Probabilistic Multi Robot Path Planning in Dynamic Environments: A Comparison between A* and DFS

Safaa H. Shwail
Assist Lecturer
College of Science
University of Babylon, Babylon, Iraq

Alia Karim
Assist Professor
Department of Computer Sciences
University of Technology, Baghdad, Iraq

Scott Turner
Associate Professor
Department of Computing and Immersive Technologies
University of Northampton, Northampton, UK

ABSTRACT
In this paper, a probabilistic roadmap planner algorithm with the multi robot path planning problem have been proposed by using the A* search algorithm in a dynamic environment. The whole process consists of two phases. In the first phase: Preprocessing phase, the work space is converted into the configuration space, constructing a probabilistic roadmap graph in the free space, and finding the optimal path for each robot using a global planner that avoids the collision with the static obstacles. The second phase: Moving phase, moves each robot in a prioritized manner from its starting point to its ending point through a near optimal path with avoiding collision with the moving obstacles and the other robots. A comparison has been done with the depth first algorithm to see the difference. The simulation results shows that choosing A* search algorithm affect positively the speed of the two phases together in comparison to the depth first search algorithm.

General Terms
Artificial intelligence, Robot path planning.

Keywords
Multi-robot, path planning, decoupled planning, A*, Depth First Search (DFS).

1. INTRODUCTION
Path planning of both single robot and multiple robots has been widely investigated [1] because of its potentially usefulness in many applications such as moving containers in harbors, storage systems in factories and luggage handling systems at airports [2]. But the applications of path planning are not just restricted to the field of robotics, but also can be used in another fields such as virtual environments, computer aided design, and maintenance planning [3].

The movement of these robots should take the shortest path between the starting point and the ending point and avoid the collision with the obstacles that may occur in its way such as walls, peoples and other robots.

In most of the problem solving approaches there are measures that specify the quality and goodness of the approach. In robot path planning, these measures as defined by [4] are:

1. Completeness: the planner should find the solution (path from start to goal) if there is one, in some situations the planner can't guarantee to find the solution even if it exists; this is due to some problems like dead-lock.

2. Optimality: the solution or the path that is found should be the shortest between all the potential solutions exists.

3. Uncertainty: in some situations the robot may have little or no information about the environment or its work space, so how the planner can deal with such situation to find the path.

Path planning problems can be categorized either according to the number of the robots in the workspace or to the type of the environment used. In the first categorization, the problem can be a single or a multi robot path planning.

The problem of multi robot in a 2D rectangle with moving rectangles as obstacles is proved to be NP-hard [5]; as the number of robots increased, the degree of freedom (DoF) increased, which causes increased computation time and complexity [4]. The multi-robot problems also can be classified as centralize (coupled) or decentralized (decoupled) according to the way that the robots are organized [6]. Decoupled approaches plan the path independently for each robot, then modify these path to prevent the collision that may occur between the robots. Coupled approaches deal with the robots as one composite robot that combines the sum of all the degree of freedom DOF [7]. The decoupled approaches can be either prioritized or coordinated. The second type of categorization is depend on the environment of the robots, static or dynamic, which means static or moving obstacles. Fig.1 shows most of the situations that can be found on the robot path planning problem.

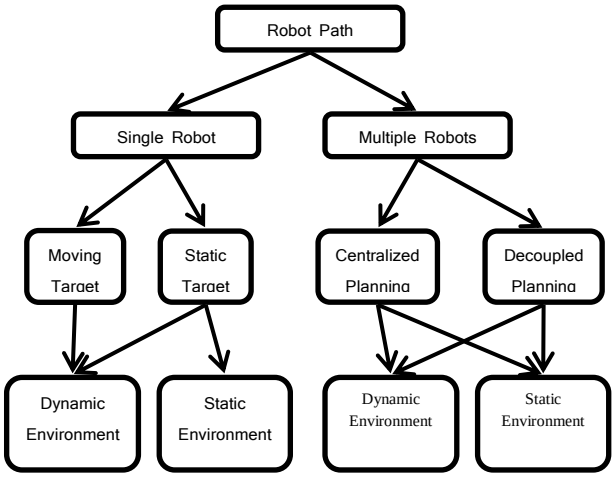

Fig.1: Categorization of Robot Path

In this paper, a probabilistic roadmap planner algorithm with the multi robot path planning problem has been implemented by using the A* search algorithm in a dynamic environment. Previously [3, 8] used a global and a local planners, in this implementation just one planner we used that specify the near

