Adaptive Path Planning for Effective Information Collection

Ayan Dutta, Prithviraj Dasgupta

***Abstract*— We consider the problem of information collection from an environment by a multi-robot system, where the locations to sample information from are known with some amount of uncertainty by the robots. Different robots are equipped with different types of sensors and the quality of information collected can be improved if information is sampled from the environment with different sensors, wherever possible. The problem facing the robots is to determine their paths through information sampling locations, so that the overlap between paths of robots with identical sensors is reduced, while the robots with complementary sensors are sllowed to have overlapping paths. To address this problem, we describe a distributed path planning algorithm that comprises of two steps - path merging and path decoupling so that the robots are able to collect more information from the environment. We have shown analytically that our proposed algorithm has polynomial time complexity and our experimental results, in simulation, show that by dynamically adapting paths, the robots are able to collect higher information than by following initially generated individual paths.**

Introduction

Dynamic path planning is a well-known and crucial aspect of autonomous navigation of multiple mobile robots. Recently, researchers have considered a practical aspect of the multi-robot path planning problem where each robot has to dynamically adapt its path based upon the information collected while sampling certain strategic points along their path [9]. In these techniques, each robot has the same set of sensors and allocating multiple robots to collect information along the same path is considered sub-optimal. Also, the path update mechanism of a robot incorporates only the information collected by a robot itself while navigating. In this paper, we consider such as information collection problem with heterogeneous robots (sensor-wise). We posit that the performance of multi-robot information collection with multiple sensors can be improved, if a robot's path is not only updated using its self-sampled information, but, the information collected by other robots as well as the sensor capabilities of those robots is incorporated dynamically into robots' path plan. However, this also introduces additional overhead to avoid redundant information collection by robots and to direct robots with appropriate sensors to improve the quality of collected information, and, finally to determine opportune locations and time intervals to periodically share the information between robots. The problem is further complicated in realistic scenarios, where robots' have limited battery and might intermittently get out of communication range with each other.

Computer Science Department, University of Nebraska at Omaha
{adutta, pdasgupta}@unomaha.edu

To address these challenges, we propose a novel framework were robots initially generate paths individually assuming a probabilistic information distribution in the environment. These paths are then shared between robots and two techniques - merging paths between robots with different sensors or decoupling paths between robots with similar sensors, are employed, to increase the information gain from the collective paths. Our proposed algorithm has polynomial time complexity. We have tested the quality of our proposed path adaptation algorithms in simulation. Our results show that with dynamic adaptation of paths based on robots' sensor capabilities, multiple robots can collect information more effectively than without adaption of paths. The run time of our proposed path adaptation algorithms are also computationally feasible - maximum of 20 seconds for adapting a path, consisting 50 cells.

Related Work

Multi-robot path planning has been a well researched aspect of robotics and several techniques for waypoint navigation and coverage with multiple robots have been proposed. Mutli-robot informative path planning (MIPP) involves an aspect of the general multi-robot path planning problem where each robot has to determine waypoints between given start and end locations in the environment so that the information gain of the resulting path is increased. In one the earliest works on MIPP, Guestrin *et al.* [4] have modeled it as a sensor placement problem and proposed a greedy algorithm that ensures that the mutual information gain across the sensors is maximized. Singh *et al.* [9] have proposed a recursive, branch and bound algorithm to solve the MIPP problem that finds the best, budget-limited path through a graph of possible waypoints. Their algorithm is verified for a maritime information collection application and shows improvements over greedy path selection. This work is inspired from a recursive greedy best walk search algorithm, as described in [2]. The MIPP problem with periodic connectivity between robots has been studied in [5]. Here robots do not need to maintain continuous connectivity, but form a connected network at certain intervals. In contrast to these works, our work in this paper considers robots with different sensors and attempts to merge or separate paths to improve the information collection based upon the similarity of sensors of robots allocated to those paths.

