International Journal of Computer Science and Engineering Research and Development (IJCSERD), ISSN 2248-9363 (Print), ISSN 2248-9371 (Online), Volume 1, Number 1, April-June (2011)
pp. 38- 49 © PRJ Publication
http://www.prjpublication.com/IJCSERD.asp

IJCSERD

© PRJ PUBLICATION

MULTI ROBOT PATH PLANNING ALGORITHMS: A SURVEY

S Vaniya[1], B Solanki[1], S Gupte[1]

[1] Department Information Technology, Gujarat Technological University
A.D.Patel Institute of Technology,
New Vallabh Vidyanagar, Anand, Gujarat, India
[1]s.vaniya@gmail.com
[1]ultybhavesh@gmail.com
[1]supangupte@gmail.com

ABSTRACT

To find the optimal path by interacting with multiple robots is the main research area in field of robotics. The task is to find the global optimal path with a minimum amount of computation time. Path planning has numerous application like industrial robotics, to design autonomous system etc. In this paper, we survey on three most recent algorithms namely Bacteria forging Optimization (BFO), Ant Colony Optimization (ACO), Particle Swam Optimization (PSO) that can apply on multiple robots to find the optimal path. The main feature of BFO is the chemotactic movement of a virtual bacterium that is helpful to investigate the all the possible path and finally arrived at optimal solution. In Ant Colony System (ACS) algorithm is integration of heuristic and visibility equation of state transition rules for finding the optimal path. PSO is stochastic optimization technique inspired by social behaviour of bird flocking. The solutions, called particles, fly through the problem space by some set of the rules.

Keywords: Bacteria forging Optimization (BFO), Ant Colony Optimization (ACO), Particle Swam Optimization (PSO), Ant Colony System (ACS), Robot path planning (RPP),Genetic algorithms (GA)

I. INTRODUCTION

Path can be defined as- "is a map of geometric locus of all the points in a given space where the robot has to travel." Path planning is to formulate the path that a robot must take in order to pass over each point in an environment.

Robot Path Planning is to create a collision free path in an environment while satisfying some optimization criteria. With the perfect path planning system, mobile robot can navigate by itself without human interference to reach the targeted destination. Path planning can be divided into two categories which are global [1] and local [2]. If the knowledge of the environment is known, the global path can be planned offline before the robot starts to move. This global path can facilitate the robot to traverse within real environment because the feasible optimal path has been less constructed within the environment. Another category, local path planning system introduces to solve the RPP problem when it encounters an obstacle. This local path is usually assembled online when the robots avoid the obstacles in a real time environment. Path planning covers a wide area of robotic navigation in both static and dynamic environment. Several

researches have lead to development of better path optimization solution with comparatively less computational time and path cost.

There are basically two main components that can be suggested for global or calculated path planning are; 1) robot representation of the world in configuration space(C-space) & 2) the algorithm implementation. These two components are interconnected and greatly dependent on one another in the manner to determine minimal path for robot traversing.

The main focus of the paper is to present a distributed PSO approach, a BFO approach and to finish with ACS algorithm for RPP. The paper is categorized as follows: Section 2 contains reviews of the PSO algorithm, section 3 reviews the BFO algorithm, section 4 gives reviews ACS, and section 5 concludes the survey.

II. CLASSIC PARTICLE SWARM OPTIMIZATION (PSO) ALGORITHM

Particle swarm optimization is also a bio-mimetic optimization technique developed by Eberhart and Kennedy in 1995 which was stimulated by social behaviour of the bird flocking and fish schooling. The critical concept of PSO, at each time step consists of changing the velocity of each particle towards its best location. In past several years PSO has been successfully applied in many application and research areas [3].

In PSO the entire population of particle is maintain throughout the exploration operation as there are no cross over and mutation operators. Each particle moves around some multidimensional search space with a pre determined velocity and position that is updated based on the particle's experience and that of the swarm (the collection of particles).information sharing may or may not occur between the particles or entire swarm during the iteration of the algorithm. This results in effectively influencing other particles' search of an area.

