of rocks on mars [35] and size of rocks on it [36]. This information is adequate to decide whether the environment is conjunct or not.

Step-move: robots' migration from one cell to a neighbor one is called step-move.

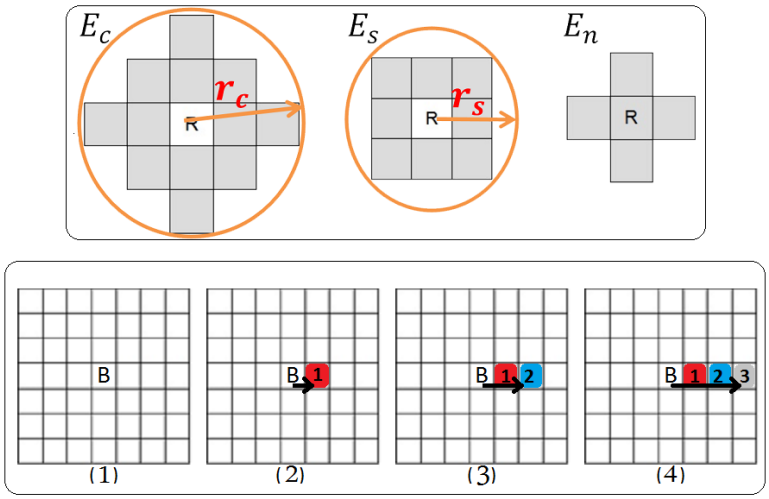

Fig. 2. Up: three subareas in the *kovacs algorithm*. All cells of each subarea is shown by grey. Note that in all of these three subareas the center cell is the location of the robot which the area is defined for. Only for E_n the occupied cell by the robot (center cell) is also a subset of the subarea. Down: result of running initialize position algorithm for three robots.

Team-move: migration of all robots together to their next assigned positions is called *team-move*. Next assigned position for a robot can be its current position, but in a *team-move* at least one robot goes to a non-visited cell.
In the *kovacs algorithm*, the length of a cell's edge is a function of the communication range. If we denote the radius of the communication range by r_c, the edge length of each square shaped cell is determined using this formula:

$$A = \frac{2r_c}{\sqrt{26}} \tag{3}$$

This equation is based on the definition of three subareas [33] (Fig. 2.Up):
$\boldsymbol{E_c}$**:** is a set of cells that neighbor robots of each robot are only allowed to be in them during the exploration process. In other words, neighbors are not allowed to leave the coverage range of the robot.
$\boldsymbol{E_s}$: is a set of cells that neighbor robots of each robot are only allowed to be in them after the completion of a *team-move*, and before executing the next team-move.
$\boldsymbol{E_n}$: is a set of cells that a robot can only go to them, during its current *team-move*.
***Torus*:** is a set of cells which their distance from the base cell (B) is equal. Here distance is measured using the infinite norm:

$$\text{Torus}_i = \{c \mid c \in E, ||c - B||_\infty = i\} \tag{4}$$

It can easily be shown that there are 8i cells in the Torus_i (see fig 1.Right). In the presented algorithm, since the first Torus is the closest Torus to the base station, robots return back to this torus in order to get recharged or deliver an object.

4 The base algorithm

In this section we rewrite the *kovacs algorithm* in our own words, and call it the *base algorithm.* The *base algorithm* is written in a way that imitates the Kovacs' algorithm's behavior in obstacle free environments, but in a simpler and clearer manner. For the sake of simplicity, we didn't put any code in the algorithm concerning connectivity and its constraints. All constraints are implicitly covered and satisfied in the *base algorithm.*

In the base algorithm, Lines 1-6 are initializations. Line 6 is a call to initialize position algorithm, which brings out all robots from the base station and builds up the exploration chain, as fig. 2.Down illustrates.

In the base algorithm (Table.1), if we set aside the number of step-moves caused by initialize position algorithm, the number of team-moves will be equal to the number of step-moves of the farthest robot at the end. Farthest robot has to do 8N step-moves, since the last torus has the same number of cells within. As an exception, the result of executing initialize position algorithm is considered as the first team-move.

In the rest of the *base algorithm*, the main loop lies between lines 8-25. The result of running the *base algorithm* for 3 robots is shown in the fig. 3. It can be seen that except for the first robot, all other robots never touch the first torus again during the process. In addition the distribution of exploration task in this algorithm is very disparate. The initialize position algorithm forces i-th robot to do i *step-move*s. Thereafter ro-

