Multi-Robot Path-Planning Using Artificial Bee Colony Optimization Algorithm

Preetha Bhattacharjee[1], Pratyusha Rakshit[2], Indrani Goswami (Chakraborty)[2], Amit Konar[2]
[1]CSE Dept, [2]ETCE Dept.
Jadavpur University
Kolkata-700032, India
pretb91@gmail.com, pratyushar1@gmail.com, konaramit@yahoo.co.in

Atulya K. Nagar
Math and Computer Science
Liverpool Hope University
L16 9JD, UK
nagara@hope.ac.uk

Abstract— **Path-planning is an interesting problem in mobile robotics. This paper proposes an alternative approach to path-planning of mobile robots using the artificial bee colony (ABC) optimization algorithm. The problem undertaken here attempts to determine the trajectory of motion of the robots from predefined starting positions to fixed goal positions in the world map with an ultimate objective to minimize the path length of all the robots. A local trajectory planning scheme has been developed with ABC optimization algorithm to optimally obtain the next positions of all the robots in the world map from their current positions, so that the paths to be developed locally for n-robots are sufficiently small with minimum spacing with the obstacles, if any, in the world map. Experiments reveal that the proposed optimization scheme outperforms two well-known algorithms with respect to standard metrics, called average total path deviation and average uncovered target distance.**

Keywords-artificial bee colony optimization algorithm;multi-robot path-planning;csntralized plannin.

I. Introduction

Swarm intelligence added a new dimension in artificial intelligence to study the collective behavior and emergent properties of complex systems with a definite social structure. In recent years a number of swarm based optimization techniques have been proposed, among which we here discuss about the artificial bee colony optimization algorithm [2] proposed by Karaboga and Basturk.

The artificial bee colony optimization (ABC) technique is a population based algorithm for numerical function optimization that draws inspiration from the stochastic behavior of foraging in bees. Recent studies have suggested that ABC is better suited in terms of convergence speed to solve multi-objective optimization problems than other evolutionary algorithms (EA) like differential evolution (DE) and particle swarm optimization technique (PSO) [3].We here apply the algorithm to the path-planning problem in mobile robotics. Given a world map for the robot, the path planning problem attempts to determine a trajectory of motion for the robot from an assigned starting point to a given goal position without collision with the given obstacles or other robots in the vicinity of the given robot.

The method we employ here, the artificial bee colony optimization algorithm, satisfies all the pre-requisites of a meta-heuristic algorithm and also performs well on a wide variety of test problems as shown by Basturk and Karaboga [3].

In ABC trial solution, the population directly affects the mutation operation since it is based on the difference of two members of the population. Hence by this method the information of a good member of the population is distributed among others due to the mutation operation and greedy selection mechanism employed to obtain a new member of the population. In the ABC, while the intensification process is controlled by the stochastic and the greedy selection schemes, the diversification is controlled by the random selection.

The path-planning problem taken up here is formulated by a centralized approach, where an iterative algorithm is invoked to determine the next position of all the robots satisfying all the constraints imposed on the multi-objective function. The algorithm is iterated until all the robots reach their destination (goal position).

For realization with ABC by the centralized approach, a fitness function is constructed to determine the next position of the robots that lie on optimal trajectories leading towards the respective goals. The fitness function of the ABC has two main components: 1) the objective function describing the selection of next position on an optimal trajectory, and 2) the constraint representing avoidance of collision with other robots and with static obstacles, placed co-ordinates.

This article is structured as follows. In Section II we present the formulation of the multi-robot motion planning problem. The ABC is discussed in Section III. The pseudo-code for solving the given constrained optimization function is scripted in Section IV and the experimental results are described in Section V. Results and discussion are given in Section VI.

II. Formulation of the Problem

The formulation considers the evaluation of next positions of the robots from their current positions in a workspace avoiding collision with other robots and static obstacles in the workspace. The current position and goal position of each robot is known with respect to a given reference co-ordinate system. The following principles are used satisfying the assumptions.