The basic pseudo code for the classic PSO algorithm taken from [4][5] is:

Step 1: Initialize population with random positions and
Velocities
Step 2: Evaluate – compute fitness of each particle in
Swarm
Step 3: Do for each particle in swarm {
Step 3.1: Find particle best (pbest) – compute fitness of the particle.
If current pbest < pbest
Pbest = current pbest
Pbest location = current location
Endif
Step 3.2: Find global best (gbest) – best fitness of all
pbest
Gbest location = location of min (all pbest)
Step 3.3: Update velocity of particle per equation 2
Step 3.4: Update position of particle per equation 3}
Step 4: Repeat steps 3.1 through 3.4 until termination condition is reached.

A population of particles is formed and then evaluated to compute their fitness and find the particle and global best from the population/swarm. The particle best, *pbest*, stands for the best fitness value for a particle up to that moment in time, whereas the global best represents the best particle in the entire swarm. A global best, *gbest*, or a local best, *lbest*, are used for a single neighborhood or multiple local neighborhoods, respectively. When multiple local neighborhoods are used each particle stays within their local neighborhood for all iterations of the program. PSO algorithm may use a fitness function based on the Euclidean distance from a particle to a target. The coordinates of the particle are *px* and *py*, while the coordinates of the target are *tx* and *ty* as shown in equation 1.

$$Fitness = \sqrt{(tx - px)^2 + (ty - py)^2} \quad (1)$$

The velocity of a particle is computed as follows [6]:

$$vn + 1 = w_i v_n + c1 \times r1 \times (pbest_n - p_n) + c2 \times r2 \times (gbest_n - p_n) \quad (2)$$

where parameters used are:

w_i= inertia coefficient

v_n= current particle velocity

$r1$=uniformly distributed random number [0:1]

$r2$= uniformly distributed random number [0:1]

$c1$, $c2$=acceleration constants

The position of the particle is based on its previous position ,p_n, and its velocity over a unit of time:

$$p_{n+1} = p_n + v_{n+1} \quad (3)$$

Each particle is accelerated towards its pbest and gbest locations using acceleration values c1 and c2, respectively, at each time step. Typical values for parameters are w_i=0.9, and c1=c2=1 [7][8]. Maximum velocities for some small robots noted in the literature are 20 cm/sec [9], 100 cm/sec [10], and 1 m/sec [11]. A large w_i favors global search while a small w_i favors local search. The search process is terminated when a predetermined number of iterations are reached, or the minimum error requirement is reached for all particles.

III. THE BACTERIA FORAGING OPTIMIZATION ALGORITHM

Bacteria foraging optimization algorithm (BFOA) proposed by Passino [12] is a new corner to the family of nature inspired optimization algorithms. BFOA is inspired by the social foraging behaviour of Escherichia Coli. A bacterium takes foraging decision considering two factors; first, communicating with others by sending signals and secondly, searches for nutrients in a manner to maximize energy obtained per unit time.

The process in which a bacterium moves by taking small steps while searching for nutrients is called chemotaxis and the key idea of BFOA is mimicking chemotic movement of virtual bacteria in the problem search space. On getting sufficient food, they increase in length. They form exact replica of themselves by breaking from the middle in presence of suitable temperature. The chemotatic progress may get damaged due to the occurrence of sudden environmental changes.

Thus a group of bacteria may shift to some other place or some other may come into the swarm of concern. This leads to the event of elimination-dispersal in the bacterial population, where all the bacteria in a region are killed or a group is dispersed into a new division of the environment.

Now suppose that we want to find the minimum of $J(\theta)$ where $\theta \in \mathbb{R}^p$(i.e. $\theta$is a p-dimensional vector of real numbers),and we do not have measurements or an analytical description of the gradient $\nabla J(\theta)$. BFOA mimics the four principal mechanisms observed in a real bacterial system: chemotaxis, swarming, reproduction, and elimination-dispersal to solve this non-gradient optimization problem. A virtual bacterium is actually one trial solution (also called a search-agent) that moves on the functional surface (see Figure 1 [13]) to locate the global optimum.

