applied sciences

MDPI

Article

Mobile Robot Path Planning with a Non-Dominated Sorting Genetic Algorithm

Yang Xue

Department of Mechanics, Tianjin University, Tianjin 300350, China; xueyang216@tju.edu.cn

Received: 18 October 2018; Accepted: 12 November 2018; Published: 15 November 2018

Abstract: In many areas, such as mobile robots, video games and driverless vehicles, path planning has always attracted researchers' attention. In the field of mobile robotics, the path planning problem is to plan one or more viable paths to the target location from the starting position within a given obstacle space. Evolutionary algorithms can effectively solve this problem. The non-dominated sorting genetic algorithm (NSGA-II) is currently recognized as one of the evolutionary algorithms with robust optimization capabilities and has solved various optimization problems. In this paper, NSGA-II is adopted to solve multi-objective path planning problems. Three objectives are introduced. Besides the usual selection, crossover and mutation operators, some practical operators are applied. Moreover, the parameters involved in the algorithm are studied. Additionally, another evolutionary algorithm and quality metrics are employed for examination. Comparison results demonstrate that non-dominated solutions obtained by the algorithm have good characteristics. Subsequently, the path corresponding to the knee point of non-dominated solutions is shown. The path is shorter, safer and smoother. This path can be adopted in the later decision-making process. Finally, the above research shows that the revised algorithm can effectively solve the multi-objective path planning problem in static environments.

Keywords: mobile robot; static environments; path planning; multi-objective optimization; NSGA-II; evolutionary operators

1. Introduction

Path planning (PP) has been fundamental in many areas in recent decades, for instance mobile robots [1–3], unmanned surface vehicles (USVs) [4,5], wireless sensor networks (WSNs) [6–8] and video games [9]. For mobile robots, path planning is devised to find one or more feasible routes from the initial location to the target in a given workspace. The related algorithms in the field of path planning are reviewed in [10,11]. In [12], several path planning and navigation algorithms commonly employed in the domain of unmanned aerial vehicles (UAVs) were studied. The commonly-used methods are the probabilistic roadmap method (PRM) [13], the artificial potential field method (ARF) [14], the rapidly-exploring random tree method (RRT) [15], A* and its variants [16,17], and so on. In [18], the fast marching method (FMM) was applied to the path planning problem. Moreover, the research revealed that the algorithm can successfully attain the collision-free shortest route in many static environments. Most conventional approaches attain available paths and seldom optimize several objectives concurrently.

However, optimization problems in most disciplines should study multiple objectives simultaneously, not just a single one. For example, in the product design process, the cost and quality of the product are often contradictory, that is and decrease in the price of the product and the quality has to decrease, and vice versa. How to decrease the cost of the product and enhance the quality comprise a dual-objective optimization problem. Path planning problems are no exception. In this study, three objectives are presented. Path length is related to the operation time of the mobile robot.

Path safety represents the length from the path to the nearest obstacle. Path smoothness indicates the degree of bending of the path. In general, at least two objectives are conflicting.

At present, based on the complexity of the path planning problem, this issue is identified as NP-hard [19]. Multiple competing goals need to be considered concurrently. Subsequently, common multi-objective weighting methods are employed to solve problems [20,21]. The principle of the weighting method is to set the weighting factors of the objectives and combine the targets into a new one. However, because there is no apparent measurable relationship between these objectives, it is difficult to set their weighting factors. In other words, the transformation process itself is also a multi-objective optimization problem. Therefore, the weighting method is not proper. On the other hand, some researchers extended the deterministic heuristic A* algorithm to multi-objective cases [22–24]. In these algorithms, the heuristic function of any node in the graph search space is a vector.

In later years, evolutionary algorithms (EA) were universally employed for path planning. A practical framework of multi-objective evolutionary algorithm (MOEA) was designed in [25]. Besides, several practical evolution operators were presented. To further advance the computational efficiency, in the initial step, some individuals were generated by the Dijkstra algorithm. In [26], the traditional method was first employed to generate the roadmap in the workspace, and then, the Hopfield neural network was applied to improve path length and safety. In [27], the modified rapidly-exploring random tree was presented to obtain the path. Then, the path length and smoothness were improved based on the neural network curve post-processing strategy. In [28], the Q-learning method was developed to optimize the path. A fast two-stage ant colony optimization algorithm was shown in [29]. This algorithm contained two phases: map preprocessing and ant colony optimization. In the map preprocessing, they calculated the minimum number of steps from all free nodes to the target node. Next, the path length was optimized by the modified ant colony optimization. In [30], the modified tabu search algorithm was designed for the optimal path length in grid environments. In [31], path length and smoothness were enhanced by two multi-objective memetic algorithms. In [32], an intelligent water drop algorithm was proposed to increase the length and safety. An available path was found by the artificial bee colony algorithm in [33], then the length and smoothness of the obtained path were optimized by evolutionary programming. In [34], a model of path length and danger degree was solved by the improved particle swarm algorithm. The degree of risk and path length were minimized by particle swarm optimization (PSO) in unknown circumstances [35]. In [36], the chaotic particle swarm optimization algorithm was used to enhance the control points of the Bezier curve to reach a short and smooth route. The hierarchical global path planning method was introduced in [37]. The method designed a three-stage structure to obtain an optimal route. First, a free geometric configuration space was determined with the triangular decomposition method, and then based on the configuration space, the Dijkstra algorithm was utilized to obtain a viable route. Finally, constrained particle swarm optimization was designed to optimize length and smoothness. Similar hierarchical strategies could be found in [38]. Individuals within the initial population were generated by the surrounding point set algorithm. Then, the length and smoothness were enhanced by the particle swarm optimization algorithm.

In this study, the multi-objective path planning is solved with the improved NSGA-II. The work is shown below:

- The framework of the improved NSGA-II is introduced. Several practical evolutionary operators are presented to enhance the feasibility of the route and optimize three objectives (length, smoothness and safety). They can enhance the local search capabilities of the improved NSGA-II.
- The parameters in the algorithm are systematically studied. The results show that larger population sizes, larger numbers of generations and high operator probabilities are indispensable in complex environments.

- The improved NSGA-II is tested with an existing evolutionary algorithm and different quality metrics. The comparisons show that the non-dominated solutions received via the improved NSGA-II have good characteristics.
- The path corresponding to the knee point of non-dominated solutions is shown. The route is shorter, smoother and safer.

The rest of the work is designed below. The associated work is introduced in Section 2. Section 3 defines environmental modeling, representation of the path and the objectives that need to be optimized. Section 4 details NSGA-II and evolutionary operators. The parametric study is presented in Section 5. Section 6 exhibits the comparison results. Lastly, the conclusions are displayed in Section 7.

2. Related Work

Since the advent of the genetic algorithm, it has been favored by many scholars due to its simple operation process and powerful search ability [39,40]. Genetic algorithms demonstrate robust optimization performance in various areas [41–43]. In path planning, genetic algorithms have also received extensive attention. In [44], the genetic algorithm was designed to shorten the path length in the grid space. In [45], the efficiency of probabilistic roadmap (PRM) and genetic algorithm (GA) for attaining a viable route was studied. NSGA-II was employed to optimize path length and clearance in the grid environments [46]. In [47], multiple techniques of representing a path in the grid environments were presented. Moreover, NSGA-II was used to improve length, smoothness and safety. A new selection operator was designed to avoid falling into a local trap and premature convergence [48]. Then, the adaptive method based on GA was designed to optimize path length. In [49], control points of the Bezier curve were enhanced by NSGA-II to reach the Pareto-optimal solutions. In [50], an efficient initialization technique based on directed acyclic graphs was proposed. This technique can provide multiple feasible minimum paths for the genetic algorithm, which enhances the computational efficiency of the entire genetic algorithm. In [51], a viable route was given with the genetic algorithm and then smoothed by the piecewise cubic Hermite interpolation polynomial. In [52], the environment was converted with a matrix-binary code-based genetic algorithm (MGA). Then, the navigation time and the path length were optimized. Control points of the Bezier curve were optimized with an improved genetic algorithm in [53]. Then, the optimum smooth path could be selected by choosing these control points. In [54], an improved crossover operator was designed in static environments. In [55], NSGA-II was used to improve the clearance and smoothness of the path by optimizing the three parameters obtained by the potential field method. Moreover, the position of the virtual obstacle was redefined for the case where the end point of the path was farther away from the obstacle. This can help the robot safely drive away from obstacles. Finally, a method for identifying obstacles that affect the robot in cluttered environments was presented. As far as we know, genetic algorithms show great vitality in path planning problems. However, due to the existence of obstacles, after using the traditional operators (crossover, mutation), the newly-generated individuals are generally no longer feasible paths. More practical evolutionary operators are needed to further enhance evolution efficiency.

In this study, the path planning problem is resolved with the improved NSGA-II in static environments. Multiple objectives are considered. More practical evolutionary operators are presented. In the remainder of this article, the improved algorithm is proposed in detail.

3. Path Planning Problem

Path planning is to find one or more available paths in the workspace. In this article, statically known environments are considered. That is, all obstacles within the workspace are static, and their location information is entirely known to the robot. The starting location and target of the robot are signified as S and T, respectively. Then, path planning is devised to find one or more paths from S to T that do not collide with obstacles. Next, the environment modeling, path representation and the three

optimization objectives are defined. In this work, two-dimensional space is adopted. Any obstacle is supposed to be an arbitrary polygonal shape. Moreover, a polygon robot can be converted into a single point by using the Minkowski sums in the field of computational geometry [56]. A path is signified with $p = [S = p_0, p_1, p_2, ..., p_n, p_{n+1} = T]$. p_i is the i-th rotation points (RPs). p_i and p_{i+1} are connected by straight segments. The representation of a path is shown in Figure 1. The coordinate of p_i is denoted as (X_i, Y_i).

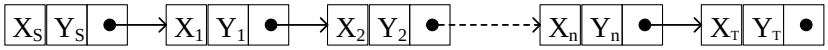

Figure 1. The representation of a path is given.

3.1. Objectives

In this paper, three objectives are introduced: smoothness, safety and length. The energy loss is associated with path smoothness and length. In addition to considering energy loss, the safety of the driving path cannot be ignored. When the distance of the robot from the obstacles during driving is less than the safe range of the sensors, this can be a critical situation, even damaging the robot and the items that are seen as obstacles. Therefore, the farther away from the nearest obstacle, the safer the route. Considering the above factors, the three objectives are designed to enhance the driving route of the robot. This is the multi-objective optimization problem. For convenience, it can be converted into a minimization problem. Next, the mathematical definition of these objectives is presented.

3.1.1. Path Length

The line segment formed by any two points in the space can be calculated based on the Euclidean distance. Thus, for two consecutive rotation points ($p_i = (x_i, y_i)$, $p_{i+1} = (x_{i+1}, y_{i+1})$), the line segment of the points can be computed. Then, the path length can be given by summing all the ordered line segments. The mathematical expression is presented in Equation (1):

$$\begin{cases} dis(p_i, p_{i+1}) = \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2} \\ Length(p) = \sum_{i=0}^{n} dis(p_i, p_{i+1}). \end{cases} \tag{1}$$

3.1.2. Path Smoothness

Path smoothness denotes the degree of sleekness of the path. The smoother the route, the less energy the robot consumes as it travels along the way. In this paper, it is defined by the average turning angle of the path. The corner formed by two consecutive line segments can be determined. Then, the average turning angle of the route can be calculated by Equation (2).

$$\begin{cases} Angle[p_i, p_{i+1}, p_{i+2}] = \pi - \cos^{-1}\left(\frac{(x_{i+1}-x_i)(x_{i+2}-x_{i+1})+(y_{i+1}-y_i)(y_{i+2}-y_{i+1})}{dis(p_i,p_{i+1})\times dis(p_{i+1},p_{i+2})}\right) \\ Smoothness(p) = \frac{1}{n}\sum_{i=0}^{n-1}\{Angle[p_i, p_{i+1}, p_{i+2}]\}. \end{cases} \tag{2}$$

3.1.3. Path Safety

Firstly, the secure interval is defined. It is the minimum distance of the path from the nearest obstacle. The barrier set ($O = O_1, O_2, ..., O_m$) contains all the obstacles in the space; the number is m. $MinDis(\overline{p_i p_{i+1}}, O_j)$ is the minimum length between the line segment ($\overline{p_i p_{i+1}}$) and the obstacle (O_j). Thus, $\underset{0\le i\le n}{\text{Min}}\ \underset{1\le j\le m}{\text{Min}}\{MinDis(\overline{p_i p_{i+1}}, O_j)\}$ is the safe interval. For convenience, the maximum safety interval problem is converted to a minimum problem. The general operation is to add a negative sign

before the safe distance to convert the function to a negative number. The safety objective of the path can be expressed by Equation (3):

$$Safety(p) = -\underset{0\le i\le n1\le j\le m}{\text{Min}}\underset{}{\text{Min}}\{MinDis(\overline{p_i p_{i+1}}, O_j)\}. \tag{3}$$

At this time, a model of path planning is presented below:

$$\begin{cases} p = [S = p_0, p_1, p_2, ..., p_n, p_{n+1} = T] \\ \text{Minimize } f_1(p) = Length(p) \\ \text{Minimize } f_2(p) = Smoothness(p) \\ \text{Minimize } f_3(p) = Safety(p). \end{cases} \tag{4}$$

In the above model, the only constraint is that the path does not collide with the obstacle, but it is allowed to reach the edge of the obstacle. If the path intersects the obstacle, then it is not feasible. The edge of the workspace is also regarded as an obstacle.

