International Journal of Robotics and Automation (IJRA)
Vol. 6, No. 1, March 2017, pp. 49~58
ISSN: 2089-4856, DOI: 10.11591/ijra.v6i1.pp49-58

"Nearest Zero-point" Algorithm for Cooperative Robotic Search Missions

Vahid Aryai[1], Mahsa Kharazi[2], Farid Ariai[3]
[1]School of Engineering, RMIT University, Melbourne, Australia
[2]Department of Aerospace Engineering, Sahand University of Technology, Tabriz, Iran
[3]Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran

Article Info

Article history:

Received Sep 7, 2016
Revised Dec 21, 2016
Accepted Jan 6, 2017

Keyword:

Artificial intelligence
Cooperative search
Grid-based algorithm
Optimization Robotic search

ABSTRACT

Four path planning and data exchange algorithms for cooperative search and coverage robotic missions are proposed and modified. The introduced methods are simulated using C++ programming environment and the results are discussed in detail for environments with static obstacles. It has been shown that using the **"nearest zero-point"** algorithm can greatly optimize the mission duration and also overlapping of the search trajectories. Finally, the results are compared with several existing algorithms.

Corresponding Author:

Vahid Aryai,
School of Engineering,
RMIT University, Melbourne, Australia.
Email: s3578357@student.rmit.edu.au

1. INTRODUCTION

Recent years have witnessed the development of robots for carrying out different missions, such as missions those are hazardous for human beings. Search and coverage missions are among the important missions, in which robots are employed for searching an environment that is unsafe for human presence. The first issue to address while searching an environment using cooperative robots is to prevent them colliding with one other or obstacles in the environment [1]. Secondly, it is important to minimise the duration of cooperative search mission, to save energy and time. The main objective of path planning is to find an appropriate path for the movement of robots in an environment in order to avoid the collision between them, reducing the mass of the data calculations for finding the right path and also minimizing the search mission duration and energy consumption [2]. Consequently, the chosen search trajectories should be optimized to account for the mentioned factors. To this aim, robots must cooperate with one another, mainly by sharing their individual search data base, to guarantee a safe and efficient search and coverage mission.

This paper applies grid-based method for defining the search environment. Grid-based methods utilized geographically distributed locations data sources in order to provide users with the access to a scattered large database. Such techniques make use of the grid resources to perform the search tasks and also to improve the search performance. The following research intends to address both aforementioned issues in the first paragraph, by proposing four grid-based cooperative search and coverage algorithms. The efficiency of each algorithm is then evaluated in terms of mission duration and overlapping search trajectories by simulating the search mission in C++ environment and finally, a comparison between the studied algorithms and several existing search algorithms is provided.

The next section introduces the related search and path planning algorithms and their demerits. After that, the problem statement is mentioned in order to identify the gap in current literature following by the

intended contribution of the presented research to the field of knowledge. Finally, the efficiency of the proposed algorithms is measured and results are discussed.

2. RELATED WORKS

Overall, the problem of cooperative path planning can be divided into four sub-problems; expression of the search environment, path calculation, path execution and also communication between agents. Most of path planning algorithms available in the literature are based upon the theory of visibility graphs [3], Voronoi diagram [4], grid-based [5] and artificial intelligence-based [6] methods each comes with the advantages and shortcomings. For instance, Voronoi diagrams methods are one-dimensional in nature that may lead to an inaccurate representation of the search environment which eventually degrades the path planning efficiency [7]. On the other hand, artificial intelligent-based methods, i.e. genetic algorithm methods, are proven to be suitable mostly for handling small-scale problems because of their huge computational burden [8].

Among the introduced methods previously, grid-based methods are of interest of this research, since such methods are easy to set-up, fast and reliable in comparison with other mentioned techniques. Koceski and Panov applied the method successfully to a gridded environment consisted of obstacle occupied and empty cells [9]. To find the optimum path Dijkstra algorithm has been used; however, the method has its own disadvantages. In this algorithm the processing of the data of individual cells of the environment takes too long which in turn degrades the overall efficiency of the algorithm. To overcome this problem, a quadtree method has been deployed in [10].

