Robotic Path Planning using Hybrid Genetic Algorithm Particle Swarm Optimization

Rahul Kala*, Anupam Shukla, Ritu Tiwari

Soft Computing and Expert System Laboratory

Indian Institute of Information Technology and Management Gwalior,

Morena Link Road, Gwalior, Madhya Pradesh – 474010, India

* Corresponding Author:

rahulkalaiiitm@yahoo.co.in

Ph +44 (0) 7424752843

http://rkala.99k.org/

Citation: R. R. Janghel, R. Tiwari, R. Kala, A. Shukla (2012) Breast Cancer Data Prediction by Dimensionality Reduction Using PCA and Adaptive Neuro Evolution, *International Journal of Information Systems and Social Change*, 3(1): 1-9.

Final Version Available At:
http://www.inderscience.com/info/inarticle.php?artid=48756

Abstract

The problem of robotic path planning has always attracted the interests of a significantly large number of researchers due to the various constraints and issues related to it. The optimization in terms of time and path length and validity of the non-holonomic constraints, especially in large sized maps of high

resolution, pose serious challenges for the researchers. In this paper we propose Hybrid Genetic Algorithm Particle Swarm Optimization (HGAPSO) algorithm for solving the problem. Diversity preservation measures are introduced in this applied evolutionary technique. The novelty of the algorithm is threefold. Firstly the algorithm generates paths of increasing complexity along with time. This ensures that the algorithm generates the best path for any type of map. Secondly the algorithm is efficient in terms of computational time which is done by introducing the concept of momentum based exploration in its fitness function. The indicators contributing to fitness function can only be measured by exploring the path represented. This exploration is vague at start and detailed at the later stages. Thirdly the algorithm uses a multi-objective optimization technique to optimize the total path length, the distance from obstacle and the maximum number of turns. These multi-objective parameters may be altered according to the robot design.

Keywords: Robotic Path Planning, Robotics, Evolutionary Algorithms, Hybrid Genetic Algorithm Particle Swarm Optimization, Momentum, Diversity Preservation, Information Technology

1. Introduction

Robotics is one of the very active fields of research. The vast amount of ongoing research in the domain is the result of the immense rise in need in automation. Now we expect many things to be done by robots which were previously performed by humans. The use of robots to clean houses, surveys, factory operations, etc are now turning into reality. All intelligent robotic systems use path planning to plan the path of the robot from one point to the other. Once the path has been found by the devised algorithm, robotic controllers are used for the physical movement of the robot from the source to the destination. This enables carrying of various intelligent tasks in autonomous robots. The notion of path planning depends a lot upon the scenario of use, robotic design, degrees of freedom, robotic capabilities, constraints, etc. The maximum allowable speeds, acceleration, robotic shape are some constraints that need to be considered.

Another constraint is the validity of non-holonomic constraints in the worked over path.

This paper assumes a simple modeling scenario that is commonly found and studied in robotic literature. We have a grid based map where each cell corresponds to the presence or absence of obstacle. There is a single source and a single destination that is known in advance. The goal is to move the robot from the source to the destination without colliding with any of the obstacles.

The entire problem of robotic path planning can be easily framed as a problem of Multi-Objective Optimization where the aim is to find the optimal value of a set of objective functions. The problem may further have numerous constraints that need to be fulfilled. Here we are required to optimize the total path length traversed by the robot. At the same time we need to avoid the robot being too close to any obstacle for validity of the non-holonomic constraints after path has been smoothened by some technique. The other factor we consider is the number of turns in the path of the robot henceforth referred as the path complexity.

Complexity in the problem of robotic path planning may be indicated by various parameters depending upon their interpretation. In solutions generated by A* algorithm, the map dimensions indicate the problem complexity. For behavioral solutions the number of behavioral rules or weights, and for evolutionary systems the genomic length is an indicative of the complexity. These more or less convey the same meaning for the same problem. In this problem we assume the complexity denotes the maximum number of turns in the computed robotic path. The solution of the robotic path problem may be very simple or may be very complex. Lesser turns naturally mean a greater scope for the robot to traverse the path with greater speeds for a better drive at less time. Also this may denote a straighter and hence shorter path. The constraints include the feasibility of the path and the avoidance of obstacles in the path.

The recent advancements into the field of robotics and the use of robots in many new scenarios have resulted in the generation of large maps of very high dimensions. As a result many of the traditionally used algorithms and approaches including A* algorithm (Shukla et al. 2008), Dynamic Programming and other graph algorithms fail to solve the problem within time constraints. This has led to the use of evolutionary algorithms to solve the problem. Evolutionary algorithms take an inspiration from the natural evolutionary processes to generate near optimal solutions in finite time. These algorithms use a set of solutions or the population pool and iteratively solve the problem where each iteration is called a generation. The lower generation individuals mate to generate more fit higher generation individuals. The use of swarm intelligence is another positive development in solving these problems.

The limitations in convergence of both evolutionary and swam algorithms lead to the hybridization of both these approaches giving rise to Hybrid Genetic Algorithms Particle Swarm Optimization (HGAPSO) (Juang, 2004). This algorithm divides the whole population pool into two halves. The first half consists of the fitter individuals, which are the elite individuals. Only these individuals participate in the evolution, the others die off. The fitter half generates the offspring by using Particle Swarm Optimization (PSO). These resultant individuals directly go into the next generation. At the same time half of the next generation population is made by the Genetic Algorithm (GA) that operates on the same individuals generated by the PSO.

Diversity preservation is of a special matter of concern for the proposed algorithm. This is carried out separately in both the GA and PSO. The GA uses a restricted selection criterion where two individuals are only allowed to crossover when the separation between them is below a specified threshold (η) (Badran and Rockett, 2007; Squillero and Tonda, 2008). For PSO, an instance of Fitness Distance Ratio based PSO (FDR-PSO) is used for diversity preservation (Peram and Veeramachaneni, 2003; Veeramachaneni et al., 2003). In this algorithm the modification of every particle has an added term that checks for the performance of the neighboring particles. Using this algorithm, the velocity

v_{id} update for d^{th} dimension for any particle *i* with position x_{id} is done by equation (1). The update on position is given by the conventional equation.

$$v_{id} = w.v_{id} + c_1.rand.(p_{id} - x_{id}) + c_2.rand.(p_{gd} - x_{id}) + c_3.rand.(p_{nd} - x_{id}) \quad (1)$$

Here *w* is the inertial factor, c_1, c_2 and c_3 are PSO constants, *rand* is a random number, p_{id} is the best solution in the path of particle, p_{gd} is the globally known best path, p_{nd} is the position of best neighboring particle *j* that maximizes term given in equation (2).

$$\frac{fitness(p_j) - fitness(x_i)}{|p_{jd} - x_{id}|} \quad (2)$$

A variety of methods have been used for solving the problem of robot path planning which include A* Algorithm (Shukla et al., 2008), D* algorithm, Artificial Neural Networks (Yang and Meng, 2000; Kala et al., 2009),Fuzzy Systems (Pradhan et al. 2009), ANFIS, Potential Fields (Tsai et al., 2001; Pozna et al., 2009), etc. A comparison of some of the algorithms including hybrid algorithms can be found in the works of Hui and Pratihar (2009).