Path merging algorithms for a single robot's path have been proposed in [6]. The robot first generates multiple paths between given start and goal locations using any sampling based path generation method. Then, a hybrid path that improves an objective function based on the quality

of the waypoints is generated by selecting subsets of waypoints from the different generated paths. In [7], authors have proposed a multi-robot path planning algorithm, where robots adjust their initially generated paths depending on their respective priorities, while in [1], a constrained path planning problem by multiple robots is proposed. Most of these algorithms are intended for waypoint navigation and are not directly applicable to information collection. In our proposed method, robots first generate their paths using a greedy method, similar to [4] and then the generated paths are dynamically adjusted to achieve higher information gain.

MODEL

Let $R = \{r_1, r_2, .., r_N\}$ denote a set of N robots and S_i denote the set of sensors on robot r_i. Each sensor $s \in S_i$ has an associated reliability value ρ_s that represents the quality of information that is sampled by that sensor from the environment. For the purpose of navigation, each robot uses a map of the environment; the map is decomposed into a grid-like cells using a cellular decomposition technique [3]. A robot enters a cell to collect information from the region within the cell. Let $\mathcal{C}$ denote the set of cells in the environment. Robot r_i's path, Π_i is defined as an ordered sequence of cells it visits, i.e., $\Pi_i = \{c_1, c_2, ...\}$.

For information collection, each cell $c_j \in \mathcal{C}$ is further sub-divided into a set of information points. Upon entering a cell, a robot dynamically selects a subset of the information points in the cell to visit and take readings from using its on-board sensors. Following Shannon's information entropy formula [8], the information gain from point $p_{k,j} \in P_j$ sampled by robot r_i in cell c_j is given by $I_{i,j} = -\sum_{p_{k,j} \in P_j} P(p_{k,j}) \log_b P(p_{k,j}) \times \sum_{s \in S_i} \rho_s$, where $P(p_k)$ denotes the probability that point p_k provides high quality information and ρ_s is the reliability value of sensor $s \in S_i$. We assume that the information collected from information points in a cell follow the law of diminishing returns. That is, the initially visited points in a cell provide new information, but as the robot collects more information from that cell, future points might add only repetitive or redundant information. To model diminishing returns from information collection, we limit the maximum allowable information gain from a cell to a threshold, MAX_INFO. Once the information gain from a cell exceeds MAX_INFO, the robot does not gain any new information but will only expend time and battery to visit more information points in that cell. The total information gain from robot r_i's path Π_i is given by $I(\Pi_i) = \sum_{c_j \in \Pi_i} I_{i,j}$.

Robots incur costs in terms of battery power spent for navigating in the environment and collecting information using their sensors. The cost of visiting cell $c_j \in \Pi_i$ by robot r_i can be written as, $cost_{i,j} = cost^{i,j}_{sense} + cost^{i,j}_{travel}$. The sensing cost is calculated from the sensor's power requirement while the travel cost is calculated as the distance between the centroids of cells $(c_{j-1}, c_j) \in \Pi_i$. The utility of path Π_i can then be written as $U(\Pi_i) = I(\Pi_i) - cost(\Pi_i)$, where $cost(\Pi_i) = \sum_{c_j \in \Pi_i} cost_{i,j}$.

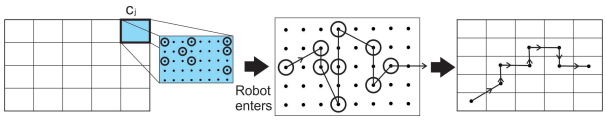


Fig. 1. (left)The environment is divided into cells using grid-based decomposition. Each cell c_j contains a set of information points that are distributed uniformly across the cell (black dots). A robot selects a subset of these information points to visit and sample data from (highlighted points). (middle) Robot's path through the selected information points in a cell; (right)One robot's path through the environment.

