Global Path Planning for Mobile Robots in Large-Scale Grid Environments using Genetic Algorithms

Maram Alajlan [||][¶], Anis Koubâa [**][¶][‡], Imen Châari [*][¶], Hachemi Bennaceur [||], Adel Ammar [||]

[¶] Cooperative Robots and Sensor Networks (COINS) Research Group, Saudi Arabia.
[**]Prince Sultan University, College of Computer and Information Sciences, Saudi Arabia.
[||] Research Unit of Sciences and Technology, Al-Imam Mohamed bin Saud University, Saudi Arabia.
[*] PRINCE Research Unit, University of Manouba (ENSI), Tunisia.
[‡] CISTER/INESC-TEC, ISEP, Polytechnic Institute of Porto, Porto, Portugal.

maram.ajlan@coins-lab.org, akoubaa@coins-lab.org, imen.chaari@coins-lab.org, hachemi@ccis.imamu.edu.sa, adel.ammar@ccis.imamu.edu.sa

***Abstract*—Global path planning is considered as a fundamental problem for mobile robots. In this paper, we investigate the capabilities of genetic algorithms (GA) for solving the global path planning problem in large-scale grid maps. First, we propose a GA approach for efficiently finding an (or near) optimal path in the grid map. We carefully designed GA operators to optimize the search process. We also conduct a comprehensive statistical evaluation of the proposed GA approach in terms of solution quality, and we compare it against the well-known A* algorithm as a reference. Extensive simulation results show that GA is able to find the optimal paths in large environments equally to A* in almost all the simulated cases.**

I. Introduction

Mobile robots global path planning is a hot research area. Indeed, in any mobile robot applications, the robot must be able to find its path to a certain goal position while avoiding obstacles and reducing the path cost. Designing an efficient path planning algorithm is an essential issue in mobile robots navigation since path quality influences the efficiency of the entire application. The constructed path must satisfy a set of optimization criteria including the traveled distance, the processing time, and the energy consumption. The traveled distance represents a typical metric of interest since it has a direct impact on time and energy.

In robotics, path planning, in its simplest form, consists in establishing an obstacle-free path for a robot starting from the initial (or current) location to the target location. Path planning makes part of the more generic navigation function of a mobile robot, that ensures how a robot moves effectively in its surrounding environment. The path planning particularly addresses the following question: *What is the best way to go there?*

In the literature, the path planning problem is influenced by two factors: (1) the environment, which can be static or dynamic, (2) the knowledge that the robot has about the environment; if the robot has a complete knowledge about the environment, this problem is known as global path planning. On the other hand, if the robot has an incomplete knowledge, this problem is classified as local path planning. In this paper, we consider the problem of global path planning in static large-scale grid environments.

Similar to other optimization problems, the global path planning problem can be solved with either exact or metaheuristic search methods. Exact methods have the advantage of being complete, meaning that they find the optimal solution if any exists. However, the scalability of these approaches is limited as their execution time increases exponentially with the size of the problem, which represents the size of the environment model in the case of global path planning. On the other hand, metaheuristic methods provide intelligent means to explore the search space, which significantly reduces the execution time, but pays the cost that they are uncertain with respect to finding the optimal solution, thus are qualified as incomplete methods.

A large number of metaheuristics have been proposed to solve the global path planning problem such as Ant Colony Optimization (ACO), Tabu Search, bees' algorithms, and Genetic Algorithms (GA), etc. Although the GA approach has been extensively used to solve the robot path planning problem as compared to the other approaches, most of the previous works limited their studies to small-size environments with different complexities. This represents the main motivation of this paper, we aim at investigating GA capabilities in large-scale environments.

In order to investigate more the properties of the GA approach for solving the global path planning problem for a mobile robot in large scale environments, we present an algorithm based on the GA approach and evaluate its performance. Different crossover operators are evaluated to help choosing the most appropriate one that improves the solution quality.

The effectiveness of the GA algorithm is demonstrated by an extensive simulation study, and compared against the A* method.

The rest of this paper is organized as follows. In Section II, we give an overview of some relevant works that addressed the robot global path planning using GA. Section III presents a detailed description of the GA algorithm. The results of simulation are presented in Section IV. Section V concludes the paper and discusses the future works.

II. Related Works

GA is a heuristic method invented in 1975 by John Holland [1]. It is based on the laws of natural selection and genetics. It starts with a set of initial candidate solutions for the problem. Each set of solutions is called a population or generation, and each individual solution is called a chromosome. For robot path planning each chromosome represents a path. Usually, the chromosome consists of start position, goal position, and intermediate positions crossed by the mobile robot. These positions represent genes of the chromosome. The evolution usually starts from a population of randomly generated chromosomes, and it includes four basic steps: fitness evaluation, selection, crossover, and mutation to create the subsequent generations.

