Local Voronoi Decomposition for Multi-Agent Task Allocation

James Guo Ming Fu, Tirthankar Bandyopadhyay and Marcelo H. Ang Jr

***Abstract*— We propose a Local Voronoi Decomposition (LVD) Algorithm which is able to perform a robust and online task allocation for multiple agents based purely on local information. Because only local information is required in determining each agent's Voronoi region, each agent can then make its decision in a distributive fashion based on its allocated Voronoi region. These Voronoi regions eliminates the occurrence of agents executing instantaneous overlapping tasks. As our method does not require a pre-processing of the map, it is also able to work well in a dynamically changing map with changing number of agents. We will show our proof of concept in the problem of exploration in an unknown environment. In our experimental evaluation, we show that our method significantly outperforms the competing algorithms: Ants Algorithm and the Brick&Mortar Algorithm. Our results also show that our method is near the theoretical best solution.**

I. INTRODUCTION

Exploration is the task of traversing an initially unknown environment in the purpose of gaining new information. Applications to this task ranges from map building [1], search and rescue operations [2][3][4], lawn mowing, cleaning [5], mine clearing [6], intrusion detection, to fault identification. In some of these situations, it is essential to complete the task in a minimal amount of time. And in some practical cases, there will not be any knowledge of the environment. Hence, strategies which rely on a pre-processing of the map will no longer be practical in such cases.

There are currently a number of strategies to deal with the exploration problem. The use of multiple agents provide for greater robustness, reliability, scalability and economy. The greatest advantage of having multiple agents in performing a task is that each individual agent does not have to be as sophisticated as compared to the case of performing a task with just a single agent entity. A single agent has to be self-contained and self-reliant. Having multiple agents give the added flexibility where the task can still be performed even if some of the agents were to malfunction. When multiple agents are used to perform large scale tasks, the biggest challenge is for each agent to be able to decide which smaller tasks to do so as to achieve an overall efficient global behavior.

There are many issues which may arise and affect the overall performance of any multi-agent exploration strategy, e.g. agents heading towards the same unexplored region, agents clearing in a non-systematic manner (or even randomly) and the possibility of agents crossing each other paths which may result in collision or blockage.

The authors are with the Department of Mechanical Engineering, National University of Singapore, Singapore 117576 james.fu@nus.edu.sg tirthankar@nus.edu.sg mpeangh@nus.edu.sg

Our proposed strategy, *Local Voronoi Decomposition* (LVD) Algorithm, makes use of decomposing the agent's viewable space into Voronoi regions[14]. By making use of Voronoi regions in dynamic task allocation, we are able to overcome the above stated issues. Each agent is able to determine its own Voronoi region based on its local information along with other agents which are within its line of sight. This results in efficient region allocation for the exploration task and eliminates the possibility of cross paths since each agent will not enter into another agent's Voronoi region. Simple agent avoidance results as a by-product of this Voronoi decomposition.

In this paper, we compare our results with two other distributed strategies: the Ants algorithm [7][8][9] and the Brick&Mortar algorithm [10]. We highlight the advantages and limitations of both algorithms and show how our proposed method outperforms them. All three methods are also benchmarked against a theoretical best for a better feel of the results.

II. RELATED WORK

In one of the earlier works, Yamauchi [15] proposed the concept of frontier cells that separated the explored and the unexplored regions. Multiple agents communicate and share a common global map of the explored vs the unexplored regions while moving towards the nearest frontier. Although local and distributed, the agents had a tendency to move towards a common frontier. There was no explicit cooperation among the agents. Simmons [16] tried to address this problem by letting the agents 'bid' for frontiers to move based on their exploration cost and expected information gain. This leads to the agents being well separated and reducing the time to explore the region. Market based approaches for task allocation for exploration was proposed by Zlot [17] and Dias [18].

Simultaneous coverage of the complete environment using mobile sensor nodes was done in Cortes [19] by decomposing the environment into Voronoi regions. These sensor nodes are deployed such that the combined sensor network coverage overlaps the entire environment. Our approach is similar to this work as we try to compute local Voronoi regions for each agent based on its local sensing. In terms of coverage, however, we are instead interested in traversing the whole environment at least once by a group of multiple agents, i.e. in autonomous cleaning applications.

