Hindawi
International Journal of Aerospace Engineering
Volume 2023, Article ID 6643903, 16 pages
https://doi.org/10.1155/2023/6643903

Research Article

An Efficient Aircraft Conflict Detection and Resolution Method Based on an Improved Reinforcement Learning Framework

Qiucheng Xu,[1,2] Zhangqi Chen,[1,3] Fangfang Li,[1] Zhiyuan Shen,[1] and Wenbin Wei[4]

[1]*College of Civil Aviation, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China*
[2]*State Key Laboratory of Air Traffic Management System and Technology, Nanjing 210007, China*
[3]*Jiangsu Sub-bureau of East China Regional Air Traffic Management Bureau, Nanjing 211100, China*
[4]*Department of Aviation and Technology, College of Engineering, San Jose State University, San Jose, CA 95192-0061, USA*

Correspondence should be addressed to Zhiyuan Shen; shenzy@nuaa.edu.cn

Received 29 November 2022; Revised 1 August 2023; Accepted 21 August 2023; Published 16 September 2023

Academic Editor: Jinchao Chen

With the steady increase of air traffic column, an auxiliary decision tool is required to compensate the operation redundancy deficiency of more sectors of air traffic control. To solve the problem of nonconflict high-density departure and arrival traffic flow, this method is expected to rapidly establish and maintain safe separation with more flexible changing strategies for aircraft heading and speed. This paper proposes an improved reinforcement learning framework to achieve conflict detection and resolution. The proposed framework includes the first development of an air traffic flow model based on a multiagent Markov decision process. The goal reward function was then maximized by improved Monte-Carlo tree search combined with an upper confidence bound tree. Three simulation scenarios were designed for illustrating the improvements of the proposed algorithm, with the results indicating that the algorithm could establish and maintain safe separation between 20 agents in the simplified hexagon-shaped airspace of Huadong, China. Furthermore, the proposed method was demonstrated to reduce the number of conflicts between aircraft agents by up to 26.32% compared to previous research.

1. Introduction

The total flight hours of the air transportation industry increased by 6.7% during 2019 [1], and hence, there were also substantial increases in air traffic controller (ATC) workload and the complexity of the air traffic sector [2]. To counteract the constraints of maximum workloads of ATCs, one single sector requires to be subdivided. Although this action can decrease the ATC workload of that sector, it can also increase coordination time between different sectors [3]. Furthermore, to increase the safety redundancy of initially setting new sectors, air traffic management (ATM) bureaus are required to assign more simulator training to ATCs [4].

As the complexity of air traffic increases, two issues could decrease ATC operation efficiency. The first one is old equipment. The daily mission of ATCs relies on secondary surveillance radar systems and very high-frequency communication to issue ATC clearances. These two types of equipment and onboard traffic collision avoidance systems (TCAS) usually have long updating cycles. The second issue is airspace complexity. The controllers distribute multiple aircrafts on the same flight level while guaranteeing a safe horizontal separation. This kind of status is fragile when emergency situations such as radar failure are encountered. In that situation, the control mode needs to switch from radar control to procedural control, but the procedural control separation is greater than radar control [5]. Furthermore, the safety risk of that sector will increase tremendously since loss of separation (LOS) could occur. If a LOS event is unavoidable, the onboard TCAS will work to help pilots execute decision-making in the vertical direction to reestablish safe separation [6]. Considering these two limits for the terminal airspace in high-density traffic flow situations, an advanced auxiliary tool to help pilots make decisions without receiving instructions from the ATC is required to maintain safe airborne separation.

Researchers have focused on two aspects of solving the above problem. One category of methods could decrease the LOS probability caused by human factors. For instance, an ATC simulator was used to improve the familiarity of controllers with hot-point distribution in new sectors. But the simulator is hardly ergodic regarding the influences of human factors. The other one is aimed at increasing the safety redundancy of the controller-centered control method by using extra tools [7]. This could help the aircraft learn to maintain safe separation in emergency situations. For identifying an auxiliary tool to compensate for controller insufficiencies when dealing with complex traffic conflicts, many researchers have proposed several nonconflict methods [8–12]. In 2003, Bayen et al. proposed a Lagrangian delay predictive model for sector-based air traffic flow management [13]. However, that model relied on the flight plan of an aircraft to predict the conflicts, and it cannot cope with the flexibility terminal scenario that induces more conflicts and uncertainty. To improve the computing speed and accuracy of the CD&R model to handle more complex situations, researchers attempted to set and train the agents to recognize conflicts. In 2010, Agogino and Turner proposed a multiagent approach to simplify traffic patterns and complexity by classifying traffic flow based on flight plans [14]. In 2019, Wang et al. trained one agent to guide the aircraft to land at certain runways under FF conditions [15]. That agent only performed activities that considered the landing airport and not the landing sequence. In 2019, Pham et al. applied the reinforcement learning (RL) method to centralize agent training in CD&R with the limitation of only using two aircrafts [16, 17]; this pairwise conflict method between aircrafts is much simpler than the global approach method in one ATC sector. However, this method is not able to perfectly manage the increasing air traffic flow of more sectors with just one or two agents, and so, a multiagent system becomes a valuable solution to complex airspace, especially in the terminal airspace. To reduce ATC workload, based on multiagent Markov desicion process (MMDP) [18] method, Wei et al. applied this method to cooperatively train agents in landing aircraft [19]. But with the fixed CD&R strategy, it is possible that the goal of one agent might violate that of another. Information sharing between agents becomes a necessary precondition to automate CD&R in terminal airspaces when centralized control is impossible or impractical [9].

In the state of art of automatic CD&R, most of the researchers focus on the cruising phase of the aircraft [20, 21], ground taxing [22, 23], and conflicts between taking off aircraft and landing aircraft on the intersecting runway [24, 25] and CD&R between unmanned aerial vehicles (UAVs) [26, 27], but the flight speed and maneuvering mode of drones are very different from civil aircraft. There are also CD&R algorithms such as OCRA [28, 29] that could be applied to MMDP problems with an extremely large magnitude of agents, the speed of the agent can be reduced to zero in the OCRA model, and each agent is only responsible for the half of the minimum safety interval, which is not applicable to CD&R in civil aviation. However, there is few research on automatic CD&R in the terminal airspace in which the aircraft altitude changes greatly, the heading changes rapidly, and the speed decreases rapidly from cruising speed to 0.

To solve the problem of the self-maintenance of safe aircraft separation at high traffic flows, this paper presents an improved RL algorithm combined with Monte-Carlo tree search (MCTS) [30]. Furthermore, to conform with national airspace system rules, upper confidence bound trees (UCTs) [31–33] are introduced and applied to complete node selection and information-sharing methods between agents.

In this paper, we present an innovative algorithm aimed at formulating the collision avoidance problem between multiple vehicles, specifically focusing on the CD&R process of civil aircraft during take-off and landing. Our research redefines the collision detection mode of aircraft, ensuring the maintenance of safe separation during continuous high-speed movement.

To address the limitations of existing strategies, our algorithm introduces dynamic search depth in the iterative process of the MCTS algorithm. This adaptability allows for efficient establishment of safe separation between departure and arrival traffic flows, even in the presence of radar failures. It also serves as a supplementary decision tool, augmenting the TCAS by providing horizontal conflict resolution strategies [34].

Furthermore, our algorithm expands the state by incorporating additional operational actions, such as vector change clearance and speed change clearance, which are more applicable to daily air traffic control scenarios. We optimize information sharing among multiple agents by employing a same-priority assumption and reduce computational consumption by adjusting the search depth.

The main contributions of this work are listed as follows:

(1) The algorithm proposed in this paper was an improved RL algorithm that could achieve a multiagent Markov decision process to avoid LOS events between aircrafts

(2) The algorithm proposed in this paper is the addition of more operational actions in the state and the optimization of information-sharing methods to reduce computational consumption

(3) The proposed algorithm could be fused to ATC automation equipment to help ATC trainees to establish rapid safe separation of all departing/landing aircraft in terminal airspace

This paper is structured as follows: Section 2 establishes the model of air traffic flow in terminal airspace and objective functions, Section 3 constructs the solution algorithm, Section 4 presents a case study consisting of seven airports in the Huadong region of China, and conclusions are drawn in Section 5. The abbreviations used in this article and their corresponding meanings are shown in Table 1.

2. Air Traffic Flow Model with Multiagent

MDP has been well studied and applied in a wide range of disciplines. The agent can choose any available action a

Table 1: Acronyms.

Acronyms	
ATC	Air traffic controller
ATM	Air traffic management
CD&R	Conflict detection and resolution
FPL	Flight plan route
LOS	Loss of separation
MCTS	Monte-Carlo tree search
UCB	Upper confidence bounds
UCT	Upper confidence bounds applied to trees
MDP	Markov decision process
MMDP	Multiagent Markov decision process
UAVs	Unmanned aerial vehicles
RL	Reinforcement learning
TCAS	Traffic collision avoidance systems

based on the current state at each time step. The process responds at the next time step, enters a new state with a certain transition probability, and gives the agent a corresponding reward r. More precisely, a MMDP process based on MDP consists of the following components:

2.1. State. The state matrix of aircraft agents is described as a vector:

$$S_i = \left(x, y, v, \psi, \boldsymbol{g}_x, \boldsymbol{g}_y\right), \tag{1}$$

where x, y are the plane coordinates of the aircraft agent, v is the airspeed, ψ is the true heading, and $\boldsymbol{g}_x, \boldsymbol{g}_y$ are the coordinates of the goal airport position. If there are n agents in the terminal airspace, the state matrix of the MMDP will be an $n \times 6$ matrix.