A different approach to Dijkstra algorithm, known as A* algorithm, has been used in parallel with the Dijkstra by Zhang and Zhao [11]. Although, the process flow of the algorithm is complex, so that efficient application of A* algorithm requires deep knowledge of mathematics. The combination of hormone- inspired path planning methods, a kind of grid cell marking i.e. by numerical value etc., and the gird based methods can provide an optimum search algorithm [12]. The marking of map cells enable the robots for updating a section of the map that may contain different sorts of information and data, such as compulsory operations, hazard warnings and also number of times each robot searches that particular area. The overall efficiency of this method becomes better as the number of searching robots increases; however, the increase in number of searching agents raise the computation burden for path planning.

Generally speaking, the current literature still demands comprehensive researches for considering majority of the parameters involved in a real search mission altogether. This paper presents a novel research, since it covers issues such as search duration minimization, search trajectory overlapping minimization, efficient data transferring among agents, alongside implementation of a real-time search mission simulation of a static environment in C++ environment. Consequently, the proposed research provides solution for all of four sub-problems mentioned earlier in this section.

3. PROBLEM STATEMENT

Before implementing a robotic search mission, it is important to simulate the whole mission using computers. As a consequence, general shortcomings of a search scheme are identified and probable failures like damages to robots due to collision can be prevented. It is however, important to note the limitations of a robotic search mission simulation, which hinder the realistic evaluation of the mission. An efficient robotic search simulation must consider the real search scenarios such as presence of obstacles in search environment, limitations in communication range of search agents in cooperative search missions, energy consumption management, etc.

The following research intends to provide a realistic cooperative search simulation algorithm called “digital environment marking”, by addressing most of aforementioned limitations. In this research search environment is represented by grid of identical digitally marked cells, each contains number of times the cell was being searched. Robots can move between adjacent cells including diagonally placed cells; while, they have limited information about the other agents surveying the area. During the search mission, information can be exchanged if robots are close enough to each other. Four search algorithms based on digital environment marking concept are proposed. Finally, the goal is that each cell is visited at least once by any of the robots as soon as possible. The novelty of the presented research can be concluded in the following lines:

a. In the proposed method, the obstacles can be defined accurately, whereas the polygon division of the environment method [13] lacks such a characteristic. This helps in producing a realistic simulation of the mission.
b. Since the number of times a cell in the environment is searched is specified, the robots can easily detect the cells those have been searched less. This approach eventually reduces the overlapping of the search trajectories and also mission duration.

c. In the introduced methods, each robot saves and processes the recorded search data (cells' information) independently and updates its own data-base only when it approaches other robots; therefore, in case of the malfunctioning of one or more robots, the rest of the properly functioning robots can carry out the search mission to the end without any limitation. This kind of encounter with possible accidents has not been dealt with in the other similar methods such as the methods based on the centralized learning methods [14].
d. The algorithms do not require the robots to be in touch with each other all the time, or even with the central operator. Therefore, the optimum use of the communicational receivers and transmitters can greatly improve the energy consumption. Lack of a central operator, or in other word, this autonomously functioning method minimizes the human intervention.
e. Using an innovative data exchange technique, called "nearest-zero" algorithm, greatly improves the cooperation efficiency between robots which results in an optimized computation burden of the algorithm and also a short search mission.

The following section explains the general methodology of the research. Four grid-based cooperative search algorithms are then introduced and comparison is made between them in terms of search trajectory overlapping and also mission duration time-step.

4. RESEARCH METHOD

This section contains explanations on representing the search environment, introducing the structure of the algorithms, path planning of search agents, and the way they interact and cooperate with each other. The whole search mission has been simulated using C++ programming environment and artificial intelligence programming techniques, which includes simulation of the robots' path planning, the environment, the sensors' range, and the way robots cooperate. In this research, it has been assumed that:

a. The obstacles in the environment are all static,
b. Obstacles have been considered as a collection of square cells. If the obstacle size is smaller the cell size, then the whole cell is considered as an obstacle.
c. The search trajectory of each robot consists of segments.
d. The communication range of the robots is limited. Each robot informs the other robots of its decisions only when it is in the communicational range of them.

The decision making and obstacle sensing delay time has been ignored; therefore it does not affect the calculation time of a path.

5. ALGORITHM DESIGN

When addressing for mathematical modeling of the obstacles, before starting the simulation, it suffices to assign a high value to the cells with obstacles (Namely 999 or 9999, etc.) and "0" to the cells without obstacles in the numerical field map of the environment. After the simulation starts, each time that a cell is being searched by robots, its value increases by one unit. Therefore, the numerical value of a cell at a given iteration indicates the number of times that the cell has been searched until that iteration.