As we are interested in the exploration of an unknown environment, it is essential that the chosen algorithms for comparison are online in nature and do not require a pre-processing of the map. For comparison, we analyze two lead-

ing online approaches: the Ants algorithm, as well as a more recently developed algorithm, Brick&Mortar. The advantages and disadvantages of both algorithms are highlighted. Both algorithms have proven to be robust in the sense they are able to function even if some of the agents were to 'die' during the exploration process. The Brick&Mortar algorithm, in particular, is indeed a very systematic way of exploring an unknown map. It has been shown in [10], that Brick&Mortar fares much better than other leading exploration strategies, namely the *Multi-Depth First Search* [10][11][12] which has been modified from the *Depth First Search* algorithm [13] used for a single agent exploration.

III. Problem Formulation

In this section, we describe in detail the nature of the problem and any assumptions made. The task being considered is the exploration of an unknown map by a group of autonomous agents. Each agent is assumed to have perfect holonomic control. Each agent is able to accurately localize itself with respect to its starting point. Each agent is equipped with sensors that is able to distinguish between obstacles and other agents. Each agent is also able to determine the distance between itself and the obstacles as well as other agents.

The overall map will be divided into square cells. Each cell will hold either of the following three states:

- **Wall**: This cell is an obstacle. It could either be found within or on the perimeter of the map. A walled cell will not be traversable by any agent.
- **Unexplored**: This is a cell in free space which has not been visited by any agent.
- **Explored**: This is a cell in free space which has been traversed at least once by any of the agents.

The goal of the problem is thus to have all the agents (as a whole) traverse the given map at least once. Both the Ant and Brick&Mortar approaches leave markers in the environment. The assumption is that a marker is left in a cell when an agent traverses it as an indication of it being explored. Leaving markers in the environment may not be practical as markers may be physically displaced by external factors during exploration and the required number of markers may not be enough for very large maps. Alternatively, this can be done virtually via a central memory rather than physically leaving markers in the environment. One additional assumption is that each agent would have perfect communication with this base station which houses the central memory.

The greatest advantage of keeping track of explored and unexplored cells virtually via a central memory is that agents can be added or removed during the exploration process with no change to the existing algorithm. Decision making is still done in a distributive fashion. The map can also be dynamically changing and exploration can still be completed using the same algorithm.

The agents begin from one corner of the map. This mimics a practical situation where agents enter an unknown area from a single entry point. Each agent is able to move one in four directions - North, East, South or West. As they traverse the map, they will leave markers in unexplored cells. Each agent will thus be able to identify if any particular cell within their visual range has been explored or not.

IV. Existing Algorithms

Two algorithms have been chosen for comparison with our proposed *Local Voronoi Decomposition* (LVD) algorithm which share the same assumptions. Exploration is done on an unknown map and thus there is no need for any pre-processing of the map. Decision making is based on their local information. The algorithms are also robust enough that the agent number can be varied during the actual exploration. The two existing algorithms which will be highlighted are Ants and Brick&Mortar.

A. Ants

The Ants algorithm [7][8][9] is a distributed one which stems from the mimicking of how ants move around in nature. Ants leave pheromone traces as they move around as an indication of how often a particular cell has been traversed. As more ants move over a cell, the pheromone trace of the cell accumulates. All cells are initialized to 0 at the start indicating that all cells are unexplored. At each step, an agent will move to the neighboring cell which has the lowest pheromone level. If there are more than one cell which have the lowest pheromone levels, then one of hose cells is picked at random. Before the agent moves, it simply updates its current cell's pheromone level, e.g. incrementing the pheromone level by one. Svennebring and Koenig [8] show that the agents will eventually cover the entire map as long as the free space is continuous.

The advantage of this algorithm is primarily in its simplicity. The computational cost of determining its next step is very low and thus the computational complexity of each agent can be minimal. As the agents make their decisions based on the pheromone levels in the environment, this strategy is unaffected by a changing number of agents during the exploration process and likewise for a dynamically changing map (where there are mobile obstacles).

The main limitation of this is its sub-optimal decision-making process due to the agents' partially randomized movements. This algorithm is inefficient in situations where there are many rooms or where bottle necks are present. In such situations, agents will take a while to reach the other unexplored parts of the map.

B. Brick&Mortar

The Brick&Mortar algorithm [10] is able to explore an unknown map in a very systematic manner. The main advantage of this strategy is its ability for each agent to know when the map has been fully explored and thus stop. Although this is very good to have, our main focus is on analyzing the efficiency of an algorithm in which a given map is first fully explored.

