Robotic Path Planning using Evolutionary Momentum based Exploration

Rahul Kala*, Anupam Shukla, Ritu Tiwari

Department of Information Technology, Indian Institute of Information Technology and Management Gwalior, Gwalior, Madhya Pradesh, India

* Corresponding Author Email: rahulkalaiiitm@yahoo.co.in

Citation: R. Kala, A. Shukla, R. Tiwari (2011) Robotic Path Planning using Evolutionary Momentum based Exploration, *Journal of Experimental and Theoretical Artificial Intelligence*, 23(4): 469-495.

Final Version Available At:
http://www.tandfonline.com/doi/abs/10.1080/0952813X.2010.490963

Abstract: In this paper we propose a new algorithm to solve the problem of robotic path planning in static environment where the source and destination are given. A grid-based map has been used to represent the robotic world. The basic algorithm is built on an evolutionary approach, where the path evolves along with generations with each generation adding to the maximum possible complexity of the path. Along with complexity we optimize the total path length as well as the minimum distance from the obstacle in the robotic path. It may be seen that the requirement of evolutionary parameter number of individuals as well as the maximum complexity is less at start and more at the later stages of the algorithm. We use a Gaussian increase in these values whose parameter may be adjusted to control the time and output. Seven Genetic Operators have been implemented that include selection, crossover, soft mutation, hard mutation, insert, delete and elite. The phenotype representation consists of the coordinate where the robot is supposed to make a turn. This happens by the traversal of the path using these points by the Evolutionary Algorithm. Momentum determines the speed of the algorithm in this traversal.

Keywords: robotic path planning; robotics; evolutionary algorithms; genetic algorithm; momentum; evolution

1. Introduction

Robotics is a highly multi-disciplinary field of research that attracts numerous people from various domains to contribute towards this field including machine learning, soft computing, pattern recognition, cognition, neurosciences, mechanical engineering, electrical engineering, wireless communication, etc (Ge and Lewis, 2006). The problem of path planning deals with the determination of path or a navigation plan for the robot (Patnaik et al, 2005). The path needs to be such that the robot may easily be able to reach the goal starting from source without collision. The problem of path planning may be studied in static environment (Wang et al, 2002) or dynamic environment (Sud et al, 2008). The quality of the path may be judged on various parameters. These include the total length of the path, the straightness of the path, possibility of collision, ease of practical navigation etc (Ashiru, 1995; Doitsidis, 2009; Han, Baek and Kuc, 1997). The various path planning algorithms may further be compared on the quality of results as well as the computational time.

Path planning forms an indispensable module in the intelligent control and application of autonomous robots (Ge and Lewis, 2006). Any robotic planning or coordination (Brooks, 1985; Hazon and Kaminka, 2008; Peasgood, Clark and McPhee, 2008) technique may rely heavily on the application of path planning algorithms to move the robot so as to make it reach the desired location. Some of the common situations include the path planning in uncertain environments (Podsedkowski et al, 2006), planning to cover an entire region or

Travelling Salesman Type problems (Sheng et al, 2005), planning to reach a specific goal, etc. In this paper we assume the problem is to reach a specific goal from a specific source, both of which are well determined.

Another important concept that comes in robotics is the robotic map. A map is a representation of the robotic world (Thrun, 1998). It denotes the presence of static and dynamic obstacles as well as accessible and inaccessible areas. Various algorithms exist to make a robot map using the information acquired by the sensors and cameras (Brooks,1985). These make use of image and signal processing techniques to interpret the various Regions of Interests (ROI) out of the acquired data and represent the same as a map. Various kinds of maps may be drawn that include the Vornoi maps, Topographical Maps, Hybrid Maps etc. In this paper we assume that map to be already formed and available in the form of a Grid of size M x N with each cell denoting the accessibility or the presence/absence of obstacles.

Evolutionary Algorithms is another area which offers effective solutions to a variety of problems (Mitchell, 1998). The concept of evolution came up from the works of Holland with a motivation to imitate the natural evolution (Holland, 1975). These algorithms try to evolve the most optimal solutions to the problem where the solution keeps improving along with time and generations. The higher generation solutions are better suited to the problem than the lower generations. Classically these are optimization algorithms that try to adjust a set of parameters to give the maximum or the minimum value of the objective function. One of the major tasks here is the encoding of the problem such that the algorithm may perceive it as an optimization problem. Evolutionary Artificial Neural Networks (ANNs) and Evolutionary Fuzzy Inference Systems (FIS) (Shukla, Tiwari and Kala, 2010) are some of the applications of these algorithms.

The natural world gives us a lot of inspiration to make effective systems (Holland, 1975). Life started millions of years back with uni-cellular organisms. Over the years we have seen the evolution of many complex species which better adapt to their environment. The beauty also lies in the fact that this adaptation is dynamic with the species adjusting themselves with every generation to the changing environment. A similar concept may be seen in Human Brain which starts as a premature entity and later develops into complex forms as the child transforms into an adult. We try to imitate the same philosophy of life where complexity increases along with time and generations. We know from our understanding of natural (uni-cellular v/s multi-cellular organisms) as well as artificial systems (ANN or FIS) that best systems employ the use of minimum complexity needed to solve the objective.

In the problem of robotic path planning, complexity is denoted by the number of turns in the path of the robot. The lesser the complexity or the number of turns, the straighter and hence better is the final path of the robot. Too much of complexity unnecessarily adds a zig-zag like path in the navigation plan of the robot which is undesirable. Hence there is a need to limit the complexity of the problem. Many times the effective path (or shorter path) may lie after taking many turns even though a simpler or a less complex counterpart may be available. Hence we see that lower complexity might not necessarily denote a better overall path.

The Evolutionary Algorithms (EAs) make excessive calls to fitness function to evaluate the individuals (Mitchell, 1998). It is hence necessary to limit the complexity of the fitness function at the same time not compromising with the fitness algorithm. The application of conventional Genetic Algorithm (GA) (Toogood, Hong and Chi, 1995; Han, Baek and Kuc, 1997; Gerke, 1995; Sedhigi, 2004; Woong-Gie, 1997; Yanrong, 2004) might involve the

fitness algorithm traversing the represented path to ensure a collision free navigation. Ideally the navigation needs to be cell by cell which would make the algorithm very slow. For this we use the concept of momentum. A higher momentum means a higher speed of navigation where the algorithm traverses after skipping some cells. Initially the solutions are premature and may be checked for feasibility by higher momentums. The momentum gradually reduces as the solutions mature. This is the time when we would like to ensure their feasibility.

Using the suggested approached we intend to solve 3 major problems that exist in the current algorithms. The first is the problem of path complexity control. It may easily be seen that the complexity of the path can never be predicted as it depends upon the complexity of the map which largely varies with map. The second is the optimization of the time complexity of fitness function by using a cautious investment of time. This may be seen a similar approach to the probability based map representation or a radix tree map representation (Kambhampati and Davis, 1986) with the role of probability being played by momentum. The third is the need based genetic individual deployment that we maintain by varying the number of individuals in the genetic search. Only the required number of individuals is deployed at every generation that goes a long way in optimization of time.

This paper is organized as follows. In section 2 we present some of the state of the art methods and techniques on robotic path planning with a special emphasis on the evolutionary and genetic methods. Section 3 talks about the solution representation for use by the EA. Section 4 deals with the fitness function used for EA. We also discuss the role of momentum in section 5. In section 6 we present the EA framework. Here we discuss the 7 different kinds of Genetic Operators that are used in this algorithm. Section 7 talks about the role and importance of variable genetic parameters. Section 8 presents the results. We give the conclusion remarks in section 9.

2. Literature Review

Robotic Path Planning is a very old and widely studied problem in the field of robotics. Since its beginning various methods have been employed to solve this problem. A* Algorithm is one of the prominent methods of study (Shukla, Tiwari and Kala, 2008). Here the problem is modeled as a graph based search problem whose solution is to be found. Other approach makes use of Dynamic Programming Technique to find the closest distance from source to goal (Bakker, 2005; Suh and Shin, 2002). These methods further give rise to the D* algorithm which mixes the benefits of these algorithms. The use of Artificial Neural Networks is another class of algorithms (Kala et al, 2009; Yang and Meng, 2000; Noguchi and Terao, 1997). Here the solution is made based on learnt past experiences. Similar approaches include Fuzzy Systems (Pradhan, 2009) and ANFIS (Khoshnejad and Demirli, 2005; Nam et al 2001). The other approaches include the potential field approach where the goal and other landmarks act as attractive forces and the various obstacles act as repulsive forces to move a robot (Pozna et al, 2009; Tsai et al 2001). The whole problem may be studied under two separate heads of static environment (Chunmiao, 2002) and dynamic environment (Sud et al, 2008). Each of these systems employed to solve the problem has some benefits and weaknesses. The A* and Dynamic Programming algorithm gives very optimized solution but may fail to perform in very large or complex maps because of the time complexities. Further these work only in static environment. The ANN, Fuzzy and ANFIS denote the behavioral planning systems where planning is done based on robot behavior (Floreano and Mattiussi, 2008). These systems give very fast results in dynamic environments, but the optimality of the path can never be guaranteed. These may fail to give

the path in complex maps. A comparison of some of the algorithms including hybrid algorithms can be found in the works of Hui and Pratihar (Hui and Pratihar, 2009).

Another novel concept introduced in this problem was the multi resolution representation or the quad-tree representation (Kambhampati and Davis, 1986) or a pyramidal representation (Urdiales, 1998) of the robot map. Here instead of working on a detailed graph, we work over a low resolution or vague graph. This graph has a probability associated with each cell that denotes the probability of occurrence of an obstacle. The resolution of a section may be improved on demand as the algorithm runs. This further made hierarchical path planning algorithms possible (Bakker, 2005; Chunmiao, 2002). A good algorithm on similar concepts is presented by Jan et al (2008) and Zhu and Latombe (1991).

A lot of work has also been done to solve the problem with the use of Evolutionary Algorithms (EA) and Genetic Algorithms (GA). The simpler models in this field make use of graph based individual representation (Shukla, Tiwari and Kala, 2008) or restricted Genetic Operations (Alvarez, 2004) to ensure feasibility of the solution generated at every step. Here the whole path is modeled as a collection of consecutive points (Sugihara, 1996). The other model includes the individual representation in terms of set of sparse points from source to destination with the line joining source and destination as one of the axis (Han, Baek and Kuh, 1997; Toogood, Hong and Chi, 1995) or otherwise as common points, metrics (Ahuactzin, 1991; Gerke, 1999; Sedhigi, 2004) or obstacle corner based points (Sadati and Taheri, 2002). MAKLINK graph is used in the works of Shibata et al (1992). Variable length representation of chromosome for the problem has been presented by Tu et al (2003). The points are traversed in a straight line using this model. A hierarchical Genetic Approach for the problem is given in the works of Wang et al (2002). GA has also been used for the optimization of the FIS models (Judette and Youlal, 2000; Shibata, Fukuda and Tanie, 1993) or for the generation of rules for behavioral modeling of robot (Shukla, Tiwari and Kala, 2010). A study of the fitness function of GA in the optimization of Fuzzy Controllers is provided by Doitsidis et al (2009). Though GA is mainly used for static environment, yet using the concept of space-time graphs we may model GA to optimize the path in dynamic environment whose dynamics is known (Patnaik et al 2005). The algorithm of Dittrich et al (2006) 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. Vadakkepat (2000) used EA along with potential field approaches to give rise to Evolutionary Potential Field.

Much of the good work in the field of robotic path planning comes from hybrid approaches and methods where combination of AI and Soft Computing techniques are used. 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. Xiao et al (1997) implemented off-line and on-line planner in an adaptive manner using evolutionary computation to make a novel planning algorithm that separately planned for static and dynamic environments. Jolly, Kumar and Vijayakumar (2009) used the novel concept of Bezier curve along with obstacle avoidance strategy to model safe robot navigation in a soccer game. O' Hara, Walker and Balch (2008) present another unique algorithm that works at the hardware level with embedded systems. Here the robot makes use of connected hardware devices at various locations of map to figure out heuristics. Other works includes the use of Rapidly-exploring Random Tree (RRT) algorithm by Cortes et al (2008) and Fast Marching Method with Vornoi Transform by Garrido, Moreno adn Blanco (2009).

A similar problem compared to robot path planning is that of robotic coordination. Some of the work in this field includes the work of Peasgood, Clark and McPhee (2008) who proposed a good algorithm to move multiple robots with different sources and goals without collision. Other works include the work of Carpin and Pagello (2009) and Hazon and Kaminka (2008).

