actuators

MDPI

Article

Route Planning for Autonomous Mobile Robots Using a Reinforcement Learning Algorithm

Fatma M. Talaat [1], Abdelhameed Ibrahim [2,*], El-Sayed M. El-Kenawy [3], Abdelaziz A. Abdelhamid [4,5], Amel Ali Alhussan [6], Doaa Sami Khafaga [6] and Dina Ahmed Salem [7]

1 Machine Learning and Information Retrieval Department, Faculty of Artificial Intelligence, Kafrelsheikh University, Kafrelsheikh 33511, Egypt
2 Computer Engineering and Control Systems Department, Faculty of Engineering, Mansoura University, Mansoura 35516, Egypt
3 Department of Communications and Electronics, Delta Higher Institute of Engineering and Technology, Mansoura 35111, Egypt
4 Department of Computer Science, College of Computing and Information Technology, Shaqra University, Shaqra 11961, Saudi Arabia
5 Department of Computer Science, Faculty of Computer and Information Sciences, Ain Shams University, Cairo 11566, Egypt
6 Department of Computer Sciences, College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University, P.O. Box 84428, Riyadh 11671, Saudi Arabia
7 Department of Computer and Software Engineering, Faculty of Engineering, Misr University for Science and Technology (MUST), 6th of October City 3236101, Egypt
* Correspondence: afai79@mans.edu.eg

Abstract: This research suggests a new robotic system technique that works specifically in settings such as hospitals or emergency situations when prompt action and preserving human life are crucial. Our framework largely focuses on the precise and prompt delivery of medical supplies or medication inside a defined area while avoiding robot collisions or other obstacles. The suggested route planning algorithm (RPA) based on reinforcement learning makes medical services effective by gathering and sending data between robots and human healthcare professionals. In contrast, humans are kept out of the patients' field. Three key modules make up the RPA: (i) the Robot Finding Module (RFM), (ii) Robot Charging Module (RCM), and (iii) Route Selection Module (RSM). Using such autonomous systems as RPA in places where there is a need for human gathering is essential, particularly in the medical field, which could reduce the risk of spreading viruses, which could save thousands of lives. The simulation results using the proposed framework show the flexible and efficient movement of the robots compared to conventional methods under various environments. The RSM is contrasted with the leading cutting-edge topology routing options. The RSM's primary benefit is the much-reduced calculations and updating of routing tables. In contrast to earlier algorithms, the RSM produces a lower AQD. The RSM is hence an appropriate algorithm for real-time systems.

Keywords: autonomous robots; routing algorithm; collision avoidance; reinforcement learning; mobile robots

Citation: Talaat, F.M.; Ibrahim, A.; El-Kenawy, E.-S.M.; Abdelhamid, A.A.; Alhussan, A.A.; Khafaga, D.S.; Salem, D.A. Route Planning for Autonomous Mobile Robots Using a Reinforcement Learning Algorithm. *Actuators* **2023**, *12*, 12. https://doi.org/10.3390/act12010012

Academic Editors: Zhuming Bi and Jih-Gau Juang

Received: 18 November 2022
Revised: 22 December 2022
Accepted: 23 December 2022
Published: 26 December 2022

1. Introduction

Current smart applications and environments (such as smart homes, intelligent industrial systems, and healthcare systems) heavily rely on cloud computing for communication and collaboration among connected smart devices. Smart gadgets are spread, whereas cloud servers and data centers are largely centralized. Thus, the response time of data transmission between the cloud and smart devices is a crucial issue, particularly for applications with severe delay requirements, such as saving human lives in emergencies. The entire fog platform has not yet been fully built, but new fog computing technological paradigms have lately been offered to address this challenge [1].

The main goal of the fog platform is to increase the end users' access to real-time interaction and location-based services by bringing processing power from the far-off cloud closer to them. The local processing power of fog significantly reduces the data burden in the cloud. Nowadays, several mobile robots can work alongside people and even replace humans in most professions. As they provide comfort, they might become involved in people's lives. They are used in the majority of industrial domains because they can reduce the labor required in manufacturing. They can also complete jobs that are impossible for humans to complete, save lives in emergencies, and be used in healthcare systems [2].

By the middle of December 2019, Wuhan, China, had an outbreak of a new oronavirus-induced pneumonia, which subsequently caught the attention of the entire globe [3]. The World Health Organization (WHO) declared the global COVID-19 outbreak a public health emergency of international concern on 30 January 2020 [4]. Healthcare workers are becoming infected as COVID-19 infections rise; up to 10% of them have been sick in some nations. Even though the number of COVID-19 cases worldwide is still being regularly tracked, experts and public health officials who are preparing a response to the virus's outbreak are still lacking crucial data regarding the precise number of physicians, nurses, and other healthcare professionals who have tested positive because they are at the greatest risk of contracting the virus and dying as a result [5].

It may seem absurd to worry that the number of healthcare professionals who will pass away will increase, but frontline healthcare workers are passing away in countries such as China and Italy. In Washington, dozens of medical staff at a nursing center tested positive for COVID-19, and in Pittsfield, Massachusetts, more than 160 of the staff of Berkshire Medical Center were quarantined due to the virus exposure. In addition, in Washington, at least 200 nurses have been removed from their duties and placed in isolation because of the lack of testing. "At least 2629 healthcare workers—roughly 8.3 percent of all cases in Italy—have contracted COVID-19". It is clear that more cases of medical employees and healthcare workers becoming infected are emerging daily [6]. It is becoming very evident that there are increasing numbers of incidents of medical staff and healthcare workers becoming infected each day. In these kinds of situations, humans are shielded from infection by an autonomous and fully automated robot systems.

Mobile robots are utilized in a variety of domains to carry out vital duties without the assistance of a human operator. These domains include military operations, industrial automation, and rescue operations [7]. The planning of the routes that mobile robots will take is one of the crucial factors that must be considered in order to maximize their effectiveness. The process of route planning can be stated as follows: in any working environment, a mobile robot chooses either an optimal or a suboptimal route that will take the robot from a beginning state to the goal state based on established performance criteria [8]. An effective route planning algorithm will, on the one hand, shorten the amount of time required to reach the objective state, and on the other hand, it will prevent the robot from experiencing unnecessary wear and tear. The significance of the function that route planning algorithms plays has contributed to the topic's recent rise in popularity within the academic research community [9,10].

