Policy Optimization in Automated Point Merge Trajectory Planning: An Artificial Intelligence-based Approach

Man Liang, Weigang Li, Daniel Delahaye, Philippe Notry

Policy Optimization in Automated Point Merge Trajectory Planning: An Artificial Intelligence-based Approach

Man Liang
School of Engineering
University of South Australia
Adelaide, Australia
annie.liang@unisa.edu.au

Weigang Li
Translab, Department of Computer Science
University of Brasilia
Brasília, Brazil
weigang@unb.br

Daniel Delahaye, Philippe Notry
OPTIM Research Lab
Ecole Nationale de l'Aviation Civile
Toulouse, France
daniel@recherche.enac.fr

***Abstract*—Air traffic management is a complex decision making process. Air traffic controllers decision on aircraft trajectory control actions directly lead to the efficiency of traffic flow management. This paper aims to realize an automated routine trajectory management in terminal manoeuvring area with an intelligent decision making agent. An artificial intelligence based approach is applied to adaptively and smartly integrate four types of deconflict actions for resolving conflicts. Especially, the reinforcement learning policy optimization process is discussed in detail. Firstly, application of reinforcement learning in adaptive trajectory planning is presented. The entire problem is adaptively divided into several sub-problems. For each sub-problem, an online policy is applied to guide the simulation and optimization modules to find out the conflict free and less delay solution. The online policy is a scale of weight distribution for choosing desirable actions. It follows the rule of roulette wheel selection with weighted probability. The highest desirable decision variable has the largest share of the roulette wheel, while the lowest desirable decision variable has the smallest share of the roulette wheel. Direct policy optimization algorithm is designed to update the online policy. Finally, experiments are built up for validation of the proposed policy optimization algorithm for the intelligent decision making process. The results in the test environment showed that learning agent with different exploration and exploitation ability will result in different system performance in conflict resolution and delay.**

***Index Terms*—air traffic management, decision making, artificial intelligence, reinforcement learning, policy optimization**

I. Introduction

Air traffic congestion is one of the most challenging problems in current air transportation system. Heavy traffic congestion causes not only economic loss, but also pollution. In order to enhance capacity, a number of new operational concepts were proposed. In view of enhancing automation in the tactical and pre-tactical levels of Air Traffic Management (ATM), reducing the controllers' intervention through advanced decision support tools is an important approach. This idea is not new. Actually, different kinds of traffic advisor have been developed since the 1990s in the U.S. and Europe, such as the Center-TRACON Automation System (CTAS) and the Final Approach Spacing Tool (FAST) [1]–[3]. However, for a long time, the acceptance of automation by controllers has been a bottleneck for successful implementation of automation in air traffic control units [4]. In en-route airspace, the key question of free flight is whether the airborne self-separation can safely accommodate the very high traffic demand [5]. ATM is a complex decision-making process that requires involvement of multiple stakeholders. Air traffic controllers' decision on aircraft trajectory control actions directly lead to the efficiency of traffic flow management. With the rapid development in Artificial Intelligent (AI) and Machine Learning (ML) technologies, the intelligent decision making could play an important role in realizing a positive capacity enhancement with better human acceptance of automation. It shapes the digital transformation of the future ATM system.

Automated Point Merge Trajectory Planning (APMTP) aims to realize an automated routine trajectory management in Terminal Maneuvering Area (TMA). In terms of the architectural concept, an intelligent decision-making agent is designed to work together with two main modules: a simulation module and an optimization module. During the implementation process, an AI-based approach, mainly Reinforcement Learning (RL) algorithm, is applied to guide the agent adaptively and smartly integrating four types of deconflict actions, so as to improve the overall system performance, i.e. solving conflicts with fewer delay in the environment. It combines tactical air traffic flow management with automated conflict detection and resolution. In this paper, we primarily discuss the policy optimization in APMTP, focusing on improving the agents' learning quality and exploration efficiency.

This paper is organized as follows, in Section II, a short discussion about previous studies on air traffic flow management. Then, in Section III, the AI-based methodologies used in addressing APMTP problem. Especially, the implementation of intelligent agent in flow managing decision making and the application of policy optimization analyzed in detail. In Section IV, experiments with relatively large time scales were designed for validation of the proposed approach. Simulation results were discussed. Finally, in Section V, conclusion and future works were outlined.