For combining the utilities of overlapping paths followed by two different robots, we consider their sensor types. If the sensors are identical, the combined utility is sub-additive as the robots each expend the cost to navigate the path but get the same information on their sensors. On the other hand, if the sensors are complementary, the utilities are super-additive as more information is gained by sampling the same points using different sensors. The utility function is given in Equation 1.

$$U(\Pi_{ij} = \begin{cases} U(\Pi_i) + U(\Pi_j) & - \sum_{c_k \in \Pi_i \cap \Pi_j} I_{i,k} \\ & \text{if } S_i \cap S_j \neq \{\emptyset\} \\ U(\Pi_i) + U(\Pi_j) & +(\sum_{c_k \in \Pi_i \cap \Pi_j} I_{i,k} \times \sum_{s \in S_i \cup S_j} \rho_s) \\ & \text{if } S_i \cap S_j = \{\emptyset\} \end{cases} \tag{1}$$

The objective of the informative path planning problem is to find a set of paths $\{\Pi_1, \Pi_2, .., \Pi_{|R|}\}$, s.t. $\max \sum_{\forall i \in R} U(\Pi_i)$.

PATH ADAPTATION ALGORITHMS FOR INFORMATION COLLECTION

Initially all the robots will generate paths for moving through the environment to collect information. Any path generation algorithm for information collection can be used. In this paper, we have used path generation algorithm for information collection, similar to as described in [4]. According to this path generation algorithm, we add 2 new cells to the current path in one cycle, addition of which to the current path yields highest amount of utility. Let C denote the set of 2 cells which will be added to robot r_i's current path Π_i. Let c_{last} be the last cell added to Π_i. We search for 2 such cells, $\{c_1, c_2\}$, which being added to Π_i, yields highest utility. c_1 is one-step distant cell from c_{last}, i.e., $c_1 \in neighbor(c_{last})$, c_2 is 2-step away from c_{last}. Once we are done adding best 2 such cells to C, we add C to Π_i. We repeat this procedure until current path's estimated battery expenditure ($EstB(\Pi_i)$) does not cross the battery budget.

Merging of robots' paths

We have discussed earlier that if robots have different sensors, then they would gain higher amount of information from visiting same cells. The objective of path merging algorithm is to merge paths of robots, having different

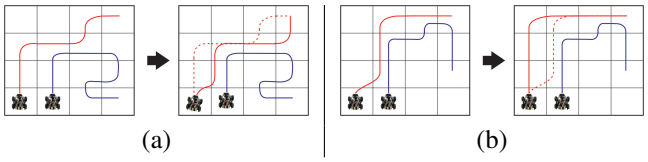


Fig. 2. (a)(left)Two robots generate two paths; (right)Left robot has changed its path to merge with the right one's path. (b)(left)Two robots generate two paths; (right)Left robot has changed its path to decouple from the other path (Dashed lines indicate the older paths)

sensors to increase the amount of information gain. For path merging, robots will exchange the information about generated paths with the robots which are in communication range. This merging of paths will be done sequentially by ordered robots. First every robot will broadcast information about their generated path along with the estimated amount of utility they will achieve by following its path. Upon receiving this path information from in-range robots, each robot will sort these paths, along with its own, according to the estimated utility of these paths. $\mathcal{K}_{sort}$ denotes the set of all paths in a sorted order. The sequence of path-merging will be according to the order of these sorted paths and it is denoted as $\mathcal{R}_{sort}$, where highest utility path generating robot will be the first member in $\mathcal{R}_{sort}$ and so on. Path merging is done by means of replacing cells in one robot's path with cells in another robot's path. One example of path merging is shown in Figure 2(a)(left), where two robots having different set of sensors merge their initially generated paths.