Because of the underlying complexity of the problem, a significant use of evolutionary techniques to solve the problem is also prevalent. A variety of mechanisms are applied to customize the conventional genetic algorithm to give an enhanced performance in solving the problem. Kala et al. (2009) represented a graph path as a genetic individual. The genetic operators of mutation and crossover were designed to work over these paths. Other customizations include the use of restrictions in genetic operations to ensure feasibility of the solution generated at every step (Alvarez, 2004). Another representation technique uses a set of sparse points from source to destination with the line joining source and destination as one of the axis (Toogood et al., 1995; Han et al., 1997). Many times the corners of the obstacles may be numbered and the individual is a sequence of numbers that the robot visits (Sadati and Taheri, 2002). Tu et al. (2003) give the concept of variable length representation of

chromosome. The robot visits these points in a sequential manner. The use of Bezier curve is also done that caters to the non-holonomic constraints in the path generated by the evolutionary algorithm (Jolly et al., 2009).

Many times the map may be of a very large dimensionality and hence it may be very difficult to work with such a graph. For such cases the problem of planning is usually solved in a hierarchical manner which is known as Multi-Resolution Path planning. This may represent the map in multiple resolutions using a variety of representation techniques (Kambhampati and Davis, 1986; Urdiales et al., 1998). This further made hierarchical path planning algorithms possible (Wang et al. 2002). Other related works using evolutionary computation include (Doitsidis et al., 2009). Here the authors make use of optimization power of the evolutionary algorithms to evolve a fuzzy based robotic controller. The GA here tries to evolve the optimal set of rules. The algorithm of Dittrich et al. (1998) is another novel work that presents the application of Genetic Programming for the problem with the fitness being measured in a combination of simulator and physical robot.

Chen and Chiang (2003) developed an adaptive algorithm by the use of Fuzzy Neural Networks and Multi-Objective Genetic Algorithms. Their solution generated and adapted rules to move a robot. Another benchmark work in the use of hybrid algorithms to solve the problem include the work of Xiao et al. (1997). Their solution separately modeled the static and the dynamic obstacles in the robotic map. These were catered to using separate off-line and on-line planning techniques. The planning was based on evolutionary algorithms. Another concept to plan the robot is with the use of sensors placed on map that can provide information to the robot regarding the closeness of obstacles and goal (O' Hara, 2008). Other good ways to solve the problem include Rapidly-exploring Random Tree (RRT) algorithm (Cortes et al., 2008) and Fast Marching Method with Vornoi Transform (Garrido et al., 2009).

Kala et al. (2010a) presented an interesting hierarchical algorithm for solving the same problem. Evolutionary algorithms are usually only used for high resolution

dynamic maps. The authors used hierarchical evolutionary algorithm for dynamic path planning. The master evolutionary algorithm considered static map of a coarser resolution. The finer path planning with dynamic obstacles was done with a slave evolutionary algorithm. The passage of individuals between evolutionary runs ensured maximization of knowledge of computed path. In another work Kala et al. (2010b) presented fusion of A* algorithm and Fuzzy Inference System for path planning of robots. They used a probabilistic form of A* algorithm on a coarser level that tried to generate approximate path of robot, ensuring avoiding paths congested with obstacles. A fuzzy planner worked over this path to ensure generation of an optimal path that also caters to the validity of the non-holonomic constraints.

This work is an extension to our earlier work (Kala et al., 2011). Here we had introduced the word momentum in exactly the same meaning as we do here. The problem of path planning was done using a Simple Genetic Algorithm. We observed that the algorithm was able to generate good results in finite time duration for the maps presented. The studied limitation of the work however was the possibility of the algorithm to return sub-optimal paths as many good individuals may be observed to be in-feasible at higher generations of the algorithm. This is possible with various critical maps in which a blockage of path is only visible at a very fine resolution of the map. In this paper we further extend the algorithm by the use of HGAPSO, which is a hybrid of GA and PSO. Diversity preservation is added to both the individual algorithms of GA and PSO separately. This further ensures the generation of paths with better characteristics as defined by the fitness function.

This paper is organized as follows. In section 2 we present the concept of momentum. Section 3 talks about the algorithmic framework of the problem. Section 4 talks about the role and importance of variable parameters. Section 5 presents the results. We give the conclusion remarks in section 6.

2. Momentum

At any time in the entire evolutionary process of the algorithm, we are required to check the feasibility of the path, while computing its fitness. Now we need to traverse point-by-point to check this feasibility, which is computationally very expensive. In place of a point-by-point check, the momentum claims a speedy check of the given path. We hence mean that we would vaguely examine the path, with the vagueness being given by the value of the momentum. Consider a path as a set of points (as we shall see later in problem representation). Suppose in its traversal, the robot is supposed to travel between points *X* and *Y*. We only check for the presence of obstacle between points *X* and *Y* for the set of points Q given by equation (3)

$$Q = \{\mathrm{X}, \mathrm{X} + m \ \frac{(\mathrm{Y}-\mathrm{X})}{|\mathrm{Y}-\mathrm{X}|} \ t, \mathrm{Y}\} \qquad (3)$$

Here *Y* and *X* are the consecutive points used for the definition of the robot path

$\frac{(Y-X)}{|Y-X|}$ is the unit vector in direction of *Y-X*, *m* is momentum, and *t* is the traversal step (*t=1, 2, 3…*) such that $\mathrm{X} + m \ \frac{(\mathrm{Y}-\mathrm{X})}{|\mathrm{Y}-\mathrm{X}|}$ lies between X and Y. The number of points checked (or the number of feasible values of t) depends upon the value of momentum m. A smaller value of momentum means a larger number of values of t may make the intermediate points lie between X and Y and all of these are checked for feasibility. It may be easily seen that momentum is inversely proportional to processing time. A larger value of momentum means lesser number of points being checked, which reduces the computational time to find the feasibility of the path.

It is further important to realize the disadvantages of large momentum as well. Larger values of momentum may claim a path to be feasible, which may otherwise be infeasible. The infeasibility of the path may many times be easy to remove by some small alterations to it (called *approximate state*). This is when there is no heavy blockage in the path, and even if some obstacle occurs in the

robotic path, it may be easily avoided. In other condition (called *false state*) there may be a heavy blockage in the path, which may make it impossible for the robot to travel even after moderate amendments to its path. Both these states are given in figure 1.