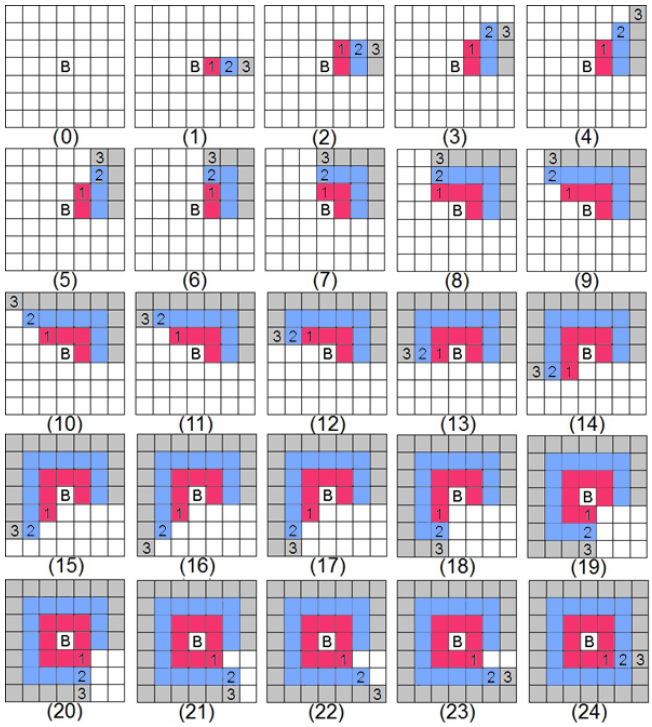

Fig. 3. The result of running the *base algorithm* for three robots. In the state zero, positions are being initialized (fig. 2.right), which results in the first *team-move*, state 1. Explored cells of each robot has shown in different colors, red for first robot, blue for second one and grey for third one.

Table 1. The *base algorithm*

Algorithm Base algorithm

```
1 N_r ← number of robots
2 Robot [] ← new Robot [N_r]
3 Torus [] ← new Torus [N_r]
4 Search_Direction ←choose from {cw, ccw} // cw=clockwise,ccw=counter clockwise
5 Initialize_Direction ← choose from {right, up, left, down}
6 Initialize_position (N_r , Initialize_Direction, Torus, Robot)
7 SFP ← 0 // SFP = swap feasibility pointer
8 for i=N_r down to 1
9     for j=1 to 8
10        if i is equal to 1 and j is equal to 8
11            break;//means one round of exploration is done
12        end if
13        for k=N_r down to SFP
14            Torus[k]. Robot. next_position_in_grid← next neighbor cell in the Torus (k)
15        end for
16        for k=SFP down to 1
17            Torus[k]. Robot. next_position_in_grid ← Torus[k]. Robot. position_in_grid
18        end for
19        move robots together to their next assigned positions
20        for k=1 to N_r
21            Robot[k]. position_in_grid ← Robot[k]. next_ position_in_grid
22        end for
23        SFP=N_r- | (N_r-i) %( 2N_r) - N_r|
24    end for
25 end for
```

bots visit remaining non-visited cells of their current torus in order to complete the exploration. Therefore first robot does 8 *step-move*s, second one 17 *step-move*s and the N-th one 9N-1 steps. It is obvious that farther robots carry the burden of the exploration task much more than the closer ones.

In order to have a simpler maintenance system, it will be much better if we use same hardware for all robots, particularly for locomotion parts and power sources. The main consumer of the power is the locomotion part. Other parts consume less in comparison.

In the *base algorithm*, if we choose the capacity of the power source according to requirements of farther robots, it results in forcing closer ones to carry a power source much heavier than their needs. On the other hand, if we choose the power source capacity according to the closer robots, the farther ones soon will get out of energy. In the next section we provide the idea to solve these problems.

5 The Homecoming algorithm

Assume that the *base algorithm* runs till the end of the initialize position algorithm. Then after every *team-move*, and before executing the next *team-move*s, the last robotR_N, is swapped c_i times with its predecessors, first time withR_{N-1}, then R_{N-2} and the same way till R_{N-c_i}. By doing so, if $c_i < N$, then R_N goes to previous position of R_{N-c_i} and robots $R_{N-c_1}, R_{N-c_1+1}, \dots, R_{N-1}$ go to the previous positions of $R_{N-c_1+1}, R_{N-c_1+2}, \dots, R_N$, with no change in the order. Positions of other robots don't change. If $\sum_1^8 c_i$=N-1 and above procedure is repeated eight times using $c_1, c_2, \dots, c_8$ for robot R_N, after these 8 team-moves R_N goes to R_1's previous cell, which lies at the first torus. All other robots are shifted one cell further.

Thereafter, if we repeat the whole procedure using the same c_i values for the current last robot, R_{N-1}, which its current position isR_N's previous position, all other robots are shifted one cell further, and R_{N-1} goes to the first cell. If this procedure is repeated totally N times, at the end 8N team-moves will be done, and all robots experience the first torus at least once. The flowchart of this process, which we call it *Homecoming* is shown in fig.4.left.

