Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 8902328, 16 pages
https://doi.org/10.1155/2021/8902328

Research Article

Parallel Swarm Intelligent Motion Planning with Energy-Balanced for Multirobot in Obstacle Environment

Shoubao Su,[1,2] **Wei Zhao**,[1,2] **and Chishe Wang**[1]

[1]*Jiangsu Key Laboratory of Data Science and Smart Software, Jinling Institute of Technology, Nanjing 211169, China*
[2]*School of Computer, Jiangsu University of Science and Technology, Zhenjiang 212003, China*

Correspondence should be addressed to Shoubao Su; showbo@jit.edu.cn and Chishe Wang; wangcs@jit.edu.cn

Received 28 June 2021; Revised 9 August 2021; Accepted 12 August 2021; Published 30 August 2021

Academic Editor: Xiaoxian Yang

Multirobot motion planning is always one of the critical techniques in edge intelligent systems, which involve a variety of algorithms, such as map modeling, path search, and trajectory optimization and smoothing. To overcome the slow running speed and imbalance of energy consumption, a swarm intelligence solution based on parallel computing is proposed to plan motion paths for multirobot with many task nodes in a complex scene that have multiple irregularly-shaped obstacles, which objective is to find a smooth trajectory under the constraints of the shortest total distance and the energy-balanced consumption for all robots to travel between nodes. In a practical scenario, the imbalance of task allocation will inevitably lead to some robots stopping on the way. Thus, we firstly model a gridded scene as a weighted MTSP (multitraveling salesman problem) in which the weights are the energies of obstacle constraints and path length. Then, a hybridization of particle swarm and ant colony optimization (GPSO-AC) based on a platform of Compute Unified Device Architecture (CUDA) is presented to find the optimal path for the weighted MTSPs. Next, we improve the A∗ algorithm to generate a weighted obstacle avoidance path on the gridded map, but there are still many sharp turns on it. Therefore, an improved smooth grid path algorithm is proposed by integrating the dynamic constraints in this paper to optimize the trajectory smoothly, to be more in line with the law of robot motion, which can more realistically simulate the multirobot in a real scene. Finally, experimental comparisons with other methods on the designed platform of GPUs demonstrate the applicability of the proposed algorithm in different scenarios, and our method strikes a good balance between energy consumption and optimality, with significantly faster and better performance than other considered approaches, and the effects of the adjustment coefficient q on the performance of the algorithm are also discussed in the experiments.

1. Introduction

Robot technology is always a remarkably interdisciplinary topic of study, one that can be applied to various engineering practices as well as emerging industrial fields. The edge intelligence, on the other hand, has been known as one of the most difficult research areas to apply multirobot systems (MRSs), in which the key technologies such as path planning, robustness, fault tolerance, and stability cooperation of MRS need to be studied. Motion planning is an extension of path planning, and there exist few differences between them. Generally, path planning is to find the path between starting node and target node in specific scenes by objectives like the shortest distance or the shortest time, while motion planning is to generate interactive trajectories in the situation when robots interact with their surroundings, where there are usually many factors such as velocities, obstacles, and energies that need be considered as constraints [1]. Path planning has always been one of the important challenges, which has been widely used in engineering practices, such as underwater robot detection, UAV cruise, vehicle tracking, warehousing, robot delivery, smart edge systems, and various navigation scenarios [2]. Multirobot path planning (MRPP) is to find the shortest way that allows each robot to follow a safe and effective path to reach a goal from an initial node, optimize the motion path and avoid obstacles, and generate a smooth motion path between nodes that conforms to the objectives.

With decades of year development, there are various studies of motion planner including traditional interpolating and heuristics approaches employed effectively in multirobot navigation over different environmental conditions so far. Even though, it is still an open challenging for multirobot regarding uncertain constraints, especially dynamic adaptive, scalable, and online replanning are still open and challenge problems for practical applications in real-world with large-scale nodes [3–5], which are far from being completely solved, and many practical engineering problems are still tackled today using learning-based heuristics algorithms, for the purpose of robustness, safety, speed, and efficiency. The main advantages of these methods lie in their ease of implementation, effectiveness for specific applications, and feasibility for small-scale and low dimensional workspace. However, some of them fails to complicated time-sequential motion planning, or limited robustness, or computational expensive, or unknown environments, especially the large-scale problem still have to be faced by the exponential growth of the search space and the disaster of dimensionality.

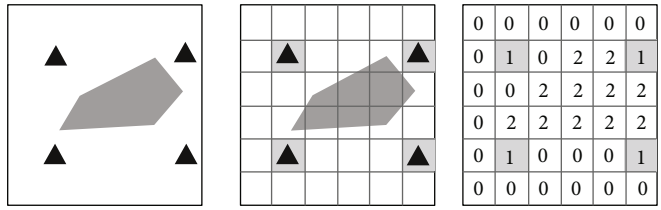
The early graphics processing unit (GPU) did not design a programming architecture for parallel computing. Programmers usually utilize special skills, such as Shader, to realize algorithm parallelism, which result in limitations and complexity of programming. With the development of GPU, especially edge intelligence, in the field of scientific computing, NVIDIA and AMD successively launched GPU-based parallel software and hardware architecture CUDA (Compute Unified Device Architecture) and ROCm (radeon open computing platform). Similarly, ROCm stack and provided a path for the parallel implementation of large-scale evolutionary computing [6]. Modern GPUs use multistream processor (SM) design and single instruction multithreading (SIMT) instruction architecture, which provides a good hardware platform for parallel algorithms. In reality, speed is essential in the applied scenes of MRSs, when the energy consumption of multirobots is not balanced, some robots may be stopped on the way, which will affect the stability and persistence of the whole multirobot system and even affect the completion of the task.

Therefore, this work is focused on motion planning for multirobot in a complex scene that has different surrounding obstacles. The goal is to design fast approaches that enable a real-time calculation of trajectories for multirobot systems which have balanced energies to finish their tasks. After the problem is to be solved modeled as a weighted MTSP, an improved RRT search combining slam construction method is proposed, which can help robots locate and map unknown areas more quickly. Then, we use particle swarm clustering and ant colony optimization methods to find the point-to-point direct path satisfying the basic constraints in the low-dimensional space and then restore the path to the actual scene through the trajectory generation algorithm. At the same time, combined with the characteristics of easy parallelization of swarm intelligence algorithm, aiming at the problems of high accuracy but long execution time of hybrid iterative algorithm, considering the energy consumption balance of multiple robots, by developing a GPU based on CUDA platform to accelerate the running speed of the algorithm, a smooth grid path algorithm combined with the minimum snap algorithm is proposed to make the final path globally optimal, safe, and fast, collision-free, energy-balanced and meet the dynamic constraints.

2. Related Works

2.1. Map Modeling. Multirobot motion planning mainly involves three stages which are building the global map model, generating a search path, and optimizing the motion trajectories. The first stage is to model a map of the given environment that is the input data for algorithms. There are algorithms commonly used to model maps that cover grid method, topology method, line of sight diagram method, tangent diagram method, and Voronoi diagram method [7, 8]. The former two are mostly utilized for global path planning, while the latter three are mostly used for local search. Among them, the grid method is simple and clear, which is recognized as an algorithm with high safety [9]. In the algorithm, the environment can be discretely gridded on a mesh map as fine as possible when the computing resources permit, to reduce the time complexity of finding both obstacles and target-nodes to $O(1)$. As shown in Figure 1, we assume that some irregular polygons are obstacles, and triangles are target nodes on the original map. After offline processing using the grid method, an original environment, the gridded mesh, and the discretized matrix are as shown in Figures 1(a)–1(c).

2.2. Solving Algorithms. Due to the characteristics and complexity of MRPP, it is a difficult and complex NP hard problem, which is essentially an optimization problem with many constraints. The shortest path planning with balanced-energy consumption for each robot is often modeled as the optimization objective. Researchers, so far, use heuristic algorithms to solve the problems, mainly including hybrid iterative method, divide and conquer method, and coevolution strategy. The A$*$ algorithm is one of the most typical graph-based in path search, and it inherits the idea of the Dijkstra and makes it different improved variants [10, 11], which can generate not only the shortest path but also the path more in line with the preference of various models, for example, it can be abstracted a variety of energies into the problems to find the optimal solution [12]. But the disadvantage is its failure to the problems of large-scale nodes, so, deep neural network (DNN) was employed to optimize the evaluation functions $g(n)$ and $H(n)$ of the A$*$ algorithm [13]. The sampling-based algorithms mainly include the probability spectrum method and RRT algorithm. The nearest sampling nodes on the map for obstacle collision detection are connected by trees, which can quickly find a reachable path in a limited time. If there are a large number of obstacles or difficult areas in the environment, the algorithm will run fast, but the algorithm has no perception of the environment. These algorithms are mostly used in local search by combining with Slam (simultaneous



(a) Original environment (b) Gridded mesh (c) Discretized matrix

Figure 1: Grid modeling.

localization and mapping), so that enhance the capabilities of local perception [14].

