Research Journal of Applied Sciences 11 (10): 1060-1068, 2016
ISSN: 1815-932X

Robot Path Planning Based on Improved Max–min Ant Colony Optimization Algorithm in Dynamic Environment

Ali Hadi Hasan
College of Information Technology, University of Babylon, Babylon, Iraq

Abstract: This paper proposes a method to find an optimal local path based on an improved version of the MAX-MIN Ant Colony Optimization (ACO) algorithm in dynamic robot path environments. It uses the grid method to decompose two-dimensional space to build class nodes that contain the information of the space environment. The proposed improvement of MAX–MIN ACO algorithm occurs in the stage of mixing pheromone trail updating with D* algorithm strategies to construct the consequence modified (deposited) pheromone trail update in each iteration. Thus the robot (ant) analyses the environment from the goal node (food) and computes the cost (pheromone deposition) for all the nodes to the start node (nest). The robot uses tour construction probabilities to choose the best solution to move it from the start node through dynamic environment which contains dynamic obstacle moving in free space by finding and displaying the optimal path. Some experimental results that are simulated in different dynamic environments, show that the robot reaches its target without colliding obstacles and finds the optimal local path with minimum iterations, minimum total path cost and minimum time occupy.

Key words: Mobile robot path planning, dynamic, environment, MAX-MIN, ACO algorithm

INTRODUCTION

The problem of path planning is to determine either existence of solution in a form of a sequence of state transitions through a graph from start state to a goal state or non- existence of such a solution. The path is said to be optimal when the total cost of transitions crossing through a possible sequence of states in the graph is the minimum (Stentz, 1994). Path planning of a mobile robot is finding a collision-free path from the initial point to target point. This is done through using an optimal performance criterion which can be time, energy or distance which is the most commonly used. Path planning algorithms are classified into off-line and on-line based on the obtainable information about environment.

When the full information about static obstacle and the direction of robots movement in environment is already available, the path planning is described as off-line (global) path planning. But when information about environment is unknown, the robot will move depended on data supplied by sensors which collected and convey data as the robot wanders in environment. In such case the path planning is said to be an on-line (local) path planning. Initially, on-line path planning acts in an off-line mode but it turns to on-line path planning when it discovers new changes in obstacle scenario (Raja and Pugazhenthi, 2012). The D*algorithms have been used extensively in robotics to move the robot navigation in unknown, partially known, and changeable environments (Stentz, 1994). Many algorithms exist for planning paths improve from D* such as Focussed D*(Stentz, 1994), D* Lite (Stentz, 1994) and field D*(Ferguson and Stentz, 2006)

Ant Colony Optimization (ACO) is a member of the meta-heuristic algorithms that emulate the capability of ant colony of discovering the shortest path between the nest and the food source. ACO as one of the prosperous application of swarm intelligence (which deals with groups' behavior like ants rather than individuals') can be utilized in solving complex optimization problems by adding problem-dependent heuristics (Saeheav *et al*., 2009). The MAX–MIN Ant System (MMAS) algorithm strongly exploits the search history by permitting the best solutions to add pheromone during the process of pheromone trail updating. Moreover, the pheromone convergence of the search is avoided due to the simplicity of the mechanism used in limiting the strength of pheromone trails. It is also possible to extend the MMAS to be added to the local search algorithms. The solutions produced by the ants with local search algorithms have been improved by the best performing ACO algorithms for many different combinatorial optimization problems. MMAS is one of the most effective performing ACO algorithm for optimal path planning as the researchers' outcomes show (Stutzle and Hoos, 2000).

Mei *et al.* (2006) developed a new hybrid method between ACO as a global path planning algorithm and APF as a local planner method to mobile robot. The pheromone generated by ACO is invested as global information to lead the APF to jump to local minimum. Their simulation results showed that the global optimal and real-time obstacle avoidance can be satisfied by hypoid APF and ACO algorithm. They were persuaded that ACO exceeds GA in path planning by comparing their performance.

Brand *et al.* (2010) found a method of using ACO for robot path planning in a dynamic environment. They found the best shortest and collision free path in a grid map. They showed that when an obstacle is add and blocked that path the basic ACO algorithm can successfully make a global re-initialization, which consisted of resetting the entire map to its initial pheromone levels and then re-running the algorithm using a block as the new start position.

Nicholas Charabaruk offered an improved Max-Min ACO algorithm by adding the Max-Min ACO algorithm to the Bug2 algorithm to find dynamic path planning with better solutions and faster convergence speeds. They believed that their method are viable for global path planning and when the path was blocked the system is re-initialized and the old pheromone trails are leaving they also increase the processing speed. They found the results of their improved method have shorter average path lengths and that they vastly reduce processing times than standard ACO and Max-Min ACO algorithms do.

Ant colony optimization algorithm: M. Dorigo's Ph. D. dissertation presented Ant Colony Optimization (ACO) algorithm in 1992 to simulate the manner of ants to get solutions for optimization problems. Biological ants follow a complicated social manner in their looking for food sources. This manner depends mainly on (Pheromones) they deposit across their path to food source. These pheromones function not only as attracter to other ants but also as a guide that leads to the food source. The more amount of pheromone laid along the path, the more number of ants will follow that path. The shortest path to food source will be the one that has the most pheromones as the greatest numbers of ants can pass it during less period of time (Brand *et al.*, 2010; Dorigo and Stutzle, 2004).