To initialize vector C we use a formula which distributes N-1 swaps almost uniformly between$c_1, c_2, \dots, c_8$. This formula plays the main role in distributing the exploration task between robots equally:

$$c_i = \begin{cases} 0 & i = 1 \\ K_i - K_{i-1} & i > 1 \end{cases} \quad \text{i=1, 2… 8 Where } K_i = \left\lfloor \frac{(i-1).(N-1)}{7} \right\rfloor \tag{5}$$

The provided scheme for return procedure is very effective. For example if during the exploration process a subject is found and the discoverer decides to send it back to the base station, there is always a further robot which is returning to the base, and the object can be given to this returning robot.

The connection between robots shouldn't be broken. When two robot have to swap, if cells which they occupy have a common edge, the connection is not broken, otherwise it will (fig. 4.Right, two upper cases). In case of a connection-breaking swap, it is post-

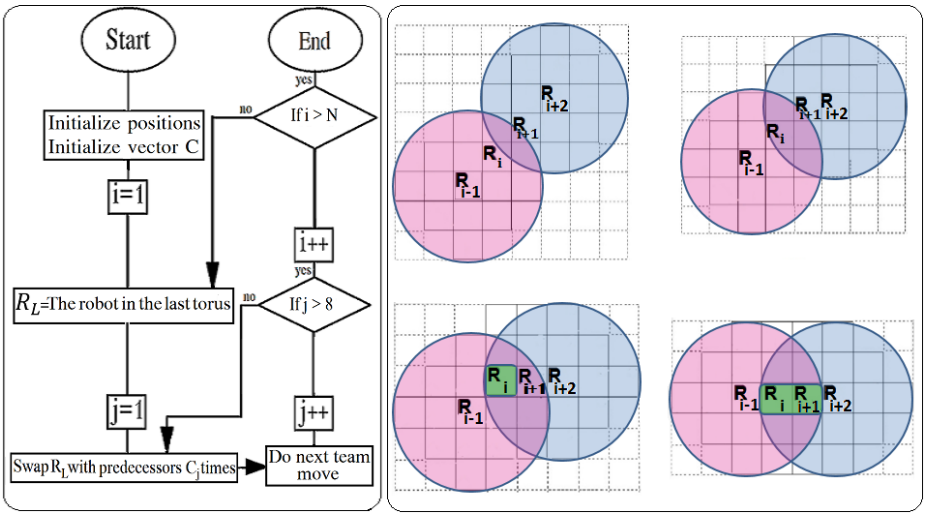

Fig. 4. Left: The flow chart of the Homecoming procedure. Right: Four possible situation of swap. In two upper situation, if R_i and R_{i+1} do swap, R_i gets out of the coverage area of $R_{i-1}$and R_{i+1}gets out of coverage area of R_{i+2}, which results in breaking the connection. But for two lower situations at least one cell is common between coverage areas of R_{i+2} andR_{i-1}, which is highlighted in green. Swap can be done by means of this common cell.

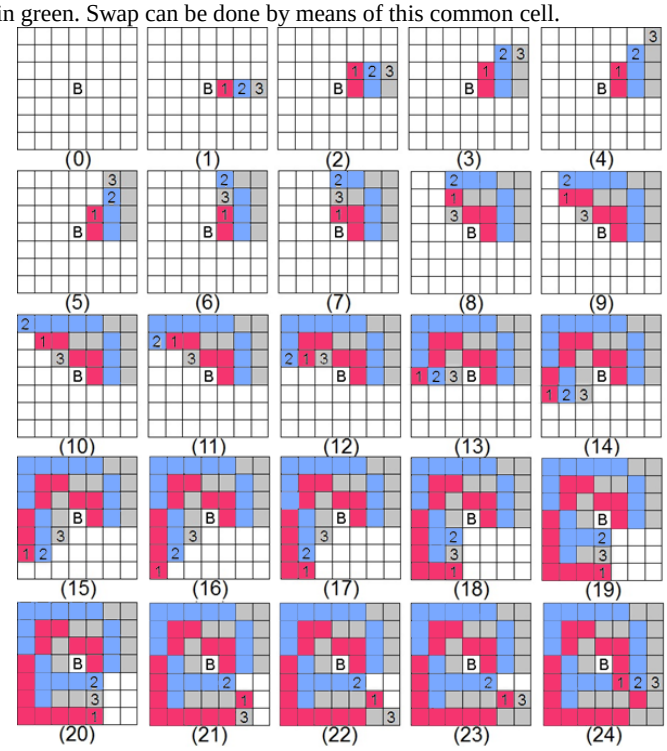

Fig. 5. The result of running the proposed algorithm for three robots. In each state the positions of robots after doing swaps are shown.

poned until the state of occupied cells change into a non-connection-breaking state (fig.4.Right, two lower cases). After every 2N *team-move*s, the chain of robots becomes completely straight and swaps can be done without breaking the connection.

6 Simulation results

In the first part of this section we provide the simulation results of the proposed algorithm without taking into account the effects of obstacle avoidance. In the second part, we provide a very simple obstacle avoidance method based on the Bug2 [38] algorithm in order to study the effects of the obstacle avoidance on the global planner's performance. All major parts of the simulator code, including the global planner (*base algorithm* and *Homecoming algorithm*) and some parts of the local planner module are implemented in the visual studio 2010 using C#.net. The rest of the local planner codes are implemented in Matlab.