II. Previous Study

In the operational reality, for a long time, traffic flow managers used their experiences to estimate the flow rate. They give flow regulation to adjacent units based on the congestion situation in the sector. It is not precise and efficient. The efficiency of controllers' conflict detection and resolution has a direct influence on the traffic flow rate. When facing the same traffic scenario, a good controller usually provides better conflict resolution. They could achieve a higher traffic flow rate without compromising safety. Facing the periodic traffic characteristics of traffic flow, in the view of intelligent decision making, those best human controlling cases should be learned, and the best strategy should be transferable to sufficiently handle future traffic circumstances. Combining the tactical flow management with automated conflict resolution is an interesting topic to investigate.

In tactical Air Traffic Flow Management (ATFM), the traffic flow rate on a specific route or some significant waypoint reflects the congestion level of this area. Controllers need to choose the right conflict resolution action to maintain an orderly flow in airspace, especially safely and efficiently merging flows. A good flow management strategy will keep the traffic flow rate at a reasonable level, so that the maximum capacity in a specific airspace could be achieved. In view of application of RL algorithm to ATFM problem, [6] proposed an agent based adaptive approach to automatically manage the traffic flow in airspace. Instead of choosing aircraft as an agent, "fix" or "waypoint" can also be assigned as an agent. All agents aim to learn the appropriate separation for their location using a RL algorithm. The agent's action could speed up or slow down traffic to manage congestion. It was tested using a simulation tool FACET (Future ATM Concept Evaluation Tool). Based on this research, [7] further enriched the RL reward function by considering the safety factors and fairness impact. Case studies in Brazil were described to show the effectiveness and efficiency of the new RL reward functions in the controller decision process of ATFM.

In fact, Multi-Agent System (MAS) has been used to address the ATFM problem for a long period. It was found that the agent selection is a fundamental problem. Aircraft waypoints could be chosen as agents [6]. If aircraft are selected as agents, then there will be hundreds of thousands of aircraft (agents) in the sky on a given day as a result, agent mobility in the environment would be very high. Massive agents' communications and negotiations make the MAS hard to be observed and controlled. The level of automation acceptance is low. Waypoint based MAS could largely reduce system complexity, however it is limited to only controlling the traffic flow rate at a specific waypoint, it cannot handle the traffic flow between the waypoints, including sequencing aircraft from different flight levels to assigned altitudes, conflict detection and resolution of in-route etc. In addition, the learning speed of the RL applied in ATFM needs to be considered to match the real-time operational requirement as well. Therefore, in our APMTP problem, instead of choosing "waypoint (fix)" as agent, we plan to design only one virtual agent. This agent aims at applying the reinforcement learning approach to adaptively choosing the right conflict resolution action by interacting with the traffic flow environment, resulting in an optimal traffic flow rate and less delays, a conflict-free flow management.

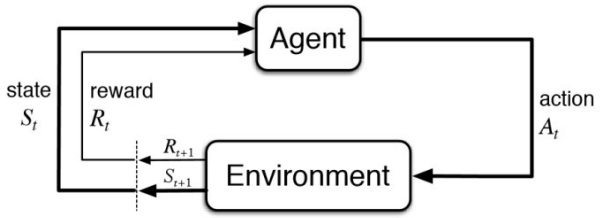


Fig. 1. Principle of Reinforcement Learning

III. Methodologies

A. Application of RL in Adaptive Trajectory Planning

RL is learning what to do, how to map situations S_t to actions A_t, so as to maximize a numerical reward signal R_t, shown by Fig. 1. The learner (agent) is not told which actions to take, but must discover which actions yield the most reward by trying them. In the most interesting and challenging cases, actions may affect not only the immediate reward but also the next situation S_{t+1} and, through that, all subsequent rewards R_{t+1} [8]. Beyond the agent and the environment, there are four main sub-elements of a RL system:

- Policy: a mapping from perceived states of the environment to actions to be taken on those states.
- Reward signal: the goal of a reinforcement learning problem. On each time step, the environment sends to the agent a single number.
- Value function: the total amount of reward an agent can expect to accumulate over the future, starting from that state.
- Environment model: it mimics the behavior of the environment.

The Markov Decision Process (MDP) is an approach in RL to take decisions in an environment. Before addressing MDP to the APMTP problem, some notations need to be mentioned:

- $k \in \{1, 2, ..., K\}$ describes the different steps of sliding window.
- S_k is a set of states of traffic for the kth sliding window.
- A_k is a set of actions for the kth sliding window.
- R_k is a set of rewards for the kth sliding window.
- $n \in \{1, 2, ..., N\}$ describes the different iterations in one sliding window.
- $(s_k^n)_{n\in\{1,2,...,N\}}$ is the process of different states of traffic in the kth sliding window.
- $(a_k^n)_{n\in\{1,2,...,N\}}$ is the process of different actions in the kth sliding window.
- $(s_k^n, a_k^n)_{n\in\{1,2,...,N\}}$ is the process of state-action pairs in the kth sliding window. The actions which can be taken depend on the current policy π.