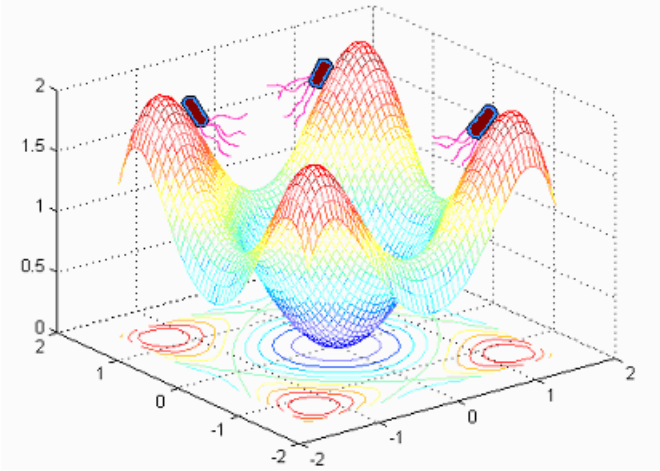

Figure 1: A bacterial swarm on a multi-modal objective function surface [13]

Let us define a chemotactic step to be a tumble followed by a tumble or a tumble followed by a run. Let j be the index for the chemotactic step. Let k be the index for the reproduction step. Let l be the index of the elimination-dispersal event. Also let

p : Dimension of the search space,
S: Total number of bacteria in the population,
Nc : The number of chemotactic steps,
Ns: The swimming length.
Nre : The number of reproduction steps,
Ned : The number of elimination-dispersal events,
Ped : Elimination-dispersal probability,
$C(i)$: The size of the step taken in the random direction specified by the tumble.

Let $P(j,k,l) = \{\theta^i(j,k,l)|i = 1,2,\dots,S\}$represent the position of each member in the population of the S bacteria at the j-th chemotactic step, k-th reproduction step, and l-th elimination-dispersal event. Here, let $J(i, j, k, l)$ denote the cost at the location of the i-th bacterium $\theta^i(j,k,l) \in \mathbb{R}^p$(sometimes we drop the indices and refer to the i-th bacterium position as θ^i). Note that we will interchangeably refer to J as being a "cost" (using terminology from optimization theory) and as being a nutrient surface (in reference to the biological connections). For actual bacterial populations, S can be very large (e.g., S =109), but p = 3. Below we briefly describe the four prime steps in BFOA.

i) Chemotaxis: This process simulates the movement of an *E-coli* cell through swimming and tumbling via flagella. Biologically an *E-coli* bacterium can move in two different ways. It can

swim for a period of time in the same direction or it may tumble, and alternate between these two modes of operation for the entire lifetime. Suppose $\theta^i(j,k,l)$ represents *i*-th bacterium at *j*th chemotactic, *k*-th reproductive and *l*-th elimination-dispersal step. *C*(i) is the size of the step taken in the random direction specified by the tumble (run length unit). Then in computational chemotaxis the movement of the bacterium may be represented by:

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}} \qquad (1)$$

Where Δ indicates a vector in the random direction whose elements lie in [-1, 1].

ii) Swarming: An interesting group behavior has been observed for several motile species of bacteria including *E-coli* and *S- typhimurium*, where intricate and stable spatio-temporal patterns are formed in semisolid nutrient medium. A group of *E-coli* cells arrange themselves in a traveling ring by moving up the nutrient gradient when placed amidst a semisolid matrix with a single nutrient chemo effecter. The cell-to-cell signaling in *E-coli* swarm may be represented by the following function.

$$J_{cc}\big(\theta, P(j,k,l)\big) = \sum_{i=1}^{S} J_{cc}^{i}\left(\theta, \theta^i(j,k,l)\right)$$

$$= \sum_{i=1}^{S}\left[-d_{attract}\exp\left(-w_{attract}\sum_{m=1}^{p}\left(\theta_m - \theta_m^i\right)^2\right)\right]$$

$$+\sum_{i=1}^{S}\left[-h_{repellant}\ \exp\left(-w_{repellant}\sum_{m=1}^{p}\left(\theta_{m-}\theta_m^i\right)^2\right)\right] \quad (2)$$

Where $J_{cc}\big(\theta, P(j,k,l)\big)$is the objective function value to be added to the actual objective function (to be minimized) to present a time varying objective function. *S* is the total number of bacteria, *p* is the number of variables to be optimized, which are present in each bacterium and $\theta = [\theta_1, \theta_{2,\ldots\theta_p}]^T$is a point in the *p*-dimensional search domain. $d_{attractant}, w_{attractant}, h_{repellant}, w_{repellant}$ are different coefficients to be chosen properly.

