Distributed Cooperative Multi–Robot Path Planning Using Differential Evolution

Jayasree Chakraborty, Amit Konar, Uday K. Chakraborty and L.C. Jain

Abstract—This paper provides an alternative approach to the co-operative multi-robot path planning problem using parallel differential evolution algorithms. Both centralized and distributed realizations for multi-robot path planning have been studied, and the performances of the methods have been compared with respect to a few pre-defined yardsticks. The distributed approach to this problem out-performs its centralized version for multi-robot planning. Relative performance of the distributed version of the differential evolution algorithm has been studied with varying numbers of robots and obstacles. The distributed version of the algorithm is also compared with a PSO-based realization, and the results are competitive.

I. Introduction

THE late 1990s have seen a significant progress in mobile robotics [1], [2], [3]. Path planning is regarded as a fundamental problem in mobile robotics. Given a world map for a robot, the path planning problem [31] attempts to determine a trajectory of motion for the robot from a predefined starting point to a given goal point without colliding with any obstacle in the map. The basic path planning problem has several extensions and classifications. One common classification of the problem includes local and global planning [12], [13], [14], [15]. In local path planning a robot navigates through the obstacle map in steps and determines its next position toward the goal, satisfying one or more predefined constraints on path-, time- or energy-optimality [4], [5], [6], [7], [10], [11]. In global planning, on the other hand, the robot plans the entire path prior to its movement towards the goal. Such type of global planning is sometimes referred to as offline planning in the literature [32].

Significant progress on single robot motion planning [34], [35], [5] has been attained in mobile robotics over the last three decades. Classical approaches such as quad-tree [34], graph-based, voronoi-diagram, heuristic algorithms such as real time A* [23], neural [35] and evolutionary [5] algorithms are some of the well-known techniques for path planning. In a multi-robot motion planning problem, each robot has a predefined starting and goal position in a given world map and the robots have to plan their paths, either globally or locally, without hitting any of the teammates or obstacles. The obstacles may be stationary or dynamic. However, we in this paper consider stationary obstacles in the given world map for the robots.

The path-planning problem in the given context attempts to minimize the total distance traveled by the robots in the given workspace, subject to the constraint that the robots do not hit each other or the static obstacles. It may be noted that like the single robot path planning problem, the multi-robot path planning is an NP-complete [16] problem, as no polynomial time algorithm to solve the problem is known at this time [9].

Researchers are keen to consider geometry/topology-based path planning problems for multi-robot systems. In a geometry-based path planning problem [17], [15], the obstacle location and the geometry of the environment are considered, whereas in a topology-sensitive path planning system [18], [19], a specialized data structure, such as a graph, describing the paths among different regions of the robots' world map is considered. In a recent work [20], researchers employed a hierarchical decomposition of sub-graph, describing road maps for the robots, for topology-sensitive path planning for multiple robots. Such works are more effective for obstacle-free environment.

The multi-robot path-planning problem can broadly be realized by two distinct approaches, centralized and distributed. In the centralized approach [18], [21], [22], the objective functions and the constraints for path/motion planning of all the robots are considered together. The computation for local planners by the centralized approach is too costly, and thus is not amenable to real-time realization. On the contrary, the distributed approach divides the complexity of centralized path planning into problems of small complexity to be shared by the robots. Consequently, in distributed planning, each robot attempts to construct its path almost independently, avoiding collision with static obstacles or teammates engaged in path planning.

Most of the distributed planning algorithms attempt to handle the problem into distinct phases. In the 1st phase, the paths for the individual robots are constructed, ignoring the presence of other robots. In the 2nd phase, the paths of the robots are retuned in order to avoid collision with other robots. One fundamental problem of this approach lies in

Jayasree Chakraborty is with Department of Electronics and Tele-Communication Engineering, Jadavpur University, Kolkata 700032, India (phone: +91 33 25142396; e-mail: jayasree2@gmail.com).

Amit Konar, is with Department of Electronics and Tele-Communication Engineering, Jadavpur University, Kolkata 700032, India (e-mail: konaramit@yahoo.co.in).

Uday K. Chakraborty is with Department of Computer Science, University of Missouri, St. Louis, MO 63121 (phone: +1 314 5166339; e-mail: chakrabortyu@umsl.edu).

L. C. Jain is with Knowledge-Based Intelligent Engineering Systems Centre, University of South Australia, Adelaide, SA 5095, Australia (e-mail: lakhmi.jain@unisa.edu.au).

978-1-4244-1823-7/08/$25.00©2008 IEEE

proving completeness of the algorithm. In other words while retuning the paths in the 2nd phase, we can't always guarantee having feasible collision-free trajectories for the robots that really lead them to the goals.