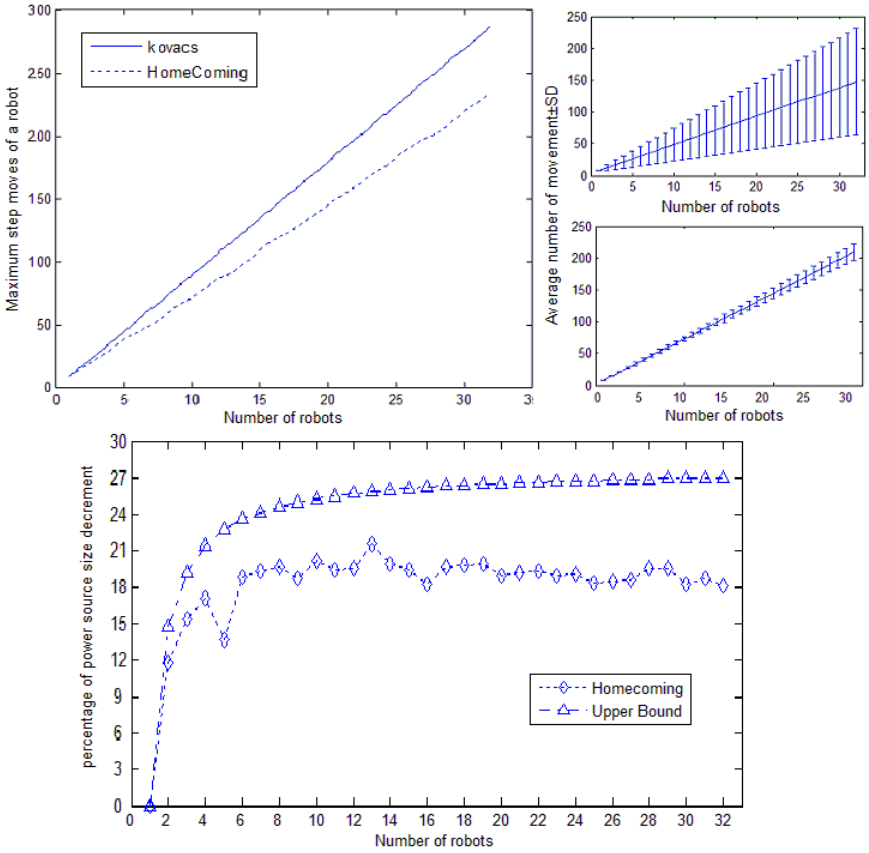

Fig. 6. Up-left: Maximum number of *step-move*s in kovacs and *Homecoming algorithm* s. Up-right: Average number of *step-move*s (mean) with standard deviation around it for *kovacs algorithm* (up) and *Homecoming algorithm* (down). Down: Percentage of reduction in maximum energy consumption for the *Homecoming algorithm*.

6.1 The global planner

In these experiments, both kovacs and the proposed algorithm was executed for 1 to 32 robots. To measure how much these algorithm are successful in uniform task allocation, the number of *step-move*s of each robot was counted and the standard deviation around the average was computed. To measure the maximum energy consumption of robots, maximum number of *step-move*s of all robots was calculated. The results are shown in Fig. 6.

6.2 The local planner

In this section, maps of three set of different conjunct environments was generated. We generated 100 maps with the obstacle density d = 2%, 100 maps with d = 5% and 100 maps with d = 8%. The spatial distribution of obstacles is uniform, but for the size of obstacles we used Gaussian distribution with mean=1 meter and standard-deviation=0.25 meter. Fig. 7 illustrates one sample map of each density.

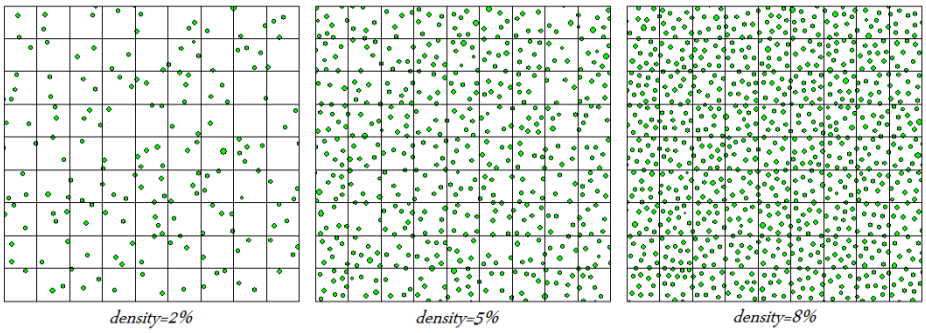

Fig. 7. Three sample maps with different obstacle densities

According to equation 3, the size of cell's edge is determined by the range of the communication device. Most of the communication devices provide a connectivity range of 10-100 meters, therefore the size of the edge can be 3.9 to 39 meters. We chose the size of the edge in our experiments A=10 meters, the diameter of robots=50 centimeters, and the prudential margin μ =10 centimeters. Our simulated robots are equipped with range finders, which are able to detect obstacles up to 7 meters in distance.

