JOURNAL OF ADVANCED TRANSPORTATION
J. Adv. Transp. 2016; **50**:326–347
Published online 22 July 2015 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/atr.1324

Conflict-free time-based trajectory planning for aircraft taxi automation with refined taxiway modeling

Tianci Zhang*, Meng Ding, Bangfeng Wang and Qian Chen

College of Civil Aviation, Nanjing University of Aeronautics and Astronautics, 29 Jiangjun Road, Jiangning, Nanjing 211106, China

SUMMARY

To mitigate airport congestion caused by increasing air traffic demand, the trajectory-based surface operations concept has been proposed to improve surface movement efficiency while maintaining safety. It utilizes decision support tools to provide optimized time-based trajectories for each aircraft and uses automation systems to guide surface movements and monitor their conformance with assigned trajectories. Whether the time-based trajectories can be effectively followed so that the expected benefits can be guaranteed depends firstly on whether these trajectories are realistic. So, this paper first deals with the modeling biases of the network model typically used for taxi trajectory planning via refined taxiway modeling. Then it presents a zone control-based dynamic routing and timing algorithm upon the refined taxiway model to find the shortest time taxi route and timings for an aircraft. Finally, the presented algorithm is integrated with a sequential planning framework to continuously decide taxi routes and timings. Experimental results demonstrate that the solution time for an aircraft can be steadily around a few milliseconds with timely cleaning of expired time windows, showing potential for real-time decision support applications. The results also show the advantages of the proposed methodology over existing approaches. Copyright © 2015 John Wiley & Sons, Ltd.

KEY WORDS: airport ground movement; time-based trajectory; taxiway modeling; conflict avoidance; zone control; sequential planning

1. INTRODUCTION

With the ever-growing air traffic demand, many airports have reached the upper-bound of their capacities under current operational practices. To improve the utilization of airport resources, the surface trajectory-based operations concept has been proposed, which has proven to be more effective than the traditional aggregate queue-based approach. It uses decision support tools to automatically determine time-based trajectories for aircraft ground movement. Other automation systems are used to guide the aircraft movement in accordance with the assigned trajectories and detect any possible deviations [1–4].

The time-based trajectory, also known as the four-dimensional trajectory (4D trajectory), describes the taxi route and the required time of arrivals (RTAs) at important control points along the route [5]. If all aircraft can strictly follow the assigned trajectories, the trajectory planning problem would become a static planning problem with known environmental information. And every time an aircraft enters the system and calls for a trajectory, we only need to consider all the previously assigned trajectories and the predefined traffic rules to avoid conflicts. However, such ideal ground movement status can seldom occur in real situations because of the highly dynamic nature of the airport transportation system. The input data are affected by uncertainties from other system modules such as runway scheduling [6] and gate assignment [7]. And the

*Correspondence to: Tianci Zhang, College of Civil Aviation, Nanjing University of Aeronautics and Astronautics, 29 Jiangjun Road, Jiangning, Nanjing 211106, China. E-mail: zhangtiancy@163.com

execution of taxi plans suffers from unexpected events like mechanical failures and human errors. As a result, the planning process must run in real time in practice and be integrated with other system modules to update planning results when new data become available [8].

1.1. State of the art

The taxi trajectory planning problem is handled using mixed integer linear programming (MILP) formulations in [9–12]. Networks are used to model the taxiway structure, where vertexes stand for runway exits, stands, or taxiway intersections and arcs stand for taxiways connecting the vertexes. An inherent problem of the MILP approach is the high computational demand. To make the solution method efficient enough to be integrated with other airport management modules, the Lagrangian decomposition method is used in [8]. Clare and Richards [12] achieved the scalability of the presented MILP formulation by means of the receding horizon scheme and the iterative conflict avoidance strategy.

Genetic algorithm (GA) based optimization is another commonly used approach in airport taxi trajectory planning [13–20]. Because GAs are heuristics rather than exact methods like MILP formulations, they give no guarantee for the solution in many cases. However, because of the fact that airports usually need real time decisions, the execution time of an algorithm is a crucial measure, which makes heuristics such as GAs outperform MILP formulations from this point of view; for a more comprehensive overview of the MILP and GA-based approaches, we recommend the recently published review [21].

In general, most of the aforementioned methods (e.g., [9,10,12,22]) try to determine taxi routes and RTAs for a group of aircraft all at once in a bid to achieve the global optima. However, the computational demand of the optimization often makes it difficult to deal with large-scale problems [12,23]. According to the recent researches, an average solution time of more than 1 min is required to find optimal taxi trajectories for a few aircraft even after introducing the receding horizon and iterative conflict avoidance strategy to improve the computational performance. And this solution time will increase remarkably with the number of aircraft within the planning horizon [8,12]. To realize real-time optimization of taxi routes and RTAs, Lesire [24] utilizes a fast sequential planning method, which determines taxi routes and RTAs for one aircraft after another. It takes all the previously assigned taxi routes and RTAs into consideration to avoid conflicts when planning the trajectory for a current aircraft. It also deals with the uncertainty in the starting time prediction and taxiing speed. In addition to the low computational cost, this sequential planning method can also avoid affecting other aircraft's trajectories when finding a feasible taxi trajectory for current aircraft, which is desired to reduce the work load for controllers and pilots. Ravizza *et al.* [25] study a similar sequential planning strategy and explore its integration with taxi time prediction [26,27] and stand-holding problems [28–30], making the solution method more realistic.

To deal with the dynamic turnaround process of an airport with aircraft arriving and departing frequently, the trajectory planning must always take into consideration the influence of current traffic status, especially the near future movement of active aircraft with already assigned taxi trajectories from previous planning horizons. This ability is achieved by the use of the rolling window in [9,12,17,22]. It works in such a way that only aircraft that are taxiing or will start taxiing within the rolling window are considered in the present optimization process. Thus, conflicts within the rolling window between already planned aircraft and newly joined aircraft can be resolved in the optimization. However, because the rolling window set boundaries on the planning horizon, it often results in partial taxi plans as well as some loss in the overall optimality [12]. Another possible disadvantage is that the taxi plan for an aircraft may keep changing because of frequent replanning. The sequential planning strategy can avoid such disadvantages in nature via its decoupled planning process and extremely low computational demand. Whenever an aircraft calls for a new trajectory because of some unexpected fault, it will try to find out a feasible solution in real time without affecting all the normally moving aircraft.

Other problems regarding optimizing aircraft ground movement such as smoothing speed profiles between successive RTAs and along the whole trajectory [31], balancing fuel consumption and taxi

DOI: 10.1002/atr

times [31,32], integrating with other related operations (e.g., runway scheduling [6], gate assignment [7]), and dealing with the uncertainty in the prediction data [24] and the dynamic update of trajectories in case of deviations [33] are also critical to make the decision support tools applicable for surface trajectory-based operations.

1.2. Contributions

To our knowledge, the existing taxi trajectory planning methods mentioned earlier have used a similar taxiway modeling approach that models taxiway intersections as network vertexes. The resulting network model of the taxiway structure brings convenience in finding feasible taxi routes and RTAs by making use of existing methods in related research fields (e.g., the graph theory). However, it results in biases to model an intersection as a vertex when considering precise trajectory-based surface operations.

Figure 1 shows an example of this problem. In ideal trajectory-based operations, the guidance system [34,35] should guide aircraft movement following the curve *ab*, instead of the broken line *avb*, when traversing the intersection showed in Figure 1. As a result, if we use the typical network model for taxi trajectory planning, we would have to further determine the position of *a* and *b*, as well as the corresponding RTAs to provide precise taxi guidance around the displayed intersection.

