KYBERNETIKA — VOLUME *60* (2024), NUMBER 3, PAGES 293–316

AN EFFECTIVE GLOBAL PATH PLANNING ALGORITHM WITH TEACHING-LEARNING-BASED OPTIMIZATION

Emad Hazrati Nejad, Sevgi Yigit-Sert, and Şahin Emrah Amrahov

Due to the widespread use of mobile robots in various applications, the path planning problem has emerged as one of the important research topics. Path planning is defined as finding the shortest path starting from the initial point to the destination in such a way as to get rid of the obstacles it encounters. In this study, we propose a path planning algorithm based on a teaching-learning-based optimization (TLBO) algorithm with Bezier curves in a static environment with obstacles. The proposed algorithm changes the initially randomly selected control points step by step to obtain shorter Bezier curves that do not hit obstacles. We also improve the genetic algorithm-based path planning algorithm. Experimental results show that they provide better paths than other existing algorithms.

Keywords: path planning, mobile robot, teaching-learning based optimization, Bezier curve

Classification: 68T40, 68W25, 78M50

1. INTRODUCTION

Planning is one of the important problems that can arise in the various areas, such as resource allocation [1], supply chain [42, 43], navigation [55, 16, 19], robotic coding [17] and many other areas. Recommender [6, 7, 47] or education systems [41] can also be seen as a type of planning tool. In recent years, the path planning problem for mobile robots has become one of the most important planning problems. Autonomous mobile robots are becoming more and more pervasive in our lives. They are used in many various areas such as mining, transportation, security, etc. and especially they are preferred in doing jobs that can cause irreparable harm to humans (e. g., radioactive chemical factories) [23]. As technology develops, humans try to equip these robots with artificial intelligence which introduced many research problems. Planning a collision-free path is one of them, besides, it contains sub-problems depending on various factors. Examples of these factors are the number of moving robots, the ability of robots to communicate with each other, and the environment in which the robot moves (global vs. local). For instance, there may be a variable number of obstacles, and these obstacles can be static or dynamic. The goal of the robot is to find the most suitable path from the starting point to the target point without hitting any obstacles around it.

DOI: 10.14736/kyb-2024-3-0293

Within a static environment, obstacles remain stationary, whereas in a dynamic setting, obstacles move in arbitrary directions at varying speeds. While global path planning involves a completely known map of the environment, local path planning operates in an environment where information is not known in advance [20], robots gather real-time environment information by their sensors.

The global path planning problem has recently drawn lots of attention from the researchers Li et al. [26] who proposed an enhanced version of the A* algorithm to address its limitations in path planning, such as long calculation times, large turning angles, and unsmoothed paths in large task spaces. They utilized a bidirectional alternating search (BAS) strategy for the efficiency of path searching, Euclidean distance weighted by exponential attenuation method for filtering paths, and Bézier curves for smoother paths. [9] proposed a path planning algorithm that utilizes the A* algorithm and potential field method to circumvent obstacles. [27] proposed a new path planning approach that combines the growth simulation concept and the level set-based heat conduction topology optimization. [14] introduced a new approach for path planning of robots moving in global environments. First, they employed the Lee algorithm and the RRT algorithm and then used Bezier curves for pruning and smoothing the raw shortest path.

The meta-heuristic algorithms especially Genetic Algorithm (GA) [2, 57, 28], have been widely employed in the global path planning of mobile robots. [13] improved the Whale Optimization Algorithm (NWOA) to address the challenges of slow convergence and local optimization in the path planning problem in dynamic environments, incorporating adaptive technology, virtual obstacles, and improved potential field factors. [38] enhanced ant colony optimization algorithm by introducing angle guidance factor, obstacle exclusion factor, adaptive adjustment factor, and pheromone volatilization factor. Then, the study takes into consideration multiple objectives, namely path length, safety degree, and energy consumption for path planning optimization. [58] introduced a hybrid algorithm, Genetic Firefly Algorithm (GFA) which combines GA and Firefly Algorithm (FA) to address FA's vulnerability to local optima, thereby enhancing both accuracy and performance in the pursuit of finding collision-free paths for mobile robots. [56] enhanced the bat algorithm (BA) by introducing a logarithmic decreasing strategy and Cauchy disturbance to improve its search capability. They employed this enhanced algorithm for global path planning as part of a hybrid solution with the dynamic window method to address local dynamic obstacle avoidance. [31] enhanced the grey wolf optimization (GWO) algorithm by incorporating the lion optimizer algorithm and dynamic weights. The performance of the proposed algorithm is evaluated in path planning applications. [60] developed a bio-inspired path planning method for mobile robots, addressing challenges in traditional algorithms by incorporating a linear path strategy, neighborhood search strategy, and an enhanced position update function for the sparrow search algorithm. [33] employed GA enhanced by integrating dynamic mutation rate and switchable global-local search method, along with cubic Bezier curves, to discover smoothed minimum-length paths. [30] showed a solution to the path planning problem deploying the ant colony optimization (ACO) algorithm improved using the geometric algorithm. [54] proposed smooth path planning of mobile robots based on a new quadruple Bezier transition curve and an improved particle swarm optimization (PSO) algorithm. [15] also utilized GA and Bezier curves for smooth path planning

for the dynamic environment. It employs genetic operations to determine Bezier curve control points, optimizing path length for efficiency and introducing safety measures for fitness function. [36] employed a modified GA to achieve the shortest and the smoothest path utilizing Bezier curves within a dynamic environment.

In recent years, numerous RRT (Rapidly-exploring Random Trees)-based algorithms proposed and utilized for addressing motion planning and obstacle avoidance challenges in mobile robot applications. RRT are a widely-used sampling-based algorithms that efficiently search for a feasible path in high-dimensional spaces, addressing complex constraints within the problem domain (see [18, 34] for details.) PQ-RRT* algorithm [29] was developed to integrate the advantages of both P-RRT* (Potential functions based RRT*) and Quick-RRT*, ensuring fast convergence to an optimal solution. However, it has some limitations, particularly concerning robot kinematic constraints and adaptability to complex environments. The study [50] improved the RRT algorithm by employing the simplified Bridge Test, point cloud clustering, and multi-tree growth strategies to tackle slow convergence and path quality issues. In [51], a convolutional neural network (CNN) trained with optimal paths generated by the A* algorithm is employed to learn a predicted probability distribution of the optimal path. Then, the predicted nonuniform sampling distribution is used to guide the sampling process of the RRT* planner. In a recent study [52], the combination of RRT* and ACO was employed to leverage the strengths of both methods, aiming to enhance search efficiency, convergence speed, and global search ability in automated guided vehicle path planning.

Reinforcement learning has been applied to find a collision-free path with the shortest path length for mobile robots. [37] presented an efficient Q-Learning algorithm for the global path planning optimization problem by introducing a new reward function for Q-table initialization and selection strategy to speed up convergence implying a reduction in computation time. [32] also studied a variant of reinforcement learning through the addition of a distance metric, modifications to the Q function, and the introduction of a virtual target. [26] presented an end-to-end path planning method starting from receiving data of camera and radar for intelligent driving vehicles in a static obstacle environment. Their algorithm, IDQNPER, is based on depth reinforcement learning and combines some DQN (Deep Q-Network) algorithms [40]. [35] introduced a deep graph model that incorporates the Floyd algorithm as a central component to optimize different paths for individual robots within a multi-robot system. Their algorithm initiates the process by extracting spatial-temporal interest points from each video clip. Subsequently, after quantization, the authors employed an enhanced version of Linear Discriminative Analysis (LDA) to effectively analyze and distinguish features for the optimization of robot paths.

In this study, we address the global path planning problem for a mobile robot in a static environment which has recently drawn significant attention from researchers. To tackle this issue, we propose using the Teaching-Learning-Based Optimization (TLBO) Algorithm [45], adapting it to determine the shortest and smoothest collision-free path by incorporating Bezier curves. The key contributions of our study are as follows:

- We propose a new TLBO-based algorithm that utilizes Bezier curves.
- Unlike all previous studies using Bezier curves, we do not select the control points defining the curve only from the obstacle-free region. After randomly selecting the

initial control points, we allow the control points created during our algorithm to be positioned above the obstacles.

