Smooth Path Planning for Mobile Robot Using Particle Swarm Optimization and Radial Basis Functions

Contact Author: Nancy Arana-Daniel,
Alberto A. Gallegos, Carlos López-Franco and Alma Y. Alanis
Department of Computer Science, CUCEI, University of Guadalajara (UDG),
Av. Revolución 1500, Col. Olímpica, C.P. 44430, Guadalajara, Jalisco, México
Email: nancy.arana, carlos.lopez, alma.alanis@cucei.udg.mx
gallegos.alberto.a@gmail.com

***Abstract*—One of the most important tasks to be performed by a mobile robot is to find a collision-free and smooth path to follow. Given a set of initial control points and using a Radial Basis Function (RBF), a method is proposed, in which is used the RBF's property to approximate smooth functions to define a collision-free and short path. In this paper we formulate the training technique of an RBF as an optimization problem and employed Particle Swarm Optimization (PSO) to solve it.**

The path planning problem is equivalent to optimize the parameters of the RBF using a set of trajectory constraints based on coverage control points as input pattern, which can be seen as places where is desirable for the robot to explore. Furthermore, a combined fitness function is proposed with respect to three requirements: (i) achieve minimum mean square RBF- function approximation error ; (ii) avoid collisions and (iii) minimize the length of the obtained path .

Keywords: Path planning, PSO, RBF.

PAPER TO BE SUBMITED TO: The 2012 International Conference on Genetic and Evolutionary Methods (GEM'12)

I. Introduction

Considerable number of research papers exist in the field of robot path planning. Classic methods used to solve path planning include grid based path planning algorithms as the most commonly used methods [5], [16], [1], [18]. Unfortunately, even though algorithms like A* and D* are complete (they'll always find a solutions if there is one), the paths made by this algorithms lack of smoothness; the robots will often have to stop and readjust their trajectory to continue following the path with every drastic change of direction. For the modeling of the environment, the map is discretized, by doing this, lots of solutions may be excluded, and also it could cause non-smooth paths in algorithms like A*, D* and potential fields. Furthermore, in the case of potential fields, the algorithm could be trapped in a local minimum formed by concave obstacles [1].

Smooth paths are important in robotics because nonholonomic mobile robots are commonly used in practice, so, a smooth path will be more suitable for such robots because this kind of paths are more preferable for designing continuous control algorithms to follow the paths.

New methods based on evolutionary computational methods have been used recently to solve path planning problems due to they are relatively simple to implement, they have fast processing speed, few parameters to be adjusted and good performance. In addition to the above, it is known that the path planning problem can be stated as an optimization (multi-objective optimization) problem and evolutionary computational methods, for instance genetic algorithms (GAs) were used in solving the optimization of path planning successfully [4], [7].

PSO, is a method for optimization of continuous nonlinear functions, created by James Kenedy and Russell Eberhart in 1995 [9]; inspired by the social behavior of bird flocks and school of fishes. In PSO, each individual would be the equivalent of a bird on a flock, each 'bird' is named 'particle', and the 'flock' is called a 'swarm'. A particle is analogous to a chromosome in Genetic Algorithms.
Compared to other Evolutionary Algorithms, classic PSO has no crossover and mutation calculation, actually this is one of the things that makes it really easy to implement. PSO only evolve their social behavior and accordingly their movement towards the best solutions [3]. The search can be carried out by the speed of the particle during the development of several generations, and only the most optimist solution can pass their information over iterations.

The main properties of collective behavior, are a few of the characteristics that make algorithms based in swarm intelligence so effective [15]:

- Homogeneity: Every particle in the swarm has the same behavioral model. The swarm moves without a fixed leader, even if a temporary leader appear.
- Locality: Its nearest swarm mates only influence the motion of each individual.
- Collision Avoidance: Avoid colliding with nearby swarm mates.

- Velocity Matching: Attempt to match velocity with nearby swarm mates.
- Flock Centering: Attempt to stay close to nearby swarm mates.