3. Individual Representation

One of the foremost tasks in this algorithm is a good individual representation. Here we represent an individual by a series of points (P_i) on the robotic map. The complete individual hence becomes $<P_0, P_1, P_2, P_3, \ldots. P_n, P_{n+1}>$. 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 as given in figure 1.

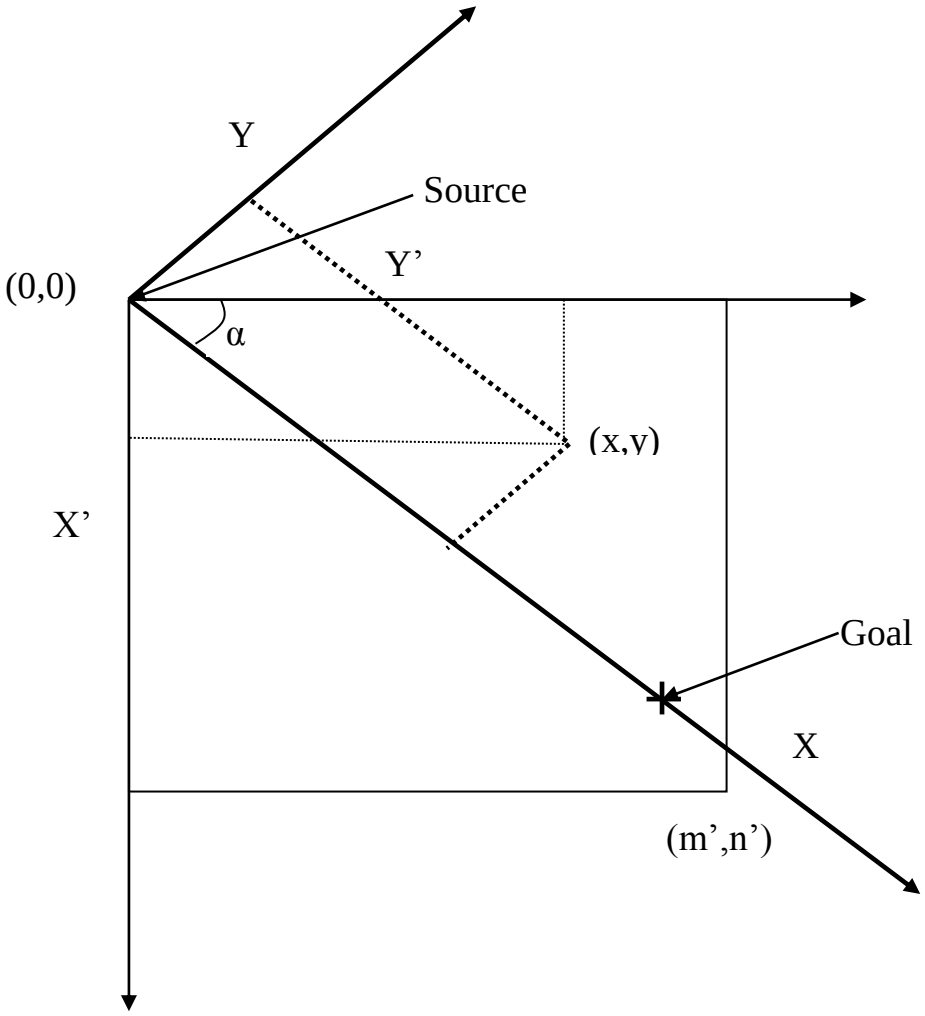

Figure 1: The Coordinate System for Individual Representation and Robotic Map

Let us suppose that the map is represented in the coordinate system *X'-Y'* given in figure 1. Now any point needs to lie within the range of *(0',0')* to *(m',n')* so as to lie within the map. Here ' represents the use of *X'-Y'* coordinate system. This range needs to be converted into equivalent range in the *X-Y* coordinate system of the individual to generate valid points in the

robotic map. This is done by a rotation of angle α in the clockwise direction, where α is the angle between the two coordinate systems given in figure 1.

Complexity of a path or a solution is defined as the total number of points in the path or solution. For the path represented by $<P_0, P_1, P_2, P_3, \ldots. P_n, P_{n+1}>$, the complexity is *n.* The complexity for any path may lie between *0* (straight line from source to foal) to a maximum permissible value C_{max}.

Another important characteristic of the individual representation is that the various points in the map are always sorted along the *X* axis. The final path is the path traversed by touching the various points in a straight line one after the other. Here the source is the first point and goal is the last point. Hence we assume that in the final path, the robot cannot move backwards.

The entire length of the chromosome is hence fixed to a maximum value of $2 \times C_{max}$. The unoccupied positions in any solutions are filled with *Inf* (Infinity).

A sample path is given in figure 2. If C_{max} is set to *10* and each P_i denotes the point (x_i,y_i) then the genetic individual of this path would be represented as $<x_1\ y_1\ x_2\ y_2\ x_3\ y_3\ x_4\ y_4$ *Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf*$>$.

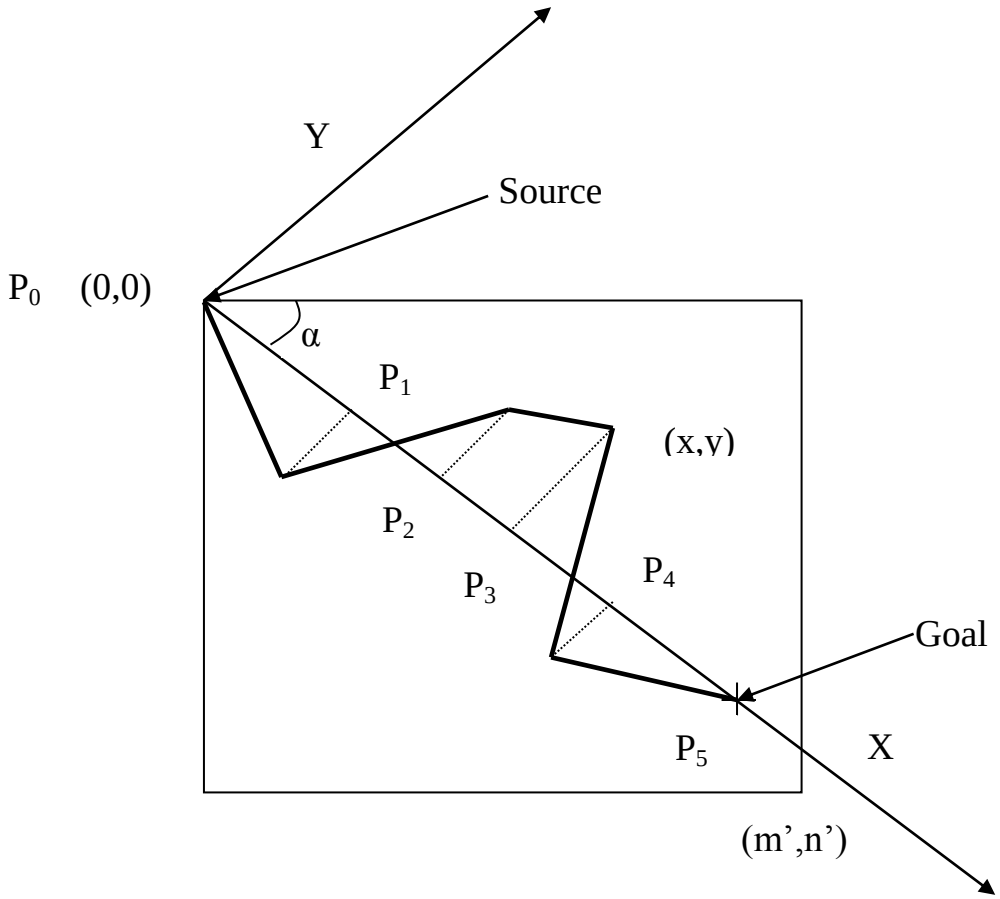

Figure 2: A Sample Path of the problem

4. Fitness Function

The next important factor in the algorithm is the fitness function. The fitness function is used to judge the quality of any path. Here we try to work upon three major factors that all contribute to the final fitness of the path. These are the total length of the path (*l*), the distance from the closest obstacle (*d*) and the path complexity (*c*). The objective is to minimize each one of *l*, *d* and *c*. Using the principles of Multi-Objective Optimizations, we frame the equation of the total fitness as given by (1).

$$\text{Fit}(P) = \alpha \text{ x } l + \beta \text{ x } d + \gamma \text{ x } c \quad (1)$$

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

We discuss each of these factors in the next sub-sections

4.1 Total Path Length

This is a measure of the total length of the path from the source to the destination. Consider the path shown in figure 2. Here we traverse the path consecutively between the points represented by the individual. The first point is taken as the source and the last as the destination. Hence we always start from the source and end at the destination. This path may be measured by the physical distances between the points P_i and P_{i+1} given by equation (2). It is natural that the lower values of this factor are desirable.

$$T = \sum_{i=0}^{n} |P_i - P_{i+1}| \quad (2)$$

Here *T* is the path length
P_i is the point represented by the individual
P_0 is the source
P_{n+1} is the goal
$| x |$ denotes the Euclidian, Manhattan or any other norm.

This length is normalized to lie between 0 and 1 by equation (3)

$$l = \frac{T}{m+n} \quad (3)$$

Here *l* is the total path length (normalized)
T is the path length (un-normalized)
m+n is the maximum length possible for any path (normalizing factor)
m and *n* are the length and breadth of the map respectively

4.2 Distance from Closest Obstacle

This factor returns the smallest distance between any point in the path of the robot and the obstacle. The motivation for this factor comes from the physical movement of the robot. It is

known that if the robot path goes through a region that is very close to an obstacle, it may have to slow down in order to avoid a collision. This is especially important if the obstacle is close to some turn of the robot. Another way to look at the same problem is through nonholonomic constraints. The algorithm we build might result in very sharp turns, but it may be impossible for the robot to make such turns due to physical limitations or the nonholonomic constraints. For this the robotic controllers or sometimes even the path planning algorithms artificially smooth the path. This requires a comfortable distance from the obstacle.

In this algorithm we assume the factor distance from closest obstacle (*d*) measures the ease of the movement of the robot that decays in a Gaussian manner as the distance increase. The relation between the minimum physical distance of the obstacle and robot (*D*) and *d* is given by equation (4).

$$d = e^{-\frac{D^2}{2(aM)^2}} \quad (4)$$

Here *d* is the distance from closest obstacle (normalized)
D is the physical distance from closest obstacle (un-normalized)
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.

It may be easily seen that effectively the equation (4) works only for distances (*D*) in the range of *0* to *M*. After this the value becomes almost zero. We hence apply equation (4) only to *D* that is less than *M* units of length. Practically *M* represents the distance which is comfortable enough for the robot to easily make its way out.

In order to avoid repetition (and save time) we make a lookup table that stores the values of *D* for all combinations of points in the graph initially at start. This is done using the principles of Dynamic Programming whose recurrence relation and initial condition is given by equation (5). This process is known as the *fuzzification* of the graph.

$$\begin{aligned} D(P,k) = {} & 0 \text{ if } k=0 \text{ and } P \text{ is an obstacle} \\ & \text{Inf if } k=0 \text{ and } P \text{ is not an obstacle} \\ & \text{Min}\{D(P,k-1), D(\delta(P),k-1)+1\} \text{ for all other cases} \end{aligned} \quad (5)$$

Here *P* is any point
k is the Dynamic Programming iterator that lies between *0* and *M*
δ(P) is the neighborhood function that returns all points in the neighborhood of *P* (the 8 surrounding points).

It may happen that paths represented by individuals happen to have obstacle in between. This may be easily found out by a path traversal from source to goal. These are in-feasible solutions of the EA and are assigned a fitness value of *Inf* (Infinity).

4.3 Path Complexity

This factor refers to the number of turns that a robot is allowed to make in its entire path. It is a measure of the straightness of the path that is a very desirable property of a good path in the field of robotics. If the path complexity is low, the path would be as straight as possible. On

the other hand, a large path complexity enables a robot to follow a zig-zag path that is usually longer and more difficult to implement in the physical motion of the robot. Hence we try to restrict or minimize the path complexity in this algorithm. The path complexity may also be taken as the used size of the individual as discussed in section 3. The path complexity c is normalized by dividing with the maximum allowable complexity C_{max} to make it lie in the interval of 0 to 1.

