Initial and Target Nodes Planning for Multi Mobile Robots

Farah S. Khoshaba
Control and Systems Engineering Department
, University of Technology; Baghdad; Iraq.
farah_sami76@yahoo.com

Abstract:

A novel algorithm for planning initial and target nodes for multi mobile robots in complex environment where all robots have the same functionality is presented. The algorithm assigns an initial node to each target node based on a ranking method. The ranking method takes into consideration the path length between the two nodes, the effort of the robot while moving between the two nodes, and the possibility to block other robots movements while moving between the two nodes. The algorithm also tries to solve collision between robots using the proposed ranking method and taking robots performance into consideration.

التخطيط للوصول من عقدة البداية الى عقدة الهدف لعدة روبوتات متحركة.

الخلاصة:

خوارزمية جديدة لتخطيط عقدة البداية وعقدة الهدف لعدة روبوتات متحركة في بيئة معقدة حيث أن جميع الروبوتات تقدم الوظيفة نفسها. الخوارزمية تُعيّن عقدة اولية لكل هدف معتمدة على طريقة التصنيف. طريقة التصنيف تأخذ بنظر الاعتبار طول المسار بين العقدتين ، الجهد الذي يقوم به الروبوت عند الانتقال بين العقدتين ، وكذلك إمكانيته لقطع الطريق على الروبوتات الاخرى اثناء حركتها بين تلك العقدتين. الخوارزمية أيضا تحاول حل التصادم بين الروبوتات باستخدام طريقة التصنيف المقترحة مع الأخذ بعين الاعتبار اداء الروبوتات.

1. Introduction

Robotics systems are widely used in modern manufacturing and industry to improve efficiency, accuracy and working environments. In the military domain robotic systems are also being adopted to perform dangerous and hazardous missions. With this increasing utilization, a single robot can't meet all aspects of our needs because of its inherent limitations. As a solution, multi robot systems are being investigated for many applications

that are difficult or time consuming for a single robot. In many situations multi robots systems can be used to improve sensing performance, robustness and reliability [1].

In many multi robot applications, there exists a need to coordinate the actions of robots to achieve a goal. When many robots operate in the same environment, a motion planning is required for the robots to reach their goals while avoiding collisions among themselves [2]. In the recent years, the problem of path planning for multi robots is receiving attention increasingly and many people have researched in this field. As an example, when [3] plans the path of each robot, the graph model of environment is dynamically changed for path correction and collision avoidance. Their algorithm avoids collusion through changing robots' paths and speeds. [4] Decomposes the path-planning problem into two modules: path plan and velocity plan. Optimization is achieved at the individual robot level by defining cost functions to minimize, and also at the team level by a global measurement function rectifying performance indices of interest as a team. [5] is based on searching through various roadmaps and defining an Elementary Kinematic Graphs that represent the link between roadmaps. Then the possible plans are defined based on a sub set of configurations and a sub set of kinematic nodes. [6] Generates a search space that relies on non-uniform density sampling of the free areas to direct the computational resources to troubled and difficult regions, such as narrow passages, leaving the larger open spaces sparsely populated. [7] Has outlined a method for reducing the search-space in multi-robot path planning problems by decomposing the map into connected sub-graphs of two types: Stacks that represents a long narrow dead-end road or corridor in the map, and Cliques that represents a large open area in the map with many exit nodes around its perimeter. Then they plan for robots' path from one sub-graph to another using special-purpose code to determine when transitions between sub-graphs are possible. All the above works can fall into one of two categories: centralized planning and decoupled planning. Centralized path planning methods, which treat the robots as a single composite entity, scale poorly as the number of robots increases. Decoupled path planning methods, which first plan for each robot independently then resolve conflicts afterwards, prove to be much faster but are incomplete because many problems require robots to deliberately detour from their optimal path in order to let another robot pass. Even if a priority ordering is used, requiring low priority robots to plan to avoid high priority robots, still problems can be found which cannot be solved with any static ordering. It seems that centralized planning is unavoidable if we want a complete planning algorithm. This does mean, however, that a naive search of the composite configuration space is the only solution that remains [4].