The GA approach has been extensively used for robot path planning: In [2], Nearchou proposed a GA approach based on visibility graphs, with fixed length chromosomes consisting of string of bits, where each bit corresponds to a specific vertex in the graph. A reordering operator was added to GA aiming at enhancing performance. The performance of the proposed approach was compared with hill-climbing and simulated annealing techniques. The results show a much better performance for GA approach as the complexity of the environment grows and the algorithm was capable of determining a near-optimal solution.

Besides this old work, the GA approach is still being investigated in recent works, including [3]-[4]. In [3], the authors presented a path planning algorithm based on a pure GA. In their simulation work, they tested their algorithm in several environments that differ by their complexities. The environments were classified in two categories: with and without obstacles. The authors evaluated the path cost while varying the population size and the number of iterations; they proved that the algorithm was effective as it is able to find the optimal solution for all the different environments. In [4], the path planning problem with sub-path constraints has been tackled. The constraints on the sub-path include particular paths that must be incorporated in the solution. Obstacle-free and obstacle-based environments were considered. A GA with a fixed-length chromosome has been used to solve the problem of obstacle free path planning. In [5], authors proposed GA with only the crossover operator to improve execution time and computational cost. They used adaptive population size and fixed length chromosomes, each path is represented by two chromosomes, one for x-coordinate and one for y-coordinate. In [6], authors proposed modified fitness function and selection operator. They also added a new operator named "modification" in order to avoid collision in paths. They adopted fixed length chromosomes where each gene represents a cell in the grid.

In addition to the variation of GA basic characteristics, some works proposed using new techniques to generate initial population for GA. In [7], the authors proposed using rough sets reduction theory to enhance and reduce the size of initial population. In [8], fast random search method based upon probability was proposed to ensure that individuals of the initial population have a higher advantage.

In the literature, some other research efforts proposed a hybrid algorithm combining GA with other approaches such as A*, ACO, etc. In [9], Zhao et al. presented a modified GA to solve global path planning in case of disasters. They used a novel method to generate the initial population of the algorithm which based on a position information feedback generated using the ACO approach and a priority grouping method to make sure that the initial population contains only feasible paths. In their simulation work, the authors claimed that their proposal provide better collision-free path in a higher speed. In [10], the authors proposed a hybrid technique combining GA and the potential field method to solve the global path planning. The potential field method is applied to generate the initial feasible paths population of GA (collision-free paths), then applied to the paths of the new generation produced after each iteration of GA. The authors implemented different strategies classified into categories: (1) the diversity methods such as crossover, mutation, and immigration. (2) The memory strategy which clones the best solution to the next generation. In their simulation work, the authors tested the algorithm in 100*100 environments; they varied the number and the size of obstacles (maximum 8 obstacles). They tested different scenarios in which they used different combinations of the aforementioned strategies. They demonstrated that the algorithm provides optimal solutions in case of high diversity (high crossover, mutation and immigration probabilities) with the use of the memory strategy. They also proved that the mutation has no impact. Our work differs in the fact that we considered only GA based planner and we evaluated larger maps with size up to 2000*2000 grid cells.

A comparative study between different metaheuristic approaches classified as trajectory-based and population-based approaches for solving the global path planning problem of mobile robots was conducted in [11]. Three methods were evaluated namely tabu search, simulated annealing and GA. It was demonstrated through simulations that simulated annealing outperforms the other planners in terms of execution time, while tabu search was proved to provide the best solution in terms of path length. The difference with our work is that we designed a different GA algorithm approach with new operators that we tested against A* algorithm for different large-scale and random maps with different obstacle ratios, whereas in [11] global comparison between different approaches where performed on one example of a campus map. In [12], the authors proposed a multi-objective GA using the elitist non-dominated sorting GA to demonstrate the flexibility of evolutionary computing approaches in solving large-scale and multi-objective optimization problems. The objective was to optimize path length, path safety, and path smoothness. Using simulations, the GA planner was evaluated with different size environments up to 128*128 grid cells with obstacle ratios up to 91%. In our work, we evaluated grids with size up to 2000*2000 cells for very large-scale grid environments, which makes substantial difference with previous works.

All of these previous works limited their studies to small and medium size environments (maximum size 128*128 grid cell environments) with different complexity. In this work, we design a GA path planner and evaluate its performance of on large-size environments starting from 100*100 up to 2000*2000. To the best of our knowledge such very large-scale grid maps were not evaluated in the literature, which represents one of the contributions of this paper as compared to related works. We aim at investigating the GA capabilities in solving the global path planning problem in large environment in terms of solution quality. For that purpose, we compare path obtained by our GA planners against those found by the A* algorithm. The next section, present the GA planner and its control parameters.

III. GA Path Planner

In this section, we first present the environment model that we used for the global path planning problem of mobile robots, then we describe our GA approach.

A. Environment Model

In our model, we consider a grid-based map to represent the environment space of the robot, which is typical for mobile robots. One main reason that leads us to consider grid map is that we would like to integrate our path planners into the Robot Operating System (ROS) framework for robot application development, which uses occupancy grid maps.