In general, at least two goals are contradictory. That is to say, one or more other objectives have to be sacrificed while increasing one objective. This problem is a typical multi-objective optimization problem. There is no longer one best solution, but a set of solutions. Moreover, these solutions are non-dominated. In other words, when one or more objectives of one solution are better than the other, there must be at least one or more poor objectives. In recent years, some surpassing evolutionary algorithms have emerged for solving path planning problems. However, due to the complexity and practicality of this issue, many scholars remain interested.

4. NSGA-II for the Multi-Objective Path Planning Problem

NSGA-II is one of the most popular multi-objective genetic algorithms [40]. It has the advantage of fast running speed and good convergence of solutions. Moreover, it has become the benchmark for evaluating numerous optimization algorithms. NSGA-II is the second generation non-dominated sorting genetic algorithm, and its improvements mainly include three aspects:

- A fast non-dominated sorting algorithm is designed, which stratifies the population according to the non-domination level of the individuals. Individuals within the same layer are non-dominated.
- The crowding distance of individuals in the same layer is calculated. Then, the individuals with higher values are preferentially selected, so that the same layer individuals can be more evenly distributed in the objective space to preserve the population diversity.
- The elite strategy combines the parental population with the offspring population. First, infeasible solutions within the collection are eliminated. Secondly, the remaining individuals are ranked according to the non-dominated sorting algorithm. From the low to high levels, the individuals in each layer are placed sequentially in the new population until the number of individuals exceeds the capacity of the population. Finally, according to the crowding distance metric, the individuals in the specific layer are placed into the new population in descending order until the population is full. The elite strategy preserves all the good individuals in the parental and offspring populations, thereby improving the accuracy and robustness of the optimization results.

Due to the above characteristics, NSGA-II has strong optimization capabilities. Therefore, the path planning is solved with NSGA-II. However, individuals formed by the traditional operators are generally not feasible paths. It is not enough to rely solely on traditional operators. This requires a larger population capacity and more evolutionary algebra. In the work, several practical evolutionary operators are introduced to solve this problem. These evolutionary operators are more purposeful to reduce the length and improve the smoothness and safety. Next, NSGA-II and all the operators are described in detail.

4.1. Flowchart

The process of NSGA-II is as follows:

1 The population (*POP*) is initialized.
2 In the population (*POP*), the objectives of each individual are evaluated.
3 An empty set (*NDPOP*) is generated to save the found feasible non-dominated individuals.
4 While the termination condition has not been reached, DO:

(1) Two empty populations (*NEWPOP*, *POPc*) are generated.
(2) The individuals are chosen from the population (*POP*) by the selection operator and put into the population (*POPc*).
(3) Individuals within the population (*POPc*) are paired. Then, the crossover operator is performed on the two individuals. Next, new individuals are generated and stored in the population (*NEWPOP*).
(4) For any individual in the population (*POPc*), the practical operators are sequentially executed. Then, the generated individuals are put into the population (*NEWPOP*).
(5) Each individual within the population (*NEWPOP*) is evaluated, and the feasible solutions are put into the set (*NDPOP*). Then, the non-dominated individuals of the set (*NDPOP*) are retained.
(6) Parental and children populations (*POP*) and (*NEWPOP*) are merged. Individuals in the combined population are classified according to the individual feasibility: feasible solutions' set and infeasible solutions' set. Then, the non-dominated sorting and crowding distance metric are performed on the two sets respectively. Next, individuals are picked from the two sets, and a new population (*POP*) is formed. The size of the population (*POP*) remains the same.
(7) The loop counter is incremented by one.

In the entire algorithm, a viable individual is a collision-free path. Steps 1–3 are the initialization process. The initial population (*POP*) is generated and evaluated. An empty set (*NDPOP*) is then made to store the found non-dominated solutions. This set is only used for storage and does not participate in the evolution of the population. Step 4 is the core of the algorithm, and the total iterative process is completed in this step until the termination condition is reached.

In each iteration, two empty populations (*POPc* and *NEWPOP*) are created. The population (*POPc*) is used to store the selected individuals of the parental population (*POP*). The population (*NEWPOP*) stores the children individuals. First, the selection operator is applied to choose individuals from the population (*POP*), and the selected individuals are stored in the population (*POPc*). Individuals of the population (*POPc*) are then paired. New individuals are generated by using the crossover operator and then placed in the child population (*NEWPOP*). Next, a series of operators is executed for each within the population (*POPc*). When each operator is applied, a new individual is generated and stored in the child population (*NEWPOP*). After the individuals in the population (*POPc*) are treated with the evolutionary operators, the individuals in the children population (*NEWPOP*) are evaluated, and the feasible individuals are put into the set (*NDPOP*). Then, individuals of the population (*NDPOP*) are updated; in other words, the non-dominated solutions of the population (*NDPOP*) are retained; other individuals are excluded.

Then, the elite strategy is executed. The parental population (*POP*) and the offspring population (*NEWPOP*) are merged, and the individuals in the merged set are classified according to individual feasibility: feasible solutions' set and infeasible solutions' set. Then, the non-dominated sorting and crowding distance metric are performed on the two sets, respectively. Then, individuals are picked from the two sets to form a new population (*POP*). In this step, the infeasible individuals are not

entirely discarded, but some better individuals are preserved. Besides, the size of the population (*POP*) is unchanged. Then, the loop counter is incremented by one.

Step 4 is iterated until the end condition is satisfied. At this point, the entire algorithm ends, and the set (*NDPOP*) is output. Furthermore, the operators are introduced. Each operator acts on an individual, and a new individual is generated and then stored in the child population (*NEWPOP*). Besides the traditional operators, these evolutionary operators can improve the length, smoothness and safety of the path more effectively. Next, these operators and the initialization process are presented in detail.

4.2. Selection Operator

The selection operator is to elect individuals from the parental population (*POP*). The constrained tournament selection operator is applied. It selects two solutions at a time, then the tournament between them is run, and the winner participates in the succeeding evolution. For two chosen individuals, if one is not attainable and the other is available, the victor is the viable one. If none are feasible or viable, the non-dominated regulation is applied to pick the victor. If both are non-dominated, the individual with a larger crowding distance is picked as the winner. Finally, the winners are stored in the population (*POPc*).

4.3. Crossover Operator

For two randomly-selected individuals in the population (*POPc*), the crossover operator is used to form new individuals by swapping parts of the individuals. The newly-generated individuals are stored in the child population (*NEWPOP*). Suppose that $p = [S, p_1, p_2, ..., p_{i-1}, p_i, ..., p_j, p_{j+1}, ..., p_n, T]$ and $q = [S, q_1, q_2, ..., q_{k-1}, q_k, ..., q_m, q_{m+1}, ..., q_{n'}, T]$ are two randomly-chosen paths, p_i and p_j are two randomly-selected rotation points (RPs) on path (p). Similarly, q_k and q_m are two randomly-selected rotation points on path (q). The line segments connecting p_i with p_j and the line segments linking q_k with q_m are exchanged. Finally, two new paths ($p' = [S, p_1, p_2, ..., p_{i-1}, q_k, ..., q_m, p_{j+1}, ..., p_n, T]$, $q' = [S, q_1, q_2, ..., q_{k-1}, p_i, ..., p_j, q_{m+1}, ..., q_{n'}, T]$) are generated. Figure 2 shows the result of exchanging line segments with the crossover operator.

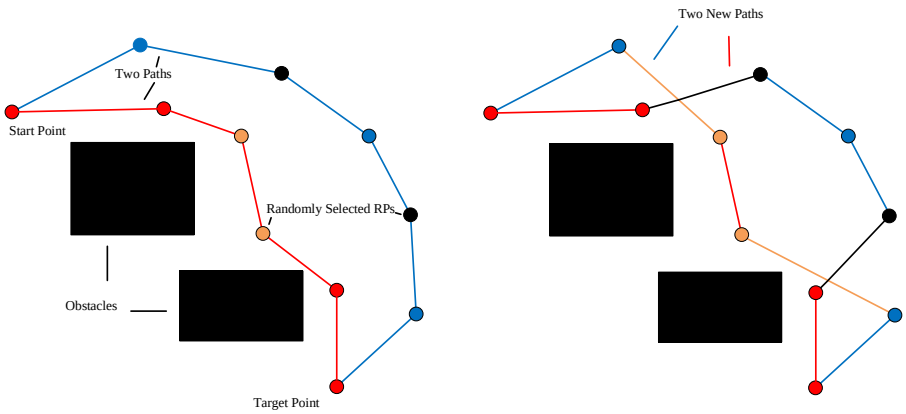

Figure 2. The crossover operator is shown. In the left sub-figure, the yellow circles are the two rotation points randomly selected on the path. Similarly, the black circles are the two rotation points on the other path. The line segments between the yellow circles are exchanged with the line segments between the black circles. Finally, two new paths are generated and displayed in the right sub-figure.

4.4. Invalid Solution Operator

The invalid solution operator is used to convert the path ($p = [S, p_1, p_2, ..., p_i, p_{i+1}, ..., p_n, T]$) that collides with the obstacles ($O = O_1, O_2, ..., O_m$) into a feasible path. The design process of this operator is expressed as follows:

Along the path, it is determined in turn whether the line segment ($\overline{p_i p_{i+1}}$) of the path collides with the obstacles. If so, the intersection points and the obstacles ($O(i)$) that collide with the line segment are found.

Then, the vertices of the obstacles ($O(i)$), the two rotation points (p_i, p_{i+1}) and the intersection points form the vertices of the graph. Any two vertices within the set of vertices are linked to form the set of edges. Meanwhile, the edges that collide with all obstacles (O) are deleted. Then, based on the vertices and edges of the graph, the Dijkstra algorithm can be applied to attain a feasible shortest path ($q = [p_i, q_1, q_2, ..., q_k, p_{i+1}]$) from p_i to p_{i+1}. Next, the line segment ($\overline{p_i p_{i+1}}$) of path (p) can be replaced with the path (q), and the path ($p' = [S, p_1, p_2, ..., p_{i-1}, q, p_{i+1}, ..., p_n, T]$) is generated. Finally, the viable path (p') is made by repeating the above operation from $i = 0$ to n. The invalid solution operator is presented in Figure 3.

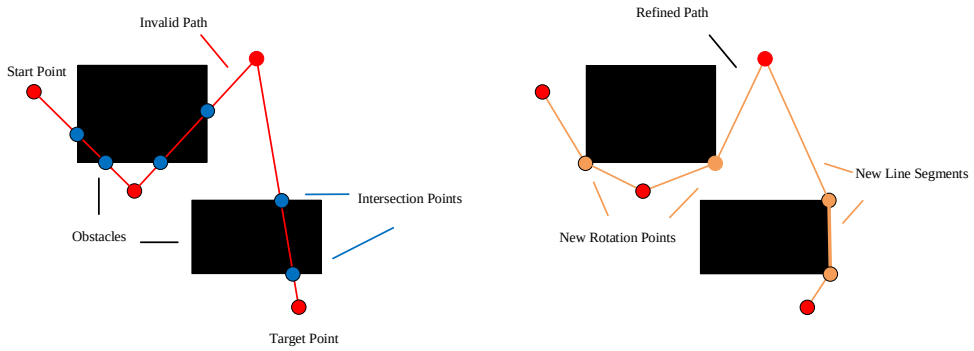

Figure 3. The invalid solution operator is presented. In the sub-figure on the left, the blue circles represent the intersection points of the path and the obstacle. The right sub-figure shows the new path generated by the Dijkstra algorithm.

4.5. Mutation Operator

The single point variation is used in this operator. A rotation point on the path is arbitrarily chosen and displaced by an arbitrary free point. It should be noted that the path after the mutation may be worse or better than before the variation. The mutation operator is displayed in Figure 4.

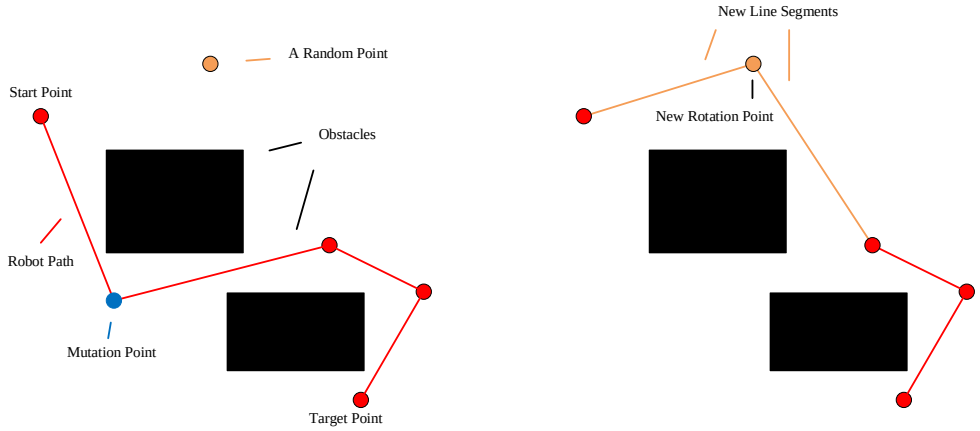

Figure 4. The mutation operator is presented. In the left sub-figure, a randomly selected rotation point (indicated by a blue circle) on the path is replaced by a free point (indicated by a yellow circle) randomly generated in the space. The resulting new path is shown in the right sub-figure.