iii) Reproduction: The least healthy bacteria eventually die where as the healthier bacteria, yielding lower value of the objective function asexually split into two bacteria, which are then placed in the same location, keeping the swarm size constant.

iv) Elimination and Dispersal: Gradual or sudden changes in the local environment where a bacterium population lives may occur due to various reasons e.g. a significant local rise of temperature may kill a group of bacteria that are currently in a region with a high concentration of nutrient gradients. All the bacteria in a region are killed or a group may disperse to a new location. To simulate this phenomenon in BFOA some bacteria are settled at random with a very small probability while the new replacements are randomly initialized over the search space.

The pseudo-code of the complete algorithm is presented below:

The BFOA Algorithm

Parameters:

Step 1 Initialize parameters *p, S, Nc, Ns, Nre, Ned, Ped, C(i),* $i = 1,2,\dots,S$, θ^i

Algorithm:
Step 2 Elimination-dispersal loop: *l*=*l*+1
Step 3 Reproduction loop: *k*=*k*+1
Step 4 Chemotaxis loop: *j*=*j*+1
[a] For $i = 1,2,\dots,S$ take a chemotactic step for bacterium *i* as follows.
[b] Compute fitness function, $J(i,j,k,l)$.
Let, $J(i,j,k,l) = J(i,j,k,l) + J_{cc}(\theta^i(j,k,l), P(j,k,l))$
(i.e. add on the cell-to cell attractant–repellant profile to simulate the swarming behavior) where, *Jcc* is defined in (2).

[c] Let J_{last}=*J (i, j, k, l)* to save this value since we may find a better cost via a run.

[d] Tumble: generate a random vector

$\Delta(i) \in R^p$ with each element $\Delta_m(i)$,m=1,2,...,p, a random number on [-1, 1].

[e] Move: Let

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}}$$

This results in a step of size C(i) in the direction of the tumble for bacterium i.

[f] Compute $J(i,j+1,k,l)$ and let

$$J(i,j+1,k,l) = J(i,j,k,l) + J_{cc}(\theta^i(j+1,k,l), P(j+1,k,l))$$

[g] Swim

i) Let *m*=0 (counter for swim length).

ii) While m < N_s (if have not climbed down too long).

• Let *m*=*m*+*1*.

• If $J(i,j+1,k,l) < J_{last}$ (if doing better), let $J_{last} = J(i,j+1,k,l)$ and let

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}}$$

And use this $\theta^i(j+1,k,l)$to compute the new $J(i,j+1,k,l)$ as we did in [f]

• Else, let *m*=N_s. This is the end of the while statement.

[h] Go to next bacterium (i+1) if $i \neq S$ (i.e., go to [b] to process the next bacterium).

Step 5 If $j < N_c$, go to step 4. In this case continue chemotaxis since the life of the bacteria is not over.

Step 6 Reproduction:

[a] For the given k and l, and for each $i = 1,2,\ldots,S$, let

$$J_{health}^{i} = \sum_{j=1}^{N_c+1} J(i,j,k,l) \qquad (3)$$

be the health of the bacterium i (a measure of how many nutrients it got over its lifetime and how successful it was at avoiding noxious substances). Sort bacteria and chemotactic parameters $C(i)$ in order of ascending cost J_{health} (higher cost means lower health).

[b] The S_r bacteria with the highest J_{health} values die and the remaining S_r bacteria with the best values split (this process is performed by the copies that are made are placed at the same location as their parent).

Step 7 If $< N_{re}$, go to step 3. In this case, we have not reached the number of specified reproduction steps, so we start the next generation of the chemotactic loop.

Step 8 Elimination-dispersal: For $i = 1,2,\ldots,S$ with probabilityP_{ed}, eliminate and disperse each bacterium (this keeps the number of bacteria in the population constant). To do this, if a bacterium is eliminated, simply disperse another one to a random location on the optimization domain. If $l < N_{ed}$, then go to step 2; otherwise end.