First, in order to achieve both efficiency and accuracy in route planning, it is necessary to create an environment model that will allow for a better understanding of the environmental parameters [11]. The level of complexity involved in route planning will be greatly reduced thanks to environmental modeling. Techniques for environment modeling can be derived from a variety of methodologies, including the framework-space approach, the free-space approach, the cell-decomposition approach, the topological approach, and the probabilistic-roadmap approach [12]. During the process of route planning, an optimization criterion needs to be established. There are a number of different variables that have the potential to play an essential part and can be incorporated into the criteria for optimization of rote planning for mobile robots. There are two primary categories of methodologies that are utilized in the process of designing optimization and route planning algorithms. These are heuristic approaches and artificial intelligence algorithms [13,14].

Due to their intelligence and prior knowledge by random searching, which are two components of population-based heuristic algorithms, metaheuristic algorithms are able to tackle unforeseen problems [15,16].

The focus of this paper is proposing a new route planning algorithm (RPA). The robots in question are autonomous robots sending signals with their locations, dimensions, and charging limits to the controller server via sensors. The controller is responsible for directing all of the robot's movements. It determines the amount of time needed to charge each robot, chooses the robot that has the best chance of completing a given mission, and finds the most efficient route for the robot to take. The RPA is comprised of three essential modules: the Robot Finding Module (RFM), the Robot Charging Module (RCM), and the Route Selection Module (RSM).

It is crucial to use autonomous systems such as RPAs in locations where there is a need for human gathering. This is especially true in the medical area, as it reduces the risk of virus transmission, potentially saving thousands of people's lives. The simulations run using the suggested framework demonstrate that the robots can move more flexibly and effectively than they could by using more traditional methods in a variety of settings. The RSM is compared to other cutting-edge topology routing choices that are currently on the market. The key advantage provided by the RSM is a significant reduction in the amount of time spent calculating and updating routing tables. In comparison to earlier algorithms, our RSM yielded the lowest AQD. As a result, the RSM is an algorithm that is suitable for use in real-time systems.

The novelties in this work are based on building a completely autonomous robot system considering all criteria in such systems:

1. Selecting the optimal robot for the incoming request using a new equation that has the needed requirements according to the problem at hand. The parameters of this can be changed and updated regarding the environment where the robot exists.
2. Finding the best path for the selected robot to move through using two different new algorithms: (a) the Graph-Based Path Finding Algorithm and (b) the Tree-Based Path Finding Algorithm.
3. Building a new recharging algorithm to recharge the robots periodically when it is mandatory to save time and cost.

The proposed algorithms ensure that the system is reliable and available.

The remainder of the work is structured as follows. In Section 2, research on routing techniques is presented. In Section 3, the proposed method is presented. Experimental evaluation is provided in Section 4. In Section 5, we conclude this work.

2. Literature Review

This section presents the latest algorithms related to autonomous robot systems. Using both static and dynamic forms of locomotion, J. Zico Kotler and Andrew Y. Ng [17] demonstrated a software solution that enables a quadruped robot to move across a range of difficult terrain swiftly and reliably. Static and dynamic gaits and specific dynamic maneuvers are all used in the software architecture. The two key components of their strategy were (1) the application of learning algorithms to learn route planning, footstep planning, and dynamic maneuvers; and (2) a focus on quick recovery and replanning to deal with circumstances where the robot deviates from its intended course.

ANYmal, a quadrupedal robot with exceptional mobility and the ability for dynamic motion, was unveiled by Marco Hutter [18]. It uses innovative, compliant joint modules. The robot was made with an emphasis on outdoor applicability, straightforward maintenance, and user-friendly handling. It is particularly robust against impulsive loads, such as running and jumping. ANYmal was viewed as a step toward merging extreme mobility with the capacity for dynamic locomotion. A particular focus was placed on having an easy-to-maintain system, which was accomplished by utilizing the modular joint components "ANYdrive", which make it incredibly simple to build robots with various kinematic structures. The robot's robustness is supported by the experiments performed on it. The

outstanding range of motion in all joins, which enables a wide range of maneuvers to overcome obstacles or to get up after falling, is undoubtedly the biggest benefit of ANYmal, aside from the improved protection. This feature also makes motion planning easier because there are fewer internal system constraints.

A sturdy and dynamic quadrupedal robot, the MIT Cheetah 3, was unveiled by Gerardo Bledt [19]. It has high-bandwidth proprioceptive actuators to govern physical interaction with the environment and uses a specialized mechanical design to offer straightforward control schemes for dynamic locomotion. It offers a unique leg style that offers a larger hip and knee ranges of motion. Cheetah 3's general balance and locomotion controls were presented; these enable the robot to adapt its gait in response to unforeseen terrain disturbances. It exhibits robustness even in the absence of external sensing, indicating its capacity for successful locomotion in difficult circumstances without relying on prior knowledge of the environment. Any number of different controllers can be used with the basic control architecture with little to no changes to the hardware or software. Initial findings from a novel, nonlinear policy-regularized model-predictive control framework (PR-MPC) are promising .

The authors of [1] suggested a novel routing method called ECRS. By using ECRS, the network is separated into distinct fog regions, and there is a master node in charge of managing communication in each area. In contrast to other caching solutions that use reactive routing protocols, ECRS uses a new built-in table-driven routing mechanism without any additional penalty. Such behavior significantly reduces the query delay. The secret is to collect the routing information while submitting message requests, then properly populate the routing tables. Using a convolutional neural network and a modified version of particle swarm optimization, the authors of [2] provided an efficient dynamic load balancing technique (EDLB) that examines the FC architecture for applications in healthcare systems.

A best-first search algorithm that is greedy will always choose the path that appears to be the best at that moment. It is a method that utilizes both the breadth-first search and the depth-first search algorithms. It makes use of the heuristic function and searching. Utilizing the best-first search gives us access to the benefits of both algorithms. With the best-first search's assistance, we can select the node that has the most potential at each step [20]. Best-first searches can take many forms, the most well-known of which is the A* search. It applies the heuristic function $h(n)$ and the cost to get from the starting state $g(n)$ to the node n. In the A* search algorithm, the search heuristic, in addition to the cost to get to the node, is taken into consideration. As a result, we are able to aggregate both costs, and the total of these two numbers is referred to as a fitness number. When it comes to multi-objective point planning, a better version of the A* algorithm in conjunction with the greedy method is used [20].

