STUDY OF TWO SWARM INTELLIGENCE TECHNIQUES FOR PATH PLANNING OF MOBILE ROBOTS

Cezar A. Sierakowski and Leandro dos S. Coelho

Pontifical Catholic University of Parana, PUCPR/PPGEPS/LAS
Imaculada Conceição, 1155, Zip code 80215-901, Curitiba, PR, Brazil
E-mail: cezar.sierakowski@pucpr.br; leandro.coelho@pucpr.br

Abstract: Swarm intelligence was originally inspired in social behaviour in nature, also considering the evolving aspects, several variations in swarm intelligence's techniques made it applicable to optimization problems. In this paper two case studies of static environment, composed with obstacles are presented and evaluated, a comparative study is evaluated between two techniques of swarm intelligence and a genetic algorithm for the presented problems. *Copyright © 2005 IFAC*

Keywords: mobile robots, path planning, swarm intelligence, evolutionary algorithms, optimization.

1. INTRODUCTION

Swarm intelligence is an emerging research area with similar population and evolution characteristics to those of genetic algorithms. However, it differentiates in empathizing the cooperative behaviour among group members. Swarm intelligence is used to solve optimization and cooperative problems among intelligent agents, mainly in computer's networks, mobile robotics (Liu and Passino, 2004) and cooperative and/or decentralized control (Baras *et al.*, 2003). Swarm intelligence is inspired in nature, in the fact that contribution among living animals of a group contribute with their own experiences to the group, making it stronger in face of others. The most familiar representatives of swarm intelligence in optimization problems are: food-searching behaviour of ants (Dorigo and Di Caro, 1999), particle swarm optimization (Kennedy and Eberhart, 2001), and artificial immune system (Castro and Timmis, 2002).

Swarm intelligence, in nature, may be composed of three main principles: evaluation, comparing and imitation. Evaluation is the capacity to analyze what is positive or negative in nature, attractive or repulsive. Even the smaller life forms have these abilities, in the case of bacteria, they are able to notice if the environment in which they are located is noxious or not. Learning won't happen unless beings are capable of evaluate the attractive and repulsive characteristics of the environment. Comparison is the way living beings use other beings as a standard to evaluate themselves, results of these comparisons may become a motivation to learning and/or modification. Imitation is an effective form of learning. However, very few animals, in nature, are capable of imitating, in fact, only human beings and some species of birds are capable of such action (Kennedy and Eberhart, 2001). These three basic principles may be combined, in a simplified version, in computer programs, opening possibilities for them to adapt to complex problems. Animals, or groups of animals, when foraging, act looking for maximizing the amount of energy obtained per unit of time spent foraging, considering the biological and environmental limitations.

This paper contribution is to present a comparative study between two swarm intelligence's techniques and a evolutionary technique, these are bacteria colony, particle swarm optimization and genetic algorithms. These techniques will be applied to two path planning problems of mobile robots.

The next sections of the paper are presented as follows. In section 2, the fundamentals of bacteria colony algorithm are presented. The particle swarm optimization is discussed in section 3. Two case studies of path planning for mobile robots are proposed in section 4. The simulation results and conclusions are presented in sections 5 and 6, respectively.

2. BACTERIA COLONY

Natural selection tends to eliminate animals with poor foraging strategies and to favor gene propagation of those with good foraging strategies, once these have higher chances of succeeding in reproduction. These evolutionary principles have taken scientists to develop the foraging strategies, turning it appropriate to optimization models (Passino, 2002).

A bacterium position, in a time instant, can be determined through equation (1), where the position in that instant is calculated in terms of the position in the previous instant and the step size *C(i)* applied in a random direction *Φ(j)*, generated in the bacterium tumble,

$$\theta'(j+1,k,l)=\theta'(j,k,l)+C(i)\times\phi(j,k,l) \quad (1)$$

To adapt such strategy to optimization problems, an equation to determinate the cost of each position is needed, to possibilitate the comparison between the position and the environment. The cost is determined by the equation,

$$J(i,j,k,l)=J(i,j,k,l)+J_{cc}(\theta'(j,k,l),P(j,k,l)) \quad (2)$$

Through equation (2) is noticed that the cost of a determined position *J(I,j,k,l)* is also affected by the attractive and repulsive forces existing among the diverse bacteria of the population $J_{cc}(\theta'(j,k,l),P(j,k,l))$.

