electronics

MDPI

Article

Improved Rapidly Exploring Random Tree with Bacterial Mutation and Node Deletion for Offline Path Planning of Mobile Robot

Aphilak Lonklang and János Botzheim *

Department of Artificial Intelligence, Faculty of Informatics, ELTE Eötvös Loránd University, Pázmány Péter Sétány 1/A, 1117 Budapest, Hungary; aphilak@inf.elte.hu

* Correspondence: botzheim@inf.elte.hu

Abstract: The path-planning algorithm aims to find the optimal path between the starting and goal points without collision. One of the most popular algorithms is the optimized Rapidly exploring Random Tree (RRT*). The strength of RRT* algorithm is the collision-free path. It is the main reason why RRT-based algorithms are used in path planning for mobile robots. The RRT* algorithm generally creates the node for randomly making a tree branch to reach the goal point. The weakness of the RRT* algorithm is in the random process when the randomized nodes fall into the obstacle regions. The proposed algorithm generates a new random environment by removing the obstacle regions from the global environment. The objective is to minimize the number of unusable nodes from the randomizing process. The results show better performance in computational time and overall path-planning length. Bacterial mutation and local search algorithms are combined at post-processing to get a better path length and reduce the number of nodes. The proposed algorithm is tested in simulation.

Keywords: RRT* algorithm; path planning; bacterial mutation; local search algorithm

Citation: Lonklang, A.; Botzheim, J. Improved Rapidly Exploring Random Tree with Bacterial Mutation and Node Deletion for Offline Path Planning of Mobile Robot. *Electronics* **2022**, *11*, 1459. https://doi.org/10.3390/electronics11091459

Academic Editor: Giuseppe Prencipe

Received: 11 April 2022
Accepted: 29 April 2022
Published: 3 May 2022

1. Introduction

The industry needs to improve productivity and decrease the time used for production processes. Automation and autonomous vehicles are solutions for this task that can be applied to production processes. The automated transportation of raw materials or equipment tasks is one of the main challenges of this situation that can be solved by improving the methodology and reducing the process time by developing mobile robots to achieve these goals. The critical point of this task is to control the vehicle to the delivery part from store to production lines and to realize communication between humans and robots for material- and tool-requesting tasks [1].

Global path planning is based on the general information of the feasible space and obstacle regions, which assume that there are no unexpected objects located in the feasible space. The algorithm works to find the feasible coordinates or path from the starting point to the desired goal point. Generally, the global path-planning algorithm requires a great deal of computational time. Dynamic objects are classified into the local path-planning of the computation tasks [2]. Local path-planning is performed during the motion of robots, getting the data from its sensors. Robots have to generate a new path to respond to the dynamic objects in the environment [3].

There are plenty of proposed path-planning algorithms. The classical algorithm with a simple structure and nature that is easy to implement in mobile robots is the artificial potential field (APF) [4], which is a gradient-based method that can lead the path planning from the starting point to the goal point by following the gradient of the created virtual potential field. The main disadvantage of the APF is that it is easy to fall into the local optimum. Researchers tried to improve the APF to avoid the local optima by steering

from the local minimum point to continue the searching algorithm [5]. In APF, a global potential field is calculated, then local fields dealing with the obstacles are also calculated, and finally the fields are summed. Suppose the obstacles have irregular shapes; then, the created field will not perfectly fit the obstacle regions. Our proposed algorithm solves this problem. Furthermore, the more compatible algorithm for path-planning problems in 2D is A* (A-star). The traditional A* is commonly used for 2D mobile robots. The A* is the point-search algorithm that creates a minor path at the center of nodes and extends to another node by the horizontal line, vertical line, or diagonal line of the square grid in space. The researchers tried to improve the algorithm by minimizing the path length and node to get the minimum path [6]. Several intelligent algorithms were proposed to perform the path planning, such as the deep-reinforcement-learning strategy [7]. Moreover, one of the most famous algorithms for path planning for mobile robots in plane space is the Rapidly exploring Random Tree (RRT)-based algorithm.

The Rapidly exploring Random Tree (RRT) algorithm was proposed in 1998 by Lavalle et al. [8] as a randomized data structure that is designed for path-planning problems. The RRT algorithm performs to expand the random branch from the starting point to explore the entire environment. The path planning begins with a root node and creates random nodes to reach the goal node. The random nodes will be searched in a $M \times N$ matrix space. RRT can explore a large, non-convex environment and the generated path is obstacle-free; thus, RRT is frequently used in robot path planning.

The RRT* algorithm has been improved in several ways for the same reason: to gain more efficiency of the algorithm both in computational times and overall path length [9,10]. In [11] intelligent steering step-size was proposed. To avoid the obstacle, the researchers tried to optimize the steering-path length of each random branch. The result shows that the registered tree can be explored without falling into the obstacle region and can explore the global environment without passing through the current iteration. Another example of improved RRT* is to assign the desired goal point to a root of another tree. The algorithm explored both trees simultaneously. The results show that the computation times were reduced because the trees connect at the feasible area between them [12,13].

In recent years, RRT and RRT* algorithms have been widely applied to solve the problem of path planning [14–16], such as mobile robots [17], collaborative robots [18], industrial manipulators [19–21], Unmanned Aerial Vehicles (UAV) [22–24], and Underground Vehicles [11]. Unlike the traditional RRT algorithm, the RRT* algorithm obtains the locally optimal path by constantly updating the old parent node, so the global path tends to be optimal. However, RRT and RRT* algorithms perform pure exploration, which can cause a prolonged rate of convergence in a high-dimensional environment. Moreover, the randomly sampled characteristics of RRT and RRT* algorithms do not make the generated path smooth enough. The essential advantage of the RRT and RRT* is that the generated global path tends to be optimal and mainly effective to the 2D environment. Consequently, RRT and RRT* are used for mobile robot path-planning tasks. Due to the nature of RRT-based algorithms, each node is generated by a simple random algorithm, and the convergence rate is not acceptable when the environment is enormous. There are many unusable nodes created by the algorithm. This paper presents the process of mapping the global environment for the RRT* algorithm to avoid unusable nodes and reduce the computational time.

Researchers proposed different ways to handle the mentioned situation to deal with both obstacle region complexity and massive environments. The Hybrid Bidirectional Rapidly exploring Random Trees Algorithm was proposed by Xue et al. [25]. The algorithm works with a combination of trees, starting from both the starting point and the goal point. After the iterations, the tree from the starting point and the tree from the end point will be connected. The result of the experiment showed that the algorithm could complete the path searching rapidly, the efficiency of the algorithm was improved, and the length of the path was reduced. However, both trees need to be stored in the memory.