Path merging procedure is shown in Algorithm 1. Merging of paths will be done only between robots which do not have same set of sensors. Let's assume $r_i \in \mathcal{R}_{sort}$ is the robot which is merging its path, Π_i with paths of other robots, which do not possess the same set of sensors as r_i. For every cell $c_k \in \Pi_i$, r_i will check whether c_k is replaceable or not with any cell $c_l \in \Pi_j$ and $S_i \cap S_j! = \{\emptyset\}, \forall j \neq i$. Whether c_k and c_l are replaceable or not, can be decided by $replaceable()$ function. The function $replaceable()$ returns TRUE, when possible replacement of two candidate cells c_k and c_l earns higher utility than without the replacement (based on the utility function described earlier); otherwise $replaceable()$ returns FALSE (line $12-15$). First robot member, $r_1 \in \mathcal{R}_{sort}$ will merge its path first and when it is done merging, r_1 will send new path information Π_1, along with the information of about which cells are merged with other paths to all in-range robots and then $r_2 \in \mathcal{R}_{sort}$ will do merging. This procedure will go on until all robots are done merging their paths with all other robots. Any robot will not change the portion of its path, which is already been merged with, by another robot's path earlier.

Note that path merging does not necessarily lead to the fact that, $c_{i+1} \in neighbor(c_i)$, where, $c_i, c_{i+1} \in \Pi_{merged}$. Although one robot has to cover the cells between c_i and c_{i+1}, but it will not do sensing (or, information collection) through the intermediate cells. We assume that low-level path-planning algorithm such as D^* [10] algorithm to go from c_i to c_{i+1} as $c_{i+1} \notin neighbor(c_i)$ is already available

Algorithm 1: Path Merging of Robots

1 *pathMerge()*
Input: $\mathcal{K}_{sort}$: Sorted set of all paths.
Output: $\mathcal{K}^*$: A set of new merged paths.
2 $\mathcal{K}^* \leftarrow \{\emptyset\}$.
3 $\mathcal{R}_{sort}$: Sorted set of all robots, acc. to $\mathcal{K}_{sort}$.
4 Each $r_i \in \mathcal{R}_{sort}$ will do the following:
5 $BEST_U \leftarrow U(\Pi_i)$
6 **for** *all* $(\Pi_i, \Pi_j)pair, \exists \Pi_j \in \mathcal{K}_{sort}, i \neq j, S_i \cap S_j! = \{\emptyset\}$ **do**
7 **for** *each* $c_k \in \Pi_i$ **do**
8 **if** *($\exists c_l \in \Pi_j$, $c_l \notin \Pi_i$ and replaceable(c_k, c_l) == TRUE)* **then**
9 $\Pi_i \leftarrow \Pi_i \cup \{c_l\} \setminus \{c_k\}$. //replace c_k with c_l.
10 $\mathcal{K}^* \leftarrow \Pi_i$.
11 Send $\mathcal{K}_{sort}$ and $\mathcal{K}^*$ to $r_{i+1} \in \mathcal{R}_{sort}$.

replaceable(c_k, c_l)
Input: $c_k \in \Pi_i$ and $c_l \in \Pi_j$.
Output: TRUE or FALSE.
12 $\Pi_i' \leftarrow \Pi_i \cup \{c_l\} \setminus \{c_k\}$.
13 **if**
$(U(\Pi_i') + U(\Pi_j) + I_{i,l} \times \sum_{s \in S_i \cup S_j} \rho_s > BEST_U + U(\Pi_j))$
then
14 return TRUE;
15 **else**
16 return FALSE;

to the robots.