In this paper, a centralized path planning is present. While all papers in planning multi mobile movements defines initial and target nodes of each robot in the beginning of their work, in this paper we will assume that we have a set of initial positions and a set of target position for all robots. The work then will assign a target node to each start node in condition that one or more paths do exist between them. We will assume that all robots functionality is

the same. i.e. any robot from the group of robots being considered can do the required job in the target node. This is important assumption because if each robot does a unique job at the target node, the start node of this robot cannot be assigned to any target node other than the one that the robot should work in. In simple environments graph we can assign a target node to a start node by hand, but in complex environment graphs there should be an automated way to do the assignment because the start node that we may assign to the target node may not be the optimum node to be assigned to that target node. We define the optimum start node to be: The node that have the shortest path with the target node, the node that requires low robots effort, and the node that have a path that will not block other robots movements. After assigning all initial nodes to all target nodes, an optimization to the chosen paths will be done to solve conflict occurred between robots.

2. Optimization

In our work we use two optimization criteria, Node Optimization that is used to find the optimum initial node for each target node, and Path Optimization that is used to solve collisions that occurs in the selected paths.

In node optimization, an initial node is considered to be the optimum node to be assigned to a target node if and only if this node satisfy the following conditions:

1. They exist a path that could be considered as the shortest path between the initial node and the target node.
2. The path that exists between the two nodes should require the lowest robot efforts. We define robot effort to be the number of turns (turn left, turn right, or rotate) that each robot performs while moving in a path. A path that have many turns will slow the robot movement because the robot will need more time to perform the turning action.
3. The path between the two nodes does not contain a node from initial nodes set or from target nodes set. If a robot has to pass through a start node or a target node during its movement, then this robot will be blocked by the other robot that is occupying this node. When other robots reaches their targets and starts to perform their job there, the robot being considered has to wait a long time before being able to pass the already occupied node. The same is true when other robots are in their initial node charging themselves. Moreover passing through other starts and targets nodes that are unoccupied will block the other robots that needs to reach these nodes.

Based on our optimization criteria, all initial nodes will be ranked related to all target nodes. Then we will start assigning initial nodes to target nodes. The over all picture is considered during the assignment process, as an example if two target nodes share the same optimum initial node, the assignment will depend on which target has a better second optimum node, i.e. which target can be assigned to another initial node without slowing down the over all system performance.

After assigning all initial nodes to all target nodes and defining the path that will be used by the robot to pass between them, we will check collusions that may occurs between robots paths. If collisions occur, path optimization will be applied to solve them and as follows:

- If there exists other paths that connect the initial node to the target node and the rank of each path of them is almost equal the rank of the already selected path (we will see later the equation that defines this relationship), we will try to use the other paths instead of the initially selected path and check collision, and if using any other path solve collision we will use this path.
- For all pairs of collision robots, if the two initial nodes have almost equals ranks with the two target nodes, we will swap between the two nodes, i.e. we will assign the first start node to the second target node and vise versa.

3. Nodes Planning

The proposed algorithm consists of two parts; the first part assigns an initial node to each target node and defines the path that will be used between them, while the second tries to solve conflicts that occur between robots. Both parts can be summarized in (Flowchart-1). In this section, the stages that are used for planning initial nodes for all target nodes are being discussed.

3-1 Initialization

The algorithm starts by getting graph information including:

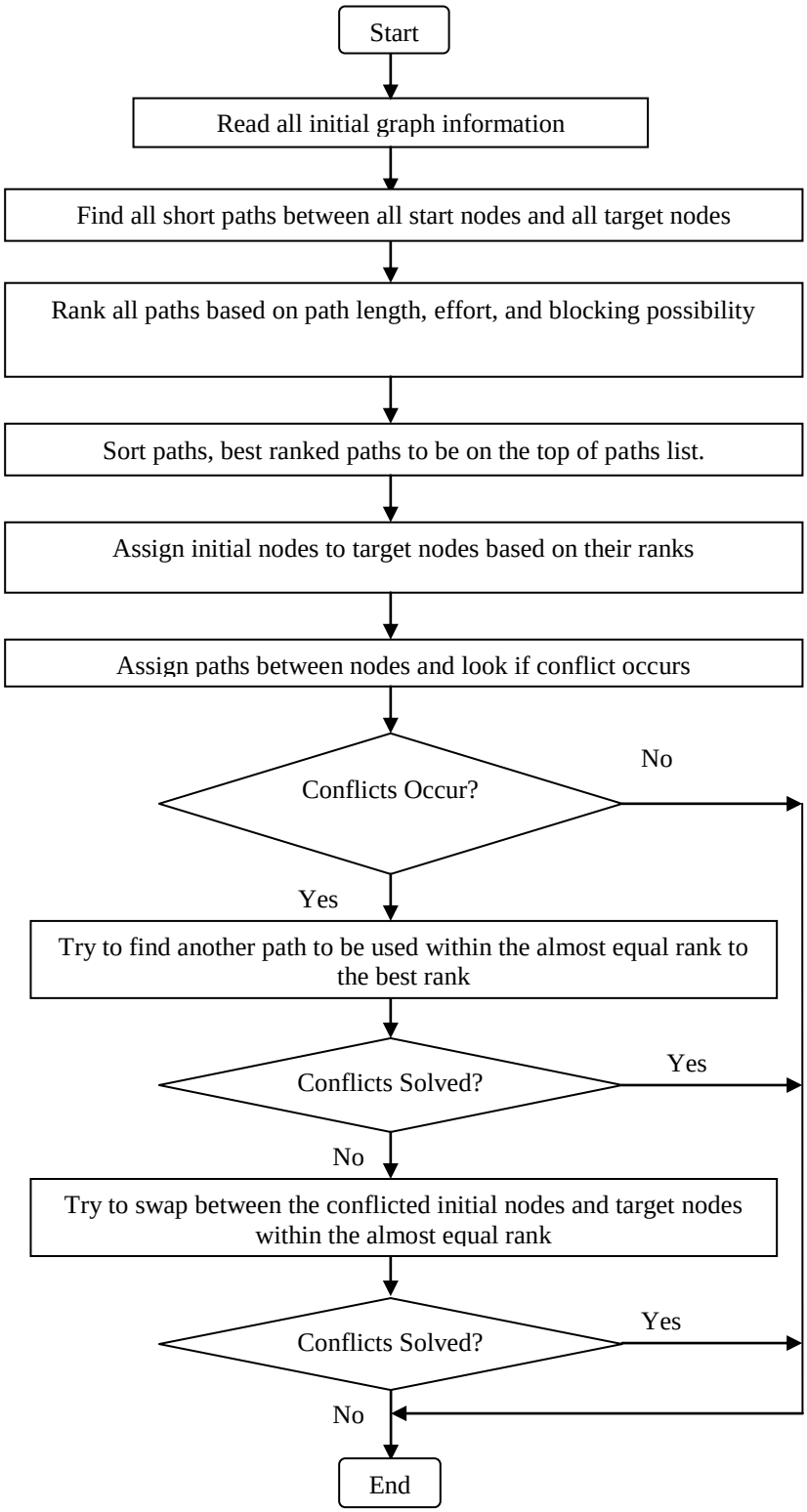
1. Number of nodes in the graph.
2. Initial nodes positions in the graph.
3. Target nodes positions in the graph.
4. Obstacles positions in the graph.
5. Distances between any two adjacent nodes.

Note that the number of initial nodes should be equal or greater than number of target nodes, or else there will be some target that will not be reached by any robot.

The algorithm then establishes links between all graph nodes. Each node in the graph has a link with its adjacent nodes. The algorithm defines a cost value for each link between two adjacent nodes to be equal to the distance between them. If the node contains obstacles, no links should be set with this node, so the algorithm set the cost value for this link to infinite.