PSO has proven to have good results in path planning to perform obstacle avoidance [8], [17], [14]. An algorithm for path planning for mobile robot using PSO with mutation operator is proposed in [12]. Its strategy consists in three steps: First the MAKLINK graph is built to describe the working space of the mobile robot. Then, the Dijkstra's algorithm is used to obtain a sub-optimal path and finally PSO is adopted to get the optimal path. In order to generate enough particles in PSO, it has to be chosen a parameter $t_i \in [0, 1]$ for each free-link in the MAKLINK graph. A free-link is defined as a line whose two ends are either corners of two obstacles or one of them is a corner and the other is a point on a working space boundary wall. Therefore, complex environments containing large number of obstacles mean great increase in the complexity of the whole path planning system. Besides, the mutation operator used to avoid local minimum problems consists in a random strategy and it does not make good use of the evolution of population and this approach produces non-smooth paths.

In 2010, an algorithm of improved PSO was applied in mobile robotic path planning [10]. This approach proposes a grid method to model the path space which results in nonsmooth paths, it also includes mutation and crossover operators in order to avoid local minimum. But in addition to avoidance of local optimum these two steps add to PSO algorithm computational complexity.

A smooth path planning of a mobile robot using Stochastic PSO is implemented in [2]. It uses a kind of cubic spline in which coefficients are trained with PSO in order to produce smooth paths. The fitness function with respect to obstacle avoidance makes necessary to know each center of each obstacle in the environment, as well as the calculation of critical points defined as points on the trajectory that are at the minimum distance to an obstacle. So, the fitness function includes these calculations for each one of the M obstacles in the environment.

In this paper it is proposed an approach using Particle Swarm Optimization (PSO) technique to train a Radial Basis Function Network (RBF) used to solve a function approximation problem in order to obtain a smooth path of a mobile robot through an environment containing static obstacles. RBF networks were chosen to be trained with PSO to meet the following objectives: 1) to keep relatively simple the implementation of the path planning system and therefore 2)to get an efficient path planner and 3) To obtain smooth paths making use of their excellent capabilities as function approximators.

Next two sections, II, III show the methods which comprise the path planner system. Section IV describes the particle modeling and the fitness function implementation. Last section, V is devoted to show the simulation results and parameter tuning of the PSO-RBF path planning algorithm.

II. Particle Swarm Optimization, PSO

PSO exploits a population of potential solutions, each solution consists of a set of parameters, representing a point in a search space$A \subset \Re^n$. The population of solutions is called swarm and each individual from a swarm is called a particle. A swarm is defined as a set of N particles. Each particle i is represented as a D-dimensional position vector $x_i(t)$. The particles are assumed to move within the search space A iteratively. This is done by adjusting their position using a proper position shift, called velocity $v_i(t)$.

Each iteration t, the velocity changes by applying equation (1) to each particle.

$$v_i(t+1) = \omega v_i(t) + c_1\varphi_1(P_{ibest} - x_i) + c_2\varphi_2(P_{gbest} - x_i), \quad (1)$$

where φ_1 and φ_2 are random variables uniformly distributed within [0,1]; c_1 and c_2 are weighting factors, also called the cognitive and social parameters, respectively; ω is called the inertia weight, which decreases linearly from ω_{start} to ω_{end} during iterations. P_{ibest} and P_{gbest} represent the best position visited by a particle and the best position visited by the swarm till the current iteration t, respectively.
The position update is applied by equation (2) based on the new velocity and the current position.

$$x_i(t+1) = x_i(t) + v_i(t+1). \quad (2)$$

The basic algorithm is as follows:

1) Initialize each particle of the swarm, with random values for position and velocity in the search space.
2) Evaluate member of the swarm with the fitness function.
3) Compare the value obtained from the fitness function from particle i, with the value of P_{ibest}. If the value of the fitness function is better than the P_{ibest} value, this new value takes the place of P_{ibest}.
4) If the value in P_{ibest} is better than P_{gbest}, then $P_{gbest} = P_{ibest}$.
5) Modify the velocity and position of the particles using equations (1) and (2), respectively.
6) If the maximum number of iterations or the ending condition isn't achieved, return to step 2.