The Brick&Mortar is novel in the sense it works by 'growing' the obstacles of the map as each agent traverses it. During the exploration phase, a cell can either be marked as *visited*, *explored* or *unexplored*. All cells are marked as

unexplored from the start. When a cell is labeled as *visited*, an agent will treat it as an obstacle in the future so that cell will not be moved to again. A cell is labeled as *explored* if the cell blocks the path between any two neighboring *explored* or *unexplored* cells. An *explored* cell indicates that at least one agent has traversed it before but it still might be used by other agents to access other *unexplored* cells. The map will eventually be covered with visited cells and a path of explored cells which link unexplored areas and other key areas. The paths of explored cells act like edges on a graph and these paths may not be the shortest path when agents are required to move across the graph. This exploration algorithm will stop when each agent is surrounded by *visited* cells. This only happens when the whole map is filled with *visited* cells.

Brick&Mortar works well on maps with little or no holes. More loops are formed when there are more holes in the map. Although Brick&Mortar has a method to solve for loop closure, it is observed it is more difficult for the loops to be closed, when the number of agents increases, resulting in more agents repeatedly traversing the loops.

Another limitation is that additional agents cannot be randomly added in during the exploration process. If an agent is added into a cell which is surrounded by only neighboring *visited* cells, it will conclude that the exploration task is complete and remain idle.

V. Our Local Voronoi Decomposition (LVD) Algorithm

Our proposed strategy is basically that of a Divide and Conquer one where our main emphasis is on the Divide part of the strategy. Our simulations show that a simplistic Conquer strategy will yield remarkably good results. Our main purpose is to show that it is possible to achieve a rather good overall exploration strategy by using this Divide strategy together with a rather simplistic Conquer strategy.

Each agent's Voronoi region can be easily calculated on an online basis given the positions of other agents as well as obstacles within its its field of view. On a macro view of the whole environment, the entire free space is sub-divided into each agent's Voronoi region. This is the Divide part of the strategy. The Voronoi regions allow each agent to plan its next move based only on its local Voronoi region. Each agent likewise does not need to bother about the other agent's Voronoi regions.

A. Local Voronoi Decomposition (LVD)

The LVD algorithm is broken down into two steps in Algorithm 1.

For the *get region step* (the Divide part of the strategy), our algorithm makes use of Voronoi regions based on line-of-sight as a means for each agent to identify which area it has to be concerned of. For all points within view of a particular agent, a point will be considered to be part of its Voronoi region if and only if there are no other agent which is nearer to that point, as seen in Fig. 2. Once an agent's region has been identified, it will then determine where to move next based on the nature of its own region. It will not consider areas outside its Voronoi region as those areas are either not within view or that it belongs to another agent.

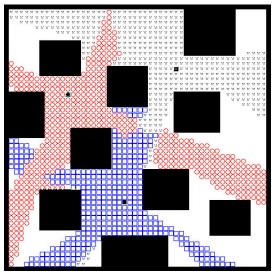


Fig. 1. The Voronoi regions of 3 agents during exploration. The agents are indicated by the darkened single cells. For a given agent, if a point on the map is within view and the closest as compared to other agents within view, that point will be part of its Voronoi region.

For the *navigation step* (the *Conquer* part of the strategy), our algorithm employs a simplistic greedy method in obtaining the next location to move to. Once an agent's Voronoi region has been identified, it simply picks the nearest unexplored cell as its target cell to move to. If there are more than one such candidate cells, it basically chooses from a list of preference, e.g. North, East, South, West.

As an agent visits a cell, it simply marks it as *explored*. Thus any cell will have either of two states - *explored* or *unexplored*.

Details of the pseudocode are provided in Algorithm 1.