- We compare our proposed algorithm with other algorithms using Bezier curves in the literature. The best among them is the Bezier Smoothing Algorithm with Increased Confidence Distance (BCA-Q) [36] based on a genetic algorithm. Our proposed algorithm outperforms BCA-Q, providing shorter and obstacle-free paths. In BCA-Q, the control points of the Bezier curve are selected from obstacle-free regions. We improve the BCA-Q algorithm by allowing the selection of control points from the entire region. We call this version of the BCA-Q algorithm the Improved Genetic algorithm (IGA).

  Since the flexibility of selecting control points increases the number of feasible Bezier curves, IGA theoretically always yields better results than BCA-Q. To highlight the effectiveness of our proposed algorithm, we compare it with the more competitive IGA, rather than BCA-Q, in environments not covered in [36].

- Our experiments demonstrate that the proposed algorithm outperforms the existing ones.

There are a few studies that have applied TLBO optimization for robot path planning. Aouf et al. [5] employed the TLBO algorithm to train an ANFIS controller for the robot navigation problem. Wu et al. [53] enhanced the TLBO algorithm by introducing inertia-weighted factors into its learning and memory mechanisms, and they applied this modified approach to solving the mobile robot path planning problem. Ansari and Katiya [4] conducted a performance comparison between ACO and TLBO algorithms for finding the shortest path from start to end points while avoiding collisions with obstacles. Although the smoothness of a path is a crucial factor in path planning, leading to a reduction in energy consumption and time wastage, they did not consider path smoothness in their study. Sabiha et al. [46] addressed the global path planning problem as a multi-objective optimization using the TLBO algorithm, considering objectives such as finding the shortest, smoothest, and collision-free path. Their method was tested in a single environment with fewer obstacles than our scenario.

The rest of the paper is organized as follows. The Bezier curve, TLBO algorithm, and GA is introduced in Section 2. Section 3 explains the proposed TLBO algorithm to get a smooth path based on the Bezier curve. Section 4 discusses the experimental results compared to GA approach. Section 5 concludes the paper and gives directions for future work.

2. BACKGROUND

2.1. Bezier curve

Pierre Bézier [8] introduced Bezier curves in 1968, and it holds significant importance in computer graphics. Recently, they have started to be used in path planning challenges. A Bezier curve of degree n is defined in a plane using $n + 1$ specified points, referred to as control points. These points are denoted as P_i, where i ranges from 0 to n, are given

by $P_i = (x_i, y_i)$. A n-degree Bezier curve is formulated as follows:

$$B(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i P_i. \tag{1}$$

Equation 1 is the parametric equation of the Bezier curve over $0 \leq t \leq 1$. Here, $\binom{n}{i} = \frac{n!}{(n-i)!i!}$ denotes the number of i^{th} combinations of n. Bernstein basis polynomials are denoted by:

$$b_{i,n}(t) = \binom{n}{i} (t)^i (1-t)^{n-i}. \tag{2}$$

where $i = 0, 1, \ldots, n$. Based on this, the formula for the n-degree Bezier curve can be expressed as follows:

$$B(t) = \sum_{i=0}^{n} b_{i,n} P_i, \qquad 0 \leq t \leq 1. \tag{3}$$

Equation 3 defines a Bezier curve on a grid, which is a rectangular domain with P_0 at the bottom-left corner and P_n at the top-right corner. Note that, a Bezier curve passes through the starting point P_0 and the ending point P_n, but it is not required to pass through all other points.

The derivative of an n-degree Bezier curve is calculated using the following formula.

$$B'(t) = \sum_{i=0}^{n-1} b_{i,n-1}(t)(P_{i+1} - P_i), \qquad 0 \leq t \leq 1. \tag{4}$$

2.2. The formula for the length of a curve given parametrically

Let a curve be defined parametrically by the formula $X = X(t) = (x(t), y(t))$, $a \leq t \leq b$. The length of this curve can be calculated using the formula:

$$\int_a^b ||X'(t)|| \, \mathrm{d}t. \tag{5}$$

Here, $||X'(t)||$ represents the Euclidean norm of the parametric function $X'(t)$. That is,

$$||X'(t)|| = \sqrt{(x'(t))^2 + (y'(t))^2}. \tag{6}$$

According to this formula, the length of the Bezier curve is equal to

$$\int_0^1 ||B'(t)|| \mathrm{d}t. \tag{7}$$

In this study, Simpson's formula has been used to approximate the value of the integral in Equation 7. When a continuous function $y = f(x)$ defined on the interval $[a, b]$ is given, the integral $\int_a^b f(x)||\mathrm{d}x$ can be calculated using the Simpson's formula as follows:

$$I = \frac{\Delta x}{3}(y_0 + 4y_1 + 2y_2 + 4y_3 + 2y_4 + \cdots + 4y_{n-1} + y_n) \tag{8}$$

where $\Delta x = \frac{b-a}{n}$, $x_i = a + i\Delta x$, and $y_i = f(x_i), i = 0, 1, \ldots, n$

2.3. Teaching-learning-based optimization algorithm

Teaching Learning Based Optimization (TLBO) [45] is a metaheuristic optimization algorithm inspired by the principles of teaching and learning in a classroom environment. It simulates the process of knowledge transfer among students and teachers to improve iteratively the quality of solutions by operating on a population of candidate solutions to optimization problems.

TLBO algorithm comes to the forefront for its simplicity and effectiveness in solving optimization problems. Unlike many traditional optimization algorithms, TLBO does not rely on complex mathematical models or sophisticated operators. Instead, it leverages the inherent parallelism of the teaching and learning process to efficiently explore the solution space and converge towards optimal or near-optimal solutions. This simplicity makes TLBO particularly suitable for solving real-world optimization problems where computational resources are limited or the problem structure is not well-defined. Furthermore, TLBO does not require any problem-specific parameter tuning, making it easy to implement and apply in practice. This versatility and ease of use have contributed to the widespread adoption of TLBO across various fields, including engineering [10, 59], finance [24], image processing [44, 3], and scheduling [49].

Since the teaching and learning phases are fundamental to TLBO, they mirror the educational process. During the teaching phase, the best-performing individual (referred to as the teacher) within the population imparts her/his knowledge to poorer-performing individuals (referred to as students). This knowledge transfer mimics the learning process, wherein poorer-performing solutions adjust themselves towards the optimal solution and refine their approaches based on the guidance provided by the teacher. Additionally, aside from learning from the teacher, students also enhance their knowledge through interaction with one another. Consequently, the TLBO algorithm comprises two phases: the teacher phase and the learner phase.

In the teacher phase, an individual from the population with the highest level of knowledge assumes the role of the teacher. Let S be a population matrix, where n represents the number of students and m represents the number of courses. Each row of this student-course matrix represents the grades of a student across different courses.

$$S = \begin{bmatrix} s_{1,1} & s_{1,2} & \dots & s_{1,m} \\ s_{2,1} & s_{2,2} & \dots & s_{2,m} \\ \vdots & \vdots & \vdots & \vdots \\ s_{n,1} & \dots & \dots & s_{n,m} \end{bmatrix}. \tag{9}$$

Let's illustrate this with an example. In our class (or population), we have 5 students, each enrolled in 4 courses, with grades assigned to each student for these courses. Assume S is as follows:

$$S = \begin{bmatrix} 3 & 4 & 5 & 2 \\ 5 & 3 & 4 & 4 \\ 5 & 2 & 1 & 3 \\ 4 & 3 & 2 & 2 \\ 3 & 4 & 3 & 2 \end{bmatrix}. \tag{10}$$

The mean value for each course is stored as a row vector denoted as S_m. Thus, the j^{th} component of S_m is the average of the j^{th} column of matrix S. The calculated value of S_m for matrix S is as follows:

$$S_m = [4 \quad 3.2 \quad 3 \quad 2.6]. \tag{11}$$

Let's clarify the process to provide a clearer understanding of the forthcoming formulas. The algorithm aims to iteratively improve a randomly generated initial matrix S to produce an optimal matrix. Each iteration involves multiple steps to generate a new matrix S from an existing matrix S through teacher and student phases. Each iteration starts with the matrix S which is denoted as S_{old}. The matrix after each step is called the current matrix S. At the end of each iteration, the current matrix S is S_{new}.