Basic ant colony optimization algorithms: When ACO algorithms is applied to the optimal path planning, arcs are utilized as solution constituents. Each arc(i,j) is connected to a trail of pheromone trail $\tau_{ij}(t)$. During the run of algorithm, the pheromone trails are altered by pheromone trail evaporation and trail reinforcement (Sun and Tian, 2010).

Tour construction: When (m) ants are put in (m) randomly selected nodes (locations), every ant starts to move to a node. Its movement to that node is probabilistic choice which is biased by pheromone trail $\tau_{ij}(t)$ and by a local heuristic information η_{ij} which is a function of arc length. All ACO algorithms for the optimal path planning use $o_{ij}(t) = 1/ d_{ij}$. Ants choose the nodes which are related by arcs that have high pheromone trails. In ACO algorithms the probability with which an ant chooses to go to a node j is:

$$p_{ij}^{k}(t) = ([\tau_{ij}(t)]^{\alpha}[ij]^{\beta})/(S(1\in N_i^k)[\tau_{il}(t)]^{\alpha}[il]^{\beta}) \quad (1)$$

The parameters α and β decide the relative importance of the pheromone trail and the heuristic information respectively, and the feasible neighbor set of nodes that have not yet been visited by ants is denoted by N_i^k. The nodes visited in every tour will be stored in the ant's memory which then will be used to define N_i^k in every step of construction to guarantee the generation of only valid Hamiltonian cycles. Moreover, it permits the ant to trace its tour again as soon as it is finished, and it deposits pheromone on the arc there (Stutzle and Hoos, 2000; Brand *et al.*, 2010; Reshamwala and Vinchurkar, 2013).

Pheromone update: When the tour construction is accomplished, the updating process of the pheromone trails starts. It is achieved through evaporating the pheromone trails by a constant factor to a lower amount, and thus the ant will be permitted to deposit pheromone on the arc they visit. The update happens according to the following rule:

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t)+\sum_{k=1}^{m}\Delta\tau_{ij}^{k}(t) \quad (2)$$

Where:

ρ = The parameter (with $0\le\rho<1$) is evaporation rate

$\Delta\tau_{ij}^{k}(t)$ = Represents the amount of pheromone ant k puts on the arcs it has used in its tour

The accumulation of the pheromone trails is avoided with help of evaporation as the pheromone trails associated with arc that are not chosen will face exponential decrease. This allow the algorithm ignore the unfavorable choices. In ACO algorithms can be $?\tau_{ij}^{k}(t)$ defined as follows:

$$\Delta\tau_{ij}^{k}(t)+\begin{cases}1/L^{k}(t) & \text{if arc (i, j) is used by ant}\\ & \text{K in iteration t.}\\ 0 & \text{otherwise}\end{cases} \quad (3)$$

where $L^k(t)$ Stands for the tour length of the k^{th} ant. By Eq. 3, the greater amount of pheromone indicates the favorability of the node selected by ants to move to. Generally speaking, when many ants used arcs in shorter tours, these arcs will receive extra pheromone and will have more chances of being selected in future iteration of algorithm (Stutzle and Hoos, 2000; Brand *et al*., 2010; Abdelbar and Wunsch, 2012).

MAX–MIN ant system: The stronger exploitation of the best solutions found throughout search and search space analysis can lead to an improved performance of ACO. However, using a curious search potentially arouse the problem of premature stagnation of the search. The best performance of ACO algorithms for avoiding early search stagnation can be obtained by combining an improved exploitation of the best solutions found during the search with an effective mechanism. MAX-MIN ant system which has these requirements is different from AS in the following aspects.

At each iteration only one single ant adds pheromone for exploiting the best solution, this may be done either during an iteration by (iteration-best ant) or during the run of algorithm by the (global-best ant) (Stutzle, and Hoss 2000; Brand *et al*., 2010; Dorigo and Stutzle, 2004) Stagnation is avoided through limiting the scope of pheromone trails on every solution to an interval [min; max]. The deliberate initialization of pheromone to τ_{max} fulfill a greater level of solutions exploration at the beginning of algorithm (Stutzle and Hoos, 2000).

Pheromone trail updating: The pheromone trails after each iteration is updated by using only one single ant in MMAS. The following rule gives the modified pheromone trail update:

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t)+\Delta\tau_{ij}^{best}(t) \quad (4)$$

where, $\Delta\tau^{best} = 1/f(S^{best})$ and $f(S^{best})$ denotes the solution cost of either the iteration-best (s^{ib}) or the global-best solution (s^{gb}). The use of only single ant in pheromone trail has also been presented in ACS where only (s^{gb}) is exploited (though s^{ib} has been used in some similar experiments), MMAS concentrate on the use of the iteration-best solutions. The choice of only one solutions (either s^{ib} or s^{gb}) to be used in the pheromone update will rule the manner of exploiting the search in MMAS. When only s^{gb} is used, it will be largely reinforced. The search will focus on this solution limiting the chances of using other solutions that can be better. This entails the possibility of exploiting inferior solutions.