Algorithm 1 Local Voronoi Decomposition

```
1:  Get Region Step %%Divide part of the strategy
2:  for all cells within view do
3:      if the cell is nearer to itself as compared to all other
        agents then
4:          mark the cell as within region
5:      end if
6:  end for
7:  Navigation Step %%Conquer part of the strategy
8:  if there is just one nearest unexplored cell that is within
    region then
9:      move to it
10: else if there are more than one nearest unexplored cell
    then
11:     move in an ordered list, e.g. [North, East, South, West]
12: else
13:     Search Mode
14: end if
15: go to 1
```

B. The Search Mode

The simple version of LVD works well for open maps and for smaller number of agents. The main problem which arises is when an agent's current Voronoi region is completely *explored*. It is possible that the map has not yet been fully explored. A quick option to this is for the agent to simply remain idle when this scenario arises. Remaining idle, in

most cases, is not a very efficient behavior of an agent, especially if its main task is to search and rescue. In cleaning applications, agents should continually move towards cells which are more dirty (or cells that have not been recently visited).

To improve on the *navigation step*, agents should move such that there is a higher probability where their changing Voronoi regions include unexplored areas. Hence, we have an additional assumption: each agent is able to identify key points in a map, namely the edges formed by the corners of the map or obstacles. These edges are similar to occlusion points [20]. These edges should make a difference in visibility when the agent swings around them and may unveil unexplored regions to the agents. These edges can be thought of as the camera positions when solving an *Art Gallery Problem*. These edges can also be thought of as skeletal nodes which run through the map, or forming a visibility graph using these edges as the nodes in the graph.

When in search mode, the main goal is for the agent to move towards unexplored regions of the map. The agent will do this by moving towards the nearest edge within its field of view. If there are more than one such point within its field of view, it will always choose the edges in which the same agent has not last visited the longest.

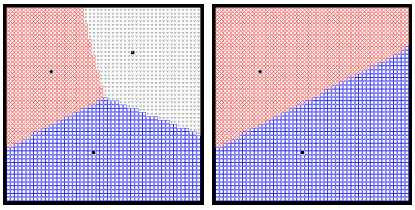
If an agent's Voronoi region is such that there are no edges or if the only edge in its region has just been visited, the agent's next strategy would be to move in a way so as to expand its own Voronoi region. The agent will do this by moving towards the nearest Voronoi boundary which is not made with obstacles or map perimeter. Details of this is given below, which is effectively the *search mode* of the previous algorithm.

Algorithm 2 Search Mode

1: Consider all edges within view. An edge is any corner formed by obstacles or the perimeter of the map, such that visibility changes when the agent swings around this corner.
2: **if** at any time an unexplored cell appears *within region* **then**
3: exit *Search Mode* and go back to Algorithm 1
4: **end if**
5: **if** there is at least one edge which has not been visited during the *Search Mode* **then**
6: move to the nearest edge which has not been visited
7: **else if** all edges have already been visited **then**
8: move to the least recently visited edge
9: **else if** there are no edges or if the only edge has just been visited **then**
10: move to the nearest Voronoi boundary which does not coincide with any wall or obstacle
11: **end if**

C. Robustness

As our LVD is an online method, agents are robust to robot failure. As intrinsic knowledge of the environment is not necessary, agents are still able to continue with their tasks even in a dynamically changing environment or when the number of agents are dynamic as well. Agents will still be able to identify its own local Voronoi regions even when agents are removed or added (as shown in Fig. 2 during the exploration process.



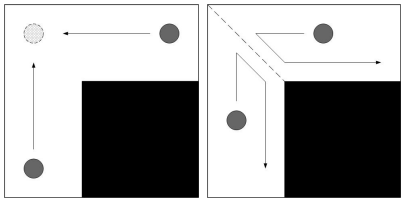
(a) Voronoi regions of 3 agents (b) Voronoi regions of 2 agents

Fig. 2. Voronoi regions dynamically adapt when agents are removed or added.

D. Emergent Cooperative Behavior

An interesting emergent cooperative behavior occurs when agents utilize LVD in exploration. LVD provides instantaneous separation of task space of each agent, but an overall systematic exploration of the free space is performed as seen in Fig. 4(a).

The LVD algorithm provides just-in-time information for each agent in its task allocation. As each agent makes its decision based on its current local voronoi region, there will be occasions where the agents' Voronoi regions may instantaneously overlap if the agents do not have all other agents within its field-of-view. This may result in agents having the same destination. However, this problem is automatically resolved when they come within view of one another, as seen in Fig. 3.