Many times a higher complexity path may drive the robot closer to the straight line path from source to destination and hence may be significantly smaller as compared to the lower complexity path. An analogy of this may be derived from the fact that many times the shortest path from source to destination is filled with many obstacles that require a large number of turns (with low speed) than another path which may have a very long route but only a couple of turns. Hence complexity (*c*) and length (*l*) do not necessarily mean the same thing. It may again be seen that the preference between complexity and length is not easy. In the same analogy, one may prefer to go with the more complex path since its length is small and we may be able to reach the destination early at lower speeds and lesser journey comfort. One may also prefer a longer length path with lesser complexity as it may lead to goal early due to possibility of higher speeds and more journey comfort at the cost of length. Choosing between a low traffic/longer route or a high traffic/shorter route between *New Delhi* to *Noida* is always a difficult choice. This necessitates the possibility to keep multiple diverse paths in possibility while the algorithm runs. We implement the same in this algorithm by Diversity preservation (Badran and Rockett, 2007; Laumanns et al 2001).

Another analogy may be seen between the evolution of the path in this algorithm and the evolution of ANNs. We know that adding neurons in ANN results in larger complexity that needs to be controlled. The larger number of neurons results in a zig-zag fitting of the trained curve between the data points (Shukla, 2010). This is accounted for in most of the modern Evolutionary ANN designs that evolve an ANN. This is one of the motivations behind the proposed algorithm.

5. Momentum

Momentum refers to the speed of traversal while working with the optimization of the path in EA. With this concept we try to optimize the total time of fitness function which would adversely affect the final computation time as well as the results.

Recall our discussion of section 4 where we had to traverse the entire path, grid by grid, from source to goal in order to calculate the distance from obstacle (*d*) as well the feasibility of the path. In a practical sense if we traverse the path in such a manner for every call of the fitness value, the algorithm speed would decrease by a large amount. We need a mechanism to optimize this time. This is done by the concept of momentum. Here we only check for the presence of obstacle and calculate the factor *d* between any point P_i and $P_{i+1}$for the set of points Q given by equation (6)

$$Q = \{P_i, P_i + mc\ x\ \frac{(P_{i+1}-P_i)}{|P_{i+1}-P_i|}\ x\ t, P_{i+1}\} \quad (6)$$

Here P_i and P_{i+1} are the consecutive points in the robot path
$\frac{(P_{i+1}-P_i)}{|P_{i+1}-P_i|}$ is the unite vector in direction of P_{i+1}-P_i
mc is momentum

t is the traversal step (*t=1, 2, 3…*)

It may be observes that if the momentum is large, less number of points are checked and correspondingly the execution time is less. But at large two types of conditions are possible. The first condition that is rare to happen is that we completely step over an obstacle. Here the algorithm assumes that the path is feasible. But in reality there would be an obstacle in the path and there would be no where out to avoid this obstacle. We call this condition as the *false state*. The other case that can happen is that there might be obstacle on the path being considered, but a feasible path may lie very close. This is a very frequent phenomenon to happen. We call this condition as the *approximate state*. Both these states are given in figure 3. The curved lines denote the traversal with momentum.

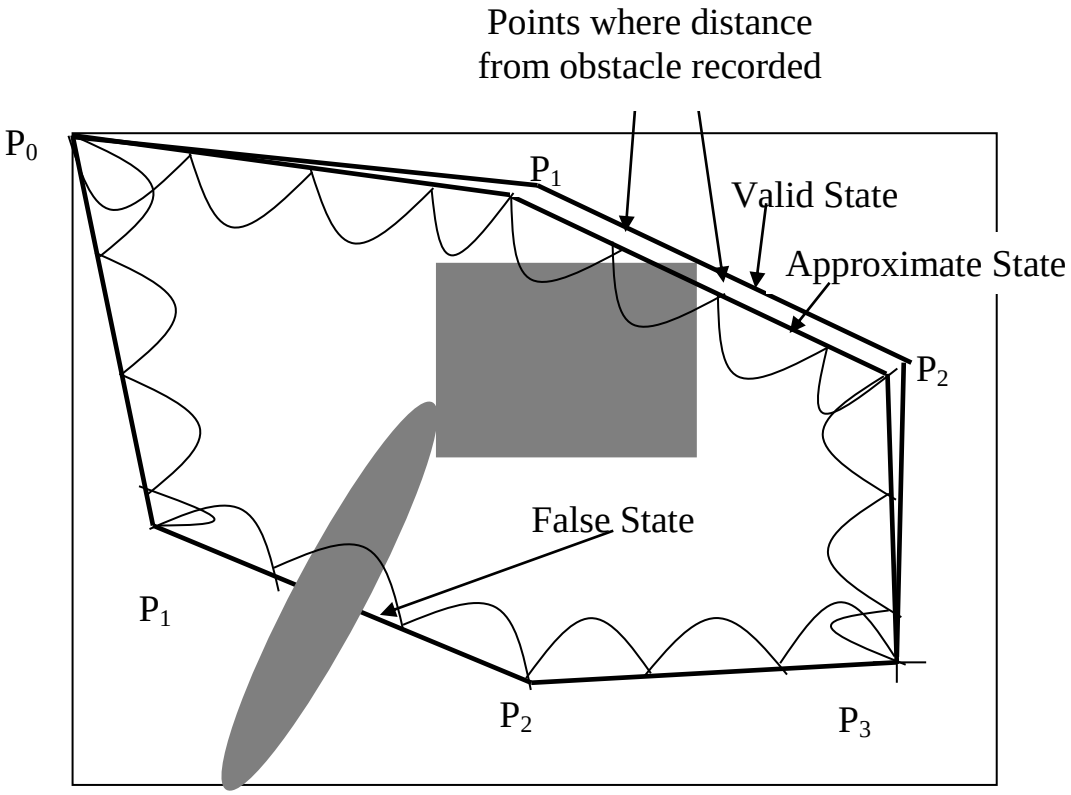

Figure 3: The concept of momentum

The ill effects of large momentum do not only apply to the path feasibility, rather they also apply to the distance from obstacle (*d*). Consider the *approximate state* in figure 3. Here even if we adjust P_2 and P_3 to make path feasible it may lie very close to the obstacle that might not be recorded at large momentum. This case is given in figure 3 as the third condition. The two points report a sufficient distance of separation from obstacle which is not true. This condition is referred as the *valid state*.

Hence large momentum increases speeds but wrongly records path feasibility as well as distance from obstacle (*d*). The smaller values of momentum on the other hand result in poor time but give correct feasibility measure. Hence a constant momentum based technique would not work. We make the momentum variable that changes in a Gaussian manner along with the generations. 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 well as

quality. As the generations go on, we keep decreasing the momentum to get the real values of the feasibility and parameters. The Gaussian decay of momentum is given by equation (7).

$$mc = mc_{min} + (mc_{max} - mc_{min})e^{-\frac{g^2}{2(bG)^2}} \quad (7)$$

Here mc_{max} is the maximum possible momentum
mc_{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

Another important factor here is of diversity preservation in regard to false state and approximate state discussed above and presented in figure 3. Suppose that the best known solution of the EA after a number of generations is found to be infeasible in regard to condition 2 or approximate state. To overcome such a condition it is necessary for the EA 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 condition 1 or false state. In such a case the solution is maintaining a population diversity which has been inbuilt in the EA (Badan and Rockett, 2007; Laumanns, 2001).

6 Evolutionary Algorithm

The base of the path planning algorithm is EA that tries to generate paths of increasing complexity. The EA tries to solve the whole problem of path planning as an optimization problem where it tries to optimize the location of various points so as to give the best path as per the set criterion. In this case the EA is also responsible to figure out the number of points that must be present in the solution or the final path. As discussed this complexity is variable and keeps on increasing as we proceed with the algorithm. In this section we present the various Genetic operators that contribute to the working of the EA. We have used a total of 7 Genetic operators. These are Selection, Crossover, Soft Mutation, Hard Mutation, Elite, Insert and Repair. We discuss each one of these one by one.

6.1 Selection

Using this genetic operator we select the individuals for the purpose of participation in reproduction or crossover. Rank based fitness scaling with stochastic uniform selection scheme has been used. A point that we have been insisting is the diversity preservation. Here a set of points qualify for crossover only when they are separated by a distance less than the threshold $\acute{\eta}$ distance in the input space. The separation (S) is measured as the average separation between corresponding points existing in the path.

Say that two solutions A and B have x and y number of points in their paths. Further suppose $x < y$. We calculate the x closest set of points (a,b) such that a lies in A and b lies in B and neither of a or b are repeated. The average distance between a and b for all points gives the separation S. This is given by equation (8). This is easy to implement as A and B are already sorted.

$$S = \frac{\sum_{a\,\acute{\in}\,A, b\,\acute{\in}\,B, b\ is\ least\ mapping\ of\ a\ for\ all\ unique\ a\ and\ b} |a-b|}{x} \quad (8)$$

6.2 Crossover

Crossover mixes two individuals for the generation of a next generation individual. Here we use a scattered crossover technique for crossover between the individuals. Suppose the two parents are *A* and *B* that have *x* and *y* number of points existing. We first make a pool of points *R* that carries all points from *A* and *B*. The points common to *A* and *B* are taken only once. The points in *R* are sorted according to the *X* axis values. Now we distribute the points in *R* to the new children such that each of the 2 generated child gets *(x+y)/2* points and each of the point in *R* belongs to either of the two children.

6.3 Soft Mutation

This mutation moves the solutions or individuals by little amounts. In other words we follow a Gaussian uniform mutation technique with very small mutation value. It may be noted that the various individuals formed as a result of this mutation are added to the solution pool without affecting the parents. This is a solution to the second condition or the approximate state we studied in section 4. Besides as a general evolutionary principle, this helps in the exploration of the search space or the optimization space. This operation is the implementation of the concept of neural walks (Floreano and Mattiussi, 2008).

6.4 Hard Mutation

This mutation tries to add more diversity and possibility of the path by exploring new areas. In other words we follow a Gaussian uniform mutation technique with high mutation value. No change is made to the number of points or the path complexity in both these mutations.

6.5 Elite

In this genetic operator we simply pass the individuals with high fitness value to the next generation. This operator plays a big role in the preservation of the good solutions in the population pool at all generations.

6.6 Insert

Using this operator we add new individuals to the population pool. This operation helps in exploring newer areas in the search space that might have been left out. Since the complexity is on a constant rise, it is important to add the higher complexity individuals in the solution pool. They would participate and interact with other individuals to increase their complexity as well, if higher complexity is desired and may provide a good solution. As per the Darwinian principles, these individuals survive and influence the solution if they result in higher fitness value.

In order to use this operation we select the individuals from the existing population pool that have the best fitness. We add random points to it till the number of points become equal to the maximum allowable number of points or complexity c_{max}. The entire individual is then sorted as per our set criterion. It may be noted that c_{max} increases with generations to mark an increase in complexity. We discuss the details in section 7.

6.7 Repair

This is the last genetic operator. 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. We make a single attempt to repair such individuals by replacing them with newer individuals of the maximum permissible complexity c_{max}. This does not guarantee the feasibility of the newly generated individual, but 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. Depending only upon the insert operator to drive the algorithm to regions with sufficient complexity would have been at a very low pace. The second problem that is solved by this operator is cleanliness. The infeasible solutions do not contribute at all to the EA. It is better to clean them by replacing them with random solutions of complexity high enough to represent feasible solution.

7. Variable Genetic Parameters

We have already discussed the variable nature of the momentum in section 5. In this section we discuss the variation or the variable nature of the number of individuals used by the EA as well as the maximum complexity c_{max}.

7.1 Variable Number of Individuals

The number of individuals plays a vital role in EA that help in the exploration of the search space. A very low number of individuals would result in a very little search space exploration but this would make more number of generations possible in the same computation time that may give enough time for crossover to converge or mutation to search nearby or even distant areas. Ideally the numbers of individuals depend upon the search space.

We start the algorithm in a condition where only limited size complexity is allowed. It may be seen that this limits the search space to a very small extent. Hence we require only a very small number of individuals. As the algorithm goes on, the maximum complexity is allowed. As a result the search space also grows more and more. This requires more number of individuals for exploration. By this time a few lower complexity paths are found which reserve some individuals for their exploration. Hence we model the algorithm such that the number of individuals is increased in a Gaussian manner along with generations. This is given by equation (9).

$$I = I_{min} + (I_{max} - I_{min})e^{-\frac{g^2}{2(cG)^2}} \quad (9)$$

Here I_{max} is the maximum possible number of individuals
I_{min} is the least possible momentum
g is the generation number of EA
c is the decay constant
G is the radius constant or the maximum number of generations possible

7.2 Variable Maximum Complexity

The maximum complexity or the maximum allowable points also follow similar trends and discussions as discussed for the number of individuals. We know that in general a smaller complexity is better. More complexity may add to longer and non-straight paths. Hence, the complexity is made to rise in a Gaussian manner as given by equation (10).

$$C_{max} = C\, e^{-\frac{g^2}{2(dG)^2}} \qquad (10)$$

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

8. Results

The testing of the algorithm was done on a simulation engine that was built by the authors themselves (Kala et al 2009; Shukla, Tiwari and Kala, 2008). We coded the algorithm using JAVA framework with Eclipse IDE. JAVA Applets was used for the depiction of the map and the solutions. 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. Here we present the results to four maps out of the various maps used for the testing purposes. All four of these maps are generated with some or the other inspiration to conditions that a robot might face in real life.