1) A robot first determines the next position in order to align itself with the goal and thus constructs a local of motion at that position.
2) This alignment may result in a possible collision with the teammates, if more than one robot tries to occupy same next position. The collision may occur with static

obstacles, if the determined next position has already been occupied by a static obstacle. To avoid such collision, the robot has to turn left or right by certain angle and hence new next position is to be determined.

3) If a robot can align itself towards its goal position without any collision, it will move to that calculated next position.
4) If turning left or right requires the same angle of rotation of the robot about z-axis, the tie is broken arbitrarily.

According to principle (1) each robot R_i first determines its next position towards its goal.

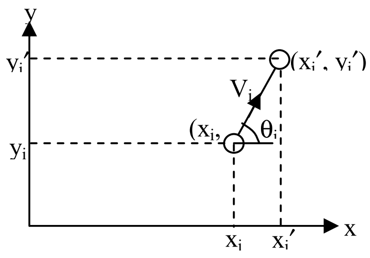

Fig. 1. Current and next position of the i-th robot

Let,

(x_i, y_i) be the current position of the robot R_i at time instant *t*

(x_i', y_i') be the next position of the robot R_i at time instant (*t+1*)

(x_{ig}, y_{ig}) be the goal position of the robot R_i

θ_i be the angle of rotation of robot R_i to align itself towards its goal position

v_i be the velocity of robot R_i

So from Fig. 1, we have

$$x_i' = x_i + v_i cos\theta_i \Delta t \quad (1)$$
$$y_i' = y_i + v_i sin\theta_i \Delta t \quad (2)$$

For Δt=1 sec, above equations are reduced to

$$x_i' = x_i + v_i cos\theta_i \quad (3)$$
$$y_i' = y_i + v_i sin\theta_i \quad (4)$$

According to principle (3) if determined next position of robot R_i, (x_i', y_i') is not occupied by any other robot or static obstacle, R_i should move to (x_i', y_i') and then (x_i', y_i') will become its current position.

According to principle (2) if determined (x_i', y_i') results in a collision, this (x_i', y_i') has to be abandoned and new (x_i', y_i') is calculated so that the line joining (x_i, y_i), (x_i', y_i') and (x_i', y_i'), (x_{ig}, y_{ig}) do not touch the static obstacle as shown in Fig. 2.

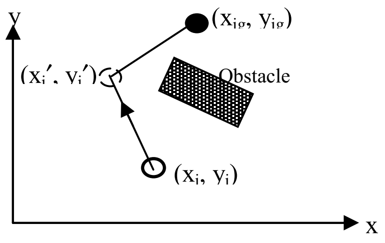

Fig. 2. Selection of (x_i', y_i') from (x_i, y_i) to avoid collision with obstacle.

In order to reach the goal position (x_{ig}, y_{ig}) the parameters which must be taken care are given as follows.

Total Euclidean distance traversed by robot R_i from current position (x_i, y_i) to next position (x_i', y_i') and from next position (x_i', y_i') to goal position (x_{ig}, y_{ig}), which is given as

$$Dist = \sqrt{((x_i - x_i')^2 + (y_i - y_i')^2)} + \sqrt{((x_i' - x_{ig})^2 + (y_i' - y_{ig})^2)} \quad (5)$$

Substituting the values of x_i' and y_i' from (3) and (4) we have for all *n* robots

$$Dist = \sum_{i=1}^{n} \left\{ v_i + \sqrt{((x_i + v_i \cos\theta_i - x_{ig})^2 + (y_i + v_i \sin\theta_i - y_{ig})^2)} \right\} \quad (6)$$