3. The Proposed Route Planning Algorithm (RPA)

This section proposes a new route planning algorithm (RPA). The autonomous robots send signals with their locations, dimensions, and charging limits to the controller server via sensors. The controller controls all the movement of the robot. It selects the best robot to achieve a specific task, detects the best path for the robot to move through, and decides the charging time for each robot. Three key modules make up RPA, as shown in Figure 1: (i) Robot Finding Module (RFM), (ii) Robot Charging Module (RCM), and (iii) Route Selection Module (RSM).

3.1. Robot Finding Module (RFM)

In this module, the best robot for a specific task is selected according to its features and requirements. RFM includes three main algorithms: Robot Information Algorithm (RIA), Robot Ranking Algorithm (RRA), and Robot Matchmaking Algorithm (RMA).

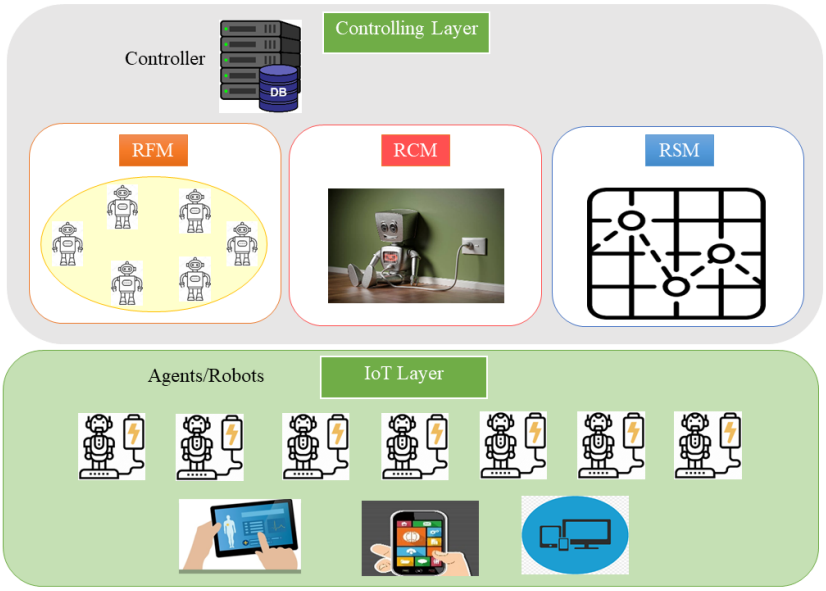

Figure 1. The proposed route planning algorithm framework.

3.1.1. Robot Information Algorithm (RIA)

In RIA, all information about each robot is collected and sent to the controller. This information is stored in a robot features table (RFT), as shown in Table 1.

Table 1. Robot features table (RFT).

Robot	S	V	CHt	D1	D2	W_r	W_i	C_p	Rank Value
R1	1	10	0.5	2	3	8	5	70%	0.26
R2	0	15	0.25	1	1	10	5	50%	0
R3	1	20	0.25	4	5	7	5	75%	0.12
R4	1	25	0.75	2	2	6	5	89%	0.69

As shown in Table 1, the data stored about each robot include the following. (i) Status (S): The status for the robot, and it equals one only if the robot is available. Otherwise, it is equal to 0. (ii) Velocity (V): the velocity of the robot. (iii) Remaining time after charging (per hour) (CHt): the approximate amount of time the robot can function "per hour". (iv) Distance from the initial point (D1): the distance from the initial point to the selected robot. (v) Distance to target (D2): the distance from the selected robot to the delivery point. (vi) Threshold weight (W_r): the robot's weight threshold. (vii) Item weight (W_i): the item's weight. (viii) Robot charge percentage (C_p): the robot's charge percentage.

To avoid collision and also to avoid delay in performance, the best robot should be selected accurately. To select the best robot, reinforcement learning (RL) is used to perform matchmaking between the incoming task and the robot, as shown in Figure 2 and Algorithm 1 (RL Matchmaking Algorithm (RMA)).

3.1.2. Robot Ranking Algorithm (RRA)

The Robot Ranking Algorithm (RRA), as shown in Equation (1), is responsible for giving each robot a rank value (R) to be selected to perform the incoming request.

$$R = S * \frac{(\alpha * V) + (\beta * CHt)}{D1 + D2} * \left(\frac{W_r - W_i}{W_r} * C_P \right) \quad (1)$$

where S is the status for the robot, and it equals one only if the robot is available. The term V is the velocity of the robot. CHt represents the approximate time remaining for the robot "per hour". α and β are the power factors for V and CHt, respectively. $D1$ is the distance

from the source to the robot, and $D2$ is the distance from the robot to the destination. W_r is the robot's weight threshold. W_i is the item's weight. C_p is the robot's charge percentage.

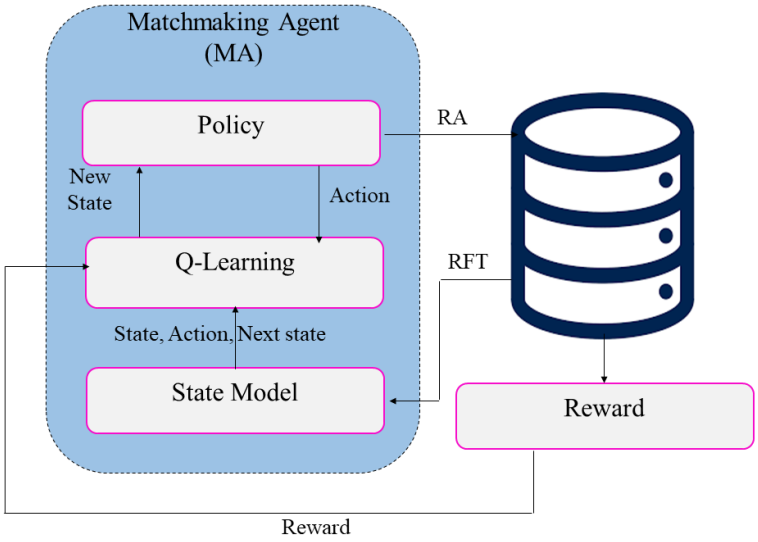

Figure 2. Matchmaking agent.