The Density Avoid Sampling (DAS) Technique was proposed by S. Khanmohammadi and Mahdizadeh [26]. In traditional path-planning methods, the environment's layout is not used during the sampling procedure (which makes extra samples that are useless) and this kind of behavior makes the methods more computationally demanding. On the other hand DAS-RRT is based on the way the branches of tree are growing, making the method more sophisticated.

The path smoothness of the result from RRT* is usually low. The commonly used method to improve the smoothness and reduce node number is proposed for post-processing. The Bacterial Mutation operator of Bacterial Evolutionary Algorithm was selected [27]. Evolutionary Algorithms are nature-inspired optimization techniques which are often suitable for global optimization in single-objective and multi-objective problems, even in the case of linear or non-linear constraints. The commonly used evolutionary algorithms for path planning are Particle Swarm Optimization (PSO) [28,29] and Ant Colony Optimization (ACO) [30,31]. Bacterial Mutation with Local Search algorithms is one of the operators in the Bacterial Memetic Algorithm (BMA) [32,33]. Using only BMA can lead to the solution, but in this research, the Bacterial Mutation and Node Deletion operators are used in post-processing for the purpose of path smoothness optimization.

Bacterial Mutation is a bacterial evolutionary operator. This operator can optimize the individuals one by one to improve them. The result of path planning from the proposed RRT* algorithm is defined as initial bacterium. After the bacterial mutation operator, the path length tends to be optimized. The number of nodes can be reduced by using a local search algorithm such as the Node-Deletion operator. The final result from the proposed path-planning algorithm is better both in terms of computational time and optimized path length.

2. Environment and Mapping

The TurtleBot3-Burger is used in the simulation, as depicted in Figure 1. A TurtleBot is an omnidirectional mobile robot with two wheels and two DC motors. A TurtleBot is a Robot Operating System (ROS) platform robot. A robot can perform forward and backward movements, left turns, and right turns. The Gazebo simulation platform is used for performing the robot's motion, following the path-planning solution from a proposed algorithm. The robot model and global environment are created using Unified Robotic Description Format (URDF) file format. The XML file format is used in ROS to describe all elements in the simulation called Gazebo world.

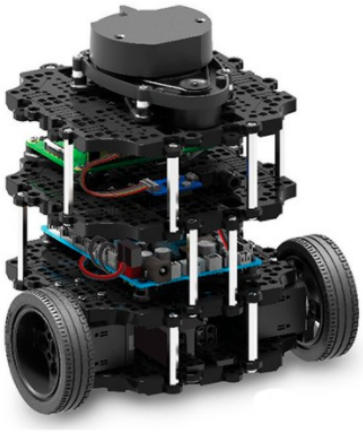


Figure 1. The TurtleBot 3 Model Name: Burger.

The map is defined by the data structure $M \times N$, where there are two fields of data in each cell. The first one is the cell number, which can be identified from the left column to the right column and from the bottom to the top row, as depicted in Figure 2. Furthermore, the second field is the 2D coordinates of the cell. The coordinates represent the Cartesian coordinate and its value is an integer only. In Figure 2 the white color represents that it is

feasible for this cell to pass through it. On the other hand, the black color represents the obstacle region. The robot cannot pass through it. The obstacle regions are cell numbers 2, 8, 13, 14, and 21, whose coordinates are (1,0), (2,1), (2,2), (3,2), and (0,4), respectively.

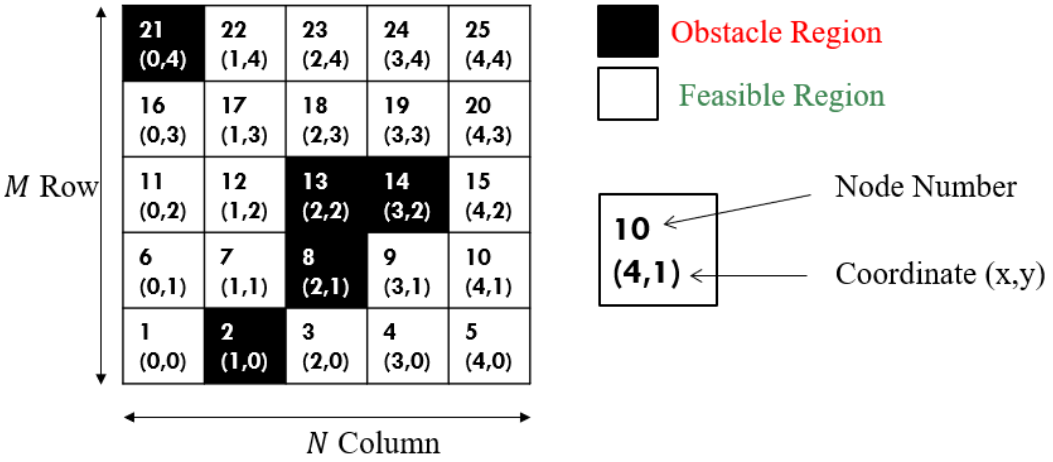


Figure 2. The global environment.

3. Traditional RRT* Algorithm, Bacterial Mutation, and Local Search Operators

3.1. RRT Algorithm*

RRT* is a random sampling tree structure search algorithm. The illustration of the RRT* algorithm is depicted in Figure 3; the algorithm is presented in Algorithm 1. In the initialization, the starting node (q_{start}) and the goal node (q_{goal}) are determined in the global environment. The number of maximum iterations is set as the $MaxItertion$. The maximum width and height of the global environment are defined as X_{max} and Y_{max}, respectively. The iteration starts by picking a random node on the map as q_{rand} and finding the closest node to q_{rand} as q_{near}. If the way between the random node (q_{rand}) and the closest node (q_{near}) is collision-free then steering is performed from q_{near} to q_{rand} by constant distance B. After steering, the new node will be q_{new}. Then, the algorithm finds the minimum cost path from an existing node q_{min} to q_{new} in the optimized radius of R, and then q_{new} is added to the Tree. The red dashed line represents that the q_{min} node is the parent node of the q_{new} node in the current iteration. In each iteration, the RRT* algorithm finds the minimum path to each node that can lead to the optimal result in path cost. The algorithm will stop when the newly added node q_{new} is close to the goal node, more minor than the expected value D. Finally, when the tree is built at the end of the maximum number of iterative nodes, the algorithm searches backwards from the goal node to the starting node to find the optimal cost path.