The swap procedure consists three steps (R_b=the returning robot, R_a=the other one):

- R_b determines the closest possible swap point on the path between centers of two neighbor cells. Then R_b moves to reach this point. R_b uses Bug2 algorithm to avoids obstacles along the path.
- Then R_a goes to reach the center of the neighbor cell. In this step R_a treats R_b like an obstacle on its path. R_a uses Bug2 algorithm to avoid obstacles too.
- Then R_b continues its way to reach the neighbor cell's center.

An example of swap procedure is shown in fig. 8. It is straightforward to determine the closest possible swap point. Firstly R_a calculates the position of the closest obstacle (CO) within the cell. Then the projection of the CO's center on the path is computed. The swap point is the closest point to R_a which its distance from the projected point is equal to $d_r/2+d_r/2+\mu$. Fig. 9 illustrates the simulation results for 3 to 10 robots.

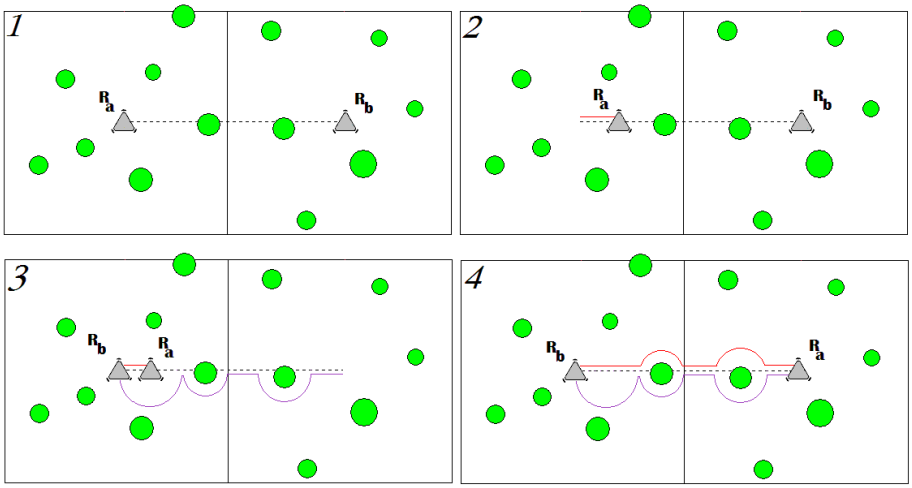

Fig. 8. State 1 represents positions of obstacles and robots (R_1in the left cell, R_2in the right cell) and the connecting line of centers. Other states represent three steps of the swap procedure respectively. The traveled paths of R_1 and R_2 are shown by red and purple trajectories. Robots always choose left turn in case of facing an obstacle, as a part of Bug2 algorithm. Robots keep their distance from obstacles equal to the prudential margin (μ).

7 Discussion

If we put aside the base station's cell, the reachable environment has $(2N+1)^2$-1 cells. Reachability here refers to ability to reach a point without breaking the connection. The initialize position algorithm does $\frac{N(N+1)}{2}$ *step-move*s, which results in visiting N unexplored cells. To visit the rest of the environment $(2N+1)^2$-N-1 other *step-move*s, totally $4.5N^2$+3.5N *step-move*s have to be done.

Here we show that for each algorithm which tries to return robots from the last torus to the first one during one complete round of exploration, at least $6.5N^2$+1.5N total *step-move*s are required. Such an algorithm has to do at least $4.5N^2$+3.5N *step-move*s to visit the whole area. Then for return procedure, if robots returns back to the first torus in a monotonic manner [37] N-1 swaps are required; otherwise robots should do more swaps. Each swap is equal to two *step-move*s, therefore totally 2(N-1) *step-move*s are needed. Thus return procedure for all robots takes 2N (N-1) moves. Therefore the optimal algorithm requires at least $6.5N^2$+1.5N *step-move*s, as a lower bound to visit the whole area. As described before, the provided algorithm acts in the same way.

The ideal equilibrium in task allocation is achieved when all robots travel same distance, equal to the mean. The mean for *kovacs algorithm* is 4.5N+3.5 and for provided algorithm is 6.5N+1.5. The increment in average number of *step-move*s is because of the Homecoming procedure. As fig.6.Up-right shows, the task allocation in provided algorithm is very close to the optimal and the standard deviation is very intensive around the mean. As a result of equalization in the task allocation, the maximum number of *step-move*s (NSM) is also less than the *kovacs algorithm* (fig. 6.Up-left).