The first step of the teacher phase is to select the student with the highest fitness value from S as the teacher, representing the student with the greatest knowledge. The fitness function is the total grades of the students. Subsequently, the teacher tries to increase the mean value of the class by employing Equation 12. s_{teacher} is the second row vector of S in the example above, as it has the highest total grade of 18.

$$s_i^{new} = s_i^{old} + r_i * (s_{\text{teacher}} - t * S_m). \tag{12}$$

where r_i is a random value such that $r_i \in [0, 1]$, and t shows the teaching factor, which can only take the values 1 or 2. s_i^{new} represents the i^{th} row of the current matrix S, while s_i^{old} represents the i^{th} row of matrix S from the previous iteration. For instance, the first row would be [3.8 3.9 5.5 2.7] when r is 0.5 and t is 1. This process is repeated for all other students.

In the learner phase following the teacher phase, each student engages with a randomly selected student from the class. Suppose we have two randomly selected students, s_i and s_j. They are the row vectors of the current matrix S, obtained at the end of the teacher phase. The student with less knowledge improves through knowledge exchange with the other student, as indicated by Equation 13.

$$s_i^{new} = s_i^{old} + h(i, j) * r_i * (s_i^{old} - s_j), \tag{13}$$

where

$$h(i, j) = \begin{cases} 1, & f(s_i) < f(s_j) \\ -1, & \text{otherwise}, \end{cases} \tag{14}$$

where $f(s_i)$ represents fitness value of solution s_i. If a new solution results in a better fitness value, the TLBO algorithm continues with this solution.

This iterative process continues until the maximum iteration number is reached. The flowchart depicting the TLBO algorithm is provided in Figure 1.

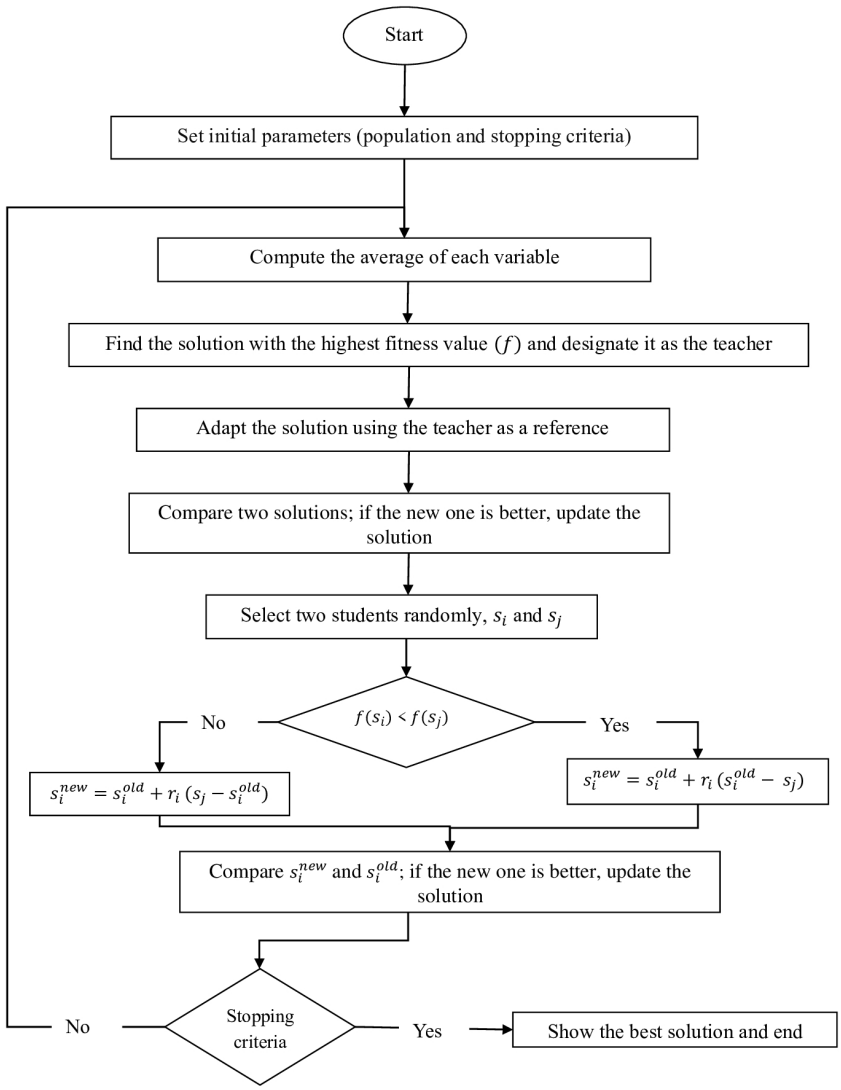


Fig. 1: Flowchart of optimization algorithm based on teaching and learning.

2.4. Genetic algorithm

Genetic Algorithm is mainly used for path planning in static and/or dynamic environments (see Section 1 for details). We also employ a genetic algorithm in the evaluation of our method as a baseline. For this reason, this section gives an overview of the genetic algorithm and its operators.

The genetic algorithm is one of the most popular evolutionary algorithms that examine search space to find optimum or approximate solutions for complex and complicated problems. It is a population-based algorithm that is inspired by biological evolution [21, 39].

The genetic algorithm begins with a random population. Each individual in the population is called a chromosome which indicates a candidate solution to the problem. A chromosome consists of some structures called genes that represent a set of variables in the problem of interest. Nature selection favors the fittest individuals for food and reproduction, causing their genes to be more highly inherited by the next generation of their species. This is the essential component of a genetic algorithm. The algorithm selects some individuals from the current population to produce the new generation. To do so, the genetic algorithm computes a fitness value for each chromosome employing a fitness function to measure their quality. The higher-valued ones are considered the best individuals and are been applied genetic operators, namely crossover and mutation, to create the new population. Crossover operation combines some parts determined by crossover point(s) of parental chromosomes that are randomly chosen among individuals in the previous step. Each new child has a new set of parents and this procedure lasts until the population size is reached. Then mutation (random bit flips) operator comes into play to incorporate diversity in the population. Through it, new solutions are avoided from becoming similar and the likelihood of trapping in local solutions is reduced.

We employ a genetic algorithm for path planning in static environments. The first task in the genetic algorithm is to create an initial population, thereby we need to define chromosomes to signify our problem. Each chromosome comprises a specific number of control points, with each gene within a chromosome representing x and y coordinates, both of which are integer values within the range of 0 to 100. The values of the first and last genes in the chromosome are (0,0) and (100,100) since the first and last genes represent the starting and destination points, respectively. The chromosomes are utilized to define a Bezier curve. We do not restrict control points to be obstacle-free, leading to an improved version of the genetic algorithm approach proposed by Ma et al. [36], which we henceforth refer to as IGA.

A single crossover operation is implemented, exchanging control points from random positions between the parents. Afterward, the newly generated individuals undergo obstacle checks; if they pass, they are integrated into the population. Next, a mutation operation is carried out, and the resulting individuals are likewise included. Roulette wheel selection is then utilized to ensure the population size remains constant. The fitness value of each chromosome is the length of the Bezier curve. The shorter the length of the curve, the higher the fitness value of the chromosome, so the better the quality solution.

3. PROPOSED APPROACH

This section explains the proposed approach based on the TLBO algorithm for finding a global path.

3.1. Initial population generation

The first step of meta-heuristic algorithms is to generate an initial population. For this reason, we create the initial population, with each individual composed of X and Y coordinates, representing the control points of a Bezier curve. The initial and final control points represent the starting and target points of the robot's path, respectively, while the intermediate points between them are randomly generated.

3.2. Teacher phase

The teacher phase in the TLBO algorithm plays a crucial role in guiding the search process toward better solutions. During this phase, the algorithm mimics the concept of a teacher imparting knowledge to students. The teacher represents the best solution concerning fitness function found so far in the optimization process. When the teacher is decided, s/he shares its knowledge with the students according to Equation 12 that helps the teacher to guide the students towards better solutions. Consequently, the entire population converges towards better solutions with each iteration.