3-2 Finding Paths

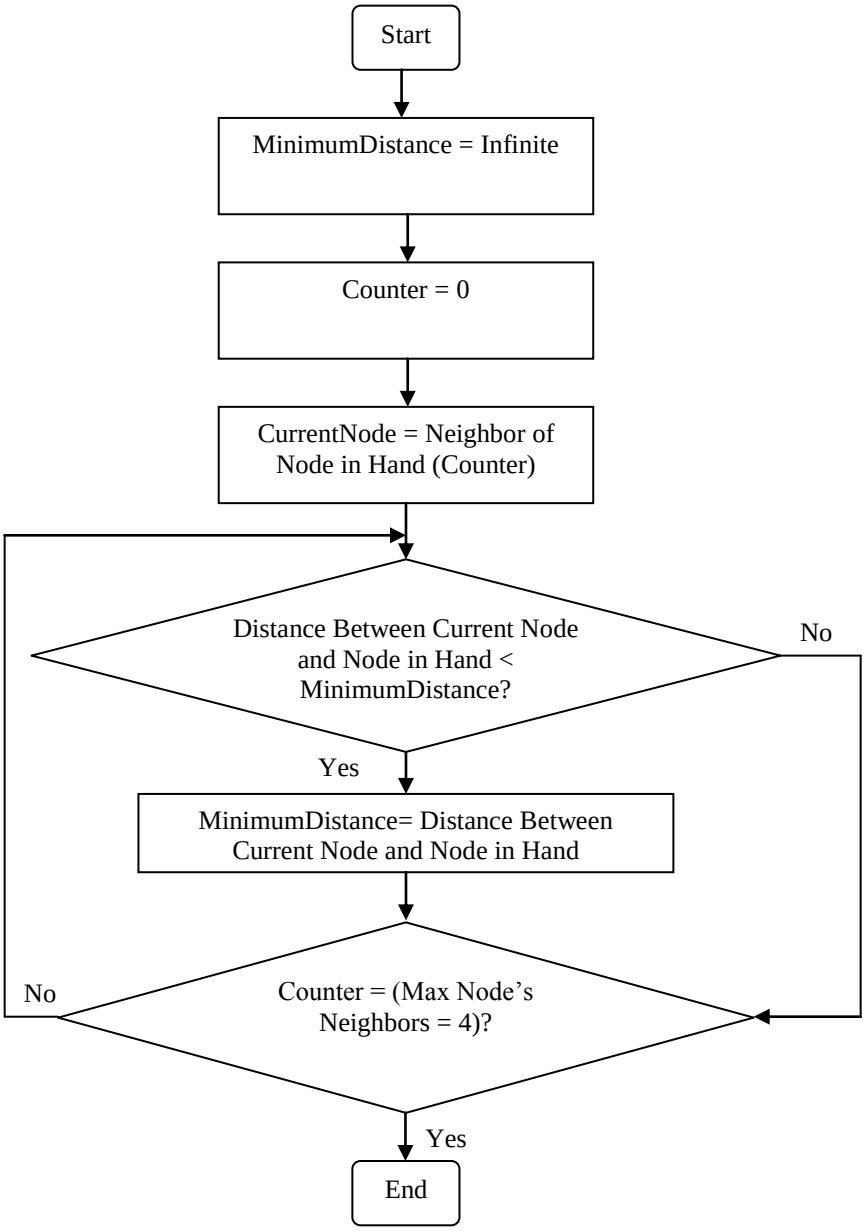
The algorithm then uses Dijkstra method combined with Depth first search to find all short paths between the initial node and the target node.



(Flowchart-1 The Proposed Algorithm)

3-3 Ranking

After that the algorithm starts to examine each path and rank it as shown in Flowchart-2.