4.6. Shortness Operator

The path is changed in arbitrary deleting manner in this operator. It randomly removes a rotation point. Figure 5 displays the shortness operator.

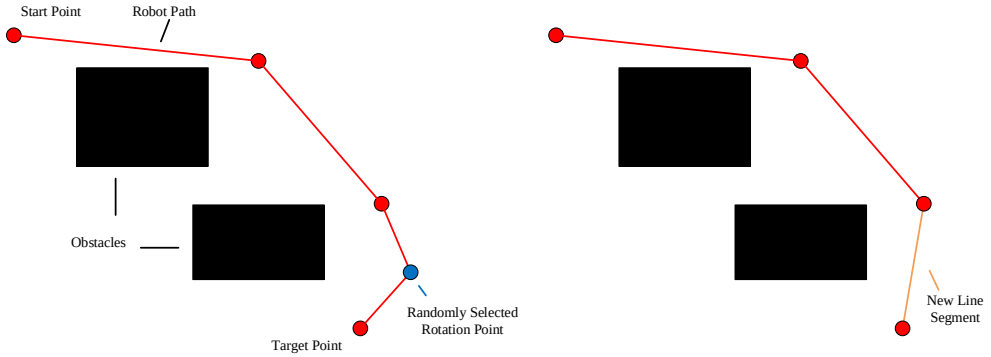

Figure 5. The shortness operator is presented. In the sub-figure on the left, a rotation point (indicated by a blue circle) on the path is randomly selected and removed. Then the new path is shown in the right sub-figure.

4.7. Insertion Operator

A single point insertion is adopted to design the insertion operator. A line segment on the path is randomly selected, and then, a random free point is inserted on the segment. The practice of the insertion operators is similar to the mutation operator. The random points in free space are used in both operators. The difference is that the mutation operator changes a rotation point and two line segments. The insertion operator increases a rotation point and converts one line segment into two new line segments. Figure 6 is an illustration of the insertion operator.

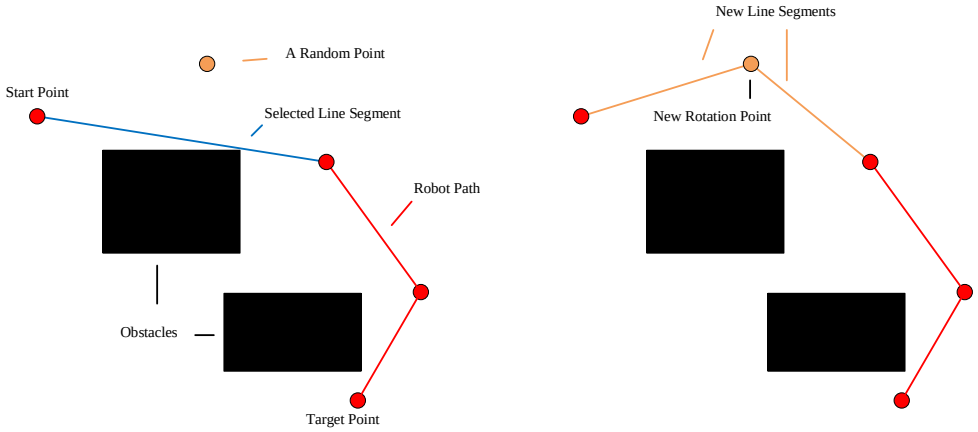

Figure 6. The insertion operator is shown. In the left sub-figure, a line segment on the path is randomly selected and indicated by a blue line. A free point is randomly generated in space and inserted into the blue line segment. The generated path is shown in the right sub-figure.

4.8. Safety Operator

The path safety is improved with the safety operator. On each line segment, the nearest point to the obstacles can be found. The point is named the critical point. Next, the region near the critical point can be meshed. Eight nearby lattices of the critical point are found. By comparing the safety distances of the lattices, the lattice with the largest safety distance is the winner. The center point of the lattice is chosen as a rotation point and then inserted into the line segment. The above method is

iterated until all line segments on the route have been considered. At this point, the safety operator is over. Figure 7 shows the safety operator.

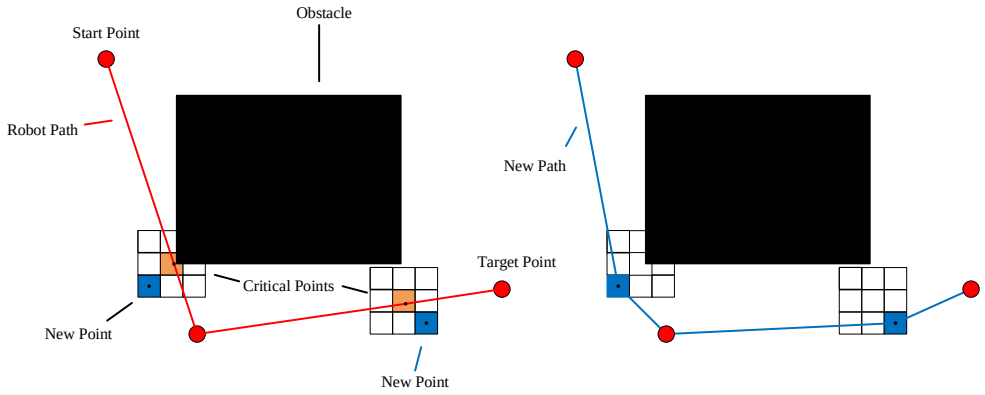

Figure 7. The safety operator is shown. In the left sub-figure, the critical point (indicated by a yellow circle) of each line segment on the path can be replaced by a point with a larger safe distance near the critical point. The sub-figure on the right shows the generated path.

4.9. Smoothness Operator

A smoother route can be found by the smoothness operator. In this operator, it is improved by decreasing the maximum turning angle formed by two consecutive segments on the path. Two free points are randomly generated on the two line segments and replace the intersection of the two line segments. Therefore, the maximum corner of the route is reduced. Figure 8 exhibits the smoothness operator.

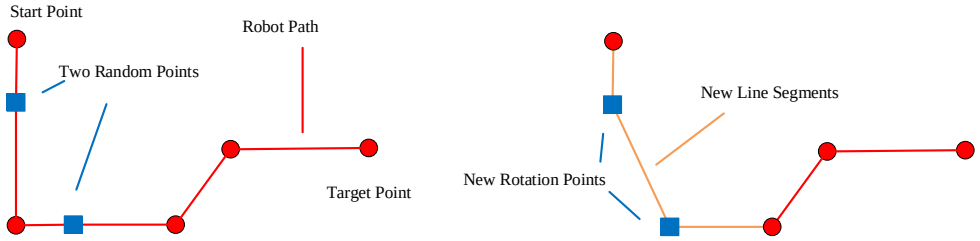

Figure 8. The smoothness operator is presented. In the left sub-figure, two free points are randomly generated on the two line segments where the maximum turning angle is located and are represented by blue squares. Then the intersection of the two segments is replaced with these two points. The right sub-figure shows the generated path.

4.10. Shortest Operator

The length of the path ($p = [S = p_0, p_1, p_2, ..., p_i, ..., p_n, p_{n+1} = T]$) can be minimized by the shortest operator. For the rotation point (p_i), another rotation point (p_j) is sequentially selected as a candidate from the target point (p_{n+1}) to the rotation point (p_{i+2}). If the new line segment ($\overline{p_i p_j}$) is feasible, other rotation points between the rotation points (p_i, p_j) are deleted. The above operation is repeated from $i = 0$ to $n - 1$. The shortest operator can remove extra rotation points to obtain the shortest path. Figure 9 presents this operator.

4.11. Position Update Operator

The idea of this operator comes from particle swarm optimization (PSO). Based on the positions of three continuing rotation points (p_{i-1}, p_i and p_{i+1}), the position of the middle point (p_i) is updated by Equation (5). This operator is shown in Figure 10.

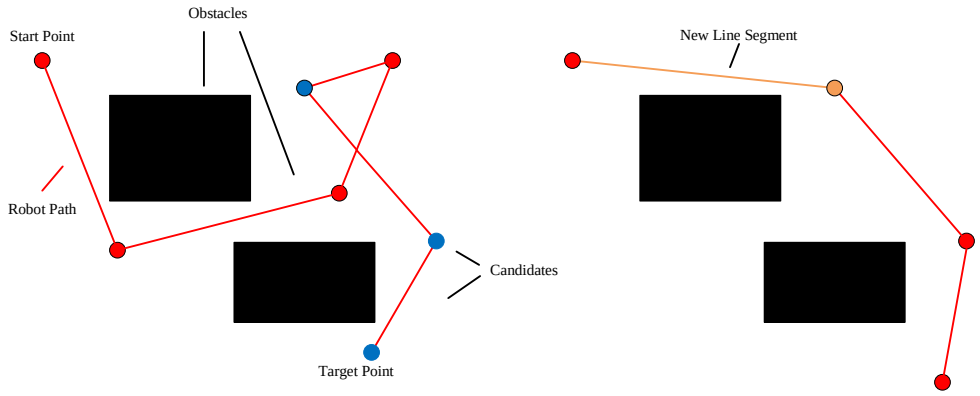

Figure 9. The shortest operator is presented. In the left sub-figure, blue circles indicate the sequentially selected candidate points. The resulting path is shown in the right sub-figure.

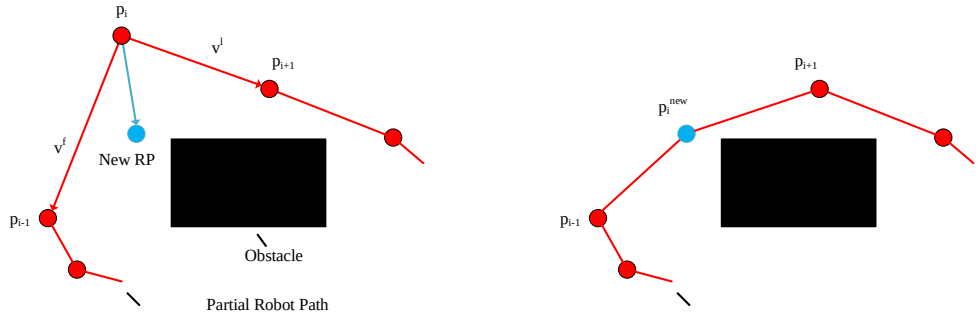

Figure 10. The position update operator is displayed. In the left sub-figure, some line segments of a path are shown. For three consecutive rotation points on the path, the new position of the middle point can be generated using Equation (5) and represented by a blue circle. The right sub-figure presents some line segments of the new path.

$$\begin{aligned} v^f &= p_{i-1} - p_i \\ v^l &= p_{i+1} - p_i \\ v &= r_1 \times v^f + r_2 \times v^l \\ p_i^{new} &= p_i + v. \end{aligned} \tag{5}$$

In Equation (5), the vector (v) should meet a restraint. The restraint is set to be $\pm 1\%$ of the workspace coordinates. In addition, the position (p_i^{new}) and the line segments ($\overline{p_{i-1}^{new} p_i^{new}}$ and $\overline{p_i^{new} p_{i+1}}$) should be feasible. The position (p_i^{new}) is adjusted continually by using Equation (6) until the above conditions are satisfied. Furthermore, r_1, r_2 and r_3 are set to random numbers from zero to one.

$$p_i^{new} = r_3 \times p_i + (1 - r_3) \times p_i^{new}. \tag{6}$$

4.12. Initialization

In general, the initialization process is essential. In the initial phase, the initial population is generated. Each within the population is a randomly generated path. That is, the rotation points on the path are random points in free space. The number of rotation points is set to a random number from one to three. Of course, these initial individuals are generally not feasible in a complicated space. Three objectives are designed to evaluate the path. They are the path length, smoothness and safety, respectively. Additionally, all parameters involved are displayed. The maximum number of generations and population size are 100 and 80 individually. The probabilities of selection, crossover

and shortest operators are 1, 0.8 and 0.1. The probabilities of the remaining operators are set to 0.5. Next, through the parameter study, the parameters of the genetic algorithm are reasonably set.

5. Parametric Study

In this section, the parameters studied are population size, the number of generations, crossover rate, mutation rate, insertion rate, invalid solution operation rate, safety operator rate, smoothness rate, position update operator rate, short operator rate and shortest operator rate. The likelihood of optimality ($Lopt$) is used to evaluate the non-dominated solutions obtained by different parameter selections [57]. Based on the same parameter setting, the algorithm is run n times independently. If the optimal solution is found m times, the likelihood of optimality ($Lopt$) of this set of parameters is the estimated probability m/n. This measurement technique was originally used to evaluate the parameter setting of the single-objective optimization algorithm. For multi-objective problems, the optimal solution can be understood as the optimal values of each objective. In the d-dimensional solution space, the $Lopt_k$ of any parameters setting k is the estimated probability $(m_1 + m_2 + ... + m_d)/(d \times n)$. In this work, Let n be 100. Figure 11A is considered. The starting and target points of this map is displayed in Table 1. These maps will be used to test the algorithm in detail in Section 6.

$$Lopt_k = \frac{m_1 + m_2 + ... + m_d}{d \times n}. \tag{7}$$

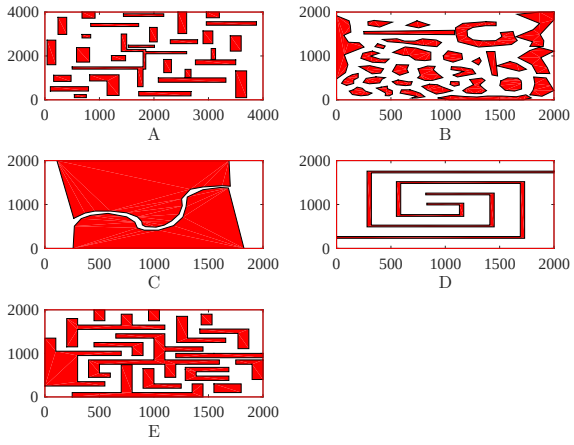

Figure 11. Five maps are used to test the algorithm. These maps have various obstacles.