After a determined number of chemotactic steps (steps comprehending the movement and the cost determination of each bacterium position), a reproductive step occurs. In this reproductive step the bacterium are sorted decreasingly by their cumulative cost. The lower half of the list die, these are the bacteria that couldn't gather enough nutrients during the chemotactic steps, and the upper half divide themselves into two new bacteria, located in the same position.

The bacteria colony algorithm is basically composed by an elimination and dispersal loop, inside this loop, there is another one, who is responsible for the bacteria reproduction. Inside this one, there is a third loop, responsible for generating the direction in which each bacterium will run, determining the period the bacterium will move and, as a consequence, determining it's position after the loop execution, and calculating the fitness of these positions. The reproductive loop is responsible for determining which of the bacteria must reproduce and which must be exterminated after the movements executed in loop 3, through a cost analysis of their positions along their movement. The first loop is responsible for eliminating some bacteria; it's ruled by an elimination probability, repositioning them into another random position of the search space. Details of this approach are presented in Passino (2002).

3. PARTICLE SWARM OPTIMIZATION

The proposal of such algorithm appeared from some scientists that developed computational simulations of the movement of organisms such as flocks of birds and fish schooling. Such simulations were heavily based in manipulating the distances between individuals, that is, the synchrony of the behaviour of the swarm was thought as an effort to keep an optimal distance between them. Sociobiologist E. O. Wilson has outlined a link of these simulations to optimization problems (Brandstätter and Baumgartner, 2002).

In theory, at least, individuals of a swarm may benefit from the prior discoveries and experiences of all member of the swarm when foraging. The fundamentals of developing particle swarm optimization (PSO) is an hypothesis in which the exchange of information among beings of a same species offer some sort of evolutionary advantage.

Similarly to genetic algorithms (GAs) (Goldberg, 1989), PSO is an optimization tool based in a population, where each member is called a particle, that is, each particle is a potential solution to the analyzed problem. However, unlike GAs, PSO does not have operators, like crossover and mutation. PSO does not implement the survival of the fittest individuals, instead, it implements the simulation of social behaviour.

The PSO algorithm works as follows, initially, a random position population exists, each of these particles has a speed, and the particles start to "fly around" the search space. Each particle has a memory, allowing it to remember the best position it has visited in history (*pbest*), and also the fitness in that position (Krohling *et al.*, 2004).

The best position ever achieved by the whole swarm is denominated the global best (*gbest*) (Gudise and Venayagamoorthy, 2003). The basic concept of PSO algorithm is to accelerate the particles towards *pbest* and *gbest*, considering a random weight at each time step. Mathematically, the particles move following the equations:

$$V_{id}^{t+1}=W\times V_{idt}+c_1\times rand_1\times\left(P_{id}-X_{id}^{t}\right)+ c_2\times rand_2\times\left(P_{gd}-X_{id}^{t}\right) \quad (3)$$

$$X_{id}^{t+1}=X_{id}^{t}+V_{id}^{t+1}\Delta t \quad (4)$$

where Δt=1, *t* represents the actual iteration and *t*+1 represents the next iteration V_{id} and X_{id} represent the particle speed and position respectively, $rand_1$ and $rand_2$ are two random numbers with uniform distribution in [0,1], used to maintain populations' diversity.

Eq. (3) is used to update each particle's speed, and Eq. (4) represents the position update, according to its previous position and its speed, considering $\Delta t=1$.

Positive constants c_1 and c_2 are denominated cognitive and social components, respectively. These are the acceleration constants, responsible for varying the particle speed towards *pbest* and *gbest*. Constants c_1 and c_2 are not critical factors for determining the algorithm convergence; however, a correct tuning may cause the algorithm convergence to occur faster.

The use of W, called inertia weight was proposed by Shi and Eberhart (1998). This parameter is responsible for dynamically adjust the speed of the particles, so, it's responsible for balancing between local and global search, consequently, needing less iterations for the algorithm to converge. A small value of inertia weight implies in a local search, by the other side, a high value leads to a global search.

Applying a high inertia weight at the start of the algorithm and making it decay to a low value through the PSO algorithm execution, makes the algorithm globally search in the start of the search, and search locally at the end of the execution. Eq. (5) shows how the inertia weight is updated, considering $iter_{max}$ the maximum number of iterations of the algorithm and *iter* the actual iteration (Shi and Eberhart, 2002).

$$W = W_{\max} - \frac{W_{\max} - W_{\min}}{iter_{\max}} \times iter \tag{5}$$

The first step of the PSO algorithm is to start each particle with random numbers, considering that the random number must belong to the search space. Next a loop starts being executed, and it remains until the stopping criteria is met, the stopping criteria may be the convergence of the algorithm, a maximum number of iterations, or anything else. Inside the loop the value of the fitness and the *pbest* of each particle is determined. Once all particles have been analyzed, it's calculated the *gbest*, and with this value, the velocity and position of all particles is achieved.