Distributed planning is currently gaining importance in multi-robot path planning applications [23], [24], [25], [26]. In [23], path planning by the distributed approach is accomplished in two steps. First, the individual paths of the robots are constructed independently. In the next step, possible conflicts between two robots are determined when their distance is within a pre-defined setting. Conflicts between robots are resolved by introducing a priority scheme, and the order of re-planning the paths of the robots are determined by the priority scheme. Distributed methods that employ re-planning strategies have limited applications in a small world map with fewer obstacles. The complexity of the algorithm increases significantly with increase in world map size and number of obstacles.

In [28], the authors decompose the problem of multi-robot motion planning into smaller sub-problems including path planning and velocity planning. Initially the individual paths of the robots are generated to satisfy the criteria of minimum path planning for the robots. In the second phase, velocity planning of the robots is needed to avoid collisions among the teammates.

A mixture of the centralized and decentralized schemes has been adopted in [27] where robots in close vicinity are considered together for planning their trajectory in a centralized manner. These robots, for which a centralized scheme of path planning is considered, may be called a group. For two or more such groups, a distributed approach is adopted for planning/re-planning.

In this paper, we propose a new approach to handle the multi-robot motion planning as a stochastic optimization problem. Any stochastic optimization algorithm, such as Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Differential Evolution (DE), and the like could have been used to solve this problem. However, we selected DE for its wide popularity and reportedly better performance with respect to PSO [33] and the classical GA.

It may, indeed, be mentioned here that there exist two alternative formulations of the given problem. First, the problem can be formulated as a centralized one, where an iterative algorithm is invoked to uniquely determine the next positions of all the robots. The algorithm is iterated until all the robots safely reach their destinations (goals). Alternatively, we can employ a distributed approach to solve the same problem. Here we consider n iterative algorithms for n robots, and the i-th algorithm determines the next position for the i-th robot, satisfying the necessary constraints.

For realization with DE in the centralized approach, we need to construct a fitness function for the DE to determine the next positions of all the robots that lie on optimal trajectories leading towards their respective goals. The fitness function of the DE has two main components: 1) the objective function describing the selection of the next position on an optimal trajectory, and 2) the constraint representing collision avoidance with peers and static obstacles.

As the distributed approach employs n algorithms, one each for one robot, each DE algorithm includes a fitness function to minimize the traversed path, avoiding collision. When realized on a single processor system, the distributed approach runs the DEs in a given order, and once all the algorithms have been executed, the next positions of the robots are planned. Apparently, the merits of the n-DE distributed approach lie in faster convergence of the DE's and a reduced total motion planning time.

Cooperation is an important issue in multi-robot motion planning. Sometimes, cooperation is achieved through communication among the robots [1]. In order to improve the quality of the time-optimal solution, researchers prefer to reduce communication overhead, while maintaining cooperation among the robots [12]. In this study, behavioral cooperation of the robots is realized through selection of alternative local trajectories for collision avoidance among teammates.

The remainder of this paper is organized as follows. Section II provides a formulation of the cost function for the centralized and the distributed systems. Section III provides a brief description of the DE algorithm. Section IV presents the algorithm for the distributed system, realized with DE. Experiments and computer simulation for performance analysis are presented in section V. Conclusions are drawn in section VI.

II. Formulation of the Problem

The formulation considers the evaluation of the next positions of the robots from their current positions in a given world map with a set of static obstacles. A set of principles based on the following assumptions is constructed to formalize the path-planning problem.

Assumptions

1) The current position of each robot is known with respect to a given reference coordinate system.

2) The robots have a fixed set of actions for motions. A robot can select one action at a given time.

3) The path-planning problem for each robot is executed in steps until all robots reach their respective (predefined) goal positions.

Principles

The following principles have been used in the present context, satisfying the assumptions.

1) A robot attempts to align itself towards the goal.

2) In case the alignment results in a possible collision with the robots /static obstacles in the environment, the robot has to turn left or right by a prescribed angle to alter its current direction of movement, thereby marginally avoiding collisions with teammates and obstacles.

3) If a robot can align itself with a goal without collision, it can start constructing a path up to the selected node.

4) If turning left or right requires the same angle of rotation of the robot around the z-axis, the tie is arbitrarily

broken.

Let (x_i, y_i) be the current position of the i-th robot at

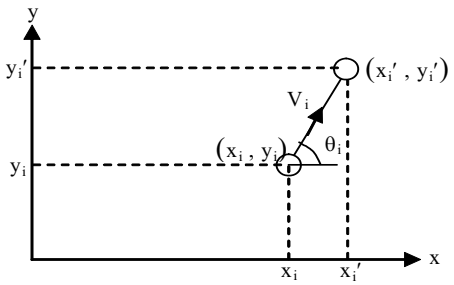

Fig. 1. Current and next position of the i-th robot.