Algorithm 1: RL Matchmaking Algorithm (RMA)

Input : The data in the RFT
Output : Balanced system with high performance and fast response
foreach *new incoming Task* (T_i) **do**
Create a Q-table containing two columns [State: Available Robots, Action: Selecting the best robot to execute T_i] initialized to zero.
//**Taking Action**
- The agent interacts with the environment (Robots) and updates the state-action pairs in the Q-table Q[state, action].
- The agent uses the Q-table as a reference and views all possible actions (all possible available robots) for a given state. Then it selects the best robot based on the fast response.
- The agent uses the proposed formula in Equation 1 as a measurement criteria.

$$R = S * \frac{(\alpha * v) + (\beta * CHt)}{D1 + D2} * \left(\frac{W_r - W_i}{W_r} * C_P\right)$$

//**Updating the Q-table**
Update the values in the Q-table.
//**Update the data in RFT**
The RMA updates the data in the RFT (updates the response and the status for each robot).
end

3.1.3. Robot Matchmaking Algorithm (RMA)

Reinforcement learning (RL) is an AI method where an agent acts in a way that results in rewards. The agent receives information about the status of the environment at the moment and acts accordingly. The action causes a change in the environment, which is subsequently communicated to the agent as a reward. The RL Matchmaking Algorithm (RMA) learns which robot will carry out the incoming request in the best way. Algorithm 1 displays the overall RMA steps. In order to choose the best robot to carry out the incoming request, the matchmaking agent selection policy learns over time.

RL is used to help the controller to make the decision to decide which robot is most suitable for the incoming request. The use of RL here is not the same as the traditional use, such as in the case of considering the robot as an autonomous intelligent agent. However, here, RL is used to make a decision on another agent, which is the controller. All decisions are made on the controlling layer. The used policy in RL is updated and modified using the new formula described in Equation (1) to judge the performance of the agent.

The RL Matchmaking Algorithm's computational complexity can be determined according to Algorithm 1 as follows. Let R_i be the number of robots and T_i be the number of tasks. The computational complexity for the RMA algorithm is $O(T_i \times R_i)$.

3.2. Robot Charging Module (RCM)

In the Robot Recharge Scheduling Module (RRSM), each robot is scheduled according to its charging percentage to be selected to recharge. We should take into consideration the number of robots that must be disconnected at a time and recharged. There should be enough available robots at a time; hence, a systematic method is needed to rank and select the robots to be disconnected.

Fuzzy logic is used to rank robots according to their features, which are: (i) charging percentage (CP), (ii) velocity (V), and (iii) weight limit (W). The RRSM assigns a charging ranking to each robot (CR) by considering its three predefined features (CP, V, and W). All those parameters are considered in the fuzzy process. The fuzzy algorithm is fast and accurate in determining the ranking. Fuzzy algorithms are often robust in the sense that they are not very sensitive to changing environments and erroneous or forgotten rules.

1. Charging percentage (CP): value of high, medium, or low.
2. Velocity (V): fast, medium, or slow.
3. Weight limit (W): heavy, medium, or light.

The ranking value can be CR1, which is an alert (the robot should be disconnected to recharge immediately). CR2 is a warning. (The robot will be ranked as available for a time (10 min). It can be disconnected to recharge if there are enough available robots. If there are not enough available robots, it will be disconnected after 10 min).

The reasoning process is often simpler than computationally precise systems, so computing power is saved. This is a very interesting feature, especially in real-time systems. Fuzzy methods usually have a shorter development time than conventional methods. The fuzzy inference process is carried out in the following sequential steps: (i) Fuzzification of inputs. (ii) Applying the fuzzy rules. (iii) Defuzzification. Those steps are illustrated in the fuzzy process shown in Figure 3.

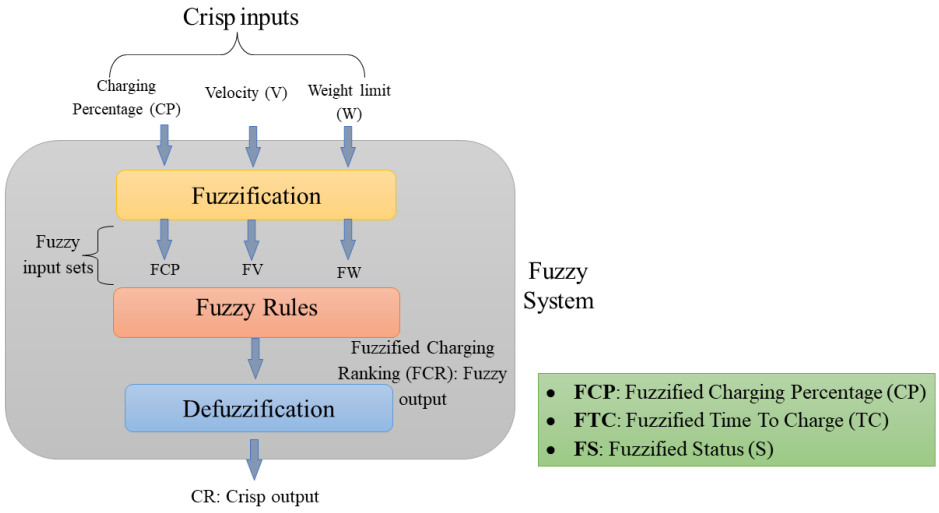

Figure 3. The fuzzy system.

3.2.1. Fuzzification

Fuzzification is the process of using an input membership function to convert sharp values into levels of membership in the fuzzy set under consideration. The three parameters, fuzzified charging percentage (FCP), fuzzified velocity (FV), and fuzzified weight limit (FW), are the fuzzy sets that are taken into account in the fuzzy process. These parameters have linear membership functions based on the CR's current information, with CP, V, and W, respectively. See the three input variables' membership functions (CP, V, and W) as shown in Equations (2)–(4).

$$FCP = (CP, \mu FCP(CP))/CP \in P \tag{2}$$

where CP = {low, medium, high}, P = [0,100], and $\mu FCP(CP) \in [0,1]$. FCP: Fuzzified Charging Percentage.

$$FV = (V, \mu FV(V))/V \in v \tag{3}$$

where, V = {slow, medium, fast}, v = [0,100], and $\mu FV(V) \in [0,1]$. FV: Fuzzified Velocity.

$$FW = (W, \mu FW(W))/W \in w \tag{4}$$

where, W = {light, medium, heavy}, w = [0,100], and $\mu FW(W) \in [0,1]$. FW: fuzzified weight.

