Optimum Path Planning for Multi Mobile Robots with Speed of Work Achievement as Constrain

Bashra Kadhim Oleiwi

Control and System Engineering Department, University of Technology,Baghdad, Iraq.

bushrakad@yahoo.com

Submission date:- 12/1/2011	Acceptance date:- 17/1/2011	Publication date:- 2/12/2018

Abstract

This paper presents optimum path planning for multi mobile robots that move between an initial point toward a target point then back to the initial point in a way that avoid collision without heavily slow down robots speed. The work system designs for taking into consideration robots velocities as well as time that robots will spend it in their target points, since both could be variant for all robots. In order to achieve the work objectives, time and space method combined with sequential entry method are used to design robots motion. In addition, two priority levels are used of carefully selected priorities to avoid collision between robots; node level priority and robot level priority. In node level priority, the priority gives to robots in a manner that minimize the number of over all system collisions. While in robot level priority the priority gives to robots that cause faster speed for work achievement. The work is tested with different number of robots and different types of maps, and the algorithm proved its efficiency in finding the optimum solution regarding system performance for each robot in the collision points. The proposed algorithm is implemented using VBA programming language.

Keywords: - Robut, Optimum poth, Planning, Multimobile robot.

Introduction

Many missions in autonomous mobile-robot systems depend on navigating in a known or an unknown environment and performing some tasks, like landmine exploration, post-office automation, cleaning, transporting load from one node to another, or assisting rescue after disasters. A multi-robot cooperative system appears to be more effective and adaptive to accomplish various such kinds of complex tasks, relative to a single robot approach, and most of these missions may be performed more efficiently by a collaboratively working multi-robot team [1].

Problems of multi-agent robot systems control have got significant importance. Each multi-agent robot system has some transport subsystem, which consists of several mobile robots. The problem of controlling such mobile robots group can be divided into two main parts: [2]

- Task decomposition into subtasks, and their optimal distribution between separate robots in the group.
- Path planning, control and movement correction for each mobile robot.

The existing methods for solving the problem of path planning for multiple robots can be divided into two categories [3].

- Centralized approach in which the configuration spaces of the individual robots are combined into one composite configuration space which is then searched for a path for the whole composite system.
- Decoupled approach that first computes separate paths for the individual robots and then resolves possible conflicts of the generated paths. Techniques of this type assign priorities to the individual robots and compute the paths of the robots based on the order implied by these priorities.

In this work, we introduce new approach to plan multi mobile robots paths combined with features of previously known approaches. We assume that the environment is known and to assure each robot achieve its work, it should travel from a user defined initial point towards a user defined target point then the robot should go back to its initial place.

Many researchers who proposed solutions for path planning assumes that robots have the same velocities. Robot velocity defines the moment and the node where the robot will have collision and this is very important when planning paths for multi mobile robots as neglecting robot velocities will produce a plan that does not represent the actual robot behavior. All researchers who have researched in path planning problem neglect the fact that robots tasks may not be similar and that means robots may need different times to achieve their tasks in their target node. This would have significant effect on the path plan because the collision points in the return path will differs from that of the original path, which mean re-planning the path. Moreover most researchers who introduce solutions for path plan problem never discuss the effect of their solution on robots travel speed and the over all system performance [4]. In real world robots are used as multi agents systems to speed up work achievement, however all available solutions (including the one we present here) will slow down robots in order to avoid possible collisions. Also the effect of robots slowing down on creating new collision points in the system was never discussed before. Finally many researchers that use decouple planning method for path plan either used fixed priority schemes to solve robots conflicts or they use random priority scheme for the same purpose.

In this work we plan for the optimum solution not only form optimal-path point of view, but also from the over all effect of the proposed robot path on the total number of collisions and robots speed. We take the fact that robots speed may be variant into consideration when planning robots path. We also take into consideration the fact that robots tasks may not be the same in their target node. Decouple planning is used to produce the path plan with two level of priorities. In each level, robot priority is carefully selected to serve our goal in maintaining system performance.

Related Work

In [5] the robot path is defined as the path through space-time with the best score as determined by a set of user-defined evaluation functions. Their algorithm takes into account the capabilities of the robot executing generated plans, traverse-ability of space, and interactions with both predictable and unpredictable dynamic objects.

In [2] the graph edges weight is dynamically changed for path correction and collision avoidance. Their algorithm applies changes of robots' paths and speeds to avoid collisions.

[3] presents a method for finding solvable priority schemes for prioritized and decoupled planning techniques. Their algorithm is guided by constraints generated from the task specification.

The problem in [4] is decomposed into two modules: path plan and velocity plan. Optimization is achieved by minimizing a weighted sum of the most expensive time for robot to reach its goal and the total idling time of all robots.

[6] avoids the computational complexity of generating a denser search area by employing a non-uniform sampling density that increased in complex areas, leaving simple areas with lower resolution density, hence directing computational resources towards the complex areas.

In [1] a vehicle routing problem-based approach is presented to construct non-intersecting routes for the members of a mobile robot team. Their path takes into consideration robots capacity when designing tours for each robot.

[7] Present an algorithm for motion planning of multiple robots amongst dynamic obstacles. Their approach is based on a roadmap representation that uses deformable links and dynamically retracts to capture the connectivity of the free space.