The first map denotes a simple map with a single obstacle in the path from source to destination. This may be a very common situation where a robot has a straight path with some obstacle on its way. The second and the third maps have a variety of obstacles and the robot needs to use some intelligent technique to figure out its way by avoiding all the obstacles. This tests the ability of the robot to calculate the shortest path. The fourth and the last map that we present is the path based map where robot tries to move itself on a non-straight and complex path from the source to the destination.

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).* All the simulations were carried on a system with 3 GB RAM and 2.40 GHz 2.40 GHz Core 2 Duo Intel processor.

For all experiments the Multi-Objective parameters α, β and γ were specified as *0.33* each (*0.5*, *0. 5* and *0.0* for fourth case). The decay constants *a, b, c* and *d* for each of the Gaussian curves was fixed to be *0.3*. Momentum could vary from *10* to *1.5*. The value of threshold $\acute{\eta}$ 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* (*7.5* for third and fourth case). The number of individuals could vary between *1* to *1000 (2000* for third and fourth case). The simulation was continued for a total of *500* generations (*2000* for third case and fourth case).

The two Gaussian mutation parameters were fixed to be *0.06* and *0.25* for soft and hard mutation respectively. At any generation, *68%* individuals came from reproduction, *2%* from eliticism, *15%* and *5%* from soft and hard mutation and rest *10%* from insert.

The maps and the path traced by the robot for each of these cases is given in figure 4(a), 4(b), 4(c) and 4(d).

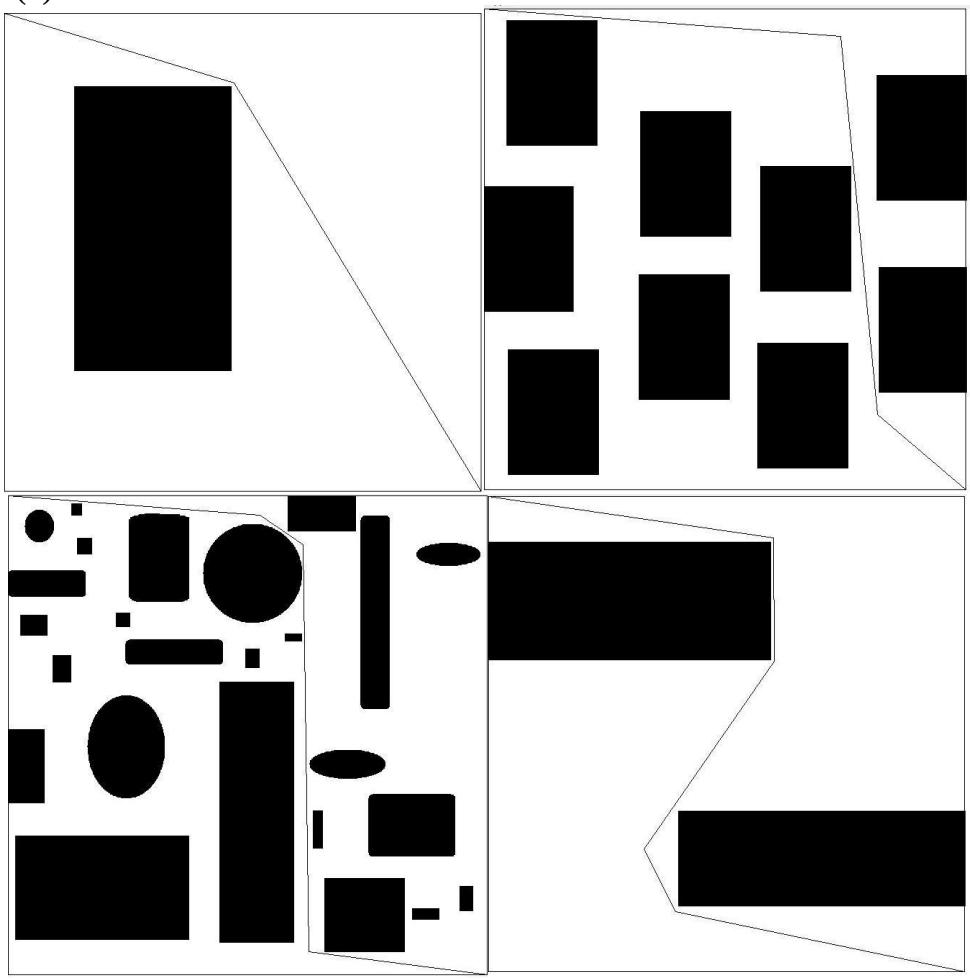

Figure 4: The path traced by the robot for various map

It may be easily seen that in all the cases the robot was able to evolve a path from the source to the destination. The path served the set criterion in the fitness function. The graph shown in figure 4(a) is an example of a simple problem where the solution was generated with a single intermediate step or a complexity of 1. It may be easily observed from the path traced that the robot kept a comfortable distance of separation from the obstacle and avoided going too close to it. A similar trend can be seen in the graph 4(b) which represents a slightly more complex problem, even though the solution is just a level more in complexity. Here it is interesting to observe that the robot could have moved diagonally in the top section which might have resulted in shorter path but it preferred not to do but traverse straight. This is due to the heavy penalty of increase in complexity that we added. In the physical movement by a robotic controller the path would get smoothen up (we have left enough room for that). This would enable the robot to traverse the path at smoothly at high speed. This would compensate the increase in length where the robot would have been forced to make two sharp turns.

The graph shown in figure 4(c) marks a still more complex graph where the robot has taken a very steep turn in the top section. It can be seen that the conditions were highly chaotic and

finding a way out was not very easy. Here a very special path is chosen by the robot as the most optimal path. In the top section it prefers to pass through a very complex structure by making two turns which makes the complexity high. After this section the work is relatively simple where the robot has a lot of straight path to traverse. This large amount of straight path compensates for the complexity added at start. Various other paths could have been possible of lower initial complexity but the straight path traversed gives this path the edge over other options. Figure 4(d) again represents an interesting map where the robot has to march in a predefined path. It may be easily seen that the robot happened to optimize the path length in this case. This is because the straight path penalty had been switched off by setting a weight age of zero.

We also study the behavior of the best fitness value in all the discussed paths and configurations. The plots of the best fitness against the generations are given in figure 5(a), 5(b), 5(c) and 5(d). Here we do not plot the regions in the graph where the best fitness value was infinity. It may be noted that the fitness value would always be infinity till the time the algorithm acquired the minimum complexity needed to solve the problem. As the algorithm proceeds, the best value generally keeps improving.

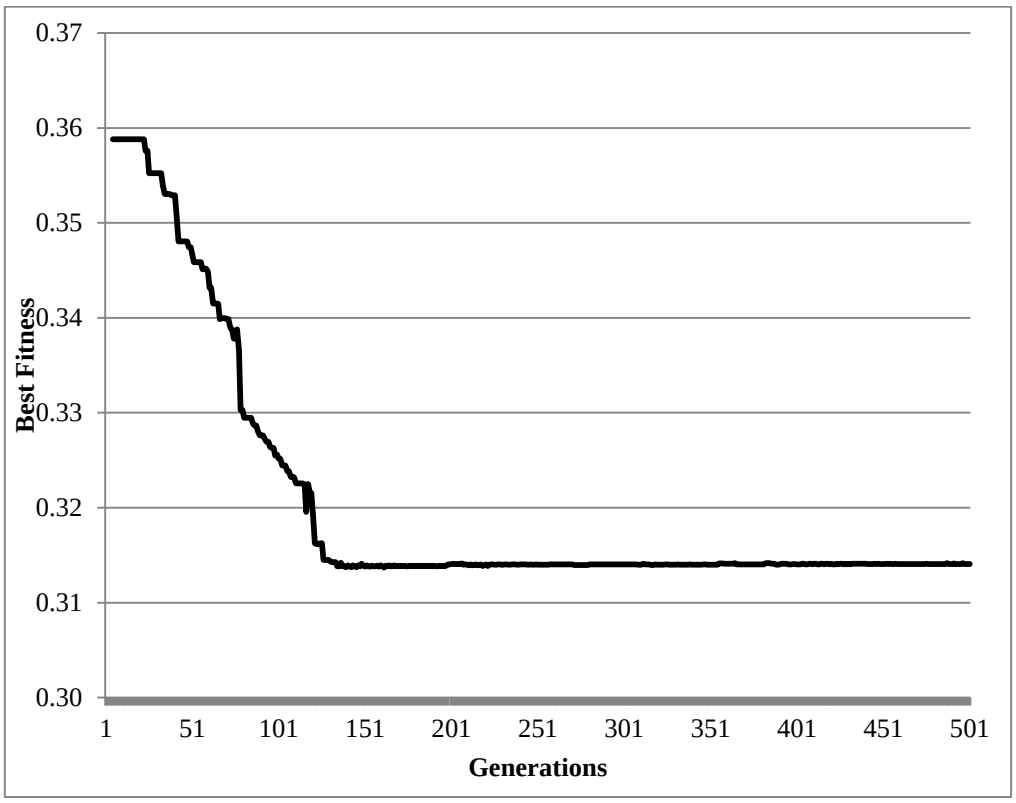

Figure 5(a): Graph showing best fitness v/s generation for the EA for map 1

In all 4 maps it can be seen that the fitness values improves with generations after the needed complexity is met. Graph given in figure 5(a) represents the algorithmic convergence where the decrease in value is very sharp at start and later on the fitness value converges to the most minimum value. A similar trend is observed in figure 5(b) with a very sharp convergence. This may be due to the limited search space in the complexity in which the solution is found. Figure 5(c) represents a very interesting graph where the best fitness value very sharply decreases and then oscillates. The rise in fitness value represents the condition 2 or approximate state discussed above where the increase in momentum reveals in-feasibility of a

path. As similar paths are available in the close vicinity, the next higher fitness value is close enough. The soft mutation re-decreases the value of the fitness function. Figure 5(d) is another very interesting graph. Here the fitness value decreases and converges and later re-decreases and re-converges. This is due to the fact that when the algorithm was at a complexity of 2, the algorithm optimized and converged. This continued till a complexity of 3 as well. When the algorithm entered a complexity of 4, the better and more optimized paths were found. As a result the value re-decreased and re-converged. The effect of condition 2 can also be seen in this graph.

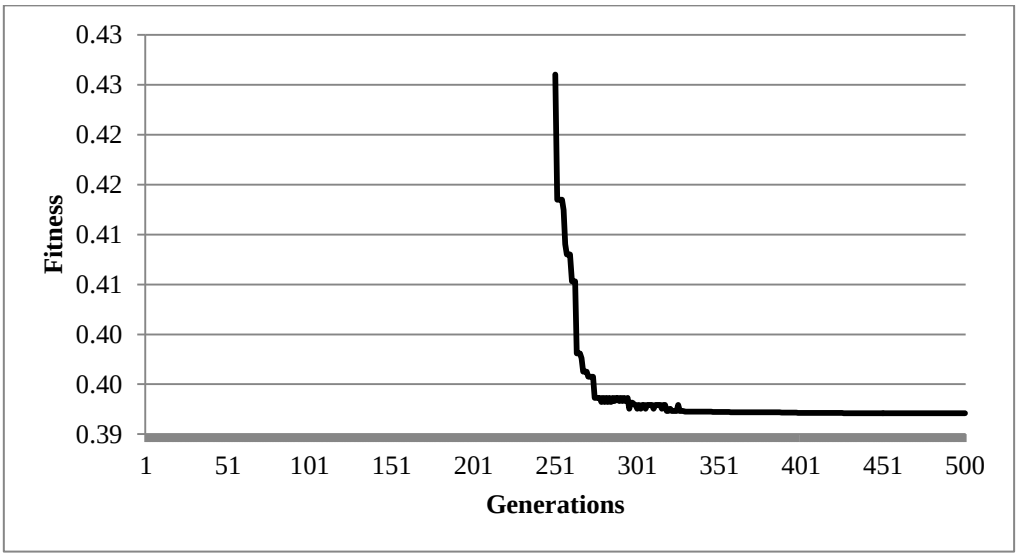

Figure 5(b): Graph showing best fitness v/s generation for the EA for map 2.