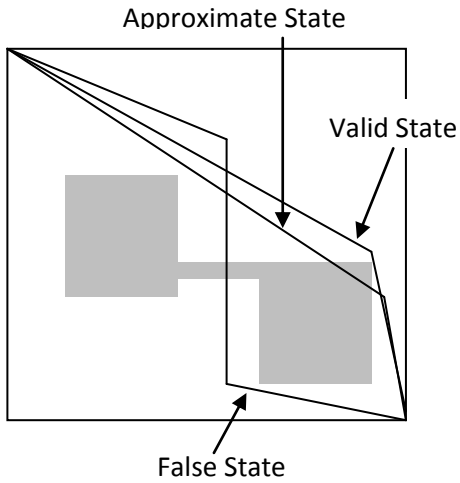


Figure 1: The concept of momentum with *false state* and *approximate state* conditions in feasibility measurement.

3. Algorithm Framework

3.1 Individual Representation

The entire algorithm is basically an evolutionary algorithm. Hence the aim is to evolve the individual that represents the path of the robot. We represent any individual *i* by a set of points $P<P_0, P_1, P_2, P_3, \ldots. P_n, P_{n+1}>$. Here P_0 is the source and P_{n+1} is the goal. Each point is a collection of *x* and *y* coordinates and may be denoted by (x_i, y_i). The *x* axis that we take for this problem is the straight line joining the source and the goal. The *y* axis is perpendicular to the x-axis. Another constraint imposed upon the individual representation is that all the

points are sorted by their x values. Hence an individual is only allowed to move such that it goes near to the goal at every step.

3.2 Fitness Function

The next important factor in the algorithm is the fitness function. Here we try to simultaneously optimize 3 objectives that is total length of the path (*l*), the distance from the closest obstacle (*d*) and the path complexity (*c*). The total fitness is given by equation (4).

$$Fit(P) = \alpha.l + \beta.d + \gamma.c \quad (4)$$

Here *P* is any path or individual of the form $\prod P_i$

α, β and γ are multi-objective constants with the constraint that $\alpha + \beta + \gamma = 1$, *l* is the total path length, *d* is the distance from closest obstacle, *c* is the path complexity

Complexity of a path or a solution is defined as the total number of points in the path or solution. The complexity may lie between *0* (straight line from source to foal) to a maximum permissible value C_{max}.

Distance from closest obstacle (*d*) returns the smallest distance between any pixel in the path of the robot (may not necessarily be the points in the individuals) and the closest obstacle. In order for the robot to easily navigate the path obeying the non-holonomic constraints, it is important that is maintains a sufficient distance from all obstacles. This factor is taken to reduce in a Gaussian manner with the increase of physical distance of robot to nearest obstacle. We assume the factor decays in a Gaussian manner as the actual physical distance increases. Equation (5) gives the value of d for any physical distance *D*.

$$d = e^{-\frac{D^2}{2(aM)^2}} \quad (5)$$

Here *a* is the decay constant governing shape of the Gaussian curve, *M* is the radius constant controlling the effective region of effectiveness of this function. After *M* units of distance, the function returns almost a zero value.

3.3 Algorithm

The basic algorithm used for the problem solving is HGAPSO. This is a hybrid of GA and PSO that solves the problem of convergence. The feasible solutions out of the entire solution pool are divided into two halves. The fitter half participates in the evolutionary process. First PSO is applied. The generated individuals go directly into the next generation. The same individuals are also worked upon by GA that adds to the next generation population. The infeasible solutions are not allowed to participate. Other operators are applied for the production of rest of the solutions for next generations. The overall algorithm is given by figure 2.

The Genetic Algorithm uses the operators of Crossover and Mutation for the generating the next generation individuals. The crossover operation of the algorithm assures that the diversity is maintained. This is done by selecting only those individuals that are separated by a distance smaller than threshold distance (*η)*. Since the two individuals may be of different length, the separation is measured by average separation between the closest lying points in the two individuals. Similarly crossover operation needs to be modified to tackle with the problem of variable individual size. Both the children generated from the crossover operation have got the number of points that are mean of the number of points in the two parents. These many points are randomly given to the children using a scattered crossover technique. Mutation operator used simply re-locates the points of the individual by moving them by small amounts within the map. The PSO optimization technique generates solutions by moving a particle in the input space. The movement of the particle is based upon his past experience, best available solution and the experience of the neighbors. The neighborhood factor is taken from FDR-PSO that is helpful in the diversity

preservation. Here also we only consider the non null positions in the individual representation while modifying the individual position or velocity.