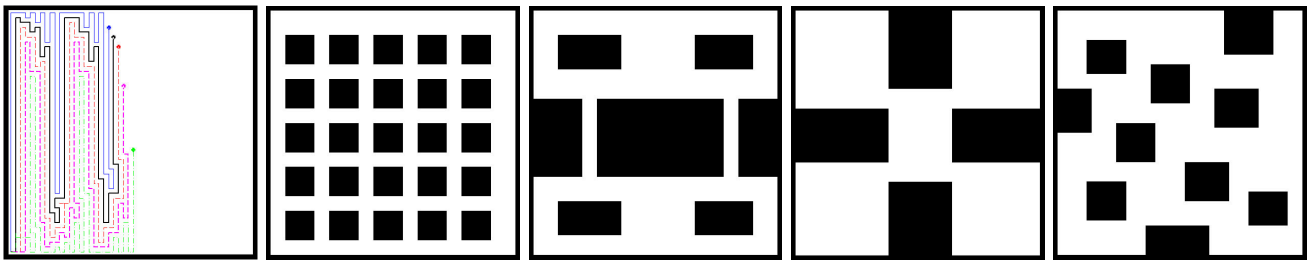


(a) 2 agents out of view of each other (b) 2 agents within view of each other

Fig. 3. (a) 2 agents which are out of one another's view. They move towards the unexplored region as indicated in their own local Voronoi region which so happen to be overlapping regions. (b) The 2 agents now come into view of one another. A local Voronoi boundary now exists between the two and both agents will continue moving without crossing this boundary.

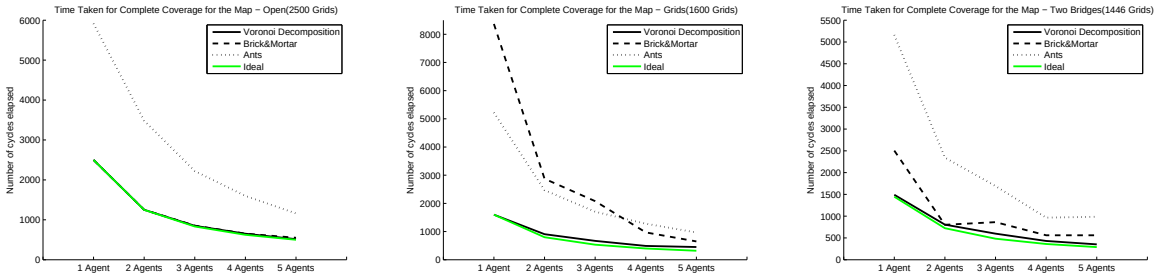
VI. Experimental Results

MATLAB was used to simulate our results. A user GUI was created for running all three algorithms to created maps as well as for easy debugging. The GUI also allowed for varying the number of exploration agents. Five maps (Fig. 4) were created to encapsulate the different possible scenarios of exploration. For each simulation run, the agents start from

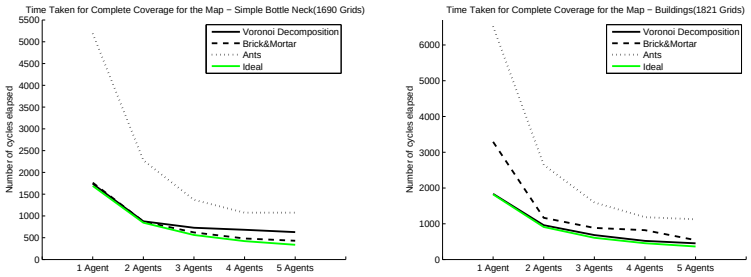


(a) Open Map (2500) (b) Grided Map (1600) (c) Two Bridges (1446) (d) Simp. Bottle Neck (1690) (e) Buildings (1821)

Fig. 4. Maps used for simulations. The bracketed numbers represent the total number of open cells present in that particular map. Each map is 50 by 50 cells. (a) A basic open map without any obstacles. (b) This map has a total of 25 obstacles (or holes) placed in an ordered manner. This arrangement allows each part of the map to have an equal likelihood of being reached. (c) This map is split into two portions (top and bottom) with two corridors linking the map portions. (d) This is an alteration of the Open Map. There are no obstacles (or holes) within the map. (e) This is an alteration of the Grided Map where the layout of the obstacles (or holes) are not in any ordered manner.



(a) Open Map - 2500 Open Grids (b) Grided Map - 1600 Open Grids (c) Two Bridges - 1446 Open Grids



(d) Simple Bottle Neck - 1690 Open Grids (e) Buildings - 1821 Open Grids

Fig. 5. Simulation results for the 5 different maps.

the bottom left cell of the given map. Figs 5(a) to 5(e) show results for the environments in Fig. 4(a) to 4(e), respectively.