4. TRAJECTORY PLANNING OF MOBILE ROBOTS

Literature is rich in approaches to solve mobile robots trajectory planning in presence of static and/or dynamic obstacles (Tu and Yang, 2003; Bennewitz *et al.*, 2002; Melchior *et al.*, 2003). One of the most popular planning methods is the artificial potential fields (Tsuji *et al.*, 2002). However, this method gives only one trajectory solution that may not be the smaller trajectory in a static environment. The main difficulties in determining the optimum trajectory are due to the fact that analytical methods are extremely complex to be used in real time, and the searching enumerative methods are excessively affected by the size of the searching space.

Recently, the interest in using evolutionary algorithms, especially genetic algorithms, has increased in last years. Genetic algorithms are used in mobile robots trajectory planning, generally when the search space is large (Fujimori *et al.*, 1997; Xiao *et al.*, 1997; Gemeinder and Gerke, 2003).

The trajectory planning is the main aspect in the movement of a mobile robot. The problem of a mobile robot trajectory planning is typically formulated as follows: given a robot and the environment description, a trajectory is planned between two specific locations which is free of collisions and is satisfactory in a certain performance criteria (Xiao *et al.*, 1997).

Seeing the trajectory planning as an optimization problem is the approach adopted in this article. In this case, a sequence of configurations that moves the robot from an initial position (origin) to a final position (target) is designed.

A trajectory optimizer must locate a series of configurations that avoid collisions among the robot(s) and the obstacle(s) existing in the environment. The optimizer must also try to minimize the trajectory length found, in order to be efficient. The search space is the group of all possible configurations.

In the present study, it's considered a 2-dimensional mobile robot trajectory planning problem, in which the position of the mobile robot R is represented by Cartesian coordinates (x,y) in the xy plan. The initial and destination points of the robot are (x_0, y_0) and (x_{np}, y_{np}), where n_p is a design parameter. The initial point is always (0,0).

Only the trajectory planning problem is empathized in this paper, the robot control problem is not the focus of this paper. However, details of the robots movement equations can be found in Fujimori *et al.* (1997). It's assumed that the obstacles are circular in the robot's moving plan. Besides, the hypothesis that the free 2-dimensional space is connected and the obstacles are finite in size and does not overlap the destiny point is true.

The optimization problem formulated consists of a discrete optimization problem, where the objective function $f(x,y)$, which is the connection between the technique used for optimization and the environment, aims to minimize the total trajectory percurred by the mobile robot and is ruled by

$$f(x, y) = \alpha d_{obj} + \lambda n_o \tag{6}$$

$$d_{obj} = \sum_{i=0}^{n_p} \sqrt{(x(i+1)-x(i))^2 + (y(i+1)-y(i))^2} \quad (7)$$

where α and λ are weighted factors, d_{obj} represents the Euclidian distance between the initial and the destiny point, n_0 denotes the number of obstacles prevented by the robot movement following the planned trajectory, and np is the number of points where a trajectory change occurs (project parameter in this article). It's noticed by the equation (6) that a λ term exists, it's an weighting (penalty) term for unfeasible solutions, meaning, the trajectory that intercepts obstacles. In this case, the fitness function to be evaluated by optimization approaches of this paper aims to maximize

$$fitness = \frac{K_c}{f(x,y)+\varepsilon} \quad (8)$$

where K_c and ε are scale constants.

5. SIMULATION RESULTS

The environment used for the trajectory planning is a 100x100 meters field. The search interval of the parameters is $x_i \in [0,100]$ meters and $y_i \in [0,100]$ m, where i=1,..,np. About the fitness it's adopted α=1, λ=200, K_c =100 and ε=1x10^{-6}. Two simulated cases and the results achieved by the GA, Bacteria Colony and PSO are presented.

The GA used to simulate the cases had population size 50, crossover probability 0.85, mutation probability 0.15, size of each chromosome 16 bits (binary codification), maximum number of generations 100, the selection operator is roulette wheeling with elitist structure.