(Flowchart-2 Minimum Distance Calculation)

3-4 Sorting

The algorithm then use bubble sort algorithm to order all paths between any initial node and target node based on their rank. The path that has the minimum rank will be at the beginning of the array. In other words, the stack that has the best rank will be at the beginning of the array while the stack that has the worst rank will be at the end of the array. This step is done in order to simplify the remaining steps of the algorithm, i.e. the remaining steps of the

algorithm will not need to search for the best rank within the available paths, instead it will find it at the top of the list. And as we will see later, when the algorithm needs to use the next best path between two nodes (initial and target) it will find the next best right below the used one.

3-5 Assigning Nodes

The algorithm now starts assigning an initial node to each target node as follows:

1. For the first target node, find the initial node that has the minimum rank with it. Assign path between them based on the founded rank.
2. For the second target, repeat step one.
3. If the initial node founded in step two is already assigned to another target do the following:
    a. Find another initial node that has a rank equals to the minimum rank. If the node founded and it is not used by another target assign it to the target node and go to step 4.
    b. Else if the node founded is used by another target, repeat step a.
    c. If there is no other node that has a rank equals to the best rank, or steps a and b failed to find a free node, repeat steps a and b for the target node that uses the first founded initial node.
    d. If step c also failed to find a free node, find the second best path (the second minimum rank) for the two conflicted target nodes. Use the second min path for one of them depending on which one has a better best rank, and which one has a second rank that is not used by any other target node.
    e. If step d failed also to find initial nodes for both targets, assign the initial node to one target. For the other target that has he better next rank repeat from step a till finding a solution using the second best initial node (founded in step a or d depending on which one found a node that is not free) and the target that uses this initial node.
4. Repeat all the above steps till you assign initial nodes to all target nodes.

Note that the above steps might failed to find an initial node if the graph contains a target node that cannot be reached by any initial node.

4. Solving Conflicts

The below stages are used to solve collisions that occurs between robots paths.

4-1 Solving Conflicts, Level One

The algorithm so far has assigned an initial node to each target node and a path between them based on the best founded rank. At this stage the algorithm will start to find collisions that might occurs between robots using the path already selected. If collision(s) occurs, the algorithm will try to solve collision(s) using the following steps:

1. Find another path within the same used rank between the two group of nodes (initial-target and initial-target that the collision(s) occur between them) that makes number of collisions = 0 between the two groups and does not increase the over all system collisions.
2. If no such path founded, find another path within other ranks between the two group of nodes that makes number of collisions = 0 and does not increase the over all system collisions. Stop the search on rank = used rank + no of collisions between the two groups of nodes. The equation above was selected based on a fact that using any available method for collision solving will either slow down one robot or make one robot wait till the other robot pass the conflict node, and that will cause a delay equals to $\sum$ distances of the collision nodes. So comparing the performance of using these methods with using a bath that is ranked as specified in our equation will leads that our equation has better performance as we delay robots by number of collisions not summation of distances of nodes where collision occur.
3. If no path founded using the above steps, use the path that minimize the over all system conflicts instead, i.e. the path that will minimize collision between the two group of nodes and also minimize (or at least does not increase) collisions for the other robots.
4. Update system collisions and repeat from step 1 till all conflicted groups are considered.

4.2 Solving Conflicts, Level Two

If stage one for solving collisions failed to solve all system collisions, the algorithm will try to solve collisions as follows:

1. Swap between the two conflicted initial nodes, i.e. use the first initial node to the second target node and vise versa. Use the path that has the best rank between the two nodes, i.e. use the minimum ranked path that is at the top of the array of all possible paths.
2. If step one makes collision between the two conflicted groups = 0 without increasing collisions for other system nodes, keep the swapped nodes.
3. Else use paths with other ranks till rank = initially used rank (not the best rank) + no of collisions. If a path found that makes number of collisions between the two groups = 0

without increasing the over all system collisions, keep the swapped nodes and use the founded path.

4. If no solution found that makes collision = 0, try to find a solution that will minimize over all system collisions. If founded, use it.
5. If no solution founded using steps 1 to 4, then there is no such a path between the two conflicted robots that might solve collision without reducing robots performance. Return the swapped nodes, i.e. return the first initial node to the first target node and vise versa. Use the path founded in stage one. Repeat step 1 to 5 for all robots that have collision in between.