When a radar failure emergency occurs, the initial stage of radar failure situation is to reduce the whole probability of less than safe separation between aircrafts. At this time, for the traditional radar control terminal area, the controller is generally adapted to the aircraft flight procedures (vector or speed) and separate the aircraft at the same altitude. Since the proposed algorithm is aimed at providing a horizontal conflict resolution to remedy TCAS deficiency in radar failure situations, the vertical dimension is not considered in whole model.

2.2. Action Space. The action space is the collection of all possible actions that the agent can perform and is the way the agent moves from one state to the next state in the state [34]. At each time step, the aircraft agent will choose and execute one movement at the current state which is determined by the last decision-making process for one time step before the next decision-making process. The action space for heading change strategy is

$$\{-n\theta, \cdots, -2\theta, -\theta, 0 \circ /\mathrm{s}, \theta, 2\theta, \cdots, \mathrm{n}\theta\}, \tag{2}$$

where $n \epsilon N^+$ and θ is the standard turning angle and $^\circ$/s is the angular velocity units for turning.

The angle change range of the aircraft is ideally varied from 0 to 360°, but the course change action of the aircraft decided by the controller is usually multiples of a minimum unit. The standard turning angle is represented by θ in the text.

The speed change strategy is automatically executed. As the distance of the aircraft agent from the destination decreases, the speed of the agent will change gradually from $v_{\max}$ to final approach speed $v_{\min}$ according to the distance between the agent and the goal position.

2.3. State Transition. The state transition for each agent is described by the following equation:

$$\begin{aligned} \dot{x} &= v \cos\psi, \\ \dot{y} &= v \sin\psi, \\ v &= \min\left\{v_{\max}, v_{\min} + \frac{d(o,g)}{\max d(o,g)}\left(v_{\max} - v_{\min}\right)\right\}, \\ \dot{\psi} &= a_\psi, \end{aligned} \tag{3}$$

where v is the current velocity of the agent, ψ is the true heading of the agent, a_ψ is the angle velocity of the agent, $v_{\max}$ is the maximum cruising speed and $v_{\min}$ is the minimum approach speed that the agent can attain, $d(o, g)$ is the distance between the current agent position and the goal position, and max $d(o, g)$ is the largest distance between the agent and its goal, which is usually set as the diagonal length of the simulation area. The transition process of the state of one agent involves choosing one heading change strategy in the action space, combined with the distance information between the agent and its goal position. And the algorithm can accomplish the state transition process with the environment state information as follows:

Using the environment state information of other agents $j\,(j = 1, 2, \cdots, N\, j \neq i)$, the next action a_i^* was obtained by

$$a_i^* = \underset{a_i}{\operatorname{argmax}} r_i^*\left(s, a_i, a_{-i}\right) i = 1, 2, \cdots, n, \tag{4}$$

where $r_i^*(s, a_i, a_{-i})$ is the reward value of agent A_i at the state $S_i = (x, y, v, \psi, \boldsymbol{g}_x, \boldsymbol{g}_y)$ and a_i is a will executing random action from $\{a_1^i, a_2^i, a_3^i, a_4^i, a_5^i,\}$, while action strategies of the other agents are represented by a_{-i}.

2.4. Termination State. Termination state is the terminal state in the state chain composed of a series of actions of the agent [35]. The termination state of one agent can be categorized as one of the following four results:

(1) The minimum distance value between the selected agent with other aircraft agents is less than $r^{\min}$, which is the minimum safe distance. When the distance between agents is less than the minimum safe distance, but greater than ε, the agent will not disappear, and it will only be recorded as a LOS event (which is the near miss state, but all agents will keep alive and continue to move forward in the scene)

(2) The agent has broken through the boundary of the terminal airspace, once the agent has broken through the boundary, it will be deleted and cannot continue to cumulate reward

(3) The agent has reached the goal airport, which is the goal state. Once the distance between the agent and the goal airport is less than ε, the agent will be judged as reach the goal state

(4) The distance between two agents is less than ε, and both two agents will be deleted and cannot continue to cumulate reward

The following steps can be used to judge whether the agent is at the goal state or not:

First, the minimum distance between target agent a_i and other agent must be judged as less than $r^{\min}$ or not. All agents are represented as $\{A_1, A_2, \cdots, A_n\}$, and the state of the other agents is $(x_j, y_j, v_j, \psi_j, g_{jx}, g_{jy}), j = 1, 2, \cdots, n, j \neq i$. The distance between target agent A_i and other agents can be defined as

$$d_{ij} = \sqrt{(x_i - x_j)^2 + \left(y_i - y_j\right)^2} \quad (j \neq i, i = 1, 2, \cdots, n, j = 1, 2, \cdots, n), \tag{5}$$

where if $\varepsilon \leq d_{ij} < r^{\min}$, ε is a tiny pure number when an LOS state has occurred.

Second, the agent must be judged as still within the terminal airspace or not:

$$d_{ic} = \sqrt{(x_i - x_c)^2 + (y_i - y_c)^2} > \frac{\sqrt{2}}{2} R, \tag{6}$$

where d_{ic} is the distance between the agent and the center point of the sector, x_c, y_c are the coordinates of the central point, and R is the length of the side of hexagon sector.

Third, the agent must be judged as having reached the goal state or not. If the distance value between the position of the agent and the goal position is less than the predetermined ε, the goal state has been reached.

Fourth, the agent must be judged as the distance with another agent is less than ε, where $\varepsilon > d_{ij}$; then, both agents will be erased from the scene.

2.5. Reward Function. Reward function Q can be formed via two steps. The first step is for a single aircraft agent. Every agent is aimed at completing its mission, which means that it flies from the departure airport toward the landing airport as rapidly as possible. The minimum safe separation also needs to be guaranteed throughout the process. The reward value is defined as follows:

$$r(s) = \begin{cases} 1, & \text{if } s \text{ is goal state,} \\ 0, & \text{if } s \text{ is LOS state or out of boundary,} \\ 1 - \dfrac{d(o, g)}{\max d(o, g)}, & \text{otherwise.} \end{cases} \tag{7}$$

When an agent reaches its landing airport/goal position, that agent will accumulate one reward point (i.e., $r = 1$). If the agent flies outside the boundary, it does not get any reward point. If the agent stays in the sector, it will consistently accumulate rewards as defined.

The second step is calculating reward function Q by summarizing all agent's rewards:

$$Q(s) = \sum_i r_i(s). \tag{8}$$

3. Problem Solution

The above optimization problem was solved using a framework that combined the MCTS-UCT algorithm with MMAP. In traditional UCT method [35–37], exploitation is preferred, and the node with the greatest value is chosen when the computational times of the algorithm are small. Based on the number of times of iterations, the UCT method will gradually switch to the exploration process by visiting nodes that were not visited or were less visited.

3.1. Illustration of MCTS-UCT Algorithm. The MCTS process is illustrated in Figure 1, and the detailed steps of the MCTS-UCT algorithm consist of four parts, as described below. Through these iterative steps, each agent can find the best path on the tree. In MCTS, the UCB (upper confidence bound) algorithm is used for node selection and expansion. UCT (upper confidence bound applied to trees) is a variation of the UCB algorithm specifically designed for tree search in the MCTS algorithm. UCT is an integral part of the MCTS algorithm.

In UCTs, the algorithm chooses the child node with the highest UCB value using the UCB formula. The UCB value reflected the trade-off between exploration and exploitation. It helped to balance the exploration of unexplored nodes and the exploitation of promising nodes. After selecting a child node based on this UCB values, the MCTS algorithm proceeds with simulations, backpropagation, and subsequent node selections. These steps are iteratively performed to gradually build the search tree and ultimately to find the best solution.

3.1.1. Node Selection. The first step is node selection; as shown in Figure 2, each state of agent will be viewed as a node in the tree, and the UCT values of each state are calculated by the following equation:

$$\mathrm{UCT}\left(S_j\right) = \bar{r}_j + 2C\sqrt{\frac{2 \ln N}{n_j}}, \tag{9}$$

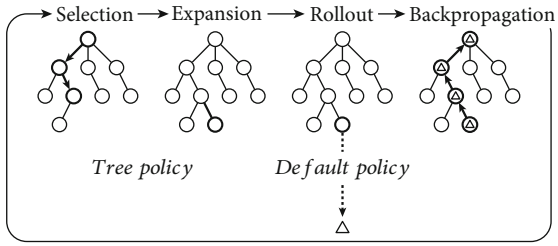


Figure 1: Illustration of MCTS process—1.