optimal path and convert it to a trajectory. In each step in this trajectory, a detection collision check is made to avoid the moving obstacles. The whole process consists of two phases. In the first phase (Preprocessing phase), the work space is converted into the configuration space, constructing a probabilistic roadmap graph in the free space, and finding the optimal path for each robot that avoids the collision with the static obstacles. The second phase (Moving phase), moves one robot at a time according to a predefined priority from its starting point to its ending point through its optimal path and avoiding collision with the moving obstacles and the other robots. The same approach is implemented using depth first search instead of A* to see which one is best and give good results in speed of execution and optimality.

This paper is organized as follows; section 2 shows the related work in the literature, in section 3 we discuss the proposed algorithm, section 4 shows the simulation results, and finally we discuss the results in section 5.

2. PREVIOUS WORK

The multi robot path planning problem can be categorized to either centralized approach or decoupled planner. The centralized approaches [9] deal with the robots as a single composite robot and any single robot path planning approach can be applied to it to find a solution. Theoretically, it is assumed complete, that it is find optimal solution if one exists. But it suffer from the complexity, that if the number of robots increased then the time needed to find the solution increased exponentially. A global cell decomposition approach was taken by [10], where in a unified configuration space representation the obstacles and other robots are incorporated. The algorithm first decomposed the free space into cells, and then it searches for a path through the resulting adjacency graph. An attractive potential fields used in [11] over the workspace which applied to a specific point on the robot body, and then these potentials are combined in configuration space to attract the whole robot toward the desired goal. An algorithm [12] is proposed to solve centralized multi robot path planning specifically on a graph have at least two empty vertices. Two primitives have been employed in this algorithm: push and swap. The former primitive used where a robot moves toward its goal until no progress can be made. The later allows two robots to swap positions without altering the position of any other robot. On the other hand, the decoupled multi robot path planning finds the path for each robot independently. It differs from the centralized in the completeness and the complexity. In some problems, there is no guarantee to find a solution even a one is exists. Planning for each robot individually make it less complexity than the centralized approaches. The decoupled approaches also categorized into prioritized [13, 14, 15, 16] planning and path coordination. The prioritized path planning was first proposed by [17], where priorities are assigned to each robot either from motion constraints or randomly.

Path coordination planner [1, 2, 18] decomposes the planning problem into path planning and velocity planning. In the first step, the path planning generates individual robot paths independently, using any common single robot path planners. The second step, it plans a velocity profile that each robot should follow while it moving to avoid collisions with other robots.

A sate time space is proposed [8, 19] as collision avoidance approach in dynamic environments. It is represented by a two dimensional diagram, the horizontal axis represents the time and the vertical axis represents the moving obstacles. When an arc of the network crosses by an object moving in the plane, the moving object cover temporarily some portion of the arc. This portion is represented as a polygonal area in the diagram.

Most of the multi robot path planning used the probabilistic roadmap method (PRM) [3, 20] to build a graph in the free space part of the configuration space. The solution obtained by this method represent a near optimal because of the randomly generation of the graph. Decomposing the map of the multi-robot path planning into subgraphs of particular known structure (cliques, halls, and rings) [21, 22, 23], which place constraints on which robots can enter or leave at a particular time. It made possible to plan hierarchically which can provide a significant improvement in planning time over a non-hierarchical planner. A modified algorithm D* or Dynamic A* [24], which is an extension to the original A* used in almost all the path planning problem. The name dynamic is used because the arc cost can change during the problem solving process, so the algorithm can re-plan locally. Then an extension to D* is presented [25], which reduce computational costs and minimize state expansions by focusses the cost updates.

3. PROPOSED ALGORITHM

The proposed algorithm consists of two phases: the preprocessing phase and the moving phase. The former is responsible of graph generation and path selection, and the latter is responsible of robots moving and collision avoidance.