In recent years, there are numerous researches focused on this issue concerning intelligent methods. In many fields related to artificial intelligence, motion planning for multirobot systems (MRS) is undoubtedly one of the crucial topics that cover all kinds of applications based on swarm optimizers [15–17], including ant colony optimization (ACO) [16] and particle swarm optimization (PSO) [17], because of their effective ways to take the advantages of population information to enhance the overall solution quality and accelerate the convergence speed [15]. For example, ACO clustering with crowding mechanism and GA-based multitask scheduling methods were developed for drones safely flight in a specific airspace [18], two phases heuristic algorithm (TPHA) [19], an improved shuffled frog leaping algorithm (SFLA) [20] was embedded into the trajectory smooth path to determine an optimal subsequent position for each robot, and so on. Neural networks and reinforcement learning (RL) are never absent from routing problems, an optimal navigation strategy with energy-aware coverage designed by using RL for self-reconfigurable robots [21], while deep learning-based warm-starting optimizing motion planner [22] was employed to reduce motion time for robots picking an e-commerce warehouse. To make full use of the advantages of various methods, hybrid intelligent methods have been paid more and more attention recently, and a hybridization planning method is demonstrated [23] by integrating 0-1 optimization and EDA and GA for UAVs and UGVs to cooperative complete the coverages with minimum travel time in urban environments. When PSO planner applied to an intricate unknown environment, Shukla et al. [24] assessed its parameters including move (number of visited nodes), coverage (area explored), energy (distance traveled), and time (time elapsed). By making the best use of limited time, Salah et al. [25] proposed a path-planning approach for multidrones to conduct multiple photographic aerial wildlife surveys.

Some theoretical methodologies are introduced to plan the shortest path in static and dynamic environments. A framework regarding Satisfiability Modulo Theory (SMT) was proposed [26], which models the path as a connected network of mass-spring-damper systems and leverages the properties of Voronoi diagrams to handle complex constraints. To plan a path for the robot on the graph with edge energies, a nonmyopic path planner based on a game-theoretic framework was presented [27] by employing an infinite-horizon Markov decision process. When facing complex and changeable factors of traffic networks, the shortest path with the maximum passing probabilities based on the mobile trajectory can be effectively obtained by calculating the Markov chain and probabilistic symbolic model [28]. To minimize the path length as well as to reduce the number of handovers while sustaining the wireless connectivity of the robots, a multirobot access node association planner was presented [29] in millimeter-wave industrial scenarios. When the motion control for multirobot navigation is used in large-scale dynamic scenarios, a personalized route algorithm was presented by introducing the Polychromatic Sets (PS) for users to obtain real-time route that meet their travel preferences [4], all obstacle features can be integrated into a scalar potential field to make decisions [30], an online adaptive replanning strategy for multiple drones flying in an urban environment with a number of dynamic changes [5], and a reliable routing optimization scheme based on the Manhattan mobility model is presented [31] for UAV real-time planning in a vehicular ad hoc networks (VANETs). While a real-time tracking method was designed [32] that combines A∗ algorithm with dynamic window to guarantee the ability of obstacle avoidance and path smoothness. Thus, it has been observed that the heuristics are more robust and conduct well in scenarios when compared to other algorithms.

3. Energy-Balanced Motion Trajectory Planning Algorithm

3.1. Problem Description and Modeling. This paper focuses on the problem of multirobot motion planning, the first stage of which can be abstracted as a single-point MTSP problem. Assume that m traveling salesmen (denoted as b_1, $b_2 \cdots b_m$) will visit n cities (denoted as $T_0, T_1 \cdots T_n$) to sell their goods, and all traveling salesmen start from the same city T_1 and finally meet in city T_1; it is required that there must be only one salesman to visit each city, and the total energy and minimum route scheme of all salesmen are obtained [12]. The objective function is

$$\min Z = \sum_{k=1}^{m} \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} p_{ij}^{k}, \tag{1}$$

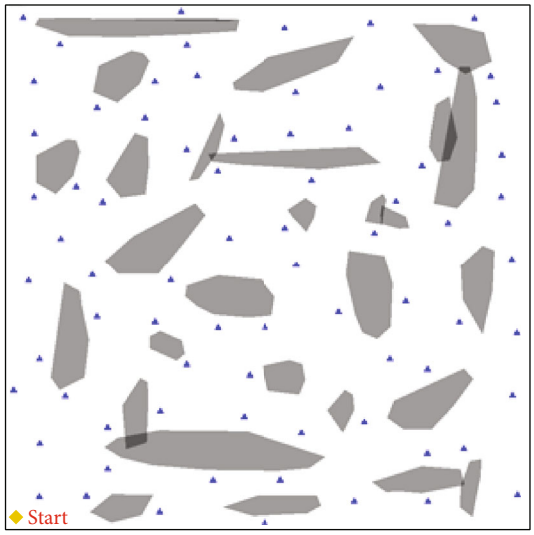


Figure 2: A schematic map of the environment with obstacles.

where p_{ij}^k is denoted as equation (2) that is the direction marking variable between cities i and j of the traveling salesman with serial number k.

$$p_{ij}^k = \begin{cases} 1, & \text{traveling salesman } k \text{ travels from city } i \text{ to } j, \\ 2, & \text{else.} \end{cases} \tag{2}$$

Then, we assume that a given environment as shown in Figure 2, in which there are a large number of obstacles (irregular polygons), m target nodes (triangles) $Q_i (i = 1, \cdots,$), a given starting node P (diamond at the bottom left). Now, n mobile robots will visit to m target nodes to perform tasks, each node is only visited by one robot, and all robots must return to the starting node after finishing their tasks.

3.2. Proposed Method. In this section, let us first explain what is energy balance and why we should balance the energy of multirobots. When each robot carries the same energy at the starting node, how to ensure that the total traveling distance of all robots is the shortest, and how to avoid the imbalance of energy consumption of each robot, that is, some robots stop on the way due to insufficient energy, while the others return to the starting node with a large amount of energy remaining. Thus, we present our motion planning scheme in this paper that has two optimization objectives, one is to minimize the total length of the multirobot traveling path, and another is to minimize the deviation of the path length of the multirobot to ensure the energy consumption balance. For these purposes, in our scheme, the environment information is simplified and merged according to the target to be optimized, the quantitative data and the model are established to solve the initial planning path in line with the optimization target, and then the initial planning path is added with obstacles constraints and dynamic constraints so that the planned path can be truly used in the actual environment. The scheme for multirobot motion planning in obstacles is first proposed as Algorithm 1, and its methodology stages of pathfinding, trajectory optimization, and smoothing can be described in details next sections.

3.2.1. Improved A ∗ Algorithm for Path Search. Based on the gridded environment, the A∗ algorithm can be selected for pathfinding with efficiency. The core evaluation function of the A∗ algorithm is as follows:

$$f(n) = g(n) + h(n), \tag{3}$$

where $f(n)$ is the total evaluation value of the current node n, $g(n)$ is the evaluation generation value from the starting node to the current node in the map, and $h(n)$ is the evaluation generation value from the current node to the endpoint. In the algorithm, Manhattan distance between nodes is usually used as the evaluation basis:

$$D(n) = |x_d - x_n| + |y_d - y_n|. \tag{4}$$

The A∗ algorithm also needs two sets to record the intermediate process: an open set and a closed set. The former is used to store the nodes that have not passed under the reachable condition, while the latter is used to store all nodes that have passed or are not reachable. Thus, the method can be presented as Algorithm 2, and the algorithm flowchart is shown in Figure 3. In addition, we also give example graph of fully connected paths with obstacle avoidance generated by the improved A∗ algorithm as shown in Figure 4.

3.2.2. Trajectory Optimization. The path search algorithm does not take into account dynamic constraints such as speed and acceleration when the robot is moving on the way. From the geometric of a view, the planned path is all splicing of line segments, which leads to the connection between the paths by turning points. It is impossible to strictly follow this path in the process, because the motion parameters such as speed and acceleration at the turning point will change unless it is assumed that the speed and acceleration at the turning point are reduced to 0, which will, of course, reduce the motion efficiency of motions. Then, a polynomial-based trajectory optimization method was presented by minimizing jerk (snap) to ensure that the generated trajectory polynomial is continuous and smooth in both speed and acceleration [32–34]. The continuous trajectories between two ends in a two-dimensional plane can be described by the analytic polynomials $x(t)$ and $Y(t)$ about time t, taking the x-axis as an example, the polynomial expression is as equation (5).

$$X(t) = p_n t^n + p_{n-1} t^{n-1} + \cdots + p_1 t + p_0 = \sum_{n=0}^{k} p_n t^n, \tag{5}$$

where k is the order of polynomial, $p_0, p_1 \cdots p_n$ is a polynomial coefficient, the corresponding velocity $V(t)$, acceleration $A(t)$, and acceleration Sanp (t) can be expressed as

Input: Structured 2D plan map of the environment
Output:*m* motion trajectories
1 define the starting node, target node, and obstacles
2 parameter settings, swarm size, the adjustment coefficient q
3 for the structured map of the environment
4 discretize for using griding method in section 2.1
5 create a gridded mesh and discretized matrix
6 until all target nodes and obstacles are discretized on the mesh
7 initialize the open set and closed set of nodes by calling A∗ algorithm
8 while the open set is not empty
9 generate all direct path between nodes
10 end while//all nodes are reachable on the fully connected path graph
11 while two nodes is not on a direct path
12 calculate the unit energies for all not direct path
13 Endwhile modelling the problem of MTSP with energies
14 for each robot of *m*, using GPSO-AC (args)
15 generate *m* route trajectories that met the objectives
16 Endfor
17 for *m* route trajectories using minimum snap algorithm
18 trajectories optimization to reduce unnecessary turning points
19 Endfor
20 for *m* route trajectories using improve A∗ and RRT algorithm
21 solving trajectory polynomial and remove all breakpoints
22 Endfor generate m smoothed motion trajectories

Algorithm 1: Parallel swarm intelligent motion planning scheme.