time t, (x_i', y_i') the next position of the same robot at time (t +1), v_i the current velocity of the i-th robot, and (x_{ig}, y_{ig}) the goal position of the i-th robot. It is apparent from Fig. 1 that

$$\left.\begin{aligned} x_i' &= x_i + v_i \cos\theta_i \Delta t \\ y_i' &= y_i + v_i \sin\theta_i \Delta t \end{aligned}\right\} \quad (1)$$

When $\Delta t = 1$, the above pair of equations reduces to

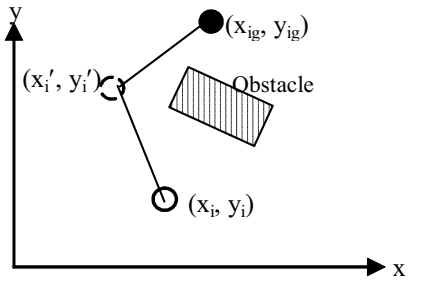

Fig. 2. Selection of (x_i', y_i') from (x_i, y_i) to avoid collision with obstacle.

$$\left.\begin{aligned} x_i' &= x_i + v_i \cos\theta_i \\ y_i' &= y_i + v_i \sin\theta_i \end{aligned}\right\} \quad (2)$$

A. Centralized Planning

Consider the robot R_i initially located at (x_i, y_i). We need to select a point (x_i', y_i'), i.e. the next position of the robot, such that line joining (x_i, y_i), (x_i', y_i') and the one joining (x_i', y_i'), (x_{ig}, y_{ig}) do not touch the obstacle (Fig. 2). This is realized with the DE algorithm.

Let f be an objective function that determines the length of the trajectory. For n robots,

$$f = \sum_{i=1}^{n} \{\sqrt{((x_i - x_i')^2 + (y_i - y_i')^2)} + \sqrt{((x_i' - x_{ig})^2 + (y_i' - y_{ig})^2)}\} \quad (3)$$

Substituting for x_i' and y_i' from expressions (2) we obtain,

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

Since the distance between two robots at any point of time should not be less than a predefined threshold (to avoid collision), we can use this as a primary constraint to this problem. Let $d_{i'j}$ be the distance between i-th and j-th robots' next positions. Then the constraint that the robot will not hit its kin is given by $d_{i'j} - 2r > 0$, where r denotes the radius of the robots.

The optimization problem includes an objective function, concerning minimization of the Euclidean distance between the current positions of the robots with their respective goal positions, constrained by obstacles/teammates on the path. The constraints have been modeled by two types of penalty, the first due to a collision between any two mobile robots, and the second due to the collision of a mobile robot with a static obstacle. Thus the constrained optimization problem is given by

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

$$f_{dp} \sum_{i'}^{n} \sum_{j'=1, i' \neq j'}^{n} (\min(0, (d_{i'j'} - 2r)))^2 + f_{st} / d_{i-obs} \quad (4)$$

where f_{dp} (> 0) and f_{st} (> 0) are scale factors and $d_{i\text{-}obs}$ represents the distance of the nearest obstacle from the i-th robot. In our experiments, we used f_{st} =5000 and f_{dp} =100.

B. Distributed Planning

In centralized planning, the current positions and the goal positions of the robots and the distance between a robot and its neighbors are submitted to a plan-manager that executes a DE algorithm to determine the next position of each robot, avoiding collisions between any two robots. To avoid the heavy computations involved, we construct an alternative arrangement for faster computation. The total task is divided into n sub-tasks, where each subtask is realized by a DE algorithm.

Let f_i denote the constrained objective function for the i-th robot. Following the formulation of the centralized system, we obtain

$$f_i = \sqrt{(x_i - x_i')^2 + (y_i - y_i')^2} + \sqrt{(x_i' - x_{ig})^2 + (y_i' - y_{ig})^2} + f_{dp} \{\min(0, (d_{i'j'} - 2r))\}^2 + f_{st} / d_{i-obs} \quad \forall_i' \quad (5)$$

For a distributed realization of the multi-robot motion-planning problem, we need to employ n DEs, where the i-th DE attempts to minimize f_i in each iteration. The advantage of this scheme is that n DEs are run in parallel, with simple objective functions and less constraint, thereby speeding up the execution time of the algorithm. The use of the distributed realization in real time makes sense.

III. The Differential Evolution Algorithm

Evolutionary algorithms (EA) have proved successful in intelligent search, optimization and machine learning applications. The genetic algorithm (GA) is the most popular member of this class of algorithms. In 1995, Storn and Price proposed an alternative form of EA that includes selection, differential mutation and recombination to generate trial vectors for the next iteration [29], [30], [8]. This algorithm is now popularly called Differential Evolution [8]. DE searches for a global optimum in a D-dimensional hyperspace. It begins with a randomly initialized population of D-dimensional real-valued parameter vectors. Each vector forms a candidate solution to the multi-dimensional optimization problem.