In general, dividing the environment into cells and the necessity of cooperation between the agents requires that each agent to deal with 9 cells simultaneously. The algorithm starts with the robots situated at their initial positions. As the search mission starts, each robot searches its 8 neighboring cells and then moves to cell with minimum value as shown in Figure 1.

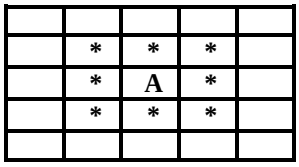


Figure 1. Eight different choices (Cells indicated by "*") for the movement of the robot A

If the minimum values of several neighboring cells are the same, algorithm chooses one of them at random. As the robots come approach each other so that their position becomes within the sensing range of the other robots, they exchange their recorded search data and update their own maps of the environment so that they all become identical. As it will be seen later, the way a robot combines its own data with those that it receives from the others significantly affects the performance of the algorithm.

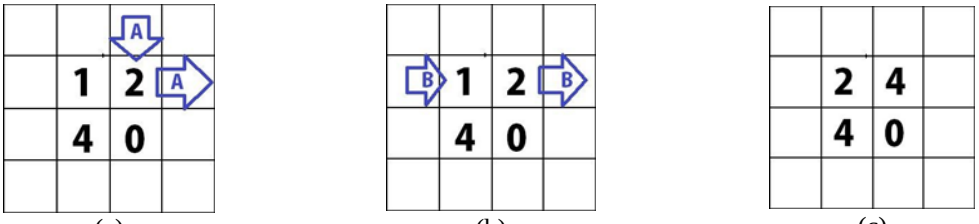
The search algorithm continues until all the cells have been searched and finally when each robot searches the last "0" value cell, the mission terminates and the number of search iterations (search duration) and the final map including final the numerical values of the cells are sent to the printer as the output. The pseudo-code of the digital marking method is as follows:

```
1:  voidsearchMap{
2:      Let time to 0
3:          for all robots do{
4:          if all the cells have been searched
5:              return number of iterations
6:          Let Around[8] to the value of the eight neighboring cells of thisRobot
7:          Let Minimums[] to the minimums of the Around array
8:          Target=choosing a random block in the Mimimums array
9:                      move the robot position to Target
10:                     value(robot.position)=value(robot. position) + 1
11:         for i from 0 to the number of robots
12:             if thisRobot and robot[i] are close to each other
13:                     changeData(thisRobot, robot[i])
14:         time=time + 1
15:                 }
16:         }
```

5.1. Z Data Exchange Algorithm

A similar approach to the digital marking method called "Z data exchange" algorithm is introduced in this section. The difference between the algorithm and the digital marking method is in the way robots update their maps when they exchange their recorded search data-base. For the sake of clarity, suppose that the agent A and B, without getting close to one another, search certain parts of the map within different time intervals as shown in Figure 2(a) and (b). According to the criteria of the Z data exchange method, when the agents A and B approach each other, their maps will look like as the one in Figure 2(c), where the numerical value of each cell indicates the total number of times the robots have searched it. The pseudo-code for the Z data exchange method reads as follows:

```
1:  voidchangeData (robot a, robot b) {
2:      Let t* to time
3:      Let t0 to last time robots have seen each other
4:      Let h(i,j,t) to the value of (i,j) cell in the map at time t
5:      Let finalValue[][]
6:          for i from 0 to row
7:          for j from 0 to column
8:              finalValue=a.h(i,j,t*) + b.h(i,j,t*) - b.h(i,j,t0)
9:
10:         a.value=finalValue
11:         b.value=finalValue
12: return
13:                                 }
```



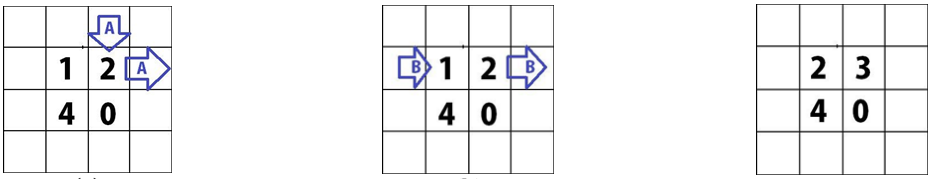
(a) (b) (c)

Figure 2. (a) and (b) are the way that robot A and B search the shown part of the map, (c) The same map after the robots met