In order to avoid this problem, we present a refined taxiway modeling approach that can reserve the inner structure of an intersection in this paper. It models the taxiway structure upon a zone-based partition of the taxiways and distinguishes intersections from other taxiway segments (which are called lanes as explained in the next section). In order to provide conflict-free taxi trajectories, we first discuss the conditions for taxi conflict avoidance and set up corresponding zone control rules to meet these conditions using time window-related concepts. Then a dynamic routing and timing algorithm is formulated accordingly to find out conflict-free taxi trajectories for a newly joined aircraft, which is a direct extension of the method presented in [36]. The presented algorithm can be naturally integrated with the sequential planning framework, which shows potential for realistic surface trajectory planning [25].

1.3. Synopsis

The remainder of this paper is organized as follows: Section 2 describes the refined taxiway modeling approach. Section 3 discusses the conflict avoidance problem and presents the zone control rules for conflict-free occupancy of intersections and lanes. Section 4 describes the trajectory-planning algorithm and analyzes its complexity. Section 5 uses experiments on different airport models to investigate the computational performance of the presented algorithm and compares it with existing approaches to show its advantages. Section 6 concludes the paper and gives future research directions.

2. TAXIWAY MODELING

To improve modeling fidelity of the taxiway structures, this paper develops a refined taxiway modeling approach based on taxiway partitioning, which can be naturally integrated with a zone control strategy presented in the next section to avoid taxi conflicts. Figure 2 is an illustration to help

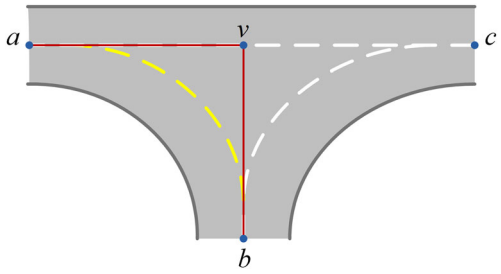

Figure 1. Intersection modeling.

DOI: 10.1002/atr

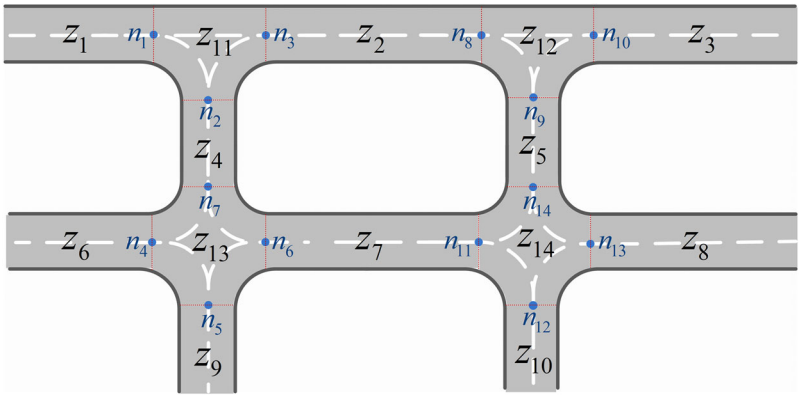

Figure 2. Illustration of the refined taxiway partition model.

describe the concept of the new taxiway model. Firstly, taxiways are partitioned into independent zones according to the physical layout. The resulting zones comprise two types: lanes ($z_1 \sim z_{10}$) and intersections ($z_{11} \sim z_{14}$), where lanes refer to taxiways connecting two intersections. Then access nodes of each intersection are identified ($n_1 \sim n_{14}$), which specify the crossover points of taxiway central lines on zone borders. Then the connectivity of every two nodes is identified. In this way, a taxi route can be described as a sequence of nodes, and the corresponding spatial taxi trajectory can be easily reproduced using the connecting taxiway central lines.

There is a major difference between a lane and an intersection from the point of view of the neighboring zones: a lane only connects with intersections and never connects directly with another lane, while an intersection can connect with both lanes and intersections. This observation inspires the hierarchical taxiway model showed in Figure 3. It consists of two layers: the bottom layer is the taxiway partition model, upon which the zone control strategy presented in Section 3 can be implemented to avoid conflicts between aircraft; the top layer is a typical network model with intersections modeled as network vertexes and lanes connecting intersections modeled as edges, upon which we shall build our trajectory planning algorithm in Section 4.

3. TAXI CONFLICT AVOIDANCE

Safety is a crucial concern for advanced airport ground movement management [37,38]. In general, the planning stage should provide conflict-free trajectories for all the aircraft. This paper adapts the zone control strategy commonly used in the automated guided vehicles system field [39–41] to avoid conflicts between aircraft during the planning process. After identifying different types of taxi conflicts upon the taxiway partition model, it sets up conditions for conflict-freeness of the planned trajectories for each zone. Extensions are made to deal with specific aspects regarding the taxi trajectory planning problem on airport. Then, four zone control rules are formulated to guide the search of taxi trajectories by the dynamic routing and timing algorithm presented in Section 4. These rules are formulated in

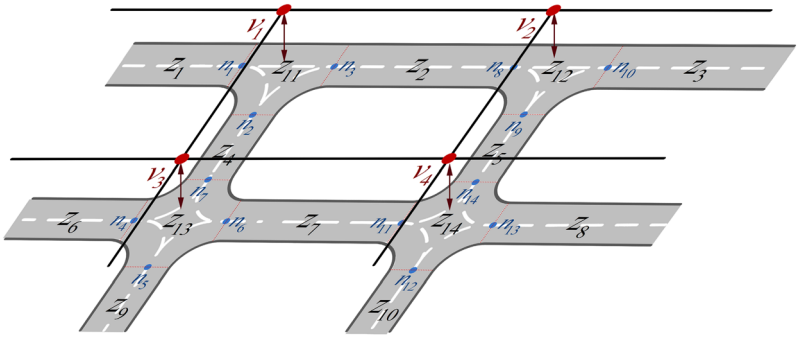

Figure 3. Hierarchical model for dynamic routing and zone control.

DOI: 10.1002/atr

compliance with the conflict-freeness conditions, so the presented algorithm is able to find out conflict-free trajectories for newly joined aircraft.

3.1. Conflict-freeness conditions

In this section, we present conflict-freeness conditions for taxi trajectories of a group of aircraft based on the time window-related concepts listed in Table I.

- Conflict-freeness within intersections. The intersection conflict-freeness condition can be described as follows:

$$\forall z \in Z_I,\ u_z^i, u_z^j \in U_z, \quad u_z^i \cap u_z^j = \varnothing \tag{1}$$

where Z_I denotes the set of intersections of the taxiway partition model; u_z^i and u_z^j denote the *real* occupancy time windows in zone z by aircraft i and j, respectively; the operator "$\cap$" calculates the overlapped part of two time windows; the symbol "$\varnothing$" in this paper means "not a valid time window", that is, being empty or a single time point.

Table I. Definitions of time window related concepts.