[8] Uses genetic algorithms to help a controllable mobile robot to find an optimal path between a starting and ending point in a grid environment.

[9] solves path plan problem for robots in bi-connected environment with only two free nodes by moving a robot to a node if there is no robot in that node and no other robot is simultaneously entering that node.

The Proposed Algorithm

The proposed algorithm take into consideration robots velocity and robots needed time in their targets point to finalize their work. Both could be referred to as work achievement time, as robots speed defines the time needed for each robot to reach its target and to return back to its initial position, and by adding the work time for each robot in its target point to this time we will get the total time needed for each robot to achieve its work. We assume that each robot is positioned on its initial position. We assume also that each robot should visit one target position and return to its initial position in order to consider that the robot finalize its work. The robot should return to its initial position either to recharge, or to move a load between the two points or etc.

Work achievement time will have a heavy effect on the time that the collision will occur in the path plan of each robot. Consider Fig (1) which shows two robots in their initial position and trying to pass through a door to reach their target position. If robots speed are equal, collision will occur on P1 as shown in Table (1). If robot one is faster than robot two, the collision will occur at the return path of robot one not at the forward path as illustrated in Table (2). If robot one need more time to achieve its goal and it stayed in its target position for a long time that enables robot two to pass P1, collision will not occur neither in the forward path nor in the return path, this is illustrated in Table(3).

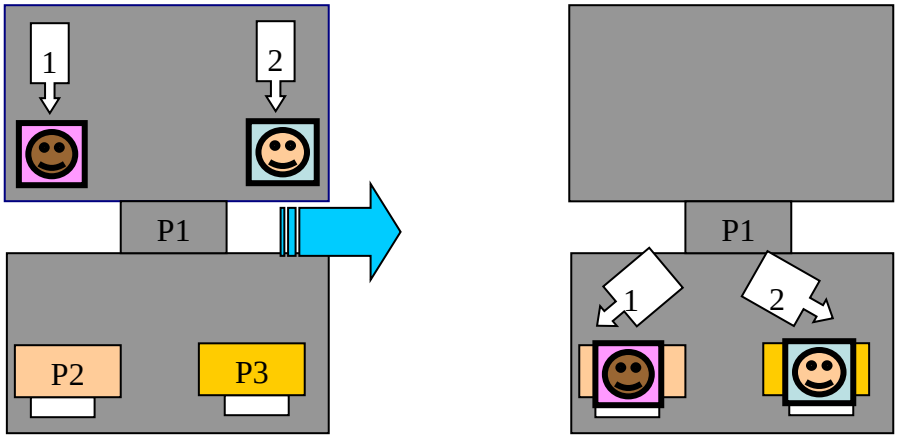

Fig (1): Two robots trying to pass through one door

Table (1): Robot one velocity equals to Robot two velocity

Time/Position	PI(1)	PI(2)	P1	P2	P3
t1	R1	R2			
t2			R1, R2 Collusion		
t3				R1	R2
t4			R1, R2 Collusion		
t5	R1	R2			

Table (2): Robot one velocity larger than Robot two velocity

Time/Position	PI(1)	PI(2)	P1	P2	P3
t1	R1	R2			
t2		R2	R1		
t3			R2	R1	
t4			R1, R2, Collusion		
t5	R1			R2	

Table (3): Robot one work time larger than Robot two-work time

Time/Position	PI(1)	PI(2)	P1	P2	P3
t1	R1	R2			
t2		R2	R1		
t3			R2	R1	
t4			R2	R1	
t5			R1	R2	
t6	R1			R2	

The algorithm starts like other decoupling planning algorithms by clustering nodes and planning shortest paths of each robot. The clustering approach used in our algorithm is specially designed for the algorithm as it takes robots steps per time as the measure of node dimensions. Robots steps will define the space that the robot will occupy per time unit. Measurements units (like mm, cm, or m for distance or sec, min, or hour for time) depend on robots speeds also. The proposed algorithm use Dijkstra (which is one of the oldest and efficient approaches in path planning) to plan the shortest path for each robot. Then the algorithm add robots speeds and robots needed time to do their work into the path plan to produce the actual time/space existence of each robot. The algorithm use sequential robot entry method to solve these conflicts, i.e. to solve a conflict between two robots one robot will occupy the node while the other robot will wait till the node is free again to be able to occupy it. In order to decide which robot should pass and which robot should wait the algorithm use two levels of priorities between robots in a way that maintain system performance. In the first level (the node level), the algorithm gives lower priority to robots that will cause more collisions to occur in the system if they pass first. This will ensures that the over all speed of work achievement of all robots is optimum as more conflicts means delaying more robots and dropping down system performance. In the second level (robot level), the algorithm gives higher priority to robots that will pass the node faster as slow robots will slow down fast robots and drop down system performance also. The combination of the two levels will produce the optimum path plan for each robot regarding system performance.

The Proposed Algorithm in Detailed

The proposed algorithm consists of the flowing steps as shown in Fig (2):

- Manual data entry

The algorithm starts by inputting robots main information including: number of robots in the graph, initial and target position for each robot, robots velocity, and robots needed time in the target node to finalize their job. In addition, the graph will be entered manually; obstacles such as walls, tables, etc. will be defined manually also.