Input: Open set, closed set for all nodes on a map
Output: Full direct path graph
1 add all nodes with obstacles to the closed set
2 set the starting node to the current node and add it to the open set.
3 for all reachable nodes
4 collect the current node that are not in the closed set
5 Endfor
6 while (the open set is not empty)
7 calculate *f(n)* of each reachable node, and add it to the open set
8 find a node with the smallest *f(n)* from the open set
9 set the node as the current node, and add it to the open set
10 set its parent node as the previous current node
11 delete the previous current node from the open set and add it to the closed set
12 for all reachable nodes in the open set
14 select the node with the smallest *g*(n) in the open combination
As the current node, and add it to the open set
15 End for
16 End do

Algorithm 2: Improved A∗ algorithm.

equations (6)–(8). Thus, the trajectory $X(t)$ can also be expressed in a matrix form as equation (9).

$$V(t) = X^{(1)}(t) = \sum_{n=0}^{k} p_n . n t^{n-1}, \tag{6}$$

$$A(t) = X^{(2)}(t) = \sum_{n=0}^{k} p_n . n(n-1) t^{n-2}, \tag{7}$$

$$Snap(t) = X^{(4)}(t) = \sum_{n=0}^{k} p_n \cdot \frac{n!}{(n-4)!} t^{n-4}, \tag{8}$$

$$X(t) = \left[1, t, t^2, \cdots, t^n\right] \cdot \left[p_0, p_1, \cdots, p_n\right]^T. \tag{9}$$

Similarly, the motion state description functions such as $X(t)$ and $A(t)$ can be expressed in matrix form. In path

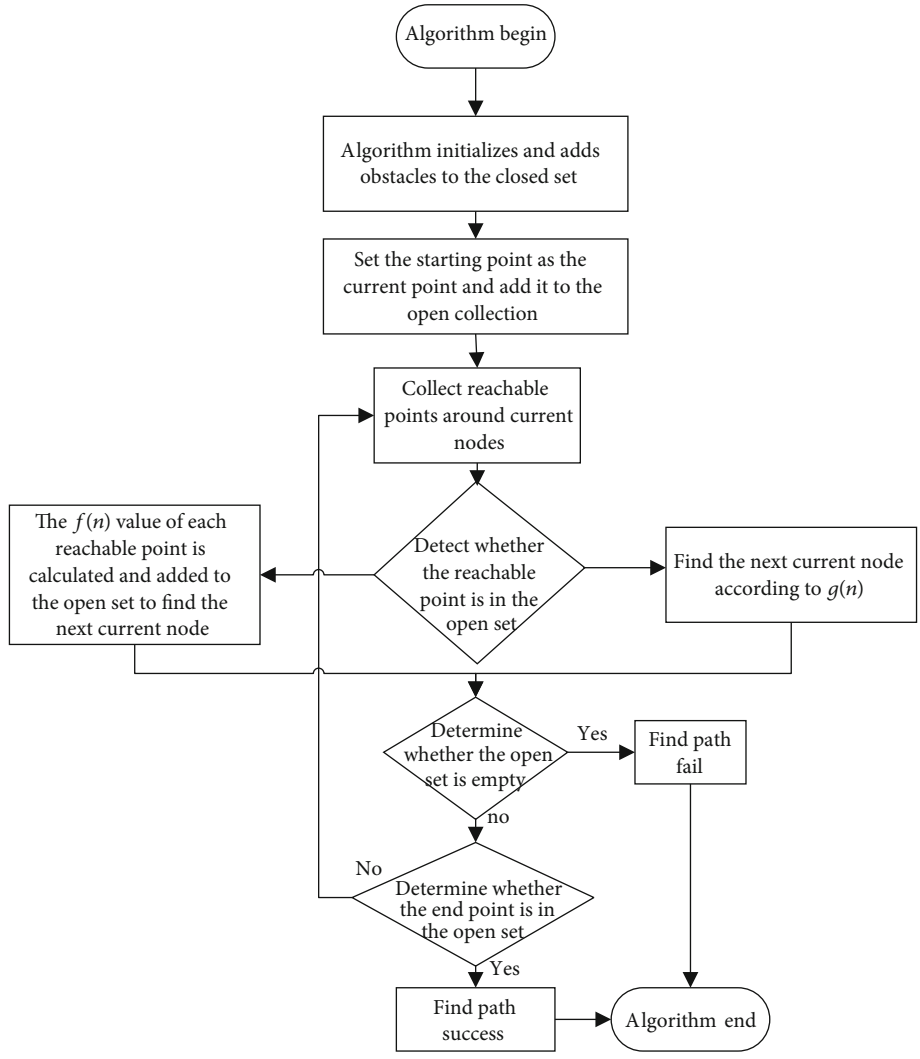


Figure 3: Flowchart of an improved A∗ algorithm.

planning, the multisegment trajectory between multiple target nodes can be denoted as equation (10).

$$X(t) = \begin{cases} \left[1, t, t^2, \cdots, t^n\right] \cdot P_1^T & t_0 \le t < t_1, \\ \left[1, t, t^2, \cdots, t^n\right] \cdot P_2^T & t_1 \le t < t_2, \\ \cdots & \\ \left[1, t, t^2, \cdots, t^n\right] \cdot P_k^T & t_{k-1} \le t < t_k. \end{cases} \tag{10}$$

To determine the trajectory of each path, it is necessary to define the value of each coefficient matrix P, minimize the Snap(t) function of multistage trajectories to obtain high-order continuous and smooth trajectories, that is, to ensure that the coordinates, velocities, and accelerations at the breakpoint between each trajectory are continuous, to ensure that the actual trajectory of the robot is in line with the dynamic constraints in the real world, and can maintain a steady-state continue to be in a stable state of motion. Considering that Snap(t) can be negative and square, the objective function of minimizing snap can be established as follows:

$$J(T) = \text{minimize}\left(\text{snap}(T)^2\right) = \int_{T_{i-1}}^{T_i} \left(X^{(4)}(t)\right)^2 dt = P^T Q P. \tag{11}$$

Among them, the total time of T segment trajectory, $\boldsymbol{P}$ is the parameter matrix of each section, and $\boldsymbol{Q}$ is the weight matrix. After the establishment of the objective function, the constraint conditions are specified, and the initial state values of the positions, velocities, and acceleration at both ends and internal nodes of a trajectory can be specified:

$$\sum_{n=0}^{k} p_n t^n = x_0, \sum_{n=0}^{k} p_n . n t^{n-1} = v_0, \sum_{n=0}^{k} p_n . n(n-1) t^{n-2} = a_0. \tag{12}$$

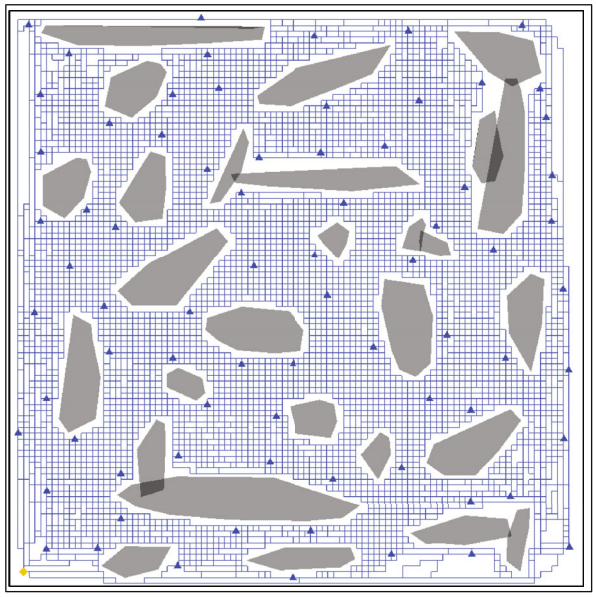


Figure 4: Fully connected path graph with obstacle avoidance.

Then, we can establish equality constraints on matrix $\boldsymbol{P}$:

$$AP_i = C, \tag{13}$$

where $\boldsymbol{A}$ is the coefficient matrix of T, and $\boldsymbol{C}$ is the initial matrix. It can be seen from equations (11)–(13) that what needs to be solved is a quadratic programming problem (QP). There are many algorithms for solving QPs omitted here. Similarly, we use the same method for y-axis to find the polynomial parameters.