For the bacteria colony algorithm the following parameters needed to be adjusted p (optimization problem's dimension), S (population size), N_c (number of chemotactic steps), N_s (maximum number of steps that a bacterium can swim in a turn), N_{re} (number of reproductions), N_{ed} (number of elimination-dispersals events), p_{ed} (elimination-dispersal probability) and $C(i)$, i=1,2,...S (speed of the movement taken in one step) it is adopted, S=50, N_c=15, N_s =10, N_{re}=4, N_{ed} =2, p_{ed} = 0.3 and $C(i)$=2.5, i=1,2,...S.

The PSO parameters are population size 50, maximum number of iterations 100, maximum speed 10, maximum inertia weight 0.9, minimum inertia weight of 0.4, and $c_1 = c_2 = 2$.

6.1 Case study 1: Environment with 4 obstacles

In Table 1 are presented the positions of the centers (x_c, y_c) of the circular obstacles and their respective radius (in meters) of case 1. The results obtained with the bacteria colony are restricted to p=3. In Table 2 the achieved solutions after executing each algorithm 10 times are presented.

Table 1: Definition of obstacles for the case study 1.

obstacle number	radius	position (x_c, y_c)
1	10	(40, 15)
2	10	(20, 35)
3	20	(75, 60)
4	15	(35, 75)

Table 2: Results for an environment with 4 obstacles for 10 experiments.

fitness	genetic algorithm	bacteria colony	PSO
mean	0.6778	0.6937	0.6909
maximum	0.6892	0.6954	0.6987
minimum	0.6448	0.6908	0.6273
standard deviation	0.0148	0.0014	0.0223

As noticed by the results presented in Table 2, the three algorithms presented relatively similar performances, when dealing with simple environments. Because the environment is simple, every experiment have achieved a feasible solution, the best trajectory were achieved by PSO with a fitness of 0.6987. In Figs. 1, 2 and 3, the best results achieved by GA, Bacteria and PSO, respectively, are presented.

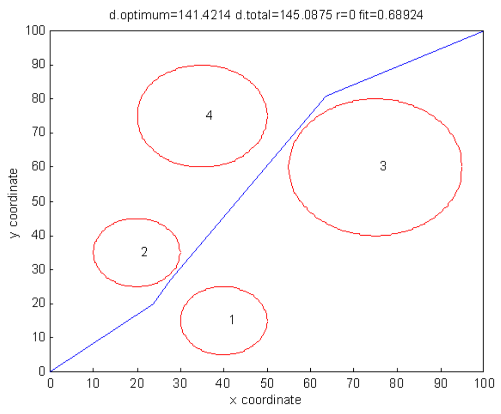

Fig. 1. Best result achieved by GA, for study case 1, after 10 experiments.

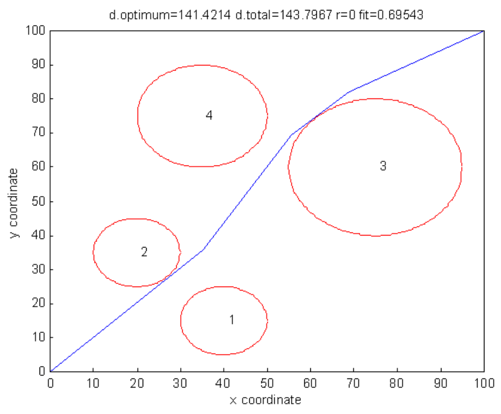

Fig. 2. Best result achieved by Bacteria Colony, for study case 1, after 10 experiments.

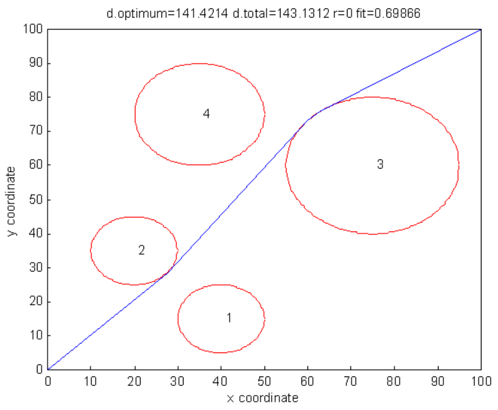

Fig. 3. Best result achieved by PSO, for study case 1, after 10 experiments.

The best result achieved for study case 1 were achieved by the coordinates:

$P_1 = (27.6117, 28.4504)$;
$P_2 = (60.0076, 73.4066)$;
$P_3 = (62.8143, 75.9458)$.

6.2 *Case study 2: Environment with 12 obstacles*

Once the algorithms presented similar performances in simple environments, a complex environment is presented to test their performance. In Table 3 are presented the center positions (x_c, y_c) of the circular obstacles and their respective radius (in meters) for case 2. The results obtained are restricted to p=5. In Table 4 the results for the case study 2 are summarized.