The case is different when s^{ib} is used in trail update because the iteration best solutions varies from iteration to iteration. This allows a greater number of solutions components to have occasional reinforcement. However, when MMAS is used with local search to solve larger optimal path, the effective strategy will be the use of mixed strategies represented by using s^{ib} as a default solution for updating the pheromones and using s^{gb} every fixed number of iterations. The best strategy appeared the use of a dynamical mixed strategy which increases the frequency of using s^{gb} for the pheromone update during the search (Stutzle and Hoos, 2000).

Pheromone trail limits: Search stagnation may happen regardless the choice between the iteration-best and the global-best ant for the pheromone trail update. In optimal path planning, it may occur when pheromone trail in one arc is higher than that in all others. In such a case and in light of the probabilistic choice governed by Eq .1 an ant which favors this solution will set it repeatedly and the exploration of the search space stops. One way to avoid stagnation is by affecting the probabilistic choices of the next solution. This can be done through pheromone trails and the heuristic information. The latter depends on the problem and is not dynamic throughout the algorithm run. Concerning the pheromone trail, the MMAS sets limits τ_{min} and τ_{max} on the minimum and maximum pheromone (such that for all pheromone trails $\tau_{ij}(t)$, $\tau_{min} \leq \tau_{ij}(t) \leq \tau_{max}$ to prevent the extreme accumulation of the pheromone during the run of algorithm .Thus the high difference between pheromone trails will be avoided. The limits imposed by the MMAS will be checked after each iteration. if we have $\tau_{ij}(t)>\tau_{max}$, we set; $\tau_{ij} = \tau_{max}$ analogously $\tau_{ij}(t)<\tau_{min}$ we set $\tau_{ij}=\tau_{min}$ Also note that by enforcing $\tau_{min}>0$ and if $\theta_{ij}<\infty$ for all solution components any solution component may probably be chosen.

For every choice point, the MMAS convergence occurs when all solution components have pheromone trails τ_{min}, except one solution has τ_{max}. In such a case, the choice of max pheromone will construct the solution which is regarded the best solution found by the algorithm, and it is constructed with probability ρbest that is higher than 0. Hence the solution component with τ_{max} is the right choice that an ant choose to construct the best solution at each point. Obviously τ_{max} and τ_{min} directly affects the probability of selecting the solution component at any choice point (Stutzle and Hoos, 2000).

Pheromone trail initialization: Pheromone trail initialization is done by setting t(o) to arbitrarily high

value and consequently all trails will correspond $\tau_{max}(1)$. This kind of initialization makes the exploration of solutions go up during the first iteration of algorithm. Because of trail evaporation (determined by parameter ρ), the difference between pheromone trails after first iteration will be of a ratio ρ, after the second one it will beρ^2...etc.

The value $(1-\rho)$ (avg τ_{dec}) $/(1-\rho_{dec})$ is the ratio between τ_{min} and the amount of pheromone deposited on a solution element. Depending on the chosen empirical parameters setting, this ration is significantly higher than the relative difference among the pheromone trail when it is initialized at τ_{max}. When the pheromone trail is initialized at τ_{max}, the selection probabilities of Eq. 1 will develop slowly. Thus, the exploration of solutions is preferable. The experimental results confirm the conjecture that the larger exploration of the search space due to setting $\tau(1) = \tau_{max}$ improves MMAS' performance (Stutzle and Hoos, 2000).

MATERIALS AND METHODS

The proposed improve path planning design: This section present the design of the new method to find the optimal path planning. First, we get 2D workspace environment with static and dynamic obstacle expressed by regularity grids. Second, the pheromone trails are updated by mixing with D* algorithm and the ants accumulate the deposit pheromone on the nodes they have visited from goal to start nodes state and it is checked if the gate raise state is found to make it non traversable cell. Later, the robot (ant) moves from start state based on tour construction probabilities through the traversable cell and it avoids the dynamic obstacle until it reaches the goal state. Fig. 1 shows the design structure of the main stage for building the new method to find the optimal path.

Determined the environment: In the beginning of design, the new method must select any environment image from the robot's environment library stored in the main storage and displays it as a grid cell environment on screen. Then start state position (nest) on the free space of the path is selected by the mouse first click and this cell is converted to red color and displayed it on screen. The second mouse click on another free space selects the goal state position (food) and converts this cell to gold color. A class data structure labeled "node" is designed which contains all the needed information and the dynamic array of type node is defined to store the information gotten from image environment. Each cell in the image environment is decomposed to define the coordinate position in terms of height and width, tag and statue. Pheromone trail updating mixing with D*algorithm