Algorithm 1 RRT* Algorithm

Initialize q_{start} and q_{goal}
for $i < MaxIteration$ **do**
  $q_{rand} \leftarrow$ random node $(0 - X_{max}, 0 - Y_{max})$
  $q_{near} \leftarrow$ find nearest node from Tree
  if obstacle free between q_{near} and q_{new} **then**
    $q_{new} \leftarrow$ steer from q_{near}
    Find minimum cost from q_{min} and q_{new} in radius of R
    Add q_{new} to Tree
  if distance between q_{new} and $q_{goal} \leq D$ **then**
    Stop iteration
Return Tree

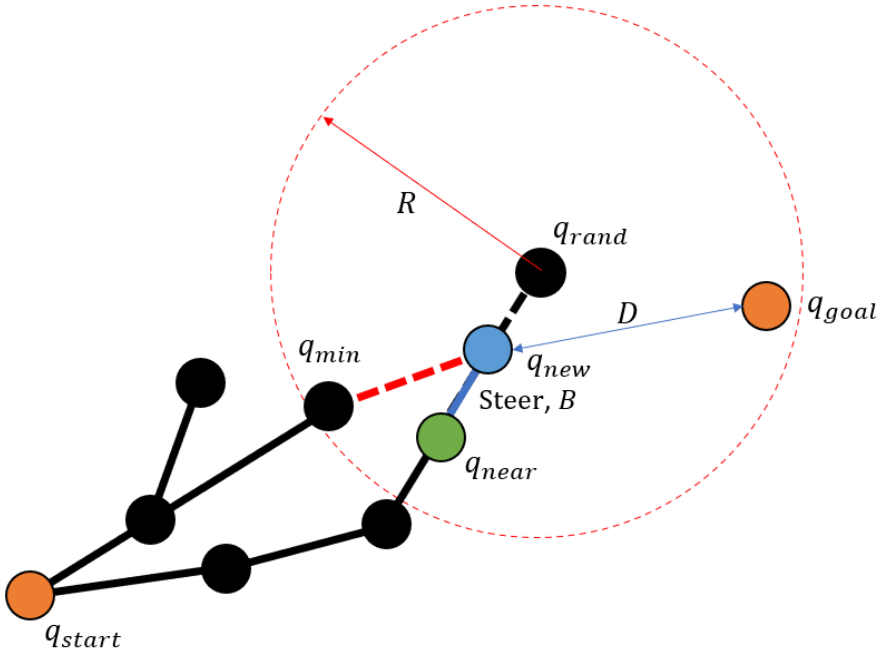


Figure 3. Illustration of RRT* Algorithm.

3.2. Bacterial Mutation Operator

A bacterial mutation is applied to the first solution from the RRT* algorithm. The solution from the RRT* algorithm is referred to as a bacterium. First, N_{clones} copies (clones) of the bacterium are created. Then, a random segment of length l_{bm} is mutated in each clone except one clone, which is left un-mutated. After mutating the same segment in the clones, each clone is evaluated using the cost function. The clone with the best evaluation result transfers the mutated segment to the other clones. These three operation steps (mutation of the clones, selection of the best clone, transfer of the mutated segment) are repeated until each segment of the bacterium has been mutated once [33]. The cost function of each bacterium is defined as

$$Cost(ind) = L(ind) + Penalty \cdot Coll(ind) + S \cdot Turn(ind), \quad (1)$$

where $L(ind)$ is the total path length from starting point to goal point, which is calculated by the Euclidean distance for the neighboring cell, and the $Coll(ind)$ is the binary value for collision (0 represents no collision or 1 represents collision found). The $Penalty$ is the penalty parameter for making the collision path get a higher cost function. The $Turn(ind)$ is the number of robot turns. S is the parameter that reflects the smoothness of the path.

3.3. Node Deletion Operator

The node deletion is performed after the Bacterial Mutation operator. The deletion operator is illustrated in Figure 4. The improper nodes are deleted from the bacterium, whose removal has the better benefit in path length. From the triangle inequality, the sum of the lengths of any two sides must be greater than the length of the remaining side, as shown in Figure 5. If no such point is found in the sense that the bacterium will be worse after removal, the operation will not be performed. In Figure 4 the red point is deleted because, after its deletion, the new subpath represented by a dashed line will be better than the old subpath.

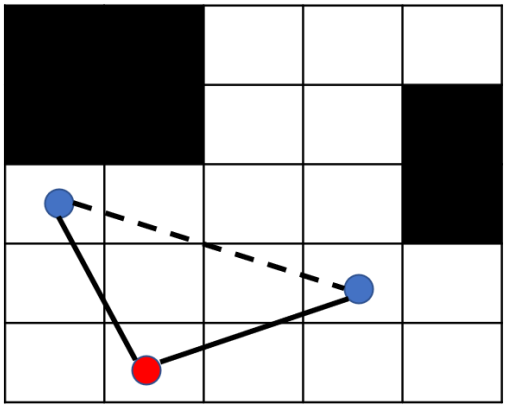


Figure 4. Node Deletion.

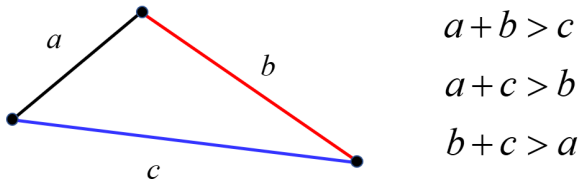


Figure 5. Triangle Inequality.

4. Proposed RRT* Algorithm

4.1. Unusable Nodes

The unusable nodes are the nodes that fall into the obstacle regions, as shown in Figure 6. The random node q_{rand} must be located in the free space region to steer the tree branch from the nearest node q_{near}. The number of generated unusable nodes is over 40% of RRT*. The main idea to improve the efficiency of the RRT* algorithm is to reduce the unusable nodes from the iteration and let the iteration flow continue with usable nodes.

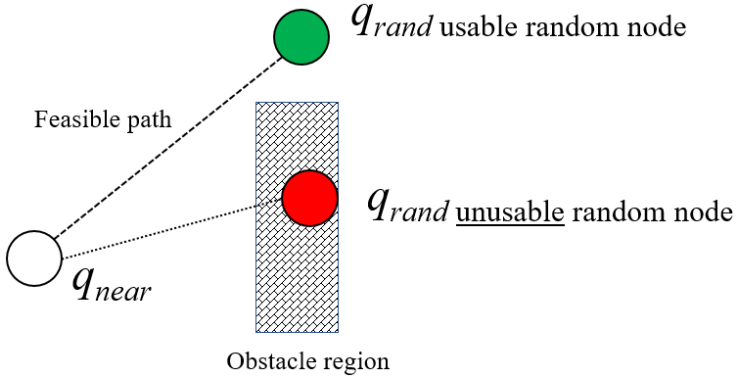


Figure 6. Unusable Node.