- $(r_k^n)_{n\in\{1,2,...,N\}}$ is the process of different rewards following a triple (state s_k^n, action a_k^n, resulting state s_k^{n+1}).

As shown in Fig. 2, in the APMTP problem, firstly, we decompose the entire optimization problem (from T_{INIT} to T_{FINAL}) into several sub-optimization problems by the application of sliding window. Then, in the active sliding window (from $T_s(k)$ to $T_e(k)$), after going through a sequence of actions and states with a policy π, together with simulation module (SI) and Optimization module (OP), we arrive at a terminal state s_k^N, which is corresponding to the best conflict-free and less-delay solution for current sub-problem. After that, we update the policy π, and move the sliding window forward to next sub-problem. The new policy π is learned by the intelligent decision making agent (DA). In brief, a finite MDP for our problem in sliding window k could be described as following:

- In some state s_k^n,
- Taking an action a_k^n by following a online policy π,
- The state-action pair (s_k^n, a_k^n) leads to another following state $s_k^{n+1}|(s_k^n, a_k^n)$ by working together with SI and OP modules,
- The triple $(s_k^n, a_k^n, s_k^{n+1})$ leads to a reward r_k^n.

The methodology with SI, OP modules and the decision making agent is shown in Fig. 3. In the SI module, trajectory is generated with consideration of the route network design and Base of Aircraft Data (BADA). In the OP module, Simulated Annealing (SA) algorithm is applied to find the optimized conflict-free solution for a set of traffic. A decision making agent is to optimize the delay at airport. The ML-PM route network helps to achieve a more dynamic and less constrained position shift in landing queue. The SA algorithm finds out a global optimal conflict-free solution, removing all potential conflicts in all the trajectories. The decision making agent adaptively chooses the right policy π for combining the four available actions. Note, that this agent aims to find a good policy by application of RL, so as to achieve a minimum cumulative delay, or a maximum cumulative reward, at the airport.

B. Action space

The action space in this problem is a discrete action space. For example, one arrival aircraft $i \in \mathcal{F}_{arr}$ is selected. It could be controlled by four possible types of actions: speed regulation at the entry point of TMA (g), entry time modification at the entry point of TMA (j), turning time controlling on the sequencing leg (h), and runway allocation in multiple runways (w).

$$a_k^n \in A_k, A_k = \{g, j, h, w\}. \tag{1}$$

The selected type of action a_k^n is transferred to update the trajectory of aircraft in the Simulation module. In the Simulation module, the actual trajectory of aircraft i is controlled by four decision variables: the actual entry time at TMA (t_i^a), the actual entry Calibrated Airspeed (CAS) at TMA (v_i^a), the actual turning time on the sequencing legs (t_i^T), and the actual runway-in-use (x_i^a). The selected type of action a_k^n is going to update one of these four decision variables. More in details, the time t_i^a is adjusted by a type of action $a_k^n = j$,

$$t_i^a = t_i^e + j\Delta|_{\Delta=2s}, j = -150, -149, ..., 449, 450. \tag{2}$$

where t_i^e is the Estimated Time of Arrival (ETA) at the entry point of TMA (p_i^e). The value range of j depends on the time constraint [-5 mins, +15 mins].

The speed v_i^a is changed in a discrete way by a type of action $a_k^n = g$,

$$v_i^a = v_i^e(1+g), g = 0, \pm 1\%, \pm 2\%, ..., \pm 15\%. \tag{3}$$

where v_i^e is the initial CAS at the entry point p_i^e.

The time t_i^T is changed in a discrete way by a type of action $a_k^n = h$,

$$t_i^T = t_{imin}^T + h(t_{imax}^T - t_{imin}^T), h = 0\%, 1\%, ..., 100\%, \tag{4}$$

where t_{imin}^T is the earliest turning time for aircraft i, and t_{imax}^T is the latest turning time.