- Clustering nodes

The nodes of the system will be clustered in a way suitable for the algorithm. Each node will represent one time step. For example if robot one velocity is 1 cm/sec and robot two velocity is 2 cm/sec, the result node length (and height) will be 0.5 cm. Robot one will pass 1 centimeter each second, robot two will pass 2 centimeter each second. For a collusion not to occur, the minimum time that should be taken into consideration in 0.5 second where robot two pass the 1 cm length, put robot two will pass 0.5 cm

during this time, so the minimum node length should be 0.5 cm. The general equation for node clustering is:

NL=NH=Min(1/VR(i))

Where NL is the node length, NH is the node height, Min is the minimum function, VR velocity of robot i, where i from 1 to M, M is the number of robots.

Robots step per time will defines the position of each robot in the time, for the above example robots step per time will be 2 steps per second (where each step is 0.5 cm as we said and the robot velocity is equal to 1 cm/sec) for the first robot and 4 step per second for the second robot. The general equation of robots steps per time is:

TR(i)=VR/NL

Where TR is the time step for robot i, where i from 1 to M.

One should note that the limitation of this clustering method is that the node size will be very small if two conditioned occurred: first number of robots is very large, and second the speed differences between robots is large also. If the two conditions occurred, the computation time of the algorithm will increase due to the increase of number of nodes in the system.

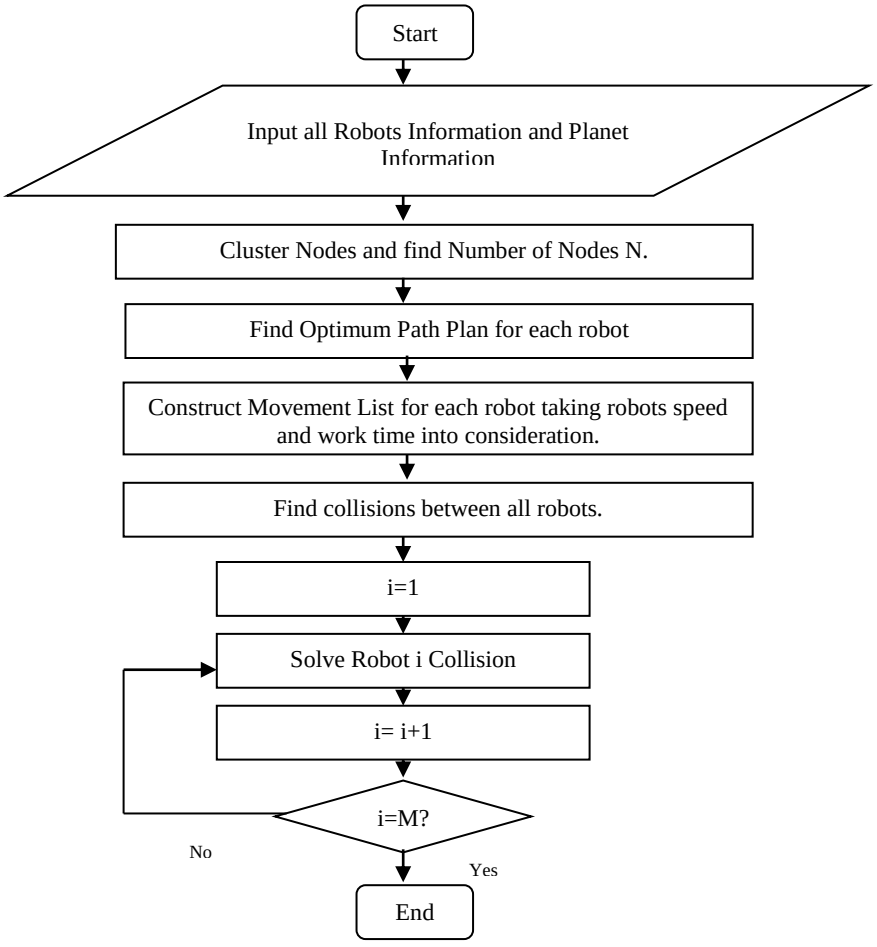

Fig (2): Flowchart of the Proposed Algorithm

- Constructing Robots Paths

The initial path for each robot will be constructed using Dijkstra algorithm that finds the shortest path of each robot from its initial position to its target position.

After that, the actual time/space for each robot will be constructed as follows:

1. Finding the Forward Path

The forward path is the path from the initial position to the target position. Robots time step will be considered during constructing this path. For example if robot one velocity is 2 cm/sec and robot two velocity is 1 cm/sec, the result will be that robot one will pass each node by one time step, while robot two will need two time step to pass each node. This will take robots speed into consideration when constructing the path, also it will take robots availability within time and space for each robot in each step. Path of each robot will be stored in a list:

Path(i, t) = n

Where i is the robot number, i is from 1 to M, t is the momentum time, n is the node number, n is from 1 to N, and N is the total number of nodes in the system.

2. Adding work time

After constructing the forward path, the time needed for each robot in its target position will be added. This is very important as other robots may need to pass through the same node while the robot is still working in that node. Also it is important as it will define the moment that each robot will starts moving back towards its target position.

3. Defining the return path

The return path is the path that robot will pass from its target node back to its initial node. We assume that each robot will use the same nodes obtained using Dijkstra to construct the return path but in backward order. Velocity of each robot will be taken into consideration also when constructing the return path.