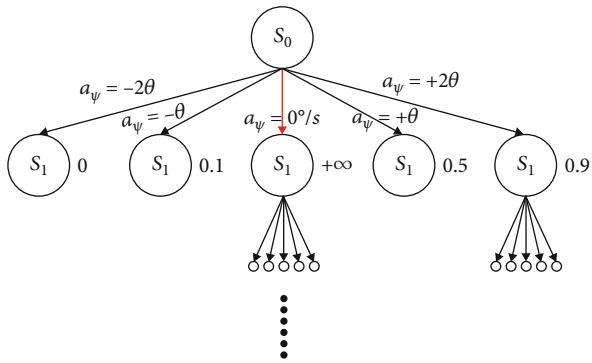


Figure 2: Illustration of MCTS process—2. The figure shows the state-action tree built in the MCTS algorithm. In this illustration, five actions were considered, $\{-2\theta, -\theta, 0\circ/s, +\theta, +2\theta\circ/s\}$. The state-action value of each node at time step $t+1$ is the average of all its child node values. Based on this illustration, the agent will select to fly straight ($a_\psi = \circ/s$) at the current state S_0.

where $\bar{r}_j$ is the mean reward value of a certain action strategy j for current agent A_j and $A_j \epsilon \{A_1, A_2, \cdots A_n\}$, N is the counter to remember how many times the leaf node has been visited based on the MCTS algorithm, and n_j is the counter to remember how many times the action strategy j has been selected. Kocsis et al. suggested setting constant C as $1/\sqrt{2}$ to satisfy the condition of Hoeffding's inequality [38] to restrict the value of the reward to between $[0, 1]$.

During the UCT process of selecting node S_j ($j = 1, 2, \cdots, n$), if node S_j has child nodes $(S_{j1}, S_{j2}, \cdots, S_{jm})$, the one with the highest UCT value will be marked as the current node. Otherwise, if S_j has no child node, the algorithm will skip to the next step:

$$S_j = \max_i \mathrm{UCT}\left(S_{ji}\right) (i \geq 2, i = 1, 2, \cdots, m). \tag{10}$$

Furthermore, if one node has never been visited, the UCT value of this node will be positive infinity because of $n_j = 0$. If more than two nodes have the same UCT value, the agent will maintain the current action strategy to the next child node.

3.1.2. Node Expansion. A tree policy is used to expand the state chain, which involves using one action strategy a_j from the action space as a path to a new child node if the number of child node of the S_j is less than or equal to the available number in the action strategy. The algorithm will not add new nodes and instead select the node with maximum UCT value for the next step. While the agent in the algorithm has reached new current node S_j, the algorithm will add a new child node for S_j by following the tree policy. This child node is then marked as new current node S_j, the cumulative reward $r_j = 0$ is initialized, the counter of the visit is set to $n_j = 1$, and action strategy A_j that leads to this node for generating the best path is stored. Every time this node is visited, the timer n_j will be updated by $n_j = n_j + 1$. n_j will also be used to calculate the average UCT reward $\bar{r}_j$ when the node has been visited more than twice:

$$\bar{r}_j = \frac{r_j}{n_j}, \tag{11}$$

where r_j in this equation will be used to accumulate rewards in the next step.

3.1.3. Rollout. After the improved MCTS-UCT algorithm has added a new node S_j in the state chain, the algorithm executes one rollout to update the r_j value of this node, meaning that this agent will execute one action strategy chosen from the action space. The chain being formed based on Markov decision process allows the agent to choose a reasonable action strategy on node S_j, because all information needed for that agent to make decisions is obtained in the state of this new node [36]. The reward of this rollout $r_{(\mathrm{rollout})}$ obtained from the following equation will be backpropagated and accumulated for r_j.

3.1.4. Backpropagation. When the rollout step has been completed, the value of $r_{(\mathrm{rollout})}$ will be backpropagated to node S_j and delivered from parent node S_j to leaf node S_0. The chain can be represented as $\{S_0, \cdots, S_j\}$. The improved MCTS-UCT algorithm can then identify the path along with maximum reward value for the node and output a series action strategy from leaf node S_0 to node S_j:

$$\begin{aligned} &\text{For } S_j : r_j = r_j + r_{(\mathrm{rollout})}, n_j + 1, \\ &\cdots, \\ &\text{For } S_0 : r_0 = r_0 + r_{(\mathrm{rollout})}, N + 1. \end{aligned} \tag{12}$$

This represents one iteration of the improved MCTS-UCT algorithm. At each iteration, the algorithm also updates the state of the current agent. When the agent obtains the action strategy a_j presented in Section 3.1.2, the algorithm will update the state of the agent at the same time as the state transition.

The improved MCTS-UCT algorithm will continue until one of the chains reaches the terminal state or the length of the chain reaches the threshold, namely, search depth D.

Figure 3 has shown the process of how an aircraft agent goes from generation to destination. The agent is initially generated at a position above the take-off airport. Given an

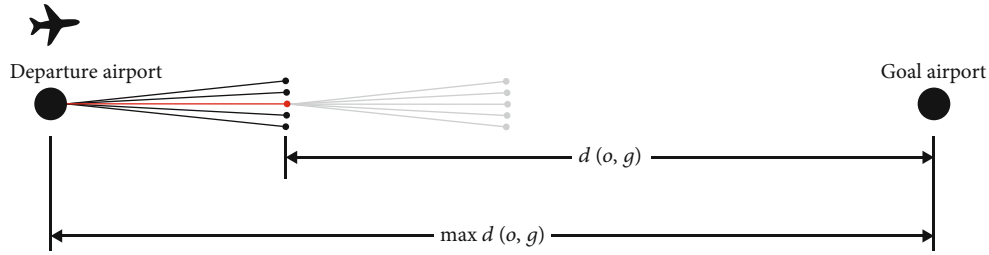


Figure 3: Illustration of process of agent flying directly to the goal airport. The red node indicates which final path agent had chosen, and the red node will be the fundamental point of next iteration step.

initial cruising speed, according to the target airport to be flown to, calculate the distance to the destination airport $d_{(o,g)}$, and according to the calculation result of the MCTS algorithm, select an action from the 5 actions provided by action state, and fly according to the heading for a time of $\Delta t = 1$ s. After reaching a new position, the information of the agent at this time is fed back to the environment, and then, the MCTS algorithm will repeat the above process according to the new position of the agent until the agent reaches the terminate state.

The pseudocode of the improved MCTS-UCT algorithm is listed in Algorithm 1, where $v_i\ (i \epsilon N)$ represents the node of the tree.

3.2. Collaborative Decision-Making Process of Multiagent. To accomplish conflict-free state, all n aircraft agents $\{A_1, A_2, \cdots, A_n\}$ in the scenario need to share their intention when choosing each individual action. K-level hierarchical model [39–43] is used to make all agents collaborate in their decision-making and share their information with each other. Since safety is equally important for each aircraft, every aircraft would try to remove the conflict state by deviating from their current trajectory. As it is assumed in the previous section, all aircraft agents are always at the same flight level, and the action strategies only include changes in heading and cruising speed. The simplified K-level model to accomplish the collaborative decision-making process is described below.

A single iteration of the MMDP algorithm is detailed in Figure 4: first, n aircraft agents $\{A_1, A_2, \cdots, A_n\}$ are initialized at the $K-1$ level. All agents continue executing the default collaborative action policy $a_{-j} = \{a_i \text{ from the default action policy} | i = 1, 2, \cdots, n\}$. Agent A_j with the minimum index in level $K-1$ is then implemented in the improved MCTS-UCT algorithm. When calculating a_j^*, the other agents will continue to follow the default action strategy set as $a_{-j}, a_{-j} = \{a_i \text{ from the default action policy} | i = 1, 2, \cdots, n, i \neq j\}$. When agent A_j obtains its optimal action strategy a_j^*, A_j is upgraded to the K level and stores the action strategy a_j^* to update the default collaborative decision strategy. The improved MCTS-UCT algorithm is then implemented into agent A_k with the minimum index of level $K-1$, as in the previous operation. This iteration process will continue until all agents are upgraded to obtain their optimal action strategy $\{a_i^* | i = 1, 2, \cdots, n\}$. These collaborative decision strategies were used for the next time step Δt for all agents. All agents will execute this collaborative decision strategy for several Δt before the next iteration; it will terminate when all agents have reached the goal state.

More precisely, take the MMDP (also called collaborate decision-making CDM) process of two aircrafts in Figure 5 as an example. The MMDP algorithm first determines the decision-making order of the agent according to the subscript order of the agent. In the figure, agent A_1 calculates the optimal action strategy set in the current state through the MCTS algorithm described above as action $\{a_1^{[1]}, a_1^{[2]}, a_1^{[3]}, a_1^{[4]}, a_1^{[5]}\}$, the MMDP algorithm uses the first step of agent A_1 to be action $a_1^{[1]} = 0 \circ$/s as a part of the environment information and passes it to the next decision-making agent A_2, and then, the agent A_2 combines the current environment information and uses the MCTS algorithm to calculate the optimal action strategy set in the current state as action $\{a_2^{[1]}, a_2^{[2]}, a_2^{[3]}, a_2^{[4]}, a_2^{[5]}\}$; then, the two agents execute the strategy set at the same time. The first step strategies taken by agent A_1 and A_2 are $a_1^{[1]}$=0/s and $a_2^{[1]} = +2\theta$, seperately. Since the decision result of the previous agent is the environmental information that the subsequent agent needs to refer to when making decisions, this can avoid the need for information in the case of no sharing, and the two agents turn to same side at the same time, which further causes the interval to decrease rapidly and develop into a situation of breaking the minimum safe separation.