Concept	Symbol	Meaning
Free time window (for intersections)	w_z	A maximal time window during which intersection z is empty of aircraft.
Free time window set (for intersections)	W_z	The ordered set of free time windows for intersection z by the start time.
Forward/backward free time window (for lanes)	w_z	A maximal time window during which lane z is not occupied by any aircraft moving in the backward/forward direction.
Forward/backward free time window set (for lanes)	$W_{z,0}/W_{z,1}$	The ordered set of forward/backward free time windows for lane z by the start time.
Occupancy time window	u_z^i	A time window during which aircraft i occupies zone z.
Occupancy time window set (for intersections)	U_z	The ordered set of occupancy time windows for intersection z by the start time.
Forward/backward occupancy time window set (for lanes)	$U_{z,0}/U_{z,1}$	The ordered set of forward/backward occupancy time windows for lane z by the start time.
Full-capacity time window (for lanes)	f_z	A maximal time window during which the number of aircraft occupying lane z simultaneously equals its capacity.
Full-capacity time window set (for lanes)	$F(w_z)$	The ordered set of full-capacity time windows within a forward or backward free time window w_z of lane z by the start time.
Occupancy and release sequence, ORS (for lanes)	$S(w_z)$	The ordered set of the start and end time of all occupancy time windows within a forward or backward free time window w_z of lane z.
Exit time window	e_z^i	A maximal time window during which aircraft i can exit from zone z without causing conflict within z.

DOI: 10.1002/atr

Condition (1) means an intersection can only be occupied by one aircraft at a time. For lanes, however, situations are much more complicated. Consider the formulation of the taxiway partition model presented in Section 2: Firstly, we can divide the intersections from the entire taxiway structure. Then we examine whether the remaining lanes between intersections are long enough to hold an aircraft. If not, we should merge them with adjacent intersections. This modeling process implies that some lanes may be long enough to hold multiple aircraft simultaneously. When multiple and bidirectional occupancy of lanes is considered, the conflict scenarios in a lane will consist of three different types, namely the head-on conflict, the overtaking conflict, and the capacity conflict. Head-on conflicts occur when aircraft try to occupy a lane in opposite directions simultaneously. Overtaking conflicts happen when multiple aircraft move in the same direction in a lane and an aircraft will overtake a leading one within the lane. Finally, the number of aircraft occupying a lane simultaneously should not exceed its capacity. Otherwise, a capacity conflict occurs because there would not be enough space between aircraft to eliminate the wake vortex impacts.

- Conflict-freeness within lanes. The conflict-freeness conditions with respect to the aforementioned three scenarios can be described as follows:

$$\forall z \in Z_L,\ u_z^i \in U_{z,0},\ u_z^j \in U_{z,1},\quad u_z^i \cap u_z^j = \varnothing \tag{2}$$

$$\forall z \in Z_L,\ u_z^i, u_z^j \in U_{z,d},\ d \in \{0,1\}\quad TS\left(u_z^i\right) > TS\left(u_z^j\right) \Rightarrow TE\left(u_z^i\right) \geq TE\left(u_z^j\right) + \varepsilon \tag{3}$$

$$\forall z \in Z_L, d \in \{0,1\},\quad MO\left(U_{z,d}\right) \leq c_z \tag{4}$$

where Z_L is the set of lanes of the taxiway partition model; *TS* and *TE* are functions retrieving the start and end time of a time window, respectively; ε is the minimum separation time required between two consecutive aircraft moving in the same lane; *MO* is the function that calculates the maximal number of overlapped time windows in a time window set (Section 4.3); and c_z is the capacity of lane z.

Condition (2) implies two aircraft cannot occupy a lane simultaneously in different directions so that the head-on conflict can be avoided. Condition (3) ensures first-in-first-out (FIFO) occupancy of a lane so that no overtaking conflict occurs within it. Condition (4) guarantees the maximal number of aircraft occupying a lane simultaneously is never more than its capacity so that the capacity conflict can be avoided.

3.2. Zone control rules

Conditions (1)–(4) provide an effective way to check the conflict-freeness for a group of time-based trajectories, which are utilized to verify the trajectory planning results in Section 5. However, when a new aircraft is going to join the ground traffic, we need a more efficient approach to guide the search for a conflict-free trajectory for it. To this end, we handle the conflict avoidance problem through exit-ability and enter-ability determination for each zone. Four rules are presented in the following to describe the corresponding constraints in different scenarios.

Rule 1
(Exit-ability from an intersection).

If aircraft i enters intersection x at time τ_x^i within free time window $w_x \in W_x$, then we say it is exit-able from x if

$$TE\left(\hat{u}_x^i\right) < TE(w_x) \tag{5}$$

DOI: 10.1002/atr

where $\hat{u}_x^i$ is the minimum occupancy time window of aircraft i in intersection x. Let δ_x^i be the traverse time of aircraft i through x, then $\hat{u}_x^i$ can be calculated as follows:

$$\hat{u}_x^i = \left[\tau_x^i, \tau_x^i + \delta_x^i\right] \tag{6}$$

Notice here $\hat{u}_x^i$ is the occupancy time window corresponds to the ideal case that aircraft i moves through x and enters a neighboring zone unimpededly. However, it may actually encounter some interaction with aircraft in the neighboring zone, which would result in some delay in x and make the real occupancy time window u_x^i longer.

Rule 2
(Exit-ability from a lane).

If aircraft i enters lane x at time τ_x^i within free time window $w_x \in W_{x,d_x^i}$ ($d_x^i \in \{0,1\}$ is the direction in which i occupies x), then we say it is exit-able from x if the following conditions hold:

$$TE\left(\hat{u}_x^i\right) < r_{(k+1)} - \varepsilon \tag{7}$$

$$\begin{gathered} \forall f_x \in F\left(w_x\right), \\ \hat{u}_x^i \bigcap f_x = \varnothing \end{gathered} \tag{8}$$

where $\hat{u}_x^i$ is the minimum occupancy time window of aircraft i in lane x; a variable $r_{(m)} \in S(w_x)$ represents the release time (i.e., the end time of an occupancy time window) of index m among all the sorted release times in $S(w_x)$; the index k satisfies $o_{(k)} < \tau_x^i < o_{(k+1)}$, where $o_{(m)} \in S(w_x)$ is the occupancy time (i.e., the start time of an occupancy time window) of index m among all sorted the occupancy times in $S(w_x)$. Notice that because aircraft always occupy a lane in the FIFO manner, $o_{(m)}$ and $r_{(m)}$ are actually the start and end time of the same occupancy time window. In case aircraft i is the last to enter x during w_x, (7) reduces to (5). An illustration of a possible occupancy and release sequence (ORS) $S(w_x)$ is presented in Figure 4, which contains three occupancy and release times.

Using similar notations to equation (6), the aforementioned $\hat{u}_x^i$ can be calculated as

$$\hat{u}_x^i = \left[\tau_x^i, \max\left\{\tau_x^i + \delta_x^i, r_{(k)} + \varepsilon\right\}\right] \tag{9}$$

where the end time of $\hat{u}_x^i$ is chosen as the larger one between $\tau_x^i + \delta_x^i$ and $r_{(k)} + \varepsilon$ to ensure aircraft i would not overtake its leading aircraft. In case aircraft i is the first to enter x during w_x, (9) would reduce to (6).

Rule 3
(Enter-ability to a neighboring intersection).

If aircraft i can exit from zone x with exit time window e_x^i, then a neighboring intersection z is enterable during free time window $w_z \in W_z$ if

$$e_x^i \cap w_z \neq \varnothing \tag{10}$$

where the exit time window e_x^i can be calculated by

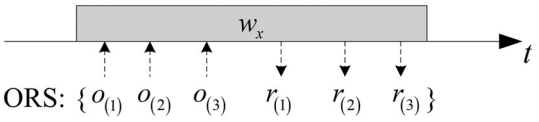

Figure 4. Illustration of the ORS.