IV. ANT COLONY SYSTEM ALGORITHM

Ant algorithms are multi-agent systems that exploit artificial stigmergy as a means for coordinating artificial ants for the solution of computational problems. The flow chart and the algorithm on the basis of the Figure 2 [14] is given below:

Step 1: Starting from the start node located at x-y coordinate of (1, 1), the ant will start to move from one node to other feasible adjacent nodes during the map construction as shown in Figure 3. As depicted, there are three possible movements available for the ants to move from the start node (1,1). The possible nodes are: node 1 located at the x-y coordinate of (1,3), node 2 at (2,3) and node 3 at (3,2).

Step 2: The ant will then take the next step to move randomly based on the probability given by equation (1) below:

$$Probability\ ij(t) = Heuristic\ ij(t) \times Pheromone\ ij(t)$$

$$= \left[\left(1/_{\substack{distance\ between\ vector\ start\ point\ to\ the\ next \\ point\ and\ start\ point\ to\ reference\ line\ to\ goal}}\right)^{\beta} \times \left({}^{trail}/_{\sum trail}\right)^{\sigma}\right] \quad (1)$$

where Heuristic ij(t) indicates every possible adjacent nodes to be traversed by the ant in its grid position at every t time. The quantity of Pheromone ij(t) is an accumulated pheromone, between nodes when the ant traverses at every time. Therefore, the probability equation depends on both values which guide the ants to choose every possible node in every t time.

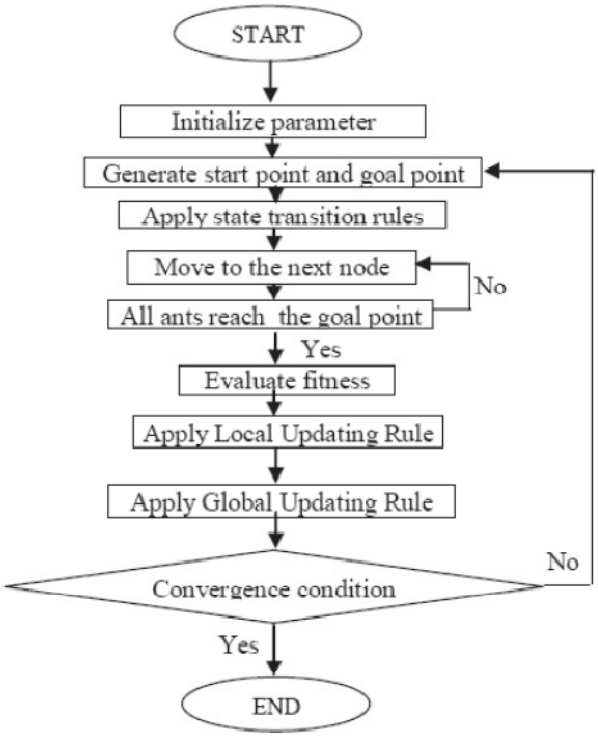

Figure 2: Outline of ACO for RPP of a mobile robot [14]

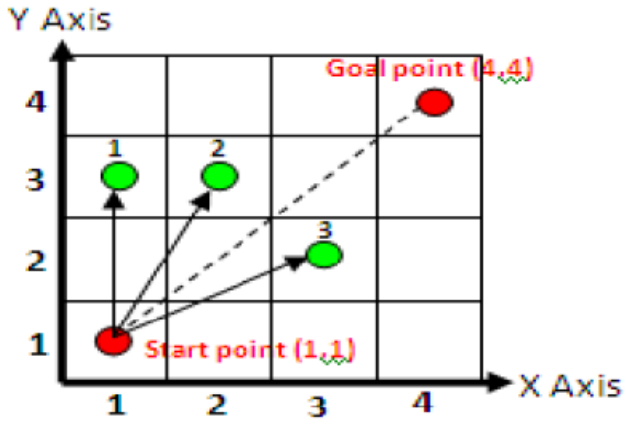

Figure 3: Ant current position (red nodes) with 3 possible next positions (green nodes) [14]

A. Derivation of heuristic@ visibility equation

The heuristic equation was derived from the idea of the local and global search solution for the RPP purposes. This represents the distance between selected adjacent nodes with intersected point at reference line where the intersected point must be perpendicular to the next node (Distance A) as shown in Figure 4.The distance between each node that have the shortest distance should have a higher probability to be chosen compared to the longer distance. To ensure this, the derivation of heuristic proposed must be inversed with the distance found as shown in equation (2) below:

Heuristic=[1/distance between next point with intersect point at reference line]$^{\beta}$ (2)

where β=heuristic coefficient

Figure 4 depicts the proposed solution of using the Pythagoras Theorem to calculate the distance A. θ here refers to the angle of vector of intersect point to next visited node with line of intersect point to goal position while Distance B refers to the distance from intersect point at reference line to the next visited node. Intersect point is the point obtained when the line of the next node which is parallel with the x-axis intersects with the reference line. Thus, the equation for distance A is equal to:

$$Distance\ A = sin\,\theta \times \sqrt{[(y2-y1)^2 + (x2-x1)^2]} \qquad (3)$$

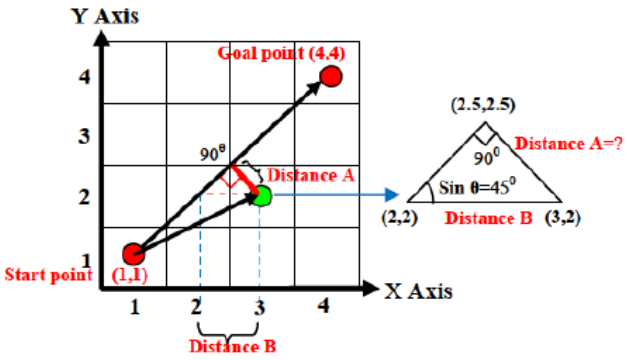

Figure 4: Derivation of Distance A [14]

As depicted in Figure 4, the heuristic calculation can be calculated as shown below:

$$Distance\ A = sin\,\theta \times \sqrt{[(y2-y1)^2 + (x2-x1)^2]}$$

$$= sin\,450 \times \sqrt{[(2-2)^2 + (3-2)^2]}$$

$$= 0.7071 \times 1 = 0.7071$$

$$Visibility = [1/0.7071]1 = 1.4142$$

B. Derivation of trail equation

The artificial ants behave similar to the real ants where the pheromone will attract the next ants to follow the shortest path until the process converges when all ants follow the same path to the target.

Equation (4) is the trail equation used in this research.

$$Trial = [trial/\sum trial]\alpha \qquad (4)$$

where α=trail coefficient

The concept of the amount of pheromone used is similar with the original ACS concept where the higher the amount of pheromone, the higher the probability value that attracts more ants to follow the shortest path and simultaneously cause the ACS to converge faster. Thus, if the amount of pheromone is smaller, the probability will become lower thus the path cost will be higher as the ants will avoid traversing a path with a lower pheromone amount.

Step 3: Each time ants construct a path from one node to another, the pheromone amount will be reduced locally by the given evaporation rate using the formula of update local rule sas shown below:

$$Tij(new\ trial) \leftarrow (1-\rho) \times Tij(old\ trial) \qquad (5)$$

where ρ=evaporation rate

This equation shows that each time the ants move from one node to another node, the amount of local pheromone will be updated in parallel. This process prevents the map from getting unlimited accumulation of pheromone and enables the algorithm to forget a bad decision that has been previously made.

Step 4: Once the ants finds its path to goal, the fitness of ants will be calculated. This covers the calculation of distance or path cost each ant takes to traverse from start point to goal point by using derivation of objective function for RPP below:

$$Distance = \sqrt{[(y2-y1)^2 + (x2-x1)^2]} \qquad (6)$$

Step 5: The fitness value will then be used for the process of global update. When all ants reach the destination, the ants will update the value of pheromone globally based on the fitness found

by each ant by using Equation (7) below. This process will be repeated as the path being traverse by ants in each generation is determined using this global value.

Generally, this process will attract the ants to follow the optimal path. During the process, the path with the shorter distance will be chosen as the probability to be chosen is higher compared to the path with the longer distance. The equation of global updating is derived in (7) and (8) below:

$$t_{ij} \leftarrow t_{ij} + \sum \Delta t_{ij}^{k} \qquad (7)$$

where Δt_{ij}^{k} is the amount of pheromone of ant m deposits on the path it has visited. It's defined as below:

$$\Delta t_{ij}^{k} = \begin{cases} Q/C^{k}; if\ arc(i,j) belongs\ to\ path\ P^{k} \\ 0 \\ 1\ ; otherwise \end{cases} \qquad (8)$$

where Q is number of nodes and Ck is the length of the path

Pk built by the ants.