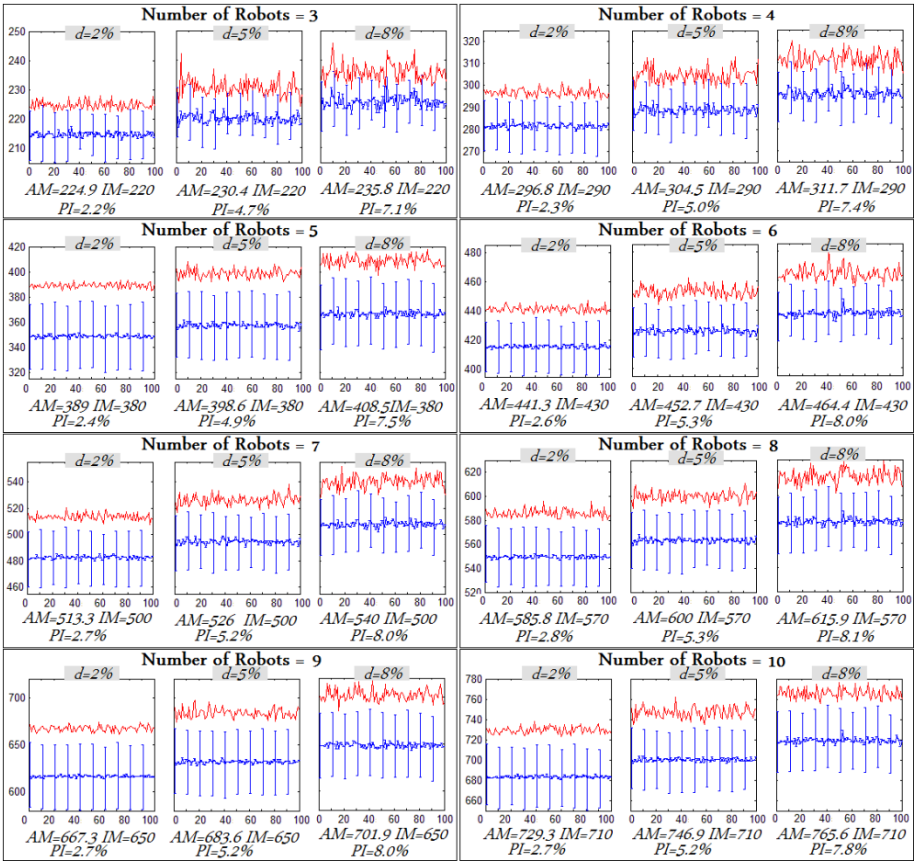

Fig. 9. Simulation results for *Homecoming algorithm* using the provided local planner. The Y axis represents the traveled distance and the X axis represents number of the map. Blue plots show average traveled distance and the standard deviation around the mean. Red plots show the maximum traveled distance in each map. For each density and for each number of robots, the Average of maximum traveled distance (AM), the ideal maximum traveled distance(IM) and the percentage of increment in the traveled distance (PI) comparing with the ideal traveled distance are computed and written under each plot. Ideal situation occurs in obstacle-free environments.

Since the robot with maximum NSM, consumes the energy more than the others, it is possible to set an upper bound on how much an algorithm is able to reduce the maximum energy consumption of robots, in comparison with the best non-returning algorithm (the *kovacs algorithm*). Obviously the maximum NSM can't be less than the average number of *step-moves* (6.5N+1.5). In the *kovacs algorithm* the maximum NSM is 9N-1, then the maximum percentage of reduction (MPR) will be:

$$\mathrm{MPR}\ (\mathrm{N}) = \frac{(9\mathrm{N}-1)-6.5\mathrm{N}+1.5}{9\mathrm{N}-1} = \frac{2.5\mathrm{N}-2.5}{9\mathrm{N}-1} * 100 \qquad (6)$$

By approaching the number of robots to infinity, $\mathrm{MPR}_{\max}$ is derived:

$$\mathrm{MPR}_{\max} = \lim_{i\to\infty} \frac{2.5i-2.5}{9i-1} * 100 = 27.78\% \qquad (7)$$

The size of the power source is directly related to the maximum energy consumption of robots. As fig. 6.Down shows, the maximum possible energy consumption needed to accomplish the exploration is decreased in the provided algorithm up to 21.5%, close to the maximum possible theoretical reduction.

The fig. 9 illustrates the effects of the local planner on the global planner's performance. For density = 2%, the maximum travelled distance by 3 to 10 robots are between 2.2 % to 2.8 % longer than the ideal situation. For densities 5% and 8% these numbers are 4.7% to 5.3 % and 7.1% to 8.1% respectively. With a slight difference, the same goes for the average travelled distance.

The provided global planner reduces the maximum energy consumption up to 21.5% in compare with the *kovacs algorithm*, and the local planner affects the global planner's performance up to 8.1 %. This means even when the density of the obstacles in the environment is 8%, the provided algorithm is able to reduce the maximum energy consumption up to 20% in compare with the *kovacs algorithm*.