The initial population (at time t = 0) is chosen randomly and should be representative of as much of the search space as possible. Subsequent generations in DE can be represented by discrete time steps: t = 1, 2, ... etc. Since the parameter vectors are likely to be changed over different generations, the following notation has been adopted here for representing the e-th vector of the population at the current generation (at time t):

$$\overset{v}{X}_e(t) = [x_{e,1}(t), x_{e,2}(t), \ldots\ldots x_{e,D}]$$

For each parameter of the problem, there may be a certain range within which the value of the parameter must lie. At the beginning of a DE run, problem parameters or independent variables are initialized somewhere in their feasible numerical range. So, if the m-th parameter of the given problem has its lower and upper bound as x_m^L and x_m^U respectively, then the m-th component of the e-th population member may be initialized as

$$x_{e,m}(0) = x_m^L + \text{rand}\,(0,1).(x_m^U - x_m^L) \qquad (6)$$

where rand(0,1) is a uniformly distributed random number lying between 0 and 1.

For each individual vector $X_{e(t)}$ belonging to the current population, DE randomly samples three other individuals $X_{r1(t)}, X_{r2(t)}$, and $X_{r3(t)}$ from the same generation (for distinct e, r1, r2 and r3), calculates the difference of the components of $X_{r2(t)}$ and $X_{r3(t)}$, scales it by a scalar F ($\subset$ [0,1]) and creates a mutant vector by adding the result to the base vector, $X_{r1(t)}$. So, we can write the mutant vector as

$$v_e(t) = X_{r1}(t) + F.(X_{r2}(t) - X_{r3}(t)).$$

To increase the potential diversity of the population, DE crosses each vector with a mutant vector, building a trial vector:

$$\left.\begin{aligned} U_e(t) = u_{e,m}(t) &= v_{e,m}(t) \quad \text{if rand}\,(0,1) < CR \\ &= x_{e,m}(t) \quad \text{otherwise} \end{aligned}\right\} \qquad (7)$$

where CR (ϵ[0,1]) is the crossover constant.

To keep the population size constant over subsequent generations, the next step of the algorithm calls for 'selection' to determine which one between the target vector and trial vector will survive in the next generation (i.e. at time t+1). DE uses the Darwinian principle of "survival of the fittest" in its selection process which may be expressed as

$$\left.\begin{aligned} X_e(t+1) = U_e(t) &\text{ if } f(\overset{r}{U}_e(t)) \le f(X_e(t)) \\ = X_e(t) &\text{ if } f(X_e(t) < f(U_e(t)) \end{aligned}\right\} \qquad (8)$$

where f(.) is the function to be minimized. If the trial vector yields a better value of the fitness function, it replaces its target in the next generation; otherwise the target is retained in the population. Hence the population either gets better (with respect to the fitness values) or remains the same but never deteriorates.

IV. Solving The Constraint Optimization Problem Using DE

In this section we propose a solution to the distributed version of the multi-robot motion planning problem (described in section II) using parallel DEs. The proposed scheme presumes current positions of n robots and their speeds, and determines the next position of each robot by optimizing the given constrained objective function. An algorithm outlining the scheme is presented below:

Pseudo Code

Input: Initial position (x_i , y_i), goal position (x_{ig} , y_{ig}) and velocity v_i for n robots where $1 \le i \le n$ and a threshold value ε.

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 ← x_i ; y_curr_i ← y_i ;
  For robot i = 1 to n
  Repeat
  Call DE ( 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|| ≤ ε
  // curr_i = ( x_curr_i , y_curr_i ), G_i = ( x_ig , y_ig ) //
  End for;
End.