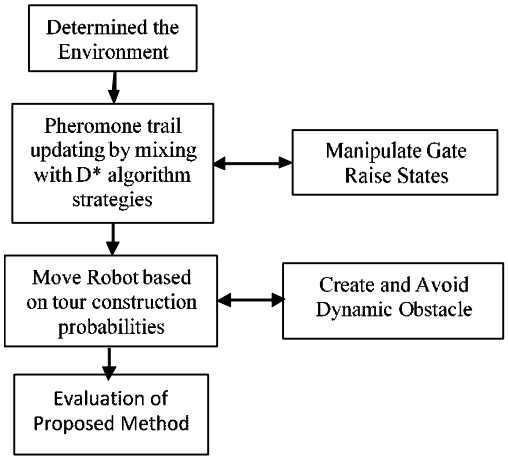

Fig. 1: The structure of improved method to find robot path planning.

strategies Pheromone trail updating maintains propagated information about the intensity of the pheromone trails values $\tau_{ij}(t)$ by mixing with D* algorithm strategy which calculate path costs to the states in the space. The propagation takes place through the repeated pheromone deposition. It is expanded to modify pheromone trails values $\tau_{ij}(t)$ to its neighbors after each iteration. These neighbors are in turn added ants to continue propagation and to modify pheromone trails values $\tau_{ij}(t)$ until the start state (nest) is found. Pseudocode procedure PheromoneTrailUpdating illustrated in Alogrthim is built in number of steps to find pheromone trail updating Eq. 2. The first step is done by selecting the initial value of evaporation rate ñ equal to 0.98, then goal state (food location) is put in the queue add ants propagation (Insert procedure) as a current state (location). The proposal pseudocode procedure to construct pheromone trail updating.

```
Procedure PheromoneTrailUpdating ( )
Begin
Goal_i = goal_pos_x;
Goal_j = goal_pos_y;
ρ = 0.98; // select the initial value of ?
goal_pher=0.0;
Insert(OpenList, goal); // add ant to queue propagation
repeat
Begin
node X = DequeueOpenList (); //moving ant to propagation
X_tag = "Close"
// expanded by finding the eight neighbor locations Y of current position X
      FindNeighbor (Y,X)
 for each Neighbor Y of X do
 begin
 PheromoneTrails (X ,Y, OpenList)
     GateRaiseState (X)
      SortQueue (OpenList)
  end
until (X_status = "Start")
end
```

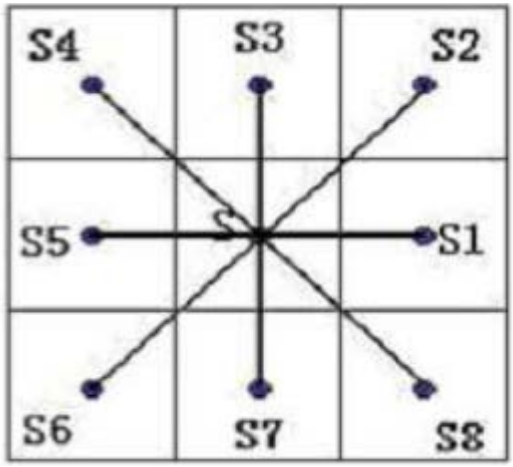

Fig. 2: The 8 direction search of the ant.

In the second step, the current state (location) is expanded by finding the eight neighbor states (locations) (FindNeighbor prcedure). Figure 2 shows the discrete set of possible connections of the 8 directions neighbor nodes in the graph taken from a uniform resolution 2D grid: Up, Down, Left, Right, Up-Left, Up-Right, Down-Left and Down-Right .

In the third step, the intensity of the pheromone trails value τ_{min} and the heuristic information τ_{min} are computed (PheromoneTrails Pseud ocode procedure is shown in Fig. 3 for each state on neighborhood. The neighborhood are checked if the status state is Obstacle, it is not given pheromone trails values τ_{ij} (t)(like D* given high value 10000). If the status state is clear, then its position is checked if ant move on diagonal of current state (location) the τ_{ij}(t) values is sum by 1.4 or it is sum by 1 in the case of horizontal or vertical position. Then it's multiplied each case by evaporation rate ñ. Also the value of heuristic information τ_{min} is found by the distant between any neighbor locations to the goal position (food).

The last step manipulates the Gate Raise States by procedure GateRaiseState which is used to check if two diagonal vertex neighbor locations of current position of ant are Obstacles status, then ant can not move to it because its size not passed through. Hence, current state is a gate raise state and it's closed by changing its status type to obstacle. The pseudocode procedure Pheromone Trails is computed the values of pheromone trails and the heuristic information.

```
Procedure PheromoneTrails (X, Y, OpenList)
Begin
 if (Y_tag = = "NEW") then
if (Y_status = "Obstacle" ) then
 Y_pher =10000;// ant is not given pheromone trails values represent a very
large value
 else if (Y_neighbor_i ?X_iold) and (Y_neighbor_j ? X_jold) then
  Y_ cost = X_ cost + 1.4;
 Y_ pher = Y_ cost * ñ ;
 else
  Y_ cost = X_ cost + 1.0;
 Y_ pher = Y_ cost * ñ ;
 endif
Y_her = sqr((Y_neighbor_i - goal_pos_x)^2 + (Y_neighbor_j -
goal_pos_y)^2);
 endif
 Insert(OpenList, Y_neighbor);
 endif
end
```