This research would like to present a method to deal with an existing map from the initialization and process it to remove obstacle region coordinates, making the map less resource-demanding. The post-processing algorithm then smooths the trajectory to minimize the number of points in the path.

4.2. Improved Algorithm

First, the global environment matrix is stretched from $M \times N$ matrix to a row vector of size $M \cdot N$. There are combinations of both feasible and obstacle regions. Then, the obstacle region cells, i.e., cell numbers 2 (with the coordinates: (1,0)) and 8 (having coordinates: (2,1)) are deleted from the row vector. The result of this algorithm is that only the nodes

in the feasible region will appear in the row vector and be used in the random process. The visualization of this proposed method is shown in Figure 7. The algorithm for mapping the feasible region is shown in Algorithm 2. The algorithm starts by reading the global environment from the file as *Map*. Then, the *Map* is stretched from a matrix to a row vector *randMap*. The iteration for deleting the obstacle region nodes starts from the first to the last index of a *randMap* vector. The advantage of the proposed mapping algorithm is that it can also deal with irregular-shaped obstacle regions.

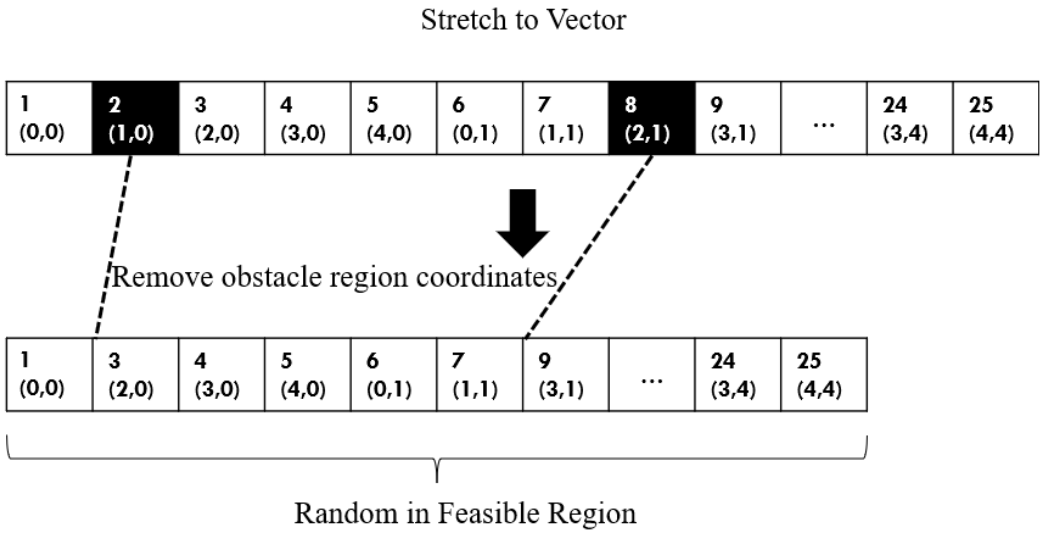


Figure 7. Feasible Region Mapping for Node Random Process in RRT*.

Algorithm 2 Feasible region mapping for use in RRT*.

Map = **ReadMap** from file (.bmp)
randMap = **StretchMap** from maxtrix to row vector
for $i <$ Length(*randMap*) **do**
  if *randMap*(i) is an obstacle region **then**
    Delete *randMap*(i) from *randMap* vector
End

4.3. Post-Processing Algorithms

The above-mentioned algorithm is combined with the Bacterial Mutation and Node Deletion operators to optimize the path-planning result from the RRT* algorithm. First, the series of coordinates will be defined as the single bacterium, as depicted with a green path in Figure 8. There are seven coordinates in this bacterium. The Bacterial Mutation operator is applied to the bacterium. The algorithm will generate the clones and mutate each cell. The selected cell of each clone for the mutation will be replaced with a random cell in the global environment. Equation (1) is used to compute the overall cost of each clone. The clone that has the minimum cost will be kept untouched and will be used for cloning for the next generation. After the last generation of bacterial mutation operator, the mutated bacterium is represented by the red dashed lines in Figure 8. Finally, the Node Deletion algorithm will be used for deleting the unnecessary nodes. The result tends to be optimized as shown with solid blue lines in Figure 8.

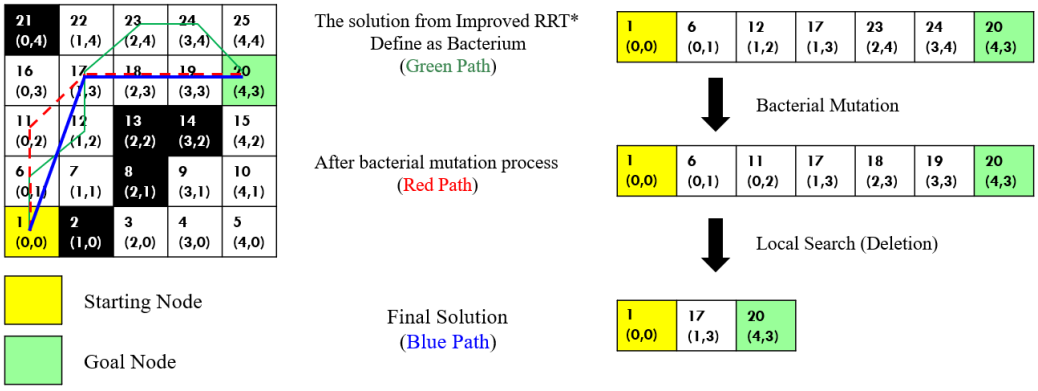
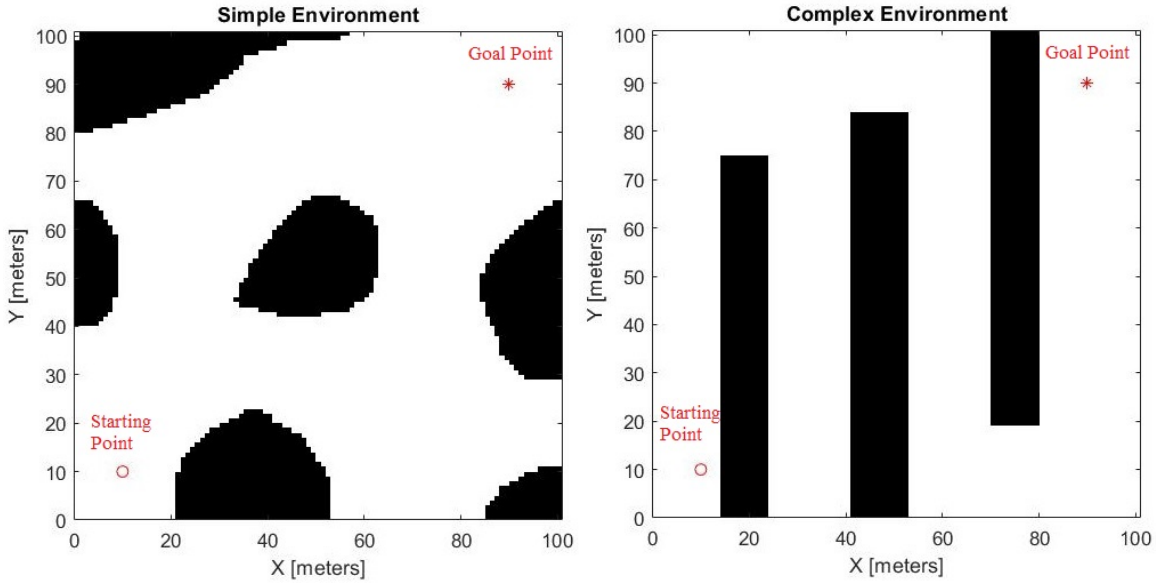