Procedure DE ( x_curr–i , y_curr–i , pos-vector)
Begin
  Initialize population.
  For Iter = 1 to Maxiter do
    Begin
    Create trial vector using equation (7).
    Evaluate fitness.
    If the trial vector is better than its target vector
    Then replace the target by trial in the next generation;
    End If;
  End for;
  Update:
        x_curr_i ← x_curr_i + v_i cos θ_r ;
```

$y_{curr\_i} \leftarrow y_{curr\_i} + v_i \sin\theta_r$;

Return;
End.

V. Experiment and Computer Simulation

The multi-robot path planning was implemented in C on a Pentium processor. The experiment was performed with 14 similar soft-bots of circular cross section. The radius of a robot was 6 pixels. For each robot the starting and the goal points are predefined prior to initiating the experiment. The experiments were performed with 0, 2, 5, 8, and 11 differently shaped obstacles. While performing the experiments, old obstacles were retained, and new obstacles were added. 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 11 dark obstacles, and the starting and the goal positions of 14 circular soft-bots. The steps of movement of the robots are shown in Fig. 3(b).

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

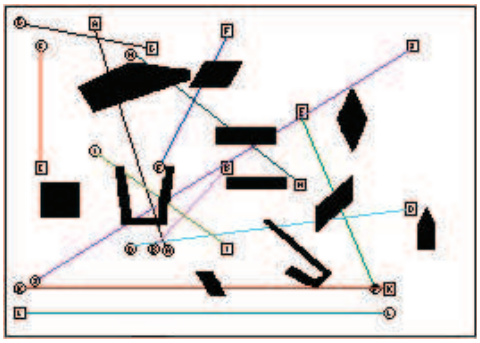

Fig. 3(a). Initial configuration of the world-map with 11 obstacles.

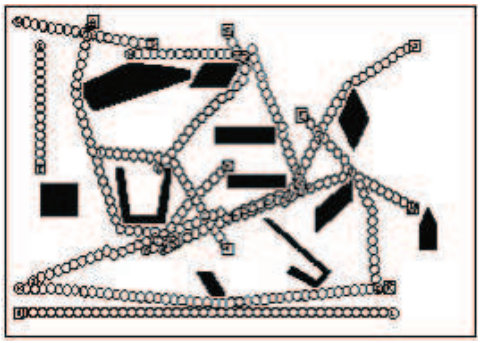

Fig. 3(b). Final configuration of the world-map.

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 $P_{i1}, P_{i2}, \ldots, P_{ik}$ are the paths generated over k runs then the average path traversed (APT) by robot R_i is given by $\sum_{j=1}^{k} P_{ij} / k$ and the average path deviation for this robot is evaluated by measuring the difference between APT and the ideal shortest path between S_i to G_i (with minimum threshold spacing with each obstacle). The threshold in our experiment was considered to be one pixel.

If the ideal path for robot R_i obtained geometrically is $P_{i-ideal}$, 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 (AUTD)

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 for 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.,

$$\text{UTD} = \sum_{i=1}^{n} \|G_i - C_i\| .$$

Now, for k runs of the program, we evaluate the average of UTDs and call it the average uncovered target distance (AUTD). For all experiments conducted in this study, we considered $k = 10$.

The experiment was first conducted using the centralized version of the algorithm, where we used (4) as the fitness function of a single DE 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. The experiment was performed by setting the same velocity for all the robots in a given program run, and AUTD readings versus the number of steps were noted for each run. The experiment was then repeated by changing velocities of the robots in each run. Fig. 4 shows that with decrease in velocity, AUTD takes a longer time to attain the 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).

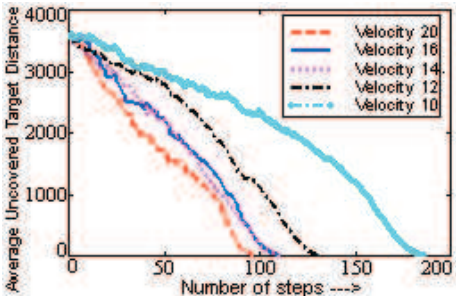

Fig. 4. AUTD vs. Number of steps with velocity as variable for number of obstacle=5 (constant) in centralized approach.

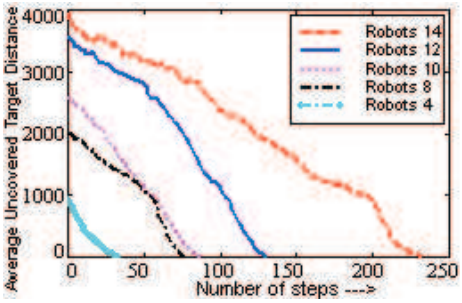

Fig. 5. AUTD vs. Number of steps with number of robots as variables for number of obstacle=5 (constant)

Experiments were also undertaken for a distributed realization of the algorithm, and the performance of the proposed multi-robot motion-planning problem is obtained from the experimental graphs (Fig. 6 - Fig. 8).

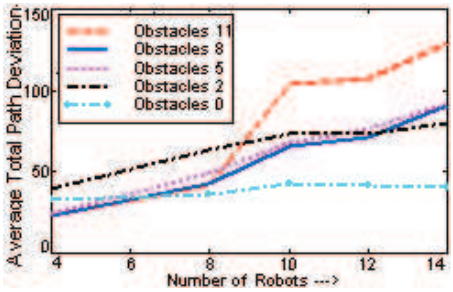

Fig. 6(a). ATPD vs. Number of Robots with number of obstacles as variables for velocity =12 unit (constant).

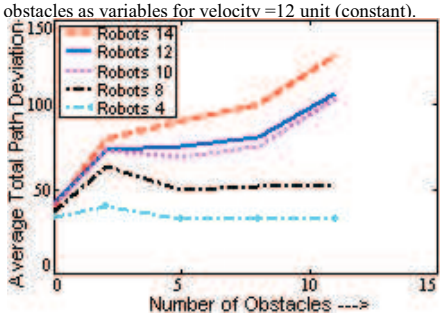