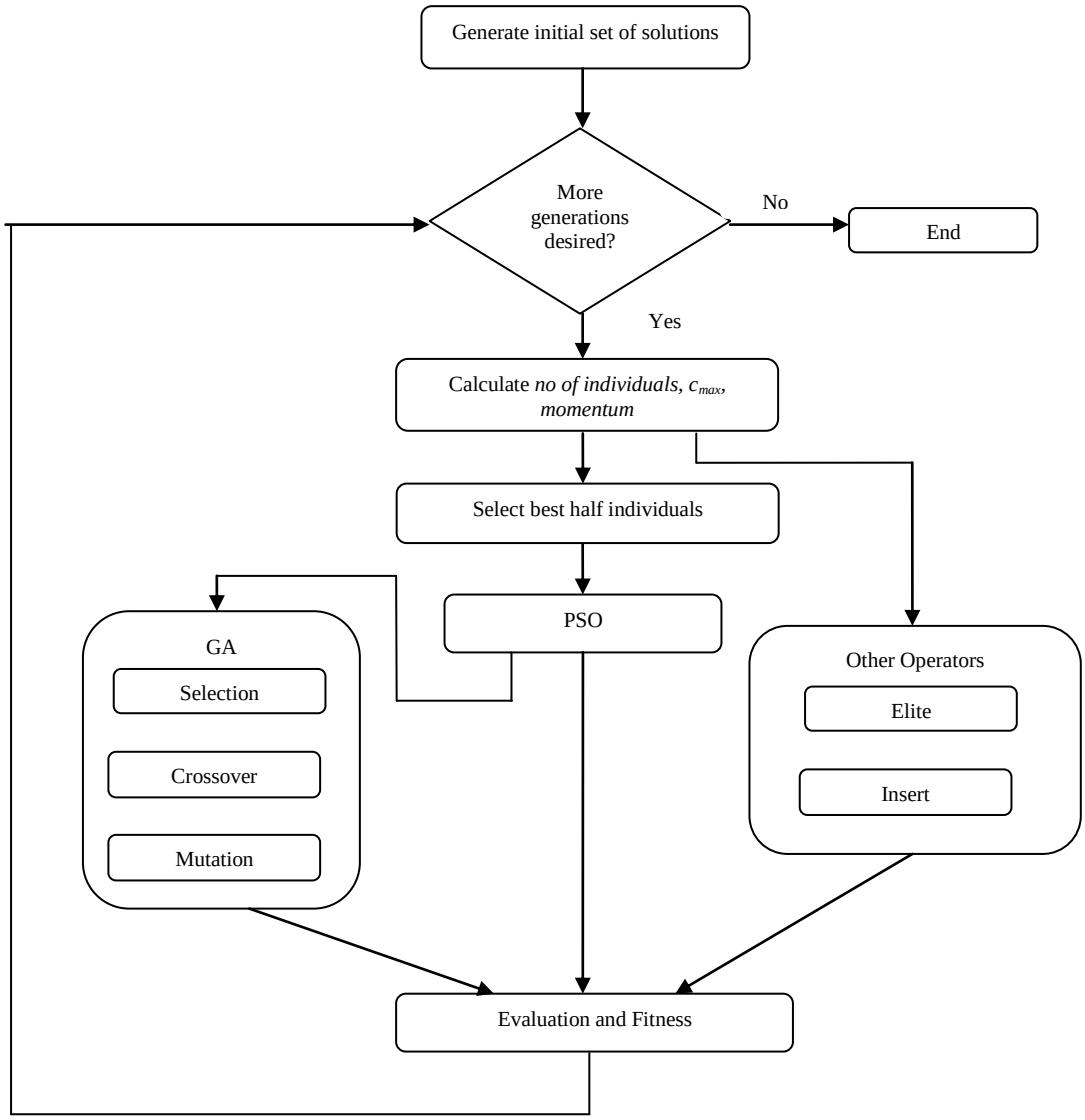


Figure 2: The Hybrid Genetic Algorithm Particle Swarm Optimization (HGAPSO) Algorithm for solving path planning problem.

The other operators used in the algorithm include elite and insert. At any generation we are likely to have a large number of infeasible solutions or individuals. The reasons for this may be the un-optimized stage of the individual, individual has a lesser complexity as per requirement, detection of an obstacle due to decrease in momentum, etc. The GA and PSO work only on the feasible solution set. Hence, depending upon conditions, a reasonable part of the next generation population is generated by another operator called as the inset operator. If there is not even a single feasible solution in the population pool, this operator generates random solutions of the maximum allowable complexity c_{max}. In case the population pool has feasible solutions, this operator may perform either of three possible actions of generating new individuals i.e. generation by a very high mutation of any random feasible solution, or by adding a new point in the path represented by any random feasible solution or by generating random new individual of maximum permissible complexity. All these methods have an equal probability of occurrence. In this manner this operator fills up the left population pool for the next generation.

This operator solves two major problems. The first is that it drives the algorithm faster from stages where it is within a complexity that is less than the complexity needed to solve the problem. Every map has a least complexity below which it would not give any feasible solution. While the algorithm is into these stages, all solutions get passed from lower complexity to higher complexity without wasting time at the lower complexity regions. The second problem that is solved by this operator is cleanliness. The infeasible solutions do not contribute at all. It is better to clean them by replacing them with random solutions of complexity high enough to represent feasible solution.

3.4 Diversity

An important characteristic of the algorithm is of diversity preservation. This is done in regard to the possibility of presence of false state and approximate state shown in figure 1. Suppose that the best known solution of the EA after a number of generations is found to be infeasible due to the presence of

approximate state. To overcome such a condition it is necessary for the algorithm to always retain sufficient number of weakly mutated individuals corresponding to the best few kinds of solutions. Now suppose that the infeasibility was due to false state. For this case the algorithm needs to maintain a high population diversity which is inbuilt in the algorithm.

4. Variable Genetic Parameters

To give the algorithm some adaptability, some of the parameters were kept variable. These are the total number of individuals, maximum allowable complexity c_{max}, and momentum.

There is an increase in the number of individuals with increase in generations due to two reasons. Firstly the rise in search space necessities more individuals. At the first few generations the complexity is low and hence the search space is limited which increases which generations. Secondly a number of individuals are already reserved by low complexity paths and cannot be killed due to diversity preservation measures. The number of individuals *I* for any generation g is given by equation (6).

$$I = I_{min} + (I_{max} - I_{min})e^{-\frac{g^2}{2(cG)^2}} \quad (6)$$

Here I_{max} is the maximum possible number of individuals

I_{min} is the least possible momentum, *g* is the generation number, *c* is the decay constant, *G* is the radius constant or the maximum number of generations possible

Similarly maximum complexity c_{max} is varied as given in equation (7).

$$C_{max} = C\, e^{-\frac{g^2}{2(dG)^2}} \quad (7)$$

Here *C* is the globally maximum possible complexity, *g* is the generation number of EA, *d* is the decay constant , *G* is the radius constant or the maximum number of generations possible

On the same lines the value of momentum is decreased as the generation increases, or as the algorithm proceeds. It may be easily seen that large momentum makes feasibility computation less time complex, but may many times return infeasible paths as feasible. The smaller values of momentum result in higher computation time but return the correct feasibility of any path. Initially the solution pool has all random solutions and even a large momentum would be able to work and give initial idea of the path feasibility. As algorithm proceeds, a precise idea of path feasibility is needed, before it may be further optimized. We make the momentum variable that changes in a Gaussian manner along with the generations. The Gaussian decay of momentum is given by equation (8).

$$m = m_{min} + (m_{max} - m_{min})e^{-\frac{g^2}{2(bG)^2}} \quad (8)$$

Here m_{max} is the maximum possible momentum, m_{min} is the least possible momentum, *g* is the generation number of EA, *b* is the decay constant , *G* is the radius constant or the maximum number of generations possible