A minimization of *Dist* confirms that the robots will follow the shortest paths. The distance between next position of robot R_i and all other teammates is given as d_{ij}. In order to avoid collision of the i-th robot with the j-th robot, we have to consider the constraint

$$(d_{ij} - 2r) > 0.$$

Let the distance between next position of robot R_i and nearest static obstacle is given as d_{i-obs}. The optimization problem here includes an objective function f, concerning minimization of Euclidian distance between the current positions of the robots with their respective goal positions, constrained by obstacles and teammates on the path. The objective function for the proposed optimization problem is given by

$$f = \sum_{i=1}^{n} \{ v_i + \sqrt{((x_i + v_i \cos\theta_i - x_{ig})^2 + (y_i + v_i \sin\theta_i - y_{ig})^2)} \} + f_{dp} \sum_{i'j'=1}^{n(n-1)/2} (\min(0, (d_{i'j'} - 2r)))^2 + f_{st} / d_{i-obs} \quad (7)$$

where f_{dp} (>0) and f_{st} (>0) are scale factors. In our experiments, we used f_{st} =5000 and f_{dp} =100. These parameters are set in a manner to have all the terms on the right hand side of (7) in the same order of magnitude.

III. Artificial Bee Colony Optimization Algorithm (ABC)

In ABC algorithm, the colony of artificial bees contains three groups of bees:

- A bee waiting on a dance area for making decision to choose a food source is called an *onlooker.*
- A bee going to the food source visited by it previously is named as *employed bee.*
- A bee carrying out random search is called a *scout.*

In ABC algorithm, the position of a food source represents a possible solution of the optimization problem and the nectar amount of a food source corresponds to the fitness of the associated solution. The number of employed bees and onlooker bees is equal to the number of solutions in the population

The ABC algorithm consists of following steps:

A. Initialization

ABC generates a randomly distributed initial population *P* (*g*=0) of $N_p$solutions (food source position). Each solution X_i

(i=0, 1, 2,…, N_p-1) is a *D* dimensional vector. Here *D* is the number of optimization parameters.

B. Placement of employed bees on the food sources

An employed bee produces a modification on the position (solution) in her memory depending on the local information (visual information) as stated by equation (9) and tests the nectar amount (fitness value) of the new source (new solution). Provided that the nectar amount of the new one is higher than that of the previous one, the bee memorizes the new position and forgets the old one. Otherwise she keeps the position of the previous one in her memory.

C. Placement of onlooker bees on the food sources

After all employed bees complete the search process; they share the nectar and position information of the food sources (solutions) with the onlooker bees on the dance area. An onlooker bee evaluates the nectar information from all employed bees and chooses a food source depending on the probability value associated with that food source, p_i, calculated by the following expression:

$$p_i = \frac{fit(X_i)}{\sum_{i=0}^{Np-1} fit(X_i)} \qquad (8)$$

where $fit(X_i)$ is the fitness value of the solution X_i evaluated by its employed bee, which is proportional to the nectar amount of the food source in the position *i* and N_p is the number of food sources which is equal to the number of employed bees. After that, as in case of employed bee, onlooker bee produces a modification on the position (solution) in her memory and checks the nectar amount of the candidate source (solution). Providing that its fitness is better than that of the previous one, bee memorizes the new position and forgets the old one.

In order to find a solution X_i'in the neighborhood of X_i, a solution parameter j and another solution X_k are selected on random basis. Except for the value of chosen parameter j, all other parameter values of $X_i'$are same as in the solution x_i, for example,

$$X_i' = (x_{i0}, x_{i1}, x_{i2}, ..., x_{i(j-1)}. x_{ij}'. x_{i(j+1)}, \ldots, x_{i(D-1)}).$$

The value of x_{ij}'parameter in X_i' solution is computed using the following expression:

$$x_{ij}' = x_{ij} + u(x_{ij} - x_{kj}) \qquad (9)$$

where u is a uniformly random variable in [-1, 1] and k is any number between 0 to N_p-1 but not equal to *i*.