Table 1. The start and destination of the maps.

Maps	Start	Destination
Figure 11A	(1500, 1200)	(3900, 3900)
Figure 11B	(400, 100)	(1450, 1450)
Figure 11C	(253.6, 403.5)	(1895.3, 1206.9)
Figure 11D	(1007.2, 1400)	(1908.6, 1008.1)
Figure 11E	(120, 120)	(1800, 1800)

Figures 12–22 present the effects of these parameters. Figure 12 presents the effect of population size. Other parameters remain the same as mentioned before. This figure shows that when the population size exceeds 60, the algorithm can run stably. For more safety, the population size is set to 80. The effect of the number of generations is shown in Figure 13. When the number of generations exceeds 80, $Lopt$ is already good. In view of this, the maximum generation of the population is set to 80. The effect of the invalid solution operator rate on $Lopt$ is given in Figure 14. The figure shows that the higher the invalid solution operator rate, the more stable the algorithm. In the next parametric study, the invalid solution operator rate is set to 0.5. Next, Figure 15 presents the effect of the safety rate.

The higher the safety operator rate, the better the *Lopt*. The value is also set to 0.5. Figure 16 shows that an excessively low crossover rate can affect the stability of the algorithm. Therefore, the crossover rate is set to 0.8. Figure 17 implies that when the shortness operator rate changes from 0.4 to one, *Lopt* is basically unchanged. The shortness operator rate is set to 0.5. The effect of smoothness operator rate is shown in Figure 18. We set the smoothness operator rate to 0.5. Figures 19–21 display the effects of the position update operator rate, insert rate and mutation rate, respectively. The effect of the shortest operator rate is presented in Figure 22. When the shortest operator rate changes from 0.1 to one, the algorithm is robust. This is different from other parameters.

The above parameter study also confirms the setting of the parameters in the initialization process. For more complex scenarios, it may be necessary to choose a higher population size, a larger number of generations, a higher invalid solution operator rate, a higher safety operator rate, a higher crossover rate and a higher smoothness operator rate.

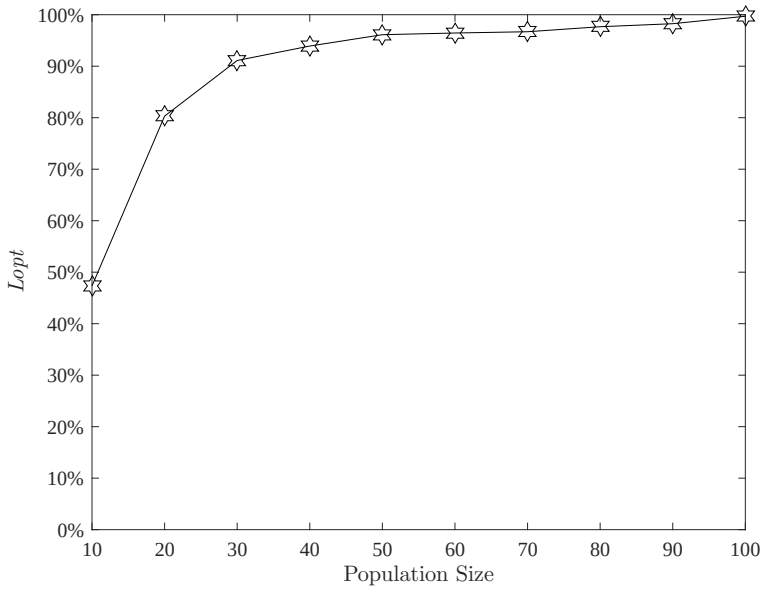

Figure 12. Effect of the population size.

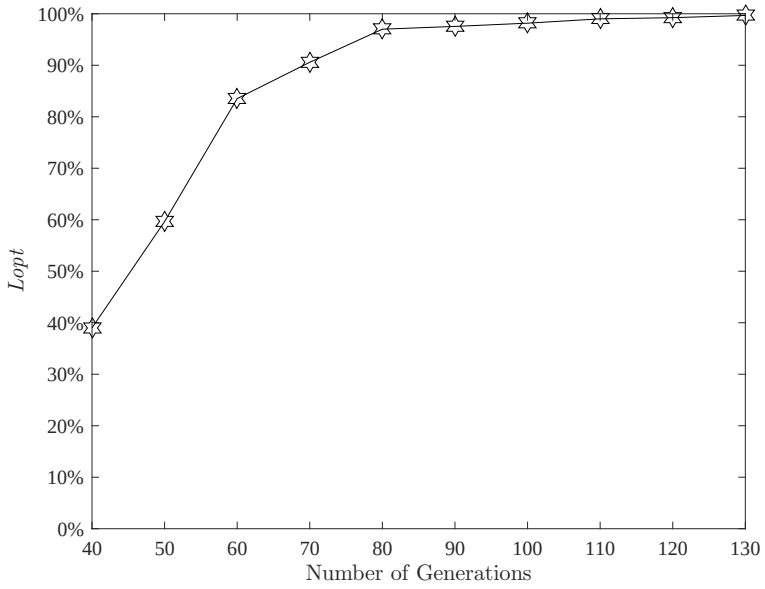

Figure 13. Effect of the number of generations.

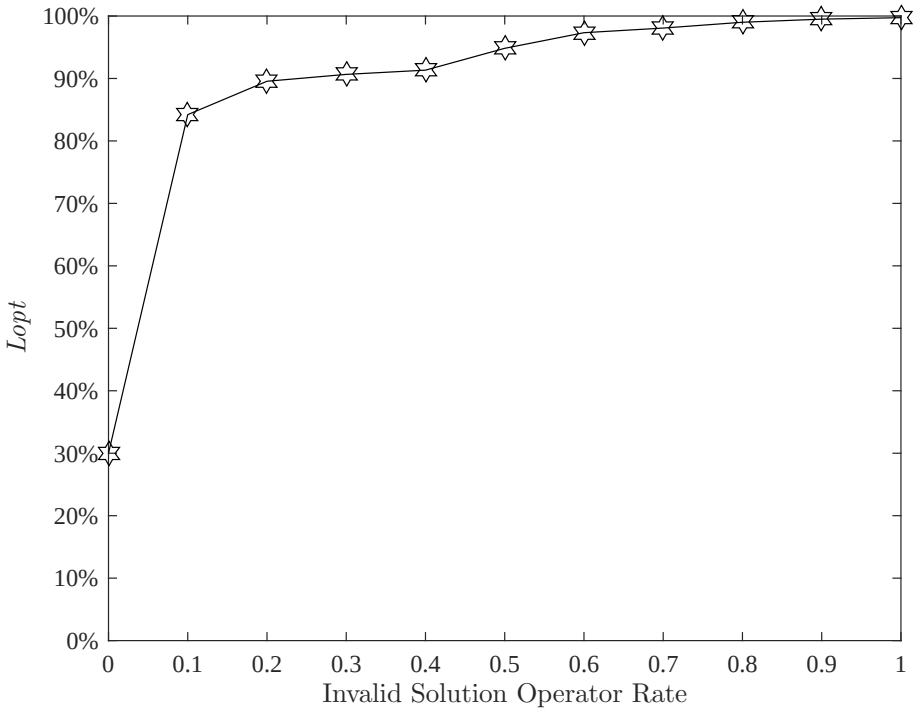

Figure 14. Effect of the invalid solution operator rate.

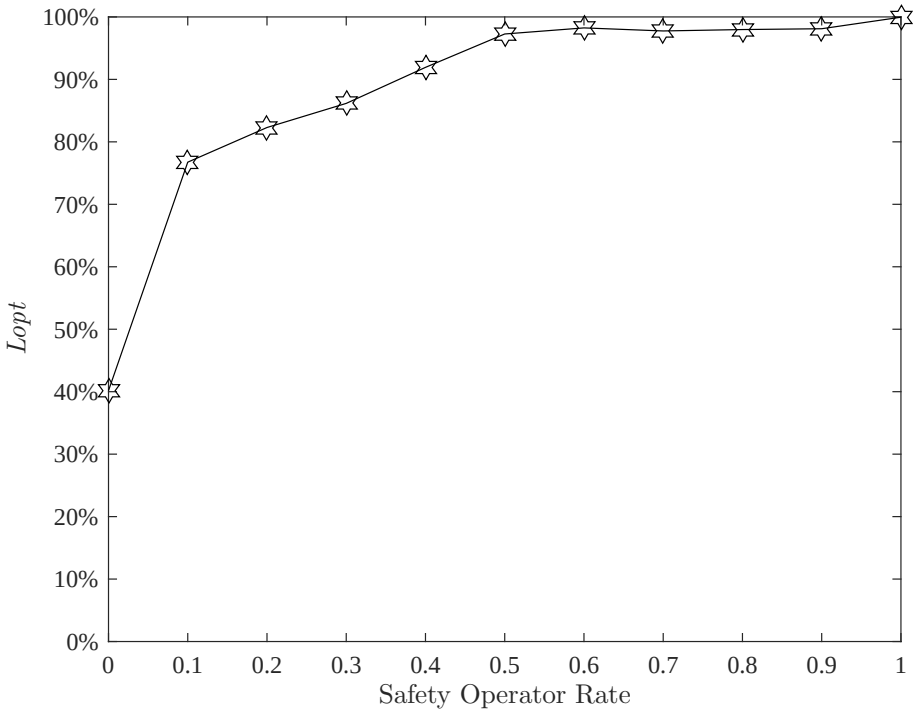

Figure 15. Effect of the safety operator rate.

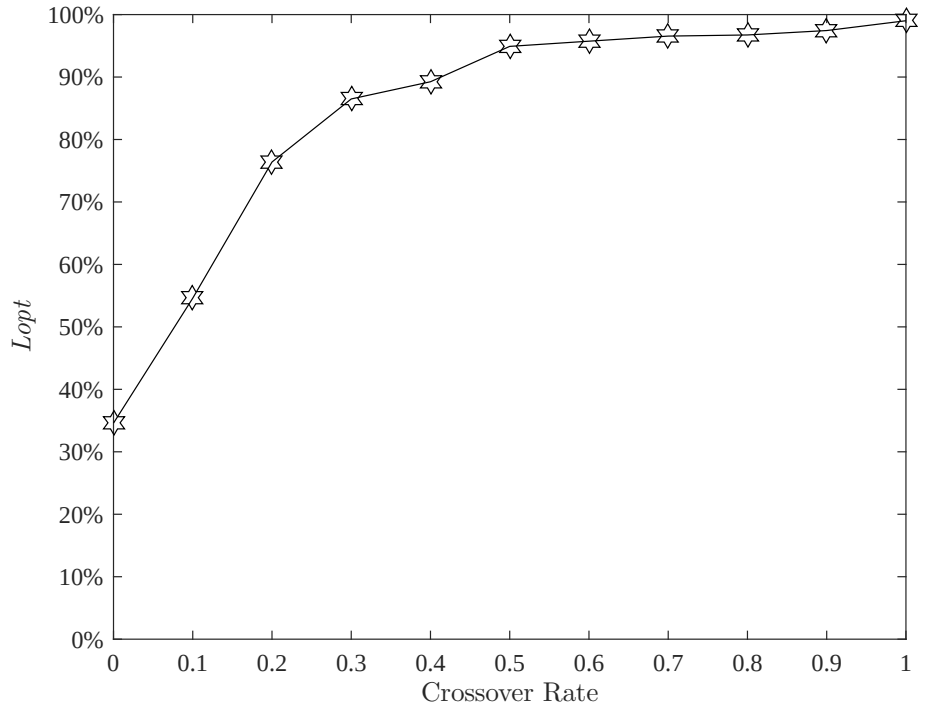

Figure 16. Effect of the crossover rate.