Fig. 6(b). ATPD vs. number of obstacles with no. of robots as variables for velocity=12 unit (constant).

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).

Average uncovered target distance (AUTD) is plotted against the number of steps in Fig. 7 for a given velocity of all the robots. Fig. 7 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.

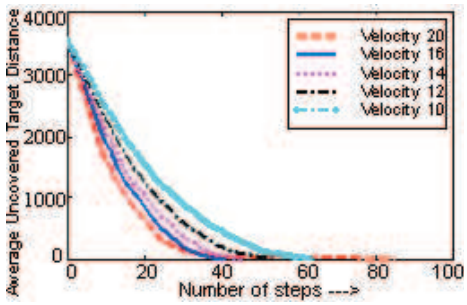

Fig. 7. AUTD vs. Number of steps with velocity as variable for number of obstacles=11 (constant).

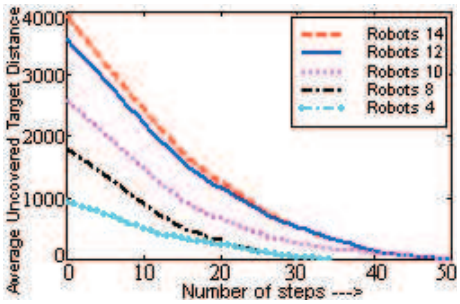

Fig. 8(a). AUTD vs. Number of steps with no. of robots as variables for number of obstacles=5 (constant).

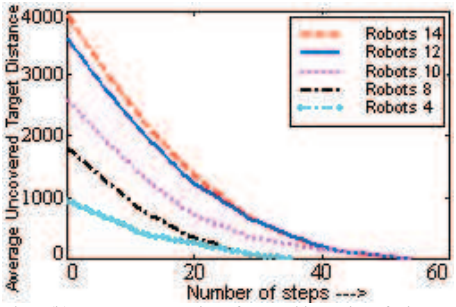

Fig. 8(b). AUTD vs. Number of steps with number of robots as variable for number of obstacle=11(constant).

The fall-off in AUTD over program steps for a given n is demonstrated in Figs. 8(a) and 8(b) 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. Figs. 8(a) and 8(b) provide similar information on AUTD versus number of steps for varying n. The differences

in these figures lie in the number of obstacles m. The number of obstacles is 5 and 11 in Figs. 8(a) and (b), respectively. It is clear from these figures that a higher value of m results in a higher convergence time. Consequently, AUTD is higher for more obstacles at a given iteration.

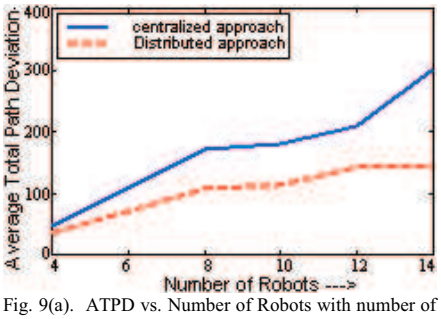

Fig. 9(a). ATPD vs. Number of Robots with number of obstacles=5 (constant), for both centralized and distributed approach.

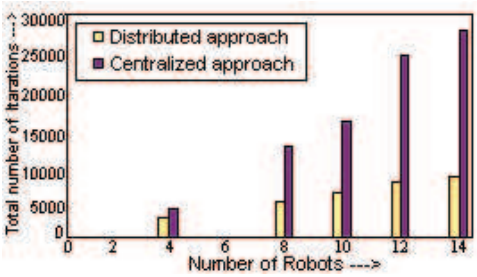

Fig. 9(b). Total number of Iterations of all the DE programs, required to reach the robots to their goal position vs. number of Robots for number of obstacles= 5 (constant).

To compare the centralized and distributed approaches we run the programs until DE algorithms converge. Fig. 9 compares the two approaches, for the world-map with 5 obstacles. From Fig. 9(a) we see that in the centralized approach, average total path deviation is more than that in the distributed approach. Fig. 9(b) shows that the number of total iterations for the centralized approach is more than the distributed approach's. We calculate the total iterations for centralized and distributed realization by $\sum_{i=1}^{s} I_i$ and $\sum_{i=1}^{s}\sum_{k=1}^{n} I_{ik}$ respectively, where s is the total number of steps for all the robots to reach their destinations, I_i = number of iterations of the DE program in the i-th step,

I_{ik} = number of iterations of the DE program for the k-th robot in i-th step.

The relative performance of DE and PSO can be studied through error estimation as indicated in Fig. 10(a). In this figure, we plotted the average of total path traversed (ATPT) obtained from DE- and PSO-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 PSO as shown in Fig. (10b).

Let E_i be the error for the i-th sample data. Since the errors for different sample data are all positive, indicating a superiority of DE over PSO, a measure of the relative goodness of DE over PSO can be defined as the root mean square error $E_{r.m.s}$= 36.9037. This shows DE as having an advantage over PSO for the multi-robot motion planning problem. Of course, the root mean square error (36.9037) at the sample points being much smaller than the root mean square value (3060.20) of the averaged ATPT profiles for PSO and DE-based simulations, DE seems to have marginally outperformed PSO.

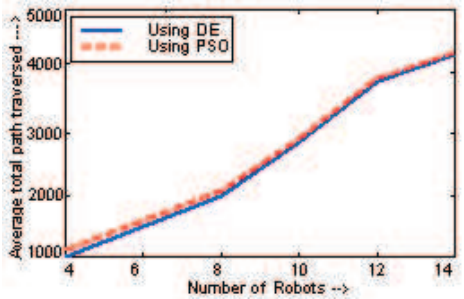

Fig. 10(a). Average total path traversed vs. number of robots.

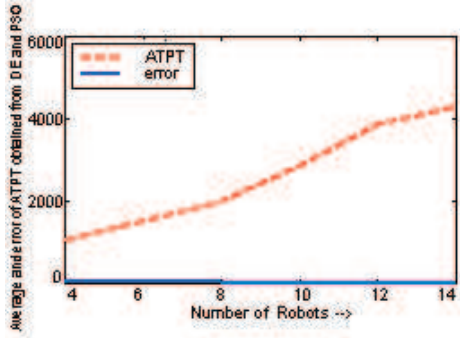

Fig. 10(b). Average (dotted line) and the difference (solid line) of ATPT vs. number of robots obtained from the Fig. 10(a).

VI. Conclusion