D. Scout Bee Phase

In the ABC algorithm, if a position cannot be improved further through a predefined number of cycles called ‘limit’, the food source is abandoned. This abandoned food source is replaced by the scouts by randomly producing a position.

After that again steps (B), (C) and (D) will be repeated until the stopping criteria is met.

IV. SOLVING HE CONSTRAINT OPTIMIZATION ALGORITHM USING ABC

In this section we propose a solution to the centralized version of the multi-robot motion planning problem using ABC. Here angles of rotation of n robots are considered to be parameters of each solution. An algorithm outlining the scheme is discussed below:

Pseudo Code:

```
Input: Initial position ($x_i, y_i$), goal position ($x_{ig}$,$y_{ig}$) and $v_i$ velocity for n robots where $1 \le i \le n$ and a threshold value $\varepsilon$.
Output: Trajectory of motion $P_i$ for each robot $R_i$ from ($x_i, y_i$) to ($x_{ig}$,$y_{ig}$)
Begin
  Set for all robot $i$   $x_{curr-i} \leftarrow x_i, y_{curr-i} \leftarrow y_i$;
  For robot $i = 1$ to n
    Repeat
      Call ABC ($x_{curr-i}$, $y_{curr-i}$, pos-vector)
      // pos-vector denotes current position of all robots //
      Move-to ($x_{curr-i}$,$y_{curr-i}$);
    Until || $curr\_i - G_i$||≤ $\epsilon$
      //$curr\_i$ = ($x_{curr-i}$,$y_{curr-i}$), $G_i$ = ($x_{ig}$,$y_{ig}$) //
  End for;
End.

Procedure ABC ($x_{curr-i}$, $y_{curr-i}$, pos-vector)
Begin
 Initialize all the food sources (initial population) and problem parameters as well as algorithm parameters like “limit”.
 Evaluate the fitness ($fit(X_i)$) of the population.
 For Iter=1 to Maxiter do
  Begin
   For each employed bee
    Begin
     Produce a new solution $X_i'$ from (9);
      Calculate its fitness value $fit(X_i')$;
      If $fit(X_i') > fit(X_i)$Then  $X_i \leftarrow X_i'$; $trial_i = 0$;
        Else  $trial_i = trial_i + 1$;
       End If;
   End For;
   For each onlooker bee
    Begin
      Select the food source $X_i$ depending on $p_i$ as in (8);
      Produce new solution $X_i'$using the same (9);
      Calculate its fitness value;
      If $fit(X_i') > fit(X_i)$Then  $X_i \leftarrow X_i'$; $trial_i = 0$;
        Else  $trial_i = trial_i + 1$;
       End If;
   End For;
   Memorize the best solution obtained so far;
   For i=0to Np-1
    Begin
     If $trial_i$>limit Then reinitialize $X_i$by scout bee;
      End If;
      End For;
 End For;
 Update:
  $x_{curr-i} \leftarrow x_{curr-i} + v_i \cos\theta$; $y_{curr-i} \leftarrow y_{curr-i} + v_i \sin\theta$ ;
Return.
```

V. Experiment and Computer Simulation

The multi-robot path planning was implemented in C on a Pentium processor. The experiment was performed with 10 similar soft-bots of circular cross-sections. The radius of robot was 6 pixels. For each robot the starting and goal position are predefined prior to initiating the experiment. The experiments were performed with 2, 4, 6, 8 and 10 differently shaped obstacles. The experiments were conducted with equal velocities for all the robots in a given run of the program; however, the velocities were adjusted over different runs of the same program.

One of our experimental world-maps is shown in Fig. 3. Fig. 3(a) demonstrates an initial configuration of the world-map with 4 dark obstacles and the starting and the goal positions of 6 circular soft-bots. The steps of the movement of the robots are shown in Fig. 3(b).