Agent A_j chooses its optimal action strategy a_j^* in the algorithm using the following equation:

$$a_j^* = \underset{a_j}{\operatorname{argmax}} r_j^*\left(s, a_j, a_{-j}\right), j = 1, 2, \cdots, n, \tag{13}$$

where $r_j^*(s, a_j, a_{-j})$ is the reward value of agent A_j at the state s, to execute random action a_j, while other action strategies of the other agents are represented by a_{-j}. The output of Equation (13) is the probability of agent A_j choosing its own optimal action strategy from the action space in state s. The entire process described above is the MMDP process. The pseudocode for its algorithm is provided in Algorithm 2.

The main advantages of the proposed algorithm are the following: first, the search depth in the proposed algorithm can be adjusted for different volumes of air traffic. The search depth is an important factor in identifying the optimal solution. When the number of agents in terminal

Input: D, C, v_{max}, v_{min}
Output: Reward of all nodes, action policy of all nodes
Initialize: root node $S_{\text{leaf node}}$, timer of root node $N = 0$
Step one: Node Selection
Let current node = S_j:
If $S_j = S_{\text{leaf node}}$ and $N = 0$:
    jump to Step Two and $N + 1$
If $S_j = S_{\text{leaf node}}$ and $N \neq 0$:
add a child node and let current node = S_j
jump to Step Three
If $S_j \neq S_{\text{leaf node}}$:
    jump to Step Two
Step two: Node Expansion
Add a child node for S_j under tree policy
calculate all child nodes' UCTs value and set current node S_j
record the action strategy $a_j$and update the state of agent
Step Three: Rollout
Execute one random action strategy from action space
and calculate the reward $r_{(\text{rollout})}$
Step Four: Backpropagation
  Update the information of all nodes along the chain from leaf node S_0 to node S_j

Algorithm 1: Improved MCTS-UCT algorithm for a single agent.

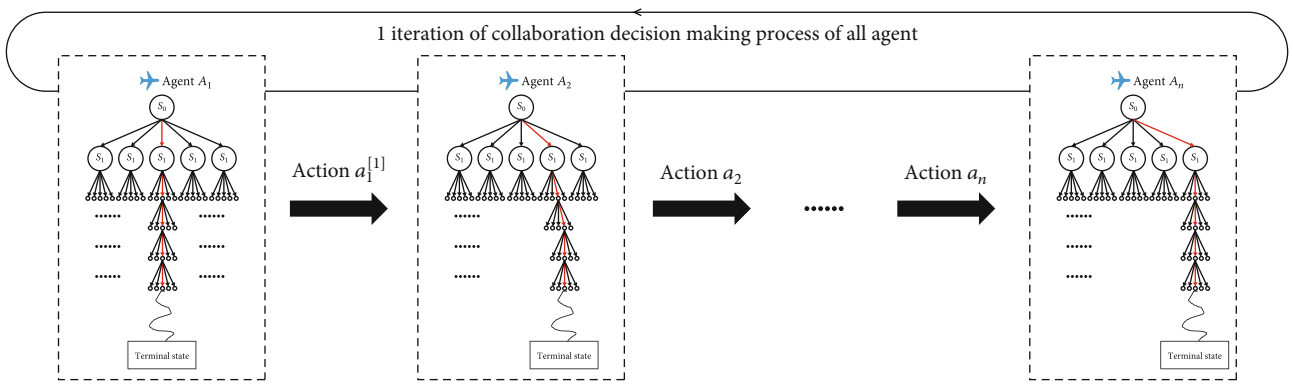


Figure 4: One iteration of collaborative decision-making process of multiagent. The red process in each blanket indicates specific action that agent had chosen, and the action will be passed through to the next agent.

airspace has increased to a high-density level, a proper search depth could decrease the number of LOS events and allow rapid calculations.

Second, the speed adjustment strategy of aircraft has been taken into consideration in action spaces. In this paper, we combined the speed adjustment strategy with the action strategy to optimize computation speed when designing the moving strategy of the algorithm. When the aircraft enters the scope of the terminal airspace, the speed of the aircraft agent was set to decrease depending on the distance to the goal position in the proposed algorithm. The speed of the agent will decrease and maintain the minimum speed of 150 kt while approaching the goal airport. Our data analysis indicates that this innovative strategy can improve the efficiency and reduce the computational time of the algorithm by 6.66% under the same conditions as a simulation experiment.

Third, the approach taken in this paper has enlarged the action space of the moving strategy by changing headings. In the process of establishing safe separation, the moving strategy of the agent was restricted to a single flight level, meaning that the agent needs to accomplish CD&R by adjusting either the speed or heading. In the proposed method, the aircraft agent is allowed to change the heading using different moving strategies. With these $2n + 1$ choices of heading change in the single decision-making process of the algorithm, agents can cope with more complicated situations and perform different strategies in a sophisticated manner [44]. For example, when the minimum distance between them and other aircraft is large, the agent would tend to choose a strategy of small heading changes. If all agents are approaching the same goal airport and the minimum distance between the agents is small, a strategy with large heading changes could

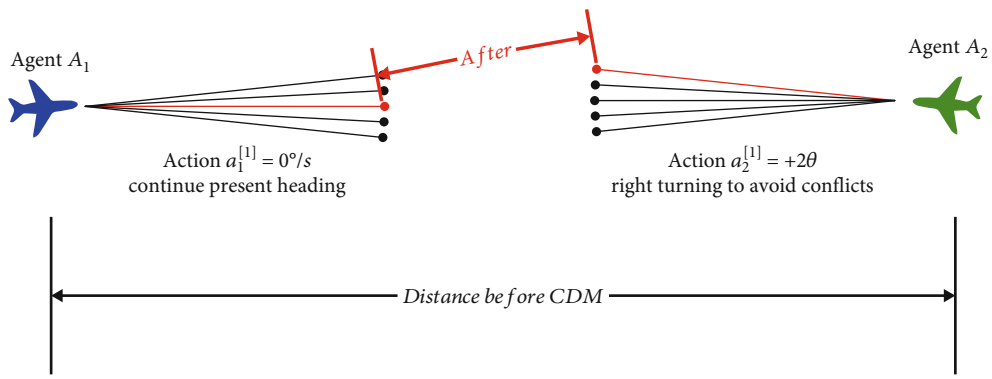


Figure 5: Illustration of MMDP process between two aircrafts in the scene. The distance between these two agents is changing as they chose different action strategy.

Input: N, Δt
Output: cumulative reward of state, number of conflicts
Initialize: Information of environment, time_step = 0, agent_index = 0
Information of environment: for every agent generated, initialize the state of the agent {initial position, initial action, initial speed, goal airport, reward counter}and add them to the environment information Dictionary
while existing agent in the map:
  Step 1: generate aircraft from airport 0~6 and initialize the state of all agents:
  for i in range (number of airports):
    if number of existing agents ≤ N and agent_index ≤ N:
      generate an aircraft agent
      agent_index += 1
      initialize the state s of agent
  Step 2: Run Improved MCTS-UCTs algorithm for each agent in order:
  if time_step % Δt = = 5:
    for i in range (number of existing agent):
      update the information of environment
      run Improved MCTS-UCTs algorithm for agent i with minimum index
      if separation between agent i with other aircraft < $r^{\min}$
        number of conflicts += 1
      time_step += 1
        update the state of agent i
End while
Output cumulative reward of state, number of conflicts

Algorithm 2: MMDP collaborative decision-making algorithm.

be more effective. The agent could also turn to a different direction without preferring a certain direction.

In the following simulation, the algorithm only had five different heading change strategies with steps of 3°/s due to the turning rates and performance limitations of the aircraft and allows for acceptable conditions for the onboard passengers. Furthermore, in accordance with ATC regulations, the controller must use a specific amount of heading change such as 5° as the minimum interval for heading change instructions, so the heading change strategies can improve the consistency between the designed simulation and the actual situation [45].

4. Simulation and Analysis

4.1. Parameter Settings. Seven airports located at the Yangtze River Delta region were used to construct the simulation, in which the Nanjing Lukou International Airport (ZSNJ) is in the center, surrounded by six other airports: Shanghai Hongqiao International Airport (ZSSS), Hangzhou Xiaoshan International Airport (ZSHC), Huangshan Tunxi International Airport (ZSTX), Hefei Xinqiao International Airport (ZSOF), Xuzhou Guanyin International Airport (ZSXZ), and Yancheng Nanyang International Airport (ZSYN). The detailed location information of these airports is shown in Figure 6.

The regular hexagonal terminal area simplified from the "Nanjing approach terminal airspace" composed of 7 busy airports in the actual Yangtze River Delta region is shown in Figure 7. The hexagon contracture is also the most common control sector shape.

At the center, ZSNJ is marked as number 0, which was the landing airport. The airports are numbered from 1 to 6

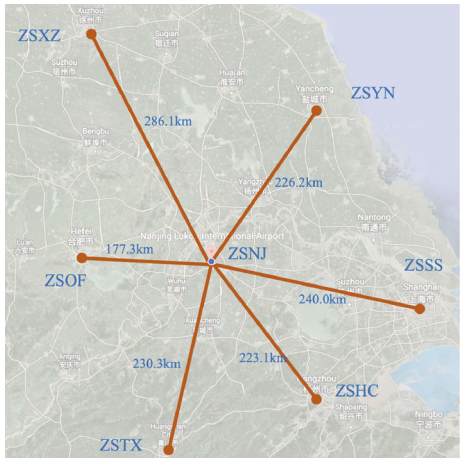


Figure 6: Geographical locations of the seven airports.