This paper addressed the issue of multi-robot motion planning by DE, and solved the problem by both centralized and distributed approaches. Performance evaluation metrics ATPT and AUTD have been used to study the performance of the proposed distributed algorithm and to study the relative performance of the two approaches. The results of performance evaluation confirm the advantage of the latter approach over the former. The proposed distributed algorithm differs from classical two-phase motion planning algorithms, and the effort on offline planning in the 1st phase of classical algorithms can be avoided by our approach. Since n parallel DEs have been employed to handle the distributed motion-planning problem of n robots, the run time required for the proposed distributed realization is small in comparison with other classical algorithms. The distributed DE-based algorithm was also found to be comparable to or better than PSO-based path planning.

REFERENCES

[1] R. C. Arkin, Behavior Based Robotics, MIT press, 2004.

[2] T. Balch and R. C. Arkin, "Behavior-based formation control for multi-robotic teams," IEEE Transactions on Robotics and Automation, vol. 14, no. 6, pp. 926-934, 1998.

[3] T. Fukuda, S. Nakagawa, Y. Kawauchi, and M. Buss, "Structure Decision for Self Organizing Robots Based on Cell Structure – CEBOT," in Proceedings of the IEEE International Conference on Robotics and automation, vol. 2, pp. 695-700, 1989.

[4] M. Gerke and H. Hoyer, "Planning of optimal paths for autonomous agents moving in inhomogeneous environments," in Proceedings of the 8th International Conference on Advanced Robotics, pp. 347–352, July 1997.

[5] J. Xiao, Z. Michalewicz, L. Zhang, and K. Trojanowski, "Adaptive Evolutionary Planner/Navigator for Mobile Robots," IEEE Transactions on Evolutionary Computation, vol. 1, no. 1, April 1997.

[6] Z. Bien and J. Lee, "A Minimum-Time Trajectory Planning Method for Two Robots," IEEE Trans. on Robotics and Automation, vol. 8, no. 3, pp. 443-450, June 1992.

[7] K. G. Shin and Q. Zheng, "Minimum time collision free trajectory planning for dual robot systems," IEEE trans. on Robotics and automation, vol. 8, no. 5, pp. 641-644, October 1992.

[8] P. Kaelo, M. M. Ali, "Differential evolution algorithms using hybrid mutation," Computational Optimization and Applications, vol. 37, no. 2, pp. 231-246, June 2007.

[9] V. Kunchev, L. Jain, V. Ivancevic, A. Finn, "Path Planning and Obstacle Avoidance for Autonomous Mobile Robots: A Review," in Lecture Notes in Artificial Intelligence, KES 2006, Part II, LNAI 4252 , Springer- Verlag, Germany, pp.537-544, 2006.

[10] I. Duleba and J. Z. Sasiadek, "Nonholonomic Motion Planning Based on Newton Algorithm with Energy Optimization," IEEE Trans. on Control Systems Technology, vol. 11, no. 3, May 2003.

[11] M. Moll, L. E. Kavraki, "Path Planning for Minimal Energy Curves of Constant Length," in Proceedings of the 2004 IEEE International Conference on Robotics & Automation, pp. 2826-2831, April 2004.

[12] R. Regele, P. Levi, "Cooperative Multi-Robot Path Planning by Heuristic Priority Adjustment," in Proceedings of the IEEE/RSJ International Conf. on Intelligent Robots and Systems, 2006.

[13] K. H. Sedighi, K. Ashenayi, T. W. Manikas, R. L. Wainwright, and H. Tai, "Autonomous Local Path Planning for a Mobile Robot Using a Genetic Algorithm," in Proceedings of the IEEE International Conference on Robotics and automation, pp. 1338-1345, 2004.

[14] M. Saska, M. Macas, L. Preucil, and L. Lhotska, "Robot Path Planning using Particle Swarm Optimization of Ferguson Splines," in Proceedings of 11th IEEE Int. Conf. on EmergingTechnologies and Factory Automation, pp- 833-839, 2006.