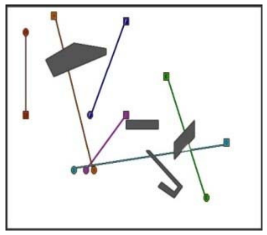
Fig. 3(a)

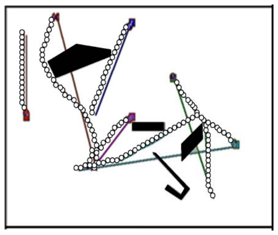
Fig. 3(b)

Fig. 3. Initial and final configuration of the world map with 4 obstacles

To analyse the performance of the proposed multi-robot motion-planning problem, we measured the following two parameters

A. *Average total path deviation (ATPD)*

Let P_{ik} be a path from the starting point S_i to the goal point G_i generated by the program for robot R_i in the k-th run. If P_{i1}, P_{i2}, …, P_{ik} are the paths generated over k runs then the average path traversed (APT) by robot R_i is given by $\sum_{j=1}^{k} P_{ij}/k$.
If the ideal path for robot R_i obtained geometrically is $P_{i\text{-}ideal}$, then the average path deviation is given by

$$P_{i-ideal} - \sum_{j=1}^{k} P_{ij}/k$$

Therefore for n robots in the workspace the average total path deviation (ATPD) is
$\sum_{i=1}^{n}(P_{i-ideal} - \sum_{j=1}^{k} P_{ij}/k)$

B. *Average Uncovered Target Distance*

Given a goal position G_i and the current position C_i of a robot on a 2-dimensional workspace, where G_i and C_i are 2-dimensional vectors, the uncovered distance of robot i is $\| G_i- C_i \|$, where $\|.\|$ denotes Euclidean norm.
For n robots, uncovered target distance (UTD) is the sum of $\|G_i- C_i \|$ i.e.,
UTD=$\sum_{i=1}^{n} ||G_i - C_i||$

For all experiments conducted in this study, we considered *k=10*.

The experiment was conducted using the centralized version of the algorithm, where we used (7) as the fitness function to determine the next position of each robot from the current position. The algorithm is iterated until all the robots reach their respective goal positions. Let the number of robots be n and the number of obstacles m.

Fig. 4 shows that with decrease in velocity, AUTD takes a longer time to attain zero value. Similar observations also follow for the number of robots n, as a variable in the AUTD versus number of steps plot (Fig. 5).

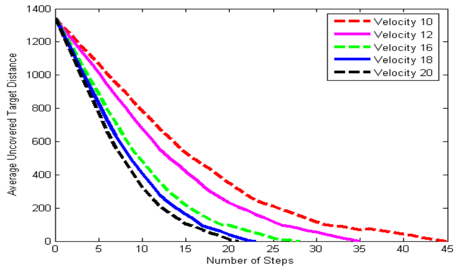

Fig. 4. AUTD vs. Number of steps with velocity as variable for number of obstacles=5(constant)

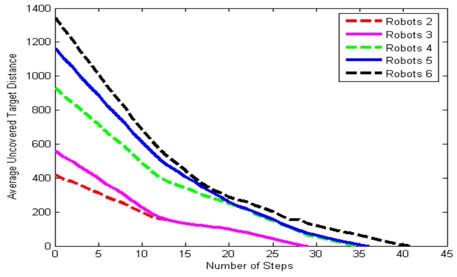

Fig. 5. AUTD vs. Number of steps with number of robots as variable for number of obstacles=8(constant)

Fig. 4 shows that the AUTD gradually diminishes with iterations. Further, it is noted that the larger the velocity settings of the robots in program run, the faster is the fall off in the AUTD profile.

The fall-off in AUTD over program steps for a given n is demonstrated in Figs. 5 where we see that the larger the number of robots, the slower the convergence. Slower convergence, in turn, causes a delayed fall-off in AUTD.