Figure 8. Post-processing.

5. Experimental Results

The MATLAB programming environment was used to realize the experiments. The proposed RRT* algorithm was tested in two types of global environments: in a simple environment, depicted in Figure 9a, and in a complex environment, depicted in Figure 9b. The red circle represents the starting point and the red star represents the goal point. There was only one obstacle between the line of sight from the starting point to the goal point in the simple environment. Furthermore, in the complex environment, there were lots of obstacle regions between them. The number of iterations, unusable nodes, overall path length, and computational time were collected from the experiments.



(**a**) (**b**)

Figure 9. Global Environment. (**a**) Simple environment; (**b**) complex environment.

5.1. Parameter Setting

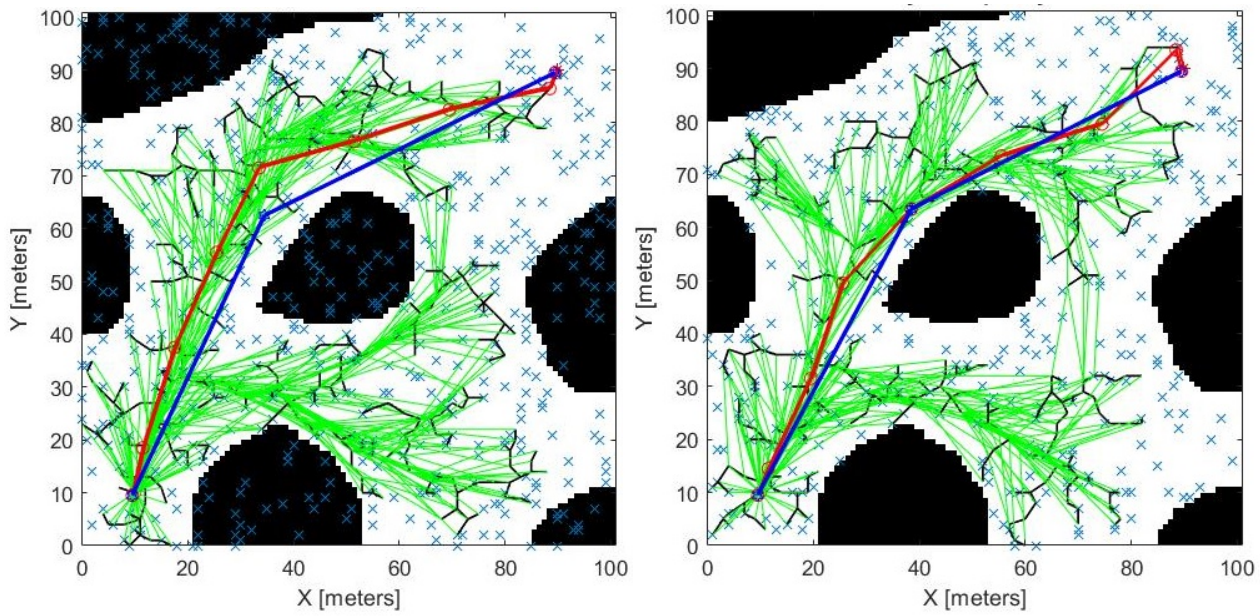
The size of the global environment was set to 100 × 100 cells. The RRT* algorithm continued to iterate until it reached the maximum iteration of 5000. The steering distance from the nearest node to the random node was set to three. For optimized path length, the optimized radius for searching the parent node of the new node in the tree was 20. For the Bacterial Mutation, the number of clones was set to 5. The penalty for the collision path was determined as 1000, and the S for smoothness of turns was defined as 0.1. In the Node Deletion algorithm, all coordinates are required to be checked in the feasible path. The number of nodes for the deletion process was equal to the size of a bacterium. All parameters are shown in Table 1.

Table 1. Parameter setting.

Algorithm	Parameter	Value
RRT*	Maximum Iteration	5000
	Map Size ($X_{max} \times Y_{max}$)	100×100
	Steering Distance (B)	3
	Optimized Radius (R)	20
Bacterial Mutation	N_{clones}	5
	Number of Generations	Size of Bacterium
	Penalty	1000
	S	0.1
Deletion	Iteration	Size of Bacterium

5.2. Results

The path-planning results of the simple environment are shown in Figure 10. The green lines represent the branches of the Tree from traditional RRT* (Figure 10a) and the Improved-RRT* (Figure 10b) processes. The red lines and circles represent the solution from RRT* and proposed RRT* algorithms, and the blue lines and dots represent the optimized path-length and nodes from the Bacterial Mutation and the Node Deletion operator. The path from the traditional RRT* is longer than the optimized one. The number of nodes was reduced from nine nodes to three nodes. The result from the proposed RRT* algorithm is depicted in Figure 10b. The overall path length is shorter than the traditional one. The number of nodes was reduced from nine nodes to three nodes. The simulation tested the repeatability of three algorithms as follows: the traditional RRT*, the traditional RRT* with post-processing, and the proposed RRT* with post-processing ten times in a row. The recorded results are shown in Table 2.