At the end of this stage we will have two possibilities for all robots: either over all system collisions = 0, and the algorithm goal is established, or system collisions > 0, and only the first goal of the algorithm is established: assigning initial nodes to target nodes based on the best rank between them. If the second possibility occurs, that means that there is no path that could solve conflict between robots without slowing down system performance. In this case we can use any other available method to solve the conflicts or simply we can use any available path that can solve conflicts regardless its rank (if this path is already exists as the system graph may not contain enough free nodes for such path to exists).

5. Experiments

We tested our system in variant environment graphs, and in all tests we did the algorithm succeeded to assign initial nodes to target nodes and to solve conflicts that occur between robots. Below we will discuss two examples; the first one is simple environment example that we will use to explain how the algorithm works, while in the second one we will prove the success of our algorithm to achieve its goals in complex environment graphs.

5.1 Simple Environment Graph

5-1-1 Initial Graph

As could be seen from (Fig-1) we have a 5×4 nodes graph, for simplicity we assume that the distance between each node and the other is equal to 1 m. we have three robots positioned in three initial positions (node number 2, 3, and 4 respectively), and three target nodes to be reached by each robot (node number 5, 16, and 18 respectively).

S	S	S	1	0
9	8	7	6	T
14	13	12	11	10
19	T	17	T	15

(Fig-1 Experiment 1 Environment Graph)

5-1-2 Assigning Nodes

The algorithm will rank all nodes in the graph; the best rank between each initial node and target node is shown in (Table-1). The algorithm will assign start node 2 to target node 5 as it has the best rank among other initial node, which is equal to 3. 3 was calculated based on that there is 2 meter apart between node 2 and 5, and the minimum robot turns equals to one turn for this path so the rank will be = 2 m + 1 effort = 3. The algorithm will assign node 3 to node 18 in the same way explained above. For nodes 4 and 16, the algorithm will check first node 2 that has a minimum rank with node 16, the node is already assigned to node 5 and the rank between node 2 and 5 is better than the rank between 2 and 16, so the algorithm check to see if there is another node to be assigned to nodes 16, the other node will be node 3. Node 3 is already used with node 16 and the rank between node 3 and 18 is better than the rank between node 3 and 16, so the algorithm assign the final node 4 to node 16. The final assignment is shown in (Fig-2)

(Table-1 The Best Ranks Between Initial and Target Nodes)

Start Node	Target Node	Best Rank
2	5	3
2	16	4
2	18	5
3	5	4
3	16	6
3	18	3
4	5	5
4	16	7
4	18	4

S3	S2	S1	1	0
9	8	7	6	T1
14	13	12	11	10
19	T2	17	T3	15

(Fig-2 Final Nodes Assignments)

5-1-3 Assigning Paths

(Fig-3) shows the paths that will be used between the assigned nodes based on the best rank between them.

S3	S2	S1	1	0
9	8	7	6	T1
14	13	12	11	10
19	T2	17	T3	15

(Fig-4 Initial Paths)

As could be seen from the figure, collusion will occur in nodes 8, 7, and 6.

5.1.4 Solving Conflicts - Level One

The algorithm first check to see if there is another path that could be used between the assigned nodes within the same rank that could solve the conflicts occurred. For nodes 2 and 5, there is another path that has a rank equals to 3 which pass through nodes 1 and 0, so the algorithm solve the conflict for the robot positioned in node 2 and headed towards node 5 as shown in (Fig-5).

S3	S2	S1	1	0
9	8	7	6	T1
14	13	12	11	10
19	T2	17	T3	15

(Fig-5 Solving Robot1 Conflicts)

Now we only have one conflict that occurs in node 8. As could be seen from (Table-2), there is only one path that connects nodes 3 and 8 and is the shortest path found using Dijkstra combined with depth first search method.

