Title: Bezier Curve Based Path Planning in a Dynamic Field using Modified Genetic Algorithm

Author: Mohamed Elhoseny Alaa Tharwat Aboul Ella Hassanien

PII: S1877-7503(17)30890-6
DOI: http://dx.doi.org/doi:10.1016/j.jocs.2017.08.004
Reference: JOCS 737

To appear in:

Received date: 29-12-2016
Revised date: 23-7-2017
Accepted date: 8-8-2017

Please cite this article as: Mohamed Elhoseny, Alaa Tharwat, Aboul Ella Hassanien, Bezier Curve Based Path Planning in a Dynamic Field using Modified Genetic Algorithm, *<![CDATA[Journal of Computational Science]]>* (2017), http://dx.doi.org/10.1016/j.jocs.2017.08.004

Bezier Curve Based Path Planning in a Dynamic Field using Modified Genetic Algorithm

Mohamed Elhoseny[a,b,f,1,*], Alaa Tharwat[c,d,f], Aboul Ella Hassanien[e,f]

[a] *Faculty of Computers and information, Mansoura University, Mansoura, Dakahlia, Egypt.*
[b] *Department of Computer Science and Engineering, University of North Texas, Denton, Texas, U.S.A.*
[c] *Faculty of Engineering, Suez Canal University, Ismailia, Egypt*
[d] *Faculty of Computer Science and Engineering, Frankfurt University of Applied Sciences, Frankfurt am Main, Germany*
[e] *Faculty of Computers and Information, Cairo University, Egypt*
[f] *Scientific Research Group in Egypt (SRGE), http://www.egyptscience.net*

Abstract

Mobile robots have been used in different applications such as assembly, transportation, and manufacturing. Although, the great work to get the optimum robot's path, traditional path planning algorithms often assume that the environment is perfectly known and try to search for the optimal path that contains sharp turns and some polygonal lines. This paper proposes an efficient, Bezier curve based approach for the path planning in a dynamic field using a Modified Genetic Algorithm (MGA). The proposed MGA aims to boost the diversity of the generated solutions of the standard GA which increases the exploration capabilities of the MGA. In our proposed method, the robot's path is dynamically decided based on the obstacles' locations. With the goal of optimizing the distance between the start point and the target point, the MGA is employed to search for the most suitable points as the control points of the Bezier curve. Using the chosen control points, the optimum smooth path that minimizes the total distance between the start and the end points is selected. Our model was

*Corresponding author
Email addresses: mohamed_elhoseny@mans.edu.eg (Mohamed Elhoseny), engalaatharwat@hotmail.com (Alaa Tharwat), aboitcairo@gmail.com (Aboul Ella Hassanien)
[1] My present address is Faculty of Computers and information, Mansoura University, Mansoura, Dakahlia, Egypt.

tested on different environments with different scales, different numbers of obstacles, and six benchmark maps. As a result, the proposed method provides an efficient way to avoid robot's energy consumption in harsh environments.

Keywords: Path Planning, Genetic Algorithm, Bezier Curve,

1. Introduction

Mobile robots have been used in different applications such as assembly, transportation, and manufacturing [Li et al., 2015, Robinson et al., 2015]. The path planning is an important problem in mobile robotics. In this problem, the goal is to find the *optimal* path from the starting to the ending/target position. This optimal or feasible path is used to move the mobile robot along it, avoid collisions with obstacles, and satisfy certain optimization constraints. Hence, the path planning can be formulated as an optimization problem on a set of indices, e.g. shortest distance, and under some constraints, e.g. collision-free path [Cimurs et al., 2017, Li and Zhang, 2016, Mohammed and Hou, 2016].

The path planning is an NP-hard optimization problem that can be solved using heuristic algorithms such as evolutionary algorithms [Manikas et al., 2007]. Genetic Algorithm (GA) is one of the well-known optimization algorithms, and it has been used frequently in different applications especially in mobile robotics [Hu and Yang, 2004, Ismail et al., 2008]. For example, Hu and Yang optimized the path planning of mobile robots [Hu and Yang, 2004]. They incorporated the domain knowledge into its specialized operators. Tuncer and Yildirim proposed a new algorithm by presenting a new mutation operator, and they applied their new algorithm to find the optimal path for mobile robots in a dynamic environments [Tuncer and Yildirim, 2012]. In another research, A vibrational GA was proposed to reduce the possibility of premature convergence and hence help the candidate/solution to find the global optimum solution [Pehlivanoglu, 2012]. Tsai *et al.* presented a parallel elite GA and the migration operator with the aim to: (1) maintain better population diversity, (2) keep parallelism and hence reduce the computational time in comparison with the classical GA, and

(3) avoid premature convergence [Tsai et al., 2011].

Traditional path planning algorithms often assume that the environment is perfectly known and try to search for the optimal path that contains sharp turns and some polygonal lines. Such algorithms, however, are inflexible and hence a mobile robot may need to switch between different modes such as stop, rotate, and restart to move along the polygonal lines, which is time and energy consuming process, and the smooth movement is a requirement for some tasks [Zhou et al., 2011]. Thus, besides the distance, some optimization constraints such as time consumption, energy saving, robot speed, and path smoothness should be included [Mahjoubi et al., 2006, Wahab and Zulkifly, 2017, Wang et al., 2007, Yuan et al., 2007].

Bezier curve has been applied in the smooth path planning problem [Arana-Daniel et al., 2014, Song et al., 2010]. GA-based was used with the Bezier curve to find the optimal paths, and this method outperformed the Gauss-Newton and Piegl's algorithms [Minghua and Guozhao, 2005]. Jolly *et al.* proposed a Bezier-curve-based approach for path planning in a multi-agent robot soccer system, and the velocity of the proposed approach was varying within its allowable limits by keeping its acceleration within the predefined safe limits [Jolly et al., 2009]. Ho and Liu presented a collision-free curvature-bounded smooth path planning approach [Ho and Liu, 2009]. The main idea of their proposed approach was dividing the nodes on the piecewise linear path into control points to generate a collision-free Bezier curve. A new approach was developed by Skrjanc and Klancar for multiple and non-holonomic robots using Bernstein-Bezier curves [Škrjanc and Klančar, 2010]. This approach was used to drive the mobile robots on the obtained reference paths. In [Zhao et al., 2013], the GA was applied to optimize the parameters of Bernstein basis function. The fitness function was the reverse of the sum of the squared error for Bezier curve to fit the given data points on 2D space. A new smooth path planning for a mobile robot by resorting to the GA and the Bezier curve was proposed [Song et al., 2016]. Moreover, a new grid-based environment has been presented, which makes it convenient to perform operations in the GA. Bezier curve was employed in dif-

ferent applications. For example, K. C. Giannakoglou utilized the Bezier curve with GA to design 2-D turbine blades using target pressure considerations, and their proposed model was applied for in a steam turbine blade design problem [Giannakoglou, 1998]. Choi *et al.* used the Bezier curve to generate the optimal trajectories for vehicles to satisfy the path constraints [Choi et al., 2008]. In another research, Chen *et al.* used the Bezier curve for path planning to generate reference trajectories, which are used by the intelligent wheelchair to Pass a doorway [Chen et al., 2012, Wang et al., 2012]. Liang *et al.* used the Bezier curve to find the optimal path for automatic parking systems [Liang et al., 2012]. They used the Fuzzy PID control method to track trajectories that were obtained using the Bezier curve.

In path planning problem, classical methods required a long time and huge storage memory. Therefore, metaheuristic optimization algorithms are employed to solve the path planning problem [Ma et al., 2013]. For example, Particle Swarm Optimization (PSO) was used to optimize the path of a mobile robot through an environment containing static obstacles [Roberge et al., 2013]. Marco. A. Contreras-Cruz *et al.* used Artificial Bee Colony Evolutionary Programming (ABC-EB) algorithm to solve the path planning problem [Contreras-Cruz et al., 2015]. Pigeon Inspired Optimization (PIO) was utilized for solving air robot path planning problem [Duan and Qiao, 2014]. PIO was compared with Differential Evolution (DE) algorithm and it achieved competitive results. The PSO was hybridized with the Gravitational Search Algorithm (GSA) to minimize the maximum path length and hence minimize the arrival time of all robots to their destination in the environment [Das et al., 2016]. Firefly Algorithm (FA) and Bezier curve were used to locate the shortest feasible (collision-free) path, and the results of the proposed algorithm were compared with GA and adaptive inertia weight PSO (PSO-w) [Li et al., 2014]. The results proved that the PSO-w achieved the shortest optimal path and the FA achieved the highest success rates. Galvez *et al.* proposed a new algorithm based on combining Tabu Search (TS) with the Bezier curve [Gálvez et al., 2013]. They conducted different experiments using 2D and 3D curves, and their

proposed algorithm achieved competitive results [Hasegawa et al., 2014]. Generally, GA requires a binary encoding step, this makes GA handles discrete variables, while many other optimization algorithms such as swarms must be modified to handle discrete design variables [Hassan et al., 2005]. Moreover, this encoding step forces the candidates to take values that are within their upper and lower bounds, while in many other algorithms the candidates can violate the side constraints. Further, a comparison between GA and PSO which is one of the well-known swarm algorithms reported that GA achieved results better than PSO [Hassan et al., 2005]. Therefore, in our model, GA was employed to search for the most suitable points as the control points of the Bezier curve.