- Finding Collisions

The algorithm now search for collision between robots. Collision is defined as two robots try to occupy the same node at the same time. Collisions are stored in a list for each robot as follows:

Collision (i, j, t) = n

Where i and j are robots number, t is the momentum time that collision will occur, n is the node where collision will occur.

- Solving Collisions

Solving collision between two robots are done in two levels.

1. Level One – Prioritizing Robots in Node Level

In node stage the algorithm checks solutions for each robots that does not effect the over all system collisions. The stage starts by

- Take two robots that have collision as shown in Fig (3).

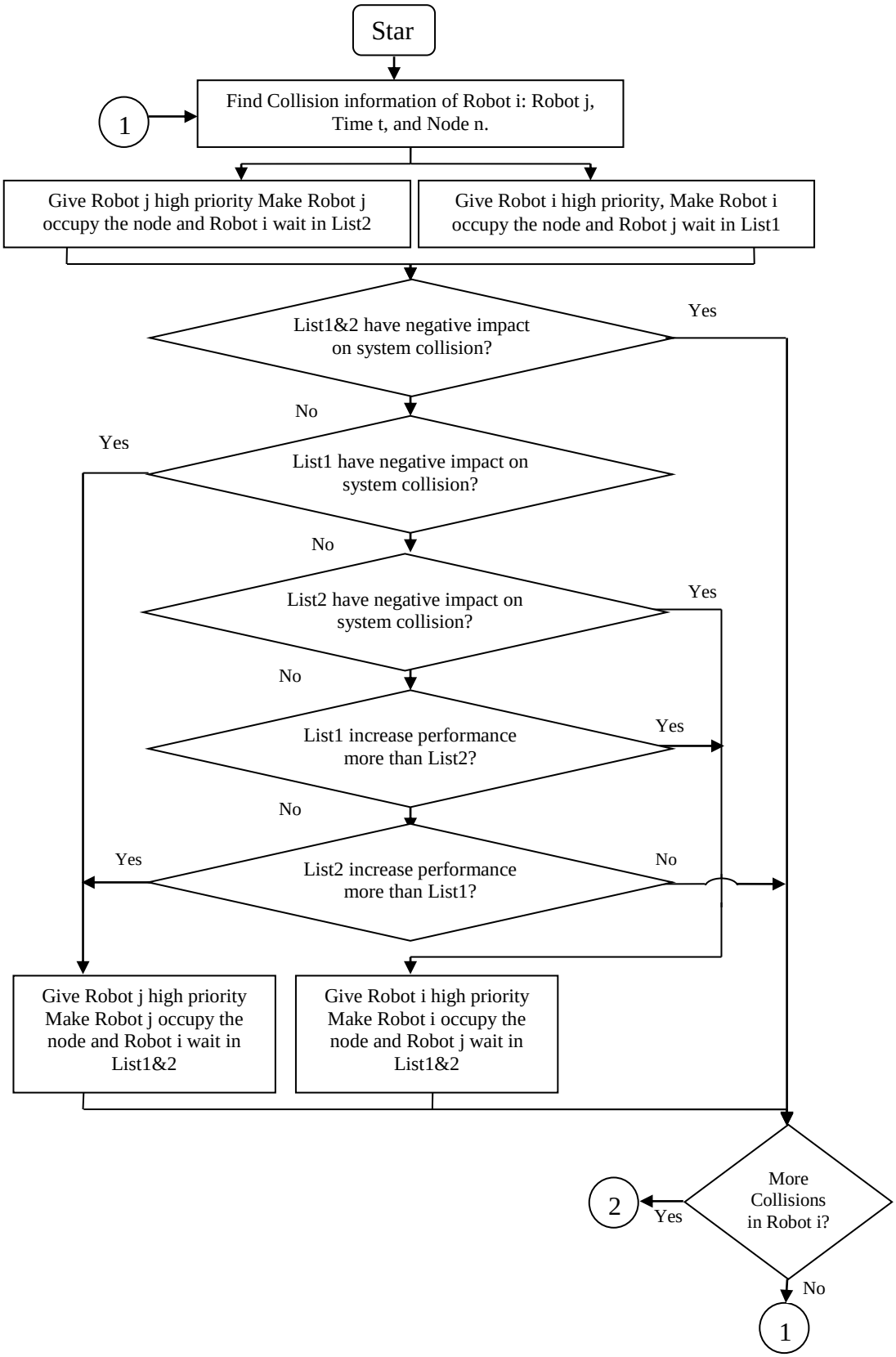

Fig (3): Solve Robot i Collision - Level One Priority – Node Level

- Give higher priority to the first robot and construct two temporary movement lists for both robots in a way that makes this robot occupy the node while making the second robot wait. We will refer to the two temporary lists in this group as movement group one (or simply List1).
- Give higher priority to the second robot and construct two temporary movement lists for both robots that makes the second robot occupy the node while making the first robot wait. We will refer to the two temporary lists in this group as movement group two (or simply List2).
- If one of the two temporary movement groups (List1 or List2) has negative impact on previously planned robots, the algorithm will not use this movement group. In other words, if one of the movement groups caused collisions to robots that their optimum path was already planned, the algorithm will neglect this movement.
- If one of the two temporary movement groups cause increase in the over all system collisions, the algorithm will not use this movement group. i.e. if one of the movement groups caused more collisions between robots on hand or other robots in the system, the algorithm will neglect this movement.
- If one of the two temporary movement groups cause noticeable decrease in the over all system collisions, the algorithm will use this movement. Normally one may assume that solving one collision will decrease the number of total system collision by one, but actually this is not the case as one of the solutions may cause to solve all collisions in the system or at least to solve many other collisions. So if the solution being considered result in less system collisions than the other solution, the algorithm choose this solution.