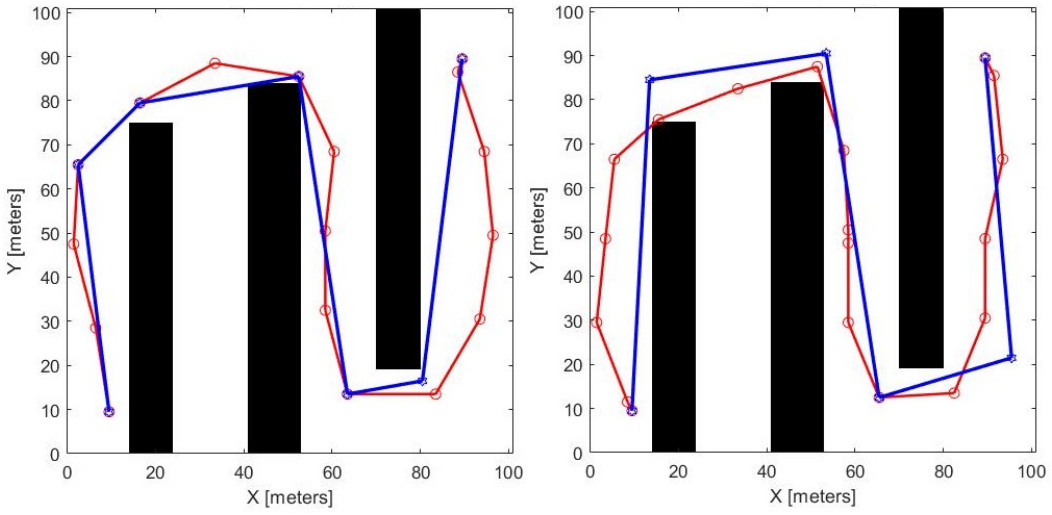
(**a**) (**b**)

Figure 10. Simple global environment results. (**a**) Traditional RRT* (red) Traditional RRT* with bacterial mutation and Node Deletion algorithm (blue); (**b**) proposed RRT* (red); proposed RRT* with Bacterial Mutation and Node Deletion algorithm (blue).

Table 2. Results [BM = Bacterial Mutation operator, ND = Node Deletion operator].

Result	Traditional RRT*	Traditional RRT* with BM and ND	Proposed RRT* with BM and ND
Simple Environment			
Number of Iterations	609	609	353
Path Length	127	120	120
Number of Unusable Nodes	183	183	35
Computation Time (s)	1.701	2.520	1.808
Number of Result Nodes	9	3	3
Complex Environment			
Number of Iterations	2791	2791	1816
Path Length	284	274	266
Number of Unusable Nodes	1411	1411	659
Computation Time (s)	13.382	16.152	12.093
Number of Result Nodes	16	7	6

From Table 2, it can be seen that the difference in computation time in the case of the complex environment is higher than in the case of the simple environment. The proposed algorithm shows better performance in a complex environment. The results for the complex environment are shown in Figure 11. The results show only the solution from the RRT* algorithms and post-processing from the Bacterial Mutation and the Node Deletion algorithms. The result from traditional RRT* can perform the collision-free path from start to goal points, as in Figure 11a, the path length is still higher. After using the post-processing algorithm, the number of nodes from the solution is reduced from 16 to 7 nodes. Compared with the result from the proposed algorithm, the number of nodes is reduced from 19 nodes to 6 nodes, as depicted in Figure 11b. In terms of overall path length, the path can be reduced by the post-processing algorithm with Bacterial Mutation and Node Deletion operators. In the simple environment, the path length from the traditional RRT* is reduced from 127 to 120. Furthermore, in the complex environment, the overall path length is reduced from 284 to 274 and 266 for traditional and proposed RRT* algorithms with bacterial mutation and Node Deletion operators, respectively. The conclusion is that the proposed RRT* algorithm with post-processing using the Bacterial Mutation and the Node Deletion operators can reduce computational times and overall path-length compared with the traditional RRT* and the traditional RRT* with post-processing.

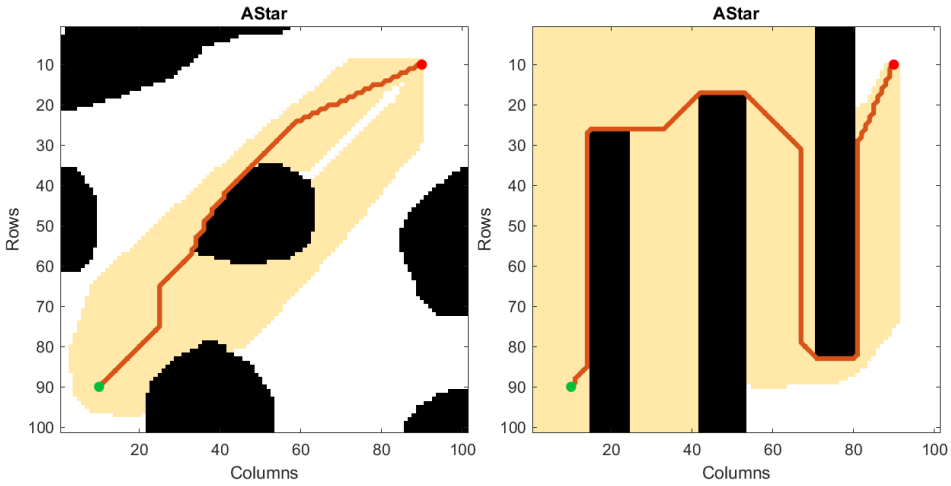


(**a**) (**b**)

Figure 11. Complex global environment Results. (**a**) Traditional RRT* (red); traditional RRT* with Bacterial Mutation and Node Deletion algorithm (blue); (**b**) proposed RRT* (red); proposed RRT* with Bacterial Mutation and Node Deletion algorithm (blue).

5.3. Comparing the Result with Commonly Used Algorithms

In this section, the path-planning results from the A* are also presented. The results from the A* algorithm are depicted in Figure 12a. The green and red points represent the starting and goal points, respectively, and the orange lines represent the resulting path. The yellow area represents the searching area of the algorithm. In the simple environment case, the path length result is 123. The resulting path's smoothness of the A* is not satisfying, as seen in Figure 12b. The path length from the A* is also higher than the proposed one. In a complex environmental situation, the overall path length of the A* is 267, which is a little higher than the result from the proposed algorithm. There are eight turning points compared to the proposed algorithm's four, as seen in Figure 11b.