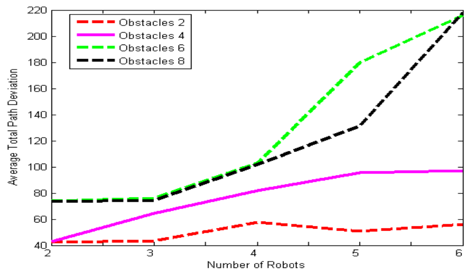

Fig. 6(a). ATPD vs. Number of robots with number of obstacles as variables for velocity=12 unit (constant)

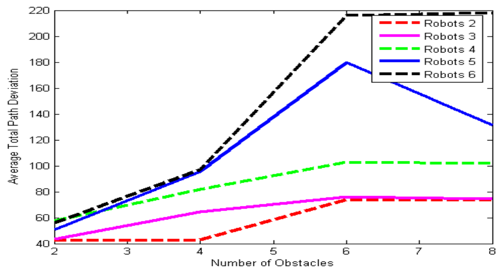

Fig. 6(b). ATPD vs. Number of obstacles with number of robots as variables for velocity=12 unit (constant)

We note from Fig. 6(a) that ATPD is a non-decreasing function of *n* for a constant *m*. An intuitive interpretation of this phenomenon is that with increase in *n*, robots face more constraints to plan local trajectories, thereby increasing ATPD. It is also noted from Fig. 6(a) that for a constant *n*, an increase in *m* causes more spatial restrictions in trajectory planning, thereby increasing ATPD. The same observations follow from Figure 6(b).

The relative performance of DE, PSO and ABC can be studied through error estimation as indicated in Fig. 7(a)-(b). In these figures, we plotted the average of total path traversed (ATPT) obtained from classical DE-, PSO- and ABC - based experiments, corresponding to each value of *n*. We also evaluated the error in ATPT by taking the difference of ATPT values obtained from DE and ABC and also from PSO and ABC as shown in Fig. 7(b). Let *Ei* be the error for the *i*-th sample data. Since the errors for different sample data are all positive, indicating a superiority of ABC over DE and PSO, a measure of the relative goodness of ABC over DE and PSO can be defined as the root mean square error *Er.m.s*= 28.584 and *Er.m.s*= 107.323 respectively. This shows ABC as having an advantage over DE and PSO for the multi-robot motion planning problem. Of course, the root mean square error (28.584 and 107.323 respectively) at the sample points being insignificantly less than the root mean square value (1091.754 and 1136.364 respectively) of the averaged ATPT profiles for DE, PSO and ABC - based simulations.

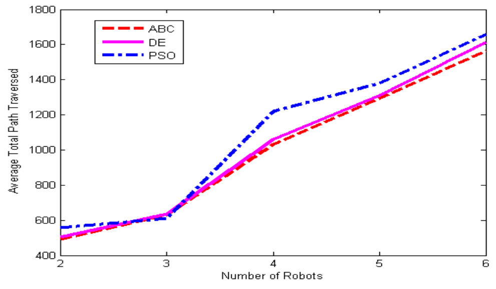

Fig. 7(a). Average total path traversed vs. Number of robots

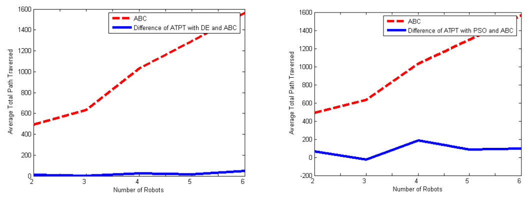

Fig. 7(b). Average (dotted line) and the difference (solid line) of ATPT vs. Number of robots obtained from the Fig. 7(a)

In Fig. 8, we plotted the average of total path deviation (ATPD) obtained from classical DE-, PSO- and ABC - based experiments, corresponding to each value of *n*. From the figure it has been noted that path deviation incurred in case of ABC-based simulation is less than that of classical DE and PSO- based simulations.

From Fig. 9, it has been noted that AUTD takes more time to attain a zero value in case of classical DE and PSO- based simulations than ABC-based simulation.

