Dual-Layer Dynamic Path Optimization for Airport Ground Equipment Using Graph Theory and Adaptive Genetic Algorithms

Pinzheng Qian
School of Transportation, Southeast University, Nanjing 211189, China
E-mail: pinzhengqian@163.com

Keywords: graph theory; conflict avoidance; airport; special unmanned equipment vehicle; path planning; genetic algorithm

Received: November 23, 2024

With the continuous promotion of smart airport construction, the application of unmanned ground equipment vehicles in airports is becoming increasingly popular. The existing path planning methods rely on manual management, which has problems such as low efficiency and poor ability to respond to unexpected situations. Given this, this study first models the airport path based on graph theory. Secondly, a two-layer dynamic path optimization algorithm is designed by combining the improved dynamic programming Dijkstra algorithm with the introduced genetic algorithm for path conflict identification and avoidance mechanisms. Performance test results showed that the running time of the improved Dijkstra algorithm was shortened by about 48.29% and the number of edges was reduced by 10.07%. The fitness value and variance of the improved genetic algorithm increased by 8.99% and decreased by 66.67%, respectively. In a high-density and high-frequency conflict environment, the path success rate of the proposed model was 90.1%, which was 7.2% higher than the comparative algorithm. In addition, its path smoothness standard deviation was 5.32°, better than 6.68° and 8.47° of the comparative algorithms. The results indicate that the optimized path planning method can reduce the total running time and effectively avoid path conflicts between multiple vehicles, providing certain technical support and new ideas for the safe operation of airport special unmanned equipment vehicles.

Povzetek: Dvoslojni algoritem načrtuje dinamično optimizacijo poti letaliških talnih vozil. S kombinacijo grafovske teorije in izboljšanih genetskih algoritmov rešuje izzive načrtovanja poti in izogibanja konfliktom.

1 Introduction

With the rapid growth of the global aviation industry, the business volume and scale of airports continue to expand, and the efficiency of ground support services has become an important factor affecting the operational capacity of airports [1]. Especially in large airports, various special equipment vehicles undertake heavy ground support tasks, such as shuttle buses, refueling trucks, luggage trucks, etc. [2-3]. In recent years, the concept of smart airports has gradually emerged. Unmanned Equipment Vehicles (UEVs), as an important component, can replace manual operations through autonomous driving technology and improve the intelligence level of ground services. However, the popularity of UEVs has also brought new challenges, especially in path planning and multi-vehicle coordination [4-5]. The main challenge currently faced by airport operations is how to effectively plan paths under complex conditions of multiple vehicles, multiple starting points, and multiple endpoints. Chen Y et al. proposed a method minimizing the number of paths with a length of 0 in a given graph and proved that the problem can be addressed in polynomial time. Meanwhile, further experiments demonstrated another variant that minimized the total number of paths with lengths of 0 and 1 was also polynomial time solvable [6]. Hu H et al. proposed a method based on multi-agent deep deterministic policy gradient and discretized the scene generated by the node network. By constructing a node network model, they analyzed the situations of opposing conflicts and same-point occupancy conflicts and established an integer programming model for solving the shortest path [7]. Li P et al. proposed a priority-free Ant Colony Optimization (ACO) algorithm. They also proposed energy-constrained ACO for single-robot energy optimal path planning by modeling unstructured rough terrain through dual-resolution grid maps [8]. Ren Z et al. proposed a conflict-based multi-objective search method, which utilized conflict-based search algorithms and the principle of superiority in multi-objective optimization to address the curse of search space dimensions caused by the increase in the number of agents [9].

The main objective of Vehicle Routing Problem (VRP) is usually to reduce transportation costs, decrease the required number of vehicles, or optimize delivery efficiency. Zhao N et al. designed an aircraft taxiing Path Planning Method (PPM) based on a spatiotemporal network model and dynamic programming algorithm. They constructed a 2D-directed graph of airport network nodes and combined it with Dijkstra's algorithm to plan the shortest taxiing distance path for the aircraft [10]. He

K et al. proposed an enhanced fast random exploration tree algorithm based on clustering and pre-search. This algorithm constructed an undirected graph using Markov clustering techniques and employs Dijkstra's algorithm for pre-search [11]. Pan Y proposed an autonomous PPM based on intelligent optical sensors and an ACO algorithm. It was found that the new model improved by an average of about 17.8% in four evaluation indicators [12]. Zhang Z et al. proposed a general robust Reference Path Selection Method. This method maintained a dynamic path array containing newly discovered paths, historical optimal paths, and local trajectories, and selected the path with the lowest cost through unified criteria evaluation [13].

At the same time, many scholars have conducted in-depth research in areas such as path optimization, target detection, and autonomous capability enhancement. Yang X et al. proposed an anchor-free and efficient real-time damage detection method YOLOv6s-GRE and combined it with re-parameterization optimization and quantization technology to improve the detection efficiency and energy consumption performance [14]. Tyagi S et al. discussed the cutting-edge applications of deep reinforcement learning in drone tactical deployment. The study highlighted the advancement of drone intelligence in surveillance and defense and looked forward to the potential of deep reinforcement learning in enhancing drone autonomy [15]. Liu Z et al. proposed a lightweight and memory-efficient target detection model to meet the real-time and accuracy requirements of target detection of UAVs in traffic systems. The model combined the FasterNet-16 backbone network and multi-scale feature fusion technology to achieve high-quality target detection [16]. Table 1 summarizes the related research, highlighting the methods, datasets, and key results.

Table 1: Comparison of related works

References	Method	Dataset	Key Results
Chen Y [6]	Minimum nontrivial path cover algorithm	Conflict graph	Proved that multiple variations of the path cover problem are polynomial-time solvable.
Hu H [7]	MADDPG (Multi-Agent Deep Deterministic Policy Gradient)	Automated container terminal scenarios	Solved conflict issues through a node network model.
Li J [8]	PFACO (Priority-Free ACO Algorithm)	Dual-Resolution grid data	Reduced multi-robot motion costs in rough terrain.
Ma H [9]	MO-CBS and its variants	Simulated multi-agent scenarios	Computed Pareto optimal solutions for multi-agent path planning problems.
Wang Y [10]	Combines time-space network with the Dijkstra	Six planes at Xi 'an Xianyang Airport	Effectively planned conflict-free paths.
Zhang T [11]	ERCP (Enhanced RRT* with clustering and pre-searching)	Unspecified	Achieved a good balance between path quality and search efficiency.
Liu Y [12]	Autonomous path planning with intelligent optical sensors and ACO	Industrial AGV scenarios	Performance indicators improved by approximately 17.8%.
Kim S [13]	RPSM (Robust Reference Path Selection Method)	Unspecified	Reduced trajectory distance, direction changes, and execution time.
Yang X [14]	YOLOv6s-GRE	Concrete bridge damage dataset	Increased detection speed by 83.5%, saved 79.7% energy.
Tyagi S [15]	Deep Reinforcement Learning	Unspecified	Enhanced UAV autonomy.
Liu Z [16]	Memory-efficient lightweight object-detection algorithm	VisDrone, COCO, Traffic-Net	Improved real-time object detection accuracy.