Most of GA algorithms of the literature were applied to the traveling salesman problem TSP [13], [14]; which can be seen as a particular case of the path planning problem. Designing GA for solving the path planning problem needs to select and wisely adjust different parameters of the algorithm. Unlike in the TSP problem where the number of nodes of the path is fixed as it represents a complete tour, the length of the chromosome in global path planning problem is of a variable size because the lengths of feasible solutions may be different, and so the length of the optimal solution (in terms of number of nodes) is unknown. This makes the global path planning problem more challenging than the TSP problem, in that respect.

Before describing the proposed path planning algorithm, we first introduce some definitions, to provide a clear formulation of the path planning problem as a search problem and to discuss some issues of applying GA algorithm to the path planning problem.

1) Grid map model: The grid map representing the environment is divided into equal size cells. Each cell is represented by a unique number, starting from 0 for the top left cell, 1 for the next cell to the right and so on (see Fig. 1). We assume that the robot can move horizontally or vertically or diagonally from a cell to another free cell with costs 1, 1 and 1.4 respectively; it means that there are eight possible moves from each cell. We also assume that all obstacles are static and known in advance. Based on ROS convention, the grid map is represented by a two dimensional matrix where each cell is assigned one of two possible values: "0" for a free cell and "100" for a obstacle cell. A feasible solution is defined by a path from the initial cell to the goal cell traversing a certain number of free cells. The cost of a feasible solution is the sum of all costs of the moves along the associated path.

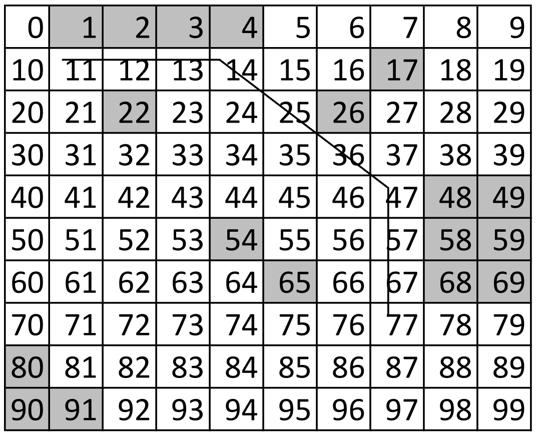


Fig. 1: A 10*10-cell grid-based environment model

2) Solution encoding: As mentioned above, in order to design an efficient GA path planner, it is judicious to carefully adjust different parameters that impact the GA performance. Unlike the TSP problem, in the global path planning problem the lengths of the feasible solutions are variable so it is mandatory to encode them with chromosomes having different sizes. We have associated a gene to each cell traversed by the path solution, so the chromosomes of each population may have different sizes. The fact of encoding solutions in such a way may lead to obtain infeasible path solutions when performing classical crossover operations. For this reason, we need to inject some conditions to carry out this operation correctly. This issue will be discussed and solved in the next section.

Fig. 1 shows an example of environment of size 10*10 and a feasible path. The path is encoded as (11, 12, 13, 14, 25, 36, 47, 57, 67, 77), the initial cell is 11, the goal cell is 77, and the cost of the path is (1+1+1+1.4+1.4+1.4+1+1+1 = 10.2).

B. The GA Path Planning Algorithm

The proposed algorithm is presented in Algorithm 1. The algorithm begins with a randomly generated initial population of candidate solutions. Each individual in the population must be a feasible path in the environment, i.e. it must be continuous and does not contain obstacles. The generation of initial population starts with generating an initial path from the start cell to the goal cell using the greedy approach based on Euclidean distance heuristic. Greedy approach is used to helps building initial paths in short times. To generate the subsequent paths in the initial population, the algorithm will choose a random intermediate cell, not in this initial path, which will be used to generate a new path from start to goal position across the selected intermediate cell. How to choose the initial population is a common problem in GA which affects the performance. For small environments it is sufficient to choose close intermediate cell each time. However, for large environments, we allow the algorithm to choose some far intermediate cells to give the GA a chance to explore better the environment.

When the initial population reaches its maximum size, the GA process starts the fitness evaluation. In each generation,

Algorithm 1 GA path planner pseudo-code

```
1:  Generate randomly the initial population (set of feasible
    paths) using the greedy approach based on Euclidean
    distance heuristic
2:  while (generation number <max generation number) do
3:      Fitness function
4:      Elitist selection
5:      Rank selection
6:      repeat
7:          choose randomly two paths (parents) from current
            generation
8:          if (random number generated < crossover rate) then
9:
10:             if (the parents have common cells) then
11:                 perform the crossover
12:                 move the resulting paths to the next generation
13:             else
14:                 choose other two parents
15:             end if
16:         else
17:             move the parents to the next generation
18:         end if
19:     until (next generation size < max population size)
20:     for each path in the next generation do
21:
22:         if (random number generated < mutation rate) then
23:             choose randomly a cell and replace it
24:
25:             if  (the resulting path feasible) then
26:                 replace the old path with the new one
27:             else
28:                 choose randomly two cells C1 and C2 from the
                    path
29:                 remove all the cells between C1 and C2
30:                 connect C1 and C2 by using the greedy approach
                    based on Euclidean distance heuristic
31:             end if
32:         end if
33:     end for
34: end while
```

the fitness function evaluates every individual in the population and gives each one a fitness value based on its quality. We used the *path length* as a primary criterion for fitness evaluation.