The relative performance of ABC, DE and PSO-based can be studied through the plot of average values of uncovered target distance (AUTD), total path traversed (ATPT) and total path deviation (ATPD) obtained from ABC-, DE- and PSO-based experiments and also by observing the number of steps required for the robots to reach their goals with ABC, DE and PSO based algorithm as shown in Fig. 10 and 11. ABC seems to have marginally outperformed classical DE and PSO considering all the cases.

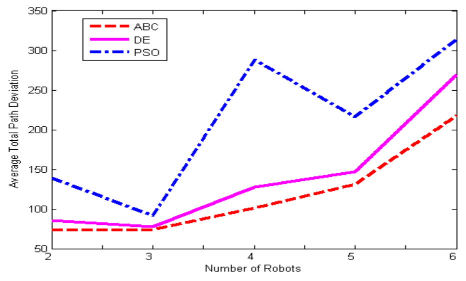

Fig. 8. Average total path deviation vs. Number of robots

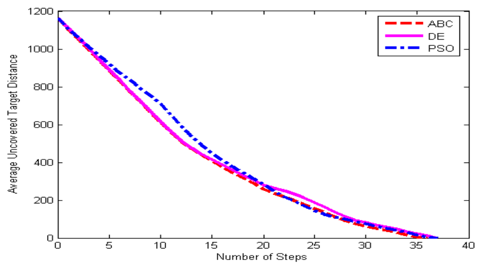

Fig. 9. Average uncovered target distance vs. Number of steps

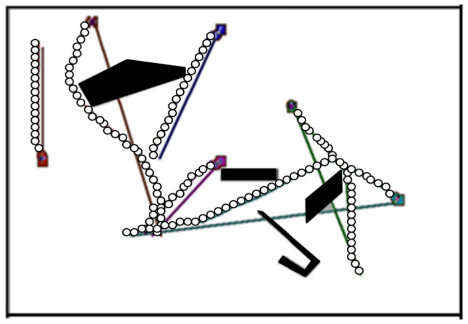

Fig. 10(a). Final configuration of the world map after execution of the ABC-based simulation with 6 robots and 4 obstacles requiring 33 steps.

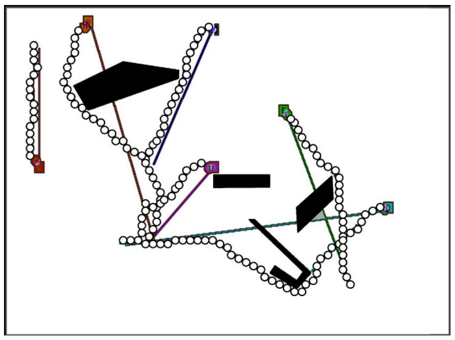

Fig. 10(b). Final configuration of the world map after execution of DE- based simulation for 6 robots and 4 obstacles requiring 34 steps.

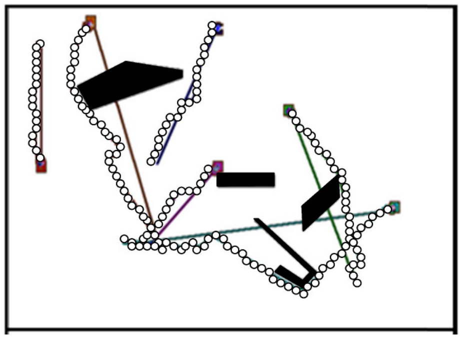

Fig. 10(c). Final configuration of the world map after execution of PSO based-simulation for 6 robots and 4 obstacles requiring 36 steps.

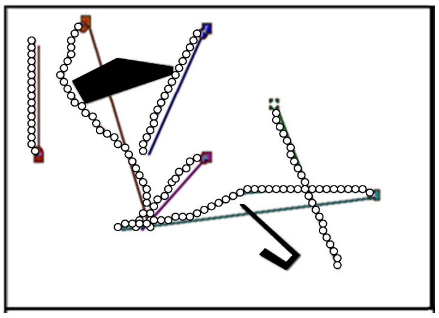

Fig. 11(a). Final configuration of the world map after execution of ABC-based simulation with 6 robots and 2 obstacles requiring 35 steps.