8 Conclusion and future works

In this work, firstly an exact definition of conjunct environments was provided. Then based on a referenced work, a new algorithm having two main features was provided, return procedure and uniform task allocation. To evaluate the efficiency of the presented algorithm, several simulation experiments was done. The experiments show the efficiency of the provided algorithm. The provided algorithm acts very close to the optimal solution. As a good side effect, the maximum energy consumption of robots was decreased up to 21.5%. Then, a simple local planner was designed and the performance for different numbers of robots was evaluated. Results show that the obstacle avoidance task, doesn't affect the global planners overall performance in a drastic manner.

In this work we distributed N-1 swaps almost uniformly among the entries of C. A more intellectual method should take the initial position of robots into account to assign an independent vector C to each robot. This may result in a more uniform task allocation scheme. Designing enhanced local planners and also studying the applications of the presented algorithm on different types of robots, can be considered for future works.

9 References

1. Rogge, Jonathan, and Dirk Aeyels. "A novel strategy for exploration with multiple robots." In *4th International Conference on Informatics in Control, Automation and Robotics*, pp. 76-83. INSTICC Press, 2007.
2. Kleiner, Alexander, Johann Prediger, and Bernhard Nebel. "RFID technology-based exploration and SLAM for search and rescue." In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pp. 4054-4059. IEEE, 2006.
3. Ko, Jonathan, Benjamin Stewart, Dieter Fox, Kurt Konolige, and Benson 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, pp. 3232-3238. IEEE, 2003.
4. Bell, Jesse, Elizabeth Freeman, and Omar Costilla. "Cooperation of Autonomous NXT Robots Using Bluetooth Wireless Technology." (2013).

5. Howard, Andrew, Lynne E. Parker, and Gaurav S. Sukhatme. "Experiments with a large heterogeneous mobile robot team: Exploration, mapping, deployment and detection." *The International Journal of Robotics Research* 25, no. 5-6 (2006): 431-447.
6. Wang, Yiheng, Alei Liang, and Haibing Guan. "Frontier-based multi-robot map exploration using particle swarm optimization." In *Swarm Intelligence (SIS), 2011 IEEE Symposium on*, pp. 1-6. IEEE, 2011.
7. Choset, Howie. "Coverage for robotics–A survey of recent results." *Annals of mathematics and artificial intelligence* 31, no. 1-4 (2001): 113-126.
8. Zheng, Xiaoming, Sven Koenig, David Kempe, and Sonal Jain. "Multirobot forest coverage for weighted and unweighted terrain." *Robotics, IEEE Transactions on* 26, no. 6 (2010): 1018-1031.
9. Even, Guy, Naveen Garg, Jochen Könemann, R. Ravi, and Amitabh Sinha. "Min–max tree covers of graphs." *Operations Research Letters* 32, no. 4 (2004): 309-315.
10. Sipahioglu, Aydin, Gokhan Kirlik, Osman Parlaktuna, and Ahmet Yazici. "Energy constrained multi-robot sensor-based coverage path planning using capacitated arc routing approach." *Robotics and Autonomous Systems* 58, no. 5 (2010): 529-538.
11. Yazici, Ahmet, Gokhan Kirlik, Osman Parlaktuna, and Aydin Sipahioglu. "A dynamic path planning approach for multi-robot sensor-based coverage considering energy constraints." In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pp. 5930-5935. IEEE, 2009.
12. Mandal, Paramita, Ranjit Kumar Barai, Madhubanti Maitra, Subhasish Roy, and Somesubhra Ghosh. "Autonomous robot coverage in rescue operation." In *Computer Communication and Informatics (ICCCI), 2013 International Conference on*, pp. 1-5. IEEE, 2013.
13. Batsaikhan, Dugarjav, Adiyabaatar Janchiv, and Soon-Geul Lee. "Sensor-Based Incremental Boustrophedon Decomposition for Coverage Path Planning of a Mobile Robot." In *Intelligent Autonomous Systems 12*, pp. 621-628. Springer Berlin Heidelberg, 2013.
14. Senthilkumar, K. S., and K. K. Bharadwaj. "Multi-robot exploration and terrain coverage in an unknown environment." *Robotics and Autonomous Systems* 60, no. 1 (2012): 123-132.
15. Couceiro, Micael S., Rui P. Rocha, Carlos M. Figueiredo, J. Miguel A. Luz, and NM Fonseca Ferreira. "Multi-robot foraging based on Darwin's survival of the fittest." In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 801-806. IEEE, 2012.
16. Hoff, Nicholas, Robert Wood, and Radhika Nagpal. "Distributed Colony-Level Algorithm Switching for Robot Swarm Foraging." In *Distributed Autonomous Robotic Systems*, pp. 417-430. Springer Berlin Heidelberg, 2013.
17. Ducatelle, Frederick, Gianni A. Di Caro, Carlo Pinciroli, Francesco Mondada, and Luca Gambardella. "Communication assisted navigation in robotic swarms: self-organization and cooperation." In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pp. 4981-4988. IEEE, 2011.
18. Simmons, Reid, David Apfelbaum, Wolfram Burgard, Dieter Fox, Mark Moors, Sebastian Thrun, and Håkan Younes. "Coordination for multi-robot exploration and mapping." In *AAAI/IAAI*, pp. 852-858. 2000.
19. Burgard, Wolfram, Mark Moors, Cyrill Stachniss, and Frank E. Schneider. "Coordinated multi-robot exploration." *Robotics, IEEE Transactions on* 21, no. 3 (2005): 376-386.
20. Vazquez, Jose, and Chris Malcolm. "Distributed multirobot exploration maintaining a mobile network." In *Intelligent Systems, 2004. Proceedings. 2004 2nd International IEEE Conference*, vol. 3, pp. 113-118. IEEE, 2004.
21. Low, Kian Hsiang, Jie Chen, John M. Dolan, Steve Chien, and David R. Thompson. "Decentralized active robotic exploration and mapping for probabilistic field classification in