5.2. Z* Data Exchange Algorithm

The second data exchange algorithm, called the "Z* data exchange", follows the same procedure as the previous algorithm; however, the only difference between them is that when the robots meet and compare their maps with each other, the maximum value of each cell is recorded on the final map as shown in Figure 3. Therefore, in this method the numerical value of the cells at a given iteration of the mission is not the number of times that the cells have been searched anymore. As we shall see later on, this method is more efficient than the previous one. The pseudo-code for the Z* data exchange method reads as follows:

```
1:  voidchangeData(robot a, robot b){
2:      Let finalValue[][]=0
3:          for i from 0 to row
4:              for j from 0 to column
5:                  finalValue[i][j]=maximom of a.value(i ,j) and b.value(i, j)
6:              a.value=finalvalue
7:              b.value=finalValue
8:  return
                    }
```



(a) (b) (c)

Figure 3. (a) and (b) are the way that robot A and B search the shown part of the map, (c) The same map after the robots met

5.3. Double-Layer Search Algorithm

The two previously mentioned algorithms were intended to alter the data exchange criteria of digital marking method. Additionally, the following two algorithms are expected to change the path planning algorithm of the digital marking method. In the first algorithm called "double-layer search", each robot not only searches its nearest 8 neighboring cells but it also searches its 16 second nearest neighbors. Therefore, when the minimum numerical values of more than one of first nearest 8 cells are equal, then the robots' next destination cell is not chosen at random.

In this case, for each cell the numerical values of its 3 nearest neighbors are also taken into account and minimum of the sum of numerical values of these 3 neighboring cells is chosen as the destination for the robot. For example, in Figure 4, if the numerical values of the cells 0* and 5* are equal and also minimum among 8 first neighboring cells of robot A, then the robot A calculates both the sum of the numerical values of the cells 8, 9, and 10, and the sum of the numerical values of the cells 11, 12, and 13. Therefore, regard to which one of them is minimal, the robot moves either to cell 0 or cell 5. The pseudo-code for the double-layer search method is as follows:

```
1:  voidsearchMap{
2:      Let time to 0
3:          for all robots do{
4:              if all the cells in the map have been searched
5:                  return number of iterations
6:                  Let Around[8] to the value of the eight neighboring cells
                        thisRobot
7:              for i from 0 to 8
8:                  Around[i]=Around[i] + aroundValus(Around[i].position)
9:                  Let Minimums[] to the minimums of the Around array
10:                 target=choosing a random block in the Minimums array
11:                 move the robot position to target
12:                 value(robot.position)=value(robot.position) + 1
13:             for i from 0 to number of robot
14:                 if thisRobot and robot[i] are close
```

```
15:                                changeData(thisRobot, robot[i])
16:                          time=time + 1
17:                                }
18:                    }
```

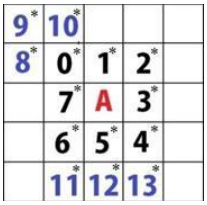


Figure 4. Double-layer search using robot A (Each number with "*" identifies a specific cell and should not confused with the numerical value of the cell)

5.4. Nearest Zero-Point Search Algorithm

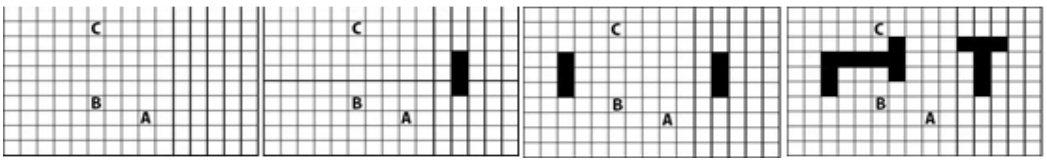
A novel path planning method, called the "nearest zero-point" search algorithm is presented herein. In this algorithm, each robot, not only searches the numerical values of its neighboring cells, but also, if it necessitates, it searches every cells in the environment and moves through the direction to the nearest cell that has not yet been searched. In fact, each agent measures the distance of its 8 neighboring cells from the nearest cell yet to be searched and chooses a neighboring cell as the origin of its next move that has the least distance from the nearest cell with zero value. If on its way to this cell, the agent encounters other agents, they start exchanging their stored data and if it finds that one of these robots is going to search a cell that it had already intended to search, changes its path and moves towards the nearest cell that has not yet been searched. In case the next destination of the robot is occupied by obstacles, the neighboring cell with minimum value that is still in nearest distance to a zero cell will be selected. The pseudo-code for the "Nearest-Zero Point" Algorithm reads as follows:

```
 1:    float distToNearestZeroPlace(position a){
 2:          Let Mine to 10000
 3:          for i from 0 to row
 4:                for j from 0 to column
 5:                if min < (i - a.x) ^ 2 + (j-a.y)^2
 6:                min=(i - a.x) ^ 2 + (j - a.y) ^ 2
 7                 return mine
8:     }
9:     void searchMap{
10:          Let time to 0
11:          Let mines[]
12:                for all Robot do
13:                      if all the cells in the map are searched
14:                      return number of iterations;
15:    Let Around[8] to the distToNearestZeroPlace of the eight cells around thisRobot
16:                                Let Minimums[] to the minimums of the Around array
17:                                target=choosing a random block in the Minimums array
18:                                move the robot position to target
19:                                value(robot.position)=value(robot.position) + 1
20:                      for i from to number of Robot
21:                                if thisRobot and Robot[i] are close
22:                                      changeData(thisRobot, robot[i])
23:                time=time + 1
24:  }
```