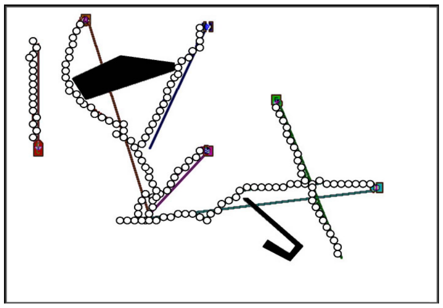

Fig. 11(b). Final configuration of the world map after execution of DE- based simulation for 6 robots and 2 obstacles requiring 37 steps.

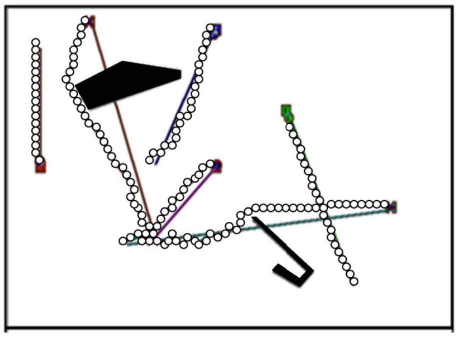

Fig. 11(c). Final configuration of the world map after execution of PSO- based simulation with 6 robots and 2 obstacles requiring 38 steps.

VI. Results and Discussion

Although the local converging speed of DE is good, it might encounter the premature convergence in optimizing multi-modal functions. This problem is however taken care of in ABC as the scout is assigned the task of searching for new food sources, after the employed and onlooker bees have exhausted the profitable food source obtained after certain iterations of the program.

Simulation results obtained in this experiment show that ABC algorithm performs better than the mentioned algorithms (DE and PSO) and can be efficiently employed to solve practical engineering problems with high dimensionality.

VII. Conclusion

The paper introduced a new technique for multi-robot path-planning in a given environment with an ultimate objective to select the shortest path length of all the robots without hitting any obstacles in the world map. The ABC algorithm has been employed here for local path-planning of the individual robots. Experiments reveal that the proposed scheme outperforms the PSO- and DE-based path-planning scheme at least with respect to two well-known metrics: ATPT and AUTD, introduced in [4].

References

[1] Karaboga, D., *An idea based on honey bee swarm for numerical optimisation*, Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

[2] B. Basturk, Dervis Karaboga, *An Artificial Bee Colony (ABC) Algorithm for Numeric function Optimization*, IEEE Swarm Intelligence Symposium 2006, May 12-14, 2006, Indianapolis, Indiana, USA.

[3] D. Karaboga, B. Basturk, *On the performance of artificial bee colony (ABC) algorithm,* In: Applied Soft Computing 8 (2008) 687-697.

[4] J. Chakraborty, A. Konar, *A Distributed Cooperative Multi-Robot Path Planning Using Differential Evolution,* Evolutionary Computation, 2008, CEC 2008, (IEEE World Congress on Computational Intelligence), Pages: 718-725, 23 September, 2008.

[5] R. Storn, K. Price, *Differential evolution- A Simple and Efficient Heuristic for Global continuous spaces*, Journal of Global Optimization, vol. 11, no. 4, pp. 341-359, 1997.

[6] (2002) The IEEE website. [Online]. Available: http://www.ieee.org/

[7] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

[8] D. Karaboga, B. Akay, *A Survey: Algorithms Simulating Bee Swarm Intelligence*, Artificial Intelligence review, vol. 31, Jun 2009, pp 64-85.

[9] J. Vesterstrom, R. Thomsen, *A Comparative Study of differential Evolution, Particle Swarm Optimisation, and Evolutionary Algorithms on Numerical benchmark Problems*.