3.2.2. Applying Fuzzy Rules

The fuzzy language rules are founded on if–then statements such as these:
$If\ CP$ is Low and V is Fast and W is Heavy $THEN\ R$ is $CR1$
$If\ CP$ is High $THEN\ R$ is $CR2$

3.2.3. Defuzzification

FCR is converted into a crisp value (CR).

3.3. Route Selection Module (RSM)

As shown in Equation (5) and Figure 4, the pathways between the groups of robots can be described as a weighted directed graph (G).

$$G = (R, E) \tag{5}$$

where E is the set of edges between the robots and R is the set of robots. The Euclidean distance is set as a cost on each edge. Two distances should be taken into account: (i) The separation between the starting location and the chosen robot. (ii) The separation between the chosen robot and the delivery location. To prevent crashes, the barriers on each route should be avoided.

The Route Selection Module (RSM) is divided into two primary sub-modules: (i) Using the checking for path procedure (CPP), which is used to determine whether a path exists between the source and the robot. (ii) The building path procedure (BPP), which creates a path from the source to the robot. Algorithm 2 displays the RSM's overall steps.

3.3.1. Checking for Path Procedure (CPP)

As demonstrated in Figure 4, when choosing a certain robot, it is important first to determine whether a path exists between it and the source and the destination. In Algorithm 2, the steps of CPP are displayed. Figure 5 shows the "Build_Path" and "Check_Path" classes.

The Route Selection Module (RSM) algorithm's computational complexity can be determined according to Algorithm 2 as follows. Let n be the requesting robots, R_r be the requesting robots, and R_c be the receiving robots. The computational complexity of the RMA algorithm is $O\ (n \times R_r \times R_c)$.

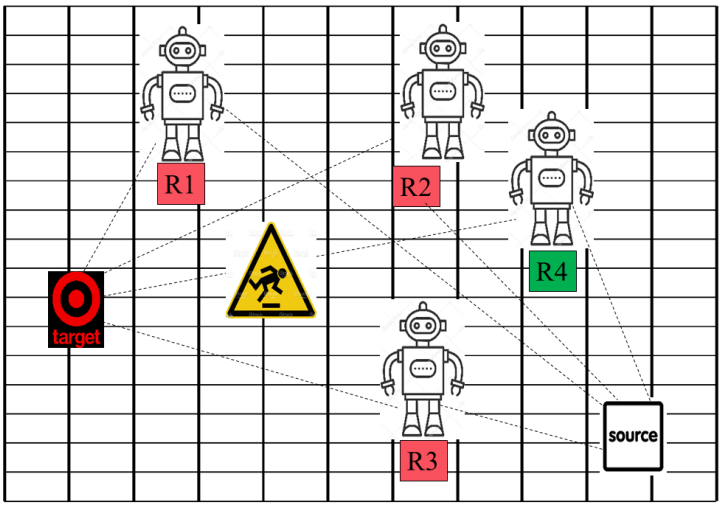

Figure 4. A grid map representation of the set of robots.

```
class Build_Path {
    //Number of FNs
    int N;
    Vertex[ ] vertices;
    int[,] adjMatrix;
    int numVerts;
    public Graph(){
      vertices = new Vertex[N];
      adjMatrix = new int[N, N];
      numVerts = 0;
      for(int j = 0; j <= N -1; j++)
          for(int k = 0; k <= N -1; k++)
              adjMatrix[j , k] = 0;  }
   public void AddVertex(string label) {
      Source_vertices[numVerts]=new Vertex(label);
      Destination_vertices[numVerts]=new Vertex(label);
      numVerts++; }
    public void AddEdge(string start, string end) {
      adjMatrix[start, end] = 1;
      adjMatrix[end, start] = 1; }
  }
```

```
class Check_Path {
   public int FindPath(int source) {
     int index;
     for(int i=0;i<N;i++) {
              If(source==Source_vertices[i])
                   index=i;
              else
                   return 0;
         }
    destination=Destination_vertices[index];
    return destination;
}
```

Figure 5. Classes of "Build_Path" and "Check_Path".

3.3.2. Building Path Procedure (BPP)

The Building path Procedure has been implemented using two proposed algorithms: (i) the Graph-Based Path Finding Algorithm and (ii) the Tree-Based Path Finding Algorithm.

Graph-Based Path Finding Algorithm (GPFA): As can be seen in Table 2, the controller routing table, also known as the CRT, includes routes to each and every robot.

Table 2. The controller routing table (CRT) for an example robot.

Initial Point	Destination	Next Hope	Route	Obstacle	Distance	Was Visited
S1	D1	R1	PL = S1, R1, D1	False	5 m	True

Tree-Based Path Finding Algorithm (TPFA): In order to get where it needs to go, a robot has to have a physical path through space, a need that is often complicated by the presence of obstacles or other movement constraints. The TPFA algorithm uses a rapidly-exploring random tree, also known as the RRT*, to find a path through a 2-dimensional environment containing several differently-sized obstacles. The steps of the TPFA algorithm are shown in Figure 6.

Algorithm 2: Route Selection Module (RSM).

```
Input  : A grid map representation of the set of robots, Robot (R), and Controller
         (C)
Output : Building Controller Routing Table (CRT)
foreach requesting robot do
    Send its id to FindPath(R_id)
     // The receiver R was not visited before
     if return_value ≠ 0 then
        Accept the returned value (destination)
         Save the R_id and destination in CRT
         Send a copy of CRT
    else
        foreach requesting R do
            The controller sends a "test" message to all the robots lying within its
             wireless transmission range.
             foreach receiving robot do
                These nearest robots are called First neighbored robots (FRs).
                 When each of the FRs receives the "test" message, it will be first
                 tested if it was visited or not.
                 // The receiver R was not visited before
                 if was_visited == False then
                    The label of this R (R_id) will be send to AddVertex(R_id)
                     Send and the label of the sender as the first parameter to
                     AddEdge
                     Send the label of this R (R_id) as the second parameter to
                     AddEdge(C_id, R_id)
                     //The receiver R will be marked as "was_visited"
                     was_visited = True
                 else
                    The receiver R will ignore the "test" message and will not reply
                 end
            end
        end
    end
end
```

To grow the tree, we pick a random spot in space, then figure out which of the existing nodes is closest to the random point. Then, we draw a straight line between the random point and the closest node and create a new node along that line a certain distance away from the closest node. The tree grows in different directions throughout this space by choosing a series of random points and creating a series of new nodes. A 200-node RRT* random tree can be seen in Figure 7.