GA is combined with path smooth planning approach, e.g. Bezier curve, to: (1) obtain smooth path planning for mobile robots, (2) deal with a dynamic environment with parallel implementation, and (3) generate diversity in paths [Hocaoglu and Sanderson, 1996, Xiao et al., 1997]. Bezier curve and GA was also applied to detect collisions in motion planning [Peter and Peter, 2010]. In [Linquan et al., 2008], the Bezier curve was optimized using GA for path planning problem to avoid obstacles. However, the mathematical analysis of the optimal path planning problem has not been discussed. Moreover, the model was applied in a very small environment with only three obstacles; and only two control points were used. In another research, Adaptive GA (AGA) was used for structural topology optimization in [Sid et al., 2005], where the AGA was used to find the optimal control points of Bezier curves and the thickness values for each curve to avoid the formation of disconnected elements and checkerboard patterns in the optimal topology design. The GA and Bezier curve were coupled and employed in electrical engineering. For example, Ziolkowski *et al.* used the GA and Bezier curve to design a shape of a solenoid which produces a uniform magnetic field on its axis [Ziolkowski and Gratkowski, 2016]. In another research, the parallel GA was combined with Bezier curve to generate trajectories for multi-unmanned aerial vehicle (UAV) systems, where the Bezier curve was utilized to increase the smoothness of the obtained path [Sahingoz, 2014]. Bezier+GA algorithm was used to design a cambered airfoil and the

proposed design achieved competitive results compared some state-of-the-art methods [Gardner and Selig, 2003].

GA has been used in many applications [Elhoseny et al., 2015, Metawa et al., 2017, Yuan, 2017]. In image processing, GA was applied in image segmentation [Chun and Yang, 1996], color image quantization [Scheunders, 1997], and edge detection [Gudmundsson et al., 1998]. GA is also employed for solving the mathematical problems. For example, a Multi-Agent GA (MAGA) was used for global numerical optimization [Zhong et al., 2004], and the proposed algorithm was evaluated using ten benchmark function with different dimensions. MAGA achieved results better than two variants of GA. In power applications, GA was employed to solve Power Economic Dispatch (PED) [Chiang, 2005], in reactive power system planning [Lee et al., 1995], and for the solution of the Optimal Power Flow (OPF) with both continuous and discrete control variables [Bakirtzis et al., 2002]. Finally, in machine learning, GA was used to optimize the parameters of different classifiers such as support Vector Machine (SVM) [Tharwat et al., 2017a,b, Wu et al., 2007] and feature selection [Tan et al., 2008]. Simply, GA achieved competitive results for various optimization problems.

Despite the great efforts to optimize the path length as discussed in the related work section, most of the previous GA-based paths planning methods are working in a static environment. However, recent path planning applications require changing the robot path during the running time as a response to some environmental changes, i.e. obstacles movement. Accordingly, searching for an optimum path in a dynamic environment is on-the-fly an open challenge. Thus, the main contribution of this work is to present a Modified GA (MGA) with the aim of improving the exploration capabilities of the standard GA, and this can be achieved by generating diversified solutions. The MGA was employed to get the shortest and the smoothest path using Bezier curve in a dynamic environment, which can be applied in different applications. In robot applications, the robot's path is dynamically decided based on the obstacles' locations. Hence, the proposed method provides an efficient way to avoid energy exhaustion especially in robot applications with dynamic and limited resources

environments.

The rest of the paper is organized as follows: Section 2 gives overviews of the techniques and methods used for the proposed approach. Section 3 explains in details the proposed GA-based method to get the robot path. Section 4 discusses the experimental results compared to some of the state-of-art methods. Finally, Section 5 concludes the paper.

2. Mathematical Model

2.1. Bezier Curve

To be self-content, we briefly review a Bezier Curve definition. Bezier curve has good geometric properties and has been widely used in computer graphics applications. One of the biggest advantages of the Bezier curve is that the Bezier curve is also convex if the control point is a convex polygon, that is, the feature polygon is convex. In addition, it can describe and express free curves and surfaces succinctly and perfectly. Thus, Bezier curve is a good tool for curve fitting, and it can be drawn as a series of line segments joining the points. Given $n+1$ points $p_i, i = 0, 1, \ldots, n$ as follows:

$$R^3 : B : [0,1] \rightarrow R^3 \tag{1}$$

A Bezier Curve of degree n is a parametric curve composed of Bernestein basis polynomials of degree n and it can be defined as:

$$P(t) = \sum_{i=1}^{n} p_i B_{i,n}(t), t \in [0,1] \tag{2}$$

where t indicates the normalized time variable, $P_i(x_i, y_i)^T$ represents the coordinate vector of the i^{th} control point with x_i and y_i being the components corresponding to the X and Y coordinate, respectively, $B_{i,n}$ is the Bernestein basis polynomials, which represents the base function in the expression of Bezier curve, and it is defined as follows:

$$B_{i,n}(t) = C_n^i t^i (1-t)^{n-i} = \frac{n!}{i!(n-i)!} t^i (1-t)^{n-i}, i = 0, 1, \ldots, n. \tag{3}$$

The derivatives of a Bezier curve is determined by the control points, and the first derivative of a Bezier curve in Equation (2) is expressed as in Equation (4). Moreover, higher-order derivatives of a Bezier curve can also be calculated.

$$\dot{P}(t) = \frac{dP(t)}{dt} = n \sum_{i=0}^{n-1} B_{i,n-1}(t)(P_{i+1} - P_i) \tag{4}$$

In the two-dimensional space, the curvature of a Bezier curve with respect to t is expressed as follows:

$$k(t) = \frac{1}{R(t)} = \frac{\dot{P}_x(t)\ddot{P}_y(t) - \dot{P}_y(t)\ddot{P}_x(t)}{(\dot{P}_x^2(t) + \dot{P}_y^2(t))^{1.5}} \tag{5}$$

where $R(t)$ represents the radius of curvature, $\dot{P}_x(t), \dot{P}_y(t), \ddot{P}_x(t)$, and $\ddot{P}_y(t)$ are the components of first and second derivatives of the Bezier curve $P(t)$ for the X and Y coordinates.

In the path planning problem, the Bezier curves are connected to form a smooth path planning for mobile robots, where the second and lower order continuities are determined for the smoothness of the path as follows:

- Zero-order continuity is held by the end and start points of the connected Bezier curves.
- The first-order continuity is ensured by the equivalent tangent at the connection of two curves.
- The second-order continuity is ensured by the equivalent curvatures [Song et al., 2016].

2.2. Genetic Algorithm (GA)

Genetic Algorithm (GA) was developed in the 1960s by Holland [Holland, 1992]. It was inspired by the evolutionist theory explaining the origin of species. In nature, weak species within the environment are susceptible to extinction by natural selection, while the strong ones have a greater opportunity to pass their genes to future generations via reproduction. Hence, the species carrying the correct and strong combination in their genes become dominant in their

population. During the evolution process, random changes may occur in genes. Sometimes, these changes may provide additional advantages for survival and new species evolve from the old ones. On the other hand, unsuccessful changes are eliminated by natural selection [Goldberg, 2006, Haupt, 1995, Konak et al., 2006].

In GA, a solution is denoted by $x \in \Omega$, and it is called a candidate, individual or chromosome, where Ω is the search space. Each chromosome consists of discrete units called genes as follows, $x = [x_1, x_2, \ldots, x_N]$, where x_i is the i^{th} gene in x chromosome and N is the number of genes or the dimension of the search space. In the original GA, genes are assumed to be binary numbers. Each chromosome corresponds to one solution in the search space. Thus, a mapping mechanism, i.e. encoding, is required between the search space and chromosomes; and hence, GA works on the encoding of a problem not the problem itself [Goldberg, 2006].

GA has a collection of chromosomes called population, and this population is randomly initialized. From this population, GA generates different solutions, and it converges to the optimal or near optimal solution. To generate new solutions, two main operators are used, namely, *crossover* and *mutation*. Both methods are explained in the next two sections.

2.2.1. Crossover

The main role of the crossover is to provide mixing of the solutions and convergence in a subspace. In the crossover operator, two chromosomes, i.e. parents, are combined together by exchanging part of one chromosome with a corresponding part of another to generate new chromosomes, i.e. offspring. The new chromosomes, i.e. offspring, are expected to inherit new good genes which make the parents fitter. After applying the crossover operator many times in GA, the genes of good chromosomes are expected to appear frequently in the population, which may lead to good solutions. Hence, the crossover leads the population to converge to the optimal solution by making the chromosomes in the population alike [Konak et al., 2006].

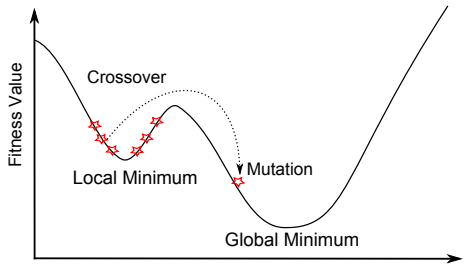

Figure 1: A comparison between crossover and mutation processes.

The crossover point, C, is chosen randomly, and the offspring consists of the pre-C section from the first parent followed by the post-C section of the other parent. Another important point is the crossover probability, P_c, which is usually very high, in the range of $[0.7, 1]$ [Yang, 2014]. If the crossover probability is too small, then the crossover occurs sparsely, which is not efficient for evolution [Reeves, 1995].