DOI: 10.1002/atr

$$e_x^i = \left[TE\left(\hat{u}_x^i\right), TE(w_x)\right] \tag{11}$$

or

$$e_x^i = \left[TE\left(\hat{u}_x^i\right), r_{(k+1)} - \varepsilon\right] \tag{12}$$

depending on whether x is an intersection or a lane. Here, the notations have similar meanings to those in Rules 1 and 2.

In (12), the end time of e_x^i is constrained by the end time of the occupancy time window of the aircraft that enters lane x directly after aircraft i to ensure i would not be overtaken by its follower. In case i is the last to enter lane x during the corresponding free time window, (12) would reduce to (11).

Rule 4
(Enter-ability to a neighboring lane).

If aircraft i can exit from zone x with exit time window e_x^i, then a neighboring lane z is enterable during free time window $w_z \in W_{z,d_z^i}$ ($d_z^i \in \{0, 1\}$ is the direction in which i occupies z) if (10) holds and

$$\begin{gathered} \forall f_z \in F(w_z), \\ e_x^i \bigcap w_z \not\subset f_z \end{gathered} \tag{13}$$

Notice that in the taxiway model presented in Section 2, a lane is always connected to intersections at both ends. So, in Rule 4, x is actually an intersection. As a result, the exit time window e_x^i should be calculated using (11).

According to the definitions of the time window-related concepts in Table I, it is easy to verify that Rules 1–4 are in compliance with conditions (1)–(4) in the sense that if a time-based trajectory for a newly joined aircraft is generated by iteratively applying the enter-ability and exit-ability rules from the start position to the destination, conditions (1)–(4) will be met for all zones along the trajectory, which means the resulting trajectory is conflict-free with already assigned ones.

4. TIME-BASED TRAJECTORY PLANNING

In this section, we first present an efficient algorithm upon the refined taxiway model to find out the shortest time conflict-free taxi trajectory for an aircraft and discuss some key operations of the algorithm in detail. After that, we analyze the algorithm's computational complexity. Finally, we briefly introduce its integration with the sequential planning framework, which is implemented in Section 5 to investigate the computational performance of the algorithm.

4.1. Dynamic routing and timing algorithm

The A* algorithm [42] has been presented as a formal basis to find out the least-cost path from a single start vertex to a set of candidate destination vertexes on an implicitly described directed graph G. It can utilize problem-specific information to heuristically guide the search, avoiding unnecessary expansion of vertexes not on the optimal path. In the following, we present an A*-based dynamic routing and timing (DRT) algorithm running upon the hierarchical taxiway model to find out the shortest time conflict-free taxi trajectory for a newly joined aircraft using the zone control rules. To this end, we first introduce the concept of *temporal node* that will be utilized to describe our problem specific search graph.

A temporal node tn is a triple defined as

$$tn := \{z, n, w_z\},$$

where z is a zone in the taxiway partition model; n is a node lying on the border of z specifying the access point for an aircraft to enter zone z; and w_z specifies the free time window of z during which

DOI: 10.1002/atr

the aircraft enters z. For each temporal node, there is also a related enter-time τ_{tn} to describe when the aircraft enters the related zone.

Now we can define the successor operator Γ required by A* (see Section I-B of [42]) as such that for each temporal node $tn = \{z, n, w_z\}$, every accessible temporal node tn' related to neighboring zones of z is chosen as the successor of tn, and the corresponding time-cost $TC(tn, tn')$ of moving from the position specified by tn to the position specified by tn' is also determined; Or, we can equally define Γ as finding out each tn' and its enter-time $\tau_{tn'}$, because the time-cost between tn and tn' can then be calculated as $TC\left(tn, tn^{'}\right) = \tau_{tn'} - \tau_{tn}$. In this way, Γ can present an implicit description of a search graph G, where vertexes correspond to temporal nodes and edges represent the accessibility between them. If the definition of Γ guarantees that G captures all the potential routing patterns among temporal nodes, we can take advantage of A* to effectively find out the shortest time conflict-free trajectory as long as it exists.

In the following, we first complete the definition of Γ for our problem by clarifying how to determine an accessible temporal node and its enter-time. Then we provide the specification of the DRT algorithm based on A* and some additional considerations to improve its performance.

4.1.1. Accessibility determination

Let $tn = \{x, n, w_x\}$ be the temporal node selected for expansion and τ_{tn} be its enter-time. For each neighboring zone z that is on the outgoing branch of x with respect to tn, we can find out the node m through which aircraft i will move into z according to the taxiway partition model. With m and n, the traverse time δ_x^i can be estimated according to the average movement speed of the aircraft. Then, we can check the exit-ability from x using Rules 1 or 2 according to the type of x. If x is exit-able, we shall further examine the enter-ability of z using Rules 3 or 4, according to the type of z. For each free time window w_z satisfying the enter-ability requirements, we say the new temporal node $tn' = \{z, m, w_z\}$ is *accessible* from tn at the specified enter-time τ_{tn}, and tn' will be selected as a successor of tn.

Proposition 1
If z is an intersection, there may be more than one accessible temporal node. Otherwise (i.e., z is a lane), at most one temporal node could be accessible.

Proof
The intersection case is easy to verify. So in the following, we only prove for the lane case.

According to the taxiway partition model, x should be an intersection if z is a lane. Assume the newly joined aircraft i tries to enter z from x in the forward direction (proof is similar for the backward direction case), and $w_z \in W_{z,0}$ satisfies (10), that is, z is enterable during w_z.

If aircraft j is the last aircraft occupying z in the backward direction before w_z, then it should enter x after exiting from z. So, u_x^j will start at $TS(w_z)$. According to condition (1), the free time window w_x during which aircraft i enters x must have a start time not smaller than $TE\left(u_x^j\right)$; otherwise, (10) cannot be met. Thus, we have $TS(w_z) < TS\left(e_x^i\right) < TE(w_z)$ according to our assumption (notice that if no aircraft occupies z in the backward direction before w_z, the aforementioned relation still holds). Now we only need to further consider how far the end time of e_x^i extends.

If another aircraft k is the first to enter x at τ_x^k after the start time of w_x, then the end time of w_x would be determined by $TE(w_x) = \tau_x^k$ according to condition (1), which is also the end time of e_x^i. If k enters x from z, then k would also occupy z in the backward direction. Let $w_z^{'} \in W_{z,0}$ be the free time window directly after w_z, we have $TS\left(w_z^{'}\right) = TE\left(u_z^k\right) = \tau_x^k$. Because we also have $TE\left(e_x^i\right) = TE(w_x) = \tau_x^k$, (10) cannot be met for e_x^i and $w_z^{'}$, which means z is not enterable during $w_z^{'}$. It is apparent the free time windows in $W_{z,0}$ after $w_z^{'}$ are also not enterable. So, z may only be enterable for aircraft i during w_z. On the other hand, if k enters x from a zone different from z, we still have $TE\left(e_x^i\right) = TE(w_x) = \tau_x^k$. Because k is the first one to enter x after the start time of w_x, we can easily deduce $TE\left(e_x^i\right) < TS\left(w_z^{'}\right)$ from the

DOI: 10.1002/atr

earlier analysis, which means z is not enterable for aircraft i during $w_z^{'}$ as well as the free time windows in $W_{z,0}$ after $w_z^{'}$. So we again draw the conclusion that z may only be enterable during w_z. Finally, if no aircraft enters x after the start time of w_x, w_z will be the last free time window in $W_{z,0}$. So w_z will still be the only one that may be enterable for aircraft i.