5. Results

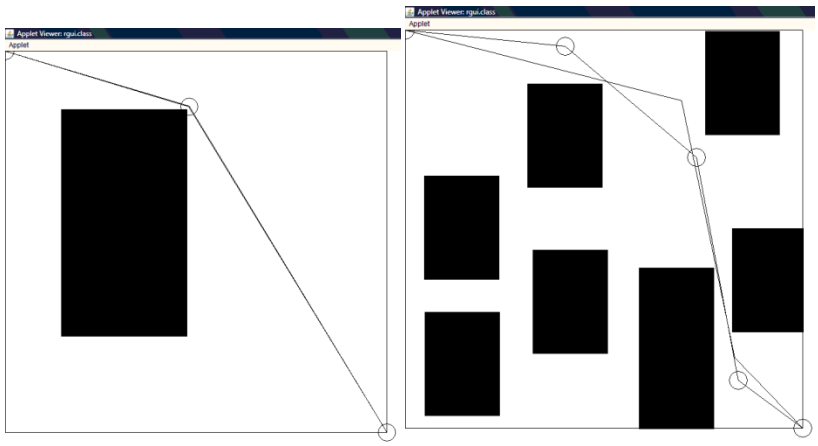
The algorithm was coded and developed as a JAVA module. The entire testing with the algorithm was done on a simulator built by the authors themselves. *JAVA Applets* was used for the depiction of the map and the obstacles. The map was fed into the simulator as a *JPEG* image with the dimensions of the image as the dimensions of map. The white regions denoted the presence of accessible areas and the black denoted obstacles. The algorithm was tested over four benchmark maps that varied from each other. All the different kinds of maps used are given in figure 3. In all the cases the maps were of size *1000 x 1000*. The coordinate axis of the map had *(0,0)* point at the top left. This was the

source specified for all cases. The goal was the bottom right point with the coordinates of *(999,999).*

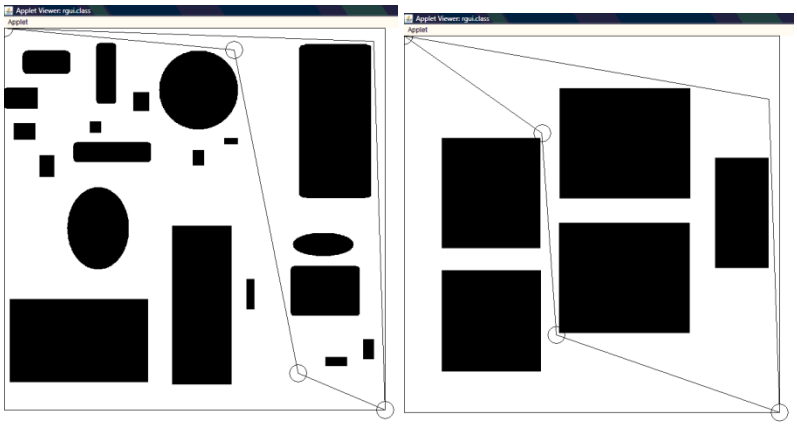
For all experiments the Multi-Objective parameters α, β and γ were specified as *0.33* each. The decay constants *a, b, c* and *d* for each of the Gaussian curves was fixed to be *0.3*. The PSO constants c_1, c_2 and c_3 were fixed to 1. Momentum could decrease from *10* to *1.5*. The mutation rate was fixed to *0.04* and crossover rate was fixed to *0.6*. Elite count was *5%* of the population pool. The value of threshold η of population diversity was fixed to be *0.03*. The radius constant *M* of distance to obstacle was kept as *10*. The maximum complexity c_{max} could vary between *0* and *5*. The number of individuals could vary from *1* to *1000*. The simulation was continued for a total of *500* generations.

The maps and the path traced by the robot for each of these cases are given in figure 3(a-d). In all figures the path denoted by straight line denotes the experimentally computed path as per discussed parameters. The path with a circular legend at corners shows the computed path with no complexity control (α, β =0.5, γ =0).

Figure 3(a) shows a simple map with a single obstacle that is a commonly found situation and tests the capability of the robot to avoid obstacle. The robot was easily able to pass the obstacle and reach the goal while maintaining a comfortable distance of separation from the obstacle. Figure 3(b) is a more complex map with a variety of points. Here as well the robot was able to reach the goal. It may be seen that the path looks longer at the central region (with complexity control). This was because of the heavy penalty of complexity added in the multi-objective function. This path can be greatly smoothened to allow the robot pass the central region at great speeds. The alternate diagonal path traversed in the absence of complexity control was shorter but would add an extra turn. The complexity increased from 2 to 3.



(a) Map 1 **(b) Map 2**



(c) Map 3 **(d) Map 4**

Figure 3: Path traced by robot for various maps with and without complexity control objective function.

Figure 3(c) shows a very long path being traversed by the robot. This was because the diagonal short route was too congested and greatly increased the path length. Without the complexity control also, the robot could not make a diagonal move due to the highly congested environment, but it was able to

figure out another path with a compromise in complexity from 1 to 2. Figure 3(d) is another graph which clearly states our notion of preference between path simplicity and total path length. The simple path would be the choice of fast and compact robots whereas the smaller path would be the natural choice of slow moving robots.

We further study the best fitness value in all the discussed paths and configurations. These for all the graphs are given in figure 4. Closely observing any of the maps, we would be able to observe small oscillations in the fitness value while the algorithm proceeds. In other words the best fitness value increases even after the application of elite genetic operator. This is attributed to the presence of approximate state.

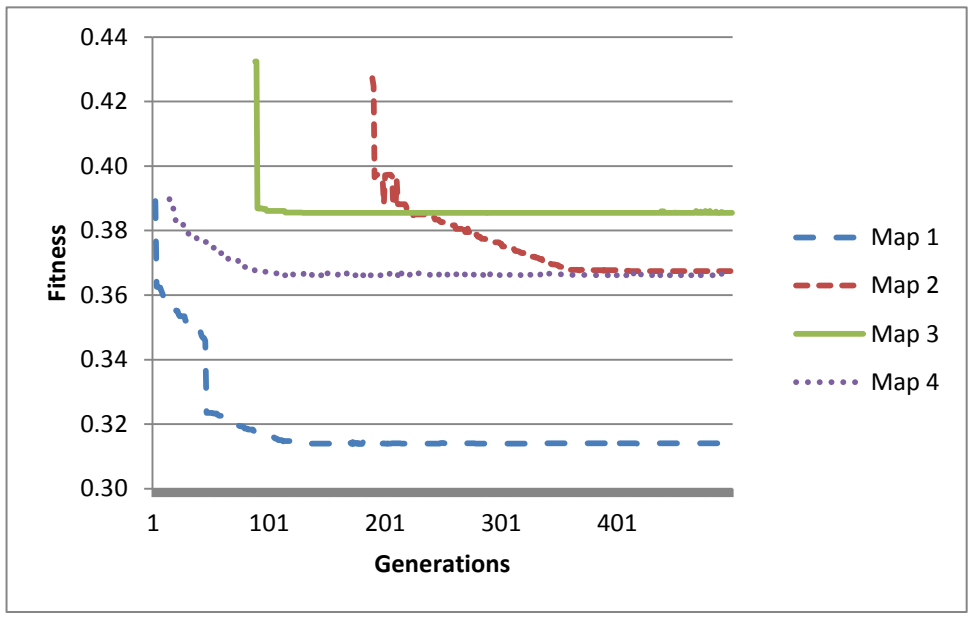


Figure 4: Graph of fitness v/s generations showing convergence in the four maps.