For instance, assume a matrix S as described in Section 2.3 with 5 rows. However, this time each row of the matrix S consists of control points. Let's refer to each row as a solution, s_i, and consider that each solution s_i has 4 control points $[(x_1, y_1) \quad (x_2, y_2) \quad (x_3, y_3) \quad (x_4, y_4)]$. Each cell of the matrix S represents (x, y) coordinates of a control point. Assume the starting point is $(0, 0)$ and ending point is $(5, 5)$ and the S is as follows:

$$S = \begin{bmatrix} (0,0) & (1.1,1.3) & (3.8,2.4) & (5,5) \\ (0,0) & (1.0,2.95) & (2.7,4.05) & (5,5) \\ (0,0) & (0.5,2.3) & (3.3,3.3) & (5,5) \\ (0,0) & (3.8,1.7) & (3.6,4.5) & (5,5) \\ (0,0) & (1.95,2.45) & (3.8,4.0) & (5,5) \end{bmatrix}. \tag{15}$$

Firstly, we compute Bezier curves using the control points in each row: If a curve collides with any obstacle, it is replaced with a random solution. If not, we compute the length of Bezier curve using the Equation 7.

Since the fitness value is the path length, the row with the shortest path length is chosen as the teacher, s_{teacher}. Assume s_3 has the smallest value, so s_{teacher} is s_3. After selecting the teacher, students will adjust their solutions according to the teacher using Equation 12. In Equation 12, S_m denotes the mean value for each y dimension, representing a course for the students. However, in our scenario, computing the mean of control points is not meaningful, so we have modified Equation 12. Instead of calculating S_m, we directly use the disparity between student and teacher control points as an indicator of potential directions for improvement.

We use pointwise subtraction to measure the distance between the current solution and the best solution and then update the current solution by adding this difference. The update rule of our algorithm is given Equation 16.

$$s_i^{new} = s_i^{old} + r_i * (s_{\text{teacher}} - s_i^{old}) \tag{16}$$

The S will be as follows when r is 0.5.

$$S = \begin{bmatrix} (0,0) & (0.8,1.8) & (3.55,2.85) & (5,5) \\ (0,0) & (0.75,2.625) & (3.0,3.675) & (5,5) \\ (0,0) & (0.5,2.3) & (3.3,3.3) & (5,5) \\ (0,0) & (2.15,2.0) & (3.5,3.9) & (5,5) \\ (0,0) & (1.225,2.375) & (3.55,3.65) & (5,5) \end{bmatrix}. \tag{17}$$

Bezier curves of the current S are now generated using Equation 2. These curves are then evaluated for collisions with obstacles. Paths that collide with obstacles are replaced with random control points.

After one iteration of the teacher phase is completed, an iteration of the learner phase described below begins and these two phases follow each other until the maximum number of iterations is reached.

3.3. Learner phase

In the learning phase of the TLBO algorithm, individuals, namely students, iteratively enhance their solutions through knowledge sharing. Two students are chosen from the population randomly, and their fitness (knowledge) is evaluated. After selecting the two student solutions, Bezier curves are generated, and their lengths are computed for comparison. It is important to note that we do not need to check for obstacle collisions because we ensure in previous iterations that each solution is collision-free. In the original algorithm, each student adjusts its solution according to the difference in fitness value with a randomly selected student. We slightly modify this approach so that the student with the lower fitness value adjusts its solution based on the better-performing student's solution. With this change, not every student modifies its solution; only those with poorer fitness values do. Suppose we have two randomly selected solutions indexed as i and j. If solution s_i has a longer path, we update this solution using the following equation:

$$s_i^{new} = s_i^{old} + r_i * (s_j - s_i^{old}). \tag{18}$$

If the opposite case occurs, where s_i has a shorter path, the equation changes to the following:

$$s_j^{new} = s_j^{old} + r_i * (s_i - s_j^{old}). \tag{19}$$

Subsequently, Bezier curves are generated using the new solutions and then validated for collisions, similar to the process in the teacher phase. The pseudocode of our proposed path planning method based on TLBO is presented in Algorithm 1.

Algorithm 1: Path Planning Algorithm based on TLBO.

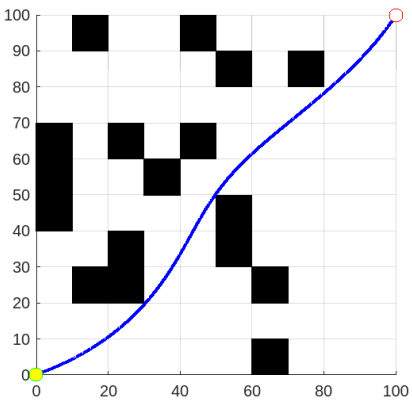
Input : N : population
M : number of control points
G : maximum iteration number
k : number of obstacles
Output: p : the shortest collision-free path

Generate a random population in which each individual consists of M control points
Set k obstacles at random points
for each individual $i \in N$ **do**
    Find the corresponding Bezier curve, p, for i
    if *p conflicts one of the obstacles* **then**
        remove p from N;
    else
        compute the length of p
    end
end
for *iter* $\leftarrow 1$ **to** G **do**
    Assign the individual with the smallest length as the teacher, t
    for each student $s \in N$ **do**
        Produce s^{new} by changing control points in s w.r.t to t
        Find the corresponding Bezier curve, p^{new}, for s^{new}
        if *p^{new} does not conflict one of the obstacles* **then**
            Compute the length of p^{new}
            if $length(p^{new}) < length(p)$ **then**
                Replace s^{new} with s
            end
        end
    end
    for $j \leftarrow 1$ **to** N **do**
        Select a random student, s_i $(i \neq j)$
        if $length(p_i) < length(p_j)$ **then**
            Produce s_j^{new} w.r.t to s_i
        else
            Produce s_i^{new} w.r.t to s_j
        end
        Find the corresponding Bezier curve for the new solution (s^{new})
        Compute the length of the new solution
        if $length(p^{new}) < length(p^{old})$ **then**
            Replace s^{new} with s^{old}
        end
    end
end

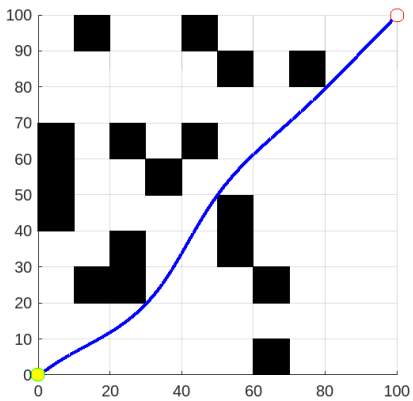
4. EXPERIMENTAL SET-UP AND RESULTS

We conduct our experiments in different environments to check the effectiveness of our proposed methods. Unlike other optimization techniques, TLBO does not require any parameters to be tuned, while GA requires parameters such as crossover and mutation rates. In the experiments, we varied the mutation rate between 0.01 and 0.3 and the crossover rate between 0.5 and 0.9 across all three environments. The best results were achieved with a mutation rate of 0.01 and a crossover rate of 0.7. Therefore, the results are presented with these values: a mutation rate of 0.01 and a crossover rate of 0.7. The population size is set to 100 for both TLBO and IGA, and we chose a maximum iteration number of 30, which serves as the stopping criteria for both algorithms. The fitness function for both TLBO and IGA aims to find the shortest and collision-free path from the start point to the target point. This fitness function evaluates paths based on the length of the Bezier curve. The safety margin around obstacles is set to 0.2 units.

To validate the path planning for the mobile robot, we conducted tests in three distinct grid environments with varying obstacles. The grid's bottom-left corner, serving as the robot's starting point, is located at (0, 0), while the top-right corner, representing the robot's destination, is at (100, 100). With these start and end points, we used four additional control points for the Bezier curves. As the degree of the Bezier curve increases, more complex Bernstein polynomials are obtained, resulting in more precise paths. This becomes particularly important when the number of obstacles increases. Higher-order Bezier curves provide more suitable paths for the robot to navigate in complex environments. We present the results of the TLBO algorithm for cases with control points set at 6 and 20 in Figure 2 to provide a clearer basis for comparison. The Bezier curve with 6 control points yields a path length of 143.72, while the curve with 20 control points yields 143.20.