All three algorithms are able to solve the five different scenarios. The performance matrix we consider is the exploration time. The exploration time is the number of steps it takes for all the agents to completely traverse all the cells in a given map at least once. Each agent makes a single step in 1 simulation cycle as it moves from a cell to an adjacent cell. So for a map with m cells, the shortest theoretical time which n agents will take to completely explore it (even if teleportation were possible and it still takes 1 step to teleport from one cell to another) would simply be m/n steps. The shortest theoretical exploration time, m/n, would also be indicated in each of the graphs (Figs 5(a) to 5(e)). This shortest theoretical exploration time would also be a good benchmark to gauge the performance of our algorithm.

Fig. 5(a) provides results of the case of a simplistic map. Both Brick&Mortar and Voronoi Decomposition algorithms show results close to the theoretical ideal one. As our Local Voronoi Decomposition method mainly focuses on the instantaneous area allocation for each agent (the Divide part of the strategy), the overall results prove to be efficient even when each agent deploys a simplistic coverage strategy (the Conquer part of the strategy) within their voronoi regions. The Ants Algorithm, however, is half as efficient in this case. This is due to the randomness involved in the Ants Algorithm.

Figs. 5(b) and 5(e) show results for maps with obstacles distributed around the map. One of the drawbacks of the Brick&Mortar algorithm is in the way it deals with obstacles in a map. When there is an increase in number of obstacles in the map, more loops are created and time is taken to identify and close these loops. The Voronoi Decomposition algorithm, on the other hand, works well despite the presence

of obstacles, as seen by the results being very close to the theoretical ideal. The Voronoi Decomposition lowers the possibility of agents traversing over already explored areas.

Fig. 5(c) shows results in the case where there are 'islands' in the map. Brick&Mortar shows a slight fluctuation in the results as more agents are introduced into the map. The results of the Voronoi Decomposition algorithm once again is very close to the theoretical ideal.

Fig. 5(d) shows results in the case where there are no obstacles or holes in the map. Brick&Mortar performs best in such maps. The absence of obstacles eliminates the occurrence of loops. Hence, there is no longer a need to detect and close them. Although the Voronoi Decomposition algorithm does not fare as well as Brick&Mortar, both algorithms are not too far away from each other and the theoretical ideal.

The results for the LVD algorithm is close to the theoretical ideal. The slight difference is due to the agents' overlapping movement - an indication of how resources (or agents) are inefficiently used. Table I shows the percentage of the cells of each map which have been traversed more than once while using the LVD algorithm. The percentages of overlapped cells (especially for the Open map) are low, which generally affirms the efficiency of the LVD algorithm. As the LVD algorithm is based on the use of local information, reduced local information will result in an overall lowered efficient use of resources. The Grided and Bottle Neck maps are examples of environments with more blind spots which in turn reduces the amount of local information that the agents have.

TABLE I

PERCENTAGE OF OVERLAPPED CELLS USING THE LVD ALGORITHM

Map:	Open	Grided	Two Bridges	Simple Bottle Neck	Buildings
Overlap(%):	1.72	24.81	12.03	20.12	13.67

VII. CONCLUSION

In this paper, we introduced a new algorithm, Local Voronoi Decomposition for task decomposition. We have shown its effectiveness in the application of the exploration problem. When using a Divide and Conquer strategy, Local Voronoi Decomposition serves as a very good Divide part of the strategy. We have shown that even with a simplistic Conquer strategy, the overall algorithm is capable of proper utilization of agents, where overlapping exploration is brought to a minimal. Our experimental results showed that our algorithm has significant improvement over leading algorithms in this area. On top of that, we have shown that our results are close to the theoretical ideal.

In the future, we would like to explore more efficient path motions to complement with the Local Voronoi Decomposition algorithm to produce a more efficient way of tackling the exploration problem. Another challenge would be to do away with relying on a central virtual memory to keep track on the explored and unexplored regions.

REFERENCES

[1] Zhao Jie and Jiang Jian, "Cooperative Multi-Robot Map-building based on Genetic Algorithms", *IEEE International Conference on Information Acquisition*, pp 360 - 364, 2006.