The runway-in-use x_i^a is defined by a type of action $a_k^n = w$,

$$x_i^a = w \in \{0, 1\}. \tag{5}$$

C. Reward structure

The key issue needs to be addressed is to select the reward structure for the intelligent decision making agent. In this paper, the difference system performance is considered as the reward r_k^n for this agent on the nth iteration in the kth sliding window,

$$r_k^n = G_k^{n+1} - G_k^n, \tag{6}$$

where G_k^n is the system performance on the nth iteration in the kth sliding window. The system performance evaluation function is considered as a linear combination of two terms:

$$G_k^n = -\alpha A_k^n - \gamma B_k^n, \tag{7}$$

where A_k^n is the total conflict penalty and B_k^n is the total delay penalty. α and γ are two experience-based parameters. Then, we have:

$$r_k^n = G_k^{n+1} - G_k^n = \alpha(A_k^n - A_k^{n+1}) + \gamma(B_k^n - B_k^{n+1}). \tag{8}$$

The total conflict penalty A_k^n is relative to the sum of total potential conflicts C_k^n with all the aircraft in the kth sliding window, and is given by:

$$A_k^n = C_k^n. \tag{9}$$

So we have:

$$r_k^n = G_k^{n+1} - G_k^n = \alpha(C_k^n - C_k^{n+1}) + \gamma(B_k^n - B_k^{n+1}). \tag{10}$$

The difference of the total delay penalty $(B_k^n - B_k^{n+1})$ is relative to all aircraft F_{arr}. Because in the kth sliding window, on one SA iteration, we randomly modify the set of aircraft $\mathcal{F}_{arr}$, thus we have:

$$B_k^{n+1} - B_k^n = \sum_{i\in\mathcal{F}_{arr}} \Theta\left(t_i^L(a_k^n) - ETA_i^L\right)\left(t_i^L(a_k^n) - ETA_i^L\right)^2, \tag{11}$$