3.2.3. Solve the Weighted MTSP Problem with Balanced Energy. The energy-balanced MTSP problem extends the MTSP problem modeled in Section 2.2. Considering the different traffic energies of each path section, we let the unit energy (UE) of connecting nodes i^{th} and j^{th} is v_{ij}, and the energy C_{ij} consumed by the robot traveling between nodes i and j is denoted as equation (14).

$$C_{ij} = d_{ij} \times \text{Energy}_{ij}, \tag{14}$$

where d_{ij} is the distance between nodes i and j. To find the total energy of all robots and the smallest route plan, the objective function equation (1) must be rewritten as equation (15). At the same time, it is required to minimize the variance of the energy of traveling between robots by equation (16).

$$\min Z = \sum_{k=1}^{m} \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} v_{ij} p_{ij}^{k}, \tag{15}$$

$$\min Y = \frac{\sum_{i=1}^{m} \left(C_i - \bar{C}\right)^2}{n}. \tag{16}$$

(1) Establish the Path Weight between Nodes. The goal of this stage is to transform the problem into a weighted MTSP problem in which the weights between nodes are the energies of obstacle constraints and path length. Based on the fully connected path graph with obstacle avoidance (as shown in Figure 4 generated by the improved A∗ algorithm), considering the increase of energy consumption between nodes due to the need to avoid obstacles, the unit energy (UE) consumed by a robot traveling two nodes is approximately defined as eq. (17).

$$\text{Energy} = \frac{|x_1 - x_2| + |y_1 - y_2|}{\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}}, \tag{17}$$

where the upper of eq. (17) actually is the Manhattan distance between two nodes, while the lower is the Euclidean distance directly connected between two nodes. If there are obstacles between two nodes, as shown in Figures 5 and 6 intercepted from the upper right corner of Figure 4, the unit energies (UEs) indicate the difficulties of moving on the direct paths between the two nodes, which are obviously more expensive than those without obstacles. The estimation matrix of UEs does not require additional information about obstacle avoidances. Therefore, when considering the obstructed MTSP problem with energies consumed by robots traveling between nodes, the objective is solved as eq. (15) that is a preparation for the optimal path of energy balance.

(2) Parallel PSO-AC Algorithm for Weighted MTSP with Energy-Balanced. In this section, we propose a hybridization of PSO and ACO algorithm to solve the weighted MTSPs. In our method, the PSO k-means algorithm is used to find clustering central nodes for each robot. Because the right initial clustering center can speed up the convergence and improve the clustering quality, this paper uses a rough classification method to classify the initial nodes by using equations (18) and (19), where the total energy R_i is consumed by a robot traveling from each node to other cities, and the total energy R is consumed by robots traveling the global path sections.

$$Ri = \sum_{j=0}^{n} d_{ij} v_{ij}, i = 0, 1, 2 \cdots, n, \tag{18}$$

$$R = \sum_{i=0}^{n} \sum_{j=0}^{n} d_{ij} v_{ij}, i \neq j. \tag{19}$$

Then, sort the distances from the samples to the clustering centers and then traverse the samples in turn. If the traversal condition satisfies equation (20), the nodes that have been traversed are classified into one route, and the clustering centers are updated using equation (21). Assume that m is the number of robots and the number of clusters, and f is a Boolean label variable. If the k^{th} robot visits i^{th} node, it will be 1, and otherwise, it will be 0. In the iterative process, the reclassification process is restricted by the condition of equation (20). The

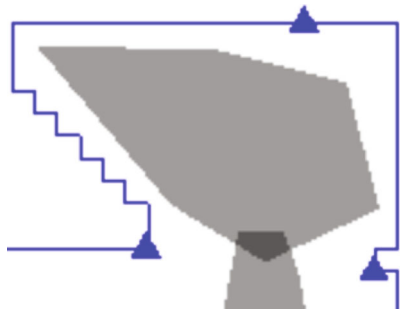


Figure 5: Path section with obstacle avoidance.

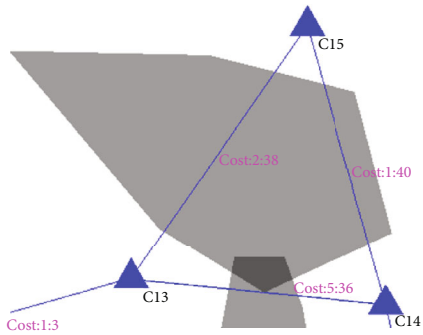


Figure 6: Sketch of UE between nodes.

purpose is to ensure that the total energy for each route is roughly balanced to m robots.

$$\sum_{i=0}^{n} f_{i^k} R_i \le \frac{R}{m}, \tag{20}$$

$$C_{jp} = \frac{\sum_{i=1}^{n} \omega_{ij} X_{ip}}{\sum_{i=1}^{N} \omega_{ij}}, \text{ where } \omega_{ij} = \begin{cases} 1 & 1 \text{ The sample belongs to route } j, \\ 0 & \text{The sample does not belong to route } j. \end{cases} \tag{21}$$

Next, ACO is utilized to calculate the minimum energy of traveling around a node that is currently classified as a route of nodes. After that, the pheromone and taboo tables of the algorithm are updated, and the fitness of the current particle, fitness, is calculating by equation (22), where C_k is the minimum energy, $S(C_k)$ represents the node dispersion of a route by variance function S, and the smaller the variance, the more balanced approximately the traversal energy consumption for robots. The normalization method, in addition, is used to uniformly quantify their energies and the dispersions as equation (22), where the adjusting coefficient q is [0,1]. When $q = 0$, the problem is a common weighted MTSP. On the other word, the larger q, the greater impact of the energy balance on the convergence of the algorithm.

$$\text{fitness} = \frac{C_k}{\sum_{i=0}^{t} C_k} + q \frac{S(C_k)}{\sum_{i=0}^{t} S(C_k)}. \tag{22}$$

To make full use of the parallel characteristics of the swarm algorithm, we present a hybridization of PSO and ACO to solve this stage of problem. The hybrid scheme is developed on GPU parallel computing platform with CUDA parallelism and multistream processor (SM), so as to enhance the convergence accuracy and executive efficiency of the hybrid swarm intelligence method described as Algorithm 3.

3.2.4. Trajectory Optimization and Smoothing

(1) Constructing the Optimal Grid Path. From equations (14) and (17), it can be seen that the energy consumed by a robot traveling between two nodes is simplified as $C_{ij} = |x_1 - x_2| + |y_1 - y_2|$, which is actually the Manhattan distance of the two nodes on direct path after the obstacle constraint is added. Thus, the total energy of a robot traveling around is the total length of a grid path with obstacle constraints. In summary, the balanced energy in the obstructed MTSP is also the equalized length of the gridded path in path planning. For the purpose of multirobot route planning, after an optimal weighted direct path is obtained, we need to convert the direct path as shown in Figure 7 to the path on which robots travel between nodes when bypassing the obstacles. At this stage, it is necessary to use the A∗ algorithm to restore the gridded path between the target nodes, so far, the initial planned path with obstacle avoidance is completed as shown in Figure 8.

(2) Smoothing Path. Because the generated path based on the gridded environment can only be turned by a multiple of 90°, there are a large number of turning points on the path like sawtooth as shown in Figure 9, which will increase both difficulty and traveling distance of the robot movement. In addition, too many invalid turning points will also hinder the polynomial trajectory optimization process, because turning points are used as constraints for trajectory optimization, and useless turning points must be deleted before trajectory optimization. Therefore, an improved RRT∗ is used to smooth trajectories that have fewer natural turning points to facilitate the follow-up operation of the minimum snap algorithm, so as to reduce useless nodes as much as possible to optimize the subsequent polynomial trajectory. If the raster path is not smoothed, there will be too many useless interference nodes, making the trajectory like S shape generated by the minimum snap algorithm [35], as shown in Figure 10. This paper also extends the A∗ algorithm and proposes a smooth grid path algorithm to reduce useless nodes, and finally, the smoothed trajectory can be obtained as shown in Figure 11.

(3) Solving Trajectory Polynomial. After the smooth processing, all breakpoints remaining on the path can be considered as the minimum effective breakpoints that are the necessary nodes to avoid obstacles, like the target nodes. The motion parameters can be quantified as the equality constraints in the minimum snap algorithm, and finally, used to solve the polynomial coefficients. Assumes that it takes time for the robot to perform the task at the target node, then, it can be considered that the speed and acceleration of the robot are 0 at this time, and at the same time, it needs to ensure that

```
Input: Data after modeling for each node in the MTSP problem
Output: The result of the MTSP problem
initialize swarm size of particles and ants
do
  Parallel_for particle in particle swarm
    parallel{update position and speed}
    parallel {//reclassification
      for cluster center in all cluster results
        calculate the distance between the sample and the current cluster center
        Sort sample by distance
        cluster center selection sample
      end
    }
    parallel {//calculate fitness
      for cluster center in all clustering results
        initialize the ant colony
        do
          Parallel_for ant in an ant colony
            parallel{search path}
            parallel{update pheromone}
          end
        while stop condition reached
        calculate optimal path fitness
      end
    }
    parallel{update individual optimal}
    parallel{update group optimal}
end
    while stop condition reached
```

Algorithm 3: GPSO-AC.

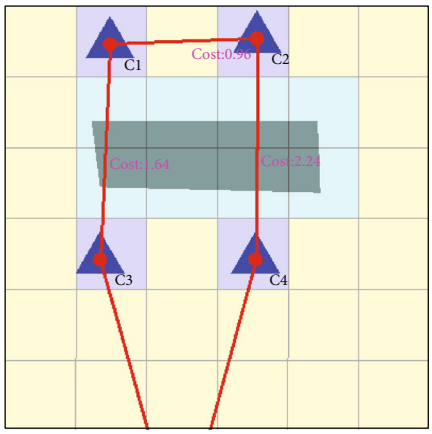


Figure 7: Gridded rough path section.

the robot does not change the motion state at the inflection node, that is, the polynomial of the speed and acceleration at the inflection node is smooth and continuous. The process of constructing equality constraints is given as follows.