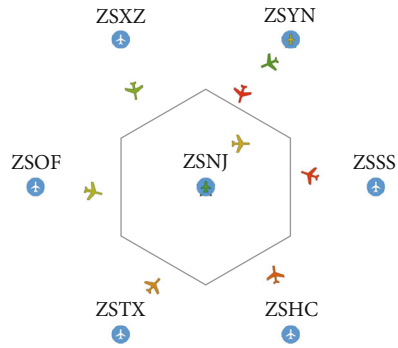


Figure 7: Simplified geometric area of the simulation scenario.

surrounding ZSNJ in an anticlockwise direction and are spread evenly, forming a bigger regular hexagon shape. There is a small hexagon surrounding the ZSNJ airport, with a side length of 50 km. If lines between any airports from numbers 1 to 6 are drawn with number 0, these lines will vertically split the side of the smaller hexagon. The smaller hexagon is called the inner sector and the larger hexagon the outer sector, which excludes the scope of the inner sector. When the aircraft is in the outer sector, the speed can be set larger.

Aircraft flow at the seven airports was generated automatically. Aircraft generated from airports 1 through 6 was defined as landing flow, while aircraft generated from ZSNJ (which had one random destination from numbers 1 to 6) was defined as the departure flow. The meaning and value of the simulation parameters in the improved MCTS-UCT and MMDP algorithms and the aircraft information are listed in Table 2. The aircraft agent's state contains the position, speed, and heading, while the agent's optional heading change and the speed change have been integrated into the state transition process. Therefore, the agent can well simulate the behavior of general aircraft airborne.

The experiments were performed using Python (version 3.7) as the programming language and PyCharm Professional (version 2019.3) as the programming software, and the computer used to run the algorithm comprised a 2.7 GHz Intel Core i5 CPU, 8 GB of 1867 MHz DDR RAM, and an Intel Iris Graphics 6100 1536 MB graphic card.

Table 2: Parameter values.

Parameters	Value
Type of aircraft	Medium
Search depth D	5
UCT parameters C	$1/\sqrt{2}$
Number of agent N	20
Minimum action step size θ	3°/s
Minimum safe separation $r^{\min}$	3 nm
Minimal time step unit Δt	1 s
Max en route speed $v_{\max}$	250 kt
Min approach speed $v_{\min}$	150 kt
K value of level K model	1
Size of landscape	800*800

Figure 8 illustrates the operational principles of the entire algorithm. The red box represents the process of updating the state for individual agents and the collaborative decision-making among multiple agents. The blue box represents the process of updating the algorithm's objective function by reinforcement learning method. The algorithm begins by generating a simulation scenario and creating the corresponding number of aircraft agents based on predetermined conditions such as the number of waves. Each agent then receives environmental information, updates its state, and adjusts the reward function. Finally, while maximizing the collaborative decision-making function, the algorithm ensures that agents complete their conflict-free flight missions from the departure airport to the destination airport.

4.2. Simulation Design. To illustrate the detailed information of improvements and achievements of the algorithm in this paper, the following three simulation scenarios were implemented. The environment contained the different numbers of agents at the same time, using action space as $\{-6\circ/s, -3\circ/s, 0\circ/s, +3\circ/s, -6\circ/s\}$ combined with a speed adjustment strategy, setting search depth from 1 to 6, and 100 simulation iterations considered as one experiment.

The purpose of the first experiment is to verify the effectiveness of the algorithm. The first is to increase the number of agents in the scene as the only variable from small to large, the search depth is 5, and a total of 100 iterations are performed. Then, on the premise of ensuring the effectiveness of the algorithm, the influence of different search depths on the parameters of the algorithm operation is explored.

The purpose of the second experiment is to verify the efficiency of the algorithm and compare it with the algorithm results of the previous paper [19]. Since the scene size and the aerodynamic model of the agent in the paper [19] are different from those of the agent in this paper, in this experiment, the basic aerodynamic model such as the flight speed of the agent is adjusted to the cruising speed of the civil aircraft. The results of the algorithms are compared

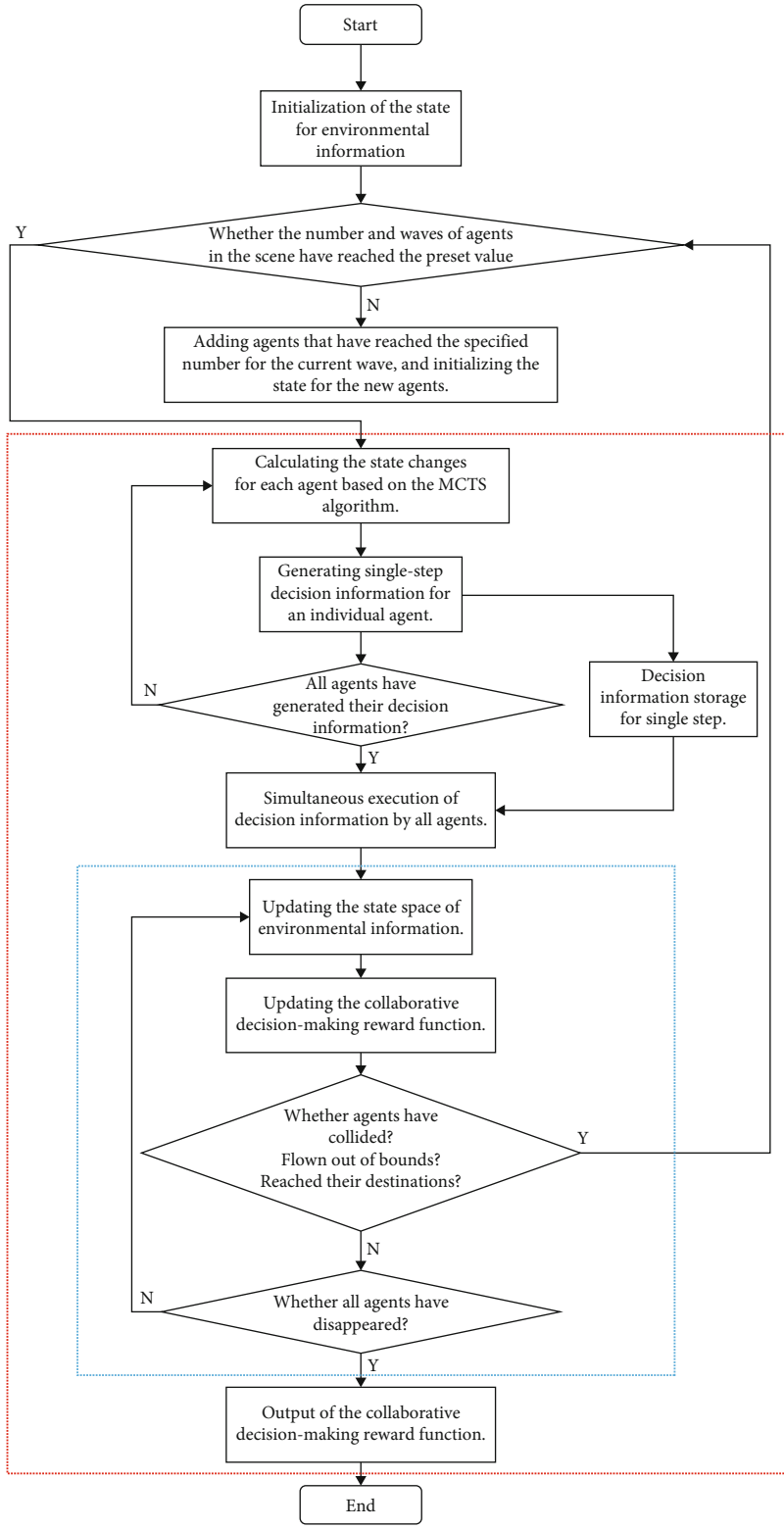


Figure 8: Overall pipeline figure for visualizing air traffic flow model.

under the conditions of the speed gradient strategy described above. The number of agents in the scene is fixed at 16, and the search depth is 3 and 5, respectively. 100 iterations are performed, respectively.

The purpose of the third experiment is to exclude that the agent in the algorithm always adopts the same strategy, that is, to verify that the algorithm can learn and perform the CD&R process independently according to the changes

of the scene. The number of agents in the scene is fixed at 16, and the search depth is dynamic. When the minimum interval between agents is more than 1.5 times the minimum safety interval, the search depth is 3, and when it is less than 1.5 times the minimum safety interval, the search depth is 5. The experiment was performed for a total of 100 iterations. In all the experimental scenarios depicted below, each scene shows agents represented by different colors, facilitating continuous tracking of their flight paths. When conflicts occur between agents, both aircraft agents disappear. Therefore, in the figures, this paper capture snapshots of the experimental process to illustrate that the algorithm not only achieves conflict-free flight goals in the scene but also accommodates a significant number of agents simultaneously.

Scenario 1. To examine the separation maintenance ability of the algorithm, aircraft flow generated from airports 1 to 6 was set airport 0 as the goal airport. The total number of aircraft increased from 5 to 20. The results are illustrated in Figure 9.

An airport from 1 to 6 was randomly chosen to generate another aircraft until the number of aircraft generated reached the setting number to simulate the real situation where departing and landing aircrafts coexist in the terminal airspace. The aircrafts departing from airport 0 are shown in Figure 10. And they are all marked with red circles.