Table 3: Obstacles for case study 2.

obstacle number	radius	position (x_c, y_c)
1	10	(13, 25)
2	08	(10, 76)
3	05	(76, 09)
4	14	(45, 45)
5	09	(12, 55)
6	15	(80, 30)
7	13	(66, 77)
8	08	(32, 15)
9	07	(75, 55)
10	06	(87, 70)
11	08	(35, 66)
12	05	(45, 90)

Table 4: Results for an environment with 12 obstacles for 10 experiments.

fitness	genetic algorithm	bacteria colony	PSO
mean	0.3039	0.5725	0.4182
maximum	0.4867	0.6564	0.6691
minimum	0.1721	0.2804	0.2584
standard deviation	0.1021	0.1168	0.1772

As seen in Table 4, the genetic algorithm does not work very well when dealing with complex environments. The bacteria colony algorithm achieved more regular solutions; this fact is noticed by the higher mean and the best minimum for the fitness after ten experiments. This fact happens because the bacteria colony algorithm is a good global optimizer algorithm. However, PSO achieved the better solution for the environment, however the mean fitness is lower because PSO a good local optimizer.

In other case to simple environments, where the performances of the algorithms were relatively similar, the performances vary from one algorithm to another in complex environments. In Fig. 4, 5 and 6, the best solutions achieved by each algorithm is presented. One point must be emphasized, in study case 2, a good number of experiments couldn't achieve a feasible solution due to the complexity of the environment.

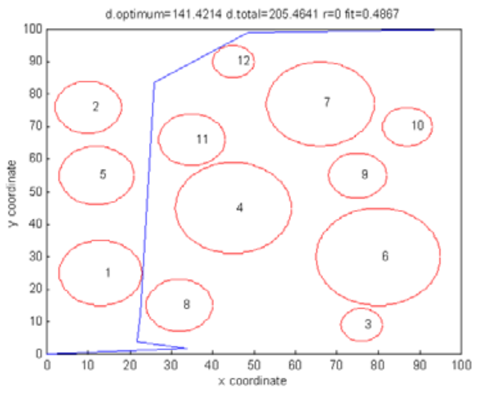

Fig. 4. Best result achieved by GA, for study case 2, after 10 experiments.

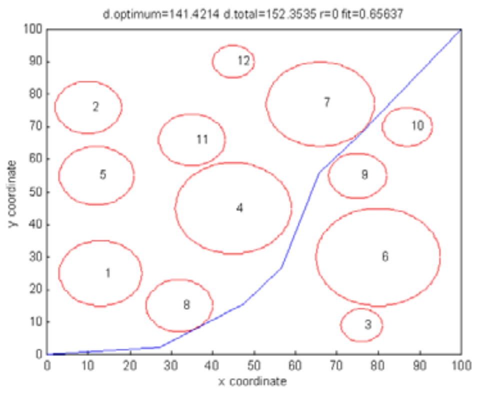

Fig. 5. Best result achieved by Bacteria, for study case 2, after 10 experiments.

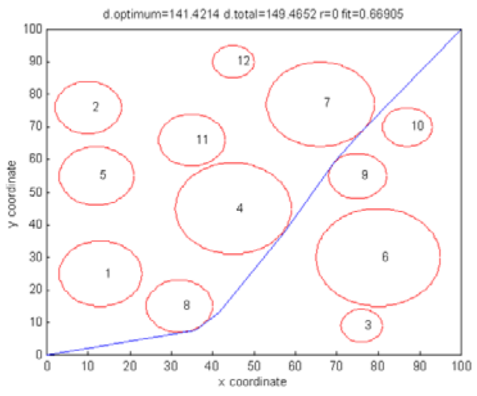

Fig. 6. Best result achieved by PSO, for study case 2, after 10 experiments.

The best result achieved for study case 2 were achieved by the coordinates:

$P_1 = (35.8869, 7.5879)$;
$P_2 = (41.5000, 13.1526)$;
$P_3 = (57.2419, 37.5937)$;
$P_4 = (69.5467, 59.5898)$;
$P_5 = (76.4572, 69.2533)$.

6. CONCLUSION AND FUTURE WORKS

A research area with special relevance to mobile robot systems is devising suitable methods to plan optimum moving trajectories. There exist many approaches within the area of evolutionary computation and swarm intelligence to solve the problem of optimization of path planning in mobile robotics. In this paper the application of the genetic algorithms, bacteria colony and particle swarm optimization is explored for this purpose.