To solve the uncontrolled increase of magnitude of the velocities (swarm explosion effect), is often used to restrict the velocity with a clamping at desirable levels, preventing particles from taking extremely large steps from their current position [11].

$$v_{ij}(t+1) = \begin{cases} v_{max} & if v_{ij}(t+1) > v_{max}, \\ -v_{max} & if v_{ij}(t+1) < -v_{max} \end{cases}$$

Although the use of a maximum velocity threshold improves the performance, by controlling the swarm explosions,

without the inertia weight, the swarm would not be able to concentrate its particles around the most promising solutions in the last phase of the optimization procedures; even if a promising region of the search space would be detected, no further refinement would be able, with the particles oscillating on wide trajectories around their best positions [11].

III. Radial Basis Functions

The structure of an RBF neural network consists on three layers [13], as seen in Fig. 2:

- The input nodes layer.
- The hidden neuron layer, that provides a nonlinear transformation from the input through RBFs.
- The output layer, is the linear combination from the outputs from the hidden layer.

When a RBF neural network is used to perform a complex pattern classification task, the problem is basically solved by first transforming it into a high dimensional space in a nonlinear manner and then separating the classes in the output layer. The underlying justification is found in Cover's theorem on the separability of patterns, which, in qualitative terms, may be stated as follows:

A complex pattern-classification problem, cast in a high-dimensional space nonlinearly, is more likely to be linearly separable than in a low-dimensional space, provided that the space is not densely populated.

From Cover's theorem, we can state that a non linear mapping is employed to transform a nonlinearly separable classification problem into a linearly separable one with high probability.
The neural network is designed to perform a nonlinear mapping from the input space to the hidden space, followed by a linear mapping from the hidden space to the output space [6].
The radial basis equation for interpolation consists in selecting F as:

$$F(x) = \sum_{i=1}^{N} w_i \varphi(\parallel x - C_i \parallel), \tag{3}$$

$$\varphi(r) = exp(-\frac{r^2}{2\sigma^2}). \tag{4}$$

Where,

- $\varphi()$: Is the set of N nonlinear functions, known as radial basis. Equation (4) is a gaussian function, but there are other type of RBFs like multicuadratics and inverse multiquadratics.
- w_i : Represents the weight of the connection between the neuron i from the hidden layer with the output layer.
- $\|\|$: Represents the euclidean norm.
- C_i : Are the centers of each of the gaussian functions; where $C_i \in \Re^p, i = 1, 2, 3, ..., N$.

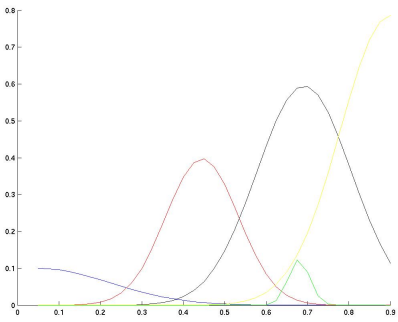

Fig. 1. Set of gaussian functions that conform the RBF neural network output

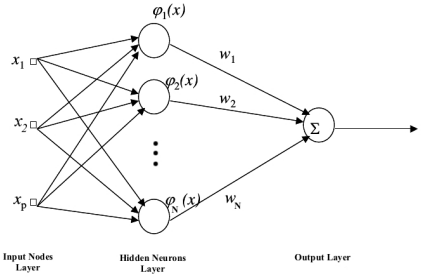

Fig. 2. RBF Neural Network Structure

- σ^2 : Represents the variance.
- x : Is the input signal.

IV. Particle Description and Fitness Function

Path planning for car-like mobile robots can be realized through a search space of functions [7], [8]. In this case we reduce the space to a sub-space of RBFs.
Usually, RBFs are trained by algorithms like k-means and a totally supervised learning method; but by using PSO we substitute this phases.
Each particle in the swarm is composed by the Ci, σ^2 and wi parameters to be used by the RBF function, to approximate a function that passes by a predefined set of points; this set of points is taken as the RBF input points for the input nodes layer. They represent trajectory constraints, i.e. coverage control points can be seen as places where is desirable for the robot to explore in the environment.