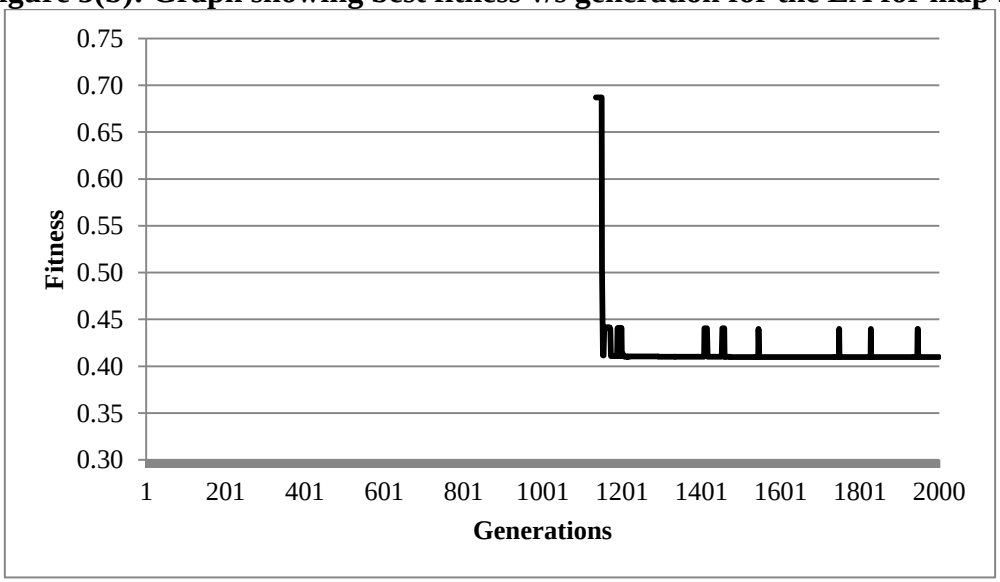

Figure 5(c): Graph showing best fitness v/s generation for the EA for map 3.

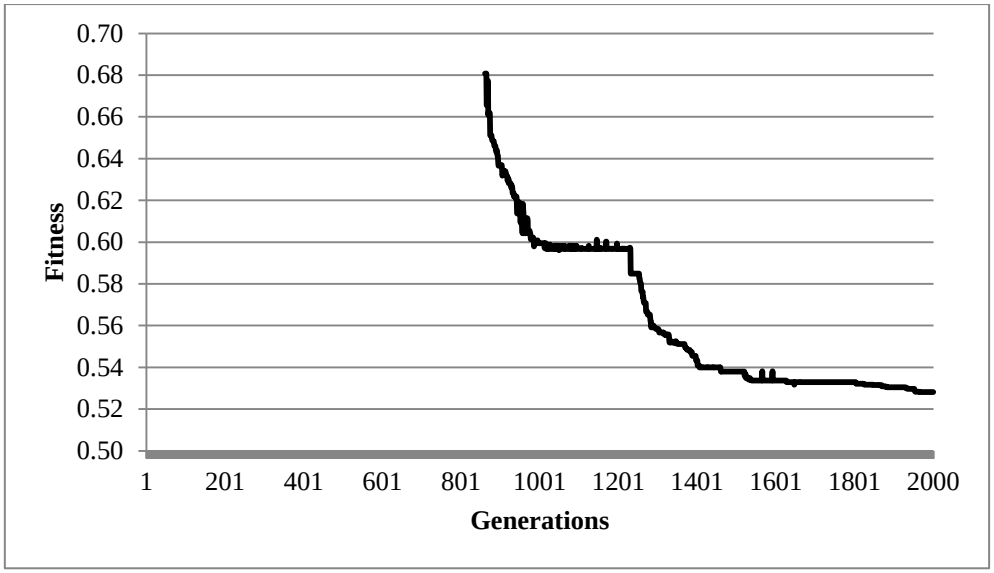

Figure 5(d): Graph showing best fitness v/s generation for the EA for map 4.

Another important factor that we would like to study is the total number of feasible solutions at every generation. This would give us an idea of how many individuals are actually contributing towards the search of the most optimal path or solution. The graph between the numbers of feasible solutions for all the 4 maps discussed is plotted in figures 6(a), 6(b), 6(c) and 6(d).

The graph shown in figure 6(a) represents a very expected Gaussian behavior that we had introduced in the total number of individuals. Figure 6(b) has no feasible solution till the minimal complexity is achieved. Afterwards the expected Gaussian increase can be observed. However we see a large oscillatory behavior here. These oscillations depict the randomness of the algorithm in search for new solutions. Recall that the hard mutation and insert genetic operator were random in their functionality. The increase in randomness is again attributed due to the decreasing momentum values which reveal the infeasibility of many individuals that were otherwise feasible in lower generations. This trend is magnified when we look at figure 6(c). This figure reveals vital information that the total number of feasible solutions is very less as compared to the total number of solutions in the population pool. The reason for the same is the map on which the algorithm is applied. Even a small change in the coordinate values can make a good solution infeasible. The higher complexity or higher generation regions of the graph make the path more flexible which adds to the possibility of infeasibility. This is the reason why the oscillation is of even larger magnitude. Similar discussion may be done in figure 6(d) as well. Here the two levels denote the convergence of the algorithm at a complexity of *2* and *4* as discussed in the earlier paragraph. As the complexity *4* possibility arrives, many individuals migrate to a higher complexity region. In this migration they happen to lose their feasibility due to the random nature of insert operator. This makes the number of feasible solutions very low. As the algorithm continues, they regain their feasibility by the repair operation.

The last parameter of study is the execution time for various generations. The execution time per generation is plotted against generations in figure 7(a), 7(b), 7(c) and 7(d). 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.

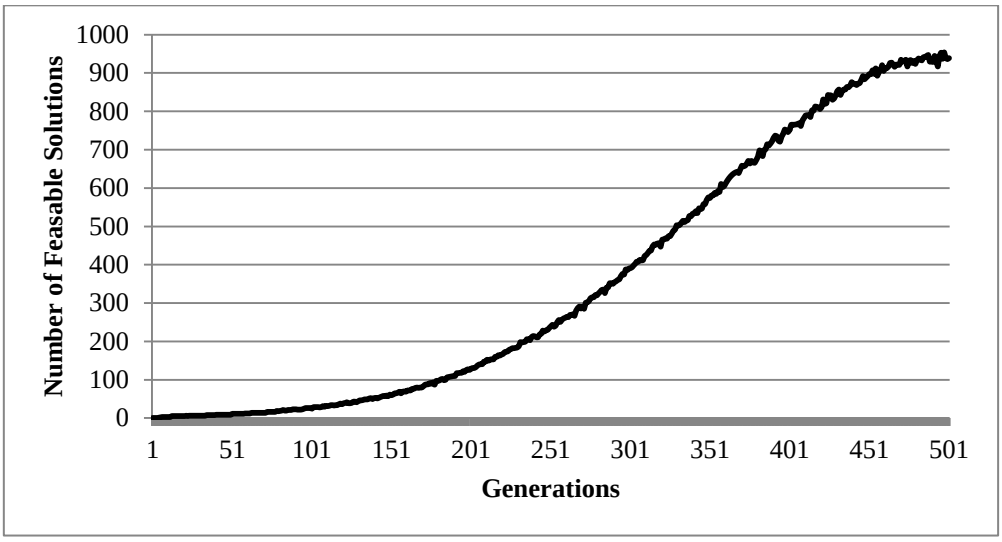

Figure 6(a): Graph showing number of feasible solutions v/s generation for the EA for map 1.

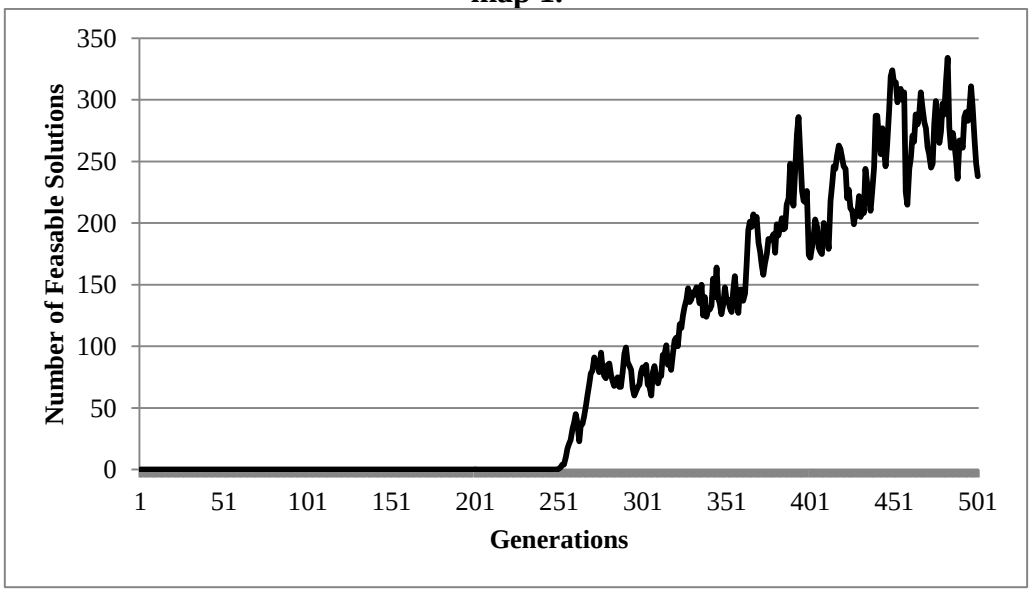

Figure 6(b): Graph showing number of feasible solutions v/s generation for the EA for map 2.

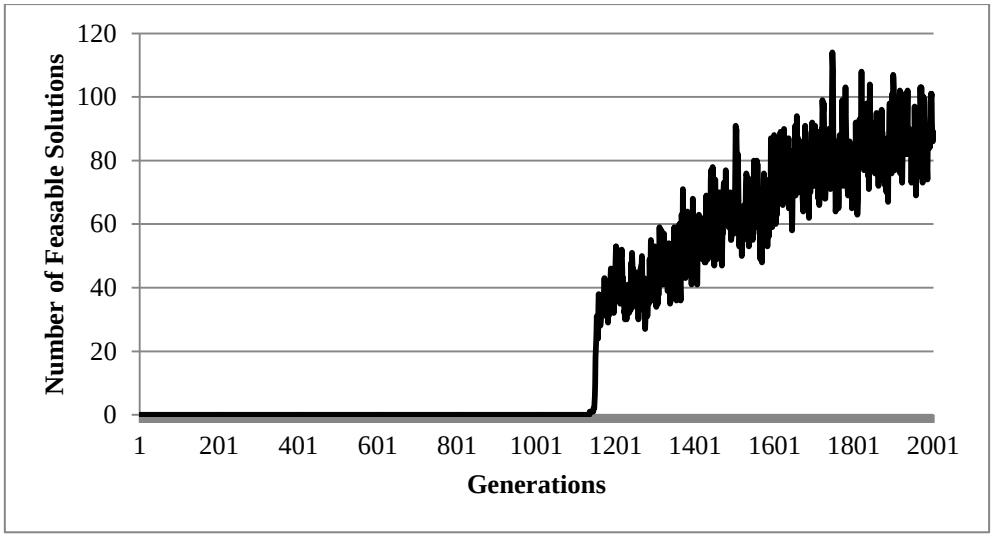

Figure 6(c): Graph showing number of feasible solutions v/s generation for the EA for map 3.

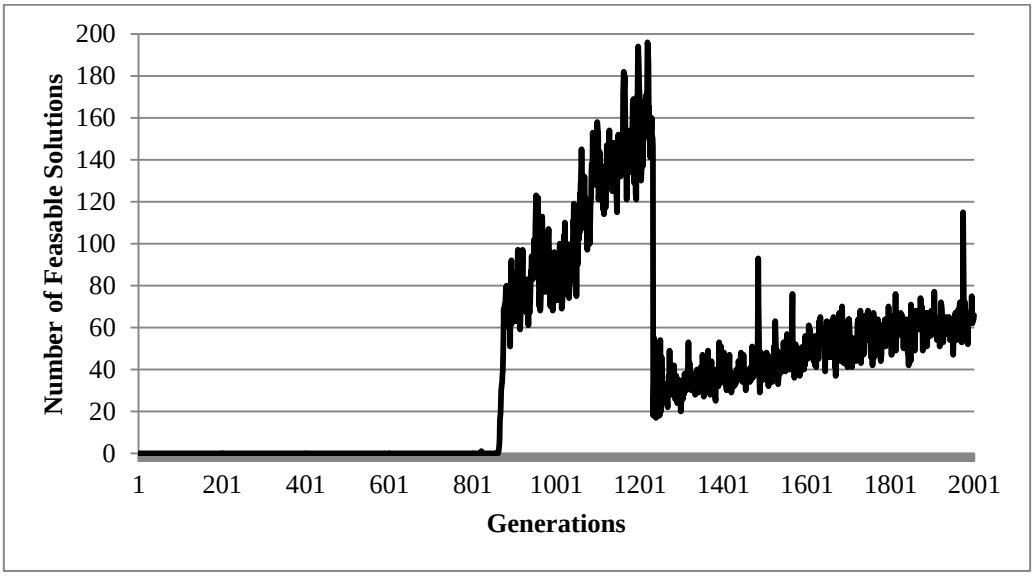

Figure 6(d): Graph showing number of feasible solutions v/s generation for the EA for map 4.