Scenario 2. To examine the influence of different search depths in the algorithm, the number of agents was fixed at 16, consisting of 2 departure and 14 arrival aircrafts. The search depth of the proposed algorithm varied from 1 to 6.

We compared the results with other algorithms from previous research [19] to validate the efficiency of our algorithm. Figure 11 shows the results for search depths of $D = 3$ and $D = 5$. The search depth in the previous research paper was set at $D = 3$.

Scenario 3. To validate the learning ability of the algorithm and exclude the possibility that the agent always repeats the same strategy, the total number of aircraft is fixed as 16. Dynamic search depth used was between 3 and 5 according to the minimal interval between pair of agents. The training processes are illustrated as time-lapse photography in Figures 12 and 13.

In Figure 11, the destination airport of the agents generated at airports 0 and 2-6 is no. 1, and the destination of the agent generated at airport 1 is another random airport. In Figures 12 and 13, the destination airport of the agents generated at airports 1-6 is no. 0, and the destination of the agents generated at airport 0 is another random airport.

In Figures 12 and 13, this paper dynamically showcases the generation and flight paths of agents by extracting frames from experimental process videos and stacking all frames in chronological order. The top left corner of each illustration in the figures represents the stack of all frames up to the current time. In Figures 12 and 13, the experimental process that lasted for 75 s in a single iteration is divided into 5 pictures, which are displayed in the form of time-lapse photography; therefore, the flight trajectory of the agent could be clearly displayed.

4.3. Results and Analysis. In Scenario 1, it is found that the proposed algorithm could maintain a safe separation between all agents for a pure arrival flow composed entirely of approaching aircraft, or a more realistic mix flow of departure/landing aircraft. The effectiveness of the proposed algorithm is established. And Scenario 2 is a further study based on the success of the Scenario 1 experiment.

In Figure 10, it is found that when the search depth is larger, the agent's maneuvering range is larger and is no longer limited to the range of the outer sector. The red line in the figure indicates the range of the outer sector. Although this increases the fly time of the aircraft, it reduces the number of LOS events between agents, shown in Figure 13.

Table 3 and Figure 14 show the results from Scenario 1 to indicate the performance of the proposed algorithm under different search depths.

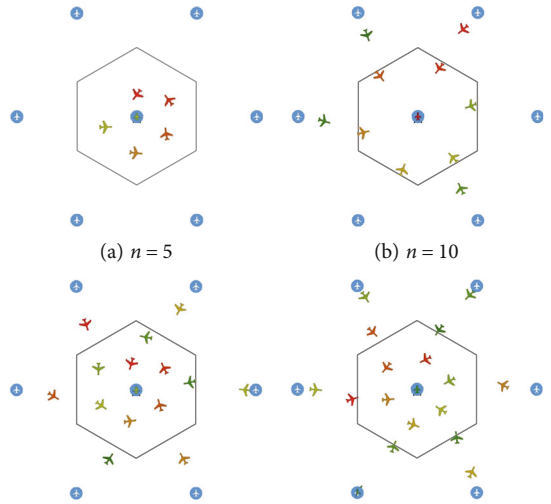
Different search depths that correspond to different average LOS numbers are shown in Figure 8.

As indicated in Tables 4 and 5, the rewards obtained by the algorithm could be stable with a fixed number of aircraft, indicating the stability of algorithm. The abnormal reward value can be explained by the iteration process of the algorithm, where LOS events occurred due to the two agents. One was delated randomly and was able to continue flying and accumulate rewards. This abnormal value could be easily observed, which helped considerably for tuning and modifying the algorithm, which finally illustrated the universality of the algorithm. Tables 4 and 5 present the cumulative total reward collected from 100 simulation experiments of agents under two different search depth scenarios. Each table consists of 10 rows, where the left column represents the experiment number, the middle column shows the reward values obtained by the agents in the 100 trials (dimensionless values), and the rightmost column indicates the average occurrence of LOS events in those 10 experiments. The LOS events are represented as a fraction; for example, 0.10 indicates that, on average, one LOS event occurred in at most one experiment out of the ten. Since the majority of experiments achieved conflict-free flight for the agents, the paper avoids listing each individual experiment's results separately to maintain clarity and readability in the table.

Table 4 shows the total reward cumulated by all agents with fixed search depth 5.

Table 5 shows the total reward cumulated by all agents of the algorithm with dynamic search depth 3-5.

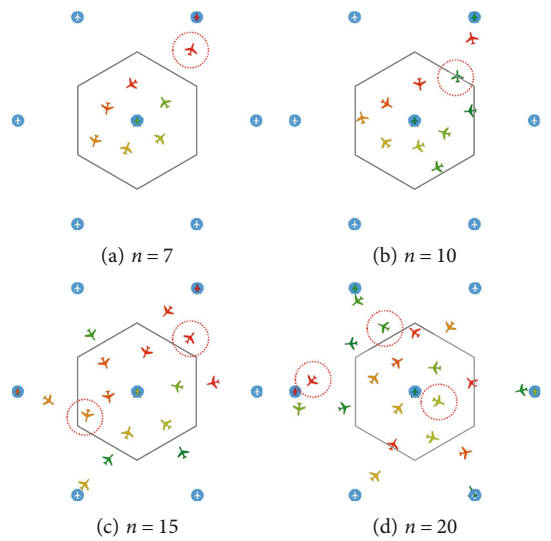
Table 6 lists the results from previous algorithm used in reference [19] and our proposed one. Four indicators were selected to compare the performance of two competing algorithms. These 4 key indicators are as follows: (1) The average value of the reward accumulated by the agent in the algorithm shows the stability of the algorithm. (2) Average en route times of every agent indicate the time it takes for the agent to travel from the starting point to the destination, measured in flight hours. This is related to the algorithm's execution time and is a dimensionless unit, representing the flight time of the agent in the scene during the



(a) $n = 5$ (b) $n = 10$

(c) $n = 15$ (d) $n = 20$

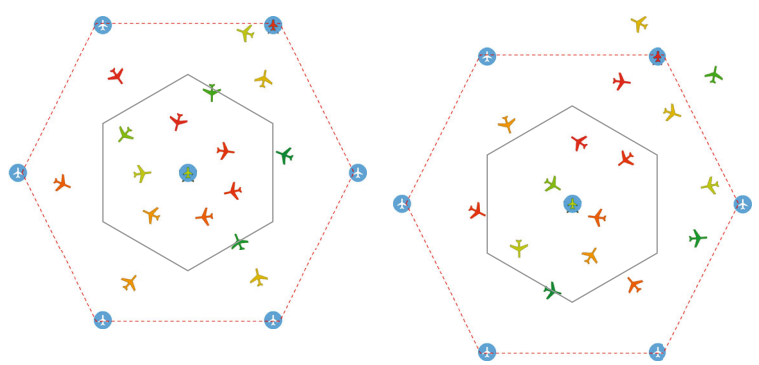
Figure 9: Landing aircraft at the goal airport with different agent numbers.



(a) $n = 7$ (b) $n = 10$

(c) $n = 15$ (d) $n = 20$

Figure 10: Flow of departure/arrival aircraft to the goal airport with different agent numbers.



(a) $D = 3$ (b) $D = 5$

Figure 11: Real-time simulation process of the algorithm with different search depths.

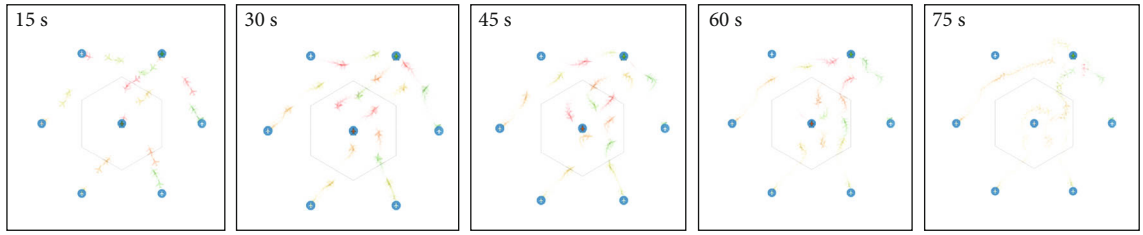


Figure 12: 75-second time-lapse photography of setting major landing airport as no. 1 with 16 agents.

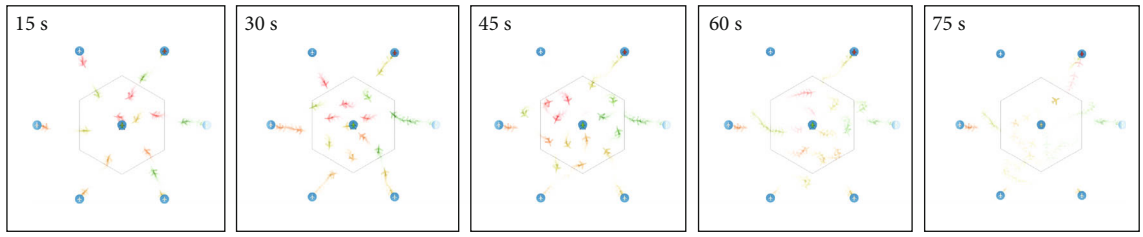


Figure 13: 75-second time-lapse photography of setting major landing airport as no. 0 with 16 agents.