As illustrated in Table 1, the process of planning the routes of UEVs in complex airport environments presents numerous challenges. These include the efficient management of path conflicts among multiple vehicles dispatched concurrently, the optimization of path length under conditions of multiple starting points and multiple endpoints, and the prompt response to emergencies in dynamic environments. Although many studies attempt to improve the accuracy and efficiency of path planning by introducing graph theory and path planning algorithms, conflict avoidance mechanisms, and intelligent optimization methods. However, there are still problems such as unsatisfactory avoidance of path conflicts, insufficient global search capability of algorithms, and low computational efficiency.

Given this, this study proposes a two-level dynamic path planning optimization algorithm based on graph theory and conflict avoidance. The research innovation is mainly reflected in the following points: First, topological modeling based on graph theory optimizes the expression of path nodes and edges, and improves modeling efficiency and accuracy. Secondly, the Dijkstra algorithm is improved and an Adaptive Path Search Dijkstra Algorithm (APSDA) is proposed. The dynamic planning strategy is used to reduce redundant node access and calculation, thereby improving the path planning efficiency. Finally, conflict avoidance and dynamic time period optimization strategies are introduced into the Genetic Algorithm (GA), and a Dynamic Adaptive Genetic Algorithm (DAGA) is proposed, which enhances the path conflict avoidance capability and global optimization effect. The combination of APSDA and DAGA forms a two-layer dynamic path optimization framework. It has achieved a comprehensive improvement in path planning efficiency and safety.

2 Methods and materials

2.1 Airport VRP modeling based on graph theory

To meet different ground service needs, the airport uses various types of UEVs. These vehicles possess autonomous driving capabilities and can perform various specific ground support tasks, including passenger transportation, luggage handling, and aircraft maintenance [17-18]. Figure 1 shows a common classification of airport ground support vehicles.

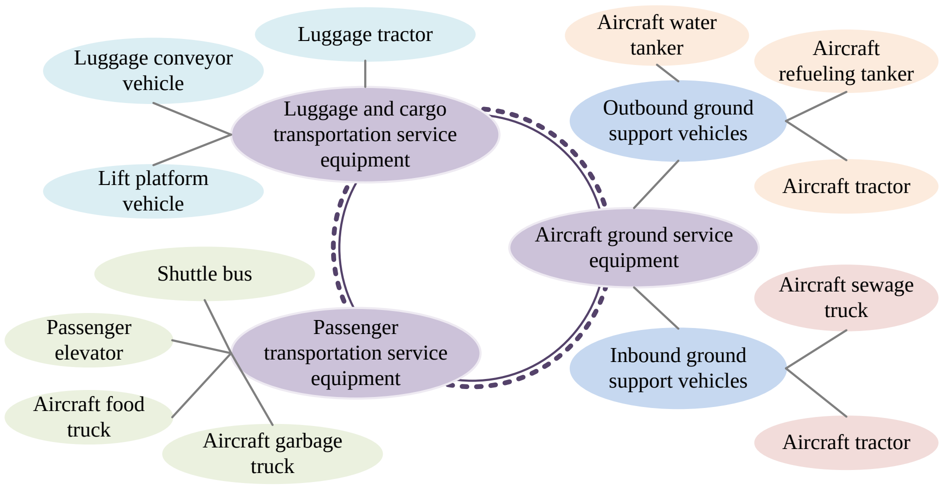

Luggage tractor
Luggage conveyor vehicle
Luggage and cargo transportation service equipment
Lift platform vehicle
Aircraft water tanker
Aircraft refueling tanker
Outbound ground support vehicles
Aircraft tractor
Aircraft ground service equipment
Shuttle bus
Passenger elevator
Passenger transportation service equipment
Aircraft food truck
Aircraft garbage truck
Aircraft sewage truck
Inbound ground support vehicles
Aircraft tractor

Figure 1: Airport ground support vehicle structure.

Figure 1 shows the classification of airport ground support vehicles, covering aircraft support equipment, passenger transportation equipment, and baggage and cargo transportation equipment. The study's objective is to analyze the key supporting roles of seven frequently used vehicles in passenger transportation, fuel supply, and baggage transfer. The vehicles selected for the study include shuttle buses, refueling trucks, water supply trucks, baggage transfer trucks, airline food trucks, health service trucks, and tractors. The path optimization and conflict avoidance capabilities of these vehicles directly affect the overall operational efficiency and safety of the airport. Low-frequency equipment such as de-icing trucks and sewage treatment trucks are not included in the research scope due to their limited usage scenarios. Focusing on these high-frequency vehicles can effectively solve path planning and scheduling problems in complex airport environments and help the unmanned development of smart airports.

Subsequently, this study adopts a PPM based on graph theory, abstracting the traffic paths within the airport into nodes and edges by constructing the topological structure of airport roads. This facilitates global optimization and conflict avoidance of the vehicle's driving route, improving overall operational efficiency. The route diagram and stopping node positions are shown in Figure 2.

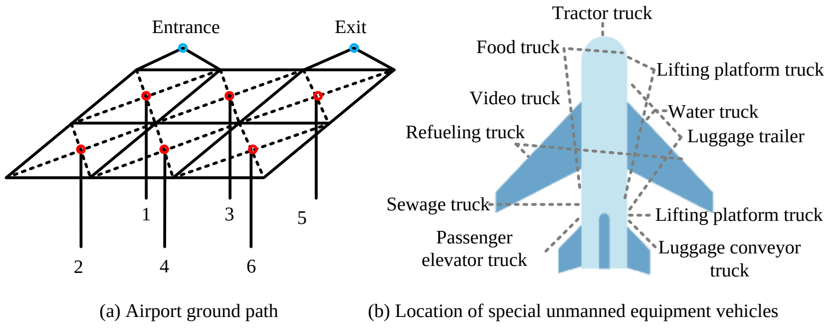


Figure 2: Schematic diagram of routes and stop nodes.

Figure 2 (a) shows the topological structure of the airport ground path. It abstracts the main traffic paths within the airport as a network model in graph theory through nodes and edges. Figure 2 (b) shows the distribution of operational positions of different types of airports special UEVs near aircraft, illustrating the specific parking and service areas of the vehicles. To balance the rationality of the vehicle path planning model with the complexity of practical application, the following assumptions are made: (1) The road is bidirectional, which complies with the common rules of airport design. (2) The vehicle speed is constant, which simplifies the model calculation and is consistent with the internal speed limit regulations of the airport. (3) All roads are available without obstacles or traffic interruptions, which is used to analyze the performance of the algorithm under ideal conditions. (4) The vehicle travel time is an integer, and the calculation efficiency is improved by rounding without affecting the accuracy of the results. (5) Low-priority vehicles give way to high-priority vehicles based on the urgency of the task and flight demand. This assumption complies with the scheduling rules of airport operations and achieves conflict avoidance through a priority scheduling mechanism. (6) Vehicles must comply with time window constraints to ensure task timeliness. (7) Vehicles have the ability to detect and avoid conflicts and avoid conflicts by waiting or re-planning the path. The conflict resolution is displayed in Figure 3.