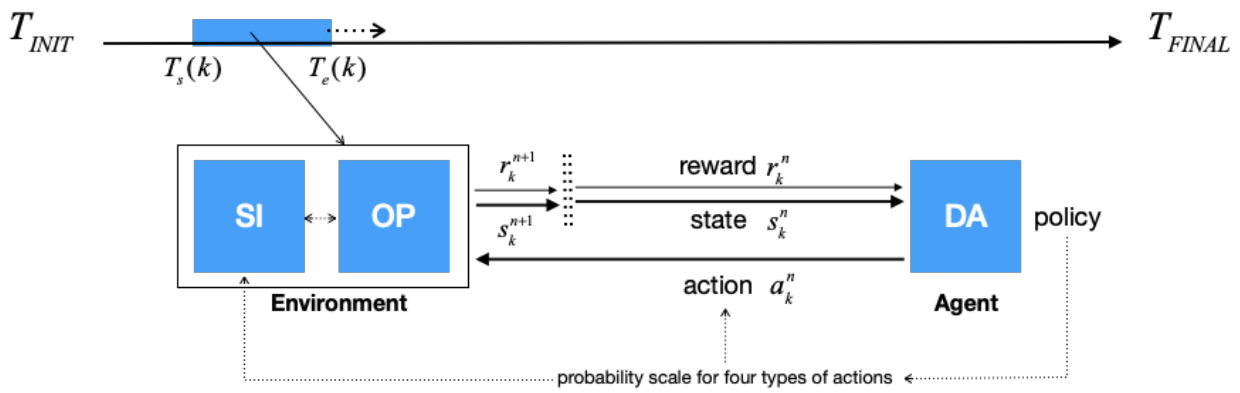


Fig. 2. Markov Decision Process

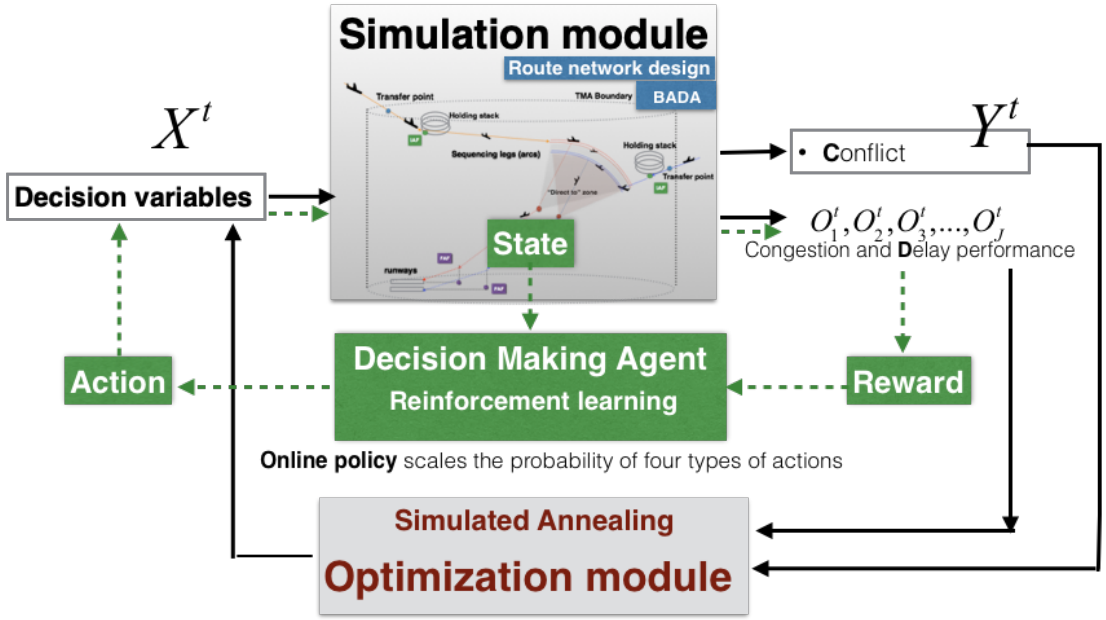


Fig. 3. New Methodology with the Decision Making Agent

where ETA_i^L is the estimated time of landing of aircraft i, and t_i^L is the actual landing time of flight i. They are computed by the Simulation module and $\Theta(\cdot)$ is the step function that equals 1 when the argument $\left(t_i^L(a_k^n) - ETA_i^L\right)$ is greater than zero, otherwise it equals 0. In particular, we use the square deviation of landing time of aircraft i for computing the difference of the total delay penalty $(B_k^n - B_k^{n+1})$, when aircraft i has to be late. It will be useful for the SA algorithm in the Optimization module to consider the fairness between airlines for sharing the possible delay due to congestion. It is to avoid single huge delay on one specific aircraft in landing queue. Finally, we have:

$$r_k^n = G_k^{n+1} - G_k^n = \alpha(C_k^n - C_k^{n+1}) - \gamma \sum_{i \in \mathcal{F}_{arr}} \Theta\left(t_i^L(a_k^n) - ETA_i^L\right)\left(t_i^L(a_k^n) - ETA_i^L\right)^2. \quad (12)$$

It is found that the reward r_k^n is both sensitive to the overall system deconflict performance and the individual aircraft delay penalty. In another words, different strategy to scaling the four types of actions will have different effect on the system deconflict performance and delay penalty performance. In this case, we would like to have a conflict-free full system performance, so we define $\alpha = 1$ for significance of conflict. The magnitude of total square delay is significantly large compared with the magnitude of conflict, so we chose average value of square delay to simplify (12). $\gamma = 0.001$ is an experience based parameter. Note, that choosing the right conflict resolution action will result in a higher conflict-free traffic flow rate and less delay.

D. Policy optimization

In the APMTP problem, it is hard to estimate the value of a (state, action) pair. Without a definite value function, a direct policy optimization is used in this study case. Thus, in the sliding windows, we keep running the same online

policy π. This online policy is a scale of weight distribution for choosing the desirable actions. It follows the rule of Roulette-wheel selection with weighted probability. The highest desirable decision variable has the largest share of the roulette wheel, while the lowest desirable decision variable has the smallest share of roulette wheel. The default weights are: $[25\%, 25\%, 25\%, 25\%]$. All the four types of actions have the same probability to be chosen. However, it may not be always the optimal policy to find the optimal conflict-free and less-delay solution. Due to each sliding window, the traffic situation is different. Then, it is better to only evaluate the advantage of current policy π at the end of SA iterations in the sliding window. Due, frequently changing the online policy on each iteration with reference to the temporal reward may lead to pay more attentions to the action itself, fail to find out the overall conflict-free solution together with the PI module.

In addition, balancing the ratio of exploration and exploitation is an important problem in reinforcement learning [9]. The agent can choose to explore its environment and try new policy in the future, or keep the default policy.

Finally, the direct policy optimization algorithm 1 is described as Algorithm 1. It is a deterministic policy optimization algorithm.