Taking x-axis as an example, y-axis similarly, given a grid path L_{AB}, there are n breakpoints on it, and the coordinates of two endpoints are C_A and C_B. The L_{AB} can be divided into $n+1$ segments according to the C_n $(n=1,\cdots,n)$ coordinate at the breakpoint. First, make sure that the static acceleration of the robot at both ends is 0, described as equations (23)–(25), then, ensure that the robot runs to the breakpoint at T_n time, and at the breakpoints, the coordinates, velocities, and accelerations of the front and back trajectories should be consistent at t time.

$$\begin{cases} X(T_0)=C_A \\ X^{(1)}(T_0)=0 \\ X^{(2)}(T_0)=0 \end{cases} \Rightarrow A_{3\times1}(T_0)P=\begin{bmatrix} C_A \\ 0 \\ 0 \end{bmatrix},$$

$$\begin{cases} X(T_{n+1})=C_B \\ X^{(1)}(T_{n+1})=0 \\ X^{(2)}(T_{n+1})=0 \end{cases} \Longrightarrow A_{3\times1}(T_{n+1})P=\begin{bmatrix} C_B \\ 0 \\ 0 \end{bmatrix}, \tag{23}$$

$$\begin{cases} X(T_1)=C_1 \\ \cdots \\ X(T_n)=C_n \end{cases} \Longrightarrow \begin{cases} A_{1\times1}(T_1)P=C_1, \\ \cdots \\ A_{1\times1}(T_n)P=C_n, \end{cases} \tag{24}$$

$$\begin{cases} X_i(T)=X_{i+1}(T) \\ X_i^{(1)}(T)=X_{i+1}^{(1)}(T) \\ X_i^{(2)}(T)=X_{i+1}^{(2)}(T) \end{cases} \Longrightarrow [A_i(T)-A_{i+1}(T)]\begin{bmatrix} P_i \\ P_{i+1} \end{bmatrix}=0, \tag{25}$$

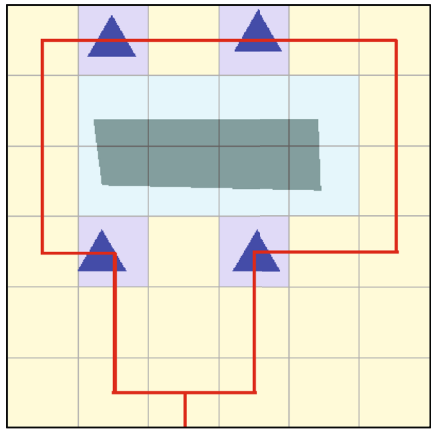


Figure 8: Gridded path section with obstacle avoidance.

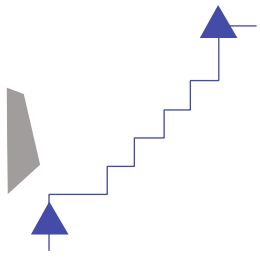


Figure 9: Serrated path.

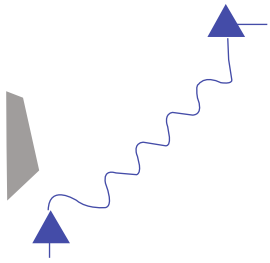


Figure 10: Optimized trajectory.

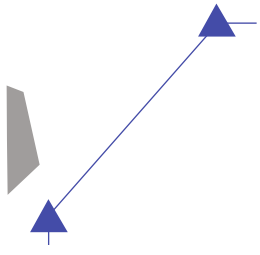


Figure 11: Smoothed trajectory.

where A is the weight matrix at time t, and P is the coefficient matrix to be solved. By taking these equality constraints into the QP objective function of equation (11), the P matrix can be obtained, and the polynomial analytic equation of the path planning trajectory can be obtained.

(4) Smooth Grid Path Algorithm. On the basis of the serrated path generated by the A∗ algorithm, aiming at the shortcoming of the path, a smooth raster path algorithm is proposed in this section to solve the invalid turning point on the serrated path. The algorithm involves two processes of trajectory optimization and smooth described in pseudocode as the following Algorithm 4. If there is no obstacle between the two target nodes, the invalid turning point is deleted.

4. Simulation Experiment

4.1. Experimental Platform and Experiment Settings. In order to verify the feasibility of the proposed method, the experimental platform as shown in Figure 12 is developed using C++ on a PC with CPU Intel® Core™ i5-8400, RAM 8 GB, GPU NVIDIA GeForce GTX 1060 under Linux Ubuntu 18.04.5 LTS. This platform provides some interactive functions including algorithm testing, map grid modeling, obstacle setting, MTSP path generation and optimization, motion trajectory planning, and data saving functions. In our platform development, an open-source SDL2.0 (Simple Direct-Media Layer) is used to the graphics layer rendering library which can make full use of GPU acceleration. For the convenience of debugging, we choose the rendering UI components (ImGUI) that support almost all C++ compilers, which can be embedded in the algorithm for debugging at any time. Because the core algorithm of path planning is to solve the weighted MTSP problem, this experimental platform also provides simulation support for MTSP algorithm experiments.

4.2. Benchmark Tests and Comparisons. The multirobot motion planning proposed in this paper is a complete solution, which involves a variety of algorithms, such as map modeling, parallel path search, and trajectory optimization and smoothing. We select several TSPLIB datasets to be used to evaluate the performance of our GPSO-AC in path search by metrics of mean running time (MeanRT) and average acceleration ratio (aveRatio).

First, for the dataset eil51 selected in TSPLIB, we assume that the number of traveling salesman is 3, the weight between nodes is set to 1 regardless of the cost balance, and the problem is a single node departure one. In this experiment, the proposed GPSO-AC algorithm and the PSO-AC algorithm [16] without GPU acceleration are run independently for 30 times, respectively, in different swarm sizes. The statistical results are shown in Table 1. From Table 1, it can be seen that PSO-AC on eil51 run much longer than the former, and its mean running time increases linearly with the swarm sizes, so it is difficult to application with high real-time requirement; in contrast, when the swarm size increases, the algorithm GPSO-AC on eil51 run much faster, and its mean running time increases less and the average acceleration ratio is greater, because the instruction alienation of clustering in GPU is greatly reduced, which enhances the parallel running efficiency.

Then, the proposed GPSO-AC algorithm is tested on 6 datasets by comparing with others PSO-AC [16], TPHA [18], and Kmeans-AC [33], assuming that the number of traveling salesman is set to 3, the swarm size is 64, the maximum iteration is 500, and the learning factors $C_1 = C_2 =$

Input: A continuous path on a raster map
Output: Path to remove useless kinks
1 index begin =0
2 index last =0
3 new route list record index (begin)
4 **while** (begin!=n) **do**
5 last=n
6 **while**(end >begin) **do**
7 **if**(no collision between connecting line and obstacles (begin, end)) **then**
8 new route list record index (last)
9 begin = last
10 **break**
11 **end if**
12 last = last -1
13 **end while**
14 **end while**

Algorithm 4: Smoothing grid path algorithm.

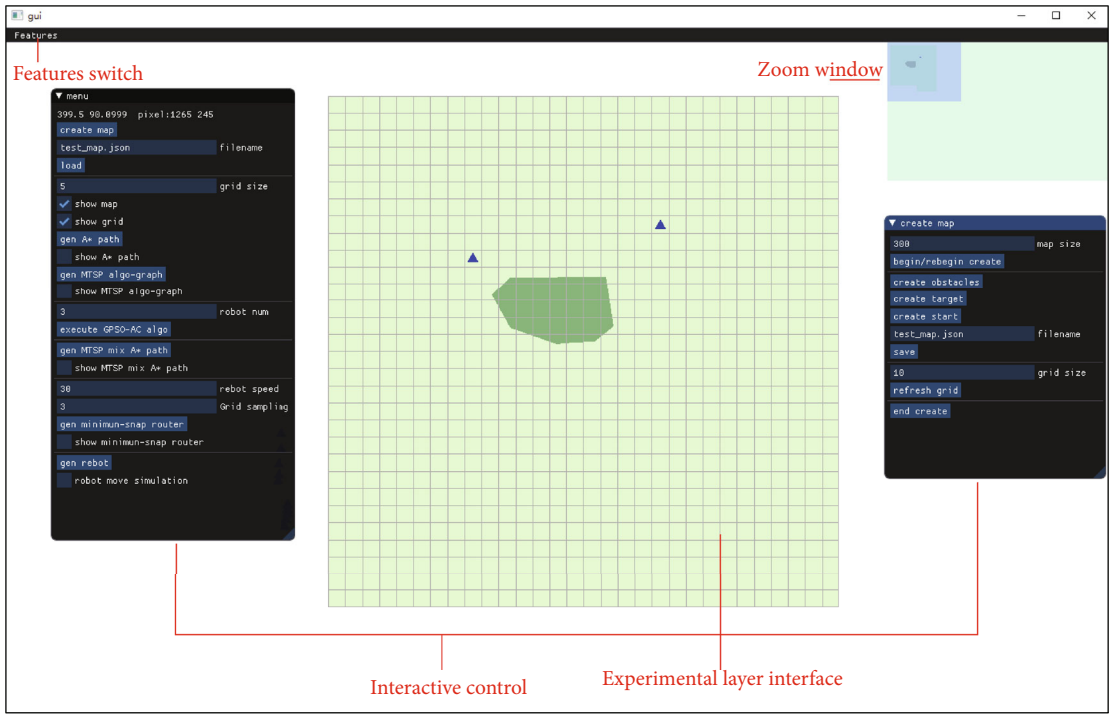


Figure 12: Multirobot path planning experimental simulation platform.

Table 1: Speedup ratio comparisons of runtimes.