The input points are not fixed, excepting for the first and the last point, that are the start and goal points respectively; actually they vary randomly with each PSO iteration in a range of 1 map state, in any direction respect to the first set of input points; this helps to make the obstacle avoidance easier, especially when one of the points is near an obstacle and it transform the problem from interpolation to function approximation.
What defines the efficiency of PSO is the fitness function

(aside of the parameters tuning). We must find a fitness function that allows to find a collision free path for the mobile robot to follow.
The global minimum should correspond whit a smooth and safe (collision free) trajectory, and an unsafe trajectory should be penalized by the fitness function.
For this approach, the following fitness function was designed:

$$f = RMSE + \frac{l}{sf} + c \tag{5}$$

Where $RMSE$ is the root mean square error (equation (6)), l is the length of the path (equation (8)), sf is a scaling factor (so that the length of the path could have more or less influence on the final result) and c (equation (10)) is the collision variable (it takes a higher value proportional to the number of states that a path crosses that are occupied by obstacles or near them).

$$RMSE(f) = \sqrt{\frac{\sum_{k=1}^{n}(e_k)^2}{n}} \tag{6}$$

$$e_k = y_k - f(x_k), \quad where\ f(x_k) \approx y_k. \tag{7}$$

$$l = \sum_{k=1}^{n-1} \sqrt{(x_{k+1} - x_k)^2 + (f(x_{k+1}) - f(x_k))^2} \tag{8}$$

$$S = \{(x, y) | (x, y) \in obstacles\, range\} \tag{9}$$

$$c = \sum f_m(s)\, where\ s \subseteq S \tag{10}$$

$$H = \begin{bmatrix} 0.0001 & 0.0006 & 0.0012 & 0.0006 & 0.0001 \\ 0.0006 & 0.0049 & 0.0099 & 0.0049 & 0.0006 \\ 0.0012 & 0.0099 & 0.0200 & 0.0099 & 0.0012 \\ 0.0006 & 0.0049 & 0.0099 & 0.0049 & 0.0006 \\ 0.0001 & 0.0006 & 0.0012 & 0.0006 & 0.0001 \end{bmatrix}$$

$$Max(f_m(s)) = 0.02 |\ it's\ an\ obstacle. \tag{11}$$

$$Min(f_m(s)) = 0.0001 |\ barely\ in\ the\ obstacle\ range. \tag{12}$$

The s value from equation (9) is the set of points in the range of an obstacle crossed by the path. As was mentioned before, the map is discretized, so a value is assigned to each state that is within a certain range from an object, like mentioned in (11) and (12), and zero if it's out of range; for this case, the gaussian mask H was used to define the values of the map f_m following the next pseudocode:

for $obstacle = 1 \rightarrow |S|$ **do**
  $(x, y) \leftarrow S(obstacle)$
  for $j = -2 \rightarrow 2$ **do**
    for $i = -2 \rightarrow 2$ **do**
      $f_m(x+i, y+j) \leftarrow MAX(f_m(x+i, y+j), H(i+3, j+3))$
    end for
  end for
end for

V. Simulation results

For all the tests a swarm with 40 particles was used, and the stop condition is that the number of iterations is t=100; each test was repeated 100 times and the average was obtained; the results can be seen in Table I. One first comparison was made between the MSE obtained from the approximated function obtained by this approach against the MSE obtained by MATLABs RBF neural network newrb; as seen in the table, the results obtained were favorable by a good margin.
Also the results illustrated in Figures from 3 to 7, show good results for the path planning problem, all the paths are collision free, pass through or acceptably near all the control points and it is obtained smooth paths for a mobile robot to follow.
It is important to note that approaches like Maklink-graph mutation PSO [12] can not deal with environments with concave obstacles, like the one showed in Fig. 4. If the navigation environment contains concave obstacles it would be necessary to compute something like convex hull of each one of them in order to produce the MAKLINK graph. This step will add complexity to the path planning system.

All the PSO adjustable parameters, where heuristically selected, of which were selected the next values as the set of parameters that gave the best results: $c_1 = 2$, $c_2 = 2.5$, $v_{max} = 0.15$, $\omega_{start} = 1$ and $\omega_{end} = 0.0005$.
The value of sf in the fitness function was selected also heuristically, for this case $sf = 37$; the value has to be high to prevent the dominating influence of the length in the fitness function, with smaller values the fitness function would favor the shorter paths instead of the paths that pass through the control points and avoid obstacles.
By giving c_2 a higher value than c_1 we are biasing the particles search ability towards P_{gbest}.
The inertia weight ω reduces the perturbations that make the particles walk away from promising positions, the best positions require strong attraction to refine the search results. A value bigger than 1 in $\omega_{start} = 1$ would make the particles spread more in the search space, making them to reach further positions, but it will take longer for the particles to converge in a promising region as ω_{end} tends to approximate ω_{start}. But also selecting a value higher than 1 form ω_{end} would make it difficult for the particles to converge faster.