(**a**) (**b**)

Figure 12. A* Algorithm Results. (**a**) Path-planning result from A* algorithm in simple environment; (**b**) path-planning result from A* algorithm in complex environment.

5.4. Simulation Result in Gazebo

To test the final solution from the proposed algorithm, a simulation using the Robot Operating System in collaboration with MATLAB programming was conducted. The path-planning result from post-processing was sent to TurtleBot3-Burger in Gazebo world by MATLAB programming via Robot Operating System. The MATLAB program received the desired points coordinate and created the wheel's velocity command via ROS to TurtleBot3-Burger. The commands included the forward and backward directional velocity and angular velocity. The result shows that TurtleBot3-Burger can perform motion planning from a path-planning result with collision-free motion from the starting point to the target point. The example of simulation is depicted in Figure 13. The dashed line represents the actual path of the robot.

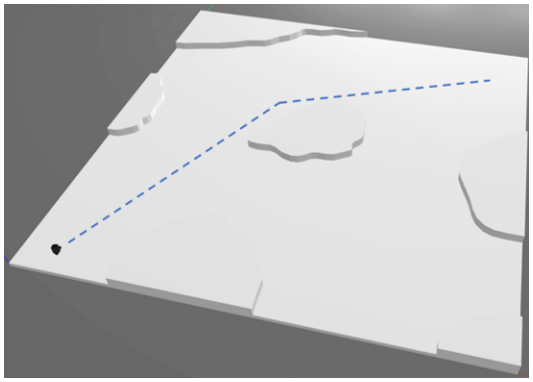


Figure 13. Example of simulation path in Gazebo using TurtleBot3-Burger.

6. Discussion

A feasible region-mapping algorithm was proposed to reduce the number of unusable nodes for the coordinate-exploring section of the RRT* algorithm. As seen in the results, the number of unused nodes was reduced. The applied changes lead the algorithm to explore the entire environment more efficiently. After obtaining the first solution from the proposed RRT* algorithm, the Bacterial Mutation and Node Deletion algorithms were applied to smoothen and minimize the path length. Comparing the proposed algorithm with other algorithms in Table 2, the proposed algorithm demands less computational power.

During the conducted experiments, our proposed algorithm was compared to a traditional RRT* algorithm. According to Table 2, the proposed algorithm returns a less complex path that is also smoother. During the experiments, a traditional method—the A*—was also compared to our proposed algorithm, and the comparison also resulted in favor of the proposed method.

7. Conclusions

This article aimed to optimize the path planning of mobile robots in large and complex environments. We proposed an improved RRT* algorithm by adding the pre-processing with feasible region mapping and post-processing with Bacterial Mutation and Node Deletion operators. The pre-processing algorithm can lead to reduced computation time by reducing the number of unusable nodes. The post-processing algorithm can optimize the overall path length and number of nodes to represent a smooth path. Compared with references to traditional RRT* and traditional RRT* with post-processing, the generated path from the proposed algorithm is shorter and smoother. The path result from the proposed method was tested in the simulation by the TurtleBot3-Burger on Gazebo in the Robot Operating System.

In future work, our proposed algorithm can be extended to deal with more complex environments (e.g., with dynamic obstacles). In this research, the feasible mapping algorithm was used only for pre-processing. The concept of feasible mapping can be applied during each iteration by adding the possible coordinates to iterate through or deleting coordinates in the obstacle's regions. The size of the feasible mapping vector will change depending on the dynamic obstacles that the robot is facing.

Author Contributions: Conceptualization, A.L. and J.B.; methodology, A.L. and J.B.; software, A.L.; validation, A.L.; formal analysis, A.L.; investigation, A.L.; resources, A.L.; data curation, A.L.; writing—original draft preparation, A.L. and J.B.; writing—review and editing, A.L. and J.B.; visualization, A.L.; supervision, J.B.; project administration, A.L.; funding acquisition, J.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Botzheim, J. Cognitive Robotics. *Electronics* **2021**, *10*, 1510. [CrossRef]
2. Gao, Y.; Hu, T.; Wang, Y.; Zhang, Y. Research on the Path Planning Algorithm of Mobile Robot. In Proceedings of the 2021 13th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA), Beihai, China, 16–17 January 2021; pp. 447–450. [CrossRef]
3. Karur, K.; Sharma, N.; Dharmatti, C.; Siegel, J.E. A Survey of Path Planning Algorithms for Mobile Robots. *Vehicles* **2021**, *3*, 448–468. [CrossRef]
4. Zheyi, C.; Bing, X. AGV Path Planning Based on Improved Artificial Potential Field Method. In Proceedings of the 2021 IEEE International Conference on Power Electronics, Computer Applications (ICPECA), Shenyang, China, 22–24 January 2021; pp. 32–37. [CrossRef]
5. Li, H. Robotic Path Planning Strategy Based on Improved Artificial Potential Field. In Proceedings of the 2020 International Conference on Artificial Intelligence and Computer Engineering (ICAICE), Beijing, China, 23–25 October 2020; pp. 67–71. [CrossRef]