Swarm size	Mean running time (MeanRT) GPU	CPU	Average acceleration ratio (AveRatio)
32	52.19 s	260.96 s	5
64	52.97 s	553.96 s	10.46
128	58.06 s	1145.35 s	19.73
256	59.07 s	1634.82 s	27.68

1.97, $\alpha = 1$, and $\beta = 3$ according to the previous literatures. The four algorithms run independently for 30 times for each problem, and the statistical metrics of their mean running time, worst solution, optimal solution, mean solution, and standard deviation (StD) are shown in Table 2. It can be seen from Table 2 that although the convergence result of PSO-AC is slightly better than the latter two algorithms, the mean running time is long; the algorithms follow the principle of classification before calculation, which runs faster than the first two hybrid algorithms, but the convergence accuracy is poor; the convergence accuracy of GPSO-AC is always

Table 2: Experimental results comparisons of four algorithms.

Problems	Performances	TPHA	*K*means-AC	PSO-AC	GPSO-AC
	MeanRT	0.23 s	0.42 s	461.28 s	47.11 s
	Worest	13184.9	13135.1	10706.1	10706.1
bayg29	Optimum	11791.1	11733	10403.2	10403.2
	Mean	12181	12494.5	10413.3	10413.3
	StD	538.296	375.314	55.31	55.31
	MeanRT	0.73 s	1.4 s	896.03 s	64.52 s
	Worest	9077.08	9042.33	8525.2	8591.67
berlin52	Optimum	8774.05	8809.45	8423.24	8470.35
	Mean	8974.17	8906.13	8493.73	8505.86
	StD	72.99	52.98	25.43	36.31
	MeanRT	1.38 s	2.72 s	1245.94 s	77.47 s
	Worest	1013.24	992.144	855.231	867.028
st70	Optimum	977.15	964.325	830.09	826.925
	Mean	989.44	979.85	841.94	77.47
	StD	8.71	7.24	9.52	9.41
	MeanRT	3.23 s	6.19 s	2117.81 s	158.65 s
	Worest	847.11	826.824	706.65	712.31
eil101	Optimum	780.26	776.452	695.27	681.1
	Mean	802.6	796.72	702.55	690.9
	StD	17.27	14.41	3.05	7.95
	MeanRT	8.47 s	16 s	3669.83 s	237.8 s
	Worest	9072.97	8873.1	7607.78	7022.99
ch150	Optimum	8488.81	8504.53	7347.33	6673.28
	Mean	8810.75	8719.8	7432.4	6833.43
	StD	125.1	93.6	68.43	101.06
	MeanRT	18.11 s	33.63 s	4603.84 s	292.34 s
	Worest	38417.8	37485.6	33820.9	33603.5
kroA200	Optimum	36926.4	36563.1	31940.7	31271.1
	Mean	37609.7	37076.2	32740.3	32723.6
	StD	377.98	279.49	442.56	547.5

better than others within a reasonable time, as two algorithms use clustering to effectively group nodes. TPHA only considers grouping and does not optimize the grouping in the later stage, resulting in poor convergence results.

4.3. Demonstration of Experimental Results. In the experiment, we assume that the proposed methods are used to planning the motion trajectory for 3 robots that visit the nodes to finish tasks in the environment as shown in Figure 13, and the original map is discretely gridded as a mesh scene as shown in Figure 14. Then, the following Figures 15–18 demonstrate the intermediate results of the problem solving. Figure 15 shows a solution after modeling the map information into an MTSP problem which is a rough solution without considering obstacle constraints. While a gridded path with obstacle constraints is generated by the improved A∗ algorithm as shown in Figure 16, from that, there are many useless turning points on the path at this time.

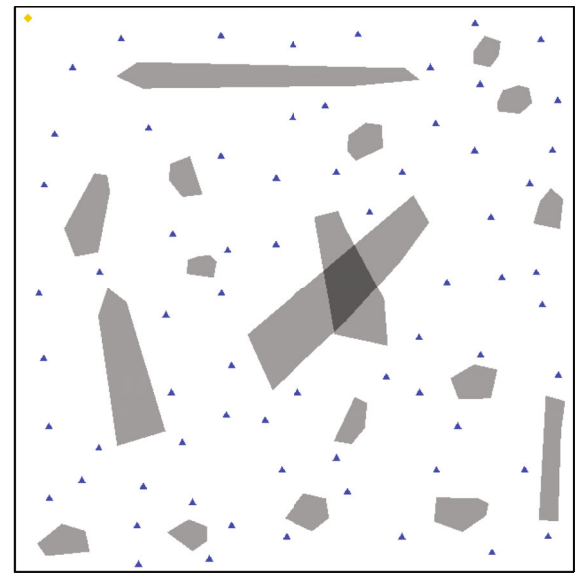


Figure 13: Original map.

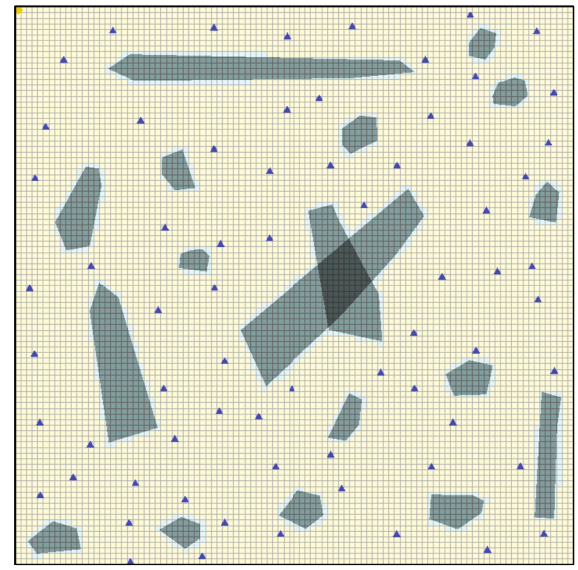


Figure 14: Gridded map.

On the basis of Figure 16, after optimizing the path using the smooth grid path algorithm, a new optimized trajectory is obtained as shown in Figure 17. The path does not have any meaningless turning points on the basis of ensuring that the path does not collide with obstacles. Finally, a path that meets the dynamic constraints is generated by using the minimized snap algorithm. As shown in Figure 18, it can be seen that at the turning point between the two target nodes, the path trajectory becomes smooth and arc-shaped, which conforms to the actual situation of the robot.

4.4. Experimental Result Analysis. In the experiment, we assume that the proposed methods with parameters the same as the above subsections are tested to planning the motion trajectory for 4 robots that visit 75 nodes to finish tasks in the environment with 28 obstacles as shown in

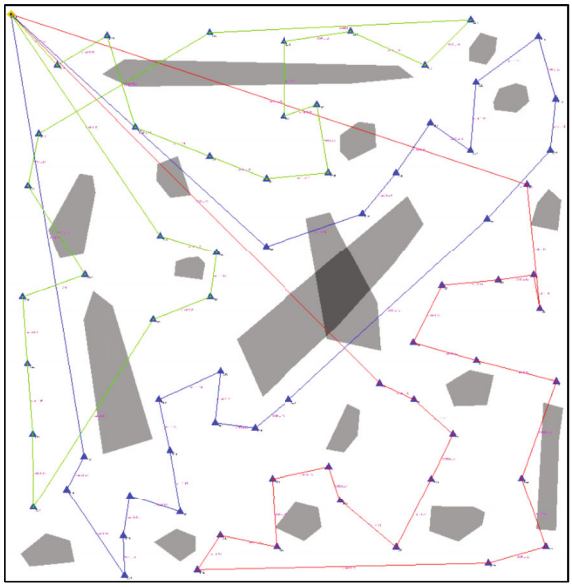


Figure 15: Results of solving MTSP problem algorithm.

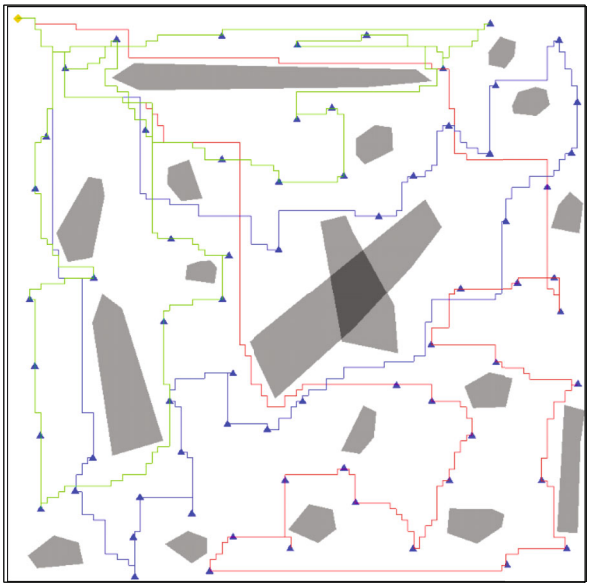


Figure 16: Building the grid path using A∗ algorithm.