Step 6: The process will be repeated from Step 1 to Step 5 until the process converges. The process will stop when all ants traverse the same path that shows the shortest path to goal has been found. The detailed flow of the algorithm structure is simplified as shown in pseudo code below.

C. Pseudo Code of ACS Algorithm for RPP

$If\ iteration\ (tmax) = 1,2,3,4,5,6, \ldots \ldots \ldots \ldots + n$

$Else\ if\ ant\ (m) = 1,2,3,4,5,6, \ldots \ldots \ldots \ldots + n$

$Else\ if\ nodes\ (n) = 1,2,3,4,5,6, \ldots \ldots \ldots \ldots + n$

Compute the probability of the mth ants next nodes
Move to the next nodes by computed probability
Store history of past location of nodes in an array
If current location of nodes is equal to destination
Break the nodes (n) loop
End
End
Evaluate fitness and store path distance of mth ants
Compute pheromone amount generated by mth ants
End
Update pheromone amount of the entire map
End

V. CONCLUSION

The survey of all the algorithms of robot path planning suggests that in PSO, the path generated in each step is feasible and the robot can reach its target without colliding with the obstacles. In BFO we have given a detailed study of algorithm wherein we have found that that the major process is the chemotactic behaviour.ACO is robust and effective in finding the optimal path as it satisfies the optimization criteria; reducing path cost and computation time, for RPP.

REFERENCES

[1] O. Khatib, Real time obstacle avoidance for manipulators and mobile robots, International Journal of Robotics Research, Vol. 5(10), 1985, pp. 90-98.
[2] G. Nagib, Gharieb W, Path planning for a mobile robot using genetic Algorithm, IEEE proceding of Robotics, 2004, pp185-189.
[3] R. C. Eberhart and Y.H. Shi. Particle swarm optimization: Developments, applications and resources. In *CEC 2001: proceedings of the IEEE congress on evolutionary computation*, pages 81–86, Seoul, South Korea, May, 2001. IEEE.
[4] M. Tasgetiren, Y. Liang, "A Binary particle Swarm Optimization Algorithm for Lot Sizing Problem", Journal of Economic ans Social Research 5(2), 1-20.
[5] W. Jatmiko, K. Sekiyama, T. Fukuda, "A PSO Based Mobile Sensor Network for Odor Source Localization in Dynamic Environment: Theory, Simulation and Measurement ", 2006 Congress on evoluntionary Computation, Vancouver, BC, pp. 3781 -3788, July 2006.
[6] L. Smith, G. Venayagamoorthy, P. Holloway, "Obstacle Avoidance in Collective Robotic Search Using Particle Swarm Optimization"
[7] S. Doctor, G. Venayagamoorthy, v. Gudise,"OptimalPSO for Collective Robotic Search Applications", IEEE Congress on Evolutionary Computation , Portland, OR, pp. 1390-1395, June 2004.
[8] S. Doctor G.Venayagamoorthy "Unmanned Vehical Navigation Using Swarm Intelligence", Intelligent Sensingand Information Processing, ICISIP 2005.
[9] R. Grabowski, l. Navaroo-Serment, P. Khosla, "An Army of Small Robots", Scientific American Reports, Vol. 18 No.1, pp. 34-39, May 2008.
[10] Wireless Meshed Network for Building Automation, Industrial Wireless Book Issue 10:2, http://wireless.industrial-networking.com/articles/articledisplay.asp?id=1264.
[11] J. Pugh, A. Martinoli, "Inspiring And Modeling Multi-Robot Search with Particle Swarm Optimization."
[12] K. M. Passino. Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Systems Magazine*, 22: 52–67, 2002.
[13] S. Das, A. Biswas, S. Dasgupta, A. Abraham, "Bacterial Foraging Optimization Algoritm: Theoretical, Foundation, Analysis and Applications."
[14] N. B. Sariff, N. Buniyamin, "Ant Colony System for Robot path Planning in Global Static Environment."