[15] S. M. LaValle and S. A. Hutchinson, "Optimal Motion Planning for Multiple Robots Having Independent Goals," IEEE Tr. on Robotics and Automation, vol. 14, pp. 912-925, 1998.

[16] E. A. Bender, MathematicalMethods on Artificial Intelligence, IEEE Computer society Press, Piscataway, NJ, 1996.

[17] C. Belta and V. Kumar, "Optimal motion generation for groups of robots: a geometric approach," ASME Journal of Mechanical Design, vol. 126, pp. 63-70, 2004.

[18] P. Svestka and M. Overmars, "Coordinated path planning for multiple robots," Robotics and Autonomous Systems, vol. 23, no. 4, pp. 125-152, 1998.

[19] M. Peasgood, J. McPhee, and C. Clark, "Complete and Scalable Multi-robot Planning in Tunnel Environments," in Proc. of the 1st IFAC Workshop on Multivehicle Systems, October 2006.

[20] M. Ryan, "Graph Decomposition for Efficient Multi-robot Path Planning," in Proceedings of the 20th International Joint Confere nce on Artificial Intelligence, pp. 2003-2008, Jan. 2007.

[21] Z. Cai and Z. Peng, "Cooperative co-evolutionary adaptive genetic algorithm in path planning of cooperative multi-mobile robot system," Journal of Intelligent and Robotic Systems: Theory and Applications, vol. 33, no. 1, pp. 61-71, Jan 2002.

[22] J. Barraquand and J. C. Latombe, "Robot motion planning: A distributed representation approach," Int. J. Robot. Res.,vol. 10, no. 6, pp. 628-649, Dec. 1991.

[23] M. Bennewitz, W. Burgard, and S. Thrun, "Optimizing schedules for prioritized path planning of multi-robot systems," in Proceedings of the IEEE International Conference on Robotics and Automation, pp. 271-276, 2001.

[24] J. P. van der Berg and M. H. Overmars, "Prioritized motion planning for multiple robots," in Proc. IEEE/RSJ Int. Conf Int. Rob. and Sys., pp. 2217-2222, 2005.

[25] C. W. Warren, "Multiple robot path coordination using artificial potential fields," in Proc. IEEE Int. Conf. Robotics and Automation, pp. 500-505, 1990.

[26] K. Azarm and G. Schmidt, "Conflict-free motion of multiple mobile robots based on decentralized motion planning and negotiation," in Proceedings of the IEEE International Conference on Robotics and Automation, pp. 3526-3533, 1997.

[27] C. M. Clark, S. M. Rock, and J. C. Latombe, "Dynamic Networks for Motion Planning in Multi-Robot Space Systems," Proceedings of the 2003 IEEE International Conference on Robotics and Automation, pp. 4222- 4227, September 2003.

[28] K. Kant and S. Zucker, "Toward efficient Trajectory Planning: The path-velocity decomposition," The International Journal of Robotics Research, pp. 72-89, 1986.

[29] R. Storn, K. Price, "Differential evolution – A Simple and Efficient Heuristic for Global optimization over continuous spaces," Journal of Global Optimization, vol. 11, no. 4, pp. 341–359, 1997.

[30] J. Lampinen, I. Zelinka, "On stagnation of the differential evolution algorithm," In Proc. of 6th International Conf. on Soft Computing (MENDEL), pp. 76-83, 2000.

[31] C. Yap, "Algorithmic motion planning," in the Advances in Robotics, vol. 1, J.T., Schwartz and C. Yap (Eds.), Hillsdale, NJ: Lawrence Erlbaum, 1987, pp. 95-143.

[32] R. Smierzchalski and Z. Michalewicz, "Path planning in Dynamic Environments," in Innovations in robot mobility and control, S. Patnaik (Ed.), Springer-Verlag, Berlin Heidelberg, 2005.

[33] J. Kennedy and R. C. Eberhart, Swarm Intelligence, ISBN 1-55860-595-9, Academic Press (2001).

[34] S. Kambhampati, L. S. Davis, "Multi-resolution Path Planning for Mobile Robots," Journal of Robotics and Automation, vol. RA-2, no. 3, pp. 135-145, September 1986.

[35] H. Meng and P. D. Picton, "Neural Network for Local Guidance of Mobile Robots," in Proc. of the third Int. Conf. on Automation, Robotics and Computer Vision (ICARCV' 94), pp. 1238-1242, Singapore, Nov. 1994.