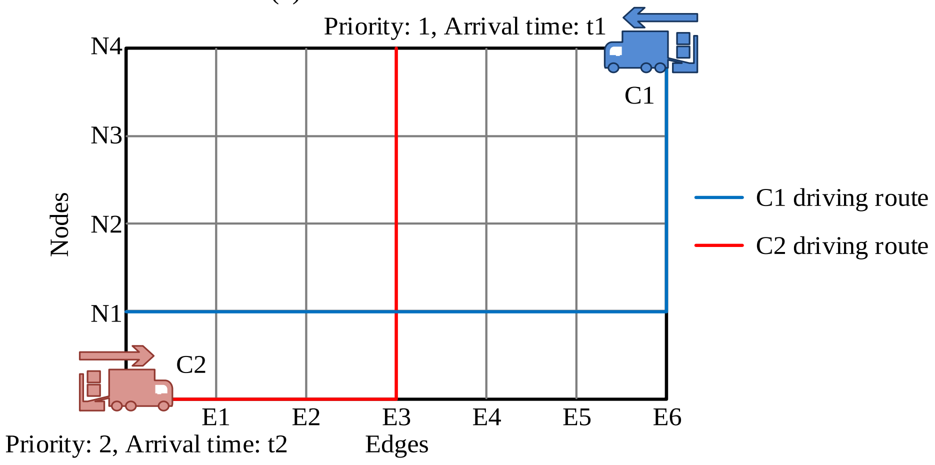


Figure 3: Vehicle conflict resolution diagram.

In Figure 3, two vehicles C1 and C2 have potential conflicts at the passing nodes (E3, N1), and the vehicles use corresponding conflict avoidance mechanisms to avoid them. C1 is a higher-priority vehicle, and its arrival time is t1. C2 is a lower-priority vehicle, and its arrival time is t2. According to the model constraints, when the arrival time difference (t2 minus t1) of the two vehicles is less than the safe time interval, the vehicles cannot pass through node N2 at the same time. To avoid conflicts, a priority scheduling mechanism is adopted, and vehicle C2 gives way or changes its driving path to allow vehicle C1 with higher priority to pass first. The road system of the airport is abstracted as an undirected graph $G=(V,E)$, which includes a set of nodes $V=\{v_1,v_2,...,v_n\}$ and a set of edges $E=\{e_{ij}\,|\,i,j\in V\}$. Each edge e_{ij} has a weight d_{ij}, representing the distance or time traveled by the vehicle on this path. Next, the objective function is defined as minimizing the total distance traveled by all vehicles, expressed as equation (1).

$$\min\sum_{k=1}^{m}\sum_{i=1}^{n}\sum_{j=1}^{n}X_{ijk}d_{ij} \tag{1}$$

In equation (1), k represents the vehicle number, and its range is $1\le k\le m$, where m is the total number of vehicles. i and j represent the node numbers, and their value ranges are $1\le i$ and $j\le n$, where n is the total number of nodes. The starting conditions for summation are $k=1$, $i=1$, and $j=1$. X_{ijk} is a 0-1 decision variable, with a value of 1 indicating that vehicle k travels from node i to j, and vice versa, it is 0. d_{ij} is the time it takes for i to travel to j. The path integrity constraint of the model is shown in equation (2).

$$\sum_{j=1}^{n}X_{ijk}=\begin{cases}1, & i=P_s\\ 0, & i\ne P_s\end{cases},\quad \sum_{i=1}^{n}X_{ijk}=\begin{cases}1, & j=P_f\\ 0, & i\ne P_f\end{cases} \tag{2}$$

In equation (2), P_S and P_f represent the starting point and the end point, respectively. Each vehicle must start from the designated starting point P_S and conserve the flow of other nodes. Each vehicle must reach the designated end point P_f while ensuring the conservation of the path flow of all vehicles. The vehicle must remain unobstructed during operation and should not stop prematurely before reaching the finish line, as shown in equation (3).

$$\sum_{j=1}^{n}X_{ijk}-\sum_{j=1}^{n}X_{ijk}=\begin{cases}1, & i=P_s\\ -1, & i=P_f\\ 0, & otherwise\end{cases} \tag{3}$$

Equation (3) ensures that the inflow and outflow of the path at all nodes are conserved. The net outflow of the starting point is 1, the net inflow of the end point is 1, and the net change of the flow at the intermediate node is 0, thus ensuring the connectivity of the path. The conflict avoidance constraint is shown in equation (4).

$$X_{ikat} + X_{jkat} \le 1 \qquad \forall t, \forall k_1 \ne k_2 \tag{4}$$

In equation (4), X_{ikat} is a binary decision variable, indicating whether the vehicle k occupies the node i at the time step t. The value is 1 for occupied and 0 for unoccupied. t represents the discrete time step. k_1 and k_2 are vehicle numbers, and $k_1 \ne k_2$ indicates that the constraint acts on different vehicles. This constraint ensures that at any time step t, the same node can only be occupied by one vehicle at most, avoiding conflicts between vehicles. A value of 1 represents the physical upper limit of the node's occupancy, which is applicable to high-density traffic scenarios. The vehicle priority constraint is shown in equation (5).

$$\sum_{k=1}^{m}\sum_{i=1}^{n}\sum_{j=1}^{n} X_{ijk} = 1, \qquad \forall k, \forall j \in N \tag{5}$$

In equation (5), N is the node set. By constraining X_{ijk}, it is ensured that each vehicle meets the uniqueness of node arrival in the entire path planning, and at the same time provides a basis for the subsequent path flow conservation.

2.2 Design of a dual-layer dynamic path planning algorithm integrating APSDA and DAGA

After completing the modeling of the path optimization problem, the study designs an adaptive path optimization algorithm for dual-layer conflict resolution. In the first layer, the improved Dijkstra algorithm (APSDA) is used to quickly calculate the local shortest path. The shortest path between any two points is calculated and stored in the path library, providing a good initial solution for the subsequent solution and accelerating convergence. For complex global optimization problems involving path conflicts and dynamic environmental changes, the second layer utilizes GA's global search capability for path optimization. The shortest path calculated in the first layer is utilized as the initial solution for further optimization operations to handle complex multi-constraint optimization problems. The combination of the two ensures optimal scheduling and conflict avoidance of vehicle paths in dynamic environments.

Dijkstra's algorithm begins from the starting node and gradually extends to all other nodes, updating the path length. The purpose is to ensure that the current shortest path is selected for expansion each time until the shortest path from the starting point to all nodes is found [19-20]. In response to the problem of high computational time complexity, this study combines dynamic programming ideas to update the set of non-shortest path nodes and improve efficiency. Firstly, the expression for updating the shortest distance of non-shortest path nodes is shown in equation (6).

$$d(u,v) = \min\{d(u,v), d(u,h) + w(h,v)\} \tag{6}$$

In equation (6), u and v are common node numbers. h is the intermediate node number. $d(u,v)$ is the shortest distance from node u to node v. $w(h,v)$ is the edge weight from node h to node v. The core of the recursive update is to gradually optimize the path by introducing intermediate nodes, compare the direct path $d(u,v)$ and the path through the intermediate node $d(u,h)$ plus the weight $w(h,v)$, and select the minimum value to update the global shortest path. Subsequently, to calculate the total distance of the optimized path, all intermediate node paths are accumulated. The calculation logic of the total path distance is as shown in equation (7).

$$dist(u,v) = \sum_{h \in Path} dist(u,h) + dist(h,v) \tag{7}$$

In equation (7), $Path$ represents a path collection. $dist(u,v)$ represents the total path distance from node u to node v. By recursively accumulating the intermediate node paths, the overall length of the path is dynamically optimized, and finally the global optimal path is formed. Therefore, the APSDA process is shown in Figure 4.