Assume we have two parents who are represented in binary as follows, $A = [0011] \in \Omega$ and $B = [0000] \in \Omega$, where $\Omega \in \mathcal{R}^4$ is the search space. The crossover process generates one of the following offspring: 0010, 0011, 0001, and 0000. As shown, the first two digits in both parents are 00; hence, the crossover process will generate a solution in a subspace where the third and fourth digits are different. Mathematically, the subspace which includes the new solutions that are generated by crossover process is defined as follows, $S = [00] \cup \mathcal{R}^2 \subset \Omega$. Thus, the crossover process helps to search locally in a subspace. In other words, crossover operator aims to search locally around the current good solution; this is so-called exploitation as in Figure 1.

2.2.2. Mutation

In mutation operator, random changes into the chromosomes are introduced by making changes in the chromosomes' genes. In GA, the mutation rate is defined as the probability of changing the properties of a gene, and this rate is very small and depends mainly on the length of the chromosome. Hence, the new chromosome that is generated by mutation operator will not be very

different from the original one. Mutation improves the searching capabilities of GA by generating new solutions [Golberg, 1989, Konak et al., 2006]. The mutation probability, P_m, is usually small, in the range of $[0.001, 0.05]$. Increasing the mutation probability makes the solutions jump around even if the optimal solution is approaching.

Mathematically, the mutation generates a new solution that may not be in the subspace; hence, it provides a global exploration. In other words, changing one digit of the current solution $A = [0011]$ jumps out of any previous subspace and generates a new solution. For this reason, mutation process explores the search space to find the optimal or near optimal solutions, this is so-called exploration, and this process assists the search escape from local optima.

2.2.3. Selection

Reproduction involves selection of chromosomes for the next generation, and this selection starting with calculating the cost/fitness for each chromosome. The chromosomes are then ranked from the most-fit to the least-fit, according to their fitness function. Unacceptable or weak chromosomes are discarded from the population, and the rest of chromosomes are selected to become parents to generate new offspring. There are many selection methods such as ranking, proportional selection, and tournament selection methods [Golberg, 1989].

As mentioned in Sections 2.2.1 and 2.2.2, both crossover and mutation processes work without using the knowledge of the fitness function; on the other hand, the selection process does use the fitness function. This elitism mechanism ensures that the best solutions must survive in the population. The parents' chromosomes reproduce enough to offset the discarded chromosomes; and hence, the total number of chromosomes remains constant after each iteration. The GA stops when an acceptable solution is obtained, or after a set number of iterations [Haupt, 1995].

3. Feasible Path Planning using Modified GA (MGA)

3.1. Problem Representation

This paper proposes an efficient, Bezier curve-based approach for the path planning in a dynamic field using a Modified GA (MGA). In our proposed method, the path of the agent is dynamically decided based on the obstacles' locations. In other words, we said dynamic field due to the ability of an obstacle to change its location during the robot moving; and MGA will be run again to update the optimum path of the robot according to the new locations; hence, the field is fully monitored during the robot moving. The goal of our model is to optimize the distance between the start point, s, and the target point, t, MGA is employed to search for the most suitable points as the control points, $\tilde{P}$, of the Bezier curve. Using the chosen control points, the optimum path, β, that minimizes the total distance, $\tilde{D}$, between the start and the end points is selected.

Mathematically, the goal is to search for the suitable Bezier curve's control points, and each cell in the field is represented with a gene that takes 1 when the cell is a control point for the Bezier curve, or 0 when the cell is not used in Bezier curve fitting. In addition, the cells occupied by the obstacles, δ, are set to be -1 and cannot be used as control points. We set a radius r to exclude the nearby cells from serving as the control points. The regular point, i.e. a point that is not an obstacle, is indicated as p.

This paper assumes that the points are generated from a captured image. Each pixel or a neighborhood in the image corresponds to a point p. Hence, the field size and the locations of points and obstacles are provided to the algorithm through that image. An example of the input image is shown in Figure 2.

Mathematically, we can represent our problem as an optimization problem with one objective function and multiple constraints as in Eq. (6). As in Eq. (6), the aim of the objective function is to minimize the total distance $\tilde{D}$.

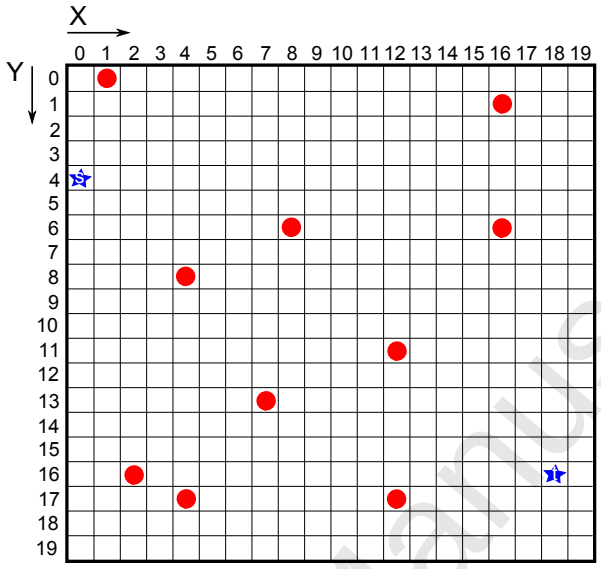

Figure 2: An example of the working field with $(20 \times 20 = 400)$ cells and ten obstacles (red circles). The starting point is located at $(4, 0)$ and the target is located at $(16, 18)$.

$$\begin{aligned} &\text{minimize} \quad \tilde{D} \\ &\text{subject to} \quad D \geq r \\ &\qquad \forall \theta \in \delta \Rightarrow \theta \notin \tilde{P} \\ &\qquad \forall p \in \beta \Rightarrow p \notin \delta \\ &\qquad s, t \notin \delta. \\ &\qquad \tilde{P} \neq \Phi. \end{aligned} \tag{6}$$

where $\tilde{D}$ is the summation of distances d between the adjacent points $(p_1, p_2, \ldots, p_i)$ on the Bezier curve as in Eq. (7), D is the distance between two control points, and δ is a set of obstacles θ.

$$\tilde{D} = \sum_{i=1}^{\eta} d(p_i, p_{i+1}) \tag{7}$$

where η represents the count of the Bezier curves' points and $\tilde{D}$ can be calculated by integral of Bezier curve as follows:

$$\tilde{D} = \oint_l P(t) = \oint \sum_{i=1}^{n} p_i B_{i,n}(t) = \int_0^1 \sqrt{(x_t'^2 + y_t'^2)dt} \tag{8}$$

where l is the distance between a point p and the target t.

3.2. The Proposed Method

Our proposed method consists of several fundamental components. First, a binary chromosome is used to encode the selection of control points within the entire field. Second, each chromosome is evaluated to make sure that it is valid to be a possible solution structure following the predefined constraints. The fitness is then evaluated based on this structure. The optimization goal is to minimize the distance between the start and the end points. The proposed method is presented in Algorithm 1.

3.2.1. Chromosome Encoding and Fitness Function

Each gene in the chromosome represents a pixel in the field. The value of a gene can be either 1 or 0, where 1 indicates that the corresponding pixel serves as the control point and 0 indicates a non-control point pixel. Figure 3 depicts a chromosome for a field with 25 nodes.

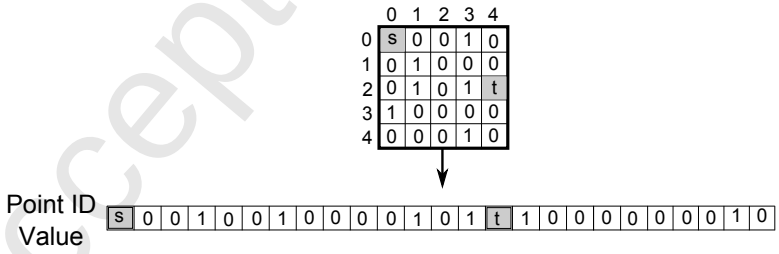

Figure 3: Chromosome representation.

The MGA's fitness function simply consists of length of the path $\tilde{D}$. The main objective is to minimize $\tilde{D}$. The fitness function is defined as follows:

$$f = \frac{1}{\tilde{D}} \tag{9}$$

Algorithm 1 Path Planning based on Bezier Curve using MGA.

Randomly generate a pool of S solution structures $X = \{x_1, x_2, \ldots, x_S\}$.

Represent each structure in S by one chromosome to get a pool P of chromosomes $U = \{u_1, u_2, \ldots, u_P\}$.

$\forall u_i \in U$, Validate u_i.

Generate β.

Evaluate the fitness of each $u_i \in U$ as in Eq. (9).

for $q = 1, 2, \ldots, Q$, where Q is the number of generations **do**

  $\tilde{U} \Leftarrow \emptyset$.

  for $p = 1, 2, \ldots, P$ **do**

    Randomly select $u_a, u_b \in U$ $(a \neq b)$ based on the normalized fitness $\tilde{f}(u)$ as follows:

$$\tilde{f}(u) = \frac{f(u)}{\sum_c f(u)}.$$

    Crossover u_a and u_b according to α as follows:

$$\mathcal{C}(u_a, u_b|\alpha) \Rightarrow u'_a, u'_b.$$

    Perform mutation on u'_a and u'_b as follows:

$$\mathcal{M}(u'_a|\beta) \Rightarrow \tilde{u}_a, \quad \mathcal{M}(u'_b|\beta) \Rightarrow \tilde{u}_b.$$

    Evaluate $f(\tilde{u}_a)$ and $f(\tilde{u}_b)$.