The above steps will be done for all nodes that the two considered robots have collision on. Some nodes will be chosen using the above criteria, for other nodes, i.e. nodes that have the same impact on the system performance (either they both have the same negative impact on the over all system collisions, or they both have the same positive impact on the over all system collisions), the selection will be done in level two.

2. Level Two – Prioritizing Robots in Robot Level

The result of stage one will be two temporary movement groups that for some nodes the solution is equal in both groups, i.e. one robot gets the higher priority in that node on the two groups. While in other nodes the first robot get higher priority in the first group, while the second robot get higher priority in the second group. Now the algorithm selects between the two groups depending on the group that cause faster speed for work achievement for both robots. As the two robots have different speeds to move and to work, giving priority to one robot may slow down the over all achievement time of the two robots more than if we give the priority to the other robot. The algorithm will choose the group that enables both robots to achieve their work faster as follows:

T1 = W(i) + W(j) in List1

T2 = W(i) + W(j) in List2

If T1 <= T2 Then Select List1

Else Select List2

Where T1 and T2 are the summation of work achievement time for robot i and robot j in the temporary movement lists one and two respectively as shown in Fig (4).

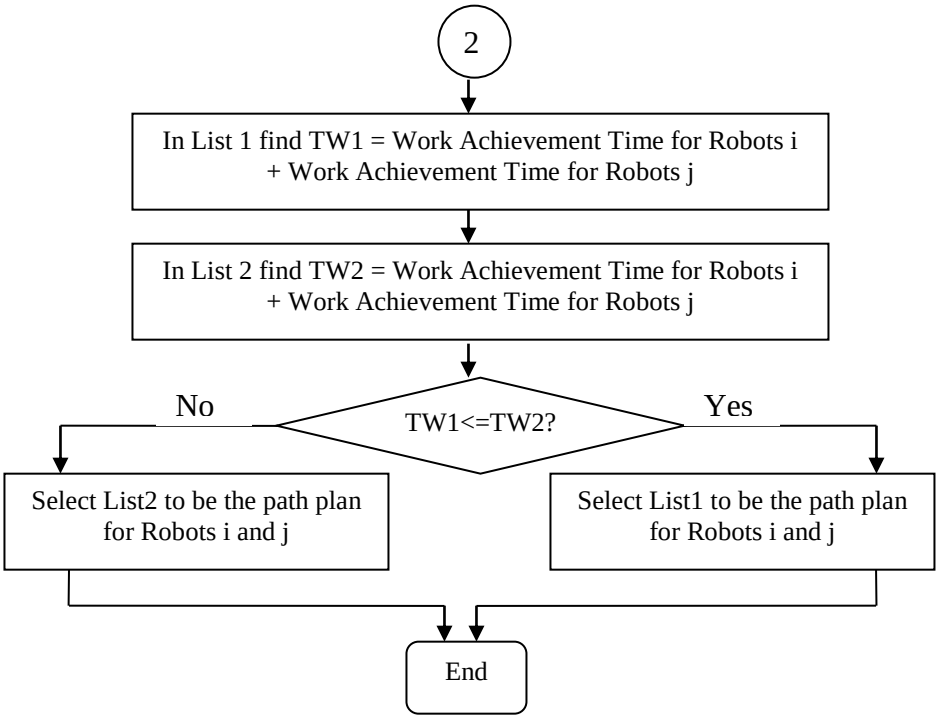

Fig (4): Solve Robot i Collision - Level Two Priority – Robots Level

Results and Discussions

In this section, we will discuss the proposed algorithm by discussing several study cases.

Case Study One: Door problem

Let discuss the situation shown in Fig (5). The planet consists of eight nodes N0 – N8, N3 and N5 contains obstacles. Robot one R1 is positioned in N0 and its target is N6, while robot two R2 is positioned in N2 and its target is N8.

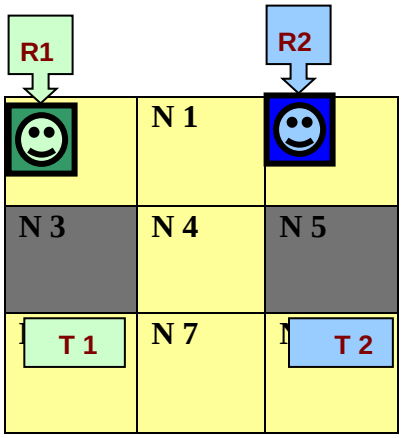

Fig (5): Case Study One – Door Problem

Case 1:

VR1 = VR2 = 1 time step and TW1 = TW2 = 0

Where V is the velocity of R1 and R2, TW is the time needed for R1 and R2 to achieve their goal in their target node.