Algorithm 1 Direct policy optimization

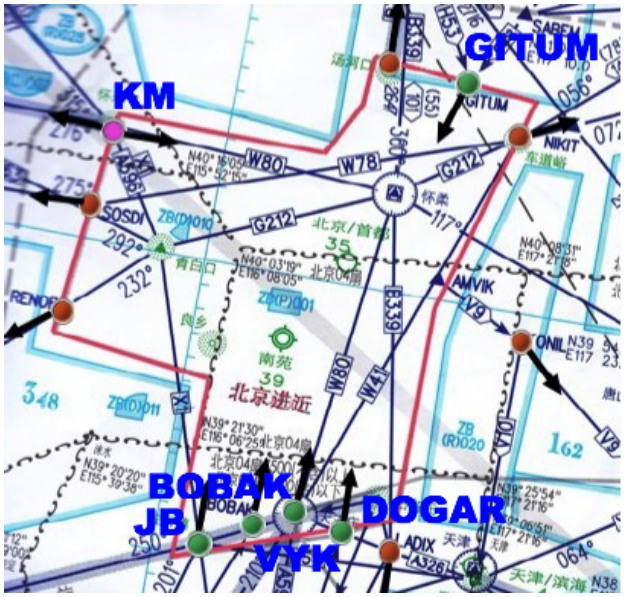
for $k = 1 \rightarrow k$ **do**
  for $n = 1 \rightarrow k$ **do**
    In some state s_k^n, follow a policy π to randomly choose the highest desirable action a_k^n;
    Put action a_k^n in SI module, update the trajectories in environment;
    Compute s_k^{n+1} together with OP module;
    Compute r_k^n;
  end for
  Compute the frequency of four types of actions for the lowest temperature in SA iterations;
  Update the online policy to $\pi*$ with probability α, keep the default policy [25%,25%,25%,25%] with probability $1-\alpha$.
end for

IV. Experiments and Results

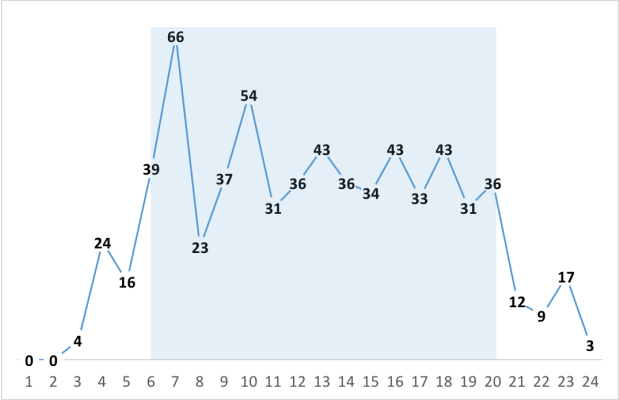
A. Environment setting up

Taking Beijing Capital International Airport (BCIA) as a study case, there are four entry points in the South: JB, BOBAK, VYK and DOGAR, two entry points in the North: KM and GITUM, see Fig. 4a. The possible manoeuvring area in this TMA for sequencing and merging dense arrival flows are very limited.

The set of aircraft (Dataset 1) for simulation representing the routine traffic data is based on the flight plan at BCIA on specific day in Dec. 2015. There are 670 flights planned to land at BCIA in 24 hours, 76.27% of which are "Medium", 23.58% of which are "Heavy". The geographic distribution of traffic is as follows: 14.03% traffic come from KM, 15.97% from JB, 12.24% from BOBAK, 18.66% from VYK, 18.66% from DOGAR, and 20.45% from GITUM. In total, 57.76% flights are planned to land on runway 01-19, and 42.24% flights will land on runway 18L-36R. The hourly distribution of flight data in 24 hours is shown in Fig. 4b. In this paper, we only focus on the range of 6:00-7:00 for initial investigation.



(a) TMA of BCIA



(b) Hourly coming flights in BCIA TMA

Fig. 4. Test data set-up

Our program is coded in Java. The default values of the parameters in the RHC-SA algorithm are designed as the same in Reference [10].

B. Numerical results and discussion

1) Experiment 1, $\alpha = 1$: In the first experiment, $\alpha = 1$ means that the agent always accept the new policy. We compare the average square delay value for each conditions, see Tab. I. Note that WIND-ON means wind data is put into SI module for generating trajectory of aircraft. DELAY-ON means delay factor is put in the OP module neighborhood function for finding the new aircraft candidate for reducing potential conflicts, meanwhile it is put in the objective function

TABLE I
AVERAGE SQUARE DELAY [MIN*MIN]

Cases	**No. arrivals/hour**	**Unsolved Conflicts**	**Average square delay** [min*min]
1) WIND-OFF+DELAY-OFF	41	0	148
2) WIND-ON+DELAY-OFF	41	0	175
3) WIND-OFF+DELAY-ON	41	0	8
4) WIND-ON+DELAY-ON	41	0	11
5) POLICY OPTIMIZATION OFF	41	0	167

TABLE II
POLICY (RWY-SPEED-TURNING-SLOT) LEARNING PROCESS

Policy Learning	**CASE 1** WIND-OFF DELAY-OFF	**CASE 2** WIND-ON DELAY-OFF	**CASE 3** WIND-OFF DELAY-ON	**CASE 4** WIND-ON DELAY-ON
Initial setting	25-25-25-25	25-25-25-25	25-25-25-25	25-25-25-25
Update 1	26-28-36-10	23-28-35-14	40-15-20-25	23-47-14-16
Update 2	20-28-40-12	30-28-34-8	33-33-27-7	27-33-27-13
Update 3	29-29-33-9	42-19-32-7	18-48-21-13	33-30-24-13

for minimize total average delay. Case 1) to case 4) are with policy optimization function ON. Case 5) is with policy optimization function OFF.