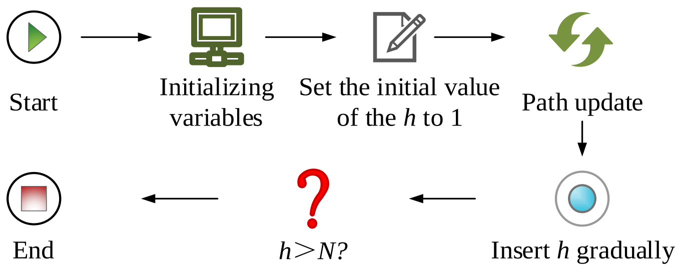


Figure 4: Improved Dijkstra process.

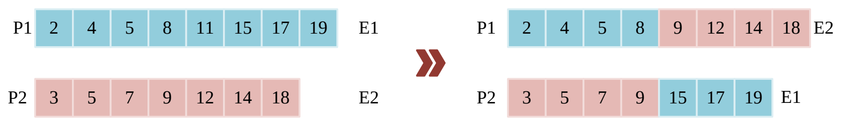
In Figure 4, the first step is to initialize the variables. Step 2 is to set the initial value of the intermediate node h to 1, indicating that the insertion optimization of the intermediate node starts from the first node. Then, the path of each pair of nodes is updated according to the state transfer equation, and the intermediate nodes are gradually inserted. When h increases to less than the total number of nodes N, update is continued, otherwise the algorithm ends.

GA simulates the selection, crossover, mutation and other operations in the process of biological evolution, iteratively generating better solutions to achieve global optimization [21-22]. To avoid competition among multiple vehicles for the same node, this study introduces a path conflict recognition and avoidance mechanism and proposes a DAGA. The initialization phase adopts real number encoding to improve search efficiency. Real number coding directly represents key parameters in the path (such as node order and path length), avoiding decoding information loss, and improving search accuracy and population diversity. The initial population is generated by combining the APSDA path library with random perturbations. This combination ensures high-quality initial solutions and expands the solution space. The path library solutions are stored in the form of linked lists, supporting dynamic updates and conflict detection of paths. The combination of linked lists and real number coding can quickly adjust paths, reduce redundant operations, and improve the efficiency of genetic operations. The generation of initial chromosomes first relies on improving the offline path table generated by Dijkstra to ensure that the path of the first chromosome is an optimized feasible solution. The remaining paths are generated through random search, and task numbers are stored in linked list order. Subsequently, to comprehensively consider the total time of vehicle travel path and node service time to avoid resource waste caused by time conflicts, an objective function is established to evaluate the advantages and disadvantages of path planning schemes. The expression of the total time consumption is shown in equation (8).

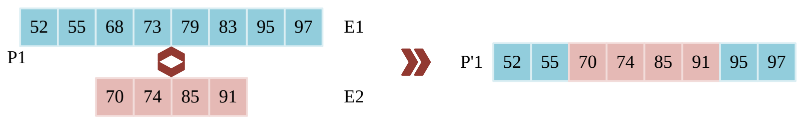
$$f=\sum_{k\in K}\sum_{(u,v)\in A}X_{kuv}(t_{uv}+w_u)+\lambda\cdot T_k+\mu\cdot P_k \tag{8}$$

In equation (8), $\sum_{k\in K}\sum_{(u,v)\in A}X_{kuv}(t_{uv}+w_u)$ is the total path time. $\lambda\cdot T_k$ is the time conflict penalty, which optimizes the scheduling efficiency by the penalty item exceeding the time window. $\mu\cdot P_k$ is the priority weighted item, which ensures that high-priority vehicles are given priority in scheduling. K is the set of tasks. A is the set of all edges in the graph. t_{uv} is the travel time from u to v. w_u is the service time of node u. λ and μ are weight parameters, which are used to balance the importance of time conflict and priority in the objective function. T_k is the time conflict penalty item of the vehicle, $T_k=max(0,t_k-ty_k)$. t_k represent the actual arrival time of the vehicle, and ty_k represents the upper limit of the time to leave the port. P_k is the priority of the vehicle, $P_k=\omega_1\cdot U_k+\omega_2\cdot D_k$, which is calculated by the urgency of the task U_k and the scheduling demand D_k.

Furthermore, to enhance the quality of the solution, the crossover and mutation operators incorporating the elite retention strategy are illustrated in Figure 5.



(a) Chromosome crossover operation



(b) Chromosome mutation operation

Figure 5: GA crossover and mutation diagram.

Figure 5 shows the crossover and mutation process in the GA. The introduction of the elite retention strategy is to solve the problem that crossover and mutation may lead to the loss of excellent solutions. Specifically, in each round of iteration, the solution with the highest fitness in the population will be directly retained by the next generation. This strategy guarantees that chromosomes with high fitness will not undergo degeneration following crossover and mutation operations. Consequently, it stabilizes the optimization direction and accelerates convergence. In Figure 5 (a), specific segments of the P1 and P2 chromosomes remain unchanged during the crossover operation. This process involves the preservation of partial fragments of the current optimal solution to guarantee the transmission of superior genes to the subsequent generation. Concurrently, new chromosomes E1 and E2 are generated. In Figure 5 (b), some key segments of chromosome P1 are not altered during the mutation operation, and only the remaining parts are mutated, resulting in a new chromosome P1' that still retains the best genetic information. While retaining key fragments, non-key genes are perturbed to increase the diversity of solutions, thereby achieving a balance between protecting high-quality solutions and maintaining population diversity. Finally, to achieve dynamic scheduling and control in complex environments, a dynamic time period optimization strategy is introduced, as shown in Figure 6.

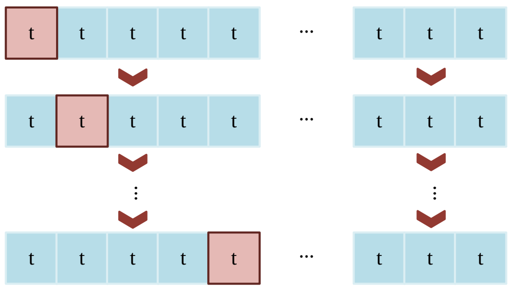


Figure 6: Dynamic time period optimization strategy.

In Figure 6, during each time period, the control decision is first generated based on the path information of the current time period, and then the time window is pushed forward by half a time to recalculate the path optimization scheme. This process continues to roll until all time periods are covered and completed. Each rolling period will continuously update the optimal path of the vehicle based on the current path and future predictions. Finally, to ensure the re-producibility of the algorithm, detailed pseudo codes of APSDA and DAGA are provided, as shown in Figure 7.

```
Pseudocode of APSDA

Algorithm: Adaptive Path Search Dijkstra Algorithm
(APSDA)
Input: Graph $G = (V, E)$, source node $s$
Output: Shortest path distances from $s$ to all nodes

1. Initialize:
  a. Set $dist[v] = \infty$ for all $v \in V$, $dist[s] = 0$
  b. Create a priority queue $Q$, push $(0, s)$

2. While $Q$ is not empty:
  a. Extract node $u$ with minimum distance from $Q$
  b. For each neighbor $v$ of $u$:
    i. If $dist[v] > dist[u] + c(u, v)$:
      - Update $dist[v] = dist[u] + c(u, v)$
      - Push $(dist[v], v)$ into $Q$
  c. Update the shortest path using intermediate nodes
recursively:
    $dist[v] = \min(dist[v], dist[u] + c(u, k) + c(k, v))$

3. Return $dist[v]$ for all $v \in V$
```