6. MISSION SIMULATION

In this section the functionality and efficiency of aforementioned algorithms is evaluated. To this end, four simulation environments have been defined. The dimensions of the environments are all equal to

10×15 and they differ only in the number and distribution of the obstacles. In the first simulation experiment, the environment is considered with no obstacles as shown in Figure 5(a). Such an environment provides a possibility to examine the efficiency of cooperation among agents using each algorithm. In the second simulation experiment, a slab covering three cells, a local maximum or, in other words, a potential barrier [14], has been created as shown in Figure 5(b). In the third simulation experiment, a second slab has also been added, so the efficiencies of the algorithms could be studied in the presence of obstacles having no corner as shown in Figure 5(c). Adding the second obstacle has decreased the search area and, on the other hand, has increased the number of the local maxima. Therefore, the encounter of the algorithms with these two factors could be checked. In the fourth simulation experiment, more complicated obstacles are introduced and therefore the study of the efficiencies of the algorithms in dealing with the cells bounded from three sides becomes possible as shown in Figure 5(d).



(a) (b) (c) (d)

Figure 5. The simulation environments and the initial positions of the robots A, B, and C, at the onset of the search mission

7. RESULTS AND DISCUSSION

The results of the simulations in different environments are presented in this section. Table 1 shows the number of iterations takes for each one of mentioned algorithms to completely search the environments A to D as shown in Figure 5 using three robots. The results reveal that when the number of obstacles increases, the Two-layer data exchange algorithm is more efficient than the Z data exchange algorithm which is due the structural difference between these two algorithms. This is a little improvement and does not show a difference more than 9 search iteration in whole of the search mission. In addition, Table 1 shows that when number of obstacles and corners of the environment increases, the Z* data exchange algorithm reduces the number of the search iterations by 21 iteration, and in comparison with the Two-layer data exchange algorithm, does a superior improvement. As is evident, "Nearest zero-point" algorithm has the least search iterations among four studied methods; it can reduce the search iterations of the mission by up to 20, 30 and 38 iteration compare with Z* data exchange, Two-layer data exchange and Z data exchange algorithm respectively.

In order to investigate the effect of the number of robots on the number of mission completion iterations using the presented algorithms, another simulation experiment has been conducted. For this simulation, the third environment in Figure 5 has been selected and different numbers of robots have been employed. The reason behind choosing the environment is its simple structure that allows for checking the data exchange parts of the algorithms. The results of the simulation are given in Table 2.

Although it may seem obvious that the increase in the number of robots would cause decreasing the search iterations, it should be noted that the rate of the decrease in the search iterations is not the same for different algorithms and, as we shall see later, this indicates that some algorithms have a better cooperative functionality than the others. Table 2 shows that, considering Z* and "Nearest zero-point" algorithms, not only the effect of the number of searching robots on the decreasing of the iterations of the search mission is much better than the other two algorithms, but also when one robot is employed in the search mission, the obtained results are comparable to the results of more complex data exchange algorithms [15] which is explained later in this section.