The algorithm will construct both robots path to be as shown in Table (4). Robot one will pass nodes 0, 1, 4, 7, and 6 in the forward path, and will pass nodes 6, 7, 4, 1, and 0 in the return path. Note that node 6 was repeated as robot velocity is taken into consideration, robot need time to turn back in the node and the turning back movement is done by a velocity equal to the robot movement velocity. R2 will pass nodes 2, 1, 4, 7, and 8 in the forward path, and the same nodes but in opposite order in the return path. The two robots will have collision on node 1,4, and 7 in both the forward and the return path. The solution found by the proposed algorithm is shown in Table (4).

First the algorithm choose to delay R1 on node 0 in the Robot level as delaying R1 in node 0 or delaying R2 in node 2 have the same impact on node level (both solve all collisions between the two robots) and the same impact in the robot level (both produce the same achievement time).

Table (4): Door problem – Case 1

Path Plan for Case One										
	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9
R1	0	1	4	7	6	6	7	4	1	0
R2	2	1	4	7	8	8	7	4	1	2

Final Path Plan for Case One											
	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
R1	0	0	1	4	7	6	6	7	4	1	0
R2	2	1	4	7	8	8	7	4	1	2	

Case 2:

VR1 > VR2 and TW1 = TW2 = 0

The algorithm will construct both robots path to be as shown in Table (5). Now the two robots has one collision only at node 1 at time 3, and that demonstrate the effect of velocity variation on robots collisions. The solution found by the proposed algorithm is shown in Table (5). The algorithm will choose to delay R2 on node 2 in node level as R1 is already occupying the node and in order to free node 0 either R1 should return and wait on node 0 for three time step and cause three more collisions to occur with R2, or putting R2 on hold one time step and cause one more collision to occur at node 7 at time 12. The second collision at node 7 is solved by delaying R1 in the robot level for one time step as delaying R2 means putting R2 on hold for three time step.

Table (5): Door problem – Case 2

Path Plan for Case Two																
	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15
R1	0	0	1	1	4	4	7	7	6	6	6	6	7	7	4	4
R2	2	2	2	1	1	1	4	4	4	7	7	7	8	8	8	8

Final Path Plan for Case Two																
	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15
R1	0	0	1	1	4	4	7	7	6	6	6	6	6	7	7	4
R2	2	2	2	2	1	1	1	4	4	4	7	7	7	8	8	8

Case 3: VR1 = VR2 and TW1 > TW2

The algorithm will construct both robots paths to be as shown in Table (6) where the two robots has three collisions at nodes 1, 4, and 7 in the forward path only, the return path has no collision at all as a result of adding robot R1 work time into the path plan. The solution found by the proposed algorithm is shown in Table (6). The algorithm choose to delay R1 at node 0 in the node level as it will solve all collisions while delaying R2 at node 2 will create 3 more collisions and this demonstrate the difference in the decision compared to case one. In case one putting R1 or R2 on hold in time 1 does not have any effect on system performance, while in this case it does. If robots priority was chosen randomly or chosen based on a fixed scheme in a manner that gives higher priority to R1, the solution will solve the conflict in node 1 at time 1 by delaying R2, but at the same time it will creates three more conflict at nodes 7, 4, and 1 in the return path.

Table (6): Door problem – *Case 3*

Path Plan for Case Three											
	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
R1	0	1	4	7	6	6	6	7	4	1	0
R2	2	1	4	7	8	8	7	4	1	2	

Final Path Plan for Case Three												
	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11
R1	0	0	1	4	7	6	6	6	7	4	1	0
R2	2	1	4	7	8	8	7	4	1	2		

Case 4: VR1 > VR2 and TW1 > TW2

The algorithm will construct both robots path to be as shown in Table (7) where the two robots has one collision at node 1 at time 3. The solution found by the proposed algorithm is shown in Table (7). Compared to case two, the first solution in case two is the same solution chosen by the algorithm here but in case two we noticed that another collision will occur in the system while here no more collations will occur as a result of increasing work time of robot one.

Table (7): Door problem – Case 4

	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16
R1	0	0	1	1	4	4	7	7	6	6	6	6	6	6	6	7	7
R2	2	2	2	1	1	1	4	4	4	7	7	7	8	8	8	8	8

	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16
R1	0	0	1	1	4	4	7	7	6	6	6	6	6	6	6	7	7
R2	2	2	2	2	1	1	1	4	4	4	7	7	7	8	8	8	8

Case 5: VR1 > VR2 and TW1 < TW2

The algorithm will construct both robots path to be as shown in Table (8) where the two robots has one collision at node 7 at time 6. The solution found by the proposed algorithm is shown in Table (8). The algorithm will choose to delay R2 on node 4 for one time step on the robot level as it is faster to achieve work for both robots compared to delaying R1 that will require that R1 waits for two time step. The alternative solution would be delaying R1 for two time steps in node 6.