All paths between node 3 and node 18				Paths Ranks
3	8	13	18	3

(Table-2 All Paths between Node 3 and Node 18)

As could be seen from (Table-3), top ranked paths between node 4 and 16 will cause conflict to occur in the system either in node 8 or in node 13 or in both nodes (note there 21 shortest paths between 4 and 16, but for simplicity only top ten ranks paths are shown in the table, the low ranked paths cause collusion to occur in initial nodes 2 and 3 or in target node 18).

Top ten ranks paths between node 4 and node 16							Paths Ranks
4	9	8	7	6	11	16	7
4	9	14	13	12	11	16	7
4	9	8	7	12	17	16	8
4	9	14	13	12	17	16	8
4	9	8	7	12	11	16	9
4	9	8	13	12	11	16	9
4	9	8	13	12	17	16	10
4	9	14	19	18	17	16	26
4	9	8	13	18	17	16	28
4	9	14	13	18	17	16	28

(Table-3 Top Ten Ranks Paths between Node 4 and Node 16)

So as there is no other path that could be used between the assigned node neither a path that has a best rank nor a path that has a second best rank based on the equation explained before, so the algorithm goes to level two of conflicts solving.

5-1-5 Solving Conflicts - Level Two

The algorithm will try to solve conflicts between the two conflicted robots by swapping their target nodes.

Because the algorithm found that:

(Using node 3 to node 18 with rank = 3) + (using node 4 to node 16 with rank = 7) has the same rank summation result for (using node 3 to node 16 with rank = 6) + (using node 4 to node 18 with rank = 4)

So this means both assignments has the same impact on the over all system rank (and performance), so the algorithm decide to swap between the two target nodes. (Table-4) shows all paths between node 3 and node 16, and (table-5) shows all paths between node 4 and node 18. As could be seen from both tables, using the best ranked path for both robots will solve all system conflicts, so the algorithm keep the swapped nodes, and use the best ranked paths, the final result is shown in (fig-6)

(Table-4 All paths between node 3 and node 16)

All paths between node 3 and node 16						Paths Ranks
3	8	7	6	11	16	6
3	8	13	12	11	16	6
3	8	7	12	17	16	7
3	8	13	12	17	16	7
3	8	7	12	11	16	8
3	2	1	6	11	16	25
3	8	13	18	17	16	25
3	2	7	12	17	16	26
3	2	7	6	11	16	27
3	2	7	12	11	16	27

(Table-5 All paths between node 4 and node 18)

All paths between node 4 and node 18					Paths Ranks
4	9	14	19	18	4
4	9	8	13	18	5
4	9	14	13	18	5
4	3	8	13	18	24

S3	S2	S1		
				T1
	T3		T2	

(Fig-6 Final System Plan)

5-2 Complex Environment Graph

Now we will see the result of using our algorithm in a complex environment graph. This example will prove the efficient of using our algorithm both to assign initial nodes to target nodes and to solve conflicts that may occur after the assignment.

5-2-1 Initial Graph

As could be seen from (fig-7) we have a 7×9 nodes graph with varied distances between the graph nodes. We have 6 robots positioned in six initial positions (node number 1, 2, 3, 34, 41 and 48 respectively), and six target nodes to be reached by each robot (node number 7, 20, 21, 35, 56, and 60 respectively). We also have some obstacles in eight nodes that prevent robot from passing these nodes (node number 6, 11, 14, 32, 39, 50, 51 and 62 respectively)

X	5	4	S	S	S	0
13	12	X	10	9	8	T
T	19	18	17	16	15	X
27	26	25	24	23	22	T
S	33	X	31	30	29	28
S	40	X	38	37	36	T
S	47	46	45	44	43	42
55	54	53	52	X	X	49
X	61	T	59	58	57	T

(Fig-7 Complex Environment Graph)