    $\tilde{U} \Leftarrow \tilde{U} \cup \{\tilde{u}_a, \tilde{u}_b\}$.

  end for

  $U \Leftarrow \{u_i; u_i \in \tilde{U} \text{ and } f(u_i)\}$.

end for

Find the chromosome u^* that satisfies

$$u^* = \arg\max_u f(u), u \in U$$

Return the network structure x^* by mapping u^* back.

To construct the path, a Bezier curve that is drawn using the proposed control points, the validation process is used to remove the invalid path, i.e. the path that intersects with an obstacle.

3.2.2. Validation Process

Some points are unable to serve as a control point and some are preferred to take the role due to their location. The aim of the validation process is to ensure either the chromosome is valid to be selected as a solution or not. This can be done by checking the predefined constraints that are required by the MGA to work as shown in Figure 4. When a point becomes invalid, i.e. obstacle, its corresponding gene value is set to -1, which exempts this point from further MGA operations.

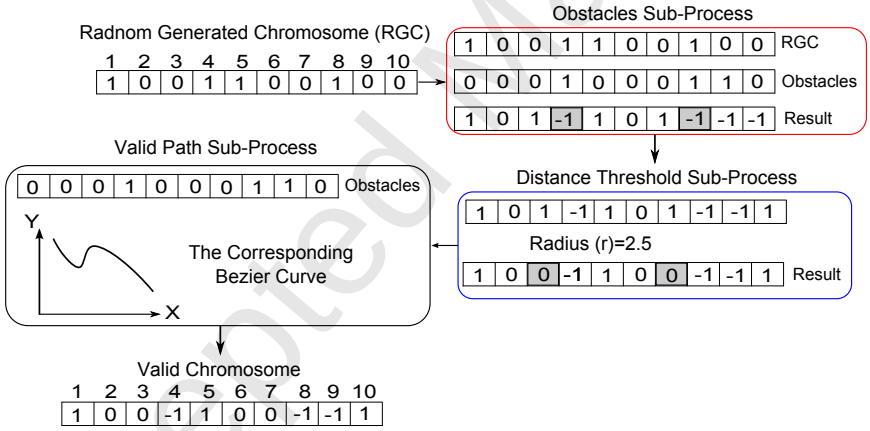

Figure 4: An example of the validation process with ten points, $r = 2.5$, and three obstacles.

Figure 4 illustrates the validation process that leverages the field properties. In the process of MGA optimization, a new chromosome represents the proposed structure for the path. Each gene in the chromosome defines the expected role of the corresponding point, i.e. whether it serves as a control point or not. The process consults *Obstacles*, *Distance Threshold*, and *Valid Path* subprocesses. The role of *Obstacles* subprocess is to determine whether the point can serve as a control point (one represents serving as control point; otherwise, regular point). While the *Distance Threshold* subprocess is used to test whether that

distance between any two control points in the proposed structure is longer than the radius r or not. Finally, *Valid Path* subprocess checks if the path intersects with an obstacle. The validation process deletes that chromosome and randomly generates a new one. The deleted chromosome is then replaced by the new one and the new generation starts.

MGA generates new chromosomes through crossover and mutation operations and evaluates their fitness. The crossover operation is performed with two randomly selected chromosomes determined by a crossover probability to regulate the operation. When the crossover is excluded, the parent chromosomes are duplicated to the offspring without change. Varying the crossover probability alters the evolution speed of the search process.

As mentioned in Section 2.2, the mutation operation involves altering the value at a randomly selected gene within the chromosome. Hence, mutation operation could create completely new species, i.e. an arbitrary locus in the fitness landscape. Hence, it is a means to get out of a local optimum. Recall that when a point becomes an obstacle, its corresponding gene is set to -1 to exempt it from mutation operations.

3.3. Modified GA (MGA)

One of the problems of optimization algorithms is that the solutions are moved in a random walk, which causes a loss of diversity in the resulting solutions in each iteration. In this work, the aim is to boost the diversity of the generated solutions in the GA. This can be achieved by measuring the diversity between different solutions, i.e., the diversity of the chromosomes, in each iteration and the solutions are accepted only if their diversity measure is lower than threshold η. This threshold is initialized with a positive value in $(0, 1)$. The value of η is linearly decreased from the initial value to 0 over the course of iterations. In the proposed model, we used a decay factor $\varrho = 0.99$ which determines the rate of decay; in other words, in each iteration, the value η will be updated as follows, $\eta_{t+1} = \varrho\eta_t$; hence, $\eta \to 0$, where $t \to \infty$. Thus, the new modification helps the GA for searching/exploring for different solutions at the beginning by

accepting only different solutions and reject identical solutions. This may help the MGA to escape from local optima problem. In each iteration, the value η will be decreased, and this means that more similar solutions are acceptable for the next generations. This makes a balance between the exploration, at the beginning, and the exploitation, at the end of our model.

There are several measures to calculate the diversity [Kuncheva, 2004], and the Q-statistic is a well-known measure and it has been used in classifier ensembles. In our proposed model, the Q-statistic method is used to measure the diversity between different solutions. Mathematically, the Q-statistic between two solutions, S_1 and S_2, is defined as follows:

$$Q_{i,j} = \frac{N^{11}N^{00} - N^{01}N^{10}}{N^{11}N^{00} + N^{01}N^{10}} \tag{10}$$

where:

- N^{11} is the number of elmenets, i.e. genes, with value a in S_1 and with value b in S_2, where $a = b = 1$.
- N^{00} is the number of elmenets, i.e. genes, with value a in S_1 and with value b in S_2, where $a = b = 0$.
- N^{10} is the number of elmenets, i.e. genes, with value a in S_1 and with value b in S_2, where $a = 1$ and $b = 0$.
- N^{01} is the number of elmenets, i.e. genes, with value a in S_1 and with value b in S_2, where $a = 0$ and $b = 1$.
- The value of Q varies between -1 and 1.

3.3.1. Illustrative Example

Assume we have three solutions, $S_i,\ i = 1, 2, 3$, and the length of all solutions is 10. The values of all solutions are summarized in Table 1. As indicated in Equation 10, to calculate the diversity between S_1 and S_2, the values of N^{11}, N^{00}, N^{10}, and N^{01}, are 3, 2, 3, and 2, respectively. Hence, the diversity between S_1 and S_2, $Q_{12} = \frac{3\times2-3\times2}{3\times2+3\times2} = 0$; hence, these two solutions are independent.

Algorithm 2 Modified Genetic Algorithm (MGA)

Randomly generate M random solutions to form the first population P_1.
Initialize MGA parameters and the threshold parameter η.
Encode the solutions into chromosomes (strings).
while $t = 1 < Max_{iter}$, where Max_{iter} is the maximum iteration **do**
  Evaluate the fitness values for all solutions in P_t.
  Generate new population C_t using crossover process and add them to P_t.
  Mutate each solution $x \in C_t$ with a predefined mutation rate and add them to P_t.
  Calculate the diversity, Q, of all solutions.
  while $(Q > \eta)$ **do**
    Generate new population C_t using crossover process and add them to P_t.
    Mutate each solution $x \in C_t$ with a predefined mutation rate and add them to P_t.
    Calculate the diversity, Q, of all solutions.
  end while
  Evaluate all generated solutions.
  Select the current best M chromosomes for the next generation (elitism).
  if the stopping criterion is satisfied **then**
    Terminate the search and return the current population P_t.
  else
    $t = t + 1$.
    Decrease η as follows, $\eta_{t+1} = \varrho\eta_t$.
  end if
end while

Similarly, the values of Q_{13} and Q_{23}, are 0 and $-\frac{1}{3}$, respectively. The average diversity is $(Q_{12} + Q_{13} + Q_{23})/3 \approx -0.11$.

Table 1: Values of all solutions in our example.

S1	1	1	0	0	1	1	0	0	1	0
S2	1	0	1	0	1	0	1	0	1	1
S3	1	1	1	1	0	0	0	0	0	0

4. Experimental Results and Discussion

4.1. Experimental Settings

To evaluate our proposed method, we created two scenarios by generating random obstacles placements in the field with different start and end points. For each scenario, we repeated the experiments for ten times and reported the average results. Comparison studies were conducted with some recent state-of-the-art methods reported in the literature (Probabilistic Road Map (PRM) [Masehian and Sedighizadeh, 2010], ABC-EB [Contreras-Cruz et al., 2015], and GA [Alajlan et al., 2013]). These two scenarios were created to study the routine method performance in small and large environments. In addition, a third scenario was conducted to evaluate the proposed method using a set of well-known benchmark maps. These maps were collected from repository motion planning maps of the Intelligent and Mobile Robotics Group from the Department of Cybernetics, Czech Technical University in Prague[2]. More details about each scenario are in the following sections.

4.2. Parameter setting for MGA

Parameters tuning for any optimization algorithm is an important step as designing the algorithm itself. In this section, the effect of the population size and the number of iterations on the computational time and the results of the

[2]Maps are available at http://imr.felk.cvut.czplanningmaps.xml.

proposed model were investigated. In this experiment, the number of obstacles was five, and the dimension of the environment was 100×100. Moreover, in MGA, the crossover and mutation ratios were 0.8 and 0.006, respectively.