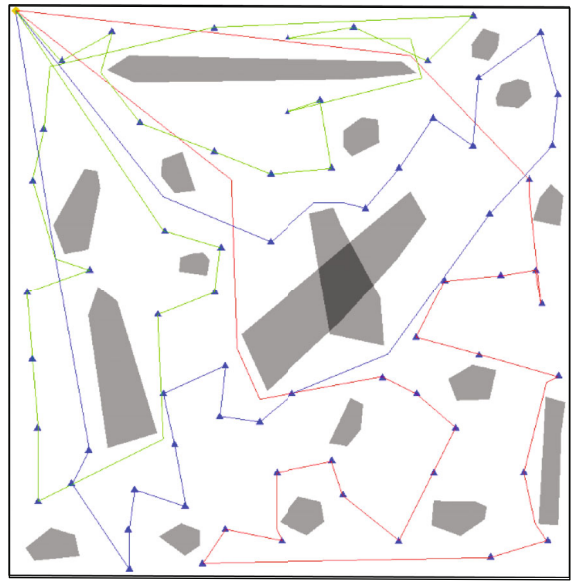


Figure 17: New path after smoothing grid path.

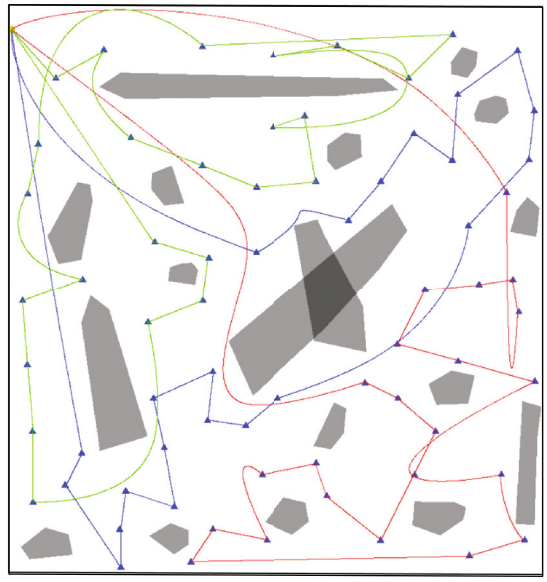


Figure 18: Optimized path with the minimum-snap algorithm.

Figure 2. The two algorithms run independently for 20 times under different adjustment coefficient q, and the statistical metrics of the total distance and standard deviation for each robot are recorded in Table 3 to investigate the adjustment coefficient of influence on generation path before and after optimization, while the relationship between the adjustment coefficient and the total length of the path and the relationship between the adjustment coefficient and the standard deviation of its total length for each robot are plotted in Figures 19 and 20, respectively.

From Table 3 and Figures 19 and 20, we can see that the total length of the path after smoothing is much shorter than that before, which shows that the proposed smooth grid path algorithm is effective and greatly reduces the energy consumption of the robot. Meanwhile, without considering the balance of energy consumption, i.e., setting the adjustment coefficient $q = 0$, the path length deviation of each robot is the largest, at this time, the total path length is the shortest. Then, gradually increasing the value of q, it is found that the standard deviation of the grid path will decrease rapidly whether it is smoothed or not, and then as the value of q continues to increase, the standard deviation tends to a smaller value. According to the proposed algorithm, we find that the energy of path smoothing increases rapidly with the increase of the standard deviation of path length, which demonstrates that the equalization algorithm is effective, and balancing the path length of each robot is the guarantee of balancing energy consumption. In addition, as the adjustment coefficient q increases, the path length shows an upward trend, because at this stage, the algorithm focuses

Table 3: Influence of different adjustment coefficients on generation path.

Adjustment coefficient q	Before and after optimization	Robot 1	Robot 2	Robot 3	Robot 4	Total distance	Standard deviation
$q = 0$	A∗ grid path	1650	2220	2380	1500	7750	428.05
	Smoothed grid path	1403.52	1893.3	1938.27	1273.68	6508.77	337.92
$q = 0.025$	A∗ grid path	2030	1540	2470	1800	7840	394.55
	Smoothed grid path	1709.26	1322.25	2142.84	1521.13	6695.48	350.31
$q = 0.05$	A∗ grid path	1690	2260	2250	1680	7880	329.14
	Smoothed grid path	1404.08	1887.27	1781.56	1442.54	6515.45	241.75
$q = 0.075$	A∗ grid path	1910	2200	1930	1960	8000	134.9
	Smoothed grid path	1608.63	1825.32	1697.47	1619.06	6750.48	99.99
$q = 0.1$	A∗ grid path	1980	2080	2000	1990	8050	45.73
	Smoothed grid path	1672.18	1742.48	1659.95	1667.45	6742.05	38.31
$q = 0.15$	A∗ grid path	2060	2050	2120	2080	8310	30.96
	Smoothed grid path	1728.8	1697.91	1754.52	1825.25	7006.48	54.27
$q = 0.2$	A∗ grid path	2140	2110	2140	2170	8560	24.49
	Smoothed grid path	1854.17	1741.49	1761.51	1909.58	7266.75	79.02
$q = 0.25$	A∗ grid path	2160	2170	2140	2130	8600	18.26
	Smoothed grid path	1855.9	1772.03	1694.56	1851.21	7173.71	76.32
$q = 0.3$	A∗ grid path	2200	2190	2190	2180	8760	8.16
	Smoothed grid path	1899.91	1804.66	1878.38	1895.44	7478.39	44.27
$q = 0.4$	A∗ grid path	2240	2230	2230	2240	8940	5.77
	Smoothed grid path	1887.13	1845.57	1867	1973.62	7573.31	56.16
$q = 0.5$	A∗ grid path	2280	2270	2270	2270	9090	5
	Smoothed grid path	1877.95	1929.97	1998.08	1865.28	7671.27	60.38
$q = 0.6$	A∗ grid path	2230	2230	2230	2240	8930	5
	Smoothed grid path	1876.77	1852.4	1947.23	1856.31	7532.72	44.02
$q = 0.7$	A∗ grid path	2190	2200	2200	2200	8790	5
	Smoothed grid path	1786.8	1847.49	1891.22	1840.25	7365.78	42.82
$q = 0.8$	A∗ grid path	2220	2230	2230	2230	8910	5
	Smoothed grid path	1840.44	1847.1	1788.94	1926.37	7402.85	56.74

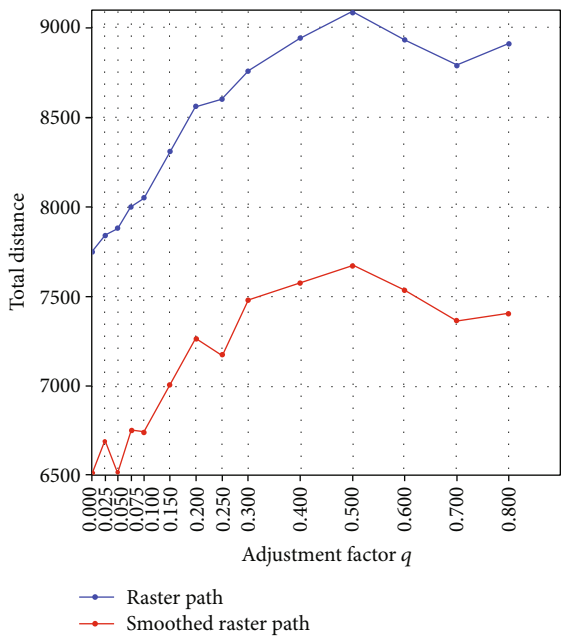


Figure 19: Effects of adjustment coefficient on the path lengths.

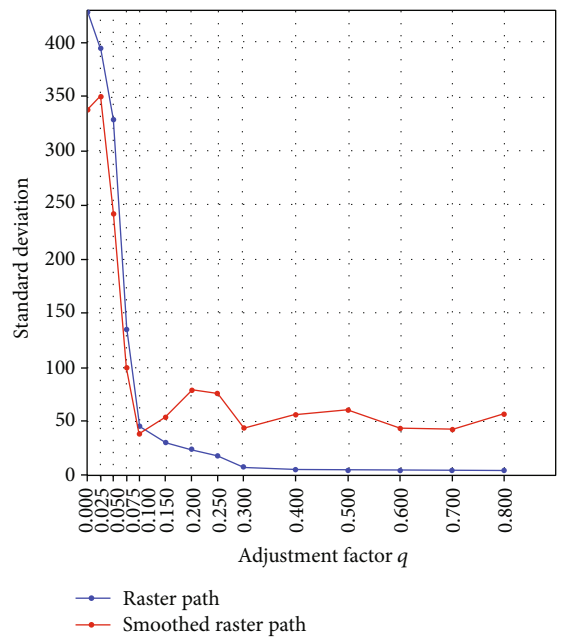


Figure 20: Influence of adjustment coefficient on the deviation of path length.

on the energy balance when calculating the fitness score and ignores the impact of the global optimal energy on the fitness. Therefore, it can be concluded that in the path standard, it is meaningless to continue to increase the value of q after the difference tends to a stable small value. Instead, it will increase the path length and increase the energy consumption of the robot.

5. Conclusion

Multirobot motion path planning is one of the hottest topics in robot technology. It is particularly difficult for multiple mobile robots in a complex obstacle environment where all robots need to execute their tasks, respectively. To speed up the running speed and guarantee the power supply of robots in a limited time, an energy-balanced parallel swarm intelligent optimizer (GPSO-AC) based on GPUs is presented for multirobot to find the shortest path in an obstacle environment where the energy consumption balance in the process of robot movement is modeled as a weighted MTSP problem. The presented method combines with grid constructing and improved the A∗ searching, smooth gridding, and minimum shaping algorithms to find a feasible oscillation-free, sharp turn-free, and collision-free path. To test the effectiveness of the proposed method, we develop an experimental platform based on GPUs where several scenes with multiple irregularly-shaped obstacles are generated to verify the performance of the presented method. In our experiments, the proposed GPSO algorithm is test on several datasets to evaluate the performances by comparing other similar approaches, and our method on different scenes has found the shortest path in which the multirobot can faster, smoother, and energy-balanced complete the tasks in a reasonable time. In practical applications, the trade-off between the shortest path and the minimum path length should be made. According to the experimental conclusion, the q value should be increased after the path length deviation is taken to a small value within a reasonably acceptable range. However, if each robot is regarded as a dynamic obstacle, it should be further studied in the future.