4.1.2. Enter-time selection

Using the same notations as Section 4.1.1, we now consider how to select the enter-time (i.e., the time-cost) for temporal node tn', suppose it is accessible from tn. From an intuitive point of view, if there exists a time window during which we can select the enter-time $\tau_{tn'}$ freely, which is generally the case as explained in the following, we prefer the start time of that time window:

- When x is a lane, z should be an intersection according to our taxiway model. So the enter-time $\tau_{tn'}$ for tn' can be simply selected as

$$\tau_{tn'} = TS\left(e_x^i \cap w_z\right) \tag{14}$$

- When x is an intersection, z can be either an intersection or a lane. In the first case, $\tau_{tn'}$ will again be selected using (14). In the second case, if every full-capacity time window $f_z \in F(w_z)$ meets $f_z \cap e_x^i = \emptyset$, then $\tau_{tn'}$ can be selected using (14) too. Otherwise, if there exists a full-capacity time window satisfying $f_z \cap e_x^i \neq \emptyset$, the enter-time will be selected as

$$\tau_{tn'} = TE\left(f_z\right) \tag{15}$$

which is reasonable because of the following fact:

The accessibility from tn to tn' implies (13) is met, which together with the condition $f_z \cap e_x^i \neq \emptyset$ means either

$$TS\left(f_z\right) < TS\left(e_x^i \cap w_z\right) < TE\left(f_z\right) < TE\left(e_x^i \cap w_z\right)$$

or

$$TS\left(e_x^i \cap w_z\right) < TS\left(f_z\right) < TE\left(e_x^i \cap w_z\right) < TE\left(f_z\right).$$

Because during e_x^i, no other aircraft will move through x and enter z according to condition (1), the full-capacity time window f_z cannot start between $TS\left(e_x^i \cap w_z\right)$ and $TE\left(e_x^i \cap w_z\right)$. As a result, only the former case is possible. Because aircraft cannot enter z during its full-capacity time windows, the earliest feasible enter-time is $TE(f_z)$, indicating (15).

Proposition 2
Selecting the enter-time using (14) and (15) results in the most promising Γ.

Proof
First, selecting the enter-time using (14) and (15) guarantees the least time-cost to move from the position specified by the current temporal node tn to the position specified by a successive temporal node tn'. Because a feasible trajectory corresponds to a routing pattern among temporal nodes (related to different zones), the total time-cost for the trajectory will be the smallest. Second, because $\tau_{tn'}$ should be within a free time window of zone z, according to the earlier analysis, it is easy to verify that a smaller $\tau_{tn'}$ will result in a longer exit time window (if exit-able), which will then bring more opportunities to enter the neighboring zones of z than a shorter exit time window with the same end time (e.g., when we select a larger enter-time). So by iteratively applying the successor operator Γ, we are able to capture all the possible routing patterns among temporal nodes for the problem, which ensures existing conflict-free trajectories can be discovered by the algorithm.

DOI: 10.1002/atr

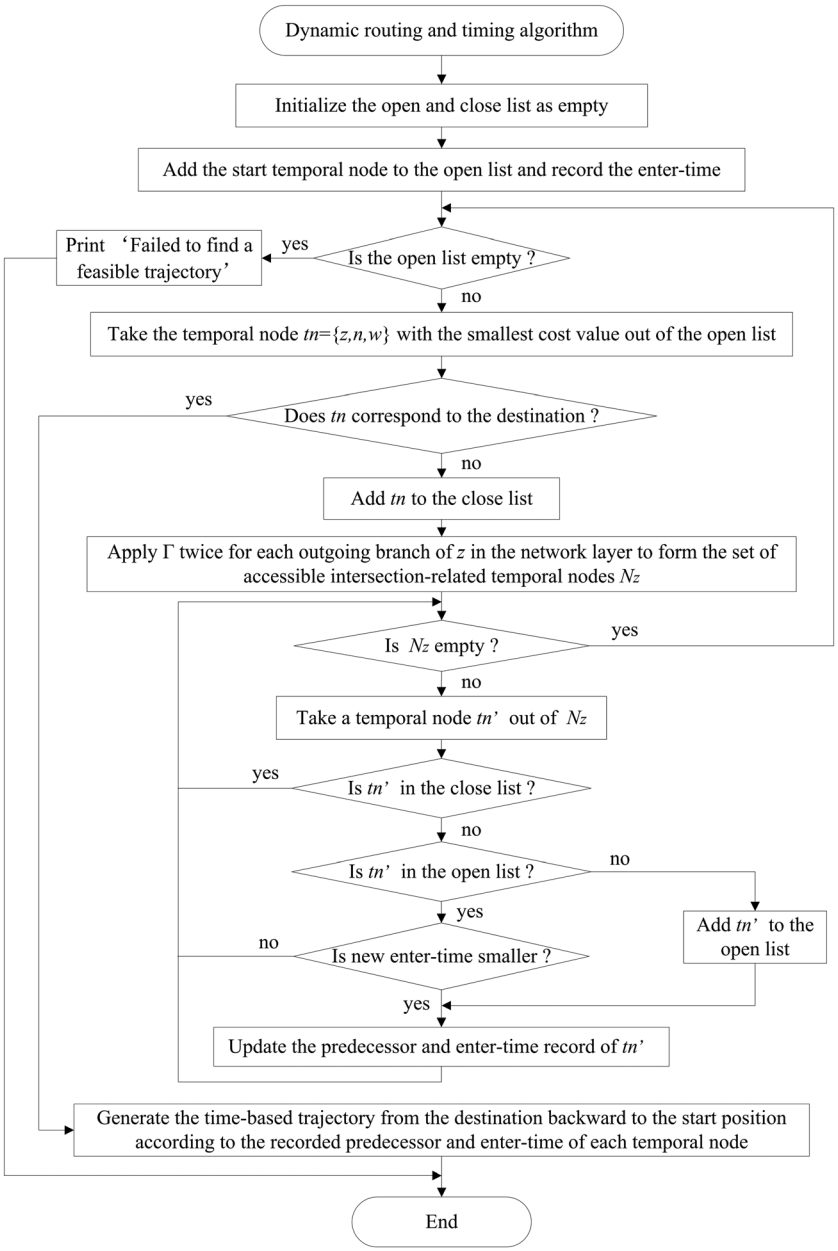

Figure 5. Flowchart of the dynamic routing and timing algorithm.

4.1.3. Algorithm specification

The flow chart of the DRT algorithm is presented in Figure 5. We use the close list to keep the already expanded temporal nodes and the open list to keep the explored but not yet expanded ones, in accordance with the terminology of A*.

As showed in Figure 5, the algorithm runs upon the network layer of the hierarchical taxiway model. After the start and target temporal nodes of current aircraft are determined from the input data (here we may assume they correspond to vertexes in the network layer), the search expands from the least-cost temporal node in the open list to its neighboring vertexes in the network model. This *conflict-free expansion* step is accomplished by applying Γ twice along each outgoing branch of the temporal node, in order to find out all the accessible temporal nodes (and the corresponding enter-time) related to a neighboring vertex in the network model (in case two intersections are directly connected, a dummy

DOI: 10.1002/atr

lane with no occupancy limitation is added between them). Meanwhile, the predecessor of each newly explored temporal node (including the intermediate temporal nodes related to lanes) will be recorded.

The aforementioned operations will repeat until the target temporal node is chosen for expansion or the open list becomes empty. The former case means a shortest time trajectory is found successfully, while the latter case implies a failure in finding a feasible trajectory. In the former case, the spatial trajectory is generated by retrieving the predecessor of each temporal node backward from the destination to the start position. And the enter-time of each temporal node is retrieved from the enter-time record to specify the timings of the trajectory.