Table 3: Results of Scenario 1 with different search depths.

Search depth	Time of computation (min)	Total flight hours (h)	Average time of iteration (×100)	Average number of LOS event
1	0.18	10.78	3.5	12.80
2	0.54	13.68	5.2	1.60
3	2.18	18.53	7.7	0.30
4	5.78	21.48	8.9	0.70
5	7.04	19.95	8.6	0.20
6	7.73	18.75	7.7	0.60

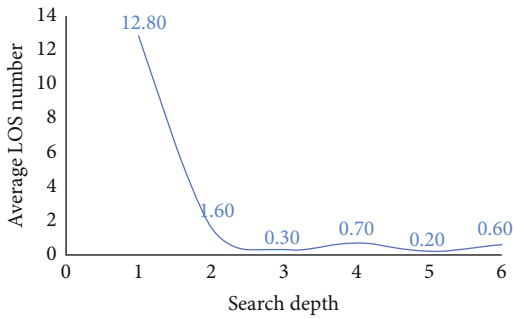


Figure 14: Line graph of search depth and average LOS number.

experiment, which demonstrates the efficiency of the algorithm. (3) The number of iterations of the algorithm is used to measure the operating efficiency of the algorithm. (4) The average number of LOS events, obtained by calculating the mean of the total occurrences of LOS events in 100 experimental trials, represents the effectiveness of the algorithm in maintaining a safe distance between agents. A lower average number of LOS events indicates a better performance of the algorithm in ensuring a safe separation between the agents.

To validate and illustrate how these innovations in the new algorithm eliminate the influence of the error caused by the shorter experimental time, Table 6 lists the key indicators selected to compare the performance of the two algorithms.

The new algorithm reached the performance of only 0.14 conflicts on average in 100 iterations. The average reward value was 148.51; it means that the agent always chose a similar optimal strategy to accomplish their goal. This provides improvements to the stability of safety maintenance in the algorithm, which is a core benefit for the air transportation industry. In the metric of "average en route times of every agent," the present algorithm demonstrates the ability to reduce the frequency of LOS events at the cost of a relatively small increase in the aircraft's airborne flight time (-3.75%). Furthermore, the average number of LOS events was reduced by 10.6%~26.32%. It was shown that the proposed algorithm has higher effectiveness and stability.

The minimum safe separation in the terminal airspace was $r^{\min} = 3$ nm. A larger safe separation compared with the minimum safe separation in areal control airspace which is 5 nm can improve airspace capacity and operation efficiency, but it also causes decreases in the reaction time of controllers required when coping with LOS events. Compared with area control, terminal control tends to implement radar vectors to change aircraft headings more frequently. The algorithm in this paper achieved rapid terminal airspace CD&R and can be carried onboard every aircraft to avoid LOS events and finally achieve and maintain rapid safe separation of all departing/landing aircraft in terminal airspace.

One of the functions of the ATC simulator was to guide ATC trainees in learning and establishing control skills and concepts [46], especially in learning to deal with emergency situations. The algorithm in this paper can visualize the whole en route process of the departure/arrival aircraft flow, including the dynamic process of the aircraft agent avoiding LOS events and ensuring the orderly arrival of aircraft. All this knowledge

Table 4: Total reward cumulated by all agents of the algorithm in this paper with fixed search depth 5.

Serial number	Total reward cumulated by all agents										LOS events/serial
1-10	151.23	134.40	152.20	152.92	152.05	153.99	149.87	153.50	151.42	152.82	0.10
11-20	151.74	150.29	153.26	151.77	151.96	151.10	146.64	153.44	152.60	112.22	0.10
21-30	153.85	153.17	153.35	151.60	135.38	152.21	152.35	77.23	153.63	154.11	0.20
31-40	110.89	153.23	152.25	161.71	153.66	151.10	153.75	151.76	153.41	153.13	0.10
41-50	151.86	152.65	153.08	152.17	163.99	153.18	153.48	146.10	151.45	162.38	0.10
51-60	151.87	154.16	153.53	152.01	152.94	152.22	152.31	151.64	152.20	130.34	0.10
61-70	153.49	152.68	152.78	153.57	145.97	153.14	150.54	152.30	127.47	152.93	0.20
71-80	151.81	120.90	152.08	151.54	162.50	162.06	151.77	153.71	153.88	149.08	0.10
81-90	107.91	105.18	95.15	153.02	151.54	153.51	136.07	154.06	154.07	151.58	0.30
91-100	152.54	152.80	153.17	147.56	161.82	152.35	149.79	153.37	152.53	154.35	0.10

Table 5: Total reward cumulated by all agents of the algorithm in this paper with dynamic search depth 3-5.

Serial number	Total reward cumulated by all agents										LOS events/serial
1-10	151.23	134.40	152.20	152.92	152.05	153.99	149.87	153.50	151.42	152.82	0.10
11-20	151.74	150.29	153.26	151.77	151.96	151.10	146.64	153.44	152.60	112.22	0.10
21-30	153.85	153.17	153.35	151.60	135.38	152.21	152.35	77.23	153.63	154.11	0.20
31-40	110.89	153.23	152.25	161.71	153.66	151.10	153.75	151.76	153.41	153.13	0.20
41-50	151.86	152.65	153.08	152.17	163.99	153.18	153.48	146.10	151.45	162.38	0.10
51-60	151.87	154.16	153.53	152.01	152.94	152.22	152.31	151.64	152.20	130.34	0.20
61-70	153.49	152.68	152.78	153.57	145.97	153.14	150.54	152.30	127.47	152.93	0.20
71-80	151.81	120.90	152.08	151.54	162.50	162.06	151.77	153.71	153.88	149.08	0.20
81-90	107.91	105.18	95.15	153.02	151.54	153.51	136.07	154.06	154.07	151.58	0.20
91-100	152.54	152.80	153.17	147.56	161.82	152.35	149.79	153.37	152.53	154.35	0.20

Table 6: Core indicators for comparing the two algorithms.

Core indicators	Previous algorithm	This paper's algorithm		Improvement
		Fixed search depth = 5	Dynamic search depth = 3 ~ 5	
Average value of reward	—	148.51	148.99	—
Average en route times of every agent/flight hours	13.04	15.19	13.52	-16.5%/-3.75%
Average times of iterations	720	890	765	-23.61%/-6.25%
Average number of LOS events	0.19	0.14	0.17	26.32%/10.6%

could help trainees to establish the key concept of air traffic control with single aircraft interactions with the entire traffic flow closely and inseparably.

5. Conclusions

This paper introduced an air traffic flow model based on a multiagent Markov decision process. The goal reward function was then optimized via an improved MCTS-UCT algorithm. High-density departure and arrival traffic flow can avoid conflict by establishing and maintaining safe separation. The proposed algorithm can reduce the number of conflicts between agents by 26.32% compared with previous methods. The experiments verified that the efficiency and accuracy of the algorithm were optimal.

The improved MCTS-UCT algorithm presented in this paper can be used as an auxiliary tool when a radar failure occurs that cannot be fixed by the ATC service and can also cover the limitation of TCAS of only helping pilots to execute decision-making in the vertical direction and finally to increase the redundancy of high-density traffic flow in terminal airspace.

The experimental scene is limited to operate in idealized airspace, while the actual East China Delta terminal contains several restricted areas such as military airports, training airspace, dangerous area such as aircraft shooting ranges, and prohibited areas in urban areas. In the future, the stability and accuracy of the algorithm could be further increased to cope with more complex simulation scenarios such as considering 3D space, unmanned aerial vehicles, dangerous

weather conditions, irregular sector shapes, and settings in restricted and prohibited areas.

Data Availability

The position and flight information data of 7 airports in Huadong Delta area was supported by Civil Aviation Administration of China. The data of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

The authors are indebted to Ms. Qiuxiang Wu with Nanjing LES Information Technology Co., Ltd. for the suggestion on the humanized display of simulation results and images. The authors acknowledge the financial support from the National Natural Science Foundation of China (grant no. U2233208).

References

[1] "2019 civil aviation industry development statistical bulletin," http://www.caac.gov.cn/big5/www.caac.gov.cn/en/HYYJ/NDBG/202011/W020201123499246549689.pdf.

[2] T. Radii, D. Novak, and B. Jurii, "Reduction of air traffic complexity using trajectory-based operations and validation of novel complexity indicators," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 11, pp. 3038–3048, 2017.

[3] Q. Zhuang and K. Deng, "The realization of flow management technology in solving the air traffic congestion pre-perception and intelligent guidance in North China," in *2020 International Signal Processing, Communications and Engineering Management Conference (ISPCEM)*, pp. 112–116, Montreal, QC, Canada, 2020.

[4] D. Meyer, T. Wypych, V. Petrovic, J. Strawson, S. Kamat, and F. Kuester, "An air traffic control simulator for test and development of airspace management schemes," in *2018 IEEE Aerospace Conference*, pp. 1–8, Big Sky, MT, USA, 2018.

[5] "Civil aviation air traffic management regulations of China," 2018, http://www.caac.gov.cn/PHONE/XXGK_17/XXGK/MHGZ/201712/P020171221370496163543.pdf.