We further plot the total number of feasible solutions at every generation. This would give us an idea of how many individuals actually contribute towards the search of the most optimal path or solution. This is shows in figure 5. Here also oscillatory behavior may be seen which is mainly due to the occurrence of condition 2 or approximate state. Also the randomly generated individuals may be infeasible.

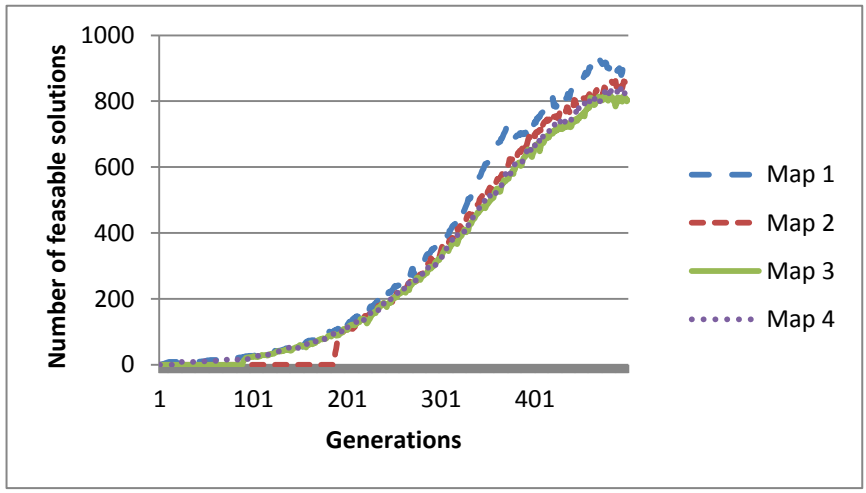


Figure 5: Graph of total number of feasible solutions in population pool v/s generations for four maps.

We also study the execution time requirement between the generations. The execution time per generation is plotted against generations in figure 6. As the generations increase the rise of time may be attributed due to the increase in number of individuals, increase in number of feasible solutions as well as decrease in momentum. Generally infeasible solutions take lesser time as compared to feasible solutions. This is because we stop the traversal as soon as we meet any obstacle on our way. The effort of rest of the journey is saved. The oscillatory nature may again be attributed due to the variation in number of feasible solutions.

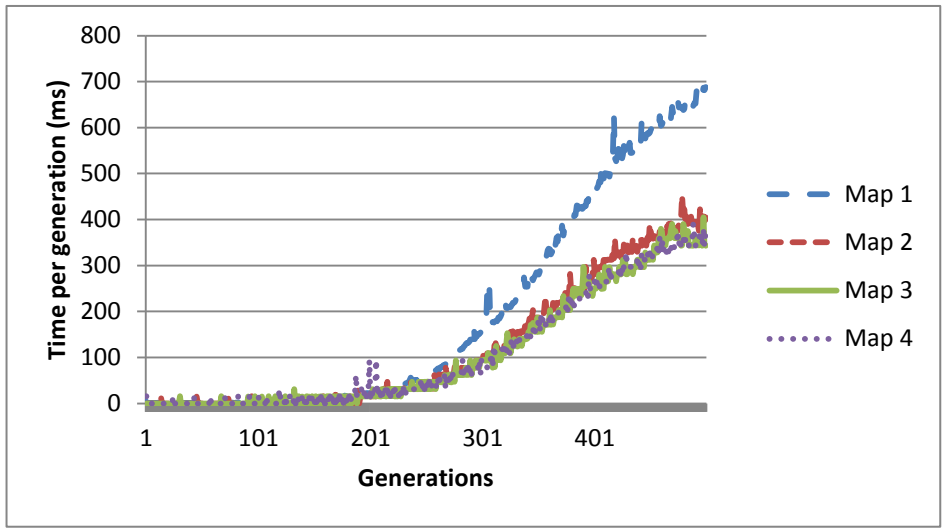


Figure 6: Graph of time of execution per generation v/s generations for four maps.

We earlier stated that an increase in momentum might lead to an incorrect recording of the path feasibility as well as the distance to obstacle (*d*). We try to study the same concept here. A total of *10000* paths of a complexity of *3* were generated for the map *2*. The momentum was varied from *1* to *1000* and all the parameters were measured. The relation of momentum with time of execution is given in figure 7. Figure 8 gives the relation between the momentum and the predicting capability of the fitness function regarding the feasibility of the solution. Figure 9 shows the relation between the momentum and mean deviation (or error in measuring) of the minimum distance from robot to obstacle (*d*).

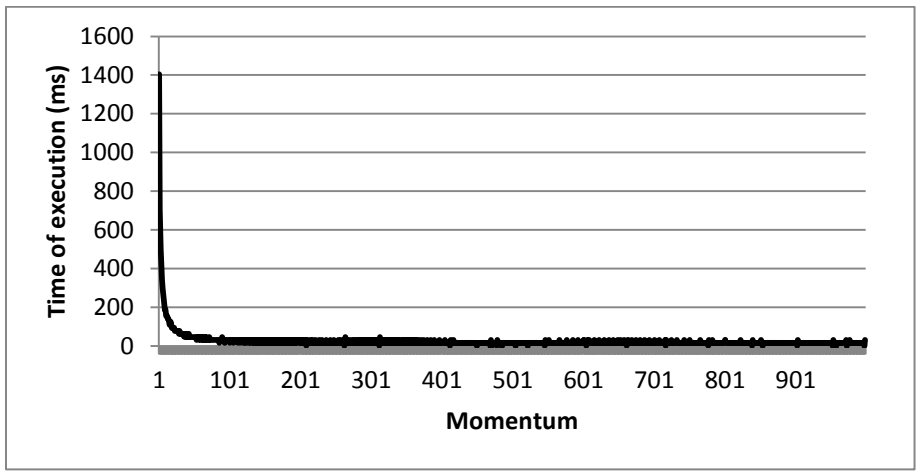


Figure 7: Graph showing the decrease in execution time per generation with an increase of momentum.