Then, the best individuals are selected to form the current generation. The selection process simulates the survival of the fittest principle of nature to make a search process. There are many selection strategies such as rank selection, elitist selection, roulette wheel selection, and tournament selection. Both elitist selection and rank selection were used in our algorithm. First, elitist selection will be executed to move the best individual in the current generation to the next generation without any change. The elitist selection was used to avoid losing the best individual because of the genetic operators randomness. Subsequently, in the rank selection, the individuals are sorted out in the ascending order according to their fitness values first. Then, the probability for each individual to be selected will be calculated such that individuals with higher fitness value (i.e. shortest path length) will have higher probability of being selected in the current generation. The rank selection is used because it gives each individual a chance to be selected, thus the diversity of the population increases.

After selecting all individuals of the current generation, they undergo two genetic operators: crossover and mutation. The objective of these operators is to create new generation of individuals from the current individuals that are shown to be temporary good.

The crossover operator (also called recombination) simulates the process of gene recombination. It is used to recombine two candidate solutions called parents to get better solutions called offspring. There are many crossover strategies such as one-point crossover, two-point crossover, and uniform crossover. In our algorithm, we used traditional one-point and two-point crossover operators, in addition to a modified crossover operator that we proposed in [15], aiming at comparing their performance for the robot path planning problem.

Fig. 2 illustrates the three crossover operators used in our algorithm. The one-point crossover is performed by randomly choosing one crossing position Cp shared in two different parent paths, and then exchanging parts from the two parents on the right side of the crossing position Cp. The two-point crossover is performed by randomly choosing two crossing positions Cp1 and Cp2 shared in the two parent paths, and then exchange the parts of the two parents existing between the two crossing positions. The modified crossover consists in randomly choosing two crossing positions Cp1 and Cp2 from two parents P1 and P2, then comparing the three sub-parts of P1 and P2 existing (i.) before Cp1, (ii.) between two crossing position Cp1 and Cp2 and (iii.) after Cp2. The best parts are selected to form the new path.

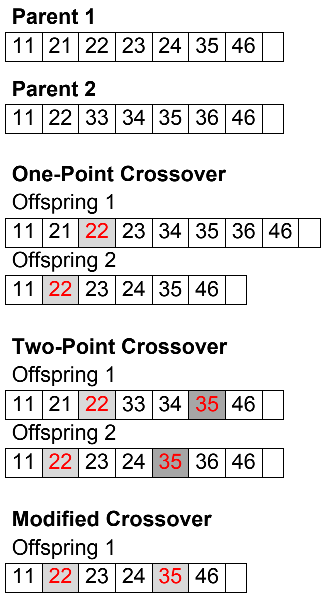


Fig. 2: Crossover operators

As mentioned in the previous section, it is necessary that some conditions hold to guarantee that the crossover operations produce feasible path solutions. Indeed, if not done carefully, the crossover would likely lead to undesirable infeasible path. Let us consider the previous example to illustrate this problem and explain how we solved it. The one-point crossover operation applied to the two feasible path solutions leads to an infeasible path solution:

P1 : (11, 12, 13, 14, 25, 36, 47, 57, 67, 77)

P2 : (11, 22, 33, 44, 55, 66, 77)

Notice that for any selected point both in P1 and P2 the one-point crossover operation leads to infeasible path. In our implementation, we established the necessary conditions to ensure that the resulting path is always feasible :

1) The two parents must traverse at least one common cell.
2) The one-point crossover operation must be performed on a common selected cell.
3) The two-point crossover operation must be performed on two common selected cells.

The example of figure 2 illustrates the crossover operation fulfilling these conditions and leading to feasible path solutions.

In all three crossover types, the parents are chosen randomly from the current generation. The resulting offspring will move directly to the next generation.

After completing crossover, individuals are subjected to the mutation operator. It aims to add a new information in a random way to the genetic search process and ultimately helps to avoid getting trapped into a local optimum. It is an operator that introduces diversity in the population. If an individual is selected for mutation, one of its genes will be replaced. The mutation is performed by randomly choosing a cell from the individual and trying to replace it with one of its neighbour cells of the grid map. Then, the resulting offspring path is evaluated: if it is feasible, it will be accepted and the individual will be replaced in the next generation. Otherwise, if after a certain number of trials, the mutation operator fails in generating a feasible path, it will randomly choose two cells C1 and C2 from the individual, and remove all the cells between them, then connect C1 and C2 by using the greedy approach based on Euclidean distance heuristic. To make sure the new path between C1 and C2 is different from the old one, the old neighbour cell of C1 will be discarded.