3.1 Preprocessing phase

The preprocessing phase is shown in Figure 2. The first step of this phase is the dilation process. The dilation operation grows or thickens objects in a binary image, it is controlled by an object called "structuring element". The result of this process is converting the work space (W) to configuration space (C). In our work, we assumed the robots to be a rigid body with the same shape which is a disc and the center of the disc as a reference point. As a result of the dilation process, the configuration space will contain point robots and expanded obstacles. So it will be easier to move a point than moving a circle.

Sliding the robot from its reference point around the each obstacles in such a way that they are always in contact, as shown in Figure 3. Where $C=\mathbb{R}^n$ for n=1, 2, and 3, and the robot is a rigid body that is restricted to translation only, for any two sets $X, Y \subset \mathbb{R}^n$ the dilation is computed from Equation 1.

$$X \oplus Y = \{x + y \in \mathbb{R}^n | x \in X \text{ and } y \in Y\} \quad (1)$$

In the second step, it is necessary to set the points of the robots (starting and ending) as obstacles to prevent them from coincide with the graph vertices when we build the graph in the next step. The third step in this phase is the graph generation, on which finding the solution and even the optimality of the solution depend. It starts by randomly generating points in the free part of the space which is not occupied by the obstacles.

These points form the vertices of the graph are constructed to be used in the search for a solution path. The number of points to be generated must be chosen carefully because it affects in execution time and the optimality of the solution. If a large number of points are generated then this may produce a near optimal solution, but it resulting in a high time execution while searching for this solution.

On the other hand, if a small number of points generated then this will cause in a poor connected graph with lower coverage in the configuration space, affecting the optimality of the solution or even find the solution itself. The proportion of the free space to the occupied space is also an important factor that must be considered in choosing the number of points.

The next step will be connecting each vertex to every reachable vertex by edges to complete building the graph. A reachable vertex is one where there are no obstacles between the vertexes. Now, the graph is ready to find the optimal path to every robot by connecting robot's starting and ending points with the nearest vertex to each. Then a check is made to see if there is at least one path from the starting point to the ending point. If this check is fail, this mean that there is no such a path connect the points then we need to generate a new graph.

As a final step, the paths of the robots beginning from the robots starting point and ending with the robots ending point are converted to trajectories. These trajectories are represents a small unit steps that the robots will move in each unit time towards the goal. The steps are computed from Equation 2, where v_p represents the velocity of the robot, Δt is a small amount of time between each move, and l is the edge length.

$$\Delta x = v_p \Delta t / l \quad (2)$$

Algorithm (1) state how to build the roadmap graph.

Algorithm 1 Graph Generation

```
1:  i=0;
2:  while i < Number_Of_Points
3:     randomly generate vertex(x,y);
4:     if vertex(i) ∈ C_free then
5:        Graph.add_vertex(vertex(i));
6:        i= i + 1;
7:        for each a ∈ neighborhood(vertex(i),Graph)
8:           if ((not Graph.same_component(vertex(i), a)) and
                connect(vertex(i), a)) then
9:           Graph.add_edge(vertex(i), a);
10:          endif
11:       endfor
12:   endif
13:  endwhile
```

3.2 Moving Phase

According to a pre-assigned priority to robots, they are moved one after another through its trajectories one step (Δx) at a time *t* with a small amount of time Δt between robots moves. In each step, the planner will check for collision against moving obstacles and other robots, depending on the result of the collision checking it decides the next move. The next move that any robot can do is either moving forward ($+vp$), stop in place (0), or moving backward ($-vp$). The speed (vp) considered to be constant and one step at a time. Figure 4 shows the moving phase which starts at time zero.