Considering the results presented through this paper it's possible to conclude that there is an advantage in using PSO instead of the other two algorithms, because it achieved the better solution in both case studies and it requires less time to execute. The results of these simulations are very encouraging and they indicate important contributions to the areas of swarm intelligence and path planning in robotics.

However, in future works, more detailed studies related to the parameters related to the three techniques, specially related to the bacteria colony.

REFERENCES

Baras, J.S., Tan, X., and Hovareshti P. (2003), "Decentralized control of autonomous vehicles," Proceedings of the 42nd IEEE Conference on Decision and Control, Maui, Hawaii, USA, pp. 1532-1537.

Bennewitz, M., Burgard, W., and Thrun, S. (2002), "Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots," *Robotics and Autonomous Systems*, vol. 41, no. 2, pp. 89-99.

Brandstätter, B., and Baumgartner, U. (2002) "Particle swarm optimization – mass-spring systems analogon," *IEEE Transactions on Magnetics*, vol. 38, no. 2, pp. 997-1000.

Castro, L. N., and Timmis, J. I. (2002), "Artificial immune systems: a new computational intelligence approach," Springer-Verlag, London.

Dorigo, M., and Di Caro, G. (1999), "The ant colony optimization meta-heuristic," in D. Corne, M. Dorigo, and F. Glover (editors), *New Ideas in Optimization,* McGraw-Hill, pp. 11-32.

Fujimori, A., Nikiforuk, P.N., and Gupta, M.M. (1997), "Adaptive navigation of mobile robots with obstacle avoidance," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 4, pp. 596-602.

Gemeinder, M. and Gerke, M. (2003), "GA-based path planning for mobile robot systems employing an active search algorithm," *Applied Soft Computing*, vol. 3, pp. 149-158.

Goldberg, D.E. (1989). *Genetic algorithms in search, optimization, and machine learning*, Addison Wesley: Reading, MA.

Gudise, V.G., and Venayagamoorthy, G.K. (2003) "Evolving digital circuits using particle swarm," *Proceedings of the International Joint Conference on Neural Networks*, vol. 1, pp. 468-472.

Kennedy, J.F., Eberhart, R.C., and Shi, R.C. (2001), *Swarm intelligence*. San Francisco: Morgan Kaufmann Pub.

Krohling, R. A., Hoffmann, F., and Coelho, L. S. (2004). Co-evolutionary particle swarm optimization for min-max problems using Gaussian distribution," *Proceedings of the Congress on Evolutionary Computation*, Portland, Oregon USA, pp. 959-964.

Liu, Y., and Passino, K.M. (2004), "Stable social foraging swarms in a noisy environment," *IEEE Transactions on Automatic Control*, vol. 49, no. 1, pp. 30-44.

Melchior, P., Orsoni, B., Lavaialle, O., Poty, A., and Oustaloup, A. (2003), "Consideration of obstacle danger level in path planning using A* and fast-marching optimization: comparative study," *Signal Processing*, vol. 83, pp. 2387-2396.

Passino, K.M. (2002), "Biomimicry of bacterial foraging for distributed optimization and control," IEEE Control Systems, vol. 22, no. 3, pp. 52-67.

Shi, Y., and Eberhart, R. C. (1998) "Parameter selection in particle swarm optimizer," *Proceedings Seventh Annual Conference on Evolutionary Programming*, V.W. Porto, N. Saravan, D. Waagen, and A.E. Eiben (eds.). Berlin: Springer-Verlag, pp. 591-601.

Shi, Y., and Eberhart, R. C. (2002) "Fuzzy adaptive particle swarm optimization," *Proceedings of the 2002 Congress on Evolutionary Computation*, Honolulu, Hawaii, USA, vol. 1, pp. 101-106.

Tsuji, T., Tanaka, Y., Morasso, P. G., Sanguineti, V., and Kaneko, M. (2002), "Bio-mimetic trajectory generation of robots via artificial potential field with time base generator," *IEEE Transactions on Systems, Man and Cybernetics - Part C*, vol. 32, no. 4, pp. 426- 439.

Tu, J., and Yang, S.X. (2003), "Genetic algorithm based path planning for a mobile robot," *Proceedings of the IEEE International Conference on Robotics & Automation*, Taipei, Taiwan, pp. 1221-1226.

Xiao, J., Michalewicz, Z., Zhang, L., and Trojanowski, K. (1997), "Adaptive evolutionary planner/navigator for robots," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 18-28.