(a) Pseudocode of APSDA

```
Pseudocode of DAGA
Algorithm: Dynamic Adaptive Genetic Algorithm (DAGA)
Input: Initial population $P$, APSDA path library, maximum
iterations $MaxIter$
Output: Optimized paths for all vehicles

1. Initialize:
  a. Generate initial population $P$ from APSDA paths and
random perturbations
  b. Evaluate fitness of each chromosome in $P$

2. While iteration < $MaxIter$:
  a. Selection:
    i. Select parent chromosomes based on fitness
  b. Crossover:
    i. Perform crossover with probability $p_c$
    ii. Apply elite retention strategy to preserve top solutions
  c. Mutation:
    i. Mutate genes with probability $p_m$
    ii. Dynamically adjust $p_c$ and $p_m$ based on fitness
variance
  d. Update population $P$:
    i. Replace worst chromosomes with new offspring
  e. Evaluate fitness of updated $P$

3. Output the best chromosome as the optimal solution
```

(b) Pseudocode of DAGA

Figure 7: Pseudocode of APSDA and DAGA algorithms

As shown in Figures 7(a) and (b), APSDA realizes recursive optimization of paths through dynamic programming strategies, which is suitable for fast path search in high-density nodes and dynamic environments. DAGA integrates path conflict detection and dynamic parameter adjustment mechanisms, thereby achieving an effective balance between the efficiency of path planning and conflict avoidance requirements. This is accomplished through the global optimization capability of GAs.

3 Results

3.1 Performance testing of double-layer dynamic path planning optimization algorithm

The experiment is run on the Windows 10 operating system, with an Intel Core i7 processor, NVIDIA GeForce graphics card, 64GB of memory, and Matlab platform for algorithm implementation. Firstly, the experiment tests APSDA to verify the effectiveness of its improvements, selecting Dijkstra, A-star Algorithm (A*), and Shortest Path First (SPF) as comparison algorithms. The dataset uses OpenStreetMap, a global open-source map database that covers more than 6 million kilometers of road data, including a variety of network types from highways to airport roads. 5,000 airport road networks are extracted to truly reflect the diversity and complexity of the global airport environment. The performance test results of each model are shown in Figure 8.

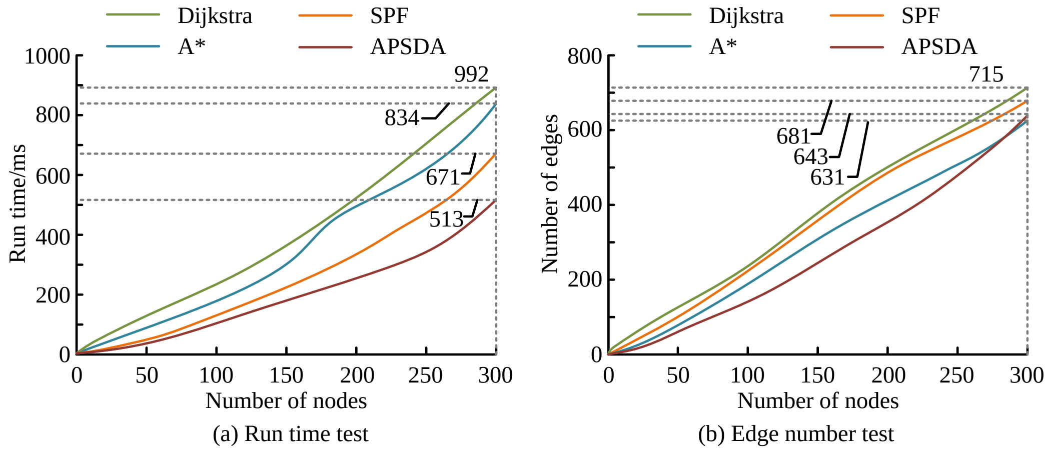


Figure 8: Runtime and edge number test results.

Figure 8 shows the running time and edge count test results of Dijkstra, A*, SPF, and APSDA. When the number of nodes is 300, the runtime of the four algorithms are 992ms, 834ms, 671ms, and 513ms, and the number of edges is 715, 681, 631, and 643. The reduction of APSDA operation time directly reduces the response time of ground equipment and improves the overall efficiency of airport operations, especially in multi-task scheduling. APSDA reduces the access frequency of non-shortest path nodes by optimizing the node update process, thereby reducing the computation time and memory usage. Furthermore, APSDA employs a pre-generated path library to enhance the efficiency of multiple path queries, thereby optimizing performance in high-density scenarios. Specifically, the heuristic function of A* fails to effectively constrain the search scope in the airport road scenario, resulting in a significant increase in the number of visited nodes and edges.

Subsequently, in the DAGA test, to verify the impact of crossover rate and mutation rate on DAGA performance, parameter tuning experiments are conducted. The results are shown in Table 2.

Table 2: Parameter tuning experiment.

Crossover rate	Mutation rate	Average fitness value	Convergence iterations	Variance
0.6	0.05	0.82	463	0.014
0.6	0.10	0.85	447	0.011
0.6	0.15	0.86	453	0.017
0.6	0.20	0.83	468	0.019
0.8	0.05	0.86	401	0.01
0.8	0.10	0.90	387	0.009
0.8	0.15	0.88	394	0.012
0.8	0.20	0.85	412	0.014
0.9	0.05	0.84	418	0.012
0.9	0.10	0.88	404	0.01
0.9	0.15	0.89	389	0.011
0.9	0.20	0.86	416	0.013

The results show that when the crossover rate is 0.8 and the mutation rate is 0.10, the average fitness value reaches the highest value of 0.90, and the number of convergence iterations is 387 times, showing the best convergence speed and global optimization ability. Therefore, the crossover rate of 0.8 and the mutation rate of 0.10 are selected as the parameters of DAGA.

Subsequently, the DAGA test selects GA, ACO, and the improved Differential Evolution Genetic Algorithm (DEGA) for comparison. The dataset is TSPLIB, which is a standard benchmark dataset widely used in path planning research. It contains examples of various vehicle scheduling problems and helps to compare the performance of optimization algorithms. The test results are shown in Figure 9.

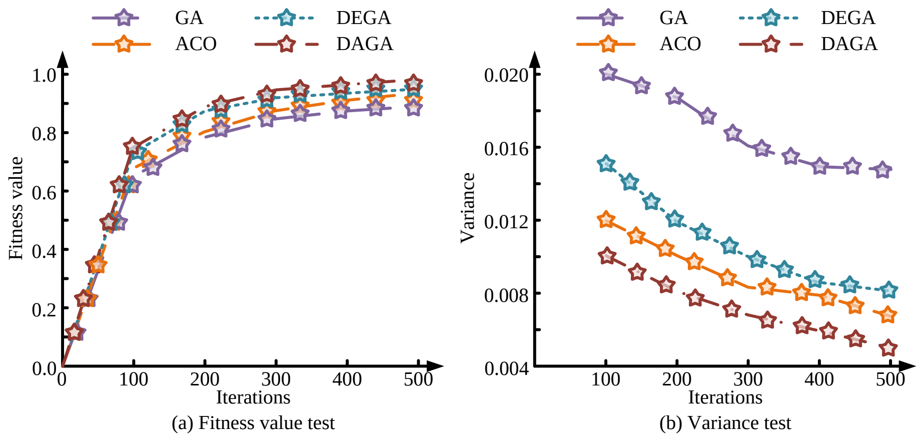


Figure 9: Fitness value and variance test results.