Because the DRT algorithm is based on the principle of A* and the successor operator Γ is properly defined (as described in Section 4.1.2), it is guaranteed by A* that the existing shortest time route can be found if the heuristic adopted is admissible. Furthermore, if the heuristic is consistent, the presented algorithm can find out the existing shortest time conflict-free trajectory in an optimal way in the sense that no other admissible algorithm could expand fewer times [42,43].

Now we briefly discuss the considerations for formulating the algorithm in the aforementioned way:

First, in a related study by ter Mors *et al.* [36], a free time window graph is constructed before the route planning process: Each free time window of a resource (similar to the zone concept of this paper) is modeled as a vertex of the graph, then arcs of the graph are defined to represent the temporal connectivity between free time windows of spatially connected resources. However, as mentioned by the authors, accessibility to the free time windows of neighboring resources still need to be examined during each expansion, because the existence of an arc between two free time windows cannot ensure valid accessibility to the related neighboring resource; the accessibility also depends on the actual exit-time from previous resource. This has motivated the dynamic accessibility determination strategy in our algorithm, which uses an implicit representation of the search graph and completely shifts accessibility determination problem from the modeling stage to the expansion process.

Another disadvantage of the approach presented in [36] is the undiscriminating treatment of lanes and intersections during the search process. According to our earlier analysis, for a given temporal node, there may only be one enterable temporal node related to a neighboring lane, while there may be more than one enterable temporal node related to a neighboring intersection. So our algorithm takes the lane-related temporal nodes as intermediate ones by not putting them into the open list. In this way, the open list can be limited to contain only intersection-related temporal nodes. On the other hand, a lane-related temporal node may be accessible from different temporal nodes related to a neighboring intersection, but the open and close list mechanism can only preserve the one with the least cost. When allowing multiple occupancy of a lane, it will prevent other possible routing patterns among temporal nodes from being exploited after the lane-related temporal node comes into the close list. As a consequence, the algorithm may fail to find out the existing feasible trajectories (see the experimental results in Section 5.5).

4.2. Algorithm complexity

Let M denote the number of vertexes in the network layer of the hierarchical taxiway model. For each vertex, the number of neighboring vertexes should be no larger than a constant C for real world problems, which represents the maximal number of branches an intersection possesses. On the other hand, the number of free time windows depends on the already assigned taxi trajectories. Suppose N-1 aircraft have been assigned with a trajectory before the newly joined aircraft, then there would be at most N free time windows for each intersection if we do not allow repeated occupancy of a zone for the same trajectory. Thus, the upper bound on the number of the intersection-related temporal nodes is $C \times M \times N$.

Proposition 3

The worst-case complexity of the DRT algorithm is $O(M \times N \times \log(M \times N) + M \times N^2)$.

Proof

Because an intersection has at most C neighboring intersections in the network layer, the application of Γ to the least-cost temporal node in the open list examines at most C-1 outgoing branches to find out all the accessible lane-related temporal nodes. For each of those temporal nodes, the accessibility determination can be carried out in $O(\log N)$ time, because according to our analysis, at most one temporal node

DOI: 10.1002/atr

may be accessible for a lane, which can be found out using binary search. Then, for each branch, the determination of accessible temporal nodes related to the neighboring vertex from the connecting lane needs $O(N)$ time, for in the worst cases, the N temporal nodes of the vertex are all accessible. So in the worst case, a conflict-free expansion process needs $O(N)$ time.

On the other hand, every temporal node in the problem may be expanded once in the worst case, so the aforementioned expansion process runs at most $C \times M \times N$ times, which needs $O(M \times N^2)$ time in total. Besides, retrieving the minimal cost element from the open list can be implemented with $O(\log(M \times N))$ time complexity, so the total computation cost for this operation is $O(M \times N \times \log(M \times N))$ for it runs at most $C \times M \times N$ times. Because other operations can all be performed in constant time, the overall complexity of the algorithm is $O(M \times N \times \log(M \times N) + M \times N^2)$.

4.3. Sequential planning

The dynamic routing and timing algorithm can be integrated with the sequential planning framework [24,25] to determine feasible trajectories for a group of aircraft. Sequential planning simplifies the combined optimization of taxi routing and timing for a group of aircraft as a sequence of planning; each planning process determines a suitable trajectory for one aircraft, respecting the occupancy of zones by already assigned trajectories.

In the sequential planning framework, the time window constraints are updated after each trajectory assignment to indicate the new occupancy. A trajectory specifies the real occupancy time windows for an aircraft in related zones. For the affected intersections, we simply remove the occupancy time windows from the free time windows in a way similar to Algorithm 2 in [25]. For an affected lane, we first find out the direction in which it is occupied by the aircraft and remove the occupancy time window from the free time windows in the opposite direction. Then we update the ORS of the affected free time window and identify any full-capacity time window arising from this occupancy as follows:

Let u_z^i be the occupancy time window in lane z by the newly joined aircraft i. Presume $u_z^i \subseteq w_z$.

- First, add both $TS(u_z^i)$ and $TE(u_z^i)$ to $S(w_z)$.
- Second, walk through the subsequence formed by $TS(u_z^i)$, $TE(u_z^i)$, and all the time points in $S(w_z)$ between them. For every time interval between two consecutive time points in the subsequence, find out the largest occupancy and release time in $S(w_z)$ smaller than the end time of this interval, suppose p and q are their indexes in $S(w_z)$ among all the occupancy and release time, respectively. Calculate the overlap-value of this time interval as p minus q, which represents the number of aircraft occupying z during this time interval.
- Third, if any of the overlap-values equals c_z, add the related time interval to the full-capacity time window set $F(w_z)$.

Notice that the aforementioned approach also provides a way to realize the *MO* function used in condition (4): In the second step, instead of walking through the subsequence, we can walk through the entire ORS induced by a group of trajectories to find out the maximal overlap-value required by *MO*.

5. EXPERIMENTS AND RESULTS

In this section, we use simulations to investigate the computational performance of the presented DRT algorithm. The algorithm is implemented with the sequential planning framework to find taxi trajectories for a group of aircraft. Experiments are designed to find out the main influencing factors on the computational performance of the algorithm. Then the presented methodology is compared with existing approaches to investigate its advantages.

Table II. Selected airport models.

	Number of model elements			
Name	Ordinary zone	Stand	Runway	Node
Lukou	44	12	1	75
Heathrow	174	129	2	399
Pudong	260	186	3	591

DOI: 10.1002/atr

5.1. Implementation

We create three airport models of different complexity from real world airports to verify the computational performance of the DRT algorithm. The details of the models are listed in Table II.

The Lukou airport model has the simplest taxiway structure. It has only one runway located on the northern border. The taxiway structure of the Heathrow airport model is more complex. It has two parallel runways on the northern and southern borders, respectively. The Pudong airport model is the most complex of the three. It has three parallel runways, two of which lie on the eastern and western borders, respectively, with the third runway lying between them. There exist runway crossing situations in the experiment for the Pudong airport model. To deal with this problem, we use the same zone control rule as applied to the intersections to manage runway occupancy, which forces a runway to be occupied by only one aircraft at any time.

For each airport model, the data set of arrival and departure aircraft is generated randomly according to the runway and stand distribution. The planning priorities of the aircraft are determined by the starting time. Earlier starting time means higher planning priority. To investigate the relationship between planning priority and computation time, we generate 20 000 aircraft for each airport model, which is adequate to reveal the computational performance characteristics of the algorithm.