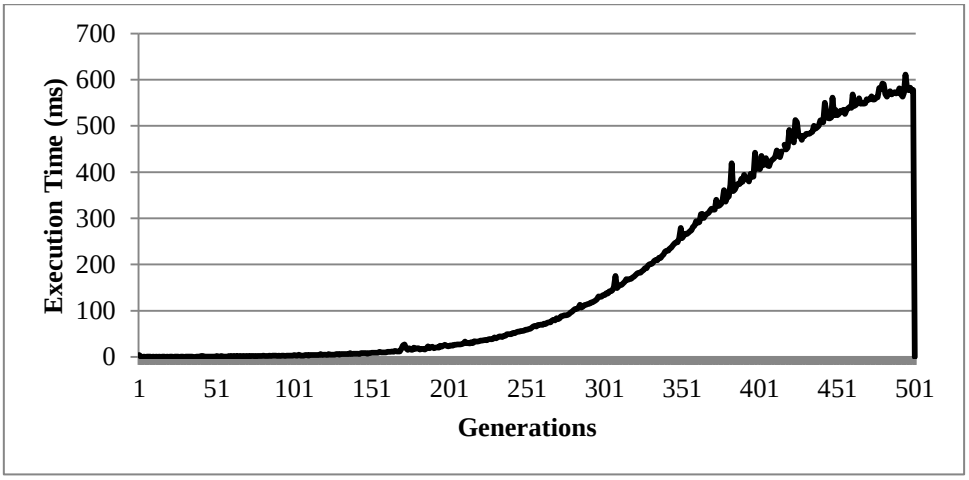

Figure 7(a): Graph showing time of execution v/s generation for the EA for map 1.

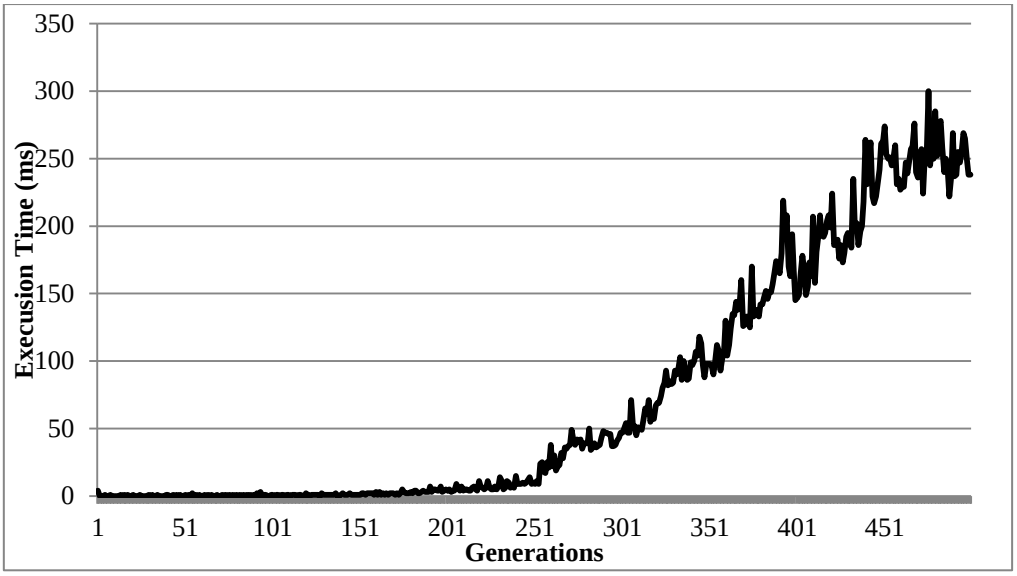

Figure 7(b): Graph showing time of execution v/s generation for the EA for map 2.

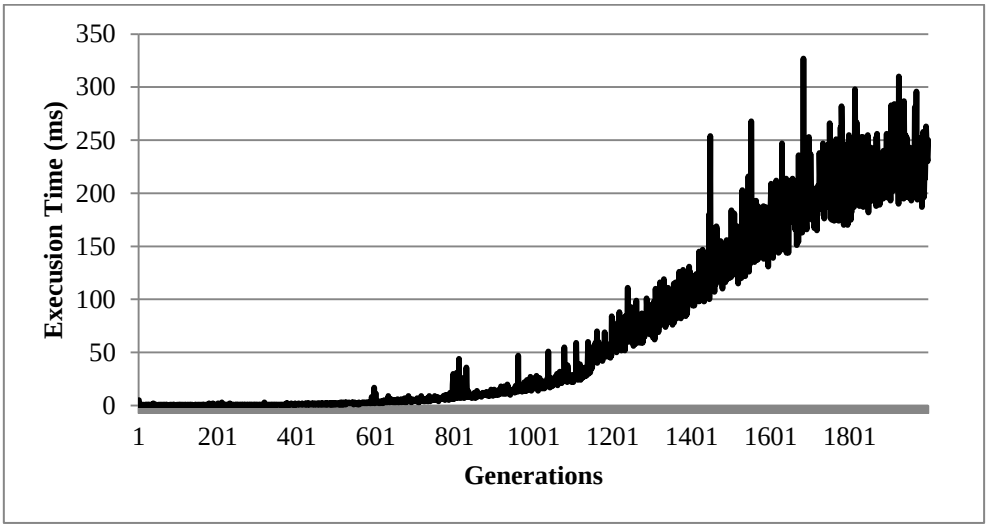

Figure 7(c): Graph showing time of execution v/s generation for the EA for map 3.

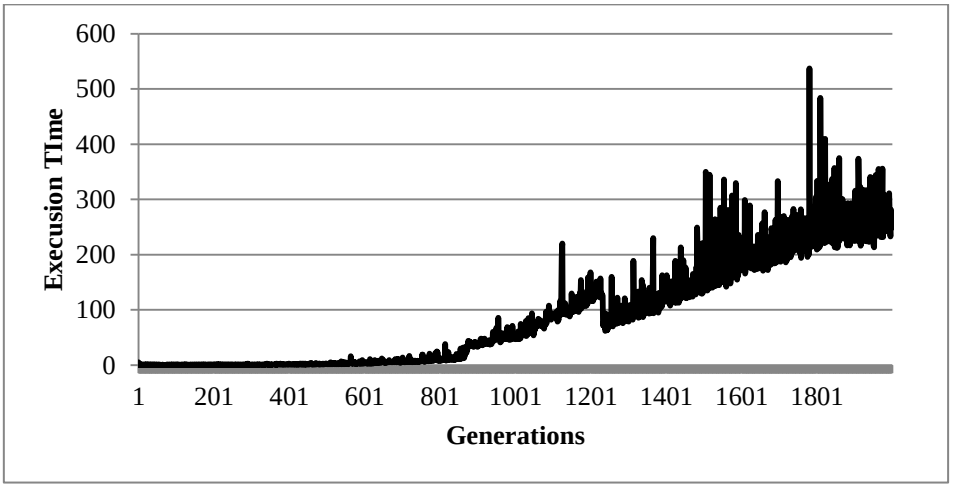

Figure 7(d): Graph showing time of execution v/s generation for the EA for map 4.

8.1 Tradeoff between the parameters of Multiobjective Optimizations

In the fitness function we had introduced the parameters α, β and γ for path length (l), distance from obstacle (d) and complexity (c_{max}). During our discussion as well as by experimentation we observed that it was very easy to work over with the distance from obstacle factor which could be easily managed in most of the non-crowded maps. Mostly this factor retained one of its lowest values during experimentation.

The other two factors were a good point of discussion while we were discussion the algorithm. We stated that sometimes we may wish to traverse through high complexity areas compromising with the complexity and speed but keeping the length optimal. Many times we would like to compromise with the path length and keep the speed high with low path complexity. We experimentally verified the same where the choice was controlled by varying the multiobjective parameters. Consider the graph used in fourth case. We present two paths traced by the robot in figure 8(a) and 8(b). Figure 8(a) shows the graph with the values of α, β and γ as *0.5, 0.5* and *0.* Here we can see that the shortest path was followed. Figure 8(b) shows the path for the values of α, β and γ as *0.33, 0.33* and *0.33.* Here the straight path was preferred.

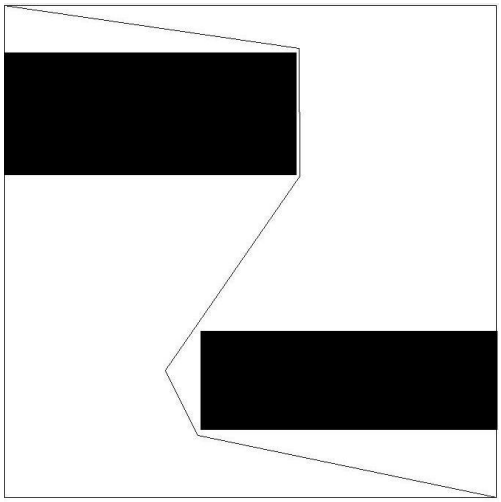

Figure 8(a): The path traced by the robot with preference to shortest path

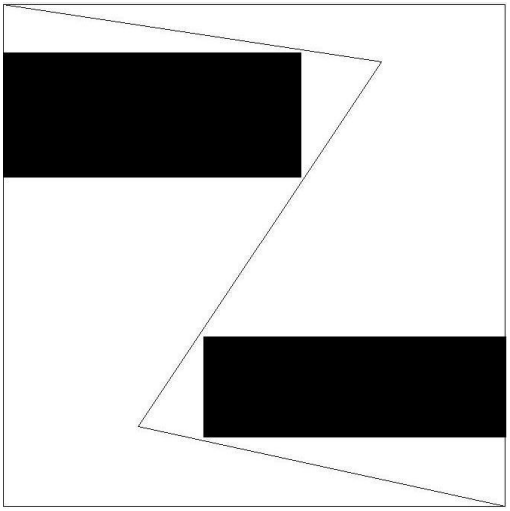

Figure 8(b): The path traced by the robot with preference to least complex path

Another similar case was found with the same set of multiobjective optimization parameter values in map given in figure 9(a) and 9(b). In figure 9(a) a straight path was preferred while in figure 9(b) a smaller complexity path was preferred.

8.2 Effect of Momentum

The last set of discussion that needs experimental verification is the effect of momentum. We discussed that increased momentum would lead to shorter time of execution at the same time may result in reporting many infeasible solutions as feasible. We also stated that large momentum would wrongly quote the value of the parameter distance to obstacle (d). We study the effect of momentum on each of these domains by applying variable momentum over a set of randomly generated paths solutions. A total of *1000* 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 10. Figure 11 gives the relation between the momentum and the predicting capability of the fitness function regarding the feasibility of the solution. Figure 12 shows the relation between the momentum and mean deviation of the physical minimum distance from robot to obstacle. As per our earlier discussion, all distances greater than 10 are taken to be zero.