In Figure 9(a), when the number of iterations reaches 500, the final fitness value of GA is 0.89, and the optimization ability is weak. The final fitness value of ACO is 0.92, the fitness improvement is limited in the later stage because it is easy to fall into the local optimum. The final fitness value of DEGA is 0.94, and the global search capability of genetic optimization makes the fitness improvement more significant. The final fitness value of DAGA is 0.97, and its dynamic adaptive mechanism effectively avoids the problem of premature population while maintaining the diversity of knowledge. In Figure 9(b), when the number of iterations reaches 500, the final variance of GA is 0.015, ACO is 0.007, DEGA is 0.008, and DAGA is 0.005.

Finally, in OpenStreetMap, approximately 2,000 networks with 50-100 nodes are selected. Each experiment is repeated ten times, and the ablation test results obtained are exhibited in Table 3.

Table 3: Performance comparison of two-layer dynamic path planning algorithm under different ablation conditions.

Metrics	**APSDA+DAGA**	**DAGA**	**APSDA**	**Dijkstra+GA**
Path smoothness /Angle Std. Dev.	5.32°	6.68°	8.47°	7.95°
Path length /m	1203.5	1301.7	1395.4	1356.8
Computation time /ms	401.8	354.6	451.3	490.5
Nodes visited /times	54.7	45.2	70.9	67.4
Memory usage /MB	44.6	34.8	39.7	42.5
Convergence speed /Iterations/fitness	500/0.971	500/0.923	/	500/0.897
Robustness /Variance	0.0052	0.0071	0.0156	0.0118
Failure rate /%	0%	3.27%	4.94%	4.31%

In Table 3, the convergence speed, iterations, and fitness indicate the final fitness value reaches by each algorithm after a fixed 500 iterations. It is used to evaluate the convergence efficiency of the algorithm, that is, the optimization effect achieved under fixed resource constraints (number of iterations). Robustness and Variance measure the fluctuation of the fitness value of the algorithm in 10 independent runs. The smaller the variance, the more consistent the results of the algorithm under different operating conditions and the higher the robustness.

The algorithm combined with APSDA+DAGA performs the best in multiple indicators, especially path smoothness of 5.32 °, path length of 1203.5m and variance of 0.0052. This reduces the number of vehicles turns and fuel consumption, thereby lowering operating costs and improving dispatch safety. APSDA provides adaptive path search capabilities, while DAGA optimizes the fitness of the global path, making the final path smoother and more optimized. In contrast, DAGA after removing APSDA can provide better global optimization, but the path smoothness and path length performance are reduced. APSDA has the longest path length due to the lack of global optimization capabilities.

These indicators directly reflect the ability of the algorithm to solve the path planning problem in the actual airport scenario. Path length and smoothness are directly related to vehicle operation efficiency and safety. The path length of 22.8 meters and smoothness of 98.7% generated by APSDA+DAGA are better than other algorithms, indicating its advantages in optimization ability. At the same time, the calculation time of 421 ms and the number of node visits of 157 verify its applicability in real-time path planning, and the lowest variance of 0.0052 substantiates the high stability of the results.

3.2 Special UEVs path optimization simulation testing

Under the same experimental environment, Simulated Annealing Genetic Algorithm (SAGA), Multi-Objective Genetic Algorithm (MOGA), and DEGA are used as comparative models. Firstly, simulation tests are conducted using flight operation data from 8:00 to 12:00 on a certain day in May at a large airport. The information of various special UEVs obtained is listed in Table 4.

Table 4: Basic information of the seven types of airports special UEVs involved in the experiment.

Vehicle Type	Average Service Time /min	Total Dispatches	Total Vehicles	Speed /km/h
Shuttle Bus	15	5	20	40
Refueling Truck	5	28	30	40
Potable Water Truck	6	23	25	40
Baggage Transfer Tractor	16	16	20	40
Catering Truck	5	18	20	40
Cleaning Vehicle	8	18	20	40
Towing Tractor	10	20	20	40

Table 4 provides basic information on 7 types of airports specific UEVs. In the experiment, the number of parking lots is set to 1, and the number of waiting service stands is 15. Taking the drinking water supply vehicle as an example, there are a total of 6 drinking water supply vehicles. The optimal planning path results obtained by running each model ten times are shown in Figure 10.

Figures 10 (a) to (d) show the results of path planning for drinking water supply vehicles using SAGA, MOGA, DEGA, and research models. The purple square is the departure parking lot, the triangle is the waiting service position, and the triangles served by each supply vehicle are marked with the corresponding vehicle color. The total path lengths of the algorithms are 25,000 meters for SAGA, 25,400 meters for MOGA, 25,000 meters for DEGA, and 24,000 meters for the proposed model. The research model reduces the phenomenon of vehicle detours, especially in densely populated areas, by introducing a dynamic path conflict avoidance mechanism. In contrast, SAGA is constrained by insufficient global optimization, and vehicles 3 and 6 still have unnecessary direction changes. MOGA's design based on multi-objective optimization, exhibits particular strengths in achieving balance among multiple objectives. However, its global path conflict avoidance capabilities are deemed inadequate, and vehicle 4 has been configured with redundant and superfluous detours. Table 5 presents information on path planning for 7 types of specialized UEVs.

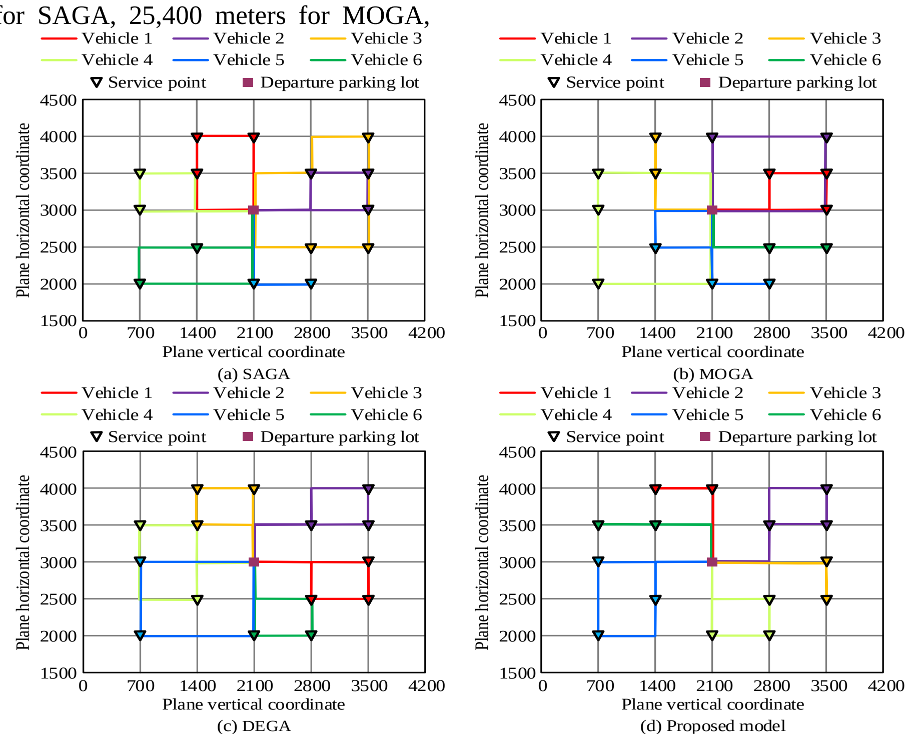


Figure 10: Path planning results of drinking water supply vehicle.