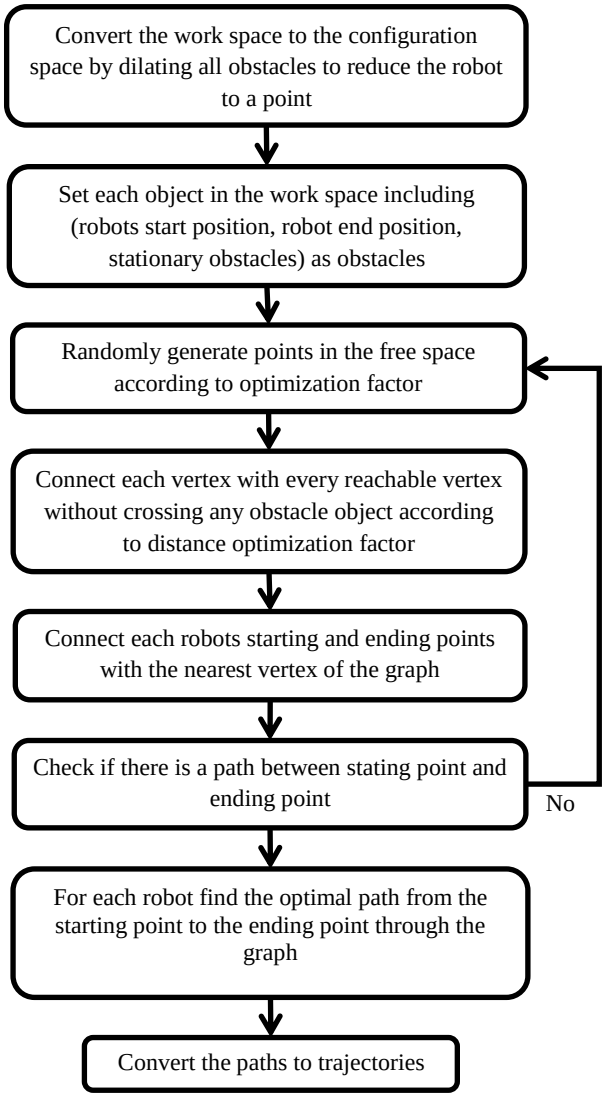

Fig.2. Preprocessing Phase

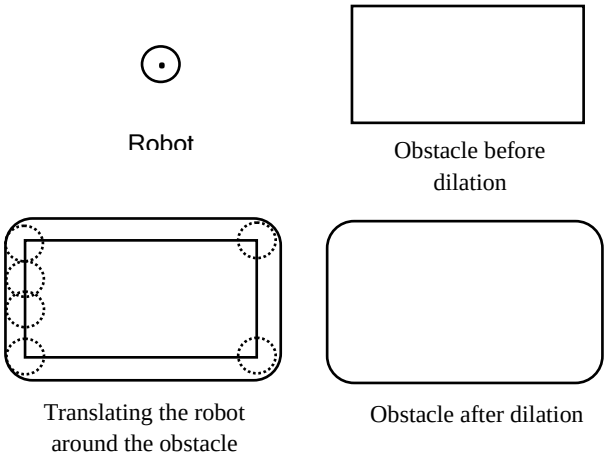

Fig.3. Dilation process

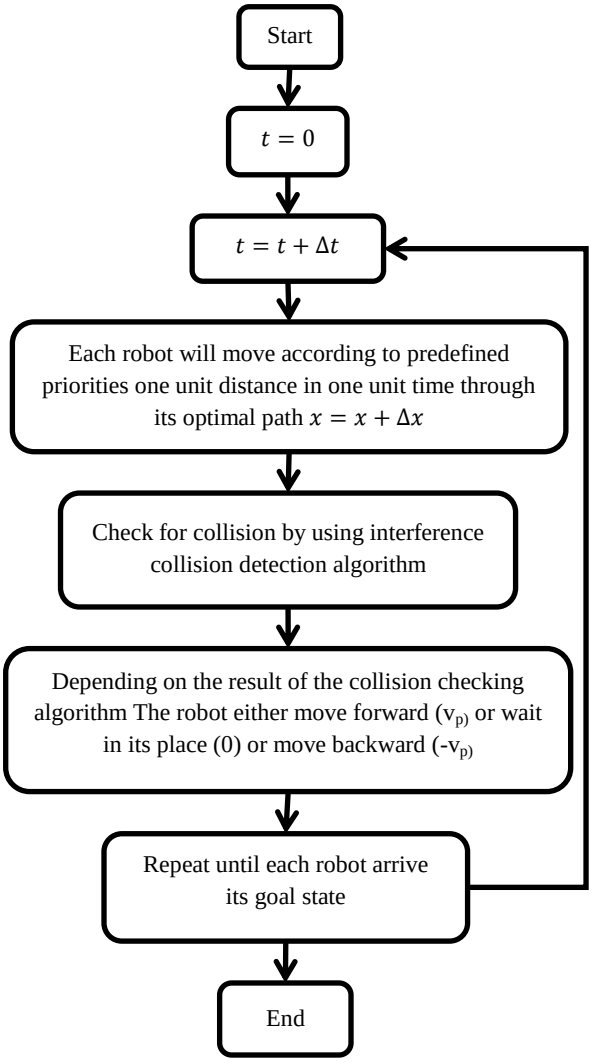

Fig.4. Moving

4. SIMULATION RESULTS

The simulation has been done by software built using MATLAB R2011a. The environment represented by a binary image of 400 * 400 dimensions. Each pixel can be either zero (free cell) or one (occupied cell). Figure 5 shows a snapshot of the configuration space with the static obstacles (big blocks) and the moving obstacles (small blocks). The robots starting points (Rs_n) are in the bottom and its goal points (Rg_n) in the top.

As a part of the preprocessing phase, the node generation in the free space part of the configuration space is shown in Figure 6, where the process is done randomly to ensure a uniform distribution. The number of nodes generated here is 100, representing a good balance between a high connected graph and a poor connected graph. Figure 7 show how these nodes are connected by edges to complete the graph that will be used in finding the near optimal path between the robots points. The connection is restricted to a threshold that specifies the long of each edge in the graph and prevents the graph to become highly connected. In Figure 8, a trace of the robots movement can be seen through the configuration space from its starting points to its ending points and avoiding the obstacles in its way.