6. Ju, C.; Luo, Q.; Yan, X. Path Planning Using an Improved A-star Algorithm. In Proceedings of the 2020 11th International Conference on Prognostics and System Health Management (PHM-2020 Jinan), Jinan, China, 23–25 October 2020; pp. 23–26. [CrossRef]
7. Liu, Y.; Gao, P.; Zheng, C.; Tian, L.; Tian, Y. A Deep Reinforcement Learning Strategy Combining Expert Experience Guidance for a Fruit-Picking Manipulator. *Electronics* **2022**, *11*, 311. [CrossRef]
8. LaValle, S.M. *Rapidly-Exploring Random Trees: A New Tool for Path Planning*; Springer: London, UK, 1998.
9. Karaman, S.; Frazzoli, E. Incremental Sampling-based Algorithms for Optimal Motion Planning. *Robot. Sci. Syst. VI* **2010**, *104*, 34–53. [CrossRef]
10. Karaman, S.; Frazzoli, E. Sampling-based Algorithms for Optimal Motion Planning. *Int. J. Robot. Res. IJRR* **2011**, *30*, 846–894. [CrossRef]
11. Wang, H.; Li, G.; Hou, J.; Chen, L.; Hu, N. A Path Planning Method for Underground Intelligent Vehicles Based on an Improved RRT* Algorithm. *Electronics* **2022**, *11*, 294. [CrossRef]
12. Yang, R.; Cai, P.; Wang, L. Comparison of Strategies for Optimizing Bi-RRT* on Mobile Robots. In Proceedings of the 2021 IEEE International Conference on Computer Science, Artificial Intelligence and Electronic Engineering (CSAIEE), Virtual, SC, USA, 20–22 August 2021; pp. 328–334. [CrossRef]
13. Ghosh, D.; Nandakumar, G.; Narayanan, K.; Honkote, V.; Sharma, S. Kinematic Constraints Based Bi-directional RRT (KB-RRT) with Parameterized Trajectories for Robot Path Planning in Cluttered Environment. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 8627–8633. [CrossRef]
14. Wu, X.; Xu, L.; Zhen, R.; Wu, X. Biased Sampling Potentially Guided Intelligent Bidirectional RRT* Algorithm for UAV Path Planning in 3D Environment. *Math. Probl. Eng.* **2019**, *2019*, 5157403. [CrossRef]
15. Živojević, D.; Velagić, J. Path Planning for Mobile Robot using Dubins-curve based RRT Algorithm with Differential Constraints. In Proceedings of the 2019 International Symposium ELMAR, Zadar, Croatia, 23–25 September 2019; pp. 139–142. [CrossRef]
16. Varghese, A.M.; Jisha, V.R. Motion Planning and Control of an Autonomous Mobile Robot. In Proceedings of the 2018 International CET Conference on Control, Communication, and Computing (IC4), Thiruvananthapuram, India, 5–7 July 2018; pp. 17–21. [CrossRef]
17. Qi, J.; Yang, H.; Sun, H. MOD-RRT*: A Sampling-Based Algorithm for Robot Path Planning in Dynamic Environment. *IEEE Trans. Ind. Electron.* **2021**, *68*, 7244–7251. [CrossRef]
18. Liu, G.; Jiang, Y. Research on Dynamic Trajectory Planning of Collaborative Robots Base on RRT-RV Algorithm. In Proceedings of the 2018 IEEE 4th Information Technology and Mechatronics Engineering Conference (ITOEC), Chongqing, China, 14–16 December 2018; pp. 1020–1023. [CrossRef]
19. Li, S.; Zhao, D.; Sun, Y.; Yang, J.; Wang, S. Path Planning Algorithm Based on the Improved RRT-Connect for Home Service Robot Arms. In Proceedings of the 2021 IEEE International Conference on Intelligence and Safety for Robotics (ISR), Tokoname, Japan, 4–6 March 2021; pp. 403–407. [CrossRef]
20. Khan, A.T.; Li, S.; Kadry, S.; Nam, Y. Control Framework for Trajectory Planning of Soft Manipulator Using Optimized RRT Algorithm. *IEEE Access* **2020**, *8*, 171730–171743. [CrossRef]
21. Wang, Z.; Chang, J.; Li, B.; Wang, C.; Liu, C. Application of Improved Rapidly-exploring Random Trees (RRT) algorithm for Obstacle Avoidance of Snake-like Manipulator. In Proceedings of the 2020 IEEE International Conference on Mechatronics and Automation (ICMA), Beijing, China, 13–16 October 2020; pp. 490–495. [CrossRef]
22. Zhang, D.; Xu, Y.; Yao, X. An Improved Path Planning Algorithm for Unmanned Aerial Vehicle Based on RRT-Connect. In Proceedings of the 2018 37th Chinese Control Conference (CCC), Wuhan, China, 25–27 July 2018; pp. 4854–4858. [CrossRef]
23. Chen, J.; Yu, J. An Improved Path Planning Algorithm for UAV Based on RRT. In Proceedings of the 2021 4th International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE), Changsha, China, 26–28 March 2021; pp. 895–898. [CrossRef]
24. Yafei, L.; Anping, W.; Qingyang, C.; Yujie, W. An Improved UAV Path Planning method Based on RRT-APF Hybrid strategy. In Proceedings of the 2020 5th International Conference on Automation, Control and Robotics Engineering (CACRE), Dalian, China, 19–20 September 2020; pp. 81–86. [CrossRef]
25. Xue, Y.; Zhang, X.; Jia, S.; Sun, Y.; Diao, C. Hybrid bidirectional rapidly-exploring random trees algorithm with heuristic target graviton. In Proceedings of the 2017 Chinese Automation Congress (CAC), Jinan, China, 20–22 October 2017; pp. 4357–4361. [CrossRef]
26. Khanmohammadi, S.; Mahdizadeh, A. Density Avoided Sampling: An Intelligent Sampling Technique for Rapidly-Exploring Random Trees. In Proceedings of the 2008 Eighth International Conference on Hybrid Intelligent Systems, Barcelona, Spain, 10–12 September 2008; pp. 672–677. [CrossRef]
27. Nawa, N.E.; Furuhashi, T. Fuzzy system parameters discovery by bacterial evolutionary algorithm. *IEEE Trans. Fuzzy Syst.* **1999**, *7*, 608–616. [CrossRef]
28. Qing, L.; Chao, Z.; Yinmei, X.; Yixin, Y. Path planning of mobile robots based on specialized genetic algorithm and improved particle swarm optimization. In Proceedings of the 31st Chinese Control Conference, Hefei, China, 25–27 July 2012; pp. 7204–7209.
29. Gou, P.; Jiang, B. Research on Path planning of Three-Dimensional UAV Based on Levy Flight Strategy and Improved Particle Swarm Optimization Algorithm. In Proceedings of the 2020 7th International Conference on Information Science and Control Engineering (ICISCE), Changsha, China, 18–20 December 2020; pp. 1199–1203. [CrossRef]

30. Praserttaweelap, R.; Kiatwanidvilai, S. Optimal A* Path Planning with Ant Colony Optimization on Multi-Robot Task Allocation for Manufacturing Model. In Proceedings of the 2021 IEEE 8th International Conference on Industrial Engineering and Applications (ICIEA), Chengdu, China, 23–26 April 2021; pp. 137–141. [CrossRef]
31. Lee, M.; Yu, K. Dynamic Path Planning Based on an Improved Ant Colony Optimization with Genetic Algorithm. In Proceedings of the 2018 IEEE Asia-Pacific Conference on Antennas and Propagation (APCAP), Auckland, New Zealand, 5–8 August 2018; pp. 1–2. [CrossRef]
32. Botzheim, J.; Cabrita, C.; Kóczy, L.T.; Ruano, A.E. Fuzzy rule extraction by bacterial memetic algorithms. *Int. J. Intell. Syst.* **2009**, *24*, 312–339. [CrossRef]
33. Botzheim, J.; Toda, Y.; Kubota, N. Bacterial memetic algorithm for offline path planning of mobile robots. *Memetic Comput.* **2012**, *4*, 73–86. [CrossRef]