C. Control Parameters

The GA performance is affected by a number of parameters: number of generations, population size, and the crossover and mutation probabilities. Usually, the number of generations determines the number of iterations after which the algorithm should stop. The population size determines how many individuals in each population. Increasing the population size will increase the diversity of population and reduce the probability of converging to a local optimum, but this will also increase the execution time of the algorithm. The probabilities (or rates) of crossover and mutation operators determine how often they are performed. After choosing the paths for the operator (i.e. crossover, mutation), the algorithm performs the operator only if a uniformly distributed randomly generated number in the range 0 to 1 is less than the operator probability (i.e. crossover probability, mutation probability). Otherwise, the paths will move to the next generation without any change. A high crossover probability leads to generate new individuals faster, but also increases the disruption of good solutions. On the other hand, a high mutation probability increases the diversity of population, but tends to transform the GA into a random search. Typically, the crossover probability should range from 0.6 to 0.9, and the mutation probability should range from 0.001 to 0.01 [16]. In the next section, we will investigate the best settings for the operators probabilities for our GA path planner.

IV. Performance Evaluation

A. Simulation Model

In this section, we present a simulation model that we used to evaluate the performance of our GA path planner in large-scale grid environments. We designed an object-oriented simulation model and implemented it using C++ under Linux OS on a PC with an Intel Core 2 Duo Processor P8700 (2.53 GHz) and 4GB of RAM. The main classes include the map class, that represents a grid environment as a two dimensional matrix, the path class, that represents the path as a vector structure for the sequence of cells forming the path, the GA class that implements all GA operators and the path planner algorithm. We tested the algorithm on four different-size maps with different complexities (i.e. obstacle ratio) and randomly chosen start and goal positions. The grid maps' sizes are (100*100), (500*500), (1000*1000), and (2000*2000) cells. To impose different environment complexities, we varied the obstacle ratio from 0.1 to 0.4 for every map. We also used different random configurations of the rectangular-shaped obstacle size.

B. Simulation Results

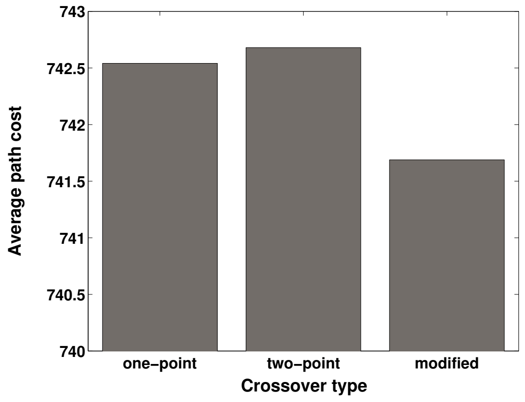
In this section, we present the simulation results of our proposed GA approach for solving the global path planning problem in large-scale grid maps. First, we investigate the impact of crossover operators and the GA parameters on the performance of the planner. Our goal is to identify the most appropriate crossover type and parameters' settings that produce the best results. Subsequently, we evaluate the performance of our GA path planner in large scale environments and we compare it against the A* algorithm with Euclidean distance heuristic, that we also implemented in our C++ simulator. We opted for the A* algorithm as a reference because it is widely used in solving the path planning problems due to its optimality and completeness [17].

For the evaluation, two performance metrics were assessed: (1) the path length: it represents the length (or cost) of the shortest path found by GA, (2) the number of generation: it represents the number of repetitions that GA takes to converge to the shortest path.

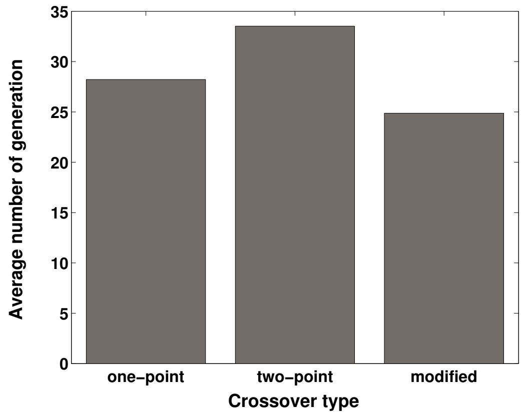
1) Impact of GA parameters: This section aims to study the impact of the GA parameters including mutation probability, crossover probability and the crossover operation on the performance of path planner.

To study the impact of the parameters, we considered maps with size 500*500 cells, although other map sizes can be considered. We generate different 500*500-cell maps with different obstacle ratios for imposing diversity and various complexities in the search problem. We fix the start position to the first cell (leftmost and topmost cell) and the goal position to the last cell in the grid (rightmost and bottommost cell). The paths between these positions will depend on the obstacles' layout of each map. We evaluate the best path generated under the aforementioned settings for different crossover type, crossover probabilities, and mutation probabilities. In all simulations, when the best path found by A* appears in the initial population of GA, it will be discarded to see if the GA can converge to it.