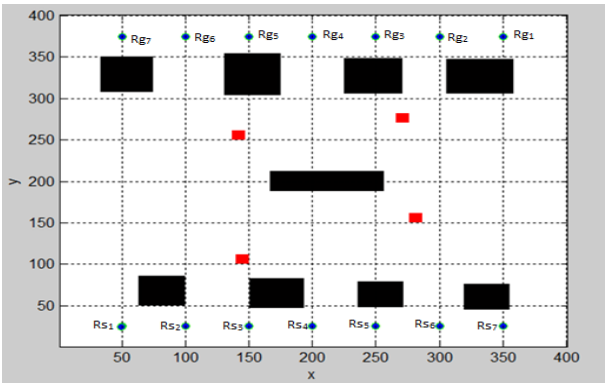

Fig.5. Configuration Space

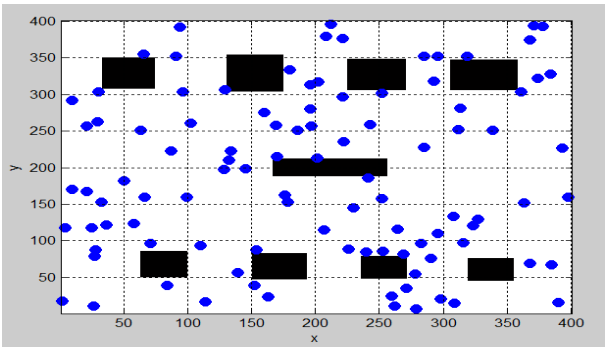

Fig.6. Nodes generated in the free space

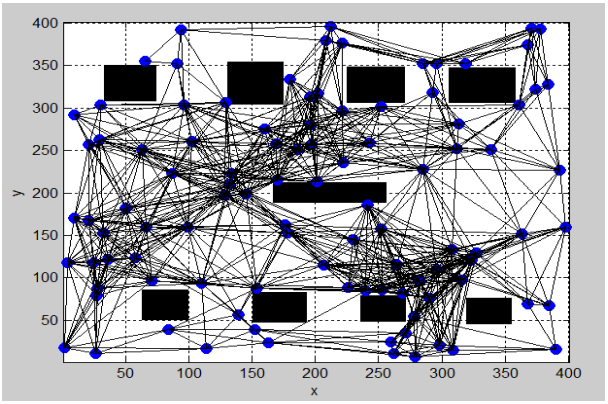

Fig.7. Probabilistic Roadmap

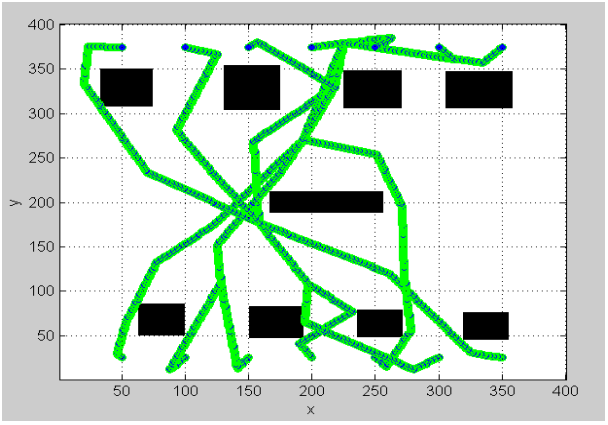

Fig.8. Robots moves using PRM

Table (1) show the time needed to complete the preprocessing phase in the A* and the depth first search and in many different number of robots in the configuration space starting with two robots until reach forty robots. In the table, the mean and the standard deviation are listed of ten runs done to each case to make ensure of the results we obtained. For each run, the vertices of the graph were randomly produced – therefore different mapping obtained. The change in time can be seen for the A* from the case of two robots which is 1.49 seconds to the case of forty robots which is 3.43, approximately duplicated. In DFS it takes 5.79 seconds in the case of two robots and 80.6 in the case of forty robots, approximately duplicated eight times.

From these numbers, the time is changed slightly in A* than the depth first as the number of robots increased. Therefore, the depth first search takes more time in the preprocessing phase than the A*. The standard deviation σ listed on the table to show how much variation or dispersion from the mean exists in the ten runs, because in each run we have a different graph generated randomly. A low standard deviation indicates that the data points tend to be very close to the mean; a high standard deviation indicates that the data points are spread out over a large range of values.

In table (2), if the case of two robots in A* taken, it spent 7.7 seconds to moves the robots from its starting points to its goals with a pulse time 0.005 second between each move. And 8.4 seconds in the case of forty robots. While in DFS it spent 8.3 in the case of two robots and 9.5 seconds in the case of forty robots. Approximately, the change from the first case to the last case is the same, but the A* takes less time. This is because the paths gathered in the preprocessing phase in A* is more optimal (shortest) than the DFS.

The data gathered in the two phases is plotted in figures 8 and 9 to make it much clearer.

Table 1. Preprocessing Phase

No. of robots	A*		Depth First	
	μ	σ	μ	σ
2	1.4918659	0.04103977	5.790242	0.021612
4	1.57886	0.06942869	10.73457	0.307229
6	1.6976037	0.10587086	13.68353	1.234082
8	1.7626784	0.02923179	18.487	0.100477
10	1.9813042	0.05396073	23.31528	0.460697
12	2.3144519	0.05519958	24.78951	1.769026
14	2.4573228	0.02399001	28.51377	2.566433
16	2.5026629	0.23287855	32.9775	3.771837
18	2.6941571	0.15438404	36.31962	2.090441
20	2.7465716	0.4901247	40.3256	2.434299
22	2.8115535	0.28449169	43.93239	1.783724
24	2.8668479	0.01619982	50.01216	9.181094
26	2.8808306	0.16248182	53.01348	2.109307
28	2.997968	0.12694829	58.60808	10.37246
30	3.0803505	0.12135649	60.22731	1.558702
32	3.113253	0.22348322	61.70389	8.341048
34	3.2962763	0.15827313	64.05341	5.408895
36	3.3495873	0.24057187	65.93867	7.766211
38	3.4026438	0.04367374	76.18719	4.450561
40	3.4377526	0.0997657	80.6765	1.100211