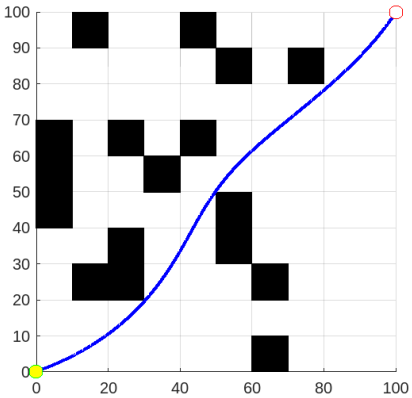
((a)) TLBO Algorithm using 6 control points.



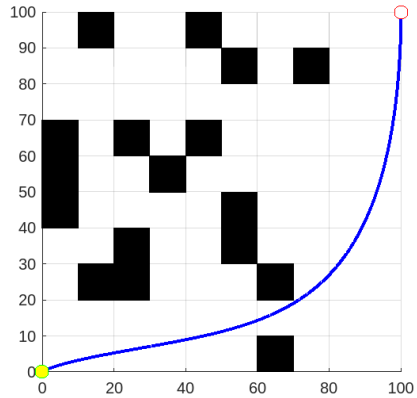
((b)) TLBO Algorithm using 20 control points.

Fig. 2: Path planning simulation results of TLBO using 6 and 20 control points.

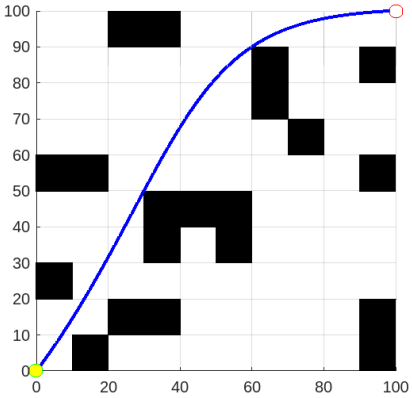
Figure 3 illustrates the optimal paths obtained by the TLBO and IGA algorithms in three distinct environments. The left side of the figure corresponds to TLBO, while the



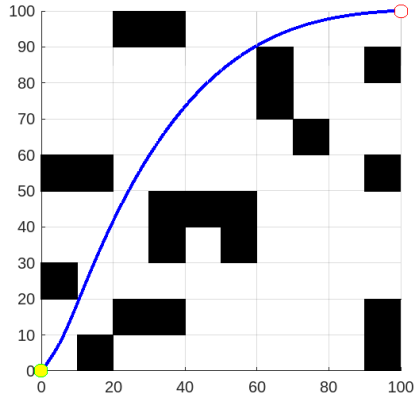
((a)) TLBO on Environment 1.



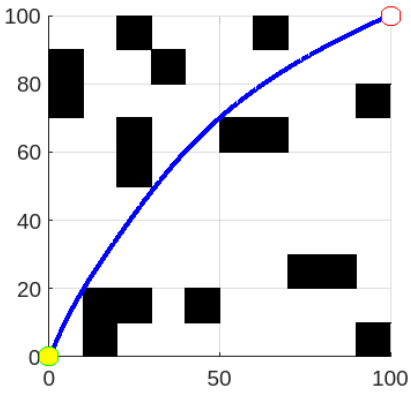
((b)) IGA on Environment 1.



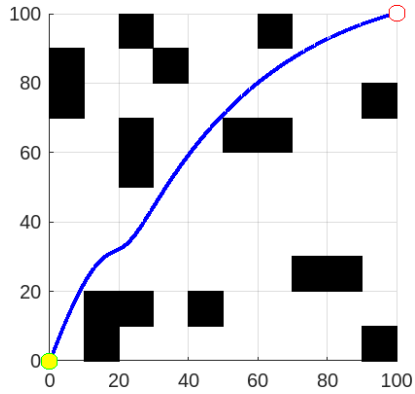
((c)) TLBO on Environment 2.



((d)) IGA on Environment 2.



((e)) TLBO on Environment 3.



((f)) IGA on Environment 3.

Fig. 3: Simulation results of path planning across three different environments using TLBO and IGA.

right side depicts IGA results. It is evident that both algorithms successfully identify the shortest collision-free path in each environment. Although visual comparison may not demonstrate TLBO's better performance over IGA, we provide the path lengths in Table 1 for a comprehensive analysis. The table presents the best, worst, and average (from ten independent runs) values of paths obtained by the TLBO algorithm and IGA. The findings show that the TLBO algorithm outperforms its counterpart suggesting a relative decline of up to 1.34% (143.72 vs. 145.67). For instance, for Environment 3, TLBO yields 145.16, 145.52, and 145.36, whereas the path lengths obtained by IGA are 146.63, 146.72, and 146.0. It is important to note that the TLBO algorithm demonstrates better performance across all cases (i. e., best, worst, and average) for each environment.

-	TLBO			IGA		
	Best	Worst	Average	Best	Worst	Average
Env. 1	**143.72**	**161.70**	**149.55**	145.67	163.13	153.59
Env. 2	**150.57**	**150.69**	**150.64**	150.90	152.25	151.65
Env. 3	**145.16**	**145.52**	**145.36**	146.63	146.72	146.0

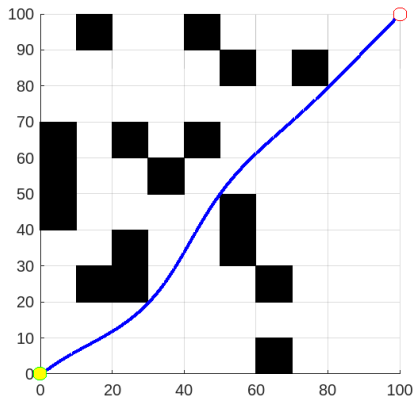
Tab. 1: The length of paths generated by TLBO algorithm and IGA using 6 control points.

In our next experiment, we aim to investigate the extent to which the number of control points affects the drawing of Bezier curves based on the TLBO algorithm and IGA, potentially resulting in shorter paths. We select the "Environment 1" for comparison and increase the number of control points to 20. While the simulation results are presented in Figure 4, the path length results are shown in Table 2. They indicate that both methods achieve shorter paths compared to those in Table 1, with TLBO outperforming IGA once again. When comparing the performance of the TLBO algorithm and IGA in Table 1, we observe a significant decrease in path lengths. For instance, in the worst-case scenario, the path length decreases from 161.7 to 143.46 with TLBO and from 163.13 to 144.04 with IGA.

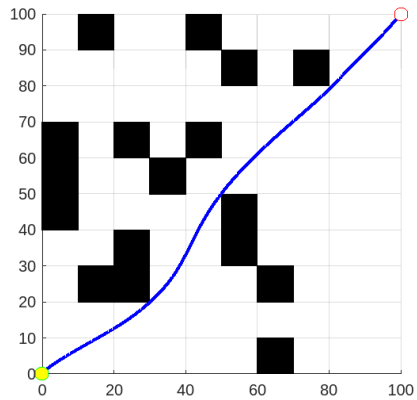
-	TLBO			IGA		
	Best	Worst	Average	Best	Worst	Average
Env. 1	**143.20**	**143.46**	**143.37**	143.27	144.04	143.56

Tab. 2: The length of paths generated by TLBO algorithm and IGA using 20 control points.

Lastly, we conduct an additional experiment to demonstrate the superiority of our approach by comparing it with a prior study [36], which uses GA with Bezier curves. They refer to their Bezier curve smoothing algorithm based on GA as "BCA" and its enhanced version, which includes increasing the safety distance and introducing an adaptive penalty factor in the fitness function, as "BCA-Q". To prevent confusion with the earlier environments, we have designated Environment 1 in [36] as "Environment 4" and Environment 2 in [36] as "Environment 5" in our study.



((a)) TLBO with 20 control points.



((b)) IGA with 20 control points.

Fig. 4: Path planning simulation results of TLBO algorithm and IGA in Environment 1 using 20 control points.