Path Decoupling

If two or more robots, having same set of sensors visit same cell to collect information, then they are not gaining any new information. Because same sensors will collect same type of information from same information points. So, they will just incur higher cost without gaining new information. Thus the total utility will be lowered. To avoid this situation, robots having same set of sensors should not visit same cells. To enable this, they should follow $pathDecoupling()$ method, shown in Algorithm 2. We will introduce 2 new terms in this subsection. First one is *similar path*. Path Π_i is labeled as *similar* to path Π_j, if corresponding robots, r_i and r_j have same set of sensors S and $|\Pi_i \cap \Pi_j| > p\% \times |\Pi_i|$, where p is a constant, which is the similarity threshold for determining whether the path is similar or not. If very few number of cells are same in two paths, then the robots do not need to change that, as this is a computationally costly operation. Second new term introduced in Algorithm 2 is *OK* cell. An *OK* cell is a cell, adding of which to a path Π_i, will not lead it to be *similar* to any other path, $\Pi_k, \forall k \neq i$. Paths are first sorted from highest cost to lowest cost order by every robot. The robot with highest cost path will first get chance to decouple its path from other paths, and then robots with lower cost paths will execute *pathDecoupling()* algorithm (lines $2-4$). Next, for every cell c_j, which is also present in other paths, the robot will find the best utility providing *OK* cell, $c_l (\neq c_j)$ from the neighbor cells of c_j's predecessor in the path and will replace c_j with c_l. An example of path decoupling operation is shown in Figure 2.(b), where robot

in the left changes its path to decouple from similar path of the other robot. Once the decoupling is done, robots will share their new path information with other in-range robots.

Algorithm 2: Decoupling of similar robot paths.

1 *pathDecoupling()*
  Input: K: Set of all paths.
  Output: Π_i: Modified path of robot r_i.
2 $K_i \subseteq K$: Sorted set of *similar* paths with robot r_i's path Π_i.
3 R_{sort}: Sorted set of robots according to K_i.
4 Each robot $r_k \in R_{sort}$ will do the following:
5 **for** *each path* $\Pi_k \in K_i$ **do**
6 **for** *each cell* $c_j \in (k \cap \Pi_i)$ **do**
7 $c_l \leftarrow$ Highest utility-earning *OK* cell in $neighbor(c_{j-1}) \setminus c_j$. (Details in text)
8 $\Pi_i \leftarrow \Pi_i \setminus c_j \cup c_l$.
9 Update K_i.
10 return Π_i.

Path planning algorithm: This is the main algorithm (shown in Algorithm 3), from which all other procedures are being called. Initially after generating the information points, robots generate their paths using *pathGeneration()* algorithm. Robots do not have unlimited communication range; they can communicate only within a fixed radius. It is computationally and battery power-wise very costly to keep all the robots in communication range at all time. But we assume that initially, before start of the information collection process, all the robots are in each others communication range, i.e., distance between any two robots is less than the communication range radius. After all the robots have generated their paths, they exchange their path information with all other robots and execute *pathDecoupling()* and *pathMerge()* algorithms (details in previous sections). Now that the robots have finally decided which paths they are taking, they start following their respective paths.

We assume that the initial knowledge about the information points is uncertain. The environment can change over time. While actually exploring, robots can find some new points to collect information from or the robots may perceive that some *a priori* selected points are currently unavailable. Again, robots can spend more/less battery than estimated amount. Thus the estimated utility and battery expenditure of path Π_i may change when the robots actually visit cells. This uncertainty is accommodated in the model by introducing noise to the number of selected points visited by the robots in each cell. If initially in cell c_j robot r_i was supposed to visit $\mathcal{K}$ number of cells, which was the estimator of the utility earned, then while actually visiting c_j, this $\mathcal{K}$ is adulterated to $\mathcal{K} \pm \delta$, where $\delta \in [0, \mathcal{K}]$. This change effects robot's information gain, incurred cost (thus utility) and actual amount of battery spent.