Table (8): Door problem – *Case 4*

Path Plan for Case Five													
	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
R1	0	1	4	7	6	6	7	4	1	0			
R2	2	2	1	1	4	4	7	7	8	8	8	8	8

Final Path Plan for Case Five													
	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
R1	0	1	4	7	6	6	7	4	1	0			
R2	2	2	1	1	4	4	4	7	7	8	8	8	8

Notice that in time T=T6 and T=T7 a swap will occur between the two robots, i.e. the two robots will swap their location R1 that occupy node 7 will occupy node 4 and R2 that occupy node 4 will occupy node 7. This is the only case that our algorithm cannot solve because in order to solve a swap one of the two robots should wait in an earlier point in the path plan i.e. one of the robots should clear the path to the other robot not only clear a step. This will require that the algorithm check for swaps in the Find collisions part of the algorithm, then to continue searching for swaps each time a collision solved. We preferred in this version of our work to keep this situation unsolved, and to solve it in detailed in another work. Note also, this problem could be solved in finding a temporary.

Case Study Two: Cross Road problem

Lets discuss the situation shown in Fig (6). The map consists of twenty five nodes N0 – N24, where N1, N6, N8 and N9 contains obstacles. Robot one R1 is positioned in N0 and its target is N19, while robot two R2 is positioned in N4 and its target is N22.

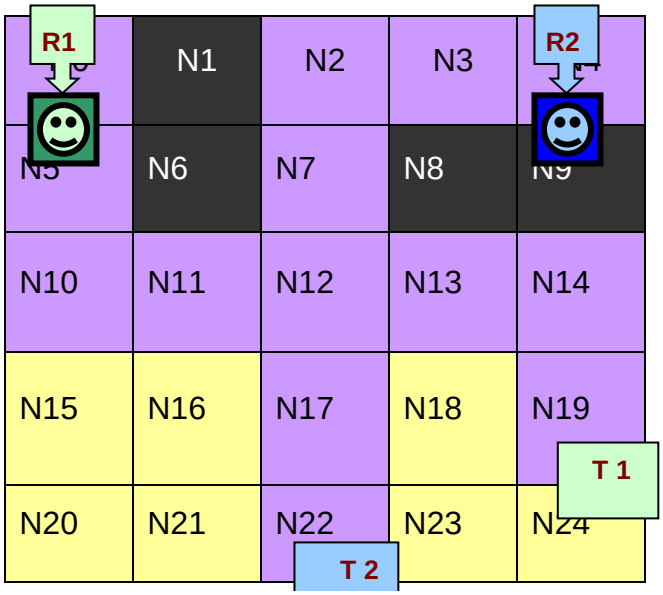

Fig (6): Case Study Two – Cross Road Problem

***Case 1:* VR1 = VR2 = 1 time step and TW1 = TW2 = 0**

Where V is the velocity of R1 and R2, TW is the time needed for R1 and R2 to achieve their goal in their target node.

The algorithm will construct both robots path to be as shown in Table (9). The two robots will have collision on node 12 at time 4. The solution found by the proposed algorithm is shown in Table (9). The algorithm choose to delay R1 on node 11 on the robot level as both solution (delaying R1 or R2) have the same impact on the node level and on the robot level.

Table (9): Cross Road Problem – *Case 1*

Path Plan for Case One																
	T0	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**	**T9**	**T10**	**T11**	**T12**	**T13**	**T14**	**T15**
R1	0	5	10	11	12	13	14	19	19	14	13	12	11	10	5	0
R2	4	3	2	7	12	17	22	22	17	12	7	2	3	4		

Final Path Plan for Case One																
	T0	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**	**T9**	**T10**	**T11**	**T12**	**T13**	**T14**	**T15**
R1	0	5	10	11	11	12	13	14	19	19	14	13	12	11	10	5
R2	4	3	2	7	12	17	22	22	17	12	7	2	3	4		

Case 2:

VR1 > VR2 and TW1 = TW2 = 0

When robot one velocity is larger than robot two velocity, in many cases the collision will not occur anymore as the two robots have only one node in common. However we took here an example where the speed of R1 and R2 will cause them to have collision again. The algorithm will construct both robots path to be as shown in Table (10). The solution found by the proposed algorithm is shown in Table (10). The algorithm will choose to delay R1 on node 13 for three time steps on the node level as R2 is already occupying the node, and delaying R2 requires 5 time steps.

Table(10): Cross Road Problem – Case 2

Path Plan for Case Two															
	T11	**T12**	**T13**	**T14**	**T15**	**T16**	**T17**	**T18**	**T19**	**T20**	**T21**	**T22**	**T23**	**T24**	**T25**
R1	13	14	14	19	19	19	19	14	14	13	13	12	12	11	11
R2	2	2	2	2	7	7	7	7	7	12	12	12	12	12	17

Final Path Plan for Case Two															
	T11	**T12**	**T13**	**T14**	**T15**	**T16**	**T17**	**T18**	**T19**	**T20**	**T21**	**T22**	**T23**	**T24**	**T25**
R1	13	14	14	19	19	19	19	14	14	13	13	13	13	13	12
R2	2	2	2	2	7	7	7	7	7	12	12	12	12	12	17

*Case 3:*VR1 = VR2 and TW1 < TW2

The algorithm will construct both robots paths to be as shown in Table (11) where the two robots has two collisions at node 12 at time 4 and 11 respective. The solution found by the proposed algorithm is shown in Table (11). The algorithm choose to delay R1 at node 0 in the robot level as both solution have the same impact on node and robot level.