The RRT* random tree can be used to navigate around obstacles between two points. When generating each new node, we perform collision detection with each obstacle, and we also note which existing node it is connected to—the "parent" node mentioned before. Another way of thinking about the parent is that it is the node in the tree that the randomly generated point was closest to. Keeping track of these parent nodes is important because it allows us to re-trace our steps once we find a clear path to the endpoint. By jumping from parent node to parent node all the way back to the starting point, the path is constructed so that we can tell the robot where to go.

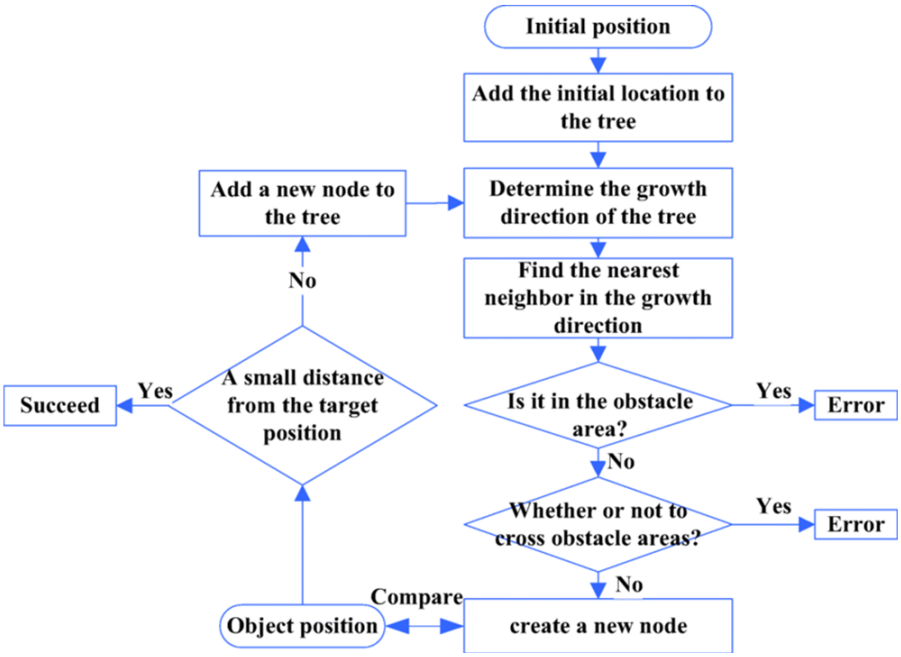

Figure 6. The Tree-Based Path Finding Algorithm's steps.

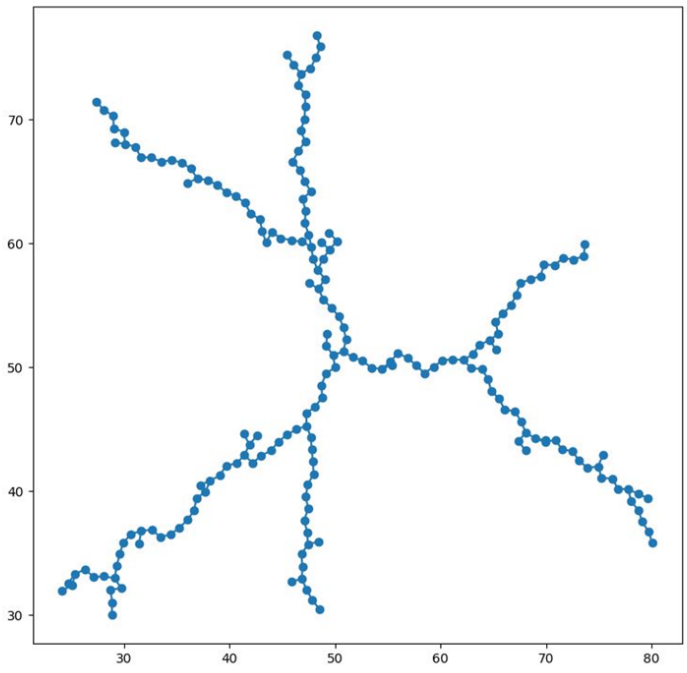

Figure 7. A 200-node RRT*.

4. Implementation and Evaluation

The employed dataset, performance indicators, and evaluation findings are all reported in this section.

4.1. Dataset

We used a grid map representing a group of robots, as shown in Figure 4. The initial values of each feature are shown in Table 3. The initial points, destination, next hope, route, whether or not an obstacle existed, distance, and whether or not it was visited are all listed in this table.

Table 3. Controller routing table (CRT).

Initial Point	Destination	Next Hope	Route	Obstacle	Distance	Was Visited
S1	D1	R1	PL = {S1, R1, D1}	False	5 m	True
S1	D1	R2	PL = {S1, R2, R1, D1}	False	6 m	False
S1	D1	R3	PL = {S1, R3, D1}	False	3 m	False
S1	D1	R4	PL = {S1, R4, D1}	True	4 m	False
S1	D1	R3	PL = {S1, R3, R1, D1}	False	3 m	False
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.

4.2. Robot Ranking Algorithm (RRA) Implementation

The features for each robot are shown in Table 4. This table contains information regarding the status of the robot, its velocity, the amount of battery power that was still left (in hours), the distance from the starting point, the distance to the target, the threshold weight, the item's weight, and the percentage of charge that the robot currently possesses.

Table 4. Features for each robot.

Robot	S	V	CHt	D1	D2	W_r	W_i	C_p
R1	Available	10	0.5	2	3	8	5	70%
R2	Busy	15	0.25	1	1	10	5	50%
R3	Available	20	0.25	4	5	7	5	75%
R4	Available	25	0.75	2	2	6	5	89%

Apply Equation (1) to give each robot a rank value (R). For α and β in Equation (1) to be equal to 1, the R value for each robot is calculated as follows, and these are shown in Table 5.

R1 is available. R can be calculated as

$$R = 1 * \frac{(10) + (0.5)}{2 + 3} * \left(\frac{8 - 5}{8} * 70\% \right) = 0.26 \tag{6}$$

R2 is not available (busy). R can be calculated as

$$R = 0 * \frac{(15) + (0.25)}{1 + 1} * \left(\frac{10 - 5}{10} * 50\% \right) = 0.0 \tag{7}$$

R3 is available. R can be calculated as

$$R = 1 * \frac{(20) + (0.25)}{4 + 5} * \left(\frac{7 - 5}{7} * 75\% \right) = 0.12 \tag{8}$$

R4 is available. R can be calculated as

$$R = 1 * \frac{(25) + (0.57)}{2 + 2} * \left(\frac{6 - 5}{6} * 89\% \right) = 0.69 \tag{9}$$

The most impressive rank value, 0.69, was achieved by R4. The rank value of R2 is 0, since it could not be used because it was either busy or unavailable.