4.2.1. Population size

The number of solutions or population size needs to be sufficient for exploring the search space. In this section, the effect of the population size on the results and computational time of the proposed model was investigated when the number of solutions was 60, 80, or 100 solutions, and the number of iterations was 100. Figure 5(a) and 5(b) show the results and computational time for this experiment, respectively. From the figure, it is observed that increasing the population size improved the results and converged faster to the optimal or near optimal solution(s), but, more computational time was needed. Moreover, due to the randomness of MGA, which is similar to the random walk in many swarms, there is a small fluctuation in the path length and computational time. In other words, MGA generates its initial solutions randomly; hence, different results can be obtained in each run.

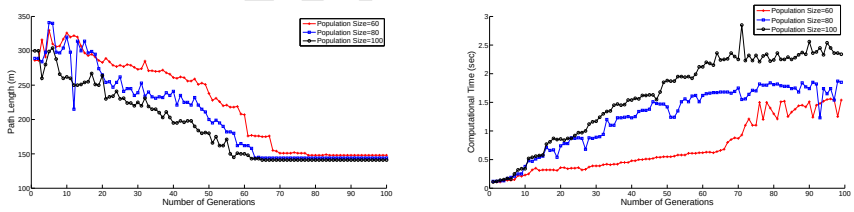

Figure 5: Effect of the population size on the performance of the proposed model (a) path length of the proposed model; (b) computational time of the proposed model.

4.2.2. Number of iterations

The number of iterations also affects the performance of the proposed model and the computational time. In this section, the effect of this parameter, i.e. number of iterations, on the performance and computational time of the proposed model was tested. In this experiment, the number of iterations was ranged

from 10 to 100. The results of this experiment are summarized in Figure 6. From the figure, it can be noticed that, when the number iterations was increased, the performance was improved until it reached an extent at which increasing the number of iterations did not affect the results, and in our results, this extent was 80. Moreover, the computational time increased when the number of iterations was increased.

On the basis of the above parameter analysis and research results, the number of iteration and the population size of the MGA that were used in our proposed model were 80 and 60, respectively. Moreover, the value of η was 0.8 and the decay rate $\varrho = 0.99$.

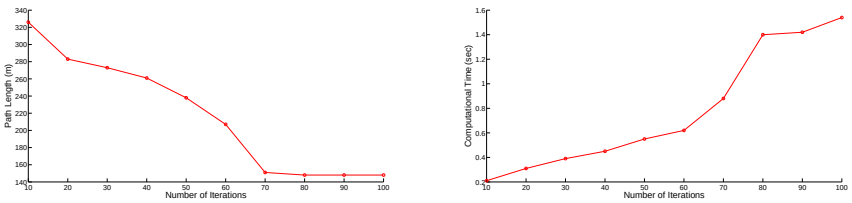

Figure 6: Effect of the number of iterations on the performance of the proposed model. (a) path length of the proposed model with different numbers of iterations; (b) computational time of the proposed model using different numbers of iterations.

4.3. First Scenario

In this scenario, the obstacles are randomly placed within a 100×100 environment. Two cases are designed: 1) the start and target points are placed in two corners of the field, i.e. at $(0, 0)$ and $(100, 100)$, respectively, (see Figure 7(a)) and 2) the start and target points are placed in two different corners of the field, i.e. at $(100, 0)$ and $(0, 100)$, respectively, (see Figure 7(b)). Figure 7 shows an illustrative example of each case. As shown, the obstacles are depicted with red circle, and the start and the target points are marked with blue stars. The results of this scenario are summarized in Table 2.

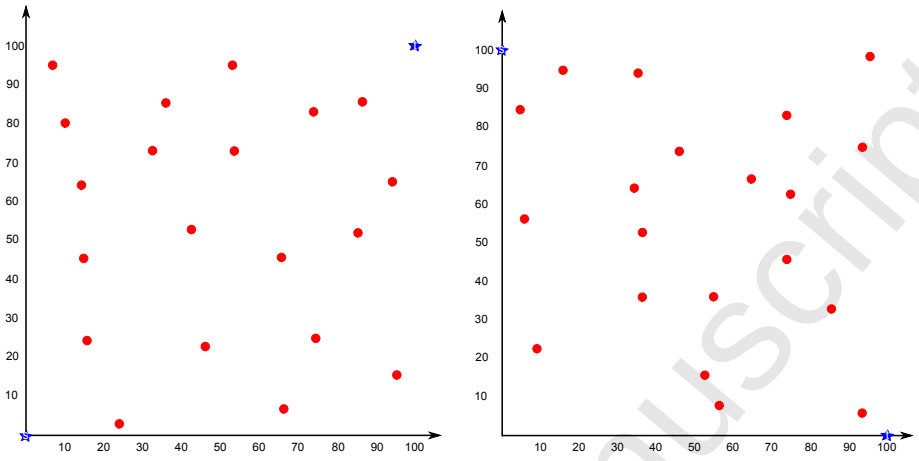

Figure 7: Start point, target point, and obstacles placement in the first scenario; (a) Case 1, (b) Case 2.

Table 2: Path length with different methods in the first scenario. Each column gives the path length when respective number of obstacles is used.

	Case 1					Case 2				
Methods	5	10	15	20	25	5	10	15	20	25
PRM [Masehian and Sedighizadeh, 2010]	198	218	227	250	256	189	223	233	246	261
ABC-EP [Contreras-Cruz et al., 2015]	186	204	216	232	244	191	213	222	243	279
GA [Alajlan et al., 2013]	174	187	202	220	231	177	193	214	220	247
Proposed Method	158	167	182	195	200	159	168	183	195	211

4.4. Second Scenario

In this scenario, the obstacles are randomly placed within a 200×200 field. Similarly, we have two cases designed: 1) the start and target points are placed at two corners of the field (see Figure 8(a)), at $(0, 0)$ and $(200, 200)$, respectively, and 2) the start and target points are placed in two different corners of the field (see Figure 8(b)), i.e. at $(200, 0)$ and $(0, 200)$, respectively. As shown in Figure 8, the obstacles are depicted with red circle, and the start and target points are marked with blue stars. The results of this scenario are summarized in Table 3.

Note that although the start and the end points were placed in a relatively

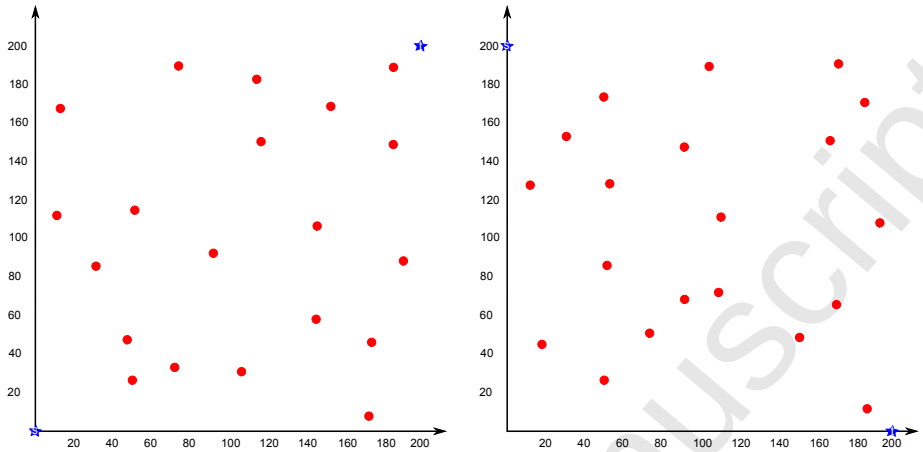

Figure 8: Start point, target point, and obstacles placement in the second scenario; (a) Case 1, (b) Case 2.

Table 3: Path length with different methods in scenario 2. Each column gives the path length when respective number of obstacles is used.

	Case 1					Case 2				
Methods	5	10	15	20	25	5	10	15	20	25
PRM [Masehian and Sedighizadeh, 2010]	401	432	493	521	545	421	469	487	524	541
ABC-EP [Contreras-Cruz et al., 2015]	392	412	441	482	510	400	433	472	498	517
GA [Alajlan et al., 2013]	384	401	421	439	455	380	399	427	463	484
Proposed Method	314	349	368	382	401	315	338	354	380	399

similar location within the field, these two scenarios were different. As the count of the obstacles were the same (20 obstacles), the size of the field and the placement of start and the end points affect the selected control points of Bezier curve. Moreover, the obstacles were placed randomly, and the two scenarios were conducted using different numbers of obstacles.

4.5. Third Scenario

In this experiment, six different planar environments were used to evaluate the proposed method. Figure 9 shows the set of benchmark maps that are used in our experiments. Table 4 lists the details (ID, name, and size) of each map

to clarify their characteristics. The ID of the map is used to identify the map in our discussion uniquely.