The algorithm is implemented using C++ in the Qt programming environment. All the experiments are run on a PC with 2 GB RAM and 2.70 GHz Inter Pentium Dual-Core CPU. In the following experiments, no infeasible solution is encountered for any aircraft, and the planning results are examined by a specific program using conditions (1)–(4). It proves that no conflict exists in the planning results.

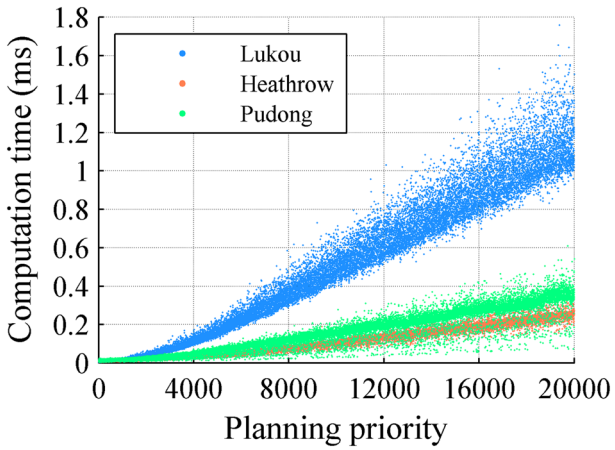

Figure 6. Average computation time for an expansion.

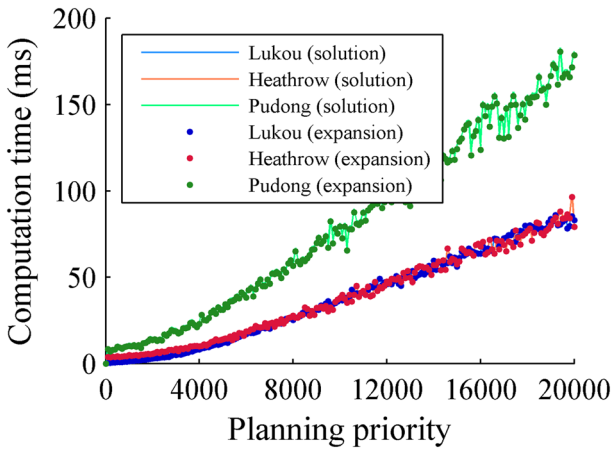

Figure 7. Average solution time and expansion computation time.

DOI: 10.1002/atr

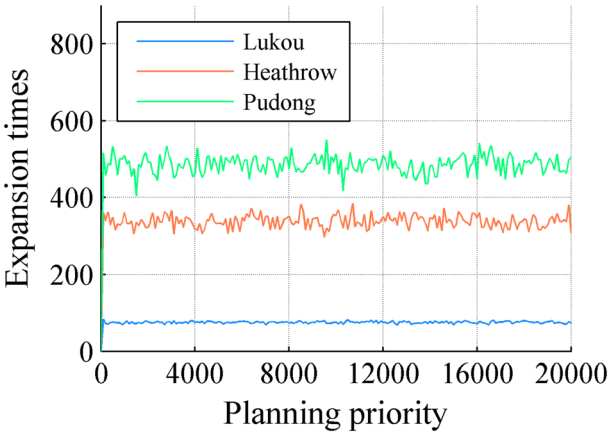

Figure 8. Average expansion times.

5.2. Computational performance

Figure 6 shows the average computation time for an expansion. When the planning priority (a larger number corresponds to a lower priority) becomes large enough (e.g., more than 4000), the computation time for an expansion shows a linear increase with the planning priority for any of the three airports. This result is consistent with the analysis in Section 4.2.

Figure 7 shows the relationship between the solution time and the planning priority. To reduce the impact of accidental factors on the solution time, we use average values of every 100 consecutively planned aircraft from the original data to draw the figure. The curves show a linear relationship between the planning priority and the solution time when the planning priority is large enough, instead of the quadratic relationship given in Section 4.2. This can be explained by the fact that the worst case computation complexity seldom occurs in reality; instead, the expansion times would not increase much with the planning priority as showed in Figure 8.

5.3. Investigation of heuristic measures

In the aforementioned experiment, the algorithm used unimpeded shortest distance (SD) as the heuristic measure for the remaining journey. In this section, another experiment is conducted to investigate the impact of heuristic measures on the solution time. In addition to the SD heuristic, we also use the Euclidean distance and the zero heuristic (i.e., all heuristic values are set to zero) for comparison. The expansion times of the Pudong data set with different heuristic measures are showed in Figure 9.

In Figure 9, the expansion times with SD and zero heuristics are nearly the same, while the Euclidean heuristic result shows an average smaller expansion times than the other two. Because the computation

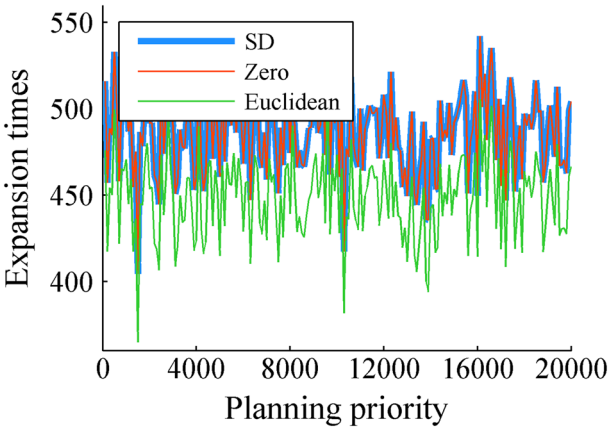

Figure 9. Comparison of expansion times with different heuristic measures.

DOI: 10.1002/atr

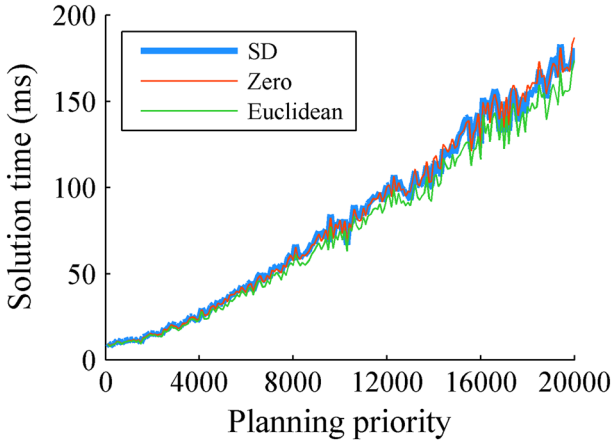

Figure 10. Comparison of solution time with different heuristic measures.

time for an expansion is basically the same with different heuristic measures, the solution time for an aircraft mainly depends on the expansion times. Figure 10 shows the average solution time with different heuristic measures. Although the differences of the solution time with the three heuristic measures are not much, the Euclidean heuristic shows an overall advantage over the other two. Besides, the resulting taxi routes and timings are technically the same, whichever of the three heuristic measures is used. Similar results are observed with the other two data sets.

5.4. Removal of expired time windows

The solution time for an aircraft depends on the expansion times and the average computation time for an expansion. From the aforementioned results, we know that the former mainly depends on the airport layout and the start and destination positions of the aircraft, tending to be statistically changeless during the whole planning process as showed in Figure 8. However, the latter is mainly determined by the number of time windows of each zone and would increase greatly with the planning priority number, resulting in increasing solution time for later planned aircraft. In fact, because the planning priority is determined by the starting time of each aircraft, the starting time of currently planned aircraft would become larger than the end time of more and more free time windows produced by earlier occupancies as the planning progresses. So we can infer that if these expired time windows are removed from the related time window sets, the solution time for later planned aircraft would greatly decrease. To this end, we can pause the planning process at some time and remove all the time windows with end times smaller than the starting time of current aircraft, and then start the planning process again.