5-2-2 Assigning Nodes and Assigning Paths

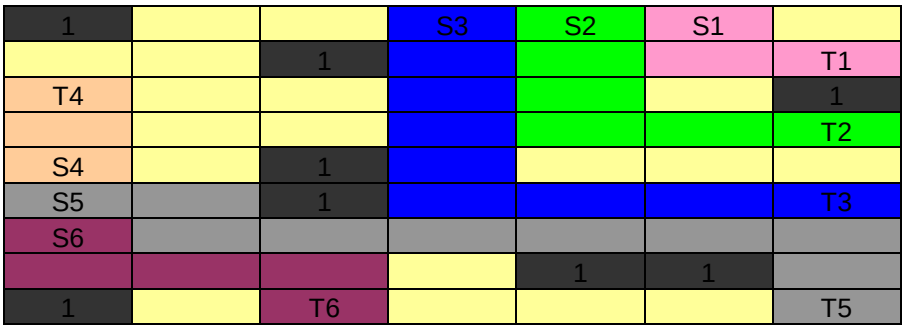
(Fig-8) shows the final result of assigning each initial node to a target node. The assignment was done as we described before. The figure shows also the paths that are chosen by the algorithm to be used by each robot. As we can see from the figure, there are two collusions that occur in nodes 53 and 54.

			S3	S2	S1	
						T1
T4						
						T2
S4						
S5						T3
S6						
		T6				T5

(Fig-8 Assigning Nodes and paths Result)

5-2-3 Solving Conflicts - Level One

Even though the graph here has very complicated environment, the algorithm simply could find another path for robot number 6 to pass through within the second best rank available (The distance of the best ranked path selected above is equal to the distance of the second ranked path because of the height/width nature of the two paths, but the second path need one more robot turns than the first path). (Fig-9) shows the final system plan for the complex environment example being considered.



(Fig-9 Final System Plan)

6. Conclusion

We discussed in this paper our novel algorithm that could be considered the first algorithm that is used to assign initial nodes to target nodes for a group of robots that all robots performs the same task. Our algorithm proves its efficiency in assigning target nodes to initial nodes in complex environments graphs where hand assignment may cause to assign an initial node to a target node without considering if these two nodes have a minimum distance between them among all other available nodes. Our algorithm also find all possible minimum paths between the initial node and the target node, and the algorithm assigns a path to a robot only if the path requires minimum robot effort to pass through, and only if the path cause minimum collusions to occur if it is used. Our algorithm also tries to solve collusions between robots in a manner that does not drop down over all robots teamwork efficiency.

For future work, we can add a solving algorithm that solve conflicts 100% with maintaining reasonable over all robots team work efficiency.

References

1. *Lei Liu, Yongji Wang, Shuanghua Yang, Graham Watson, and Brian Ford,* "**Experimental Studies of Multi-robot Formation and Transforming**", Proceedings of the UKACC International Conference on Control, 2008.

2. *Osman PARLAKTUNA, Aydın SİPAHİOĞLU, Ahmet YAZICI,* "**A VRP-Based Route Planning for a Mobile Robot Group**", Turkish Journal of Electrical Engineering and Computer Sciences, Vol. 15, Issue 2, 2007.

3. *Fedor A. Kolushev and Alexander A. Bogdanov,* "**Multi-agent Optimal Path Planning for Mobile Robots in Environment with Obstacles**", Proceedings of Third International Andrei Ershov Memorial Conference on Perspectives of System Informatics, Vol. 1755, Springer-Verlag, 1999.

4. *Yi Guo and Lynne E. Parker,* "**A Distributed and Optimal Motion Planning Approach for Multiple Mobile Robots**", Proceedings of IEEE International Conference in Robotics and Automation (ICRA), May 2002.

5. *F. Gravot, and R. Alami ,* "**Preliminary Results on Planning Multi-robot Cooperative Manipulation Tasks**", IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Oct. 2002.

6. *Tarek Taha, Jaime Valls Miró, and Gamini Dissanayake,* "**Sampling Based Time Efficient Path Planning Algorithm for Mobile Platforms**", Proceeding of the 2006 IEE International Conference on Man-Machine Systems (ICoMMS), Sep. 2006.

7. *Malcolm Ryan,* "**Multi-Robot Path-Planning with Subgraphs**", Australasian Conference on Robotics and Automation, Dec. 2006.