The results show that with DELAY-ON condition, the proposed system could find less delay conflict-free solutions. For example, with DELAY-ON, the average square delay with WIND-OFF in case 3 is 8 and with WIND-ON in case 4 is 11, which improves around 95% delay performance than the previous two cases with DELAY-OFF condition. In addition, delay performance is sensitive to wind effect, because the average square delay in case 2 increases around 18% than that in case 1, and in case 4 it increases around 38% than that in case 3. Note that, case 5 is WIND-OFF and DELAY-OFF with the default setting [25%:25%:25%:25%]. Thus, policy optimization improves the delay performance.

The policy learning results are shown in Tab. II. It is found that entry slot time modification is not very desirable to find less delay conflict-free solutions. In all cases, the weight for slot time is relatively lower. However, it does not mean that slot time modification is useless for conflict resolution. The fact is that the maximum number of potential conflicts is in the first window and the minimum number of potential conflict is in the third window. It is found that when the potential conflict is greater, then more weight is distributed to slot time modification. Thus, slot time modification is a very useful deconflict action to handle dense traffic demand. But, when the potential conflict is less and we don't consider the delay performance, such as in update 3 of case 1 and case 2, it is found that weight distribution to slot time modification is even less than that of case 3 and case 4. Note, that the average square delays for case 1 and case 2 are relatively higher than case 3 and case 4. The reason may be that, the slot time modification is less desirable in case 1 and case 2, but once it is selected for deconflict, its modification may be relatively large. However, in case 3 and case 4, the slot time modification is less desirable, but once it is selected for deconflict, its modification has to be relatively small, which leads to less delays.

2) Experiment 2: The second experiment is to analyze the impact of α with different values. For simplification, here $\alpha = 0, \alpha = 0.5, \alpha = 0.8, \alpha = 1$. The traffic sample A with 29 new arrivals in the first sliding window (3600s) is compared with the traffic sample B with 13 new arrivals in the second sliding window (3600s). The results are shown in Fig. 5 and Fig. 6. The axis Y is system performance which is considered as a linear combination of two terms: conflict penalty and delay penalty. Conflict penalty is with heavier weighting. The axis X is the iteration. At the beginning of the iteration, there is a great number of potential conflicts, thus the overall system performance is bad. Then, after resolving all the potential conflict the overall performance is improved.s

The decision agent learns the policy from traffic sample A, and apply the new policy to traffic sample B. In the case $\alpha = 1.0$, the decision agent always accepts the new policy. Comparing the system performance with traffic sample A and traffic sample B, it is found that when $\alpha = 1.0$, the policy for sample B is better, because the system performance is improving rapidly. When $\alpha = 0$, the default policy setting [25%:25%:25%:25%] is always applied, then system performance in traffic sample B is not improved. While when $\alpha = 0.8$, the system performance is relatively stable, and when $\alpha = 0.5$, the system performance is improved.

V. CONCLUSION

Air traffic management is a complex decision-making process with multiple stakeholder requirements. Air traffic controllers' decisions on tactical control actions directly lead to the efficiency of traffic flow management. This paper describes a novel framework of integrating one intelligent agent with simulation module and optimization module. It discusses the approach of reinforcement learning to address automated decision making problem in ATFM. Meanwhile, it discusses about the integration problems with APMTP in the agent implementation process, mainly Markov decision making, action space and policy optimization. The numerical results show the possibility of applying intelligent agent in