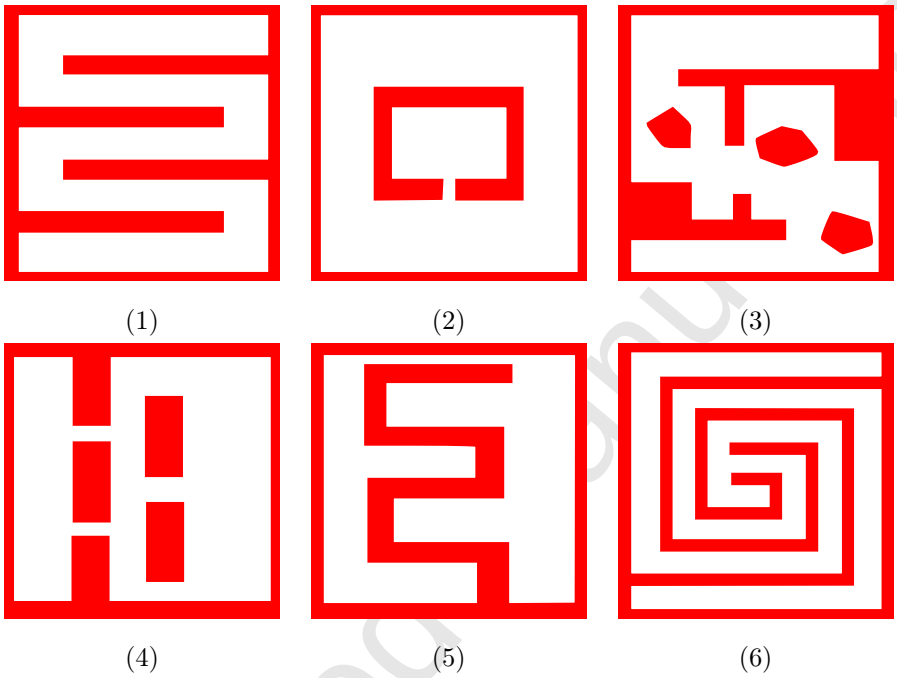

Figure 9: Set of benchmark maps that were used in our experiment.

Table 4: Benchmark maps that are used in our experiments.

ID	Name	Size ($[m \times n]$)
1	back and forth	10.0 × 10.0
2	bugtrap1	36.4 × 28.8
3	complex2	20.0 × 20.0
4	gaps	20.0 × 20.0
5	hidden_U	20.0 × 30.0
6	square_spiral	20.0 × 20.0

4.6. Results and Discussion

Tables 2 and 3 present the length of the path using different methods. This length is the average of ten experiments with random obstacles placement. The path length is calculated based on the count of points on the curve. The length is incremented by 1.5 if the robot moves diagonally. Otherwise, its value is incremented by 1. As shown, in the first scenario (see Table 2), in both cases, the proposed method yielded the shortest length, and the GA method achieved the second best results, while the PRM method achieved the worst results. Table 3 summarizes the results of the second scenario. As shown, the results of the proposed method yielded the shortest path length and also the worst results are achieved by the PRM method. In both scenarios, the path length proportional with the number of obstacles.

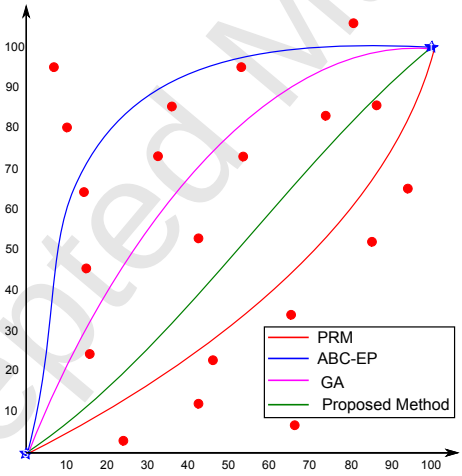

Figure 10: Simulated example for robot path in the first scenario (small field) based on the results in Table 2.

Figure 10 depicts the change in choosing the optimum path in each method. It is evident that the improvement of our proposed method is significant. It is also interesting to examine the running time to get the optimum path. Table 5 lists the average time and standard deviation of our model using different numbers of obstacles. Moreover, Table 6 lists the average running time for all

methods using different numbers of obstacles. The time was reported at the last generation. Further, the average time used by the proposed model is compared with the running time of the other methods as shown in Figure 11, and the computational time of the proposed model was much lower than the PRM and ABC-EP models.

Kolmogorov complexity analysis seems uniquely suited for application to genetic algorithm implementation as our model. It was created, among other reasons, to provide a way to evaluate objects statically. Kolmogorov complexity K of an object x, using method S, is the length of the smallest program p that can output x, and then terminate. Mathematically, it can be represented as the following:

$$KS(x) = \min|p| : S(p) = n(x) \tag{11}$$

where $n(x)$ represents the numbers of some standard enumeration of objects x. Hence, the complexity of a program p_1 is less than the complexity of a program p_2 if the running time to get the output x_1 of p_1 is less than the running time to get the output x_2 of p_2. Dependently, we measured the running time of the proposed algorithm using different scenarios. Accordingly, the results conclude that the computational complexity of the proposed method is much better than the computational complexity of the state-of-the-art methods in all cases.

It is worth mentioning that the most time-consuming process is evaluating fitness, which can be implemented with parallel programming to improve efficiency in case of complicated multi-objective problems.

Figures 12, 13, and 14 show a comparative study between our proposed method and some of the sate-of-art methods using the benchmark maps that are declared at the third scenario (see Section 4.5) to measure their performance in terms of the path length, the smoothness of the plan, and the required time to get the optimum path. The experiments of this scenario were executed ten times, and the corresponding results are shown in Figures 12, 13, and 14.

From these findings, we can conclude that our proposed method yielded the shortest length. It is clear that our proposed method greatly extended the

Table 5: Average time (Avg.) and standard deviation (STD) (in seconds) to determine the robot path using the proposed method with different numbers of obstacles.

Algorithm	Results	Number of Obstacles				
		5	10	15	20	25
GA	Avg.	0.42	0.45	0.51	0.65	0.74
	SDT	0.10	0.22	0.14	0.28	0.47
Proposed Method	Avg.	0.51	0.56	0.59	0.62	0.70
	STD	0.16	0.19	0.19	0.25	0.32

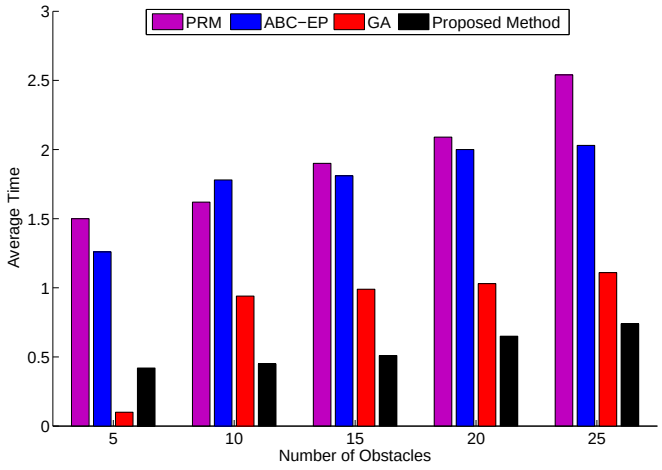

Figure 11: The average running time to get the optimum path.

robot life by avoiding its energy consumption. As the field enlarges, the robot life reduces given the same initial conditions, which is evident by comparing results of the three scenarios.

5. Conclusions

In this paper, a new method for the mobile robot path planning was proposed. In the proposed method, a Modified Genetic Algorithm (MGA) was introduced; and the goal of MGA was to increase the exploration capability

Table 6: Comparison between the proposed method and some state-of-the-art methods in terms average time (Avg.) to determine the robot path with different numbers of obstacles.

Methods	Number of Obstacles				
	5	10	15	20	25
PRM [Masehian and Sedighizadeh, 2010]	1.50	1.62	1.9	2.09	2.54
ABC-EP [Contreras-Cruz et al., 2015]	1.26	1.78	1.81	2.00	2.03
GA [Alajlan et al., 2013]	0.10	0.94	0.99	1.03	1.11
Proposed Method	0.51	0.56	0.59	0.62	0.70

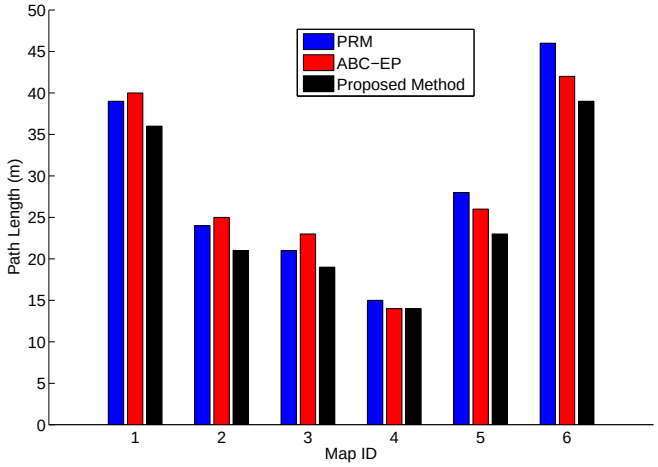

Figure 12: A comparison between path planning methods in terms of average path length (m) using benchmark maps that were listed in Table 4.

of the standard GA. The MGA was employed with the Bezier curve-based approach for the path planning in a dynamic field. Hence, the robot's path can be dynamically decided based on the obstacles' positions, and the goal of the proposed method is to search for the most suitable points as the control points of the Bezier curve. Using the selected control points, the optimal smooth path that minimizes the total distance between the start and end points is selected. Different experiments with different scenarios were conducted, and the

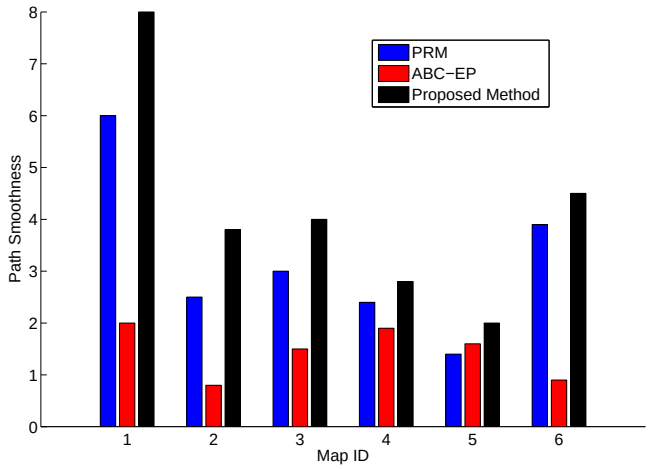

Figure 13: A comparison between path planning methods in terms of average smoothness of the path using benchmark maps that were listed in Table 4.