Data Availability

All experimental data and platforms used to support the findings of this study are available from https://ds3.jit.edu.cn/data/8902328Datasets&Results.rar, or from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The work is partially supported by the Scientific Research Foundations and the Virtual Experimental Class Projects of Jinling Institute of Technology (nos. JIT-rcyj-201505 and D2020005), the National Natural Science Foundation of China (nos. 61375121 and 41801303), the Natural Science Foundation of Jiangsu Province (no. BK200170116), and sponsored by the funds from Jiangsu Provincial Sci-Tech Innovation Team of Swarm Computing and Smart Software led by Prof S.B. Su (corresponding author).

References

[1] A. Madridano, A. Al-Kaff, and D. Martín, "Trajectory planning for multi-robot systems: methods and applications," *Expert Systems with Applications*, vol. 173, no. 1, p. 114660, 2021.

[2] Z. Qadir, F. Ullah, H. S. Munawar, and F. al-Turjman, "Addressing disasters in smart cities through UAVs path planning and 5G communications: a systematic review," *Computer Communications*, vol. 168, no. 1, pp. 114–135, 2021.

[3] F. Rubio, F. Valero, and C. Llopis-Albert, "A review of mobile robots: concepts, methods, theoretical framework, and applications," *International Journal of Advanced Robotic Systems*, vol. 16, no. 2, article 172988141983959, 2019.

[4] P. Li, X. Wang, H. Gao, X. Xu, M. Iqbal, and K. Dahal, "A dynamic and scalable user-centric route planning algorithm based on polychromatic sets theory," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, pp. 1–11, 2021.

[5] Y. Wu and K. H. Low, "An adaptive path replanning method for coordinated operations of drone in dynamic urban environments," *IEEE Systems Journal*, vol. 14, pp. 1–12, 2020.

[6] B. Zarebavani, F. Jafarinejad, M. Hashemi, and S. Salehkaleybar, "cuPC: CUDA-based parallel PC algorithm for causal structure learning on GPU," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 3, pp. 530–542, 2020.

[7] N. Zafar and J. C. Mohant, "Methodology for path planning and optimization of mobile robots: a review," *Procedia Computer Science*, vol. 133, no. 3, pp. 141–152, 2018.

[8] K. Shah, G. Ballard, A. Schmidt, and M. Schwager, "Multidrone aerial surveys of penguin colonies in Antarctica," *Science Robotics*, vol. 5, no. 47, p. eabc3000, 2020.

[9] X. Zhong, J. Tian, H. Hu, and X. Peng, "Hybrid path planning based on safe a algorithm and adaptive window approach for mobile robot in large-scale dynamic environment," *Journal of Intelligent & Robotic Systems*, vol. 99, no. 1, pp. 65–77, 2020.

[10] A. Candra, M. A. Budiman, and K. Hartanto, *Dijkstra's and A-Star in Finding the Shortest Path: a Tutorial Inter Conf Data Science*, Artificial Intelligence, and Business Analytics (DATABIA), IEEE CS, 2020.

[11] S. M. Bagheri, H. Taghaddos, A. Mousaei, F. Shahnavaz, and U. Hermann, "An A-Star algorithm for semi-optimization of crane location and configuration in modular construction," *Automation in Construction*, vol. 121, no. 1, p. 103447, 2021.

[12] K. R. Jensen-Nau, T. Hermans, and K. K. Leang, "Near-optimal area-coverage path planning of energy-constrained aerial robots with application in autonomous environmental monitoring," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 3, 2021.

[13] J. Wang, N. Wu, and W. X. Zhao, "Empowering A search algorithms with neural networks for personalized route recommendation," in *Proceedings of the 25th Inter Conf Knowledge Discovery & Data Mining (ACM SIGKDD)*, pp. 539–547, 2019.

[14] Z. Tian, C. Guo, and Y. Liu, "An improved RRT robot autonomous exploration and SLAM construction method," in *2020 5th International Conference on Automation, Control and Robotics Engineering (CACRE)*, pp. 612–619, IEEE, 2020.

[15] Y. Wu, "A survey on population-based meta-heuristic algorithms for motion planning of aircraft," *Swarm and Evolutionary Computation*, vol. 62, p. 100844, 2021.

[16] R. Uriol and A. Moran, "Mobile robot path planning in complex environments using ant colony optimization algorithm," in *3rd inter Conf control, automation and robotics (ICCAR)*, pp. 15–21, Nagoya, Japan, 2017.

[17] B. Y. Song, Z. D. Wang, and L. Zou, "An improved PSO algorithm for smooth path planning of mobile robots using continuous high-degree Bezier curve," *Applied Soft Computing Journal*, vol. 100, no. 1, p. 106960, 2021.

[18] Y. Wu, K. H. Low, B. Pang, and Q. Tan, "Swarm-based 4D path planning for drone operations in urban environments," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 8, pp. 7464–7479, 2021.

[19] X. L. Xu, H. Yuan, M. Liptrott, and M. Trovati, "Two phase heuristic algorithm for the multiple-travelling salesman problem," *Soft Computing*, vol. 22, no. 19, pp. 6567–6581, 2018.

[20] H. K. Paikray, P. K. Das, and S. Panda, "Improved shuffled frog leaping algorithm for path planning of multiple mobile-robot," in *2nd Int Conf Innovations in Electronics, Signal Processing and Communication (IESC)*, pp. 132–137, Shillong, India, 2019.

[21] A. V. Le, R. Parween, P. T. Kyaw, R. E. Mohan, T. H. Q. Minh, and C. S. C. S. Borusu, "Reinforcement learning-based energy-aware area coverage for reconfigurable hRombo tiling robot," *IEEE Access*, vol. 8, pp. 209750–209761, 2020.

[22] J. Ichnowski, Y. Avigal, V. Satish, and K. Goldberg, "Deep learning can accelerate grasp-optimized motion planning," *Science Robotics*, vol. 5, no. 48, p. eabd7710, 2020.

[23] Y. Wu, S. Wu, and X. Hu, "Cooperative path planning of UAVs & UGVs for a persistent surveillance task in urban environments," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4906–4919, 2021.

[24] S. Shukla, N. Shukla, and V. K. Sachan, "Multi robot path planning parameter analysis based on particle swarm optimization (PSO) in an intricate unknown environments," in *Int Conf Issues and Challenges in Intelligent Computing Techniques (ICICT)*, pp. 1–10, Ghaziabad, India, 2019.

[25] N. Salah, M. Hassen, and K. Bouallegue, "A multi-scroll chaotic system for a higher coverage path planning of a mobile robot using flatness controller," *Chaos, Solitons and Fractals*, vol. 118, no. 2, pp. 366–375, 2019.

[26] F. Imeson and S. L. Smith, "An SMT-based approach to motion planning for multiple robots with complex constraints," *IEEE Transactions on Robotics*, vol. 35, no. 3, pp. 669–684, 2019.

[27] A. Muralidharan and Y. Mostofi, "Path planning for minimizing the expected cost until success," *IEEE Transactions on Robotics*, vol. 35, no. 2, pp. 466–481, 2019.

[28] H. Gao, W. Huang, and X. Yang, "Applying probabilistic model checking to path planning in an intelligent transportation system using mobility trajectories and their statistical data," *Intelligent automation and soft computing*, vol. 25, no. 3, pp. 547–559, 2019.

[29] C. Tatino, N. Pappas, and D. Yuan, "Multi-robot association-path planning in millimeter-wave industrial scenarios," *IEEE Networking Letters*, vol. 2, no. 4, pp. 190–194, 2020.

[30] F. Bayat, S. Najafinia, and M. Aliyari, "Mobile robots path planning: electrostatic potential field approach," *Expert Systems with Applications*, vol. 100, no. 6, pp. 68–78, 2018.

[31] H. Gao, C. Liu, Y. Li, and X. Yang, "V2VR: reliable hybrid-network-oriented V2V data transmission and routing considering RSUs and connectivity probability," *IEEE Transactions on Intelligent Transportation Systems (T-ITS)*, vol. 22, no. 6, pp. 3533–3546, 2021.

[32] F. Zhang, T. Wang, Q. Li, and J. Xin, "An iterative optimization approach for multi-robot pattern formation in obstacle environment," *Robotics and Autonomous Systems*, vol. 133, no. 7, p. 103645, 2020.

[33] M. M. De Almeida, R. Moghe, and M. Akella, "Real-time minimum snap trajectory generation for quadcopters: algorithm speed-up through machine learning," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 683–689, IEEE, 2019.

[34] G. Tang, Z. P. Hou, and X. Hu, "UAV polynomial trajectory optimization based on minimizing snap method," *Computer Application Research*, vol. 38, no. 5, pp. 1455–1458, 2021.

[35] M. G. B. Atia, H. El-Hussieny, and O. Salah, "A supervisory-based collaborative obstacle-guided path refinement algorithm for path planning in wide terrains," *IEEE Access*, vol. 8, pp. 214672–214684, 2020.