Table 1 Simulation Results for the Environments with Different Number of Obstacles being Searched by Three Robots, The Total Number of the Cells in all these Environments is 150

Environment Types	Number of obstacles	Number of search iterations in the Z data exchange Algorithm	Number of search iterations in the two-layer data exchange Algorithm	Number of search iterations in the Z* data exchange Algorithm	Number of search iterations in the "Nearest zero-point" data exchange Algorithm
A	0	77.8	73.2	63.7	57.2
B	3	85.2	80.8	72.2	52.4
C	6	88.8	79.8	70.5	50.8
D	15	80.3	76.9	55.3	46.2

Table 2 Effect of Number of the Agents on the Number of Mission Iterations

Algorithm	Number of iterations for **one** agent	Number of iterations for **two** agents	Number of iterations for **three** agents
The Z data exchange	175	110	85
Two-layer data exchange	162	92	80
The Z* data exchange	156	88	72
"Nearest Zero-point" data exchange	121	77	52

In the second simulation experiment, the overlapping of the search trajectories has been studied. One of the criteria for measuring the optimality of a search algorithm is that the robots must search the environment uniformly, so that the number of the overlapping trajectories has been kept at minimum. It is clear that if the mission is performed with lesser iterations, then the overlapping of the search trajectories will decreased. However, in the simulation of some of the search algorithms, although the number of the search iterations is low, it is observed that the environment has been searched irregularly and non-uniformly and therefore some of the cells have been searched repeatedly. This, in turn, degrades the efficiency of the algorithm. In this experiment, by assigning different colors to different number of times each cell has been searched, a visual map of the overlapping searches has been prepared. The results of this experiment performed using the aforementioned algorithms considering different number of agents and search environments are depicted in Figure 6, 7 and 8.

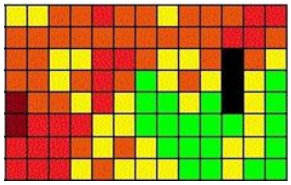
Figure 6(a) and 6(b), 7(a) and 7(b), and also 8(a) and 8(b) show that, compare to the Z data algorithm, the overlapping of the search paths is much lesser for the Two-layer data exchange algorithm. As a matter of fact, some of the simulations done with the Z data exchange algorithm revealed that some environment cells were searched more than 14 times, while this did not occur when using the Two-layer data exchange algorithm.

Z* data exchange algorithm, where the data exchange pattern has been altered, yielded much better results than the Z and Two-layer data exchange algorithms as shown in Figure 6(c), 7(c) and 8(c). The numerical value of the environment cells indicates that when the Z* algorithm exchanges the data, the numerical values of the cells in the resultant maps are uniformly distributed over the search map, which therefore avoids generation of the large localized numerical values that is considered as a local maxima. This, in turn, provides a better search efficiency, especially for searching cells around the obstacles during the search mission. As is evident from Figures 6(c), 7(c) and 8(c), no cell has been searched more than 4 times while using Z* data exchange algorithm.

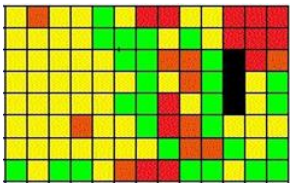
In general, considering the overlapping of the search trajectories indicates that the more uniform are the colors of the cells, the closer are the numerical values of the cells and therefore the search process is performed more smoothly. For example, the colors are more scattered in Figure 6a than in Figure 6d. Since the concept of the gradient has been used in this research to search the neighboring cells, it is clear that the "nearest zero-point" algorithm is more efficient than the other three algorithms. This is because when there is a cell that has a numerical value much larger than the numerical value of its adjacent cells then it produces a greater gradient in that region and the robot searching the region gets confused and as a result the number of the search iterations increases.

In the "nearest zero-point" algorithm, a different method of path planning is used in all of the simulation experiments. This algorithm reduces the number of iterations in the search mission. In an identical environmental conditions, when the effect of the number of the searching robots is considered, the efficiency of this algorithm and also Z* data exchange algorithm are approximately up to 10% better than that of given in [15]. When number of robots increases from one to two robot, the number of search iterations drops approximately 43% and 37% using "Nearest zero-point" and Z* data exchange algorithm respectively, while this rate is 33% for the given algorithm in [15].

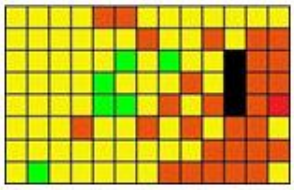
Although the efficiency of the "nearest zero-point" algorithm in comparison to that of the Depth-first [16] is up to 30% better (Considering identical search environments) while using one robot (61 iteration in comparison with 86 iteration of the Depth-first), when the number of the robots is increased the situation is reversed and the Depth-first based algorithm shows up to 25% better performance while using three robots. On the other hand, Figures 6(d), 7(d) and 8(d) show that the number of the overlapping trajectories is a minimum in the "nearest zero-point" and most of the cells have not been searched more than twice. However, the "nearest zero-point" algorithm is superior to the rest of algorithms studied in this research, it searches every points on a given map and therefore its computational load is much more than that of the other studied algorithms and it is expected to degrade the efficiency of the search when the simulation of the missions carried out by real robots.