Moving robot based on tour construction probabilities: In this section, the robot (ant) moves to next location at each iteration by finding the best solution depending on tour construction probabilities. It travels through free cell (location) and it avoids the dynamic obstacle until it reaches the goal state. Procedure MoveRobotTourConst present in Alogrthim a number of steps to find tour construction. In the beginning the start state (nest) is selected as a current node (position) for moving the ants, giving the weigh values α and β which have the importance of the pheromone and heuristic values. Then a procedure CrateDynObs is used to select randomly a position in a free space environment to create a dynamic obstacle. The robot (ant) then must move depending on a probabilistic decision to next location in the neighborhood which has the best solution and convergence is repeated at each iteration (iteration-best ant) until the robot reaches the goal position (food). It uses the procedure FindNeighbor to find the eight neighborhood nodes (locations) to move from current node (position). Procedure UpdateDynObs presents the moving dynamic obstacle in a random position of a neighbor clear space environment.

After that the tour construction probabilities are computed by Eq. 1 which depends on the pheromone trail τ_{min} and on a locally available heuristic information θ_{ij} for all locations (nodes) of neighbor. The decision of best solution to ant that has minimum probability compared to other ants in neighbor locations. The neighbor locations (nodes) are testing to not contain a static or dynamic obstacle before computed it. This best ant which has the best solution is moving to the new position to be a current node and convergence is repeated until it reaches the goal position (food). At last, the result of optimal path is shown as the path displays on the selected dynamic environment as a green color, No. of iteration (locations) of path, total path costs and time occupy to find a path.The pseudocode procedure MoveRobotTourConst is construct for moving the ants to find the optimal path based on tour construction probabilities.

```
Procedure MoveRobotTourConst ( )
begin
 // Starting with start node to find the optimal path planning as current node
X
```

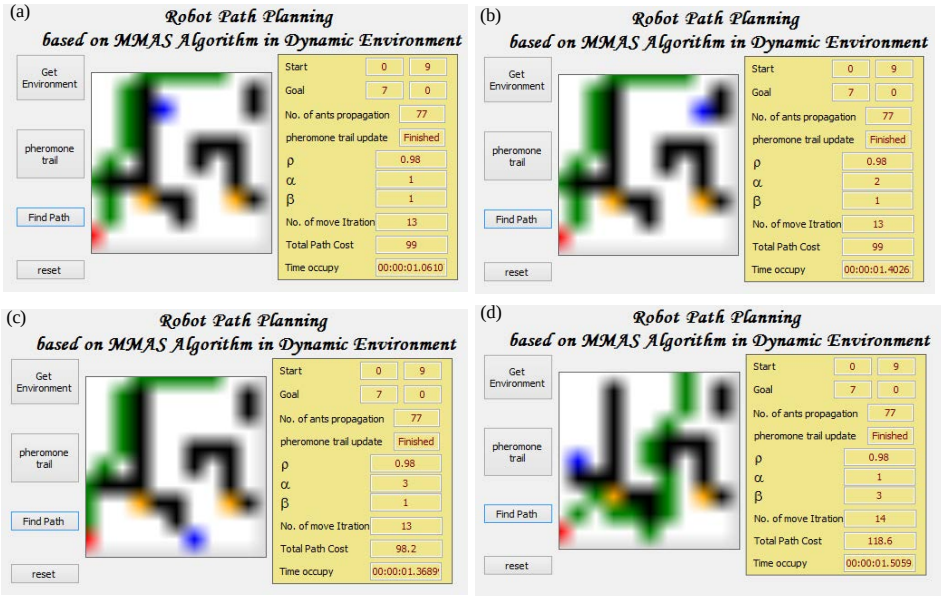

Fig. 3: Optimal path planning using improve MMSA algorithm

```
X_curri = start_pos_x
X_currj = start_pos_y
// given an initials weigh values of α and β
α =3;
β =1;
// find the eight neighborhood nodes Y of current node X.
FindNeighbor (Y,X)
// create the random position of dynamic obstacle.
CrateDynObs(X1, temporal_k )
repeat
begin
//updating randomly neighbor position of dynamic obstacle.
UpdateDynObs (X1, temporal_k )
// calculated the values of tour construction probabilities for each neighbor locations
for each neighbor Y of X do
if (Y_status = "Obstacle" or Y_status = "Dobstacle" ) then
P[i] = ( (state[Y].pherm)^α *(state[Y].hur) ^β) / ?((state[Y].pherm) ^α *(state[Y].hur) ^β );
// find the minimum best value by sorting it and given its position
for each item of (P[]) do
 if (P[] < min_best_pphr) then
begin
min_ min_best_pphr = P[]
mini_ pphr = neighbori[]
minj_ pphr = neighborj[]
end
X_corri = mini_ pphr;
X_corrj = minj_ pphr;
iter_no = iter_no + 1;
X_SetPixel (Green);
end
until (current_position = goal_position or iter_no = max_itr)
end
```

Simulation: Some simulations are present in variant cases to validate the feasibility of the new method. This variety different size of dynamic environments, numbers and shapes of static obstacles, number of gate raise states closeted before found an optimal path, position of dynamic obstacle selected randomly in valid (free) space of environment, and in the continually moving dynamic obstacle in random variant neighbor positions until the robot reaches the goal state. This simulation results are implemented in C# Microsoft Visual Studio 2010 on Pentium 4 PC. We implemented the improve MMSA algorithm on same dynamic environment. To illustrate results choose small environment (10*10) pixel size and select the start state at position (0,9) and goal state at position (7,0) as an illustrated example.