Table 2. Moving Phase With a pulse time 0.005

No. of robots	A*		Depth First	
	μ	σ	μ	σ
2	7.7299584	0.50377116	8.385449	0.298326
4	7.8905785	0.19978029	8.500048	0.08062
6	7.9514495	0.27516283	8.715455	1.045796
8	7.9989442	0.20033608	8.826854	0.001166
10	8.0400533	0.06645531	8.852039	0.538816
12	8.0947616	0.33460929	8.904574	0.351353
14	8.0961602	0.61022467	8.915853	0.306134
16	8.1105329	0.08459967	8.982046	0.380711
18	8.1106007	0.24875451	9.07284	0.778784
20	8.1405839	0.52406441	9.089187	0.815411
22	8.1524353	0.26768376	9.103185	0.463318
24	8.2095119	0.20291348	9.169197	0.130077
26	8.2222411	0.96684266	9.169517	0.457939
28	8.2591723	0.58365725	9.187313	0.339228
30	8.2600198	0.29234411	9.287045	0.509409
32	8.2646549	0.21895915	9.336926	0.757852
34	8.2852437	0.03498764	9.341339	0.762055
36	8.3168383	0.53355591	9.358175	0.434831
38	8.3384204	0.10590763	9.526453	0.041421
40	8.4307449	0.5139839	9.56306	0.076024

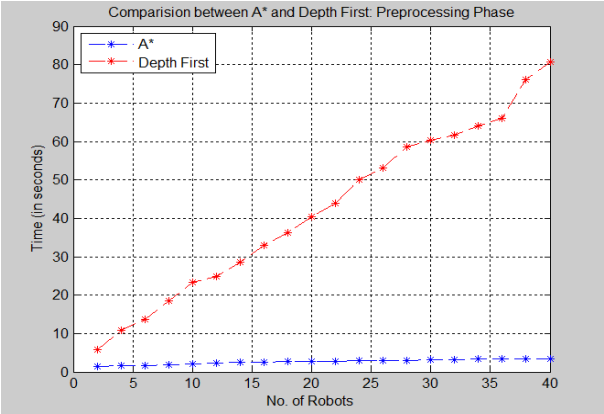

Fig.9. Plotting the times of the A* and DFS search algorithms in Preprocessing Phase

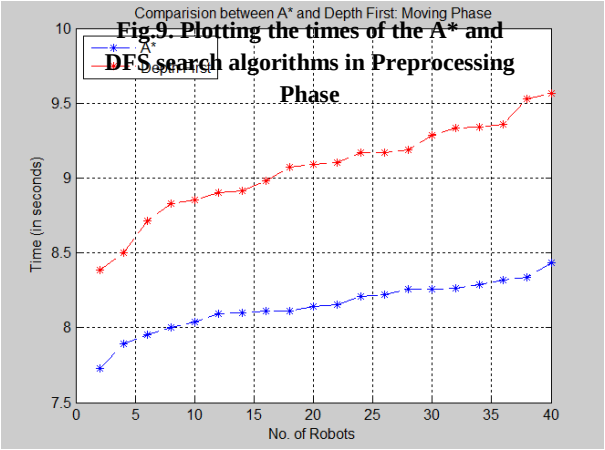

Fig.10. Plotting the times of the A* and DFS search algorithms in Moving Phase

5. CONCLUSION

In this paper, the problem of multi robot path planning by using A* search algorithm in a dynamic environment have been presented. The problem is divided into two phases: preprocessing phase and moving phase. Most of the work is done in the first phase to reduce the computation time needed in the second phase. The A* is a heuristic search which use heuristic functions when it decide which path must be taken. In some applications, the computation of these functions need more time which affect negatively in execution time. On the other side, the Depth First is a blind search and it doesn't

spend any time in the process of path selection because it chooses blindly the first path founded. The main reasons behind choosing depth first search against A* is to see the impact of computing the heuristic function in the speed and if there is optimality in choosing A* rather than depth first.

From the obtained results in table (1) which represents the preprocessing phase, the time needed by A* is less than the time needed by DFS. And the increasing in time from case to case is also less than the DFS. From this, it can be concluded that A* is better than DFS because as the number of robots increased to very large numbers in a planner using DFS, the time needed to execution also become very large. In the second phase and from table (2) we noticed the time of execution of the A* is also less than the DFS, this means that the paths chosen in the first phase are more optimal than those chooses by Depth First which led to less execution time.