Table 5. Rank value for each robot.

Robot	S	V	CHt	D1	D2	W_r	W_i	C_p	R
R1	1	10	0.5	2	3	8	5	70%	0.26
R2	0	15	0.25	1	1	10	5	50%	0
R3	1	20	0.25	4	5	7	5	75%	0.12
R4	1	25	0.75	2	2	6	5	89%	0.69

4.3. Robot Recharge Scheduling Module (RRSM)

Charging percentage (CP): low (0–30%), medium (30–60%), and high (60–100%). Velocity (V): slow (0–10), medium (10–20), or fast (20–30). Weight limit (w): light (0–5), medium (5–10), heavy (10–15). CR can be CR2 (low priority) or CR1 (high priority). The results for the RRSM are shown in Table 6.

Table 6. Features for each robot.

Robot	CP	V	W	CR
R1	70%	10	8	CR2
R2	50%	15	10	CR2
R3	75%	20	7	CR2
R4	89%	25	6	CR2

4.4. Route Selection Module (RSM)

To investigate PRM's scalability and robustness in a dynamic context, MATLAB simulations and tests were conducted.

4.4.1. TPFA Algorithm Performance Evaluation

The TPFA first uses the RRT* algorithm to find the best path, which is the most suitable path found in the smallest time while avoiding any collisions, as shown in Figure 8. Then, it saves this path and uses it for the robot's movement.

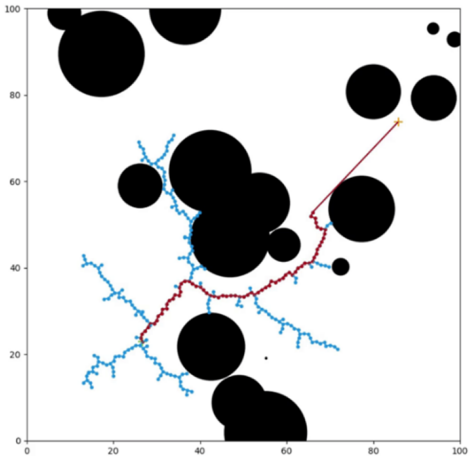

Figure 8. Best path avoiding collisions.

This algorithm outperforms the other path-finding algorithms, as it is fast and avoids any obstacle without any collision.

4.4.2. The Performance Metrics for Routing

The cache hit ratio (CHR), average query delay (AQD), average hop count (AHC), and power consumption are standard performance indicators that are used to assess how well the routing schemes work (PC). The definitions of the performance metrics are compiled in Table 7.

Table 7. The performance indicators used to assess the proposed RSM.

Metric	Definition
Cache Hit Ratio (CHR)	The proportion of requests that were successful to all requests, which may be determined by Equation (10). $$CHR = \frac{R_{hit}}{R_{total}} = \frac{R_{hit}}{R_{hit} + R_{miss}} \quad (10)$$ where R_{total} denotes the total number of requests, R_{hit} is the number of things that were found in the cache and were valid, and R_{miss} denotes the number of items that were requested but were not located, either locally or remotely, or were located but were invalid.
Average Query Delay (AQD)	The average query delay is the average query delay overall successful queries sent by all requesters. The query delay is the amount of time between the requester's sending the query and receiving the data back. One method for calculating AQD is as in Equation (11). $$AQD = \frac{\sum_{k=0}^{R_{succ}} (QT^k)}{R_{succ}} = \frac{R_{hit}}{R_{hit} + R_{miss}} \quad (11)$$ where R_{succ} is the number of successful inquiries, and QT is the query delay time.
Average Hop Count (AHC)	The average hop count is the average hop count overall successful queries sent by all requesters.
Power Consumption (PC)	The typical amount of electricity used over a multi-hop route.

4.4.3. Rsm Evaluation

As demonstrated in Table 8, RSM is contrasted with the leading cutting-edge topology routing systems, namely, ECRS [21].

Table 8. PRM comparison with the best cutting-edge routing solutions.

Protocols	Year	CHR	AQD	AHC	PC
RSM	2022	97%	2.37	10	4720
ECRS [21]	2019	96%	2.67	11	4910

The key advantage provided by RSM is a significant reduction in the amount of time spent calculating and updating routing tables. RSM is the method that yielded the lowest AQD, in contrast to earlier algorithms. As a result, RSM is an algorithm that is suitable for use in real-time systems.

5. Conclusions

This study presented a new route planning algorithm (RPA) based on reinforcement learning that provides efficient medical services by collecting and transmitting data between robots and human healthcare providers while keeping humans out of the patient's field of view. There are three key modules that make up the RPA: (i) Robot Finding Module (RFM), (ii) Robot Charging Module (RCM), and (iii) Route Selection Module (RSM). In order to reduce the risk of viral propagation and save thousands of lives, it is crucial to use such autonomous systems as RPA in settings where people must congregate. The simulation results using the suggested framework demonstrate the flexible and effective mobility of the robots under diverse environmental conditions, as opposed to conventional methods. RSM is contrasted with the leading cutting-edge topology routing options. RSM's primary benefit is the much-reduced calculations and updating of routing tables. In contrast to earlier algorithms, RSM produces the lowest AQD. RSM is hence an appropriate algorithm for real-time systems. In future work, the proposed algorithm will be applied to a larger number of robots in more complex environmental conditions.

Author Contributions: Conceptualization, F.M.T.; Methodology, F.M.T.; Validation, E.-S.M.E.-K.; Formal analysis, A.A.A. (Abdelaziz A. Abdelhamid), D.S.K. and D.A.S.; Investigation, E.-S.M.E.-K. and D.A.S.; Resources, E.-S.M.E.-K. and D.S.K.; Writing—original draft, F.M.T.; Writing—review & editing, A.I. and A.A.A. (Abdelaziz A. Abdelhamid); Visualization, A.A.A. (Abdelaziz A. Abdelhamid); Supervision, A.I.; Project administration, A.A.A. (Amel Ali Alhussan); Funding acquisition, A.A.A. (Amel Ali Alhussan) and D.S.K. All authors have read and agreed to the published version of the manuscript.