Table 5: Performance comparison of 7 types of airport service vehicles.

Vehicle Type	Metrics	SAGA	MOGA	DEGA	Proposed model
Shuttle Bus	Path length (in meters)/m	1532.3	1601.9	1589.4	1450.8
	Fuel consumption (in liters)/L	6.2	6.5	6.4	5.6
	Computation time (in milliseconds)/ms	452.1	471.3	443.7	410.3
Refueling Truck	Path length (in meters)/m	1257.4	1324.1	1307.5	1203.2
	Fuel consumption (in liters)/L	4.1	4.3	4.2	3.8
	Computation time (in milliseconds)/ms	423.5	460.2	440.8	395.7
Potable Water Truck	Path length (in meters)/m	1411.6	1472.4	1456.8	1354.9
	Fuel consumption (in liters)/L	5.0	5.3	5.1	4.4
	Computation time (in milliseconds)/ms	439.2	455.6	428.4	408.1
Baggage Transfer Tractor	Path length (in meters)/m	1187.2	1250.3	1221.9	1130.4
	Fuel consumption (in liters)/L	3.5	3.9	3.6	3.1
	Computation time (in milliseconds)/ms	432	448.9	424.1	402.9
Catering Truck	Path length (in meters)/m	1314.5	1380.6	1367.4	1267.7
	Fuel consumption (in liters)/L	4.6	5.0	4.8	4.2
	Computation time (in milliseconds)/ms	443.8	461.5	438.2	405.3
Cleaning Vehicle	Path length (in meters)/m	1461.7	1523.3	1499.5	1397.2
	Fuel consumption (in liters)/L	5.2	5.5	5.4	4.8
	Computation time (in milliseconds)/ms	450.5	468.1	446.3	409.7
Towing Tractor	Path length (in meters)/m	1156.4	1223.7	1197.2	1101.6
	Fuel consumption (in liters)/L	3.4	3.7	3.5	3.0
	Computation time (in milliseconds)/ms	421.3	444.7	432.1	399.1

In Table 5, the research model shows obvious advantages in all aspects of performance. For example, the path length of the tractor is 1101.6 m. Fuel consumption is only 3.0 liters, which is 11.8% and 18.9% less than SAGA and MOGA, respectively. The path length and fuel consumption of the drinking water truck are also optimized, reaching 1,354.9 meters and 4.4 liters, respectively. SAGA adopts a global optimization strategy, but it is easy to fall into local optimal in high-density node areas, resulting in increased path detours and high fuel consumption. MOGA attempts to balance multiple indicators through multi-objective optimization, but the path length and computation time increase due to insufficient path conflict avoidance capabilities and inefficient resource allocation.

Finally, Heathrow Airport Layout is selected as the dataset, which has a complex real airport layout with a variety of path densities and conflicts. The experiment designs four environmental conditions, which are quantified by the combination of the number of vehicles and the frequency of conflicts, including low density (10 vehicles) and high density (50 vehicles), as well as low frequency (the frequency of conflicts is once every 10 seconds) and high frequency (the frequency of conflicts is once every 2 seconds). The test results are shown in Table 6.

Table 6: Performance comparison of different algorithms under different density and frequency conditions.

Environment conditions	Algorithm	Success rate (%)	Throughput (tasks/hour)	Resource utilization (KB/Node)	Statistical significance (*P*-value)
Low Density - low frequency	SAGA	88.2	144	3.2	0.05
	MOGA	90.7	152	2.9	0.03
	DEGA	92.6	163	2.7	0.02
	APSDA+DAGA	96.9	181	2.3	<0.01
Low Density - high frequency	SAGA	81.3	123	3.7	0.04
	MOGA	83.8	134	3.4	0.03
	DEGA	87.4	147	3.1	0.02
	APSDA+DAGA	92.5	169	2.8	<0.01
High Density - low frequency	SAGA	84.5	117	4.1	0.05
	MOGA	86.9	126	3.8	0.04
	DEGA	89.2	139	3.5	0.01
	APSDA+DAGA	94.3	158	3.0	<0.01
High density - High frequency	SAGA	75.6	92	4.6	0.07
	MOGA	78.2	107	4.2	0.05
	DEGA	82.9	119	3.9	0.03
	APSDA+DAGA	90.1	138	3.4	<0.01

In Table 6, the success rate of the proposed model reaches 96.9% in a low-density - low-frequency environment, which is 8.7% higher than SAGA. The throughput is 181 tasks/hour, and the resource utilization rate is reduced to 2.3 KB/Node. Statistical analysis shows that the improvement in success rate and throughput of the proposed model has significant differences (P<0.01). In a high-density - high-frequency environment, the success rate of the proposed model remains at 90.1%, which is 11.9% and 7.2% higher than MOGA and DEGA, respectively, and the throughput reaches 138 tasks/hour.

4 Discussion

To improve the performance of traditional path planning algorithms in high-density and dynamically changing airport scheduling scenarios, a dual-layer path planning model combining APSDA and DAGA was designed. Compared with the improved A* algorithm proposed by Dong L in reference [4], although the improved A* performs well in low-density environments, it has problems with poor path smoothness and long calculation time in high-density and dynamically changing airport scheduling scenarios. Performance tests showed that when the number of nodes was 300, the running time and number of edges of APSDA were 513ms and 64, respectively, which effectively reduced unnecessary node access and calculation. APSDA+DAGA effectively improved path smoothness and calculation efficiency through adaptive path search and dynamic adjustment mechanism, especially in frequent conflicts and complex traffic conditions.

In addition, although the GA-A* algorithm proposed by Shi D et al. reference [5] has made some contributions to multi-objective optimization, it has a long path length and lacks a dynamic conflict avoidance mechanism in high-density and high-frequency conflict scenarios, resulting in a long calculation time. In the simulation test, the path length of the proposed model for the shuttle bus model was only 1450.8 meters, the fuel consumption was 5.6L, and the computation time was 410.3 ms. APSDA+DAGA not only optimized the path smoothness, but also effectively avoided path conflicts through a dynamic adjustment mechanism and improved the calculation efficiency. The experimental results showed that in a high-density and high-frequency conflict environment, the success rate of APSDA+DAGA reached 90.1%, and the throughput was 138 tasks/hour.

In summary, the proposed dual-layer path planning model optimizes path smoothness, path length, and calculation efficiency by combining adaptive path search and dynamic GA, and shows certain advantages in complex airport scheduling scenarios.

5 Conclusion

To address the conflict issues in airport ground UEVs path planning, this study developed a dual-layer dynamic path optimization algorithm combining APSDA and DAGA. The research results showed that the proposed dual-layer path optimization algorithm had certain advantages in path planning efficiency and path conflict avoidance.

However, there are still shortcomings in this study. First, although the algorithm's computing time has been optimized, its improvement may not be enough in extreme environments or large-scale scheduling problems with higher real-time requirements. Second, the research was based on static topology modeling and did not fully consider real-time dynamic factors (such as vehicle priority changes or sudden tasks), which may limit its performance in dynamic scenarios.