An Improved MMSA algorithm implemented: At first, the improve MMSA algorithm is implemented to find the local path planning which is shown as a green color. There are different cases that are tested to find and study the results of implementation with the same value of evaporation rate ρ that equals 0.98 and in random location of dynamic obstacle of blue color. The first testing case is implemented by giving the same value equal to1 for both weigh of pheromone α and weigh of heuristicβ. The results illustrated in Fig. 3a show No. of iteration equals13, the total path cost equals 99 and a time occupy equals 1.06 sec.. The second test presented in Fig. 3b changes the value of α to 2 only, it leads to the same results of first test except time occupy is equal to 1.40 sec.. In the third test, the change of value α to 3 finds the best results shown in Fig. 3c and gives a smooth and more optimal local path with total path cost = 98.2 and time occupy is 1.36 sec. The fourth test is implemented by giving a weigh value of pheromone α = 1 which is less than the weigh value of heuristic β = 3, the path changes to another locations as shown in Fig. 3d and the results of the local path are No. of iterations that equals 14, the total path cost equals 118.6 and a time occupy equals 1.5 sec.

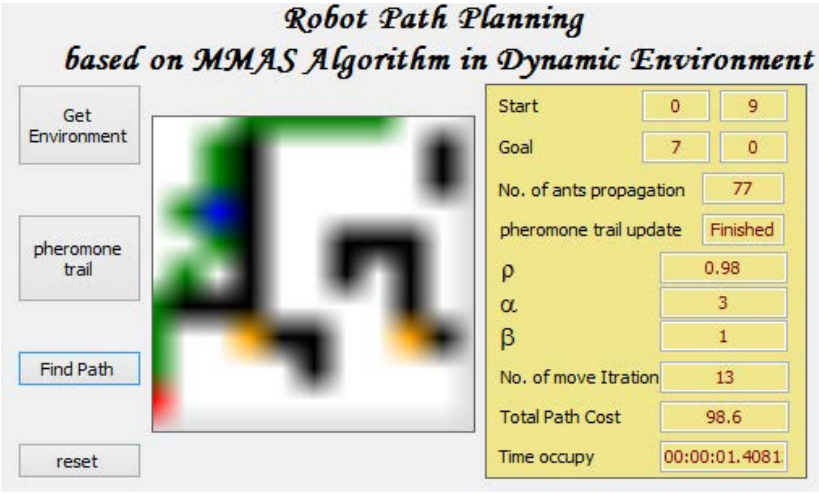

Fig. 4: Finding local path by improve MMSA algorithm when avoid dynamic obstacle

Avoid dynamic obstacle: The local path is found in the same environment and exploits the same method used above except that the dynamic obstacle is intersection with robot in the same position. In this case the robot (ant) avoids collision with dynamic obstacle and moves away in best neighbor location of ant that has minimum probability. Fig. 4 illustrates that the total path cost changes to 98.6, a time occupy equals 1.4 sec., the local path shown in green color is different from the above path shown in Fig. 3c without intersection dynamic obstacle.

RESULTS AND DISCUSSON

Experiment results: Finally the improve MMSA algorithm is implemented to find the local path planning on five different complex dynamic environments size 50*50 cells. The experiment simulation result are found by giving the same value of evaporation rateρ that equals 0.98, choosing a different weigh values of pheromone α and heuristicβ and also selecting different positions of start and goal as show in Fig. 5. The experiment simulation result illustrated in Fig. 5a is implemented on a selected environment that contains the static obstacles which are organized in curve shapes and some of these curve block obstacles are open from different directions. The positions of start and goal are selected to lie on both sides of long curve sequence of static obstacles that is forced to choose a large weigh value of $\alpha = 7$ to get the converge solution to find the local robotics path. Figure. 5b is implemented on environment contain the similar organization of curve shapes static obstacles overlapping each other. The positions of start and goal are chosen to be adapt from each other and outside of curves. The improve MMSA algorithm finds the convergence solution and local robotics path by giving the value of α equal 5 and value of $\beta = 1$. The improve MMSA algorithm is also implemented on simple environment contain vertical sequence of obstacles and big area of free space. It reaches the convergence solution in small value of $\alpha = 1$ and value of $\beta = 1$ and it finds local robotics path as illustrated in Fig. 5c.