[6] G. Xiao, D. Liu, X. Liu, F. He, and D. Ding, "Study on aircraft generalized conflict resolution and trajectory optimization with multiple constraint in complex airspace environment," in *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*, pp. 1–6, San Diego, CA, USA, 2019.

[7] Y. I. Jenie, E. van Kampen, J. Ellerbroek, and J. M. Hoekstra, "Safety assessment of a UAV CD&R system in high density airspace using Monte Carlo simulations," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 8, pp. 2686–2695, 2018.

[8] B. Pang, W. Dai, T. Ra, and K. H. Low, "A concept of airspace configuration and operational rules for UAS in current airspace," in *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*, pp. 1–9, San Antonio, TX, USA, 2020.

[9] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep reinforcement learning for multiagent systems: a review of challenges, solutions, and applications," *IEEE Transactions on Cybernetics*, vol. 50, no. 9, pp. 3826–3839, 2020.

[10] G. G. N. Sandamali, R. Su, K. L. K. Sudheera, Y. Zhang, and Y. Zhang, "Two-stage scalable air traffic flow management model under uncertainty," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 12, pp. 7328–7340, 2021.

[11] R. Breil, D. Delahaye, L. Lapasset, and É. Féron, "Multi-agent systems for air traffic conflicts resolution by local speed regulation and departure delay," in *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, pp. 1–10, Sacramento, CA, USA, 2016.

[12] Y. Jian, Y. Dong, Z. Yu, and J. Lixuan, "Multi-agent coordination in high velocity UAVs conflict detection and resolution," in *2015 34th Chinese Control Conference (CCC)*, pp. 2621–2626, Hangzhou, China, 2015.

[13] A. Bayen, P. Grieder, G. Meyer, and C. J. Tomlin, "Langrangian delay predictive model for sector-based air traffic flow," *Journal of Guidance, Control, and Dynamics*, vol. 28, no. 5, pp. 1015–1026, 2005.

[14] A. K. Agogino and K. Turner, "A multiagent approach to managing air traffic flow," *Autonomous Agents and Multi-Agent Systems*, vol. 24, no. 1, pp. 1–25, 2012.

[15] Z. Wang, H. Li, H. Wu, F. Shen, and R. Lu, "Design of agent training environment for aircraft landing guidance based on deep reinforcement learning," in *2018 11th International Symposium on Computational Intelligence and Design (ISCID)*, pp. 76–79, Hangzhou, China, 2018.

[16] D. T. Pham, N. P. Tran, S. K. Goh, S. Alam, and V. Duong, "Reinforcement learning for two-aircraft conflict resolution in the presence of uncertainty," in *2019 IEEE-RIVF International Conference on Computing and Communication Technologies (RIVF)*, pp. 1–6, Danang, Vietnam, 2019.

[17] X. Yang and P. Wei, "Autonomous on-demand free flight operations in urban air mobility using Monte Carlo tree search," in *Proceedings of the International Conference on Research in Air Transportation (ICRAT)*, pp. 26–29, Barcelona, Spain, 2018.

[18] D. O. Stahl II and P. W. Wilson, "Experimental evidence on players' models of other players," *Journal of Economic Behavior & Organization*, vol. 25, no. 3, pp. 309–327, 1994.

[19] X. Yang and P. Wei, "Scalable multi-agent computational guidance with separation assurance for autonomous urban air mobility," *Journal of Guidance, Control, and Dynamics*, vol. 43, no. 8, pp. 1473–1486, 2020.

[20] H. A. O. Siqi, S. Cheng, and Y. Zhang, "A multi-aircraft conflict detection and resolution method for 4-dimensional trajectory-based operation," *Chinese Journal of Aeronautics*, vol. 31, no. 7, pp. 1579–1593, 2018.

[21] E. Sunil, J. Hoekstra, J. Ellerbroek et al., "Metropolis: relating airspace structure and capacity for extreme traffic densities," in *Proceedings of the ATM Seminar 2015, 11th USA/EUROPE Air Traffic Management R&D Seminar*, pp. 23–26, Lisbon, Portugal, 2015.

[22] N. Okuniek, Z. Zhu, Y. C. Jung, S. Gridnev, I. Gerdes, and H. Lee, "Performance evaluation of conflict-free trajectory taxiing in airport ramp area using fast-time simulations," in *2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)*, pp. 1–10, London, UK, 2018.

[23] P. Scala, *Optimization-Simulation Implementations for Harmonizing Operations at Big Airports*, Hogeschool van Amsterdam, 2019.

[24] E. Hernández Romero, A. Valenzuela Romero, D. Rivas Rivas, M. Steiner, and J. Pinto, "Probabilistic aircraft conflict detection in the terminal maneuvering area," in *8th International Conference on Research in Air Transportation (2018). International Conference on Research in Air Transportation (ICRAT)*, Barcelona, 2018, https://idus.us.es/handle/11441/104838.

[25] M. Liang, *Aircraft Route Network Optimization in Terminal Maneuvering Area*, Université Paul Sabatier (Toulouse 3), 2018.

[26] R. Tang, Z. Yang, J. Lu, H. Liu, and H. Zhang, "Real-time trajectory prediction of unmanned aircraft vehicles based on gated recurrent unit," in *Green Connected Automated Transportation and Safety*, pp. 585–596, Springer, Singapore, 2022.

[27] G. Xie and X. Chen, "Efficient and robust online trajectory prediction for non-cooperative unmanned aerial vehicles," *Journal of Aerospace Information Systems*, vol. 19, no. 2, pp. 143–153, 2022.

[28] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics Research*, pp. 3–19, Springer, Berlin, Heidelberg, 2011.

[29] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *2008 IEEE International Conference on Robotics and Automation*, pp. 1928–1935, Pasadena, CA, USA, 2008.

[30] S. Chen and Y. Li, "An overview of robust reinforcement learning," in *2020 IEEE International Conference on Networking, Sensing and Control (ICNSC)*, pp. 1–6, Nanjing, China, 2020.

[31] C. F. Sironi, J. Liu, and M. H. M. Winands, "Self-adaptive Monte Carlo tree search in general game playing," *IEEE Transactions on Games*, vol. 12, no. 2, pp. 132–144, 2020.

[32] T. Cazenave, "Sequential halving applied to trees," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 7, no. 1, pp. 102–105, 2015.

[33] Y. Wang and S. Gelly, "Modifications of UCT and sequence-like simulations for Monte-Carlo Go," in *2007 IEEE Symposium on 490 Computational Intelligence and Games*, pp. 175–182, Honolulu, HI, USA, 2007.

[34] S. Wollkind, J. Valasek, and T. Loerger, "Automated conflict resolution for air traffic management using cooperative multiagent negotiation," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, p. 4992, Providence, RI, USA, 2004.

[35] X. Lin, P. A. Beling, and R. Cogill, "Multiagent inverse reinforcement learning for two-person zero-sum games," *IEEE Transactions on Games*, vol. 10, no. 1, pp. 56–68, 2018.

[36] B. Arneson, R. B. Hayward, and P. Henderson, "Monte Carlo tree search in hex," *IEEE Transactions on Computational Intel-496 ligence and AI in Games*, vol. 2, no. 4, pp. 251–258, 2010.

[37] C. B. Browne, E. Powley, D. Whitehouse et al., "A survey of Monte Carlo tree search methods," *IEEE Transactions on Computational Intelligence & Ai in Games*, vol. 4, no. 1, pp. 1–43, 2012.

[38] L. Kocsis, C. Szepesvári, and J. Willemson, *Improved Monte-Carlo Search*, Tech. Rep., University of Tartu, 2013.

[39] C. F. Camerer, "Behavioral game theory: experiments in strategic interaction," in *Advances in Behavioral Economics*, pp. 374–392, Princeton University Press, 2011.

[40] D. O. Stahl and P. W. Wilson, "On players' models of other players: theory and experimental evidence," *Levines Working Paper Archive*, vol. 10, no. 1, pp. 218–254, 1995.

[41] P. Xuan, V. Lesser, and S. Zilberstein, "Communication in multi-agent Markov decision processes," in *Proceedings Fourth International Conference on MultiAgent Systems*, pp. 467-468, Boston, MA, USA, 2000.

[42] Y. Rizk, M. Awad, and E. W. Tunstel, "Decision making in multiagent systems: a survey," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 10, no. 3, pp. 514–529, 2018.

[43] A. Feriani and E. Hossain, "Single and multi-agent deep reinforcement learning for AI-enabled wireless networks: a tutorial," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 1226–1252, 2021.

[44] A. Alonso-Ayuso, L. F. Escudero, and F. J. Martín-Campo, "Multiobjective optimization for aircraft conflict resolution. A metaheuristic approach," *European Journal of Operational Research*, vol. 248, no. 2, pp. 691–702, 2016.

[45] Y. Purwananto, W. Wibisono, C. Fatichah, and B. J. Santoso, "A heuristic approach for multi-objective aircraft conflict detection and resolution," *2019 12th International Conference on Information & Communication Technology and System (ICTS)*, 2019, pp. 349–354, Surabaya, Indonesia, 2019.

[46] K. Richard, C. Shannon, M. Moallemi, B. Chhaya, K. Patel, and S. Jafer, "Smart web-based air traffic control training technology," in *2019 Integrated Communications, Navigation and Surveillance Conference (ICNS)*, pp. 1–9, Herndon, VA, USA, 2019.