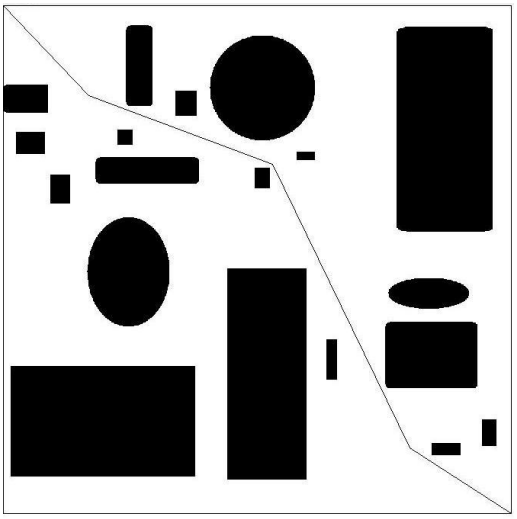

Figure 9(a): The path traced by the robot with preference to shortest path

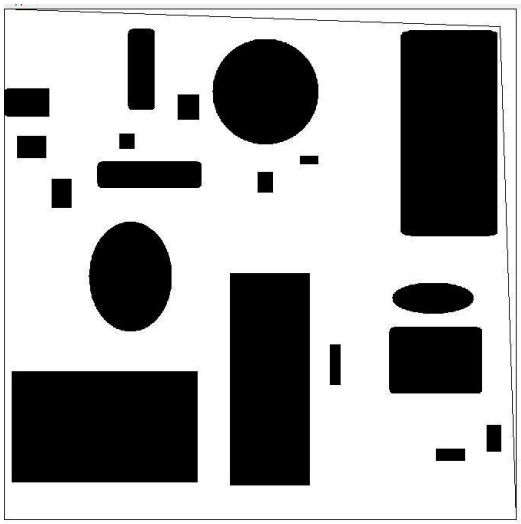

Figure 9(b): The path traced by the robot with preference to least complex path

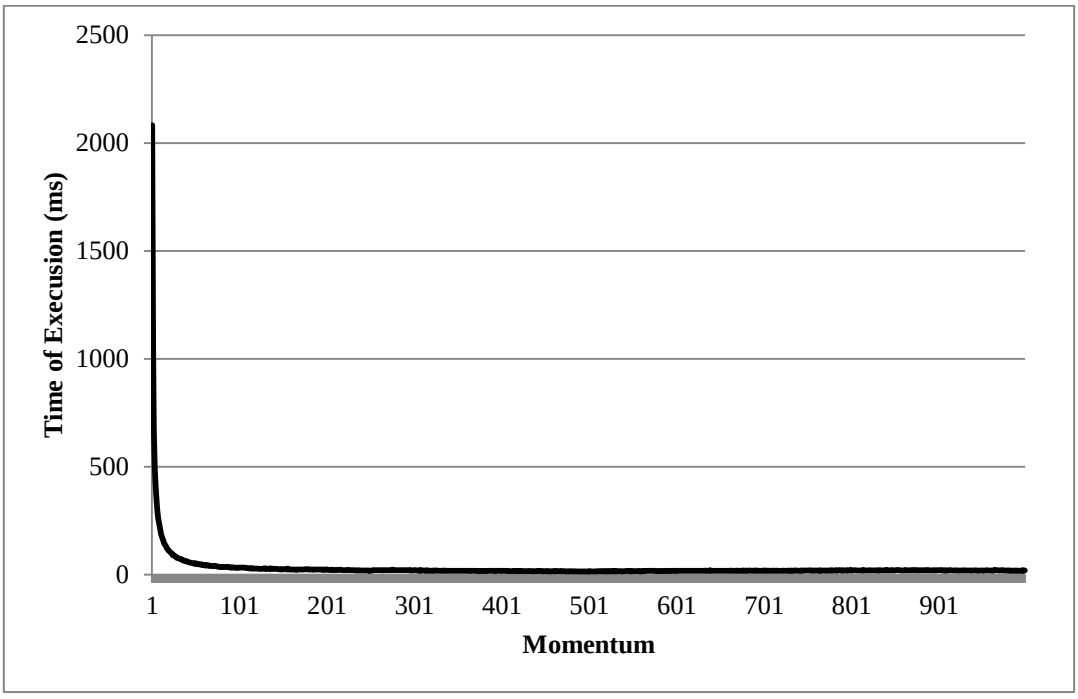

Figure 10: Relation between time of execution and momentum

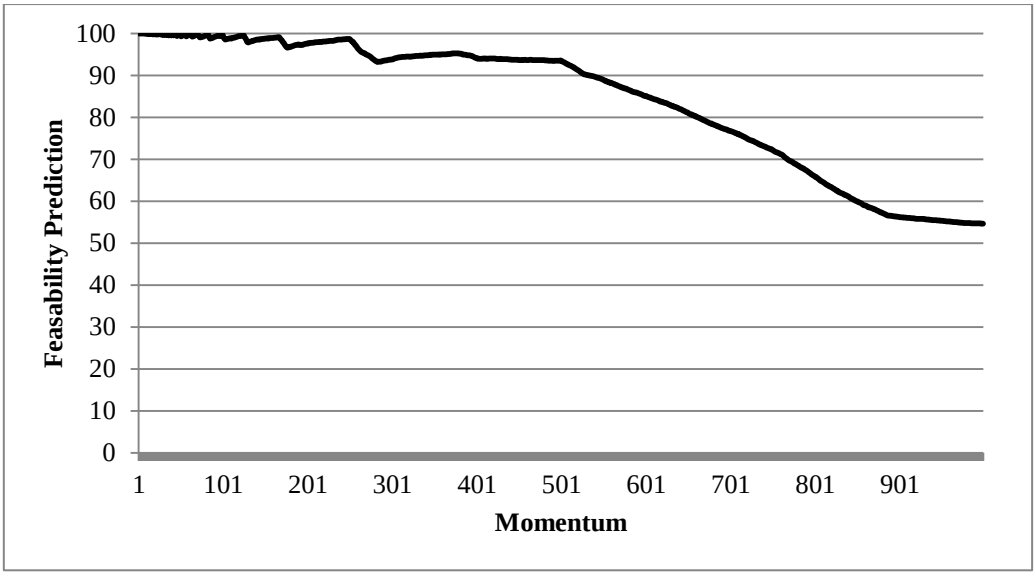

Figure 11: Relation between feasibility prediction and momentum

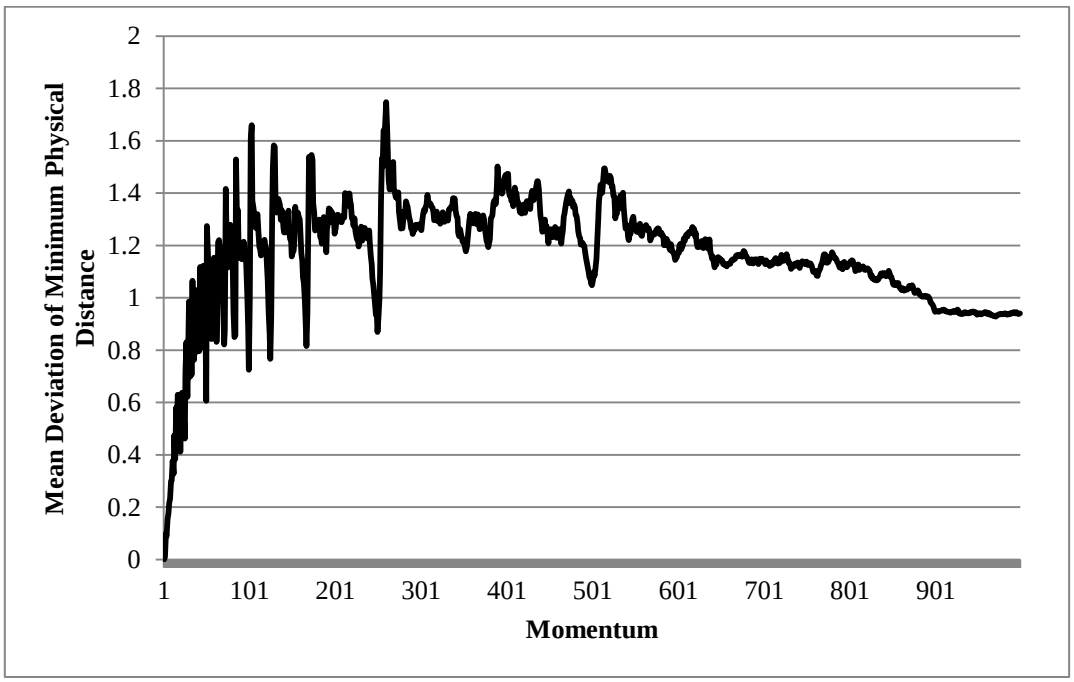

Figure 12: Relation between Mean Deviation of Minimum Physical Distance and Momentum

An interesting observation is in figure 12 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 deviations as momentum increases.

9. Conclusions

In this paper we proposed a novel approach to solving the problem of robotic path planning. The motivation was to evolve the final most optimized path considering the fact that the

complexity of the path is a completely unknown that depends largely upon the map. The algorithm optimized he fitness function by making use of a variable momentum based approach that decayed in a Gaussian manner. A similar approach was used for the number of individuals for the EA that increased in a Gaussian manner along with the number of generations. The fitness function used in the problem optimized the total path length (l), distance of the path from closest obstacle (d) and the total path complexity (c_{max}). The individual contribution of these could be controlled by the use of multiobjective optimization parameters of the algorithm.

We coded the algorithm and simulated it on a self-built simulator using JAVA framework. The simulator took the map as a JPEG input and showed the map and the evolved path. We presented four kinds 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 highly optimized in terms of the set optimization parameters. The solution in all cases was of the most optimal path length. The solution avoided going too close to the obstacle and always maintained a fair amount of distance from it. Also the robot path hence generated avoided too many turns what was referred to in the paper as the path complexity.

The results conveyed a lot of additional information to us as far as the problem is considered. We observed that the multiobjective optimization parameters could be easily varied to obtain a tradeoff between the path length and complexity. We showed that the algorithm was flexible to optimize both of them to varying degrees. The final evolved path completely changed when we made a transition in the preference of one from the other. The actual choice depends a lot on the physical constraints of the robot. A speed restricted robot may chose to go for path optimization and a carlike robot built to operate at higher speeds might want to optimize the total path complexity to be able to run at high speeds. Another interesting behavior was the momentum that showed us signs of expected trends as well as randomness when measured on the parameters of time, feasibility prediction, accuracy and distance to obstacle prediction. While the trends were attributed to the reasons illustrated in theory, the randomness stated a typical behavior of the parameter.

The major motive of this algorithm was to make an algorithm where the underlying assumption of the path complexity is not present. The complexity can be decided on its own by the algorithm that best suits the map. By using diversity preservation we also kept a room for the simultaneous co-existence of various paths with different complexities. This relieves the use of the algorithm to work over one of the major parameter. It can hence be thought that we are taking a step forward in the direction of making the problem of path planning parameter-less. It would be wrong to state that the nature had set some parameters for life to evolve. Or that the child has some parameters that he sets before planning his route from kitchen to living room. The same must be true for robots as well where the paths only depend upon the maps and not the other parameters. The other parameters impose a hindrance in effective path planning.

Now one may argue that in order to replace a single parameter we happened to add too many parameters in the conventional evolutionary path planning algorithm. This should increase the problem in place of decreasing. Here we state two major points. The first is the role of parameter. Path complexity was a major parameter that we replaced with a set of passive parameters. While experimentation we observed that the added parameters depended little on the robotic and map constraints and hence could be assumed to have global constant values.

The other major point what differs from the natural counterparts is that the robotic processing capability and the physical constraints are unknown and influence the parameters. A child knows his limitations and has a lifetime to train him according to those limitations. People driving cars have initial problems driving buses or trucks. That is precisely what we are doing when we set parameter. We are trying to optimize the algorithm according to the unknown constraints and demands.

While we have been able to effectively solve the problem of robotic path planning, there is 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 present algorithm does not speak about the dynamic obstacles at all. Also the work of intelligent planning system over the introduced parameters may be built to analyze the algorithm working and set the various parameters accordingly. This would complete our endeavor of having robotic path planning completely parameter less where the various parameters are set to their most optimal values automatically.