Impact of crossover type: Fig. 3 shows the impact of the crossover type on the average cost of the best paths, and average number of generations to find the best paths. The generation number was set to 100, the population size to 200, the mutation probability to 0.1 and the crossover probability to 1. Fig. 3 shows that the path quality or cost does not really affected by the crossover type. In the most cases, GA can find the same best path with all types. The major effect of the crossover type is on GA convergence time. GA using the modified crossover can find its best paths faster than two other types. This does not mean that the modified crossover always faster, but it is faster in most cases.



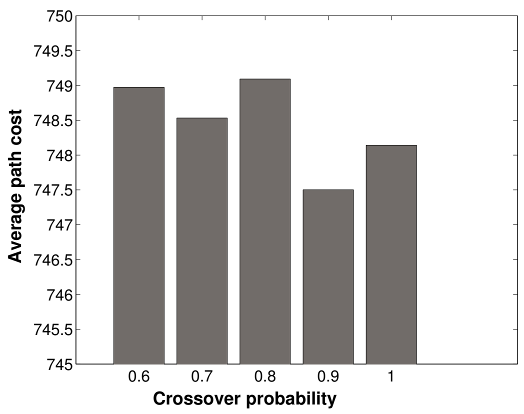
(a) Impact on the path cost



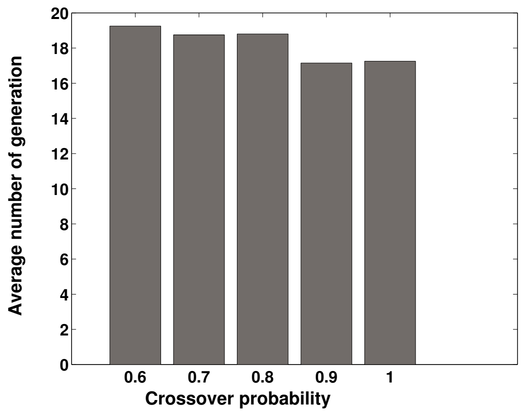
(b) Impact on the number of generations

Fig. 3: Impact of crossover types

Impact of crossover probability: Fig. 4 shows the impact of the crossover probability (with one-point crossover type) on the average cost of the best paths, and average number of generations to find the best paths. The generation number was set to 50 and mutation probability to 0.1. The crossover probability also does not effect the cost of the best paths, and mainly effect the GA convergence speed. Increasing the crossover probability will make GA converge faster.



(a) Impact on the path cost

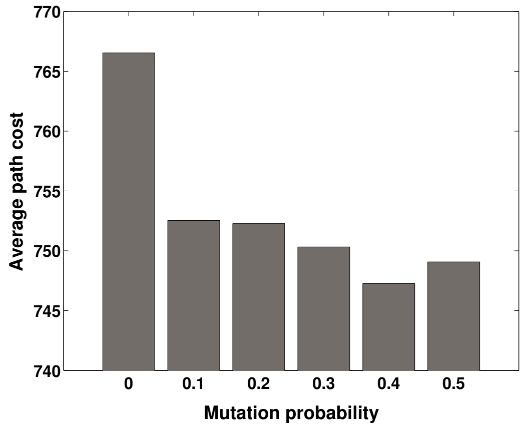


(b) Impact on the number of generations

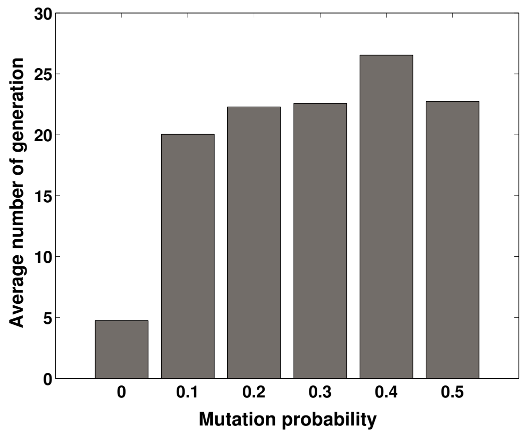
Fig. 4: Impact of crossover probability

Impact of mutation probability: Fig. 5 shows the impact of varying the mutation probability on the average cost of the best paths, and average number of generations to find the best paths. The generation number was set to 50 and crossover probability to 1. Fig. 5 shows that without using the mutation operator, (i.e. mutation probability equals zero), GA converges faster to non-optimal solution. In fact, without the mutation operator, GA could not improve the initial paths in most cases. As the mutation probability increases, the convergence speed will decrease and the quality of the solutions enhanced.

2) Performance Evaluation of GA planner: In this paragraph, we evaluate the performance of the GA planner in terms of solution quality (i.e. path length) and we compare it against A*. Based on the results of the previous section, we choose the GA parameters which produced the best average values: population size = 200, crossover probability = 1, mutation probability = 0.15, and number of iterations = 150 (for 100*100), 200 (for all other environments). However, as GA is an incomplete method, it is not guaranteed that all of the paths are optimal, higher population sizes and number of



(a) Impact on the path cost



(b) Impact on the number of generations

Fig. 5: Impact of mutation probability

generations may be essential for GA to achieve better results, but at expenses of greater simulation times.