The next experiment simulation shown in Fig. 5d is implemented on environment that has the static obstacles which are established in circular sequence shapes and some of circular block obstacles are open from different positions. The improve MMSA algorithm is forced to get the convergence solution at a value of $\alpha = 5$ and value of $\beta = 1$ and find the local robotics path. The last two experiments simulation are implemented on same dynamic environment that contains large numbers of static obstacles which have different shapes and sizes and lies randomly in environment. In the middle it connects a horizontal sequence of obstacles that it closes in left side and open from the right side. The first experiment simulation result illustrated in Fig. 5e is implemented by choosing the positions of start and goal that lie on upper left corner and lower left corner respectively. The improve MMSA algorithm reaches the convergence solution in a value of $\alpha = 3$ and value of $\beta = 1$ and finds local robotics path. The second experiment simulation result is implemented on the same environment but the positions of start and goal are selected to lie on both sides of the middle horizontal sequence of obstacles. The improve MMSA algorithm cannot reach the convergence solution

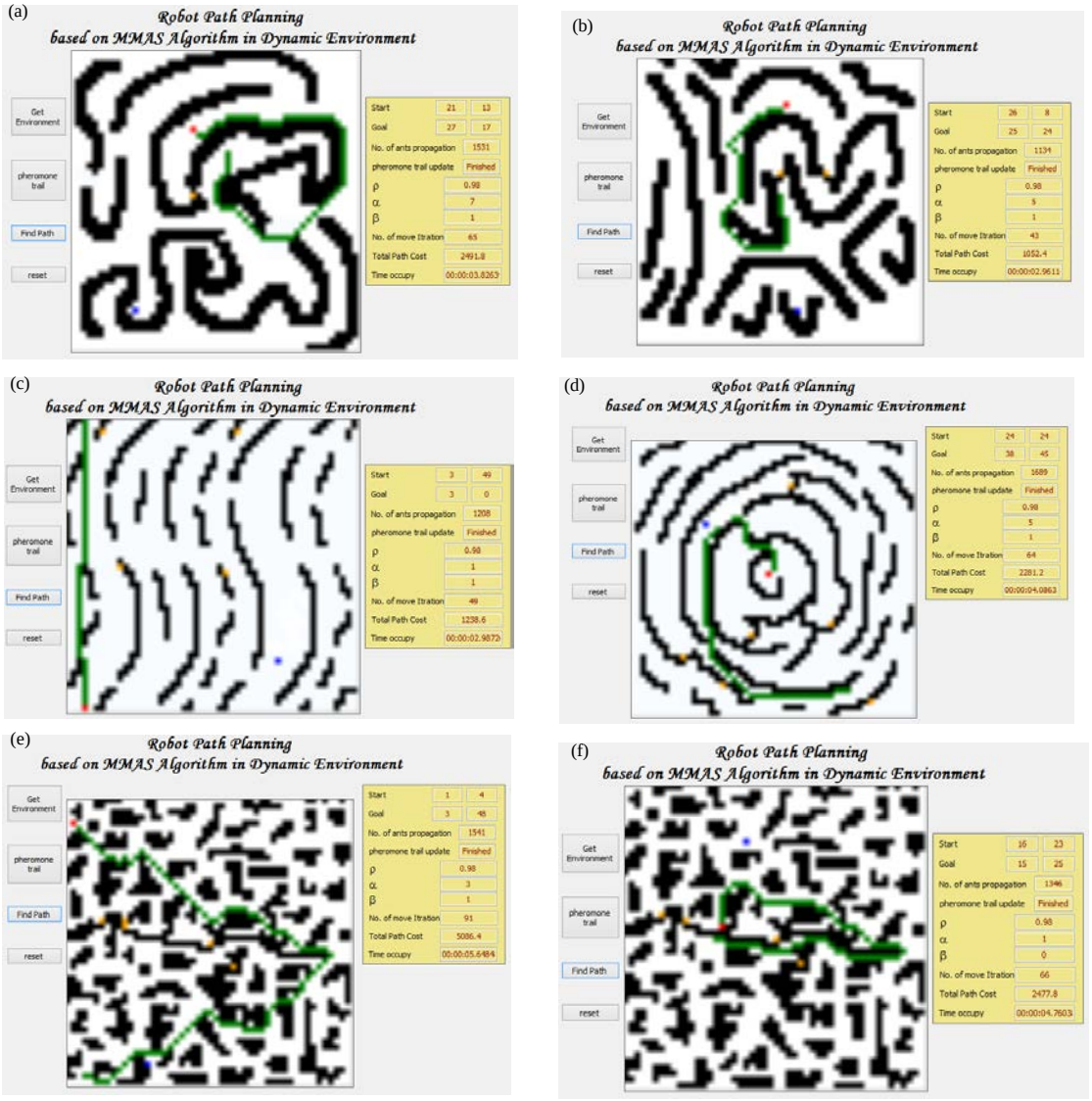

Fig. 5: Finding the local robotic path for six selected implementation on different five complex dynamic environments

Table 1: comparison of experiment simulation results are implemented in Fig. 8 with basic D* algorithm.