References

Ahuactzin, Juan Manuel ; Talbi, El-Ghazali; Pierre Bessière; Mazer, Emmanuel, 'Using Genetic Algorithms for Robot Motion Planning', *Lecture Notes In Computer Science*; Vol. 708, pp 84 – 93, 1991

Alvarez, A.; Caiti, A.; Onken, R., 'Evolutionary path planning for autonomous underwater vehicles in a variable ocean,' *Oceanic Engineering, IEEE Journal of* , vol.29, no.2, pp. 418-429, April 2004

Ashiru, I.; Czarnecki, C., 'Optimal motion planning for mobile robots using genetic algorithms ,' *Industrial Automation and Control, 1995 (I A & C'95), IEEE/IAS International Conference on (Cat. No.95TH8005)*, pp.297-300, 5-7Jan 1995

Badran, Khaled M. S.; Rockett, Peter I, 'The roles of diversity preservation and mutation in preventing population collapse in multiobjective genetic programming', *Genetic and evolutionary computation, Proceedings of the 9th annual conference on*, pp 1551 - 1558, 2007

Bakker, B.; Zivkovic, Z.; Krose, B., 'Hierarchical dynamic programming for robot path planning,' *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on* , vol., no., pp. 2756-2761, 2-6 Aug. 2005

Brooks, R., 'Visual map making for a mobile robot,' *Robotics and Automation. Proceedings. 1985 IEEE International Conference on* , vol.2, no., pp. 824-829, Mar 1985

Carpin, Stefano; Pagello, Enrico, 'An experimental study of distributed robot coordination', *Robotics and Autonomous Systems*, Vol 57, Issue 2, pp 129-133, 28 Feb 2009

Chen, Liang-Hsuan; Chiang, Cheng-Hsiung, 'New approach to intelligent control systems with self-exploring process,' *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* , vol.33, no.1, pp. 56-66, Feb 2003

Chunmiao Wang; Soh, Y.C.; Han Wang; Hui Wang, 'A hierarchical genetic algorithm for path planning in a static environment with obstacles,' *Electrical and Computer Engineering, 2002. IEEE CCECE 2002. Canadian Conference on* , vol.3, no., pp. 1652-1657 vol.3, 2002

Cortes, J.; Jaillet, L.; Simeon, T., 'Disassembly Path Planning for Complex Articulated Objects,' *Robotics, IEEE Transactions on* , vol.24, no.2, pp.475-481, April 2008

Deb, Kalyanmoy, 'Multi-objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems', *Evolutionary Computation,* Vol 7, No 3, pp 205-230, 1999

Dittrich, Peter; Bürgel, Andreas ; Banzhaf, Wolfgang, 'Learning to move a robot with random morphology ', Vol 1468/1998, *Springer Lecture Notes in Computer Science,* pp 165-178, Apr 2006

Doitsidis, L.; Tsourveloudis, N. C.; Piperidis, S., 'Evolution of Fuzzy Controllers for Robotic Vehicles: The role of Fitness Function Selection,' *Journal of Intelligent and Robotic Systems*, 23 Apr 2009

Floreano, Dario; Mattiussi, Claudio, *Bio-Inspired Artificial Intelligence*, MIT Press, Sep 2008

Garrido, S.; Moreno, L., Blanco, D., 'Exploration of 2D and 3D Environments using Voronoi Transform and Fast Marching Method,' *Journal of Intelligent and Robotic Systems,* Vol 55, Issue 1, pp 55 - 80 , 2009

Ge, Shuzhi Sam; Lewis Frank L., *Autonomous Robots*, CRC Press. 2006

Gerke, M., 'Genetic path planning for mobile robots,' *American Control Conference, 1999. Proceedings of the 1999* , vol.4, no., pp.2424-2429 vol.4, 1999

Han, W.; Baek, S.; Kuc, T., 'GA Based On-Line Path Planning of Mobile Robots Playing Soccer Games', *Circuits and Systems. Proc. of the 40th Midwest Symposium on,* Part 1 of 2, vol 1, 1997,.

Hazon, Noam; Kaminka, Gal A., 'On redundancy, efficiency, and robustness in coverage for multiple robots', *Robotics and Autonomous Systems,* Vol 56, Issue 12, pp 1102-1114, 31 December 2008

Holland, J. H. , 'Adaptation in Natural and Artificial Systems', *Ann Arbor, Univ. Michigan Press*, 1975.

Hui, Nirmal Baran; Pratihar, Dilip Kumar, 'A comparative study on some navigation schemes of a real robot tackling moving obstacles', *Robotics and Computer-Integrated Manufacturing,* Vol 25, Issues 4-5, pp 810-828, Aug-Oct 2009

Jan, G.E.; Ki Yin Chang; Parberry, I., 'Optimal Path Planning for Mobile Robot Navigation,' *Mechatronics, IEEE/ASME Transactions on* , vol.13, no.4, pp.451-460, Aug. 2008

Jing Xiao; Michalewicz, Z.; Lixin Zhang; Trojanowski, K., 'Adaptive evolutionary planner/navigator for mobile robots,' *Evolutionary Computation, IEEE Transactions on* , vol.1, no.1, pp.18-28, Apr 1997

Jolly, K.G.; Kumar, R. Sreerama; Vijayakumar, R., 'A Bezier curve based path planning in a multi-agent robot soccer system without violating the acceleration limits,' *Robotics and Autonomous Systems,* Vol 57, Issue 1, pp 23-33, 31 Jan 2009

Juidette, H.; Youlal, H., 'Fuzzy dynamic path planning using genetic algorithms', *Electron. Lett.* 36, pp 374-376, 2000,

Kala, Rahul et al, 'Mobile Robot Navigation Control in Moving Obstacle Environment using Genetic Algorithm, Artificial Neural Networks and A* Algorithm', *Computer Science and Information Engineering, IEEE Proceedings of the World Congress on, IEEE CSIE 2009,* pp 705-713, April 2009

Kambhampati, S.; Davis, L., 'Multiresolution path planning for mobile robots', Vol 2, Issue 3, pp 135-145, Sep 1986

Khoshnejad, M.; Demirli, K., 'Autonomous parallel parking of a car-like mobile robot by a neuro-fuzzy behavior-based controller,' *Fuzzy Information Processing Society, 2005. NAFIPS 2005. Annual Meeting of the North American,* pp 814- 819, 26-28 June 2005

Laumanns, Marco; Thiele, Lothar; Deb, Kalyanmoy; Zitzler, Eckart, 'On the Convergence and Diversity-Preservation Properties of Multi-Objective Evolutionary Algorithms,'

TIK Report No. 108, Computer Engineering and Networks Laboratory(TIK), Swiss Federal Institute of Technology (ETH) Zurich, May, 2001.

Mitchell, Melanie, *An introduction to genetic algorithms,* MIT Press, Feb 1998

Nam, D.; Singh, H.; Muench-Casanova, S.; Gerhart, G.; Goetz, R., 'Scaling a neuro fuzzy system and applications to 3D visualization and robot path planning,' *IFSA World Congress and 20th NAFIPS International Conference, 2001. Joint 9th ,* vol.2, no., pp.1074-1079 vol.2, 25-28 July 2001

Noguchi, Noboru; Terao, Hideo, 'Path planning of an agricultural mobile robot by neural network and genetic algorithm', *Computers and Electronics in Agriculture,* Vol 18, Issues 2-3, pp 187-204, Aug 1997

O'Hara, K.J.; Walker, D.B.; Balch, T.R., 'Physical Path Planning Using a Pervasive Embedded Network,' *Robotics, IEEE Transactions on ,* vol.24, no.3, pp.741-746, June 2008

Patnaik, Srikanta; Jain, Lakhmi C.; Tzafestas, Spyros G., Resconi, Germano; Konar, Amit, *"Innovations in Robot Mobility and Control"*, Springer 2005

Peasgood, M.; Clark, C.M.; McPhee, J., 'A Complete and Scalable Strategy for Coordinating Multiple Robots Within Roadmaps,' *Robotics, IEEE Transactions on,* vol.24, no.2, pp.283-292, April 2008

Podsedkowski, Leszek; Nowakowski, Jacek; Idzikowski, Marek; Vizvary, Istvan, "A new solution for path planning in partially known or unknown environment for nonholonomic mobile robots", *Robotics and Autonomous Systems,* vol 34, issues 2-3, pp 145-152, 28 Feb 2001

Pozna, Claudiu; Troester, Fritz; Precup, Radu-Emil; Tar, Jozsef K.; Preitl, Stefan, 'On the design of an obstacle avoiding trajectory: Method and simulation,' *Mathematics and Computers in Simulation,* Vol 79, Issue 7, pp 2211-2226, March 2009

Pradhan, Saroj Kumar; Parhi, Dayal Ramakrushna; Panda, Anup Kumar; 'Fuzzy logic techniques for navigation of several mobile robots', *Applied Soft Computing,* Vol 9, Issue 1, pp 290-304, January 2009

Sadati, N.; Taheri, J., 'Genetic algorithm in robot path planning problem in crisp and fuzzified environments,' *Industrial Technology, 2002. IEEE ICIT '02. 2002 IEEE International Conference on ,* vol.1, no., pp. 175-180 vol.1, 2002

Sedighi, K.H.; Ashenayi, K.; Manikas, T.W.; Wainwright, R.L.; Heng-Ming Tai, 'Autonomous local path planning for a mobile robot using a genetic algorithm,' *Evolutionary Computation, 2004. CEC2004. Congress on ,* vol.2, no., pp. 1338-1345 Vol.2, 19-23 June 2004

Sheng, Weihua; Xi, Ning; Song, Mumin; Chen, Yifan, 'Robot Path Planning for Dimensional Measurement in Automotive Manufacturing', *J. Manuf. Sci. Eng.,* issue 127, pp 420-429, 2005

Shibata, T.; Fukuda, T.; Kosuge, K.; Arai, F., 'Selfish and coordinative planning for multiple mobile robots by genetic algorithm,' *Decision and Control, 1992., Proceedings of the 31st IEEE Conference on ,* vol., no., pp.2686-2691 vol.3, 1992

Shibata, T.; Fukuda, T.; Tanie, K., 'Fuzzy critic for robotic motion planning by genetic algorithm in hierarchical intelligent control,' *Neural Networks, 1993. IJCNN '93-Nagoya. Proceedings of 1993 International Joint Conference on ,* vol.1, no., pp. 770-773 vol.1, 25-29 Oct. 1993

Shukla, Anupam; Tiwari, Ritu; Kala, Rahul, 'Mobile Robot Navigation Control in Moving Obstacle Environment using A* Algorithm', *Intelligent Systems Engineering Systems through Artificial Neural Networks,* ASME Publications, Vol. 18, pp 113-120, Nov 2008

Shukla, Anupam; Tiwari, Ritu; Kala, Rahul, *Real Life Applications of Soft Computing*, CRC Press, Feb 2010

Sud, A.; Andersen, E.; Curtis, S.; Lin, M.C.; Manocha, D., 'Real-Time Path Planning in Dynamic Virtual Environments Using Multiagent Navigation Graphs,' *Visualization and Computer Graphics, IEEE Transactions on* , vol.14, no.3, pp.526-538, May-June 2008

Sugihara, Kazuo; Yuh, Junku, 'GA-Based Motion Planning For Underwater Robotic Vehicles', *Proc. 10th International Symp. on Unmanned Untethered Submersible Technology. Autonomous Undersea Systems Institute*, pp 406—415, 1996

Suh, S.-H.; Shin, K.G., 'A variational dynamic programming approach to robot-path planning with a distance-safety criterion,' *Robotics and Automation, IEEE Journal of,* Vol 4, Issue 3, pp 334-349, Aug 2002

Thrun, Sebastian, 'Learning metric-topological maps for indoor mobile robot navigation', *Artificial Intelligence,* Vol. 99, Issue 1, pp 21-71, Feb 1998

Toogood, R. ; Hong Hao; Chi Wong, 'Robot Path Planning Using Genetic Algorithms', *Systems, Man and Cybernetics, 1995. Intelligent Systems for the 21st Century., IEEE International Conference on,* Vol. 1, pp 489-494, 22-25 Oct 1995

Tsai, Chi-Hao; Lee, Jou-Sin; Chuang, Jen-Hui, 'Path planning of 3-D objects using a new workspace model,' *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* , vol.31, no.3, pp.405-410, Aug 2001

Tu, J.; Yang, S., 'Genetic algorithm based path planning for a mobile robot,' *Robotics and Automation, Proc. of IEEE Intl. Conf on,* Taiwan, September 2003

Urdiales, C.; Bandera, A.; Arrebola, F.; Sandoval, F., 'Multi-level path planning algorithm for autonomous robots,' *Electronics Letters* , vol.34, no.2, pp.223-224, 22 Jan 1998

Vadakkepat, P.; Kay Chen Tan; Wang Ming-Liang, 'Evolutionary artificial potential fields and their application in real time robot path planning,' *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on* , vol.1, no., pp.256-263 vol.1, 2000

Wang , Chunmiao; Soh, Y.C.; Wang, Han; Wang, Hui, 'A hierarchical genetic algorithm for path planning in a static environment with obstacles,' *Electrical and Computer Engineering, 2002. IEEE CCECE 2002.* Canadian, vol 3, pp 1652- 1657, 2002

Woong-Gie Han; Seung-Min Baek; Tae-Yong Kuc, 'Genetic algorithm based path planning and dynamic obstacle avoidance of mobile robots,' *Systems, Man, and Cybernetics,* 1997. *'Computational Cybernetics and Simulation'., 1997 IEEE International Conference on* , vol.3, no., pp.2747-2751 vol.3, 12-15 Oct 1997

Yang, Simon X.; Meng, Max, 'An efficient neural network approach to dynamic robot motion planning,' *Neural Networks,* Vol 13, Issue 2, pp 143-148, March 2000

Yanrong Hu; Yang, S.X., 'A knowledge based genetic algorithm for path planning of a mobile robot,' *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on* , vol.5, no., pp. 4350-4355 Vol.5, 26 April-1 May 2004

Zhu, D.J.; Latombe, J.-C., 'New heuristic algorithms for efficient hierarchical path planning ,' *Robotics and Automation, IEEE Transactions on* , vol.7, no.1, pp.9-20, Feb 1991