environmental sensing." In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pp. 105-112. International Foundation for Autonomous Agents and Multiagent Systems, 2012.
22. Hawley, John, and Zack Butler. "Hierarchical distributed task allocation for multi-robot exploration." In *Distributed Autonomous Robotic Systems*, pp. 445-458. Springer Berlin Heidelberg, 2013.
23. Wawerla, Jens, and Richard T. Vaughan. "A fast and frugal method for team-task allocation in a multi-robot transportation system." In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 1432-1437. IEEE, 2010.
24. Rooker, Martijn N., and Andreas Birk. "Multi-robot exploration under the constraints of wireless networking." *Control Engineering Practice* 15, no. 4 (2007): 435-445.
25. Fazli, Pooyan, Alireza Davoodi, and Alan K. Mackworth. "Multi-robot repeated area coverage." *Autonomous Robots* 34, no. 4 (2013): 251-276.
26. Tanoto, Andry, and Ulrich Rückert. "Local Navigation Strategies for Multi-Robot Exploration: From Simulation to Experimentation with Mini-Robots." *Procedia Engineering* 41 (2012): 1197-1203.
27. Mei, Yongguo, Yung-Hsiang Lu, Yu Charlie Hu, and CS George Lee. "Deployment of mobile robots with energy and timing constraints." *Robotics, IEEE Transactions on* 22, no. 3 (2006): 507-522.
28. Mei, Yongguo, Yung-Hsiang Lu, CS George Lee, and Y. Charlie Hu. "Energy-efficient mobile robot exploration." In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pp. 505-511. IEEE, 2006.
29. Stirling, Timothy, and Dario Floreano. "Energy-time efficiency in aerial swarm deployment." In *Distributed Autonomous Robotic Systems*, pp. 5-18. Springer Berlin Heidelberg, 2013.
30. Hazon, Noam, and Gal A. Kaminka. "On redundancy, efficiency, and robustness in coverage for multiple robots." *Robotics and Autonomous Systems* 56, no. 12 (2008): 1102-1114.
31. Jaimes, Aldo, Srinath Kota, and Jose Gomez. "An approach to surveillance an area using swarm of fixed wing and quad-rotor unmanned aerial vehicles UAV (s)." In *System of Systems Engineering, 2008. SoSE'08. IEEE International Conference on*, pp. 1-6. IEEE, 2008.
32. Koveos, Yannis, Athanasia Panousopoulou, Efthymios Kolyvas, Vasiliki Reppa, Konstantinos Koutroumpas, Athanasios Tsoukalas, and Anthony Tzes. "An integrated power aware system for robotic-based lunar exploration." In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pp. 827-832. IEEE, 2007.
33. Kovács, Tamás, Attila Pásztor, and Zoltán Istenes. "A multi-robot exploration algorithm based on a static Bluetooth communication chain." *Robotics and Autonomous Systems* 59, no. 7 (2011): 530-542.
34. Cox, Trevor F. "A method for mapping the dense and sparse regions of a forest stand." *Applied Statistics* (1979): 14-19.
35. Christensen, Philip R. "The spatial distribution of rocks on Mars." *Icarus* 68, no. 2 (1986): 217-238.
36. Golombek, M. P., A. F. C. Haldemann, N. K. Forsberg-Taylor, E. N. DiMaggio, R. D. Schroeder, B. M. Jakosky, M. T. Mellon, and J. R. Matijevic. "Rock size-frequency distributions on Mars and implications for Mars Exploration Rover landing safety and operations." *Journal of Geophysical Research: Planets (1991–2012)* 108, no. E12 (2003).
37. o'Rourke, Joseph. *Computational geometry in C*. Cambridge university press, 1998.
38. Choset, Howie M., ed. Principles of robot motion: theory, algorithms, and implementation. MIT press, 2005.