[2] J. L. Baxter, E. K. Burke, J. M. Garibaldi and M. Norman, "Multi-Robot Search and Rescue: A Potential Field Based Approach", *IEEE International Conference on Robotics and Automation*, Volume 2, Issue , pp 1217 - 1222, 2002.

[3] George Kantor and Sanjiv Singh, Ronald Peterson and Daniela Rus, and Aveek Das, Vijay Kumar, Guilherme Pereira and John Spletzer, "Distributed Search and Rescue with Robot and Sensor Teams", *The 4th International Conference on Field and Service Robotics*, 2003.

[4] M. Kulich, J. Faigl, and L. Preucil, "Cooperative planning for heterogeneous teams in rescue operations", *IEEE International, Safety, Security and Rescue Robotics*, Workshop, Volume , Issue , 6-9 June 2005, pp 230 - 235, 2005.

[5] I. Wagner and A. Bruckstein, "Cooperative cleaners: a study in ant robotics," Tech. Rep. 9512, CIS Report, Technion, IIT, Haifa, June 1995.

[6] V. Kumar, and F. Sahin, "Foraging in ant colonies applied to the mine detection problem", *Proceedings of the IEEE International Workshop on Soft Computing in Industrial Applications, 2003. SMCia/03*, 23-25 June 2003, pp 61 - 66, 2003.

[7] S. Koenig and Y. Liu, "Terrain coverage with ant robots: a simulation study," in *AGENTS01: Proceedings of the fifth international conference on Autonomous agents*. ACM Press, 2001, pp 600 607.

[8] J. Svennebring and S. Koenig, "Building terrain-covering ant robots: A feasibility study." *Auton. Robots*, Volume 16, No. 3, pp 313 332, 2004.

[9] Israel A. Wagner, Michael Lindenbaum and, Alfred M. Bruckstein, "Cooperative Covering by Ant-Robots using Evaporating Traces", *IEEE Transactions on Robotics and Automation*, Volume 15, Issue 5, pp 918 - 933, 1999.

[10] E. Ferranti, and N. Trigoni, and M. Levene, "Brick&Mortar: an on-line multi-agent exploration algorithm", *IEEE International Conference on Robotics and Automation*, pp 761 - 767, 2007.

[11] N. Hazon, F. Mieli, and G. A. Kaminka, "Towards robust on-line multi-robot coverage," in *ICRA06: Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, pp 1710 - 1715, 2006.

[12] I. Rekleitis, G. Dudeck, and E. Milios, "Multi-robot exploration of an unknown environment, efficiently reducing the odometry error," in *Proceedings of the International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, pp 1340 - 1345, 1997.

[13] C. Icking, T. Kamphans, R. Klein, and E. Langetepe, "Exploring simple grid polygons," in *COCOON 2005: Proceedings of Computing and Combinatorics: 11th Annual International Conference*. Volume 3595 of Lecture Notes Comput. Sci., pp 524 - 533, Springer, 2005.

[14] Christian Trefftz and Joseph Szakas, "Parallel Algorithms to Find the Voronoi Diagram and the Order-k Voronoi Diagram", *IPDPS '03: Proceedings of the 17th International Symposium on Parallel and Distributed Processing*, IEEE Computer Society, Washington, DC, USA, pp 270.1, 2003.

[15] B. Yamauchi, "Frontier-based exploration using multiple robots", in *Proceedings of the Second International Conference on Autonomous Agents*, Minneapolis, MN, USA, pp. 47-53, 1998.

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

[17] R.M. Zlot, A. Stentz, M.B. Dias, and S. Thayer, "Multi-Robot Exploration Controlled By A Market Economy", *IEEE International Conference on Robotics and Automation*, May, Vol. 3, pp. 3016 - 3023, 2002.

[18] M.B. Dias and A. Stentz, "A free market architecture for distributed control of a multirobot system", In 6th International Conference on Intelligent Autonomous Systems (IAS-6), PAGES 115-122, 2000.

[19] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," IEEE Transactions on Robotics and Automation, April 2005, Vol. 20, Issue 2, pp. 243 - 255, 2004.

[20] T. Bandyopadhyay, and Z. Liu, M.H. Ang Jr. and W.K.G. Seah, "Visibility based Exploration in Unknown Environment Containing Structured Obstacles", *Proceedings of the IEEE International Conference on Advanced Robotics*, 2005.