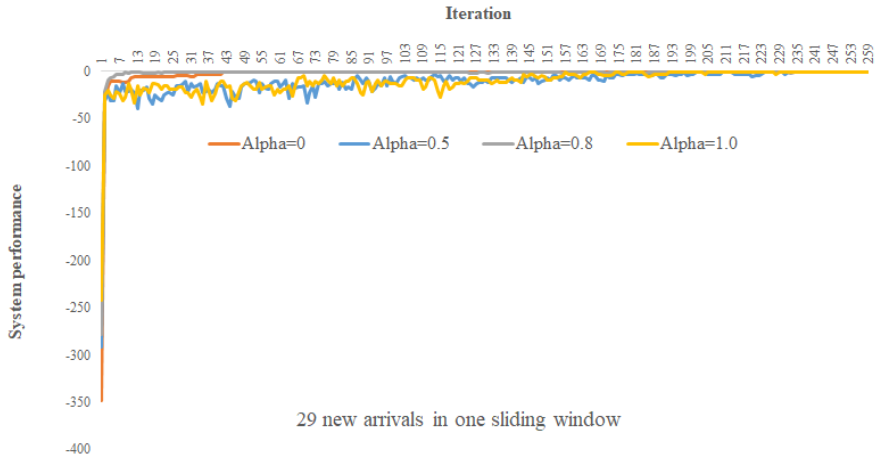


Fig. 5. System performance with traffic sample A

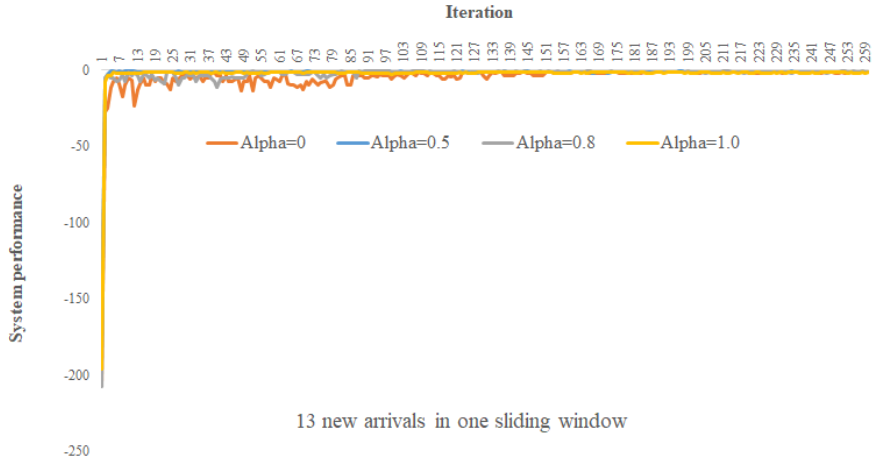


Fig. 6. System performance with traffic sample B

the trajectory planning to help reducing delay. Learning agent with different exploration and exploitation ability will result in different system performance in conflict resolution and delay. Because the size of traffic dataset is not of sufficient quantity, so more experiments are required to validate the learning model. Meanwhile, another policy optimization algorithm may be required to do a comparative study.

REFERENCES

[1] H. Erzberger and W. Nedell, "Design of automated system for management of arrival traffic," 1989.

[2] H. Erzberger, "Ctas: Computer intelligence for air traffic control in the terminal area," 1992.

[3] T. J. Davis, H. Erzberger, S. M. Green, and W. Nedell, "Design and evaluation of an air traffic control final approach spacing tool," *Journal of Guidance, Control, and Dynamics*, vol. 14, no. 4, pp. 848–854, 1991.

[4] G. Spinardi, "Up in the air: Barriers to greener air traffic control and infrastructure lock-in in a complex socio-technical system," *Energy Research and Social Science*, vol. 6, pp. 41 – 49, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2214629614001340

[5] H. A. Blom and G. Bakker, "Safety evaluation of advanced self-separation under very high en route traffic demand," *Journal of Aerospace Information Systems*, vol. 12, no. 6, pp. 413–427, 2015.

[6] K. Tumer and A. Agogino, "Distributed agent-based air traffic flow management," in *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, ser. AAMAS '07. New York, NY, USA: ACM, 2007, pp. 255:1–255:8. [Online]. Available: http://doi.acm.org/10.1145/1329125.1329434

[7] L. L. Cruciol, A. C. de Arruda, L. Weigang, L. Li, and A. M. Crespo, "Reward functions for learning to control in air traffic flow management," *Transportation Research Part C: Emerging Technologies*, vol. 35, pp. 141 – 155, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0968090X13001320

[8] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, 2018.

[9] R. S. Sutton, A. G. Barto *et al.*, *Introduction to reinforcement learning*. MIT press Cambridge, 1998, vol. 135.

[10] M. Liang, D. Delahaye, and P. Maréchal, "Integrated sequencing and merging aircraft to parallel runways with automated conflict resolution and advanced avionics capabilities," *Transportation Research Part C: Emerging Technologies*, vol. 85, pp. 268–291, 2017.