6. REFERENCES

[1] P. Svestka and M. H. Overmars, "Coordinated path planning for multiple robots," Robotics and Autonomous Systems, Elsevier, pp. 125-152, 1998.

[2] S. Carpin and E. Pagello, "An experimental study of distributed robot coordination," Robotics and Autonomous Systems, Elsevier, pp. 129-133, 2009.

[3] J. P. van den Berg and M. H. Overmars, "Roadmap-Based Motion Planning in Dynamic Environments," IEEE TRANSACTIONS ON ROBOTICS, vol. 21, no. 5, pp. 885-897, 2005.

[4] J. C. Latombe, Robot Motion Planning, Kluwer Academic Publishers, 1991.

[5] J. E. Hopcroft, J. T. Schwartz and M. Sharir, "On the complexity of motion planning for multiple independent objects; PSPACE hardness of the "warehouseman's problem"," The International Journal of Robotics Research, vol. 3, no. 4, pp. 76-88, 1984.

[6] S. M. LaValle, PLANNING ALGORITHMS, Cambridge University Press, 2006.

[7] P. Velagapudi, K. Sycara and P. Scerri, "Decentralized prioritized planning in large multirobot teams," Taipei, 2010.

[8] K. Fujimura, "Time-Minimum Routes in Time-Dependent Networks," IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION, vol. 11, no. 3, pp. 341-351, 1995.

[9] J. T. Schwartz and M. Sharir, "On the piano movers' problem: III. Coordinating the motion of several independent bodies: the special case of circular bodies moving amidst polygonal barriers," The International Journal of Robotics Research, vol. 2, no. 3, pp. 46-75, 1983.

[10] D. Parsons and J. Canny, "A Motion Planner for Multiple Mobile Robots," Cincinnati, OH, 1990.

[11] J. Barraquand and J.-C. Latombe, "Robot Motion Planning: A Distributed Representation Approach," The International Journal of Robotics Research, vol. 10, no. 6, pp. 628-649, 1991.

[12] R. Luna and K. E. Bekris, "Efficient and Complete Centralized Multi-Robot Path Planning," IEEE/RSJ International Conference on Intelligent Robots and Systems, 2011.

[13] S. J. Buckley, "Fast Motion Planning for Multiple Moving Robots," Scottsdale, AZ, 1989.

[14] J. P. van den Berg and M. H. Overmars, "Prioritized motion planning for multiple robots," 2005.

[15] M. Bennewitz, W. Burgard and S. Thrun, "Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots path planning techniques for teams of mobile robots," Robotics and Autonomous Systems, vol. 41, no. 2-3, pp. 89-99, 2002.

[16] P. Velagapudi, K. Sycara and P. Scerri, "Decentralized prioritized planning in large multirobot teams," Taipei, 2010.

[17] M. Erdmann and T. Lozano-Perez, "On multiple moving objects," 1986.

[18] S. S. Chiddarwar and N. R. Babu, "Conflict free coordinated path planning for multiple robots using a dynamic path modification sequence," Robotics and Autonomous Systems, Elsevier, pp. 508-518, 2011.

[19] T. Fraichard and I. Rhone-Alpes, "Trajectory Planning in a Dynamic Workspace: a 'State-Time Space' Approach," Advance Robotics, vol. 1, no. 13, pp. 75-94, 1999.

[20] L. E. Kavraki, P. Svestka and J.-C. Latombe, "Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces," IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION, vol. 12, no. 4, pp. 566-580, 1996.

[21] M. Ryan, "Multi-Robot Path-Planning with Subgraphs," Australia, 2006.

[22] M. Ryan, "Graph Decomposition for Efficient Multi-robot Path Planning," San Francisco, CA, USA, 2007.

[23] M. Ryan, "Constraint-based multi-robot path planning," IEEE International Conference on Robotics and Automation, 2010.

[24] A. Stentz, "Optimal and Efficient Path Planning for Partially-Known Environments," IEEE International Conference on Robotics and Automation, 1994.

[25] A. Stentz, "The Focussed D* Algorithm for Real-Time Replanning," International Joint Conference on Artificial Intelligence, 1995.