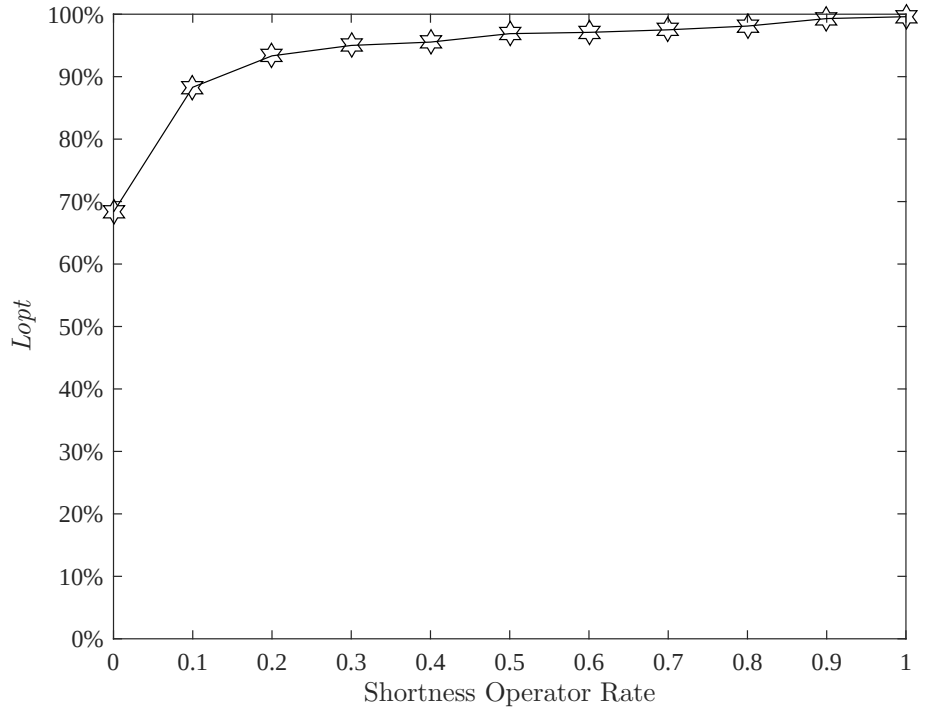

Figure 17. Effect of the shortness operator rate.

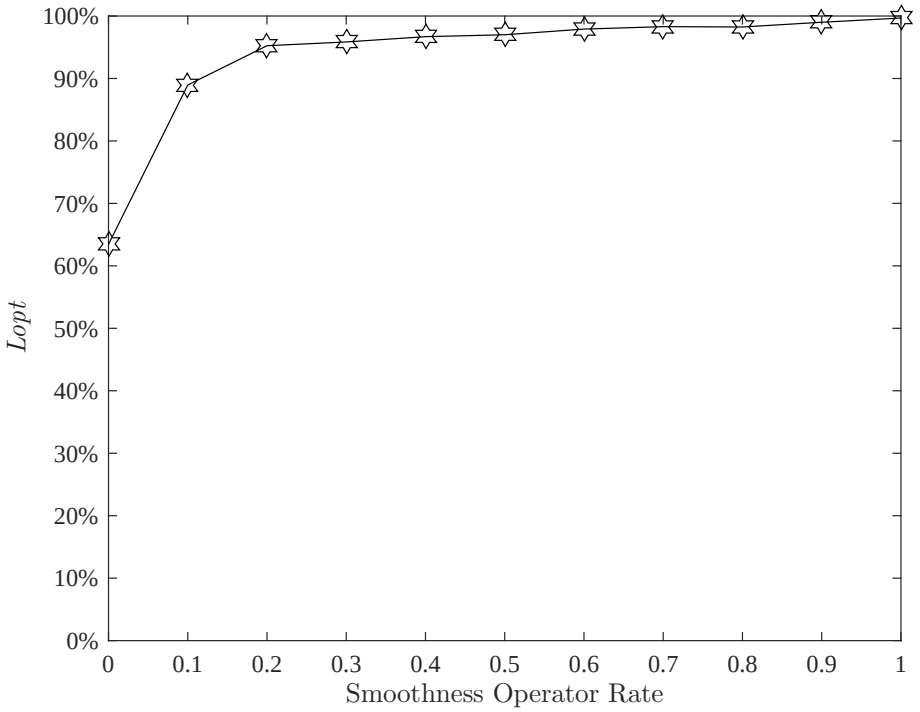

Figure 18. Effect of the smoothness operator rate.

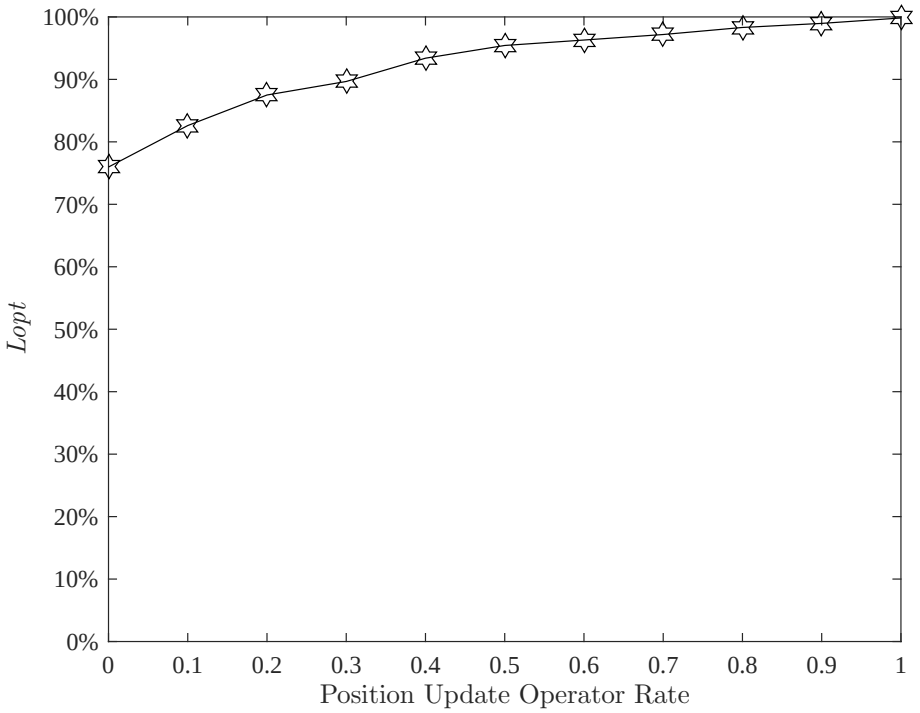

Figure 19. Effect of the position update operator rate.

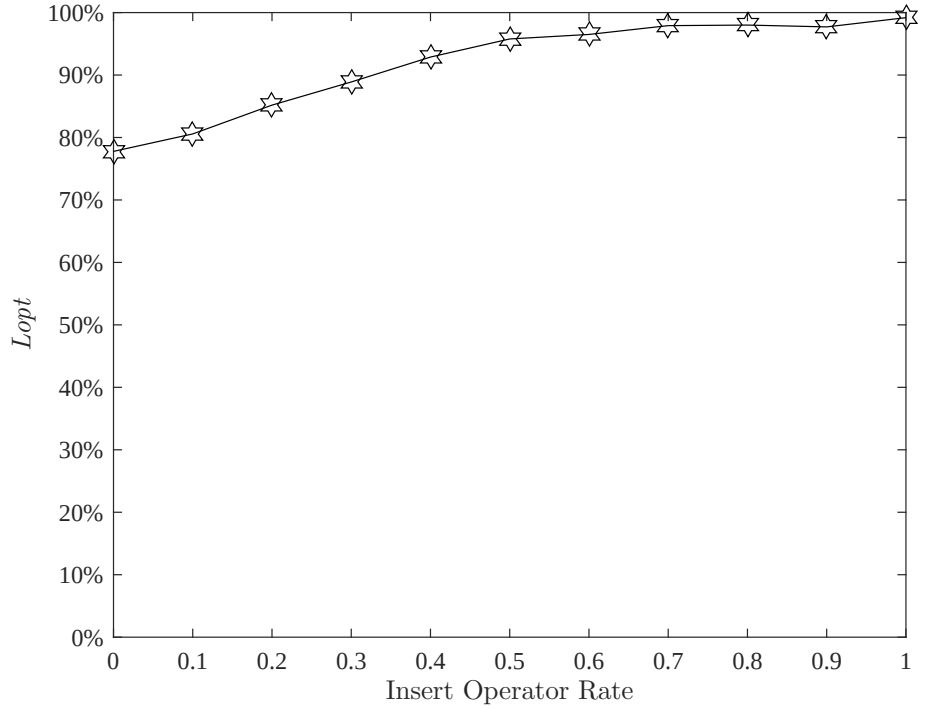

Figure 20. Effect of the insert operator rate.

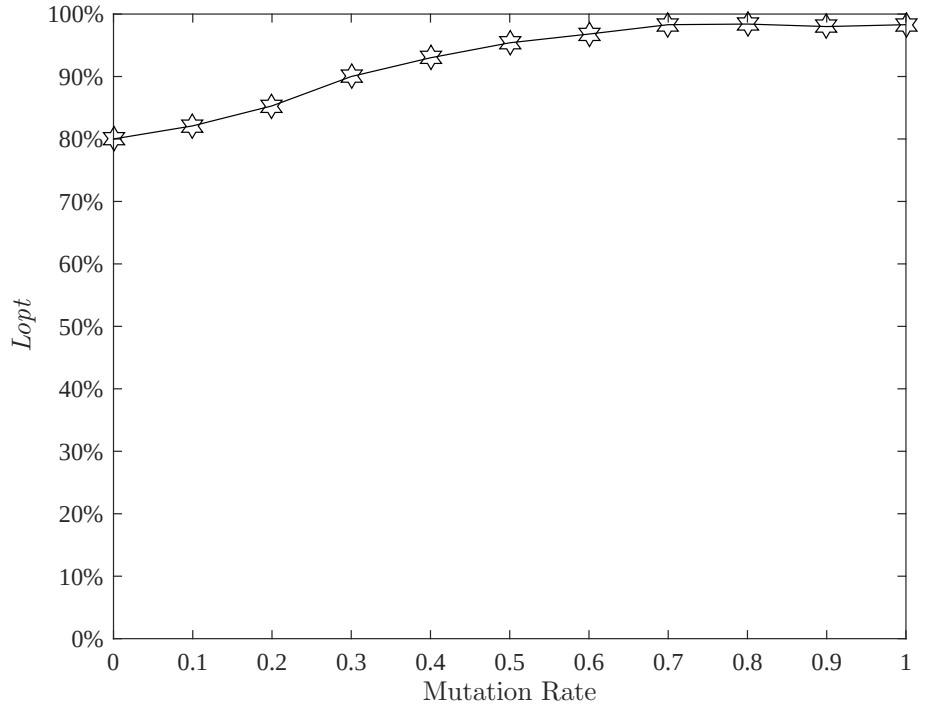

Figure 21. Effect of the mutation rate.

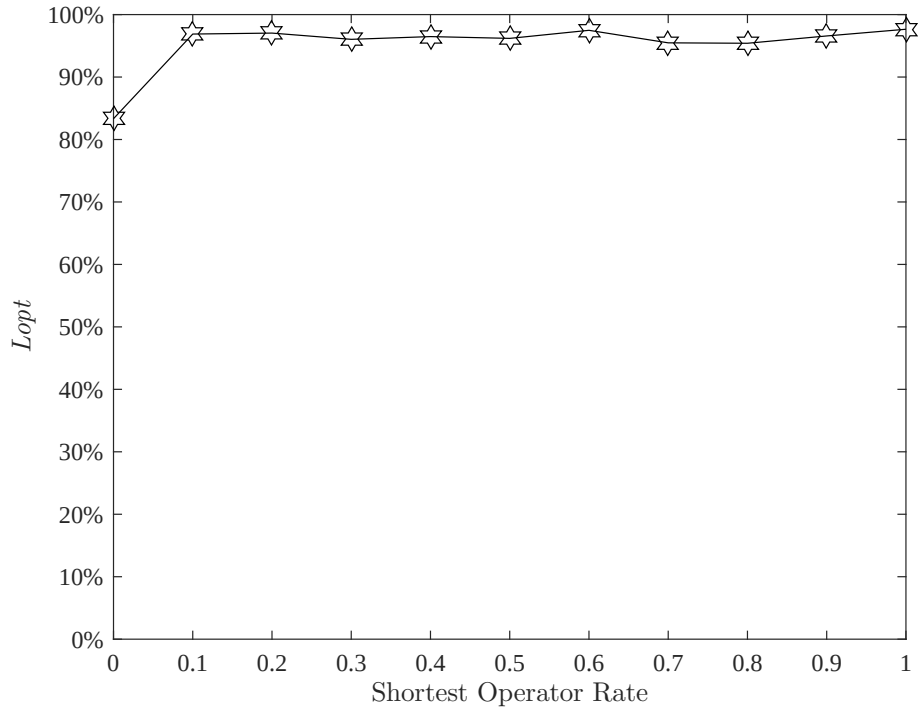

Figure 22. Effect of the shortest operator rate.

6. Results

In this work, the results of the improved non-dominated sorting genetic algorithm (NSGA-II) in various spaces are presented. To better examine the results, different quality metrics [58,59] and the multi-objective evolutionary algorithm (MOEA) algorithm [25] were employed.