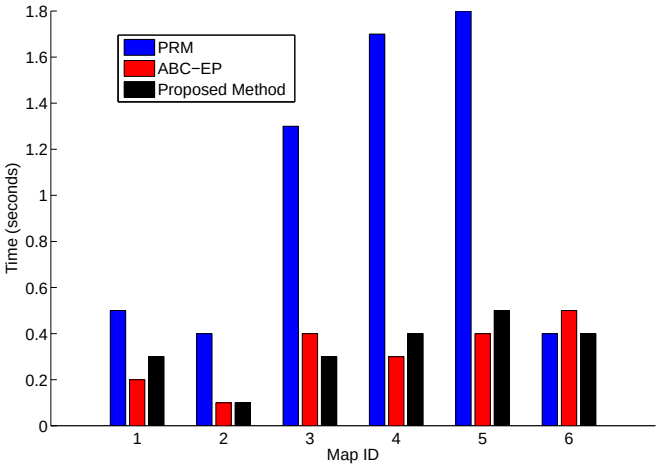

Figure 14: A comparison between path planning methods in terms of average total computation time (in seconds) using benchmark maps that were listed in Table 4.

proposed method generates paths in a small time compared with some state-of-the-art methods. Moreover, the proposed method provides an efficient way to avoid robot's energy consumption in harsh environments.

Our future work will be directed towards improving the local exploration process to avoid local optima problem. Moreover, conducting different experiments in a real large-scale dynamic environment to evaluate and improve our model.

References

Alajlan, M., Koubâa, A., Châari, I., Bennaceur, H., Ammar, A., 2013. Global path planning for mobile robots in large-scale grid environments using genetic algorithms. In: Proceedings of International Conference on Individual and Collective Behaviors in Robotics (ICBR). IEEE, pp. 1–8.

Arana-Daniel, N., Gallegos, A. A., López-Franco, C., Alanis, A. Y., 2014. Smooth global and local path planning for mobile robot using particle swarm optimization, radial basis functions, splines and bezier curves. In: IEEE Congress on Evolutionary Computation (CEC). IEEE, pp. 175–182.

Bakirtzis, A. G., Biskas, P. N., Zoumas, C. E., Petridis, V., 2002. Optimal power flow by enhanced genetic algorithm. IEEE Transactions on power Systems 17 (2), 229–236.

Chen, L., Wang, S., Hu, H., McDonald-Maier, K., 2012. Bézier curve based trajectory planning for an intelligent wheelchair to pass a doorway. In: International Conference on Control (CONTROL). IEEE, pp. 339–344.

Chiang, C.-L., 2005. Improved genetic algorithm for power economic dispatch of units with valve-point effects and multiple fuels. IEEE transactions on power systems 20 (4), 1690–1699.

Choi, J.-w., Curry, R., Elkaim, G., 2008. Path planning based on bézier curve for autonomous ground vehicles. In: Advances in Electrical and Electronics Engineering-IAENG Special Edition of the World Congress on Engineering and Computer Science, (WCECS'08). IEEE, pp. 158–166.

Chun, D. N., Yang, H. S., 1996. Robust image segmentation using genetic algorithm with a fuzzy measure. Pattern recognition 29 (7), 1195–1211.

Cimurs, R., Hwang, J., Suh, I. H., 2017. Bezier curve-based smoothing for path planner with curvature constraint. In: IEEE International Conference on Robotic Computing (IRC). IEEE, pp. 241–248.

Contreras-Cruz, M. A., Ayala-Ramirez, V., Hernandez-Belmonte, U. H., 2015. Mobile robot path planning using artificial bee colony and evolutionary programming. Applied Soft Computing 30, 319–328.

Das, P., Behera, H., Panigrahi, B., 2016. A hybridization of an improved particle swarm optimization and gravitational search algorithm for multi-robot path planning. Swarm and Evolutionary Computation 28, 14–28.

Duan, H., Qiao, P., 2014. Pigeon-inspired optimization: a new swarm intelligence optimizer for air robot path planning. International Journal of Intelligent Computing and Cybernetics 7 (1), 24–37.

Elhoseny, M., Yuan, X., Yu, Z., Mao, C., El-Minir, H. K., Riad, A. M., 2015. Balancing energy consumption in heterogeneous wireless sensor networks using genetic algorithm. IEEE Communications Letters 19 (12), 2194–2197.

Gálvez, A., Iglesias, A., Cabellos, L., 2013. Tabu search-based method for bézier curve parameterization.

Gardner, B., Selig, M., 2003. Airfoil design using a genetic algorithm and an inverse method. In: 41st Aerospace Sciences Meeting and Exhibit. p. 43.

Giannakoglou, K., 1998. A design method for turbine-blades using genetic algorithms on parallel computers. Computational Fluid Dynamics 98 (1), 1–2.

Golberg, D. E., 1989. Genetic algorithms in search, optimization, and machine learning. Addion wesley 1989, 102.

Goldberg, D. E., 2006. Genetic algorithms. Pearson Education India.

573 Gudmundsson, M., El-Kwae, E. A., Kabuka, M. R., 1998. Edge detection in
574 medical images using a genetic algorithm. IEEE transactions on medical imag-
575 ing 17 (3), 469–474.

576 Hasegawa, A. Y., Tormena, C., Parpinelli, R. S., 2014. Bezier curve parameter-
577 ization using a multiobjective evolutionary algorithm. International Journal
578 of Computer Science and Applications 11 (2), 1–18.

579 Hassan, R., Cohanim, B., De Weck, O., Venter, G., 2005. A compari-
580 son of particle swarm optimization and the genetic algorithm. In: 46th
581 AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Ma-
582 terials Conference. p. 1897.

583 Haupt, R. L., 1995. An introduction to genetic algorithms for electromagnetics.
584 IEEE Antennas and Propagation Magazine 37 (2), 7–15.

585 Ho, Y.-J., Liu, J.-S., 2009. Collision-free curvature-bounded smooth path plan-
586 ning using composite bezier curve based on voronoi diagram. In: Proceeding
587 of IEEE International Symposium on Computational Intelligence in Robotics
588 and Automation (CIRA). IEEE, pp. 463–468.

589 Hocaoglu, C., Sanderson, A. C., 1996. Planning multi-paths using speciation
590 in genetic algorithms. In: Proceedings of IEEE International Conference on
591 Evolutionary Computation. IEEE, pp. 378–383.

592 Holland, J. H., 1992. Genetic algorithms. Scientific american 267 (1), 66–72.

593 Hu, Y., Yang, S. X., 2004. A knowledge based genetic algorithm for path plan-
594 ning of a mobile robot. In: Proceedings of IEEE International Conference on
595 Robotics and Automation (ICRA'04). Vol. 5. IEEE, pp. 4350–4355.

596 Ismail, A., Sheta, A., Al-Weshah, M., 2008. A mobile robot path planning using
597 genetic algorithm in static environment. Journal of Computer Science 4 (4),
598 341–344.

599 Jolly, K., Kumar, R. S., Vijayakumar, R., 2009. A bezier curve based path plan-
600 ning in a multi-agent robot soccer system without violating the acceleration
601 limits. Robotics and Autonomous Systems 57 (1), 23–33.

602 Konak, A., Coit, D. W., Smith, A. E., 2006. Multi-objective optimization us-
603 ing genetic algorithms: A tutorial. Reliability Engineering & System Safety
604 91 (9), 992–1007.

605 Kuncheva, L. I., 2004. Combining pattern classifiers: methods and algorithms.
606 John Wiley & Sons.

607 Lee, K. Y., Bai, X., Park, Y.-M., 1995. Optimization method for reactive power
608 planning by using a modified simple genetic algorithm. IEEE Transactions on
609 Power Systems 10 (4), 1843–1850.

610 Li, B., Liu, L., Zhang, Q., Lv, D., Zhang, Y., Zhang, J., Shi, X., 2014. Path
611 planning based on firefly algorithm and bezier curve. In: IEEE International
612 Conference on Information and Automation (ICIA). IEEE, pp. 630–633.

613 Li, R., Wu, W., Qiao, H., 2015. The compliance of robotic hands–from func-
614 tionality to mechanism. Assembly Automation 35 (3), 281–286.