Map showed in Fig. 8 was used to compare PSO-RBF approach against Maklink graph-Mutation PSO [12] for path planning. Map showed in Fig. 8 includes some coverage control points as blue squares. All the obstacles are modeled as convex polygons and therefore the Maklink graph-Mutation PSO can be applied to find optimal paths. As mentioned in Section I the strategy consists in three steps: First the MAKLINK graph is built to describe the working space of

TABLE I
MSE VALUES OBTAINED FROM AN RBF TRAINED WITH PSO AND MATLAB'S RBF (NEWRB)

	MSE	
	RBF trained with PSO	RBF (MATLAB newrb)
Map 1	1.4236e-04	0.0395139
Map 2	0.0053	0.08875
Map 3	8.5718e-005	0.0738776
Map 4	4.9906e-004	0.065
Map 5	0.0176	0.0726276

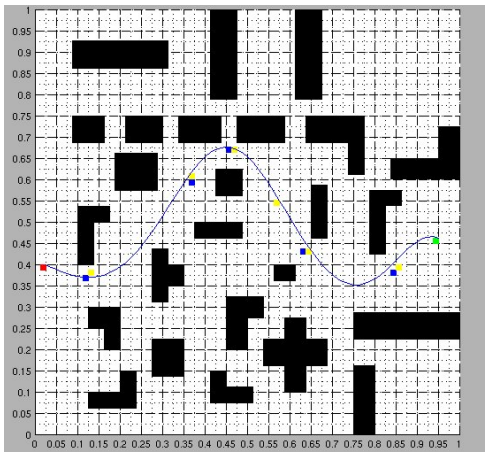

Fig. 3. Simulation Results from Map No. 1. Initial coverage control points of the trajectory are illustrated as blue squares, meanwhile final control points on which obtained trajectory passes on are shown as yellow squares. Smooth and collision free trajectory is obtained regardless of the concavity of the obstacles or the number of these.

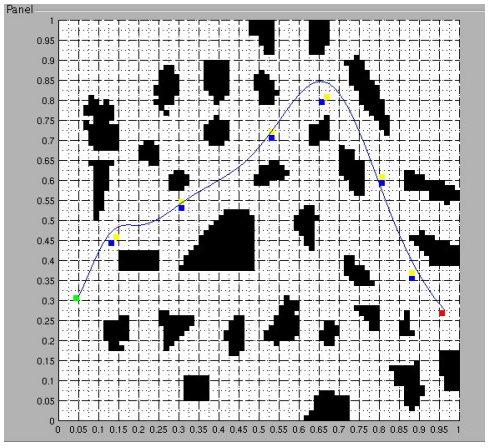

Fig. 4. Simulation Results from Map No. 2.Initial coverage control points of the trajectory are illustrated as blue squares, meanwhile final control points on which obtained trajectory passes on are shown as yellow squares. Smooth and collision free trajectory is obtained regardless of the concavity of the obstacles or the number of these.

the mobile robot. Then, the Dijkstra's algorithm is used to obtain a sub-optimal path and finally PSO is adopted to get the optimal path. In order to generate enough particles in PSO, it has to be chosen a parameter $t_i \in [0, 1]$ for each free-

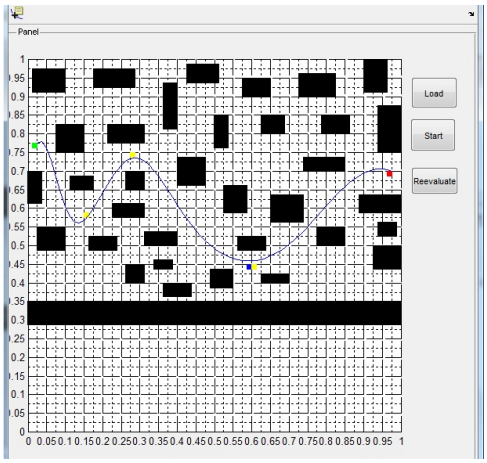

Fig. 5. Simulation Results from Map No. 3.Initial coverage control points of the trajectory are illustrated as blue squares, meanwhile final control points on which obtained trajectory passes on are shown as yellow squares. Smooth and collision free trajectory is obtained regardless of the concavity of the obstacles or the number of these.