For a comprehensive comparison, we visually present the optimal paths obtained by our algorithms and BCA-Q in Figure 5 and Figure 6, using the environments provided by [36]. Figure 5 displays the optimal paths generated by TLBO and IGA, while Figure 6 shows the simulation results of BCA-Q, extracted directly from the paper. The first column of figures, (a), corresponds to Environment 4, while the second column of figures, (b), corresponds to Environment 5. It can be observed that both methods, namely TLBO and IGA, yield collision-free, smooth paths; however, the TLBO algorithm provides a shorter path compared to IGA and BCA-Q.

	Best	Worst	Avg
ACO[12]	34.62	50.38	42.54
GA[22]	32.87	38.69	35.71
ACO-GA[11]	31.80	35.41	33.42
ASFA-GA[36]	31.21	33.55	32.36
BCA[36]	29.94	32.45	30.94
BCA-Q[36]	31.18	34.85	33.00
IGA	29.70	29.82	29.75
TLBO	**29.66**	**29.72**	**29.69**

Tab. 3: For Environment 4, the length of paths generated by some algorithms presented in [36] with our proposed methods.

Table 3 and Table 4 present the path length results of several heuristic methods, namely ACO[12], GA[22], ACO-GA[11], BCA, and BCA-Q, as reported in [36], alongside our proposed methods for two different environments in [36], respectively. When we examine tables, BCA and BCA-Q show inferior performance compared to our proposed methods, implying longer paths. TLBO shows the best performance yielding the shortest paths with the scores of 29.66, 29.72, 29.69 for Environment 4 and 29.61, 28.87, and 29.76

	Best	Worst	Avg
ACO[12]	35.45	56.11	46.00
GA[22]	33.80	40.45	37.54
ACO-GA[11]	32.97	37.37	35.15
ASFA-GA[36]	32.38	34.77	33.66
BCA[36]	30.35	33.77	32.14
BCA-Q[36]	31.85	35.45	33.76
IGA	30.08	30.41	30.23
TLBO	**29.61**	**29.87**	**29.76**

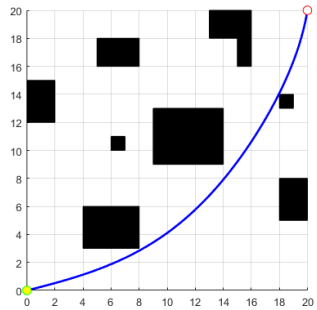
Tab. 4: For Environment 5, the length of paths generated by some algorithms presented in [36] with our proposed methods.

for Environment 5.

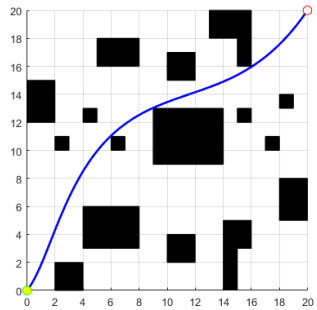
When comparing the performance of TLBO to BCA-Q [36], we observe a relative decline of 4.87%, 14.72%, and 11.84% for TLBO and 4.75%, 14.43% and 9.85% for the IGA in Environment 4. For Environment 5, the relative decline reaches up to 15.74% and 14.22% for TLBO and IGA, respectively.

The reader may wonder why there is a performance difference between BCA/BCA-Q and IGA, despite both employing GA with Bezier curves for the same environments and under similar settings. The reason for this might arise from the selection of control points for Bezier curves. In the former, control points are chosen outside of obstacles. However, in our approach, we do not impose restrictions on control points being obstacle-free; some of the control points may be in the obstacle. Instead, we discard paths with collisions and compel the algorithm to find a collision-free path.

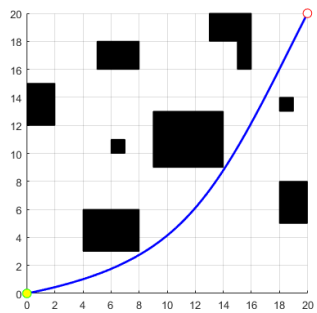
The idea of assuming that robots move along a specific spline has been around for nearly 30 years [48]. In the earliest studies, there were no obstacles in the environment, but this idea ensures that the robot moves along a smooth curve. As it is known, there are many types of splines, some of them pass through control points, and some do not. After determining the spline on which the robot moves in an obstacle-free environment, it is necessary to check whether this spline intersects with obstacles at each step of the proposed algorithm. If the algorithm determines the best path in m steps and collision checks are performed at n points in each step, this check alone requires $O(mn)$ processing time. Since splines passing through control points were used in the first studies, it was naturally necessary to take control points from outside the obstacles (or even from a certain area around the obstacles). However, although there is no guarantee of not colliding, researchers did not perform collision checks in their algorithms they proposed to avoid the extra $O(mn)$ processing time explained above. Later, when splines that did not pass through control points began to be used, researchers continued this tradition. Instead of performing collision checks, they were content to select control points from areas around the obstacles. The Bezier splines in this study only pass through the first and last control points. Selecting control points outside obstacles reduces the number of feasible Bezier curves that robots can move. This means that even if some control points are over obstacles, the Bezier curve determined by them might not collide with the obstacles. Additionally, the best curve could be one of the reduced curves. Therefore,



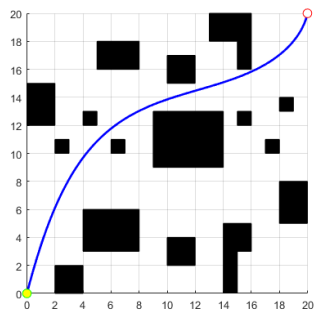
((a)) TLBO in Environment 4.



((b)) TLBO in Environment 5.

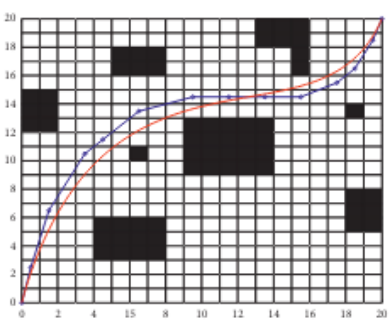


((c)) IGA in Environment 4.

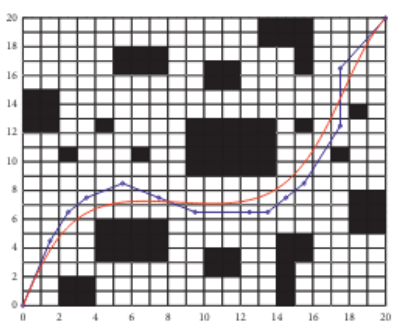


((d)) IGA in Environment 5.

Fig. 5: Path planning simulation results of our methods across two different environments from [36].



((a)) BCA-Q in Environment 4.



((b)) BCA-Q in Environment 5.

Fig. 6: Path planning simulation results of [36] in the same environments as in Figure 5.

in the first step, we randomly select control points from the entire environment, and in the next steps, we do not check whether the control points newly created by the algorithm are on obstacles. Instead, we check whether the curve we obtained collides with obstacles at each step.

Let's demonstrate what we explained above with a simple example.

Example. Let control points $P_0 = (0, 0), P_1 = (1, 2), P_2 = (3, 3)$ be given. Let also the obstacle are given as a rectangle determined with $0 \leq x \leq 1.2$ and $1.8 \leq y \leq 3$. Control point P_1 is inside this rectangle. The parametric formula of the Bezier spline determined by the given control points is as follows:

$$x(t) = t^2 + 2t = (t+1)^2 - 1$$

$$y(t) = 4t - t^2 = -(t-2)^2 + 4.$$

From here, we have

$$\frac{dy}{dx} = \frac{t+1}{2-t} > 0$$

. In other words, the function $y = -(\sqrt{x+1})^2 + 4$ determined by these parametric equations is an increasing function in the range $[0,\ 3]$. Since $y = -(\sqrt{2.2} - 3)^2 + 4 \approx 1.7 < 1.8$ for $x = 1.2$, it can be seen that the Bezier curve does not intersect with the given obstacle. This example is illustrated in Figure 7.