615 Li, Y.-W., Zhang, C.-Z., 2016. A study of bezier curve used in cnc based on dsp
616 and fpga. Journal of Computers 27 (2).

617 Liang, Z., Zheng, G., Li, J., 2012. Automatic parking path optimization based
618 on bezier curve fitting. In: IEEE International Conference on Automation
619 and Logistics (ICAL). IEEE, pp. 583–587.

620 Linquan, Y., Zhongwen, L., Zhonghua, T., Weixian, L., 2008. Path planning
621 algorithm for mobile robot obstacle avoidance adopting bezier curve based on
622 genetic algorithm. In: Chinese Control and Decision Conference. IEEE, pp.
623 3286–3289.

624 Ma, Y., Zamirian, M., Yang, Y., Xu, Y., Zhang, J., 2013. Path planning for mo-
625 bile objects in four-dimension based on particle swarm optimization method
626 with penalty function. Mathematical Problems in Engineering 2013.

Mahjoubi, H., Bahrami, F., Lucas, C., 2006. Path planning in an environment with static and dynamic obstacles using genetic algorithm: a simplified search space approach. In: 2006 IEEE International Conference on Evolutionary Computation. IEEE, pp. 2483–2489.

Manikas, T. W., Ashenayi, K., Wainwright, R. L., 2007. Genetic algorithms for autonomous robot navigation. IEEE Instrumentation & Measurement Magazine 10 (6), 26–31.

Masehian, E., Sedighizadeh, D., 2010. A multi-objective pso-based algorithm for robot path planning. In: Proceedings of IEEE International Conference on Industrial Technology (ICIT). IEEE, pp. 465–470.

Metawa, N., Hassan, M. K., Elhoseny, M., 2017. Genetic algorithm based model for optimizing bank lending decisions. Expert Systems with Applications 80, 75 – 82.
URL http://www.sciencedirect.com/science/article/pii/S0957417417301677

Minghua, Z., Guozhao, W., 2005. Genetic algorithm-based least square fitting of b-spline and bezier curves [j]. Journal of Computer Research and Development 1, 018.

Mohammed, G. A., Hou, M., 2016. Optimization of active muscle force–length models using least squares curve fitting. IEEE Transactions on Biomedical Engineering 63 (3), 630–635.

Pehlivanoglu, Y. V., 2012. A new vibrational genetic algorithm enhanced with a voronoi diagram for path planning of autonomous uav. Aerospace Science and Technology 16 (1), 47–55.

Peter, I., Peter, S., 2010. Path planning based on firefly algorithm and bezier curve. In: Proceedings of the 21st International DAAAM Symposium. DAAAM International, pp. 630–633.

Reeves, C. R., 1995. A genetic algorithm for flowshop sequencing. Computers & operations research 22 (1), 5–13.

Roberge, V., Tarbouchi, M., Labonté, G., 2013. Comparison of parallel genetic algorithm and particle swarm optimization for real-time uav path planning. IEEE Transactions on Industrial Informatics 9 (1), 132–141.

Robinson, D. C., Sanders, D. A., Mazharsolook, E., 2015. Ambient intelligence for optimal manufacturing and energy efficiency. Assembly Automation 35 (3), 234–248.

Sahingoz, O. K., 2014. Generation of bezier curve-based flyable trajectories for multi-uav systems with parallel genetic algorithm. Journal of Intelligent & Robotic Systems 74 (1-2), 499–511.

Scheunders, P., 1997. A genetic c-means clustering algorithm applied to color image quantization. Pattern Recognition 30 (6), 859–866.

Sid, B., Domaszewski, M., Peyraut, F., 2005. Topology optimization using an adaptive genetic algorithm and a new geometric representation. WIT Transactions on The Built Environment 80.

Škrjanc, I., Klančar, G., 2010. Optimal cooperative collision avoidance between multiple robots based on bernstein–bézier curves. Robotics and Autonomous systems 58 (1), 1–9.

Song, B., Song, B., Wang, Z., Wang, Z., Sheng, L., Sheng, L., 2016. A new genetic algorithm approach to smooth path planning for mobile robots. Assembly Automation 36 (2), 138–145.

Song, B., Tian, G., Zhou, F., 2010. A comparison study on path smoothing algorithms for laser robot navigated mobile robot path planning in intelligent space. JOURNAL OF INFORMATION &COMPUTATIONAL SCIENCE 7 (14), 2943–2950.

Tan, F., Fu, X., Zhang, Y., Bourgeois, A. G., 2008. A genetic algorithm-based method for feature subset selection. Soft Computing 12 (2), 111–120.

Tharwat, A., Hassanien, A. E., Elnaghi, B. E., 2017a. A ba-based algorithm for parameter optimization of support vector machine. Pattern Recognition Letters 93, 13–22.

Tharwat, A., Moemen, Y. S., Hassanien, A. E., 2017b. Classification of toxicity effects of biotransformed hepatic drugs using whale optimized support vector machines. Journal of Biomedical Informatics 68, 132–149.

Tsai, C.-C., Huang, H.-C., Chan, C.-K., 2011. Parallel elite genetic algorithm and its application to global path planning for autonomous robot navigation. IEEE Transactions on Industrial Electronics 58 (10), 4813–4821.

Tuncer, A., Yildirim, M., 2012. Dynamic path planning of mobile robots with improved genetic algorithm. Computers & Electrical Engineering 38 (6), 1564–1572.

Wahab, A. F., Zulkifly, M. I. E., 2017. A new fuzzy bezier curve modeling by using fuzzy control point relation. Applied Mathematical Sciences 11 (1), 39–57.

Wang, S., Chen, L., Hu, H., McDonald-Maier, K., 2012. Doorway passing of an intelligent wheelchair by dynamically generating bezier curve trajectory. In: IEEE International Conference on Robotics and Biomimetics (ROBIO). IEEE, pp. 1206–1211.

Wang, Y., Sillitoe, I. P., Mulvaney, D. J., 2007. Mobile robot path planning in dynamic environments. In: Proceedings 2007 IEEE International Conference on Robotics and Automation. IEEE, pp. 71–76.

Wu, C.-H., Tzeng, G.-H., Goo, Y.-J., Fang, W.-C., 2007. A real-valued genetic algorithm to optimize the parameters of support vector machine for predicting bankruptcy. Expert systems with applications 32 (2), 397–408.

Xiao, J., Michalewicz, Z., Zhang, L., Trojanowski, K., 1997. Adaptive evolutionary planner/navigator for mobile robots. IEEE transactions on evolutionary computation 1 (1), 18–28.

Yang, X.-S., 2014. Nature-inspired optimization algorithms. Elsevier, First Edition.

Yuan, J., Yu, T., Wang, K., Liu, X., 2007. Step-spreading map knowledge based multi-objective genetic algorithm for robot-path planning. In: 2007 IEEE International Conference on Systems, Man and Cybernetics. IEEE, pp. 3402–3407.

Yuan, Xiaohuiand Elhoseny, M. E.-M. H. K. R. A. M., 2017. A genetic algorithm-based, dynamic clustering method towards improved wsn longevity. Journal of Network and Systems Management 25 (1), 21–46.

Zhao, L., Jiang, J., Song, C., Bao, L., Gao, J., 2013. Parameter optimization for bezier curve fitting based on genetic algorithm. In: International Conference in Swarm Intelligence. Springer, pp. 451–458.

Zhong, W., Liu, J., Xue, M., Jiao, L., 2004. A multiagent genetic algorithm for global numerical optimization. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 34 (2), 1128–1141.

Zhou, F., Song, B., Tian, G., 2011. Bézier curve based smooth path planning for mobile robot. Journal of Information & Computational Science 8 (12), 2441–24450.

Ziolkowski, M., Gratkowski, S., 2016. Genetic algorithm coupled with bézier curves applied to the magnetic field on a solenoid axis synthesis. Archives of Electrical Engineering 65 (2), 361–370.

Mohamed Elhoseny is an associate professor at the Faculty of Computers and Information at Mansoura University, Egypt. He got his PhD 2016 in Computer and information Sciences a research scholar at Department of Computer Science and Engineering, University of North Texas, USA. His research interests include artificial wireless sensor network, data security, intelligence techniques, and big data.

Alaa Tharwat is a lecturer at the Faculty of Engineering, Suez Canal University, Ismailia, Egypt. His research interest includes evolutionary computing, image processing, and quantum computing. He has several Publications in reputed and high impact journals like Nature Scientific Reports, QINP, and Security and Communications. He has publications in international conferences held by IEEE, Springer and ACM.

Aboul Ella Hassanien is a Professor at Cairo University, Faculty of Computers & Information. He is the founder and Chair of the Scientific Research Group in Egypt (SRGE). He is the Ex-Dean of faculty of computers and Information, Beni-Suef University, Egypt

Highlights

- This paper proposes an efficient, Bezier curve based approach for the path planning in a dynamic field using Modified Genetic Algorithm (MGA).
- The challenges of selecting the shortest robot path in dynamic environments are discussed.
- The path planning problem is represented as an NP-hard optimization problem to be solved using heuristic algorithms such as evolutionary algorithms
- MGA is used to choose the optimum control points to draw the Bezier curve from the start and the end points.
- Bezier curve is applied to make get the smooth path.
- The proposed model is tested based on different scenarios. In addition to simulated environments, well-known benchmark maps are used to evaluate the performance of our proposed method.