The hypervolume indicator (HV) [58] can be used to evaluate the volume of non-dominated solutions in the solution space. In the d-dimensional solution space, the related concepts are defined. $A = \{a_1, a_2, ..., a_n\}$ is the non-dominated solutions, and $r = (r_1, r_2, ..., r_d)$ denotes the reference point, respectively. The Lebesgue measure [60] can be used to compute the hypervolume area of the non-dominated solutions bound by the reference point. The calculation of the HV is shown in Equation (8).

$$\begin{aligned} h(a_i) &= [a_{i_1}, r_1] \times [a_{i_2}, r_2] \times ... \times [a_{i_d}, r_d] \\ HV(A, r) &= L(\bigcup_{i=1}^{|A|} h(a_i) | a_i \in A). \end{aligned} \tag{8}$$

Furthermore, the set coverage metric (SCM) [59] can be applied for computing the dominance ratio of the two sets of non-dominated solutions. $A = \{a_1, a_2, ..., a_n\}$ and $B = \{b_1, b_2, \ldots, b_m\}$ denote the two sets. Consequently, $scm(A, B)$ can be computed with Equation (9). As the relationship ($\preceq$) represents a weak dominance relationship and is asymmetrical, it is also essential to determine $scm(B, A)$.

$$scm(A, B) = \frac{|\{b \in B | \exists a \in A : a \preceq b\}|}{|B|}. \tag{9}$$

In this work, several maps are considered in Figure 11. There are different shapes of obstacles on each map. The starting and target points for these maps are displayed in Table 1. On each map, both algorithms run 30 times. In each run, the two sets of non-dominated solutions obtained by the two algorithms are compared with the quality metrics. The results of the hypervolume are given in Table 2. In contrast, these two sets are dense. Besides, in Table 3, the results of the set coverage metric (SCM) are exhibited. The average of scm (NSGA-II,MOEA) and SCM (MOEA,NSGA-II) is 95.01% and 94.47%,

respectively. It is evident that the most solutions within any set are non-dominated relative to another set. This also implies that the solutions obtained by these two algorithms are close to the Pareto front.

Table 2. Hypervolume results of NSGA-II and MOEA.

Maps	NSGA-II($\text{Median}_{\text{Interquartile}}$)	MOEA($\text{Median}_{\text{Interquartile}}$)
Figure 11A	$0.8873_{0.0577}$	$0.8905_{0.0495}$
Figure 11B	$0.8967_{0.0339}$	$0.8840_{0.0391}$
Figure 11C	$0.8975_{0.0837}$	$0.8934_{0.0885}$
Figure 11D	$0.8931_{0.0386}$	$0.9000_{0.0422}$
Figure 11E	$0.8963_{0.0301}$	$0.8950_{0.0322}$

Table 3. The results of the set coverage metric.

Maps	scm(NSGA-II, MOEA)	scm(MOEA, NSGA-II)
Figure 11A	0.9077	0.9032
Figure 11B	0.8974	0.8882
Figure 11C	0.9947	0.9894
Figure 11D	0.9878	0.9680
Figure 11E	0.9630	0.9747
Average	0.9501	0.9447

Additionally, in each map, the reference points need to be calculated. The ideal reference point is optimal in every dimension, and the nadir reference point is worst. Table 4 shows the reference points. An ideal shortest route is the line segment from the start to the destination. The smoothness of the ideal smoothest path should be zero. For the minimum safety of path, this value can be estimated with enhancing the safety of the safest route by 10% in the non-dominated solutions. At this point, the three values of the ideal reference point have been set. Next, the three values of the nadir reference point are set. For the length and smoothness of the nadir reference point, the two values can be calculated by referring to the above means of estimating the minimum safety. The maximum safety of path is zero.

Table 4. Reference points.

Maps	Ideal	Nadir
Figure 11A	$(3612.5, -60.5, 0)$	$(6602.0, 0, 48.5)$
Figure 11B	$(1710.3, -35.4, 0)$	$(8445.0, 0, 59.1)$
Figure 11C	$(1827.7, -15.4, 0)$	$(3545.2, 0, 31.8)$
Figure 11D	$(982.9, -88, 0)$	$(6062.9, 0, 82.5)$
Figure 11E	$(2375.9, -52.3, 0)$	$(4931.6, 0, 63.2)$

Figures 23–27 present the approximate Pareto front of maps. The yellow circle is the ideal reference point. Red triangles and black asterisks are the non-dominated solutions with the improved NSGA-II and MOEA, respectively. In non-dominated solutions, the knee spot is a solution closest to the ideal reference point. Figures 28–32 show the path of the knee point obtained via NSGA-II. Comparison results are analyzed as follows. From the two different quality metrics of hypervolume and set coverage, the improved NSGA-II demonstrates a characteristic that is no worse than MOEA. Moreover, the non-dominated solutions generated via NSGA-II are denser. Besides, the path corresponding to the knee point also takes into account several designed objectives. The path is shorter, smoother and safer.

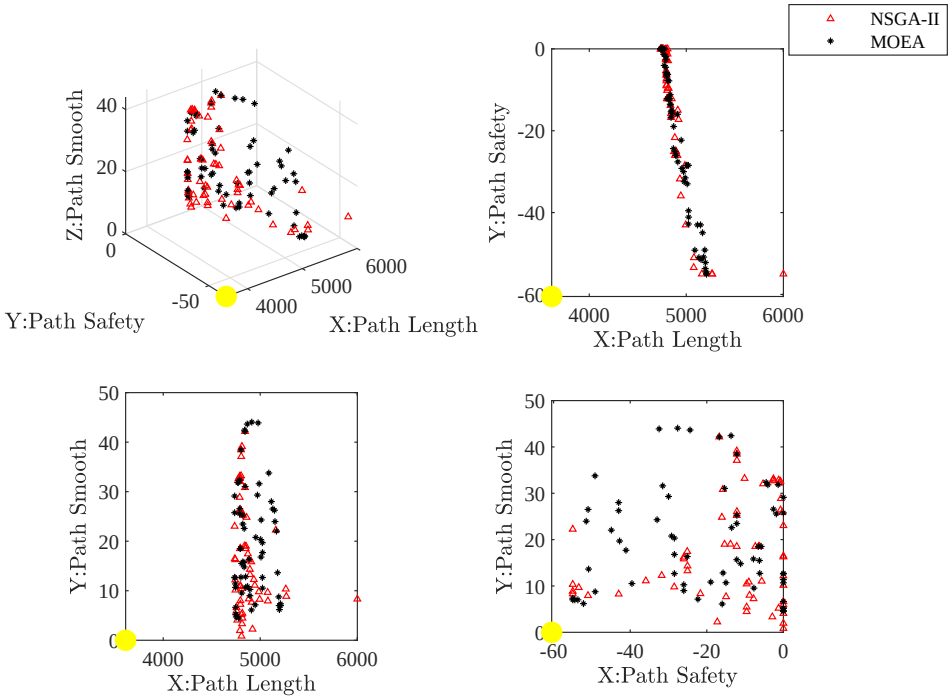

Figure 23. Based on Figure 11A, the approximate Pareto fronts generated by the two algorithms NSGA-II and MOEA. The sub-figure of the upper left corner shows the image of the approximate Pareto front in three-dimensional space. The remaining three sub-figures are the projections of the approximate Pareto front in the two-dimensional space.

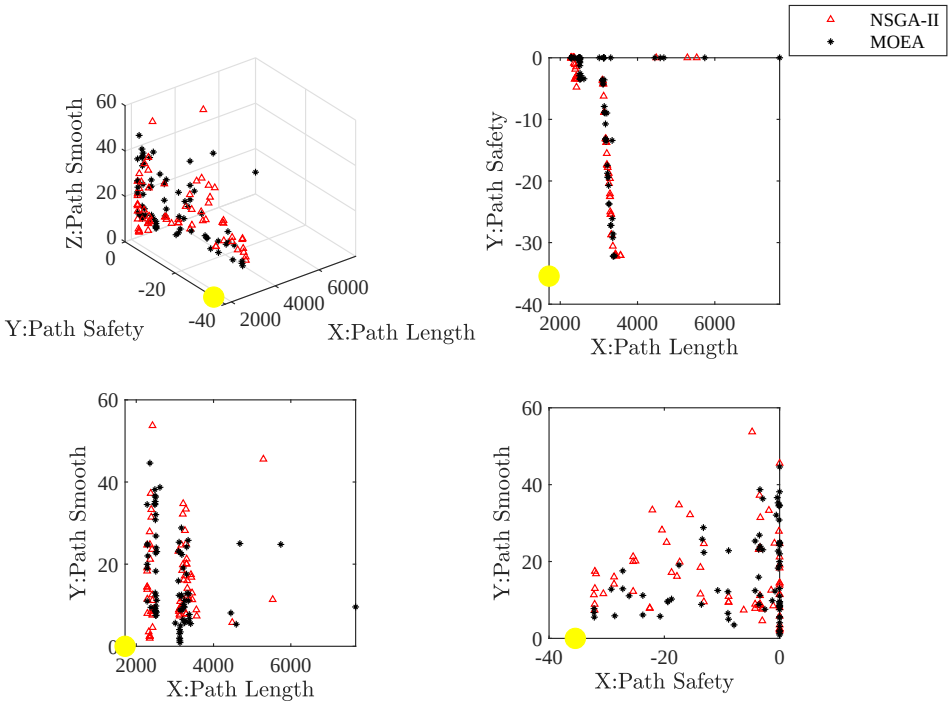

Figure 24. Based on Figure 11B, the approximate Pareto fronts generated by the two algorithms NSGA-II and MOEA. The sub-figure of the upper left corner shows the image of the approximate Pareto front in three-dimensional space. The remaining three sub-figures are the projections of the approximate Pareto front in the two-dimensional space.

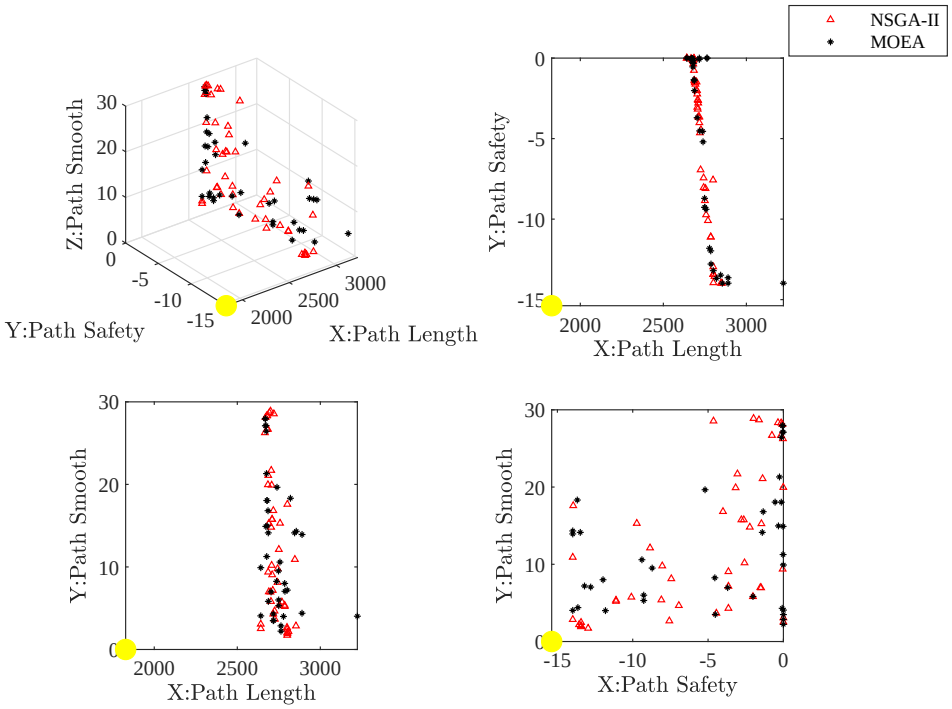

Figure 25. Based on Figure 11C, the approximate Pareto fronts generated by the two algorithms NSGA-II and MOEA. The sub-figure of the upper left corner shows the image of the approximate Pareto front in three-dimensional space. The remaining three sub-figures are the projections of the approximate Pareto front in the two-dimensional space.

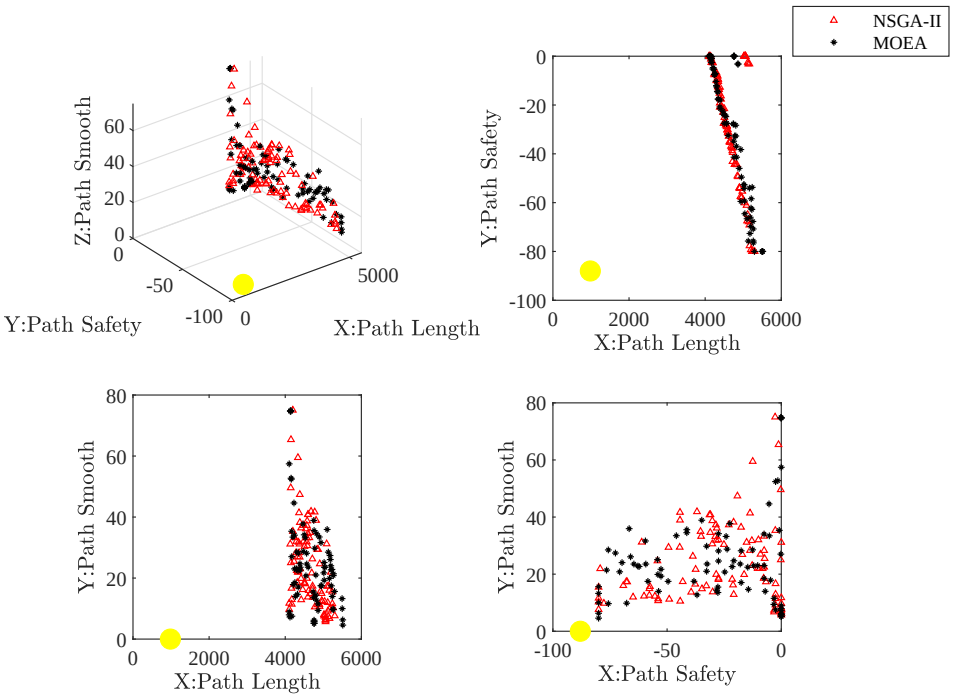

Figure 26. Based on Figure 11D, the approximate Pareto fronts generated by the two algorithms NSGA-II and MOEA. The sub-figure of the upper left corner shows the image of the approximate Pareto front in three-dimensional space. The remaining three sub-figures are the projections of the approximate Pareto front in the two-dimensional space.