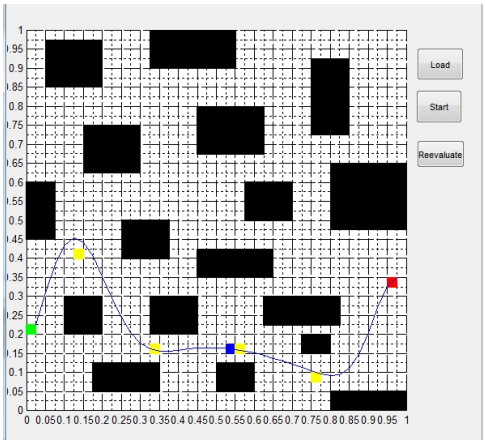

Fig. 6. Simulation Results from Map No. 4.Initial coverage control points of the trajectory are illustrated as blue squares, meanwhile final control points on which obtained trajectory passes on are shown as yellow squares. Smooth and collision free trajectory is obtained regardless of the concavity of the obstacles or the number of these.

link in the MAKLINK graph. A free-link is defined as a line whose two ends are either corners of two obstacles or one of them is a corner and the other is a point on a working space boundary wall. Fig. 9 shows the MAKLINK graph generated from map in Fig. 8. Maklink graph-Mutation PSO was forced to take into account coverage control points by making that Dijkstra's algorithm looked for the sub-optimal path into the set of free links that pass on (or near of) these control points. The resulting sub-optimal path includes $D = 17$ free links and $t_i \in 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9$. All the steps mentioned above make the computational complexity of the Makling graph-Mutation PSO algorithm grows with respect to the number of obstacles in the environment. Large number of obstacles generates MAKLINK graph with large number of nodes and edges and therefore it makes the Dijktra's algorithm takes long time to run. Nevertheless, the path obtained with

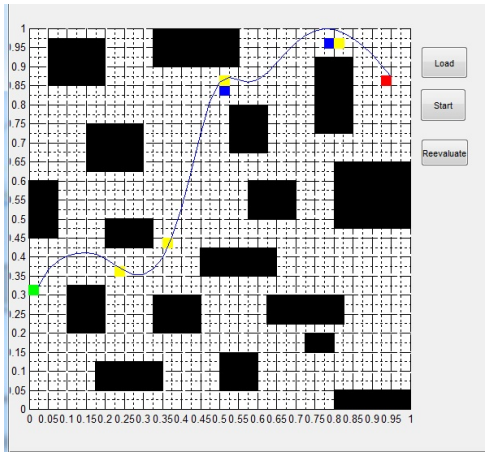

Fig. 7. Simulation Results from Map No. 5. Initial coverage control points of the trajectory are illustrated as blue squares, meanwhile final control points on which obtained trajectory passes on are shown as yellow squares. Smooth and collision free trajectory is obtained regardless of the concavity of the obstacles or the number of these.

this approach is the shortest one that covers (or pass near of) all the control points.

On the other hand, Fig.11 shows the path obtained with the path planner proposed in this paper. This path was obtained with a swarm with 40 particles, evolved by 100 iterations. Although this path is not the shortest one, it is a smooth path easy to be followed by a nonholonomic robot, and our approach needs fewer particles than Maklink graph approach which makes the last one of these, slower than PSO-RBF when the number of obstacles in the environment are numerous and the coverage control points are used.

All simulations were executed in a computer with Windows 7, Intel Core i5, 2.4 GHz, 4 GB RAM, Matlab 2009a. Maklink graph-Mutation PSO algorithm and PSO-RBF algorithm evolved 40 particles with 100 iterations. First approach mentioned took a total time for getting the final path showed in Fig. 10 of $40.56s$, considering all the steps of the approach, since the generation of MAKLINK graph to the training of PSO. Meanwhile our approach took $8.39s$ to get the path showed in Fig.11, although it is not the shortest path is a smooth path which is much easier to follow for a mobile nonholonomic robot than the one showed in Fig.10.