As actual and initial estimated utility and battery expenditure of a path might be different, so each robot needs to refine its earlier generated path periodically to accommodate the changes. At every time interval T^α, where T is a constant and $\alpha = \{0, 1, 2, ..\}$, if the difference between actual and initial estimated utilities and battery expenditure cross respective threshold values $THRES\_U$ and $THRES\_B$, the robots will generate new paths (lines $8-12$). As we know that robots have limited communication range - they necessarily do not need to stay within other robot's communication range and at every time interval T^α each robot generates a new path, therefore it would result in better utility achievement, if the robots can merge and decouple paths, when possible and appropriate after new path generations. That is why, whenever any robot r_i comes in communication range with $r_k (k \neq i)$, for the first time between two successive path generation intervals, i.e., between time intervals T^α and $T^{\alpha+1}$, they first exchange their paths and then either decouple or merge depending on the specific criteria discussed in previous sections (lines $13-19$). This algorithm runs until the robots have completed visiting their generated paths or they have spent their allocated battery budget (line 20).

Algorithm 3: Multi-Robot Informative Path Planning under Uncertainty

1 **Each robot** r_i **will follow these steps**:
2 Generate path Π_i, using *pathGeneration()* algorithm.
3 Exchange path information with in-range robots.
4 Decouple Π_i from similar paths, using *pathDecoupling()* algorithm.
5 Merge Π_i with appropriate robots' paths, using *pathMerge()* algorithm.
6 Send new path information to the next robot in $\mathcal{R}_{sort}$.
7 Start following Π_i.
8 Calculate actual utility earned and actual amount of battery spent.
9 **if** $(|U(\Pi_i) - U_{act}(\Pi_i)|) > THRES\_U \bigvee (|EstB(\Pi_i) - ActB(\Pi_i)|) > THRES\_B$ *at time interval* T^α **then**
10 Generate a new path Π_i' and follow the path.
11 **else**
12 Follow path Π_i.
13 **if** r_i *comes in comm. range of* r_k *for the first time in time interval* T^α *and* $T^{\alpha+1}$ **then**
14 **if** $S_i \cap S_k! = \{\emptyset\}$ **then**
15 Follow steps in line 4 and $6 - END$.
16 **else**
17 Follow steps in line $4, 5$ and $7 - END$.
18 **else**
19 Follow path Π_i.
20 **Termination Condition:** Generated path has been completely visited OR spent budgeted amount of battery.

ANALYSIS

Complexity of path merging algorithm: Let n and l denote the total number of robots and maximum number of cells in any robot's path. Line 3 in Algorithm 1 would take $O(nlogn)$ computations for sorting. For the loop in line 6, there can be $(n-1)$ of such pairs in worst case and for each of the $(n-1)$ pairs, inner loop in line 7 will run maximum of l^2 times. *replaceable()* function takes only constant time. Thus the final time complexity for path merging algorithm is $O(nl^2)$ (as $logn$ is much lesser than l^2), whereas time

complexity of comparable path hybridization algorithm [6] is $O(n^2l^2)$, which is clearly much worse than our proposed path merging algorithm.

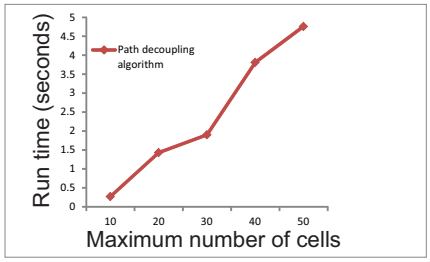
Complexity of path decoupling algorithm: Similar to *pathMerging()* algorithm, sorting will take $O(nlogn)$ computations, outer loop (line 5 in Algorithm 2) will run for maximum of $(n-1)$ times and the inner loop will run for l times, which is the maximum number of cells in any path. But computation inside the inner loop will not be of constant time. Complexity of finding best cell and checking for *OK* cell will be $O(b)$ and $O(n)$ respectively. Thus time complexity of path decoupling algorithm will be $O(n^2bl)$.

Any communication, with all other robots in worst case, will be of $O(n^2)$ complexity. Note that, in real world scenario, as the robots explore different regions, not all the robots will come into within every robot's communication range very often (except for the initial state). Thus actual communication complexity will be much less.