Env No	Fig. 5 part	Algorithm type	Start state	Goal state	No. of gate	α	β	No of ants propagation	No. of Iteration	Total path cost	Time occupy sec
1	a	Improve MMSA	21.13	27.17	1	7	1	1531	65	2491.8	3.82
1		Basic D*	21.13	27.17	1					3318.4	3.90
2	b	Improve MMSA	26.80	25.24	2	5	1	1134	43	1052.4	2.96
2		Basic D*	26.80	25.24	2					2559.8	10.06
3	c	Improve MMSA	3.49	3.00	0	1	1	1208	49	1238.6	2.98
2		Basic D*	3.49	3.00	0					1238.6	3.49
4	d	Improve MMSA	24.24	38.45	2	5	1	1689	64	2281.2	4.08
4		Basic D*	24.24	38.45	2					5857.8	11.23
5	e	Improve MMSA	1.40	2.48	3	3	1	1541	91	5086.4	5.64
5		Basic D*	1.40	2.48	3					5380.6	32.89
5	f	Improve MMSA	16.23	15.25	3	1	0	1346	66	2477.8	4.76
5		Basic D*	16.23	15.25	3					3249.2	10.04

until it gives the value of $\alpha = 26$ with the value of heuristic $\beta = 1$. But when the heuristic information is ignored by giving value of $\beta = 0$, the improve MMSA algorithm reaches the convergence solution in a small value of $\alpha = 1$ and finds the local robotic path. Table (1) illustrates the overall results of implementation shown in Fig. 5

compared to the results generated from the basic D* algorithm on the same dynamic environment and the same start and goal positions. The table also contains more requirements and results of improved MMSA algorithm like a given values of α andβ, No. of ants propagation, a number of the ant moving iterations, time occupy in sec. and the total path cost.

CONCLUSION

This researcher presents a new method for on-line robotic path planning in dynamic environment. It finds the local and efficient path in variant complex environment maps. The improved MMAS algorithm can achieve the best performance by a combination of optimization algorithms to improve the solutions generated by the ants with search algorithms. The improve MMSA algorithm reaches the convergence solution depending on balance between the values of α and β with the positions of start (nest) and goal (food) and the numbers of static obstacles that have different shapes and sizes and lie randomly on environment to find local robotic (ant) path.

The comparison results proved that the new fusion improvement is more efficient in time occupy and total path cost than basic D* algorithm. The experiments simulation result from the new method show that the robot (ant) avoids collision with dynamic obstacle and moves away in best neighbor location of ant that has minimum probability. Also the experimental results find the optimal local path with minimum iterations, minimum total path cost and minimum time occupy depending on the complexity of dynamic environment which is based on the number of obstacles or the free space the robot passes through moving the ant (robot) and number of cross dynamic obstacles through moving it.

REFERENCES

Abdelbar, A.M. and D.C. Wunsch, 2012. Improving the performance of MAX-MIN ant system on the TSP using stubborn ants. Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation, July 7-11, 2012, ACM, New York, USA., ISBN:978-1-4503-1178-6, pp: 1395-1396.

Brand, M., M. Masuda, N. Wehner and X.H. Yu, 2010. Ant colony optimization algorithm for robot path planning. Proceedings of the 2010 International Conference on Computer Design and Applications (ICCDA), June 25-27, 2010, IEEE, New York, USA., ISBN:978-1-4244-7164-5, pp: 436-440.

Dorigo, M. and T. Stutzle, 2004. Ant Colony Optimization. MIT Press, Cambridge, MA., USA., ISBN-13: 9780262042192, Pages: 305.

Ferguson, D. and A. Stentz, 2006. Using interpolation to improve path planning: The field D* Algorithm. J. Field Robotics, 23: 79-101.

Koenig, S. and M. Likhachev, 2002. Improved fast replanning for robot navigation in unknown terrain. Proceedings of the IEEE International Conference on Robotics and Automation, May 11-15, 2002, IEEE, New York, USA., ISBN:0-7803-7272-7, pp: 968-975.

Mei, H., Y. Tian and L. Zu, 2006. A hybrid ant colony optimization algorithm for path planning of robot in dynamic environment. Intl. J. Inf. Technol., 12: 78-88.

Raja, P. and S. Pugazhenthi, 2012. Optimal path planning of mobile robots: A review. Intl. J. Phys. Sci., 7: 1314-1320.

Reshamwala, A. and D.P. Vinchurkar, 2013. Robot path planning using an ant colony optimization approach: A survey. Int. J. Adv. Res. Artif. Intell., 2: 65-71.

Saeheaw, T., N. Charoenchai and W. Chattinnawat, 2009. Application of ant colony optimization for multi-objective production problems. World Acad. Sci. Eng. Technol., 60: 654-659.

Stentz, A., 1994. Optimal and efficient path planning for partially-known environments. Proceedings of the IEEE International Conference on Robotics and Automation, May 8-13, 1994, Carnegie Mellon Robotics Institute, USA., pp: 3310-3317.

Stutzle, T. and H.H. Hoos, 2000. Max-min ant system. Future Generation Comput. Syst., 16: 889-914.

Sun, Y. and J.W. Tian, 2010. WSN path optimization based on fusion of improved ant colony algorithm and genetic algorithm. J. Comput. Inf. Syst., 6: 1591-1599.