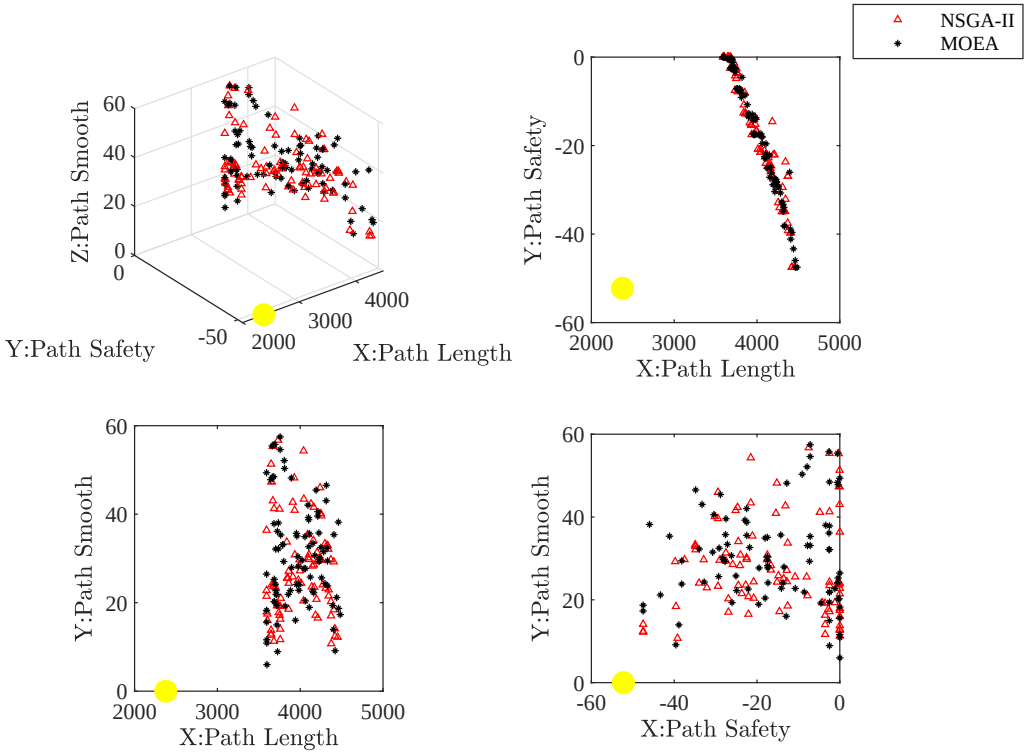

Figure 27. Based on Figure 11E, the approximate Pareto fronts generated by the two algorithms NSGA-II and MOEA. The sub-figure of the upper left corner shows the image of the approximate Pareto front in three-dimensional space. The remaining three sub-figures are the projections of the approximate Pareto front in the two-dimensional space.

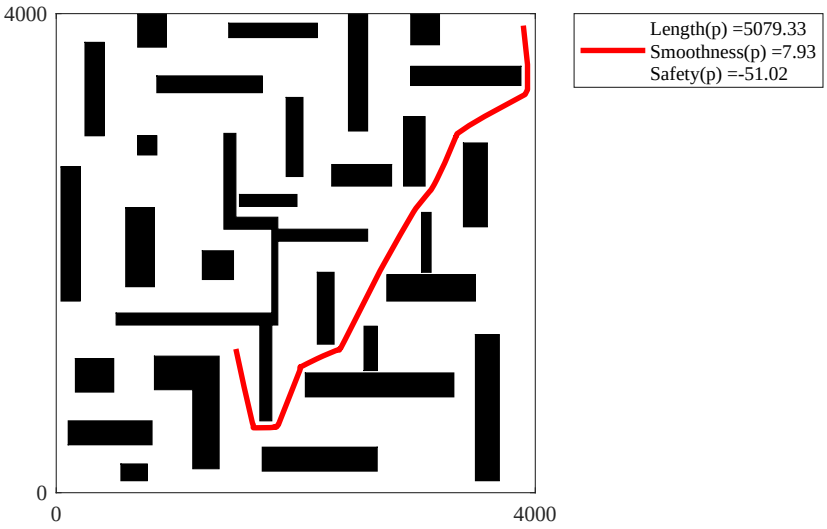

Figure 28. Path of the knee point generated via NSGA-II for Figure 11A.

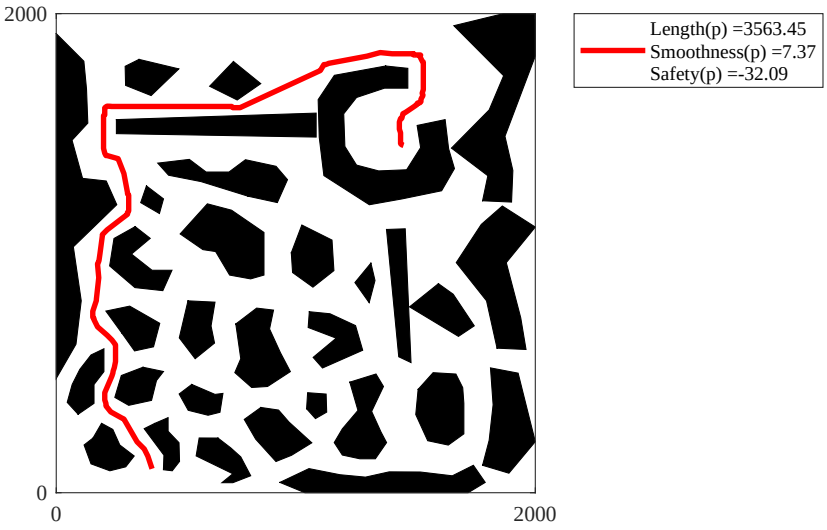

Figure 29. Path of the knee point generated via NSGA-II for Figure 11B.

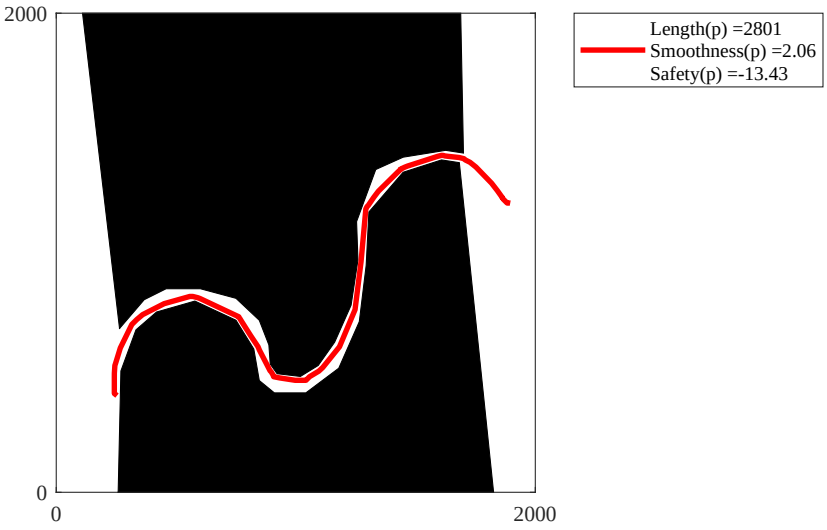

Figure 30. Path of the knee point generated via NSGA-II for Figure 11C.

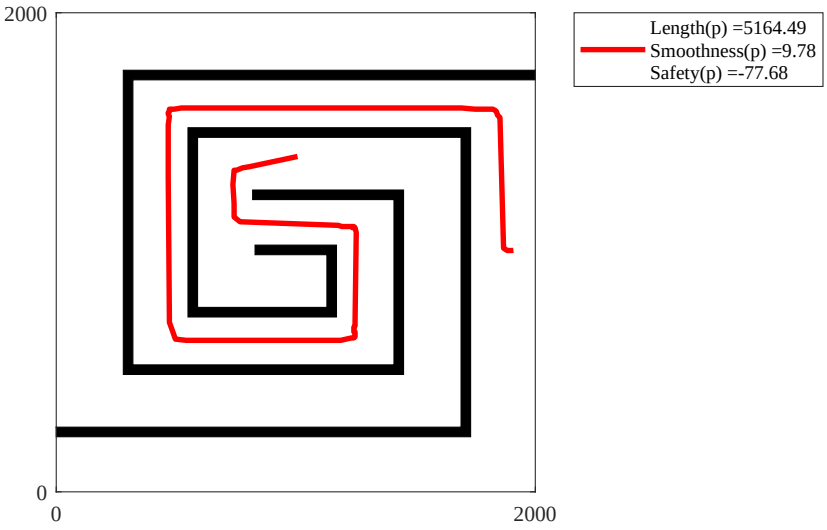

Figure 31. Path of the knee point generated via NSGA-II for Figure 11D.

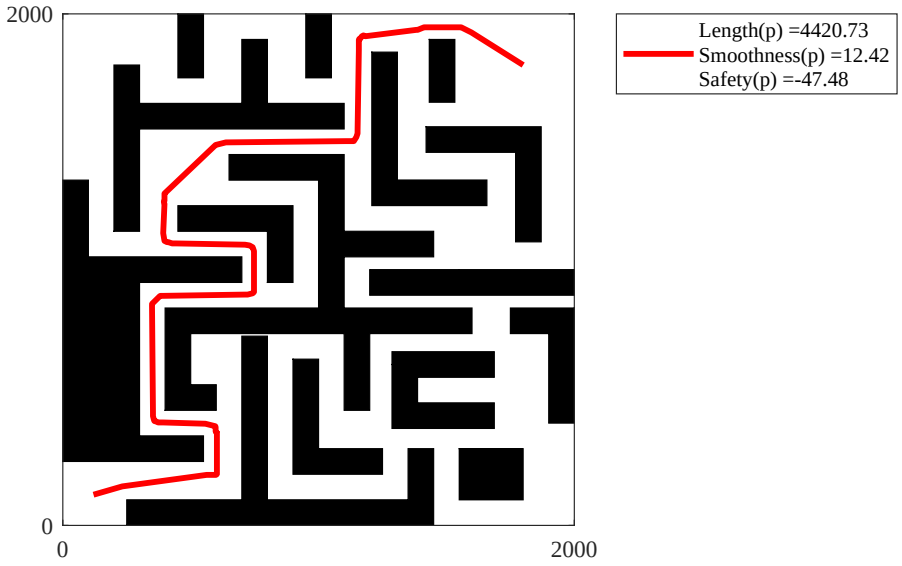

Figure 32. Path of the knee point generated via NSGA-II for Figure 11E.

7. Concluding Remarks

In this work, an improved non-dominated sorting genetic algorithm (NSGA-II) is presented to solve the multi-objective path planning problem in statically known environments. The model and the three objectives to be optimized are introduced. Based on the framework of NSGA-II, more practical operators are proposed to accelerate the evolutionary speed of individuals. These operators help individuals become shorter, safer and smoother. Then, the parameters in the algorithm are systematically studied. To discern the capabilities of the algorithm, an effective multi-objective evolutionary algorithm is employed for comparison. The set coverage metric and hypervolume are adopted. Comparison results demonstrate that the non-dominated solutions generated by the improved NSGA-II have excellent characteristics in the solution space. Finally, the path corresponding to the knee point is displayed. The path is shorter, smoother and safer. It can be adopted as the tracking path of the mobile robot in the later decision.

Due to the speed of the operators in the algorithm, it is currently only applicable to offline path planning. Moreover, the kinematics of the robot is not considered. Besides, the improved algorithm only consider the statically known environments, so the situation with dynamic obstacles in the unknown environments deserves further research.

Author Contributions: Y.X. investigated the relevant literature, proposed and implemented the algorithm, confirmed the feasibility of the algorithm, wrote and revised the paper.