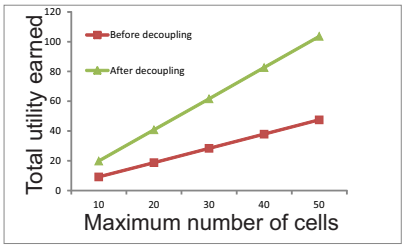
Experimental Evaluation

The algorithms presented in this paper are tested in simulated environment. An environment with 100 cells, in a grid structure has been used. Initial cell positions of robots are drawn from $\mathcal{U}[(0,3),(0,3)]$. We assumed that initially all the robots are with everyone's communication range. Robots first generate paths and then depending on sensor set on each robot, *pathDecoupling()* or *pathMerging()* algorithm is executed. Each robot is only allowed to visit upto a certain number of cells in the environment (which is the representative of battery budget). Maximum number of cells that each robot can collect information from is varied through 5 to 50. Maximum number of information points in each cell in drawn from $\mathcal{N}(15,3)$. Information collection is abstracted in the experiments and travel cost inside each cell is also not taken into account. But sensing cost is accounted for the cost calculations. We assume that each robot is equipped with one sensor. Reliability values of the sensors are drawn from $\mathcal{U}[0,1]$. Sensor costs, in terms of electrical energy needed, is also normalized to $[0,1]$. Each test is run for 5 times, but the deviation is nominal; thus not included in the figures. Note that, the run times reported in this paper are running time of the algorithm, not the time to visit cells and information collection by robots.

We have tested the *pathDecoupling()*, with 2 robots having same sensors. Run time of this decoupling algorithm is very nominal. For decoupling path with consisting 50 cells, algorithm took only 5 seconds. Our main objective was to show that by decoupling their paths, robots with same sensors earn more utility than if they followed the same path and the result is shown in Figure 3(b). As can be seen in this figure that with increasing length of the paths, with decoupling, robots earn more utility than following the same path and collecting redundant information. For example, with 50 cell-consisting path, in total, robots earn an utility of 103.53, after executing *pathDecoupling()* algorithm, as opposed to an utility of 47.43 without decoupling. Run time for this algorithm is also very nominal; for 2 robots,

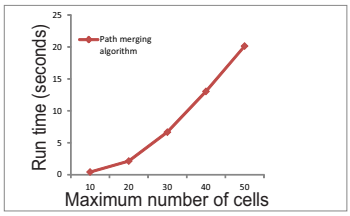


(a)

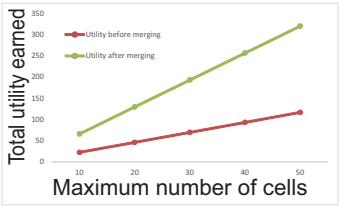


(a)

Fig. 3. (a) Run time of decoupling algorithm. (b) Utility earned by 2 robots - before and after path decoupling.



(a)



(b)

Fig. 4. (a) Run time of merging algorithm. (b) Utility earned by 2 robots - before and after path merging.

this algorithm only takes 4.76 seconds for decoupling 50 cell-consisting paths. A scenario of 5 cell-consisting path has been implemented with 2 robots, where they had same sensor and their initial paths were also exactly same. Actual resultant decoupled paths are shown in Figure 5(a)[right].

Next we have tested the *pathMerging()* algorithm, with 2 robots having different sensors. Run time for this algorithm is slightly higher than the previous algorithms; for 2 robots, this algorithm takes 0.41 and 20.13 seconds for decoupling 10 and 50 cell-consisting paths respectively (Figure 4(a)). With path merging, total utility earned increases consistently from the utility earned from paths before merging, as the path length increases (Figure 4(b)). For example, with path length 10, total utility before merging was 22.09 and it increased to 65.8 after merging and with path length 50, utility increased to 320.14 from 116.43. In this case also, we have implemented a scenario with 2 robots, where they have different sensors and the robots have to generate 5 cell-consisting paths. Actual resultant merged paths are shown in Figure 5(b)[right]. Notice that, actual path length of the resultant merged path is more than 5, but only 5 cells marked 'S' are visited by the robots for information collection, while other cells are passed by the robot to reach different