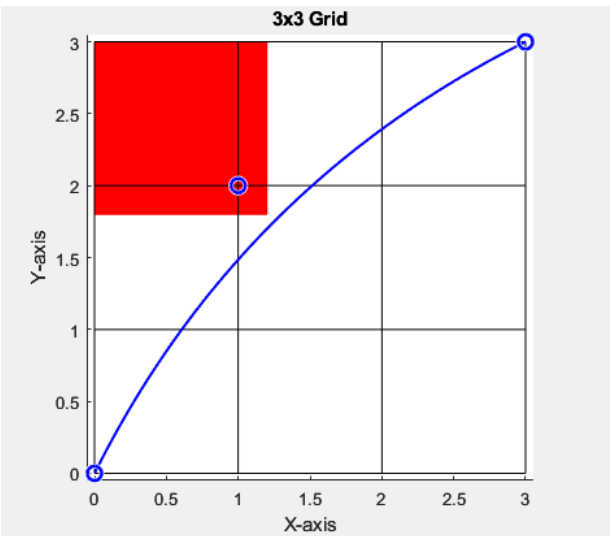


Fig. 7: An obstacle-free Bezier curve with a control point inside the obstacle area.

5. CONCLUSION

This paper presents a novel mobile robot path planning approach that combines the TLBO algorithm with Bezier curves. The TLBO algorithm efficiently determines the control points for the Bezier curve, which inherently generates smooth and continuous paths. This smoothness minimizes energy consumption during robot movement. We tested the proposed algorithm in diverse environments, including complex scenarios with numerous obstacles. The results demonstrate the effectiveness of the method in finding optimal paths that are collision-free, smooth, and short. Nevertheless, several meaningful topics remain to be addressed, determining the optimal number of control points for the Bezier curve to balance smoothness and computational complexity, and adapting the algorithm for dynamic environments.

REFERENCES

[1] R. M. Alguliyev, R. M. Aliguliyev, and R. G. Alakbarov: Constrained K-means algorithm for resource allocation in mobile cloudlets. Kybernetika *59* (2023), 1, 88–109. DOI:10.14736/kyb-2023-1-0088

[2] S. Alnasser and H. Bennaceur: An efficient genetic algorithm for the global robot path planning problem. In: Sixth International Conference on Digital Information and Communication Technology and its Applications (DICTAP), Turkey 2016, pp. 97–102.

[3] K. M. Ang, E. S. M. El-kenawy, A. A. Abdelhamid, A. Ibrahim, A. H. Alharbi, D. S. Khafaga, S. S. Tiang, and W. H. Lim: Optimal design of convolutional neural network architectures using teaching-learning-based optimization for image classification. Symmetry *14* (2022), 2323. DOI:10.3390/sym14112323

[4] A. Q. Ansari and I. Katiyar: Comparison and analysis of obstacle avoiding path planning of mobile robot by using ant colony optimization and teaching learning based optimization techniques. In: Proc. First International Conference on Information and Communication Technology for Intelligent Systems, Volume 2. Smart Innovation, Systems and Technologies, 2016, pp. 563–574.

[5] A. Aouf, L. Boussaid, and A. Sakly: TLBO-based adaptive neurofuzzy controller for mobile robot navigation in a strange environment. Comput. Intell. Neurosci. *4* (2018). DOI:10.1155/2018/3145436

[6] Y. Ar: An initialization method for the latent vectors in probabilistic matrix factorization for sparse datasets. Evolution. Intell. *13* (2020), 2, 269–281. DOI:10.1109/MITS.2021.3116446

[7] Y. Ar, S. Emrah Amrahov, N. Gasilov, and S. Yigit-Sert: A new curve fitting based rating prediction algorithm for recommender systems. Kybernetika *58* (2022), 3, 440–455. DOI:10.14736/kyb-2022-3-0440

[8] P. Bezier: Style, mathematics and NC. Computer-aided Design *22* (1990), 9, 524–526. DOI:10.1016/0010-4485(90)90037-D

[9] D. Bodhale, N. Afzulpurkar, and N. T. Thanh: Path planning for a mobile robot in a dynamic environment. In: IEEE International Conference on Robotics and Biomimetics, Thailand 2009, pp. 2115–2120.

[10] H. Bouchekara, M. Abido, and M. Boucherma: Optimal power flow using teaching-learning-based optimization technique. Electric Power Systems Research *114* (2014), 49–59. DOI:10.1016/j.epsr.2014.03.032

[11] I. Chaari, A. Koubaa, H. Bennaceur, S. Trigui, and K. Al-Shalfan: A hybrid ACO-GA algorithm for robot path planning. In: IEEE Congress on Evolutionary Computation, Brisbane 2012, pp. 1–8.

[12] S. H. Chia, K. L. Su, J. H. Guo, and C. Y. Chung: Ant colony system based mobile robot path planning. In: IEEE International Conference on Genetic and Evolutionary Computing, China 2010, pp. 210–213.

[13] Y. Dai, J. Yu, C. Zhang, B. Zhan, and X. Zheng: A novel whale optimization algorithm of path planning strategy for mobile robots. Appl. Intell. *53* (2023), 10843–10857. DOI:10.1007/s10489-022-04030-0

[14] Z. Duraklı and V. Nabiyev: A new approach based on bezier curves to solve path planning problems for mobile robots. J. Comput. Sci. *58* (2022), 101542. DOI:10.1016/j.jocs.2021.101540

[15] M. Elhoseny, A. Tharwat, and A. E. Hassanien: Bezier curve based path planning in a dynamic field using modified genetic algorithm. J. Comput. Sci. *25* (2018), 339–358.

[16] S. Feng, S. Zhang, M. Xu, and G. Deng: Parallel navigation for 3-D autonomous vehicles. Kybernetika *59* (2023), 4, 592–611. DOI:10.14736/kyb-2023-4-0592

[17] N. Gasilov, M. Dogan, and V. Arici: Two-stage shortest path algorithm for solving optimal obstacle avoidance problem. IETE J. Res. *57* (2011), 3, 278–285. DOI:10.4103/0377-2063.83650

[18] B. C. Guevara: An Overview of the Class of Rapidly-Exploring Random Trees. M.Sc. Thesis, Utrecht University 2018.

[19] M. S. Güzel, M. Kara, and M. S. Beyazkilic: An adaptive framework for mobile robot navigation. Adaptive Behavior *25* (2017), 1, 30–39. DOI:10.1177/1059712316685875

[20] M. A. Hossain and I. Ferdous: Autonomous robot path planning in dynamic environment using a new optimization technique inspired by bacterial foraging technique. In: International Conference on Electrical Information and Communication Technology (EICT), Bangladesh 2014, pp. 1–6. DOI:10.1155/2014/904374

[21] J. H. Holland: Adaptation in natural and artificial systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence. MIT Press, 1992.

[22] A. T. Ismail, A. Sheta and A. Al-Weshah: A mobile robot path planning using genetic algorithm in static environment. J. Computer Sci. *4* (2008), 4, 341–344. DOI:10.3844/jcssp.2008.341.344

[23] A. Kroll and S. Soldan: Survey results on status, needs and perspectives for using mobile service robots in industrial applications. In: 11th International Conference on Control Automation Robotics and Vision, Singapore 2010, pp. 621–626. DOI:10.1109/icarcv.2010.5707243

[24] A. Kumar, G. Ahmad, and M. Shahid: Portfolio selection strategy: A teaching–learning-based optimization (TLBO) approach. In: Proc. International Joint Conference on Advances in Computational Intelligence, Singapore 2023, pp. 553–564. DOI:10.1007/978-981-99-1435-7_46

[25] C. Li, X. Huang, J. Ding, K. Song, and S. Lu: Global path planning based on a bidirectional alternating search A* algorithm for mobile robots. Comput. Industr. Engrg. *168* (2022), 108123. DOI:10.1016/j.cie.2022.108123

[26] J. Li, Y. Chen, X. Zhao, and J. Huang: An improved DQN path planning algorithm. J. Supercomput. *78* (2022) 616–639. DOI:10.1007/s11227-021-03878-2

[27] X. Li, G. Zhao, and B. Li: Generating optimal path by level set approach for a mobile robot moving in static/dynamic environments. Appl. Math. Modell. *85* (2020), 210–230. DOI:10.1016/j.apm.2020.03.034

[28] Y. Li, Z. Huang, and Y. Xie: Path planning of mobile robot based on improved genetic algorithm. In: 3rd International Conference on Electron Device and Mechanical Engineering (ICEDME), China 2020, pp. 691–695.