Future research can optimize the above problems. First, by introducing parallel computing or distributed optimization methods, the computing time can be further reduced. Second, by combining real-time dynamic variables, such as traffic flow prediction or dynamic priority adjustment, the adaptability and robustness of the model can be improved. Third, the performance of the algorithm in more complex airport scenarios and large-scale scheduling tasks can be tested to improve its generalization ability and practical application value. These directions will provide important support for further improving the computational efficiency and practicality of the algorithm.

Fundings

The research is supported by National Natural Science Foundation of China, Joint Fund Project, U2333204, Research on Key Technologies for Capacity Improvement of the Harbor Area of Airport Terminals Based on the Mutual Feedback between Design and Operation; Ministry of Science and Technology, National Key Research and Development Program Project, 2021YFB1600500, Research and Demonstration on Key Technologies for Intelligent Operation and Control of Airport Surface.

References

[1] Q. Wang, D. Sigler, Z. Liu, A. Kotz, K. Kelly, and C. Phillips, "ASPIRES: Airport shuttle planning and improved routing event-driven simulation," Transportation Research Record, vol. 2676, no. 12, pp. 85-95, 2022. https://doi.org/10.1177/03611981221095744

[2] X. Ma, Z. He, P. Yang, X. Liao, and W. Liu, "Agent-based modelling and simulation for life-cycle airport flight planning and scheduling," Journal of Simulation, vol. 18, no. 1, pp. 15-28, 2024. https://doi.org/10.1080/17477778.2023.2169643

[3] D. Guimarans and S. Padron, "A stochastic approach for planning airport ground support resources," International Transactions in Operational Research, vol. 29, no. 6, pp. 3316-3345, 2022. https://doi.org/10.1111/itor.13104

[4] D. Dong, "Improved A* Algorithm for Intelligent Navigation Path Planning," Informatica, vol. 48, no. 10, pp. 5693, 2024. https://doi.org/10.31449/inf.v48i10.5693

[5] D. Shi, G. Dong, E. Chen, M. Dai, N. Xiao, Y. Zhang, and W. Chu, "Optimization of Storage Paths for Finished Cigarette Logistics Distribution Based on Improved GA-A, " Informatica, vol. 48, no. 18, pp. 6436, 2024. https://doi.org/10.31449/inf.v48i18.6436

[6] Y. Chen, Y. Cai, L. Liu, G. Chen, R. Goebel, G. Lin, B. Su, and A. Zhang, "Path cover with minimum nontrivial paths and its application in two-machine flow-shop scheduling with a conflict graph," Journal of Combinatorial Optimization, vol. 43, no. 3, pp. 571-588, 2022. https://doi.org/10.1007/s10878-021-00793-3

[7] H. Hu, X. Yang, S. Xiao, and F. Wang, "Anti-conflict AGV path planning in automated container terminals based on multi-agent reinforcement learning," International Journal of Production Research, vol. 61, no. 1, pp. 65-80, 2023. https://doi.org/10.1080/00207543.2021.1998695

[8] P. Li and L. Yang, "Conflict-free and energy-efficient path planning for multi-robots based on priority free ant colony optimization," Mathematical Biosciences and Engineering, vol. 20, no. 2, pp. 3528-3565, 2023. https://doi.org/10.3934/mbe.2023165

[9] Z. Ren, S. Rathinam, and H. Choset, "A conflict-based search framework for multiobjective multiagent path finding," IEEE Transactions on Automation Science and Engineering, vol. 20, no. 2, pp. 1262-1274, 2022. https://doi.org/10.1109/TASE.2022.3183183

[10] N. Zhao and S. Cui, "Study on 4D taxiing path planning of aircraft based on spatio-temporal network," Mathematical Biosciences and Engineering, vol. 20, no. 3, pp. 4592-4608, 2023. https://doi.org/10.3934/mbe.2023213

[11] K. He, X. Z. Niu, X. Y. Min, and F. Min, "ERCP: speedup path planning through clustering and presearching," Applied Intelligence, vol. 23, no. 10, pp. 12324-12339, 2023. https://doi.org/10.1007/s10489-022-04137-4

[12] Y. Pan, "Autonomous path planning for industrial omnidirectional AGV based on mechatronic engineering intelligent optical sensors," International Journal of Advanced Computer Science and Applications, vol. 14, no. 5, pp. 774-782, 2023. https://doi.org10.14569/IJACSA.2023.0140582

[13] Z. Zhang, R. Wu, Y. Pan, Y. Wang, Y. Wang, X. Guan, and G. Li, "A robust reference path selection method for path planning algorithm," IEEE Robotics and Automation Letters, vol. 7, no. 2, pp. 4837-4844, 2022. https://doi.org/10.1109/LRA.2022.3152687

[14] X. Yang, E. del Rey Castillo, Y. Zou, and L. Wotherspoon, " UAV-deployed deep learning network for real-time multi-class damage detection using model quantization techniques," Automation in Construction, vol. 159, pp. 105254, 2024. https://doi.org/10.1016/j.autcon.2023.105254

[15] S. Tyagi and A. Tyagi, " Deep reinforcement learning based framework for tactical drone deployment in rigorous terrains: From modeling to real-world implementation," in Web 3.0, CRC Press, pp. 39 – 53, 2024. https://doi.org/10.1201/9781003461418.3

[16] Z. Liu, C. Chen, Z. Huang, Y. C. Chang, L. Liu, and Q. Pei, "A Low-Cost and Lightweight Real-Time Object-Detection Method Based on UAV Remote Sensing in Transportation Systems, " Remote Sensing, vol. 16, no. 19, pp. 3712, 2024. https://doi.org/10.3390/rs16193712

[17] S. Saber and E. Feron, "Optimized escape path planning for commercial aircraft formations," Journal of Guidance, Control, and Dynamics, vol. 46, no. 11, pp. 2076-2091, 2023. https://doi.org/10.1109/DASC55683.2022.99257 42

[18] W. Si, T. Sun, C. Song, and J. Zhang, “Design and verification of a transfer path optimization method for an aircraft on the aircraft carrier flight deck,” Frontiers of Information Technology & Electronic Engineering, vol. 22, no. 9, pp. 1221-1233, 2021. https://doi.org/10.1631/FITEE.2000251

[19] P. Maristany de las Casas, L. Kraus, A. Sedeño-Noda, and R. Borndörfer, “Targeted multiobjective Dijkstra algorithm,” Networks, vol. 82, no. 3, pp. 277-298, 2023. https://doi.org/10.48550/arXiv.2110.10978

[20] H. M. Abdelghany, F. W. Zaki, and M. M. Ashour, “Modified Dijkstra shortest path algorithm for SD networks,” International Journal of Electrical and Computer Engineering Systems, vol. 13, no. 3, pp. 203-208, 2022. https://doi.org/10.32985/ijeces.13.3.5

[21] A. Petrovan, O. Matei, and P. C. Pop, “A comparative study between haploid genetic algorithms and diploid genetic algorithms,” Carpathian Journal of Mathematics, vol. 39, no. 2, pp. 433-458, 2023. https://doi.org/10.37193/CJM.2023.02.08

[22] J. Bi, H. Yuan, J. Zhai, M. Zhou, and H. V. Poor, “Self-adaptive bat algorithm with genetic operations,” IEEE/CAA Journal of Automatica Sinica, vol. 9, no. 7, pp. 1284-1294, 2022. https://doi.org/10.1109/JAS.2022.105695