Tables I, II, III, and IV present the results of GA and A* for 100*100, 500*500, 1000*1000, and 2000*2000 environments respectively. As seen from table I, the GA is capable of producing optimal or near optimal paths. GA with the modified crossover type produces exactly the same best paths as A*. The first map in table I is the least complex map, and GA can always find an optimal solution quickly in this map particularly with any set of parameters and with all crossover types.

As the obstacles become smaller with higher ratios, the capability of GA gets reduced, and it still can find optimal or near optimal solution, but after more iterations and requires larger population size. Although for some maps, GA could not improve initial paths as in last map in table II, it could produce the optimal paths in less than 10 iterations for all 2000*2000 environments. Also, GA achieves good results for 1000*1000 environments but with more iterations. This demonstrates that the GA, with appropriate design of operators and appropriate parameters' settings, could produce optimal or near optimal solutions in very large environments.

V. Conclusion

In this paper, we studied the efficiency of the GA approach in the context of robot global path planning, which represents a fundamental problem in robotics. The major contributions of this paper are the definition of a suitable environment modeling of the path planning problem for applying GA algorithms and the consideration of large-scale grid environments as the previous works limited their studies to small-size environments.

We designed a GA path planner to solve the aforementioned problem, we considered three differents crossover operators: one-point crossover, two-point crossover and a modified crossover proposed in [15], in order to choose the suitable one which ameliorates the solution quality and accelerates the search procedure.

The GA path planner was extensively evaluated through simulation: we studied the impact of the crossover operator, the impact of the different GA parameters including the mutation and crossover probabilities. We also compared its performance against the A* algorithm. It has been shown that the algorithm based on the GA approach is able to generate the optimal solution generated by A* in most cases. We were not interested in comparing the execution time of GA against that of A* because the two methods are of different nature. Indeed, A* is a constructive method that progressively build the solution, whereas GA performs local search to improve existing solution, and its execution time depends on the number of generations and not related to the instant of finding the best path.

So, in this work we showed that GA can produce optimal solution in solving the path planning problem in large scale grid environments when used as a post-optimization process of constructive metaheuristic methods such as Ant Colony Optimization (ACO) (see [15]) or Particle Swarm optimization (PSO).

Currently, we are working towards designing hybrid path planners based on constructive metaheuristic methods (such as ACO) and local search methods (such as GA and Tabu Search [17]) to optimize both solution quality and execution time.

Acknowledgment

This work is supported by the iroboapp project "Design and Analysis of Intelligent Algorithms for Robotic Problems and Applications"[18] under the grant of the National Plan for Sciences, Technology and Innovation (NPSTI), managed by the Science and Technology Unit of Al-Imam Mohamed bin Saud University and by King AbdulAziz Center for Science and Technology (KACST).

This work is partially supported by Prince Sultan University.

References

[1] K. Tang, K. Man, S. Kwong, and Q. He, "Genetic algorithms and their applications," *Signal Processing Magazine, IEEE*, vol. 13, no. 6, pp. 22–37, 1996.

[2] A. C. Nearchou, "Path planning of a mobile robot using genetic heuristics," *Robotica*, vol. 16, pp. 575–588, 9 1998.

[3] I. AL-Taharwa, A. Sheta, and M. Al-Weshah, "A mobile robot path planning using genetic algorithm in static environment," *Journal of Computer Science*, pp. 341–344, 2008.

TABLE I: Results of GA and A* for 100*100 environments (1:One-point crossover, 2: Two-point crossover, 3: Modified crossover)

Obstacles Ratio	**Obstacles Size**	**A* best path cost**	**Best initial path cost**	**GA best path cost**			**GA best path generation**		
				1*	2*	3*	1*	2*	3*
0.209	10	62.6	65	62.6	62.6	62.6	2	1	1
0.2221	2	76.6	86.2	76.6	76.6	76.6	27	32	6
0.1977	5	77	77.8	77	77	77	8	5	2
0.3235	2	64.2	65.6	65	65	64.2	5	3	22
0.3544	10	54.4	116.4	54.4	54.4	54.4	9	16	9

TABLE II: Results of GA and A* for 500*500 environments

Obstacles Ratio	**Obstacles Size**	**A* best path cost**	**Best initial path cost**	**GA best path cost**			**GA best path generation**		
				1*	2*	3*	1*	2*	3*
0.10092	5	347	359.4	351	351	347	75	69	30
0.19635	5	312.8	321.4	316	316	318.8	86	185	197
0.18672	20	376.2	389.8	379.4	379.4	379.4	26	17	7
0.18065	50	317	329	317	317	317	13	9	5
0.28726	5	318.8	339.2	319.4	318.8	318.8	200	147	49
0.27155	20	388.999	400.6	399.8	399.8	399.8	32	7	3
0.26104	50	153.8	170	153.8	153.8	153.8	1	1	1
0.37161	5	30	40	30	30	30	4	7	11
0.3555	20	389	449	390.4	389	389.799	198	69	9
0.34302	50	460.199	485.999	485.999	485.999	485.999	1	1	1