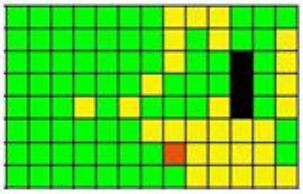
(a) The Z data exchange algorithm



(b) Two-layer data exchange algorithm

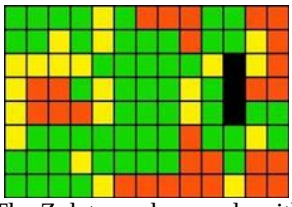


(c) The Z* data exchange algorithm

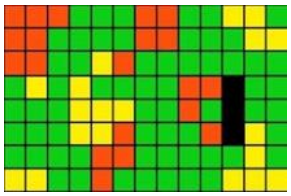


(d) "Nearest Zero-point" data exchange algorithm

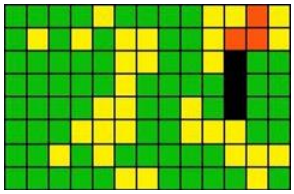
Figure 6. The overlapping of the search paths, using different search algorithms in the second environment in Figure 5b by three agents, where green, yellow, orange, and red cells are the cells that have been searched once, twice, three to eight, and eight times or more, respectively



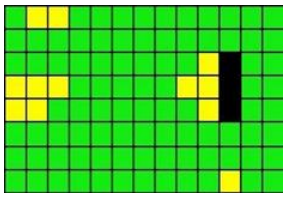
(a) The Z data exchange algorithm



(b) Two-layer data exchange algorithm

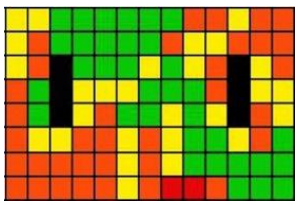


(c) The Z* data exchange algorithm

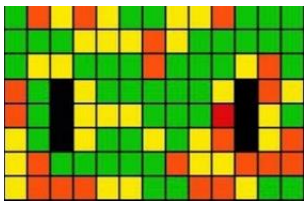


(d) "Nearest Zero-point" data exchange algorithm

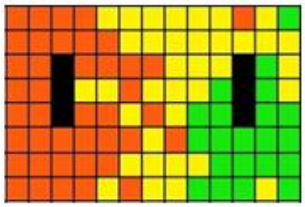
Figure 7. The overlapping of the search paths, using different search algorithms in the second environment in Figure 5b by five agents, where green, yellow, orange, and red cells are the cells that have been searched once, twice, three to eight, and eight times or more, respectively



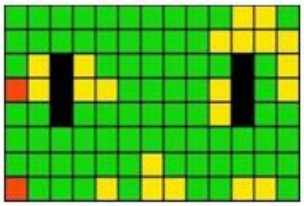
(a) The Z data exchange algorithm



(b) Two-layer data exchange algorithm



(c) The Z* data exchange algorithm



(d) "Nearest Zero-point" data exchange algorithm

Figure 8. The overlapping of the search paths, using different search algorithms in the third environment in Figure 5c by three agents, where green, yellow, orange, and red cells are the cells that have been searched once, twice, three to eight, and eight times or more, respectively

8. CONCLUSION

Several search algorithms have been introduced in this research. The final algorithm, "nearest zero-point" algorithm, shows an effective performance in comparison with previously introduced methods. The results of simulations have been also comparable, and even better in aspects like search iteration and data transfer, with existing algorithms such as those introduced in [15, 16]. However, when the number of robots increases, the performance of the Modified Depth First Method becomes better than that of "nearest zero-point". The number of overlapping of the search paths in "nearest zero-point" had been also the least among four other methods studied in this paper so that each cell was not searched more than twice. Although the "nearest zero-point" method performs better than the other methods in many respects and can provide more realistic simulation of the robotic missions, it requires much heavy computational load than simpler algorithms like Z* data exchange, since in this technique every point of the search environment is taken into account. Therefore, more study is needed in order to solve the problem. On the other hand, in the future studies, the proposed algorithms must be simulated using real robots, since software simulations cannot reflect the real characteristics of a search mission.