Funding: Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2022R308), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

Acknowledgments: The authors would like to thank Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2022R308).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Talaat, F.M.; Ali, S.H.; Saleh, A.I.; Ali, H.A. Effective cache replacement strategy (ECRS) for real-time fog computing environment. *Clust. Comput.* **2020**, *23*, 3309–3333. [CrossRef]
2. Talaat, F.M.; Ali, H.A.; Saraya, M.S.; Saleh, A.I. Effective scheduling algorithm for load balancing in fog environment using CNN and MPSO. *Knowl. Inf. Syst.* **2022**, *64*, 773–797. [CrossRef]
3. Wang, C.; Horby, P.W.; Hayden, F.G.; Gao, G.F. A novel coronavirus outbreak of global health concern. *Lancet* **2020**, *395*, 470–473. [CrossRef] [PubMed]
4. El-kenawy, E.S.M.; Ibrahim, A.; Mirjalili, S.; Eid, M.M.; Hussein, S.E. Novel Feature Selection and Voting Classifier Algorithms for COVID-19 Classification in CT Images. *IEEE Access* **2020**, *8*, 179317–179335. [CrossRef] [PubMed]
5. Schwartz, J.; King, C.C.; Yen, M.Y. Protecting Healthcare Workers During the Coronavirus Disease 2019 (COVID-19) Outbreak: Lessons From Taiwan's Severe Acute Respiratory Syndrome Response. *Clin. Infect. Dis.* **2020**, *71*, 858–860. [CrossRef] [PubMed]
6. El-Kenawy, E.S.M.; Mirjalili, S.; Ibrahim, A.; Alrahmawy, M.; El-Said, M.; Zaki, R.M.; Eid, M.M. Advanced Meta-Heuristics, Convolutional Neural Networks, and Feature Selectors for Efficient COVID-19 X-Ray Chest Image Classification. *IEEE Access* **2021**, *9*, 36019–36037. [CrossRef] [PubMed]
7. Fragapane, G.; de Koster, R.; Sgarbossa, F.; Strandhagen, J.O. Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda. *Eur. J. Oper. Res.* **2021**, *294*, 405–426. [CrossRef]
8. Karur, K.; Sharma, N.; Dharmatti, C.; Siegel, J.E. A Survey of Path Planning Algorithms for Mobile Robots. *Vehicles* **2021**, *3*, 448–468. [CrossRef]
9. Lu, X.; Bao, H.; He, Y.; Huang, J.; Wang, Q.; Mai, K. Intelligent route planning model of industrial robot based on inertia moment parameter optimization. In Proceedings of the 2021 IEEE International Conference on Computer Science, Electronic Information Engineering and Intelligent Control Technology (CEI), Fuzhou, China, 24–26 September 2021. [CrossRef]
10. Cai, K.; Wang, C.; Cheng, J.; Silva, C.W.D.; Meng, M.Q.H. Mobile Robot Path Planning in Dynamic Environments: A Survey. *Instrumentation* **2019**, *6*, 90–100. [CrossRef]
11. Alhussan, A.A.; Khafaga, D.S.; El-Kenawy, E.S.M.; Ibrahim, A.; Eid, M.M.; Abdelhamid, A.A. Pothole and Plain Road Classification Using Adaptive Mutation Dipper Throated Optimization and Transfer Learning for Self Driving Cars. *IEEE Access* **2022**, *10*, 84188–84211. [CrossRef]
12. Zhong, X.; Tian, J.; Hu, H.; Peng, X. Hybrid Path Planning Based on Safe A* Algorithm and Adaptive Window Approach for Mobile Robot in Large-Scale Dynamic Environment. *J. Intell. Robot. Syst.* **2020**, *99*, 65–77. [CrossRef]
13. El-Kenawy, E.S.M.; Mirjalili, S.; Abdelhamid, A.A.; Ibrahim, A.; Khodadadi, N.; Eid, M.M. Meta-Heuristic Optimization and Keystroke Dynamics for Authentication of Smartphone Users. *Mathematics* **2022**, *10*, 2912. [CrossRef]
14. Abdelhamid, A.A.; El-Kenawy, E.S.M.; Alotaibi, B.; Amer, G.M.; Abdelkader, M.Y.; Ibrahim, A.; Eid, M.M. Robust Speech Emotion Recognition Using CNN+LSTM Based on Stochastic Fractal Search Optimization Algorithm. *IEEE Access* **2022**, *10*, 49265–49284. [CrossRef]
15. Ding, Y.; Luo, W.; Sycara, K. Heuristic-based Multiple Mobile Depots Route Planning for Recharging Persistent Surveillance Robots. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, 3–8 November 2019. . [CrossRef]
16. Khafaga, D.S.; Alhussan, A.A.; El-Kenawy, E.S.M.; Ibrahim, A.; Eid, M.M.; Abdelhamid, A.A. Solving Optimization Problems of Metamaterial and Double T-Shape Antennas Using Advanced Meta-Heuristics Algorithms. *IEEE Access* **2022**, *10*, 74449–74471. [CrossRef]
17. Kolter, J.Z.; Ng, A.Y. The Stanford LittleDog: A learning and rapid replanning approach to quadruped locomotion. *Int. J. Robot. Res.* **2011**, *30*, 150–174. [CrossRef]

18. Hutter, M.; Gehring, C.; Jud, D.; Lauber, A.; Bellicoso, C.D.; Tsounis, V.; Hwangbo, J.; Bodie, K.; Fankhauser, P.; Bloesch, M.; et al. ANYmal—A highly mobile and dynamic quadrupedal robot. In Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, Republic of Korea, 9–14 October 2016. [CrossRef]
19. Bledt, G.; Powell, M.J.; Katz, B.; Carlo, J.D.; Wensing, P.M.; Kim, S. MIT Cheetah 3: Design and Control of a Robust, Dynamic Quadruped Robot. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018. [CrossRef]
20. Xiang, D.; Lin, H.; Ouyang, J.; Huang, D. Combined improved A* and greedy algorithm for path planning of multi-objective mobile robot. *Sci. Rep.* **2022**, *12*, 13273. [CrossRef] [PubMed]
21. Umamaheswari, S.; Radhamani, G. Enhanced ANTSEC framework with cluster based cooperative caching in mobile ad hoc networks. *J. Commun. Net.* **2015**, *17*, 40–46. [CrossRef]