TABLE III: Results of GA and A* for 1000*1000 environments

Obstacles Ratio	**Obstacles Size**	**A* best path cost**	**Best initial path cost**	**GA best path cost**			**GA best path generation**		
				1*	2*	3*	1*	2*	3*
0.19223	20	233.4	245.4	233.4	233.4	233.4	2	2	2
0.18584	50	204	210	204	204	204	2	2	2
0.27745	20	233.4	245.4	233.4	233.4	233.4	4	4	4
0.27774	50	394.2	400.6	394.2	394.2	394.2	138	60	71
0.35961	20	233.4	245.4	233.4	233.4	233.4	3	2	3

TABLE IV: Results of GA and A* for 2000*2000 environments

Obstacles Ratio	**Obstacles Size**	**A* best path cost**	**Best initial path cost**	**GA best path cost**			**GA best path generation**		
				1*	2*	3*	1*	2*	3*
0.192491	50	438.599	454	438.599	438.599	438.599	4	2	2
0.192772	100	461	481.6	461	461	461	6	5	6

[4] J. Gyorfi, D. Gamota, S. Mok, J. Szczech, M. Toloo, and J. Zhang, "Evolutionary path planning with subpath constraints," *Electronics Packaging Manufacturing, IEEE Transactions on*, vol. 33, no. 2, pp. 143–151, 2010.

[5] N. A. Shiltagh and L. D. Jalal, "Path planning of intelligent mobile robot using modified genetic algorithm," in *International Journal of Soft Computing and Engineering (IJSCE)*, vol. 3, 2013, pp. 31–36.

[6] H. Qu, K. Xing, and T. Alexander, "An improved genetic algorithm with co-evolutionary strategy for global path planning of multiple mobile robots," *Neurocomputing*, vol. 120, no. 0, pp. 509 – 517, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231213005195

[7] Z. Yongnian, Z. Lifang, and L. Yongping, "An improved genetic algorithm for mobile robotic path planning," in *Control and Decision Conference (CCDC), 2012 24th Chinese*, 2012, pp. 3255–3260.

[8] W. Jianguo, D. Biao, M. Guijuan, B. Jianwu, and Y. Xuedong, "Path planning of mobile robot based on improving genetic algorithm," in *Proceedings of the 2011 International Conference on Informatics, Cybernetics, and Computer Engineering (ICCE2011) November 1920, 2011, Melbourne, Australia*, ser. Advances in Intelligent and Soft Computing, L. Jiang, Ed. Springer Berlin Heidelberg, 2012, vol. 112, pp. 535–542.

[9] J. Zhao, L. Zhu, G. Liu, G. Liu, and Z. Han, "A modified genetic algorithm for global path planning of searching robot in mine disasters," in *Mechatronics and Automation, 2009. ICMA 2009. International Conference on*, 2009, pp. 4936–4940.

[10] Y.-Q. Miao, A. Khamis, F. Karray, and M. Kamel, "A novel approach to path planning for autonomous mobile robots," *International Journal on Control and Intelligent Systems.*, vol. 39, no. 4, pp. 1–27, 2011.

[11] A. Hussein, H. Mostafa, M. Badrel-din, O. Sultan, and A. Khamis, "Metaheuristic optimization approach to mobile robot path planning," in *Engineering and Technology (ICET), 2012 International Conference on*, 2012, pp. 1–6.

[12] F. Ahmed and K. Deb, "Multi-objective optimal path planning using elitist non-dominated sorting genetic algorithms," *Soft Computing*, vol. 17, no. 7, pp. 1283–1299, 2013.

[13] P. dos Santos, J. Alves, and J. Ferreira, "A scalable array for cellular genetic algorithms: Tsp as case study," in *Reconfigurable Computing and FPGAs (ReConFig), 2012 International Conference on*, 2012, pp. 1–6.

[14] L. Nian and Z. Jinhua, "Hybrid genetic algorithm for tsp," in *Computational Intelligence and Security (CIS), 2011 Seventh International Conference on*, 2011, pp. 71–75.

[15] I. Chaari, A. Koubaa, H. Bennaceur, S. Trigui, and K. Al-Shalfan, "smartpath: A hybrid aco-ga algorithm for robot path planning," in *Evolutionary Computation (CEC), 2012 IEEE Congress on*, 2012, pp. 1–8.

[16] M. Srinivas and L. Patnaik, "Genetic algorithms: a survey," *Computer*, vol. 27, no. 6, pp. 17–26, 1994.

[17] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2009.

[18] iroboapp: Design and analysis of intelligent algorithms for robotic problems and applications, http://www.iroboapp.org. [Online]. Available: http://www.iroboapp.org