VI. Conclusions and Future Work

This paper presents an approach to solve the path planning problem as an optimization problem using the RBF networks trained with PSO algorithm. A set of trajectory constraints based on coverage control points as input pattern, which can be seen as places where is desirable for the robot to explore were used to approximate functions with PSO-RBF approach, in order to obtain smooth and collision-free paths. Furthermore, a combined fitness function is proposed with respect to three requirements: (i) achieve minimum mean square RBF- function approximation error ; (ii) avoid collisions and (iii) minimize the length of the obtained path .

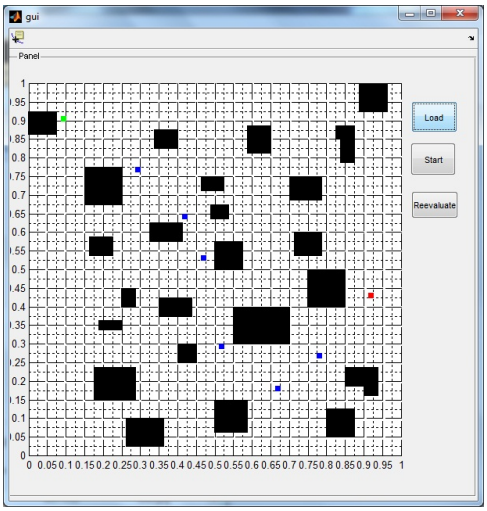

Fig. 8. Map with convex obstacles and coverage control points showed as blue squares.

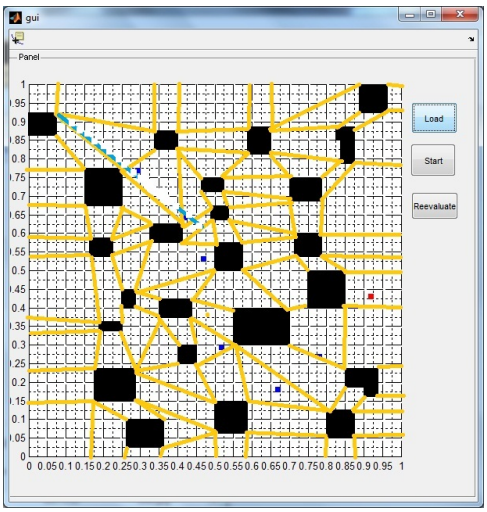

Fig. 9. Maklink graph generated from map showed in Fig.8

Results in simulation environments show that our approach obtains smooth and collision free trajectories regardless of the concavity of the obstacles or the number of these taking advantage of the using of coverage control points as trajectory constraints.

Future work includes the development of an hybrid algorithm which combines PSO-RBF with some PSO-based obstacle avoidance methodology in order to solve motion planning in dynamic environments

References

[1] J. Barraquand, B. Langlois, and J.C. Latombe. Numerical potential field techniques for robot path planning. *Systems, Man and Cybernetics, IEEE Transactions on*, 22(2):224–241, 1992.

[2] Xin Chen and Yangmin Li. Smooth path planning of a mobile robot using stochastic particle swarm optimization. In *Proceedings of the IEEE International Conference on Mechatronics and Automation*, pages 1722–1727, Luoyang, China, 2006.

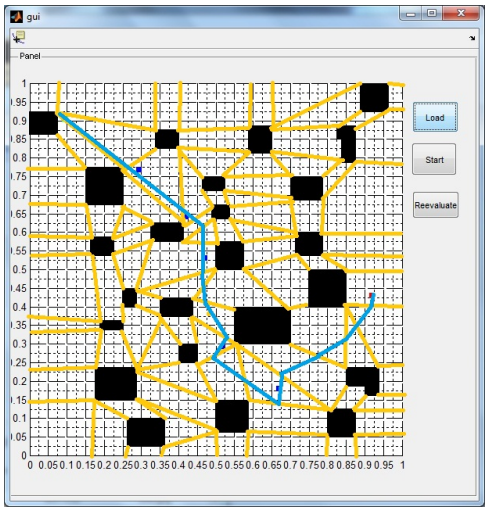

Fig. 10. Path obtained with Maklink-graph mutation PSO approach. This path was obtained with Maklink-graph, Dijkstra's algorithm and evolving 40 particles with PSO in a total time of $40.56s$. The path obtained with this approach is the shortest one that covers (or pass near of) all the control points.