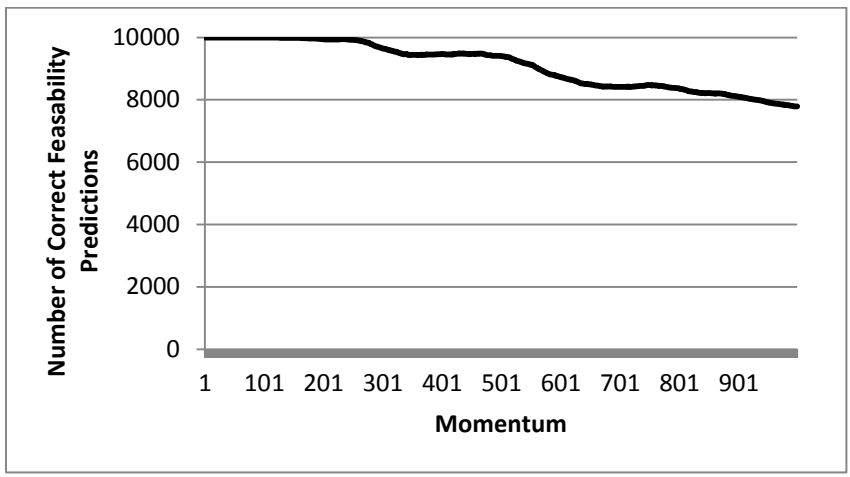


Figure 8: Graph showing the decrease in correctness of feasibility prediction with an increase of momentum.

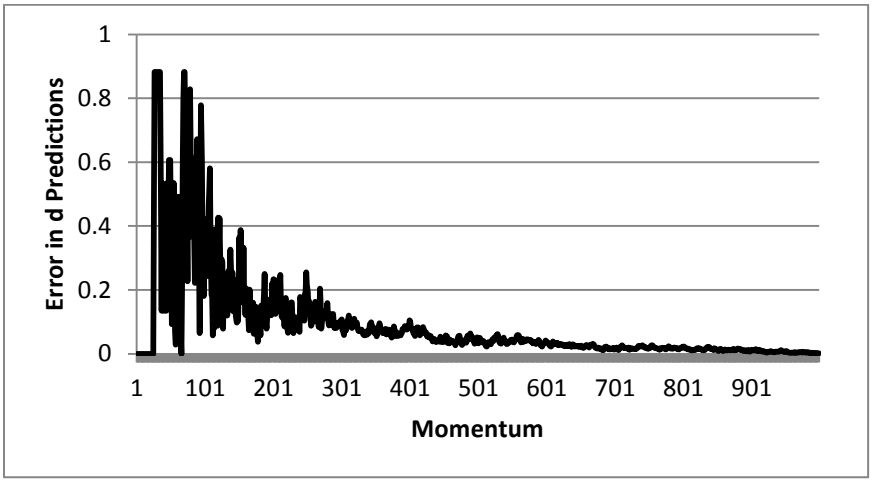


Figure 9: Graph showing correctness in prediction of distance to closest obstacle (*d*) for various values of momentum.

An interesting observation is in figure 9 that reports oscillatory behavior. Here it may be seen that many times larger jumps may lead the algorithm closer to the obstacle. Hence it cannot be generalized that the factor *d* would be having large errors as momentum increases.

6. Conclusion

The paper proposed a novel algorithm that evolved the robotic path using rising complexities using HGAPSO. The combined effects of GA and PSO enabled the generation of path that was optimal based on the set criterion. The basic motivation was that most robotic paths are simple enough with less number of turns. The straightness of the path enables a robot to navigate through the map at high speeds. We also optimized the total path length and the distance to closest obstacle.

The momentum, number of individuals and maximum allowable path complexity increased in a Gaussian manner as the number of generations

increased. This helped in optimization of the entire algorithm by enabling the algorithm to depute only required time for every generation. At the time of higher generations there are two processes that happen, each occupying some proportion of the total population. The first is the intent to generate paths of high complexity, and the other is to further optimize the paths of low complexity, which may be having good fitness value. Jumping from a lower complexity to a higher complexity is also carried out.

The algorithm was simulated over four types of maps with all varying complexities. We saw that the robot was easily able to solve all of the four maps and return a solution. The solution to all these cases could be visually seen to be optimal. The solution controlled its complexity based on set multi-objective parameters. The graphs plotted between convergence of the four maps as well as the execution time, number of feasible solutions and predicting capability of *d* show oscillations. This is a very interesting behavior in the implementation of the algorithm. The prime reasons for this may be attributed to the approximate state. Because of this phenomenon many solutions that were claimed to be feasible in lower generations are now found to be infeasible because of the reduction in momentum. This changes the complete metrics of the system. The presence of weakly mutated solutions in the population pool as well as the PSO and crossover/mutation exploration ensure the generation of better individuals for an overall convergence.

Using the suggested approach we have been effectively able to generate optimal paths for a variety of maps. There is still a lot that may be done in the future. The path generated by this algorithm is a set of guiding points. The work of smoothening of these points making the path much realizable for the physical working of the robot may be carried out. The algorithm may be further extended to the dynamic obstacles as well. More evolutionary techniques may be tried over the developed problem base.

References

[1] A. Alvarez, A. Caiti, R. Onken, Evolutionary path planning for autonomous underwater vehicles in a variable ocean, IEEE J. Ocean. Eng. 29 (2) (2004) 418-429.

[2] K. M. S. Badran, P. I. Rockett, The roles of diversity preservation and mutation in preventing population collapse in multiobjective genetic programming, In: Proc. 9th Annual Conf. Genetic and Evolutionary Computation, GECCO'07, pp 1551 – 1558, 2007.

[3] L. H. Chen, C. H. Chiang, New approach to intelligent control systems with self-exploring process, IEEE Trans. Systems, Man, and Cybernetics, Part B: Cybernetics 33 (1) (2003) 56-66.

[4] J. Cortes, L. Jaillet, T. Simeon, Disassembly Path Planning for Complex Articulated Objects, IEEE Trans. Robotics, 24(2) (2008) 475-481.

[5] P. Dittrich, A. Bürgel, W. Banzhaf, Learning to move a robot with random morphology, Lecture Notes in Computer Science Vol. 1468, pp 165-178, 1998.

[6] L. Doitsidis, N. C. Tsourveloudis, S. Piperidis, Evolution of Fuzzy Controllers for Robotic Vehicles: The role of Fitness Function Selection, J. Intelligent and Robotic Systems 56(4) (2009) 469-484.

[7] S. Garrido, L. Moreno, D. Blanco, Exploration of 2D and 3D Environments using Voronoi Transform and Fast Marching Method, J. Intelligent and Robotic Systems 55(1) (2009) 55 – 80.

[8] W. Han, S. Baek, T. Kuc, GA Based On-Line Path Planning of Mobile Robots Playing Soccer Games, In: Proc. 40th Midwest Symposium Circuits and Systems, Vol 1, pp. 522-525, 1997.

[9] N. B. Hui, D. K. Pratihar, A comparative study on some navigation schemes of a real robot tackling moving obstacles, Robotics and Computer-Integrated Manufacturing 25(4-5)(2009) 810-828.