REFERENCES

[1] Zheng-Hua, Y., et al., "Path Planning for Coalmine Rescue Robot Based on Hybrid Adaptive Artificial Fish Swarm Algorithm", *Indonesian Journal of Electrical Engineering and Computer Science*, 12(10): 7223-7232, 2014.

[2] Benaoumeur, I., et al., "Backstepping Approach for Autonomous Mobile Robot Trajectory Tracking", *Indonesian Journal of Electrical Engineering and Computer Science*, 2(3), 2016.

[3] Huang, H. P. and Chung, S. Y., "Dynamic visibility graph for path planning. In Intelligent Robots and Systems", 2004.(IROS 2004). *Proceedings. 2004 IEEE/RSJ International Conference* on (Vol. 3, pp. 2813-2818). IEEE. 2004.

[4] Bhattacharya, P. and Gavrilova, M. L., "Roadmap-based path planning-Using the Voronoi diagram for a clearance- based shortest path.", *IEEE Robotics & Automation Magazine*, 15(2), pp.58-66, 2008.

[5] Zhou Shao, David Taniar, Kiki Maulana Adhinugraha, "Voronoi-based Range-NN search with Map Grid in a mobile environment", *Future Generation Computer Systems*, Volume 67, Pages 305-314, 2016, ISSN 0167-739X.

[6] Kapanoglu, M., Alikalfa, M., Ozkan, M., Yazıcı, A. & Parlaktuna, O., "A pattern-based Genetic Algorithm for Multi-Robot Coverage Path Planning Minimizing Completion Time", *Journal of Intelligent Manufacturing*, 23, 1035-1045, 2012.

[7] Masehian, E. & Amin-Naseri, M. R., "A Voronoi Diagram-Visibility Graph-Potential Field Compound Algorithm for Robot Path Planning", *Journal of Robotic Systems*, 21, 275-300, 2004.

[8] Coello, Coello, Carlos A., Gary Lamont, and David Van Veldhuizen, "Evolutionary Algorithms for Solving Multi- Objective Problems", Genetic and Evolutionary Computation Series. New York: Springer Science & Business Media LLC. 2007. doi:10.1007/978-0-387-36797 2

[9] Panov, S., Koceski, S., "Harmony search based algorithm for mobile robot global path planning", In: 2nd *Mediterranean Conference on Embedded Computing (MECO)*, 15-20 June 2013, pp. 168-171, 2013.

[10] M, R. H., Hring, Schilling, H., Sch, B., Tz, Wagner, D. & Willhalm, T., "Partitioning Graphs to speedup Dijkstra's Algorithm", *J. Exp. Algorithmics*, 11, 2.8, 2007.

[11] Koceski, S., Panov, S., Koceska, N., Zobel, P.B., "Durante, F.: A Novel Quad Harmony Search Algorithm for Grid- based Path Finding Regular Paper", *Int J Adv Robot Syst 11*. 2014, doi: Artn 144.

[12] Zhang, Z. and Z. Zhao, "A Multiple Mobile Robots Path Planning Algorithm Based on a-Star and Dijkstra Algorithm", *International Journal of Smart Home*, 8(3): 75-86, 2014.

[13] Maza, I., Ollero, A., "Multiple UAV Cooperative Searching Operation Using Polygon Area Decomposition and Efficient Coverage Algorithms", In: Alami, R., Chatila, R., Asama, H. (eds.) Distributed Autonomous Robotic Systems 6. pp. 221-230, 2007, Springer Japan, Tokyo.

[14] Park, M. G., Lee, M. C., "A New Technique to Escape Local Minimum in Artificial Potential Field Based Path Planning", *KSME International Journal*, 17(12), 1876-1885, 2003, doi: 10.1007/bf02982426.

[15] Sujit, P.B., Beard, R., "Multiple UAV Exploration of an Unknown Region", *Ann Math Artif Intel* 52(2), 335-366, 2009, doi:10.1007/s10472-009-9128-7

[16] 13. Ghoul, S. E., Hussein, A. S., Abdel-Wahab, M. S., Witkowski, U., Rückert, U., "A Modified Multiple Depth First Search Algorithm for Grid Mapping Using Mini-Robots Khepera", *JCSE* 2(4), 321-338, 2008.