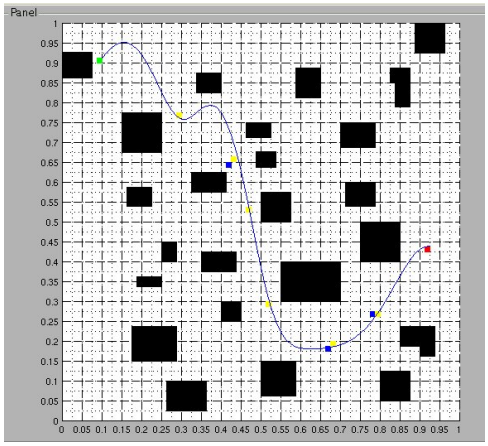

Fig. 11. Smooth Path obtained with PSO-RBF approach. It was obtained evolving 40 particles in a total time of $8.39s$.

[3] E. Elbeltagi, T. Hegazy, and D. Grierson. Comparison among five evolutionary-based optimization algorithms. *Advanced Engineering Informatics*, 19(1):43–53, 2005.

[4] M. Gerke. Genetic path planning for mobile robots. In *Proceedings of the American Control Conference*, pages 2424–2429, San Diego California, 1999.

[5] P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107, 1968.

[6] S.S. Haykin. *Neural networks and learning machines*, volume 3. Prentice Hall, 2009.

[7] Y. Hu and S. X. Yang. A knowledge based genetic algorithm for path planning of a mobile robot. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 4350–4355, New Orleans, 2004.

[8] M. Hua-Qing, Z. Jin-Hui, and Z. Xi-Jing. Obstacle avoidance with multiobjective optimization by pso in dynamic environment. In *Proceedings of International Conference Machine Learning and Cybernetics*, volume 5, pages 2950–2956, Luoyang, China, 2005.

[9] James Kennedy and Russell C. Eberhart. Particle Swarm Optimization. In *Proceedings of IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948, Washington, DC, USA, November 1995. IEEE Computer Society.

[10] Wei Li and Gai-Yun Wang. Application of improved pso in mobile robotic path planning. In *Proceedings of the International Conference on Intelligent Computing and Integrated Systems (ICISS) 2010*, pages 45–48, Guilin, 2010.

[11] K.E. Parsopoulos and M.N. Vrahatis. *Particle swarm optimization and intelligence: advances and applications*. Information Science Reference, 2010.

[12] Yuan-Qing Qin, De-Bao Sun, Ning Li, and Yi-Gang Cen. Path planning for mobile robot using the particle swarm optimization with mutation operator. In *Proceedings of the Third International Conference on Machine Learning and Cybernetics 2004*, pages 2473–2478, Shanghai, 2004.

[13] E. N. Sánchez and A. Alanis. Redes neuronales: conceptos fundamentales y aplicaciones a control automático. *Cinvestav Unidad Guadalara. Editorial Prentice Hall*, 2006.

[14] M. Saska, M. MacaÌs, L. Preucil, and L. Lhotska. Robot path planning using particle swarm optimization of Ferguson splines. In *Emerging Technologies and Factory Automation, 2006. ETFA'06. IEEE Conference on*, pages 833–839. IEEE, 2006.

[15] V. Selvi and R. Umarani. Comparative Analysis of Ant Colony and Particle Swarm Optimization Techniques. *International Journal of Computer Applications IJCA*, 5(4):1–6, 2010.

[16] A. Stentz. Optimal and efficient path planning for unknown and dynamic environments. Technical report, DTIC Document, 1993.

[17] Li W., Yushu L., Hongbin D., and Yuanqing X. Obstacle-avoidance path planning for soccer robots using particle swarm optimization. In *Proceedings of IEEE International Conference on Robotics and Biomimetics, ROBIO 2006*, pages 1233–1238, 2006.

[18] Yunfeng Wang and G. S. Chirikjian. A new potential field method for robot path planning. In *Proc. IEEE Int. Conf. Robotics and Automation ICRA '00*, volume 2, pages 977–982, 2000.