Table (11): Cross Road Problem – Case 3

Path Plan for Case Three																
	T0	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**	**T9**	**T10**	**T11**	**T12**	**T13**	**T14**	**T15**
R1	0	5	10	11	12	13	14	19	19	14	13	12	11	10	5	0
R2	4	3	2	7	12	17	22	22	22	22	17	12	7	2	3	4

Final Path Plan for Case Three																	
	T0	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**	**T9**	**T10**	**T11**	**T12**	**T13**	**T14**	**T15**	**T16**
R1	0	5	10	11	11	12	13	14	19	19	14	13	12	11	10	5	0
R2	4	3	2	7	12	17	22	22	22	22	17	12	7	2	3	4	

Conclusion and future work

In this paper, optimum path planning has been presented for multiple mobile robots. The work uses decoupled planning in time and space and sequential robot entry according to selective priority schemes to solve collisions between robots. The work takes into consideration robots with variant velocities and variant time to achieve their goal in their target node which both referred to as work achievement time.

Discussion result proves the effect of work achievement time on the time and space where collision will occur. Also it proves the effect of work achievement time on the priority that will be assigned to each robot on the collision nodes and the importance to include work achievement time as a constraint when planning robots paths. In this work, two levels of priority are used to maintain system performance. The first priority level maintain the over all system collisions by giving high priorities to robots that does not increase the total number of system collisions, while the second level of priority maintain robots work achievement time by giving priorities to robots in a manner that does not slow down the over all system speed. The work is tested with different number of robots and different types of maps, and the algorithm proved its efficiency in finding the optimum solution regarding system performance for each robot in the collision points. Robots with repetitive work was not studied too often in path planning. In this study we noticed that robot with repetitive work will have collisions in the second work cycle differs from that of the first cycle (work cycle is the time from initial node to target node and back to initial node). We also noticed that after a predefined time (that lies in the 1st, 2nd, or etc.) of work cycle, collision will be fixed in the same nodes. We extend this work to briefly study this phenomenon. The work extension will take into consideration the swap problem, and the future proposed work is to find an optimum point where one of the two robots should wait on till the second robot pass the swap area.

CONFLICT OF INTERESTS.

- There are no conflicts of interest.

References

[1] 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**.

[2] 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**.

[3] Maren Bennewitz, Wolfram Burgard, and Sebastian Thrun**,** "***Exploiting Constraints During Prioritized Path Planning for Teams of Mobile Robots***", Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), **2001**.

[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] Marc G. Slack and David P. Miller**,** "***Path Planning Through Time and Space in Dynamic Domains"***, Proceeding of the 10th International Joint Conference on Artificial Intelligence, August **1987**.

[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), **2006**.

[7] Russell Gayle, Avneesh Sud, Ming C. Lin, and Dinesh Manocha**,** "***Reactive Deformation Roadmaps: Motion Planning of Multiple Robots in Dynamic Environments***", Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), **2007**.

[8] Ismail AL-Taharwa, Alaa Sheta and Mohammed Al-Weshah**,** "***A Mobile Robot Path Planning Using Genetic Algorithm in Static Environment***", Journal of Computer Science, Vol. 4, Issue 4, April **2008.**

[9] Pavel Surynek**,** "***Path Planning for Multiple Robots in Bi-connected Environments***", ITI Series, Vol. 431, **2009**.

التخطيط الامثل لمسار عدة روبوتات متنقلة مع سرعة انجاز العمل كمحدد

بشرى كاظم عليوي

قسم هندسة السيطرة والنظم، الجامعة التكنولوجية، بغداد، العراق

bushrakad@yahoo.com

الخلاصة

هذا البحث يمثل التخطيط الامثل لمسار عدة روبوتات متنقلة بين نقطة البدء باتجاه نقطة الهدف ثم العودة الى نقطة البداية وتجنب الاصطدام وبدون ابطاء شديد في سرعة الروبوتات. وتم الاخذ بنظر الاعتبار في تصميم النظام سرع الروبوتات وكذلك الوقت المستهلك لحركة الروبوتات باتجاه نقطة الهدف، حيث كلاهما متغيران لجميع الروبوتات. من اجل تحقيق اهداف العمل استخدمت طريقة الوقت والفراغ (time and space method) مع طريقة تسلسل الدخول (sequential entry method) لتخطيط حركة الروبوتات. ايضا تم استخدام اثنين من مستويات الاولوية ليتم اختيار الاولوية للروبوتات بعناية لتجنب التصادم بين الروبوتات والمتمثلة بأولوية مستوى العقدة واولوية مستوى الروبوت، نلاحظ في اولوية مستوى العقدة فالاولوية للروبوتات التي تقلل عدد التصادمات في كل النظام، في حين الاولوية في مستوى الروبوت للروبوتات التي تسبب سرعة في انجاز العمل. تم اختبار العمل على عدد مختلف من الروبوتات وانواع مختلفة من الخرائط، حيث اثبتت الخوارزمية كفائتها في ايجاد الحل الامثل بخصوص اداء النظام لكل الروبوتات في نقاط التصادم. حيث تم تطبيق الخوارزمية بأستخدام لغة البرمجة فيجوال بيسك/ اكسل.

الكلمات الداله: – الانسان الالي، التخطيط الامثل للمسار، الربوت المتعدد السيطرة.