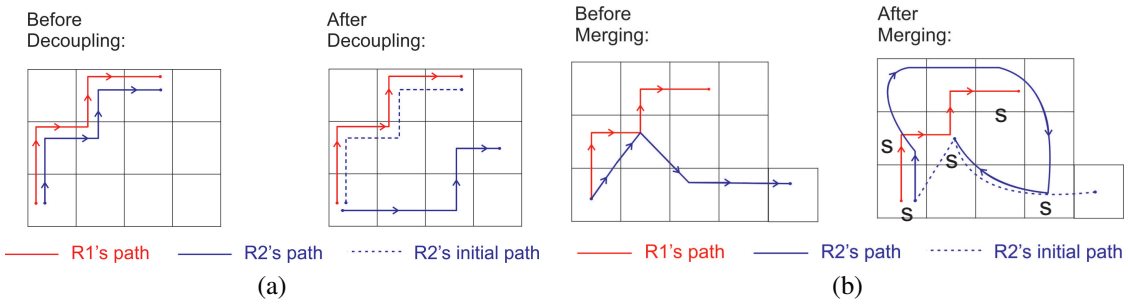


Fig. 5. (a) Decoupling of 2 robots' paths. (b) Merging of 2 robots' paths.

'S' marked cells. As our results indicate, by dynamically adapting paths, instead of following initial generated paths, robots can earn higher information and thus utility.

Conclusions and Future Work

In this paper, we have proposed path adaptation algorithms for information collection from an environment by a set of mobile robots. These path adaptation algorithms take sensor information into account and adjust the initially generated paths to collect higher amount of information. Our proposed algorithms are fast and they assure higher information collection than information collected by initially generated paths. In this paper, we have used already existing algorithm for informative path generation. We are working towards developing more sophisticated path generation algorithm and we will also test our proposed path adaptation algorithms with more number of robots and will compare the results with existing comparable algorithms.

References

[1] Pramod Abichandani, Hande Y Benson, and Moshe Kam. Decentralized multi-vehicle path coordination under communication constraints. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 2306–2313. IEEE, 2011.

[2] Chandra Chekuri and Martin Pal. A recursive greedy algorithm for walks in directed graphs. In *Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on*, pages 245–253. IEEE, 2005.

[3] Howie Choset, Kevin M. Lynch, Seth Hutchinson, George A. Kantor, Wolfram Burgard, Lydia E. Kavraki, and Sebastian Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, 2005.

[4] Carlos Guestrin, Andreas Krause, and Ajit Paul Singh. Near-optimal sensor placements in gaussian processes. In *Proceedings of the 22nd international conference on Machine learning*, pages 265–272. ACM, 2005.

[5] Geoffrey Hollinger and Sanjiv Singh. Multi-robot coordination with periodic connectivity. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 4457–4462. IEEE, 2010.

[6] Barak Raveh, Angela Enosh, and Dan Halperin. A little more, a lot better: Improving path quality by a path-merging algorithm. *Robotics, IEEE Transactions on*, 27(2):365–371, 2011.

[7] Ralf Regele and Paul Levi. Cooperative multi-robot path planning by heuristic priority adjustment. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 5954–5959. IEEE, 2006.

[8] Claude E Shannon. Prediction and entropy of printed english. *Bell system technical journal*, 30(1):50–64, 1951.

[9] Amarjeet Singh, Andreas Krause, Carlos Guestrin, and William J. Kaiser. Efficient informative sensing using multiple robots. *J. Artif. Intell. Res. (JAIR)*, 34:707–755, 2009.

[10] Anthony Stentz. Optimal and efficient path planning for unknown and dynamic environments. Technical report, DTIC Document, 1993.