[10] K. G. Jolly, R. S. Kumar, R. Vijayakumar, A Bezier curve based path planning in a multi-agent robot soccer system without violating the acceleration limits, Robotics and Autonomous Systems 57(1)(2009) 23-33.

[11] C. F. Juang, A hybrid of genetic algorithm and particle swarm optimization for recurrent network design, IEEE Trans. Systems, Man, and Cybernetics Part B: Cybernetics, 34(2) (2004) 997-1008.
[12] R. Kala, A. Shukla, R. Tiwari, S. Rungta, R. R. Janghel, Mobile Robot Navigation Control in Moving Obstacle Environment using Genetic Algorithm, Artificial Neural Networks and A* Algorithm, In: IEEE Proc. World Congress Computer Science and Information Engineering, CSIE '09, pp 705-713. 2009.
[13] R. Kala, A. Shukla, R. Tiwari, Dynamic Environment Robot Path Planning using Hierarchical Evolutionary Algorithms, Cybernetics and Systems, 41(6) (2010a) 435-454
[14] R. Kala, A. Shukla, R. Tiwari, Fusion of probabilistic A* algorithm and fuzzy inference system for robotic path planning, Artificial Intelligence Review, Springer, 33(4)(2010b) 275-306
[15] R. Kala, A. Shukla, R. Tiwari, Robotic Path Planning using Evolutionary Momentum based Exploration, Journal of Experimental and Theoretical Artificial Intelligence (2011) Accepted. In Press.
[16] S. Kambhampati, L. Davis, Multiresolution path planning for mobile robots 2(3)(1986) 135-145.
[17] K. J. O' Hara, D. B. Walker, T. R. Balch, Physical Path Planning Using a Pervasive Embedded Network, IEEE Trans. Robotics 24(3)(2008) 741-746.
[18] T. Peram, K. Veeramachaneni, C. K. Mohan, Fitness-distance-ratio based particle swarm optimization, In: Proc. 2003 IEEE Swarm Intelligence Symp., SIS '03, 2003, pp. 174-181, 2003.
[19] S. K. Pradhan, D. R. Parhi, A. K. Panda, Fuzzy logic techniques for navigation of several mobile robots, Applied Soft Computing 9(1) (2009) 290-304.
[20] N. Sadati, J. Taheri, Genetic algorithm in robot path planning problem in crisp and fuzzified environments, In: Proc. 2002 IEEE Int. Conf. Industrial Technology, ICIT '02, 2002, vol.1, pp. 175-180, 2002.
[21] A. Shukla, R. Tiwari, R. Kala, Mobile Robot Navigation Control in Moving Obstacle Environment using A* Algorithm, In: Proc. Intl. Conf. Artificial Neural Networks in Engg., ANNIE'08, ASME Publications, Vol. 18, pp 113-120, 2008.

[22] S. Squillero, A. P. Tonda, A novel methodology for diversity preservation in evolutionary algorithms, In: Proc. 2008 Conf. companion on Genetic and Evolutionary Computation, GECCO'08, pp 2223-2226, 2008.
[23] R. Toogood, H. Hong , W. Chi, Robot Path Planning Using Genetic Algorithms, In: Proc. IEEE Int. Conf. Systems, Man and Cybernetics. Intelligent Systems for the 21st Century, Vol. 1, 1995, pp 489-494, 1995.
[24] C. H. Tsai, J. S. Lee, J. H. Chuang, Path planning of 3-D objects using a new workspace model, IEEE Trans. Systems, Man, and Cybernetics, Part C: Applications and Reviews 31(3)(2001) 405-410.
[25] J. Tu, S. Yang, Genetic algorithm based path planning for a mobile robot, In: Proc. of IEEE Intl. Conf. Robotics and Automation, ICRS'03, Vol. 1, pp 1221-1226, 2003.
[26] C. Urdiales, A. Bandera, F. Arrebola, F. Sandoval, Multi-level path planning algorithm for autonomous robots, IEEE Electronics Letters 34(2)(1998) 223-224.
[27] K. Veeramachaneni, T. Peram, C. Mohan, L. A. Osadciw, Optimization Using Particle Swarms with Near Neighbor Interactions, In: Proc. Genetic and Evolutionary Computation, GECCO '03, vol 2723, pp 110-121, 2003.
[28] C. Wang, Y. C. Soh, H. Wang, H. Wang, A hierarchical genetic algorithm for path planning in a static environment with obstacles, In: Proc. IEEE Canadian Conf. Electrical and Computer Engineering, CCECE '02, vol. 3, pp. 1652-1657, 2002.
[29] J. Xiao, Z. Michalewicz, Z. Lixin, K. Trojanowski, Adaptive evolutionary planner/navigator for mobile robots, IEEE Trans. Evolutionary Computation 1(1) (1997)18-28.
[30] S. X. Yang, M. Meng, An efficient neural network approach to dynamic robot motion planning, Neural Networks 13(2)(2000) 143-148.

Rahul Kala

Rahul Kala did his Integrated Post Graduate (BTech and MTech in Information Technology) degree at Indian Institute of Information Technology and Management Gwalior in 2010. He is currently pursuing PhD in Robotics from School of Cybernetics, University of Reading, UK.

His areas of research are hybrid soft computing, robotic planning, autonomous vehicles, biometrics, pattern recognition and soft computing. He is the author of two books and has published about 50 papers in various international and national journals/conferences. He also takes a keen interest toward free/open source software. He is recipient of Commonwealth Scholarship 2010 (UK) and winner of Lord of the Code Scholarship Contest organised by KReSIT, IIT Bombay and Red Hat. He secured All India 8th position in Graduates Aptitude Test in Engineeging-2008 Examinations.

Prof. Anupam Shukla

Prof. Anupam Shukla is serving as a Professor in Indian Institute of Information Technology and Management Gwalior. He heads the Soft Computing and Expert System Laboratory at the Institute. He has 20 years of teaching experience. His research interest includes Speech processing, Artificial Intelligence, Soft Computing, Biometrics and Bioinformatics. He has published over 120 papers in various national and international journals/conferences. He is editor and reviewer in various journals. He received Young Scientist Award from Madhya Pradesh Government and Gold Medal from Jadavpur University.

Dr. Ritu Tiwari

Dr. Ritu Tiwari is serving as an Assistant Professor in Indian Institute of Information Technology and Management Gwalior. Her field of research includes Biometrics, Artificial Neural Networks, Speech Signal Processing, Robotics and Soft Computing. She has published over 70 research papers in various national and international journals/conferences. She has received Young Scientist Award from Chhattisgarh Council of Science & Technology and also received Gold Medal in her post graduation.