[29] Y. Li, W. Wei, Y. Gao, D. Wang, and Z. Fan: PQ-RRT*: An improved path planning algorithm for mobile robots. Expert Systems Appl.*152* (2020), 113425. DOI:10.1016/j.eswa.2020.113425

[30] J. Liu, J. Yang, H. Liu, X. Tian, and M. Gao: An improved ant colony algorithm for robot path planning. Soft Comput. *21* (2017), 5829–5839. DOI:10.1007/s00500-016-2161-7

[31] J. Liu, X. Wei, and H. Huang: An improved grey wolf optimization algorithm and its application in path planning. IEEE Access *9* (2021), 121944–121956. DOI:10.1109/ACCESS.2021.3108973

[32] E. S. Low, P. Ong, C. Y. Low, and R. Omar: Modified q-learning with distance metric and virtual target on path planning of mobile robot. Expert Systems Appl. *199* (2022), 117191. DOI:10.1016/j.eswa.2022.117191

[33] P. G. Luan and N. T. Thinh: Hybrid genetic algorithm based smooth global-path planning for a mobile robot. Mechanics Based Design Structures Machines *51* (2023), 1758–1774. DOI:10.1080/15397734.2021.1876569

[34] S. Luo, M. Zhang, Y. Zhuang, C. Ma, and Q. Li: A survey of path planning of industrial robots based on rapidly exploring random trees. Frontiers Neurorobotics *17* (2023). DOI:10.3389/fnbot.2023.1268447

[35] D. Lyu, Z. Chen, Z. Cai, and S. Piao: Robot path planning by leveraging the graph-encoded floyd algorithm. Future Generation Computer Systems *122* (2021), 204–208. DOI:10.1016/j.future.2021.03.007

[36] J. Ma, Y. Liu, S. Zang, and L. Wang: Robot path planning based on genetic algorithm fused with continuous Bezier optimization. Comput. Intell. Neurosci. (2020). DOI:10.1155/2020/9813040

[37] A. Maoudj and A. Hentout: Optimal path planning approach based on q-learning algorithm for mobile robots. Appl. Soft Comput. *97* (2020), 106796. DOI:10.1016/j.asoc.2020.106796

[38] C. Miao, G. Chen, C. Yan, and Y. Wu: Path planning optimization of indoor mobile robot based on adaptive ant colony algorithm. Comput. Industr. Engrg. *156* (2021), 107230. DOI:10.1155/2021/6853809

[39] S. Mirjalili: Genetic Algorithm, Evolutionary Algorithms and Neural Networks. Springer Cham 2019, 43–55.

[40] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al.: Human-level control through deep reinforcement learning. Nature *518* (2015), 529–533. DOI:10.1038/nature14236

[41] H. F. Naji, P. Kullu, and S. Emrah Amrahov: An augmented reality-based system with sound effects for teaching english in primary school. Educat. Inform. Technolog. (2023), 1–13. DOI:10.1007/s10639-023-12350-y

[42] N. Kartli, E. Bostanci and M. S. Guzel: A new algorithm for the initial feasible solutions of fixed charge transportation problem. In: 7th International Conference on Computer Science and Engineering (UBMK), IEEE, 2022, pp. 82–85. DOI:10.1109/ubmk55850.2022.9919524

[43] N. Kartli, E. Bostanci and M. S. Guzel: A new algorithm for optimal solution of fixed charge transportation problem. Kybernetika *59* (2023), 1, 45–63. DOI:10.15625/2615-9023/18488

[44] V. Rajinikanth, S. C. Satapathy, S. L. Fernandes, and S. Nachiappan: Entropy based segmentation of tumor from brain mr images – a study with teaching learning based optimization. Pattern Recognit. Lett. *94* (2017), 87–95. DOI:10.1016/j.patrec.2017.05.028

[45] R. V. Rao, V. J. Savsani, and D. Vakharia: Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. Computer-aided Design *43* (2011), 303–315. DOI:10.1016/j.cad.2010.12.015

[46] A. D. Sabiha, M. A. Kamel, and E. Said,and W. H. Hussein: Path planning algorithm based on teaching-learning-based-optimization for an autonomous vehicle. Communications *24* (2022), C33–C42. DOI:10.26552/com.C.2022.2.C33-C42

[47] S. Y. Sert, Y. Ar, and G. E. Bostanci: Evolutionary approaches for weight optimization in collaborative filtering-based recommender systems. Turkish J. Electr. Engrg. Computer Sci. *27* (2019), 3, 2121–2136. DOI:10.3906/elk-1812-175

[48] D. H. Shin and A. Ollero: Mobile robot path planning for fine-grained and smooth path spcification. J. Robotic Syst. *12* (1995), 7, 491–503. DOI:10.1002/rob.4620120704

[49] H. Tang, B. Fang, R. Liu, Y. Li, and S. Guo: A hybrid teaching and learning-based optimization algorithm for distributed sand casting job-shop scheduling problem. Appl. Soft Comput. *120* (2022), 108694. DOI:10.1016/j.asoc.2022.108694

[50] H. Tu, Y. Deng, Q. Li, M. Song, and X. Zheng: Improved RRT global path planning algorithm based on bridge test. Robotics Autonomous Systems *171* (2024), 104570. DOI:10.1016/j.robot.2023.104570

[51] J. Wang, W. Chi, C. Li, C. Wang, and M. Q. H. Meng: Neural RRT*: Learning-based optimal path planning. IEEE Trans. Automat. Sci. Engrg. *17* (2020), 1748–1758. DOI:10.1109/tase.2020.2976560

[52] W. Wang, J. Li, Z. Bai, Z. Wei, and J. Peng: Towards optimization of path planning: An RRT*-ACO algorithm. IEEE Access (2024). DOI:10.1109/access.2024.3359748

[53] Z. Wu, W. Fu, R. Xue, and W. Wang: A novel global path planning method for mobile robots based on teaching-learning-based optimization. Information *7* (2016), 39. DOI:10.3390/info7030039

[54] L. Xu, M. Cao, and B. Song: A new approach to smooth path planning of mobile robot based on quartic bezier transition curve and improved pso algorithm. Neurocomputing *473* (2022), 98–106. DOI:10.1016/j.neucom.2021.12.016

[55] H. B. Yildirim, K. Kullu, and S. Emrah Amrahov: A graph model and a three-stage algorithm to aid the physically disabled with navigation. Universal Access Inform. Soc. (2023), 1–11. DOI:10.1007/s10209-023-00981-4

[56] X. Yuan, X. Yuan, and X. Wang: Path planning for mobile robot based on improved bat algorithm. Sensors *21* (2021), 4389. DOI:10.3390/s21134389

[57] L. Zhang, H. Min, H. Wei, and H. Huang: Global path planning for mobile robot based on A* algorithm and genetic algorithm. In: IEEE International Conference on Robotics and Biomimetics (ROBIO), China 2012, pp. 1795–1799.

[58] T. W. Zhang, G. H. Xu, X. S. Zhan, and T. Han: A new hybrid algorithm for path planning of mobile robot. J. Supercomput. *78* (2022), 4158–4181. DOI:10.1007/s11227-021-04031-9

[59] Y. Zhang, Z. Jin, and Y. Chen: Hybrid teaching-learning-based optimization and neural network algorithm for engineering design optimization problems. Knowledge-Based Systems *187* (2020), 104836. DOI:10.1016/j.knosys.2019.07.007

[60] Z. Zhang, R. He, and K. Yang: A bioinspired path planning approach for mobile robots based on improved sparrow search algorithm. Adv. Manufactur. *10* (2022), 114–130. DOI:10.1007/s40436-021-00366-x

Emad Hazrati Nejad, Computer Engineering Department, Ankara University, 06830, Ankara. Turkey.
e-mail: hazratiemad@gmail.com

Sevgi Yigit-Sert, Computer Engineering Department, Ankara University, 06830, Ankara. Turkey.
e-mail: syigit@ankara.edu.tr

Şahin Emrah Amrahov, Computer Engineering Department, Ankara University, 06830, Ankara. Turkey.
e-mail: emrah@eng.ankara.edu.tr