Funding: This research received no external funding.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Choset, H.M.; Hutchinson, S.; Lynch, K.M.; Kantor, G.; Burgard, W.; Kavraki, L.E.; Thrun, S. *Principles of Robot Motion: Theory, Algorithms, and Implementation*; The MIT Press: Cambridge, MA, USA, 2005.
2. LaValle, S.M. *Planning Algorithms*; Cambridge University Press: Cambridge, UK, 2006.
3. Mac, T.T.; Copot, C.; Tran, D.T.; De Keyser, R. Heuristic approaches in robot path planning: A survey. *Robot. Auton. Syst.* **2016**, *86*, 13–28. [CrossRef]
4. Singh, Y.; Sharma, S.; Sutton, R.; Hatton, D.; Khan, A. A constrained A* approach towards optimal path planning for an unmanned surface vehicle in a maritime environment containing dynamic obstacles and ocean currents. *Ocean Eng.* **2018**, *169*, 187–201. [CrossRef]

5. Ma, Y.; Hu, M.; Yan, X. Multi-objective path planning for unmanned surface vehicle with currents effects. *ISA Trans.* **2018**, *75*, 137–156. [CrossRef] [PubMed]
6. Alomari, A.; Phillips, W.; Aslam, N.; Comeau, F. Swarm intelligence optimization techniques for obstacle-avoidance mobility-assisted localization in wireless sensor networks. *IEEE Access* **2018**, *99*, 22368–22385. [CrossRef]
7. Alomari, A.; Phillips, W.; Aslam, N.; Comeau, F. Dynamic fuzzy-logic based path planning for mobility-assisted localization in wireless sensor networks. *Sensors* **2017**, *17*, 1904. [CrossRef] [PubMed]
8. Chen, Y.; Lu, S.; Chen, J.; Ren, T. Node localization algorithm of wireless sensor networks with mobile beacon node. *Peer-to-Peer Netw. Appl.* **2017**, *10*, 795–807. [CrossRef]
9. Algfoor, Z.A.; Sunar, M.S.; Kolivand, H. A comprehensive study on pathfinding techniques for robotics and video games. *Int. J. Comput. Games Technol.* **2015**, *2015*, 7. [CrossRef]
10. Zafar, M.N.; Mohanta, J.C. Methodology for Path Planning and Optimization of Mobile Robots: A Review. *Procedia Comput. Sci.* **2018**, *133*, 141–152. [CrossRef]
11. Zhang, H.Y.; Lin, W.M.; Chen, A.X. Path Planning for the Mobile Robot: A Review. *Symmetry* **2018**, *10*, 450. [CrossRef]
12. Radmanesh, M.; Kumar, M.; Guentert, P.H.; Sarim, M. Overview of Path-Planning and Obstacle Avoidance Algorithms for UAVs: A Comparative Study. *Unmanned Syst.* **2018**, *6*, 95–118. [CrossRef]
13. Wang, Z.; Cai, J. Probabilistic roadmap method for path-planning in radioactive environment of nuclear facilities. *Prog. Nucl. Energy* **2018**, *109*, 113–120. [CrossRef]
14. Sudhakara, P.; Ganapathy, V.; Priyadharshini, B.; Sundaran, K. Obstacle Avoidance and Navigation Planning of a Wheeled Mobile Robot using Amended Artificial Potential Field Method. *Procedia Comput. Sci.* **2018**, *133*, 998–1004. [CrossRef]
15. Wang, W.; Deng, H.; Wu, X. Path planning of loaded pin-jointed bar mechanisms using Rapidly-exploring Random Tree method. *Comput. Struct.* **2018**, *209*, 65–75. [CrossRef]
16. Duchoň, F.; Babinec, A.; Kajan, M.; Beňo, P.; Florek, M.; Fico, T.; Jurišica, L. Path Planning with Modified a Star Algorithm for a Mobile Robot. *Procedia Eng.* **2014**, *96*, 59–69. [CrossRef]
17. Ammar, A.; Bennaceur, H.; Châari, I.; Koubâa, A.; Alajlan, M. Relaxed Dijkstra and A* with linear complexity for robot path planning problems in large-scale grid environments. *Soft Comput.* **2016**, *20*, 4149–4171. [CrossRef]
18. Mrudul, K.; Mandava, R.K.; Vundavilli, P.R. An Efficient Path Planning Algorithm for Biped Robot using Fast Marching Method. *Procedia Comput. Sci.* **2018**, *133*, 116–123. [CrossRef]
19. Canny, J. *The Complexity of Robot Motion Planning*; The MIT Press: Cambridge, MA, USA, 1988.
20. Elshamli, A.; Abdullah, H.A.; Areibi, S. Genetic algorithm for dynamic path planning. In Proceedings of the Electrical and Computer Engineering, Niagara Falls, ON, Canada, 2–5 May 2004; pp. 677–680. [CrossRef]
21. Chen, X.; Li, Y. Smooth Path Planning of a Mobile Robot Using Stochastic Particle Swarm Optimization. In Proceedings of the IEEE International Conference on Mechatronics and Automation, Luoyang, China, 25–28 June 2006; pp. 1722–1727. [CrossRef]
22. Mandow, L.; De La Cruz, J.L.P. Multiobjective A* search with consistent heuristics. *J. ACM* **2010**, *57*, 27. [CrossRef]
23. Lavin, A. A pareto optimal D* search algorithm for multiobjective path planning. *arXiv* **2015**, arXiv:1511.00787.
24. Oral, T.; Polat, F. MOD* Lite: An incremental path planning algorithm taking care of multiple objectives. *IEEE Trans. Cybern.* **2016**, *46*, 245–257. [CrossRef] [PubMed]
25. Xue, Y.; Sun, J.Q. Solving the Path Planning Problem in Mobile Robotics with the Multi-Objective Evolutionary Algorithm. *Appl. Sci.* **2018**, *8*, 1425. [CrossRef]
26. Ghatee, M.; Mohades, A. Motion planning in order to optimize the length and clearance applying a Hopfield neural network. *Expert Syst. Appl.* **2009**, *36*, 4688–4695. [CrossRef]
27. Zhang, P.; Xiong, C.; Li, W.; Du, X.; Zhao, C. Path planning for mobile robot based on modified rapidly exploring random tree method and neural network. *Int. J. Adv. Robot. Syst.* **2018**, *15*. [CrossRef]
28. Konar, A.; Goswami, I.; Singh, S.J.; Jain, L.C.; Nagar, A.K. A Deterministic Improved Q-Learning for Path Planning of a Mobile Robot. *IEEE Trans. Syst. Man Cybern. Syst.* **2013**, *43*, 1141–1153. [CrossRef]
29. Chen, X.; Kong, Y.; Fang, X.; Wu, Q. A fast two-stage ACO algorithm for robotic path planning. *Neural Comput. Appl.* **2013**, *22*, 313–319. [CrossRef]

30. Châari, I.; Koubâa, A.; Bennaceur, H.; Ammar, A.; Trigui, S.; Tounsi, M.; Shakshuki, E.; Youssef, H. On the Adequacy of Tabu Search for Global Robot Path Planning Problem in Grid Environments. *Procedia Comput. Sci.* **2014**, *32*, 604–613. [CrossRef]
31. Zhu, Z.; Xiao, J.; Li, J.Q.; Wang, F.; Zhang, Q. Global path planning of wheeled robots using multi-objective memetic algorithms. *Integr. Comput.-Aided Eng.* **2015**, *22*, 387–404. [CrossRef]
32. Salmanpour, S.; Monfared, H.; Omranpour, H. Solving robot path planning problem by using a new elitist multi-objective IWD algorithm based on coefficient of variation. *Soft Comput.* **2017**, *21*, 3063–3079. [CrossRef]
33. Contreras-Cruz, M.A.; Ayala-Ramirez, V.; Hernandez-Belmonte, U.H. Mobile robot path planning using artificial bee colony and evolutionary programming. *Appl. Soft Comput.* **2015**, *30*, 319–328. [CrossRef]
34. Gong, D.W.; Zhang, J.H.; Zhang, Y. Multi-objective particle swarm optimization for robot path planning in environment with danger sources. *J. Comput.* **2011**, *6*, 1554–1561. [CrossRef]
35. Zhang, Y.; Gong, D.W.; Zhang, J.H. Robot path planning in uncertain environment using multi-objective particle swarm optimization. *Neurocomputing* **2013**, *103*, 172–185. [CrossRef]
36. Tharwat, A.; Elhoseny, M.; Hassanien, A.E.; Gabel, T.; Kumar, A. Intelligent Bézier curve-based path planning model using Chaotic Particle Swarm Optimization algorithm. *Clust. Comput.* **2018**, *2018*, 1–22. [CrossRef]
37. Mac, T.T.; Copot, C.; Tran, D.T.; De Keyser, R. A hierarchical global path planning approach for mobile robots based on multi-objective particle swarm optimization. *Appl. Soft Comput.* **2017**, *59*, 68–76. [CrossRef]
38. Han, J.; Seo, Y. Mobile robot path planning with surrounding point set and path improvement. *Appl. Soft Comput.* **2017**, *57*, 35–47. [CrossRef]
39. Goldberg, D.E. *Genetic Algorithms in Search, Optimization and Machine Learning*; Addison-Wesley: Boston, MA, USA, 1989.
40. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T.A.M.T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [CrossRef]
41. Yao, L.; Lim, W.; Tiang, S.; Tan, T.; Wong, C.; Pang, J. Demand Bidding Optimization for an Aggregator with a Genetic Algorithm. *Energies* **2018**, *11*, 2498. [CrossRef]
42. Martínez-Bahena, B.; Cruz-Chávez, M.; Ávila-Melgar, E.; Cruz-Rosales, M.; Rivera-Lopez, R. Using a Genetic Algorithm with a Mathematical Programming Solver to Optimize a Real Water Distribution System. *Water* **2018**, *10*, 1318. [CrossRef]
43. Mahmood, A.; Khan, S.A.; Bahlool, R.A. Correction: Mahmood et al. Hard Real-Time Task Scheduling in Cloud Computing Using an Adaptive Genetic Algorithm. *Computers* **2017**, *6*, 15. *Computers* **2018**, *7*, 35. [CrossRef]
44. Ismail, A.T.; Sheta, A.; Al-Weshah, M. A Mobile Robot Path Planning Using Genetic Algorithm in Static Environment. *J. Comput. Sci.* **2008**, *4*, 341–344. [CrossRef]
45. Santiago, R.M.C.; De Ocampo, A.L.; Ubando, A.T.; Bandala, A.A.; Dadios, E.P. Path planning for mobile robots using genetic algorithm and probabilistic roadmap. In Proceedings of the Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management, Manila, Philippines, 1–3 December 2017; pp. 1–5. [CrossRef]
46. Davoodi, M.; Panahi, F.; Mohades, A.; Hashemi, S.N. Multi-objective path planning in discrete space. *Appl. Soft Comput.* **2013**, *13*, 709–720. [CrossRef]
47. Ahmed, F.; Deb, K. Multi-objective optimal path planning using elitist non-dominated sorting genetic algorithms. *Soft Comput.* **2013**, *17*, 1283–1299. [CrossRef]
48. Karami, A.H.; Hasanzadeh, M. An adaptive genetic algorithm for robot motion planning in 2D complex environments. *Comput. Electr. Eng.* **2015**, *43*, 317–329. [CrossRef]
49. Mittal, S.; Deb, K. Three-dimensional offline path planning for UAVs using multiobjective evolutionary algorithms. In Proceedings of the IEEE Congress on Evolutionary Computation, Singapore, 25–28 September 2007; pp. 3195–3202. [CrossRef]
50. Lee, J.; Kim, D.W. An effective initialization method for genetic algorithm-based robot path planning using a directed acyclic graph. *Inf. Sci.* **2016**, *332*, 1–18. [CrossRef]
51. Bakdi, A.; Hentout, A.; Boutami, H.; Maoudj, A.; Hachour, O.; Bouzouia, B. Optimal path planning and execution for mobile robots using genetic algorithm and adaptive fuzzy-logic control. *Robot. Auton. Syst.* **2017**, *89*, 95–109. [CrossRef]
52. Patle, B.K.; Parhi, D.R.K.; Jagadeesh, A.; Kashyap, S.K. Matrix-Binary Codes based Genetic Algorithm for path planning of mobile robot. *Comput. Electr. Eng.* **2017**, *67*, 708–728. [CrossRef]

53. Elhoseny, M.; Tharwat, A.; Hassanien, A.E. Bezier Curve Based Path Planning in a Dynamic Field using Modified Genetic Algorithm. *J. Comput. Sci.* **2018**, *25*, 339–350. [CrossRef]
54. Lamini, C.; Benhlima, S.; Elbekri, A. Genetic Algorithm Based Approach for Autonomous Mobile Robot Path Planning. *Procedia Comput. Sci.* **2018**, *127*, 180–189. [CrossRef]
55. Shehata, H.H.; Schlattmann, J. Non-dominated sorting genetic algorithm for smooth path planning in unknown environments. In Proceedings of the IEEE International Conference on Autonomous Robot Systems and Competitions, Espinho, Portugal, 14–15 May 2014; pp. 14–21. [CrossRef]
56. De Berg, M.; Van Kreveld, M.; Overmars, M.; Schwarzkopf, O. Computational geometry. In *Computational Geometry*; Springer: Berlin, Germany, 1997; pp. 1–17.
57. Sugihara, K. Measures for performance evaluation of genetic algorithms. In Proceedings of the Joint Conference on Information Sciences, Durham, NC, USA, 1–5 March 1997; pp. 172–175.
58. Bader, J.; Zitzler, E. HypE: An algorithm for fast hypervolume-based many-objective optimization. *Evol. Comput.* **2011**, *19*, 45–76. [CrossRef] [PubMed]
59. Zitzler, E.; Deb, K.; Thiele, L. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evol. Comput.* **2000**, *8*, 173–195. [CrossRef] [PubMed]
60. Bartle, R.G. *The Elements of Integration and Lebesgue Measure*; Wiley Classics Library: New York, NY, USA, 1995.

© 2018 by the author. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).