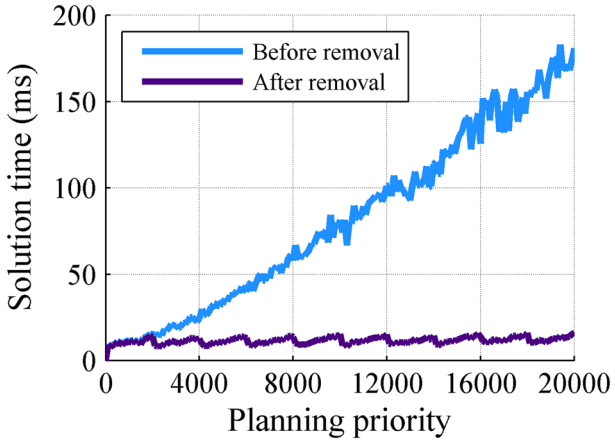

Figure 11. Solution time for each aircraft before and after expired time window removal.

DOI: 10.1002/atr

Figure 11 shows the comparison of the solution time on the Pudong data set using the SD heuristic measure before and after applying the time window removal strategy, which invokes a time window removal procedure after every 2000 aircraft. We can see that the solution time after removal is almost unaffected by the planning priority and keeps around 10 ms. The results with the other two data sets demonstrate similar characteristics. This result implies that the presented taxiway routing and timing algorithm is potential for continuous use in real-time airport surface movement management decision support systems.

5.5. Comparison with other methods

In this section, we compare the DRT algorithm with two sequential planning methods presented in related researches [25,36,44] to show its need and advantages.

First, we investigate the method presented in [36], which uses an A*-based search algorithm similar to the presented approach to find out the shortest time routes for transportation agents. Here, we adapt this algorithm to incorporate the concept of temporal nodes, with which it can carry out the conflict-free expansion according to the zone control rules presented in this paper. The difference with our approach is that it deals with zones identically during expansion, regardless of their types. In the following, we refer to this approach as the basic search (BS) algorithm.

Second, we compare the presented algorithm with the label-setting (LS) algorithm presented in [25,44]. The time interval of each label in the LS algorithm corresponds to an exit time window of our approach. So we can apply the label-setting algorithm to our model by setting labels for each node in the taxiway partition model. In order to meet the conflict-freeness conditions, the time intervals of new labels should be calculated in the same way as the exit time windows (i.e., using Rules 1 and 2 to determine the exit-ability and using (11) and (12) to calculate the exit time window). And the expansion of a label should be constrained by Rules 3 and 4, in order to ensure valid enter-ability for a neighboring zone, with the enter-time for each neighboring zone being selected using (14) and (15). Besides, the dominance principle [25,44] should also be adapted to distinguish labels of the same node corresponding to movements in different zones.

In the following, we compare the three approaches from the aspects of computational cost and search ability. The illustrated results are based upon the Pudong data set.

5.5.1. Computational cost

Figure 12 shows the comparison of the solution time. From the results, we can see that the LS algorithm presents a higher increase rate in the planning priority number compared with the DRT and BS algorithm, which is especially obvious when the planning priority number is large where the computation time of the LS algorithm becomes much larger than the other two algorithms.

The aforementioned results can be explained by the fact that the time complexity of the LS algorithm has a cubic relationship with the total number of time windows [44]. In contrast, the time

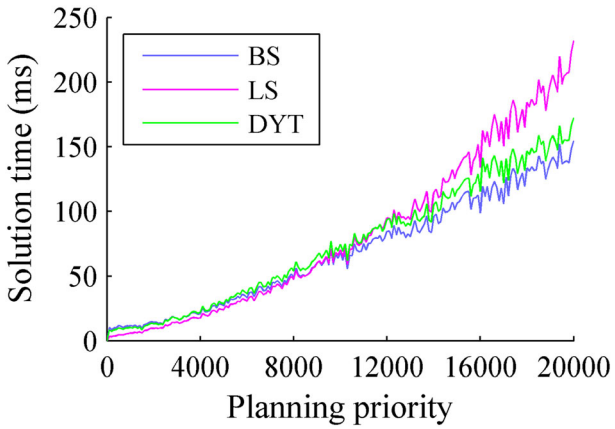

Figure 12. Comparison of solution time using different approaches.

DOI: 10.1002/atr

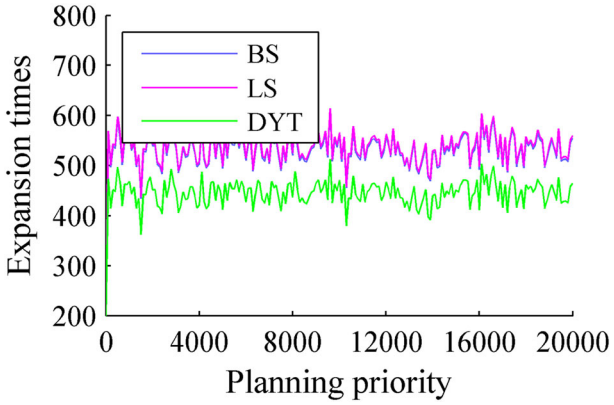

Figure 13. Comparison of expansion times using different approaches.

complexity of the presented DRT algorithm has a quadratic relation with the total number of time windows according to the analysis in Section 4.2, and so does the BS algorithm [36]. The results with the other two data sets present the same characteristics.

Figure 13 further explores the expansion times for each algorithm. Because the BS and LS algorithm are both based upon the taxiway partition model, they need similar expansion times throughout the experiment. On the other hand, the DRT algorithm shows a much smaller number of expansion times, for it runs upon the network layer of the hierarchical taxiway model. Despite the fact that the computation time of an expansion process of the DRT algorithm is expected to be

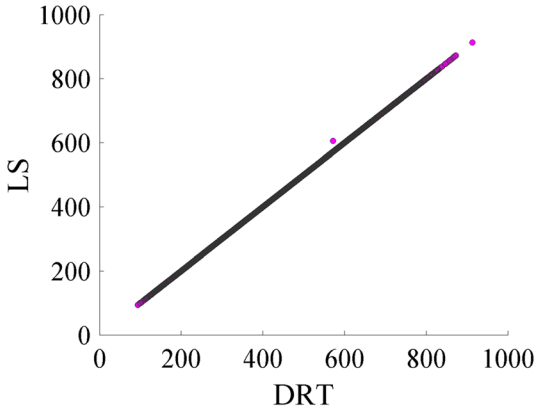

Figure 14. Planning results of dynamic routing and timing and label-setting.

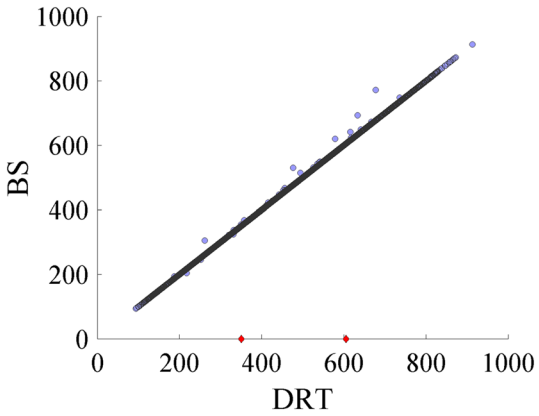

Figure 15. Planning results of dynamic routing and timing and basic search.

DOI: 10.1002/atr

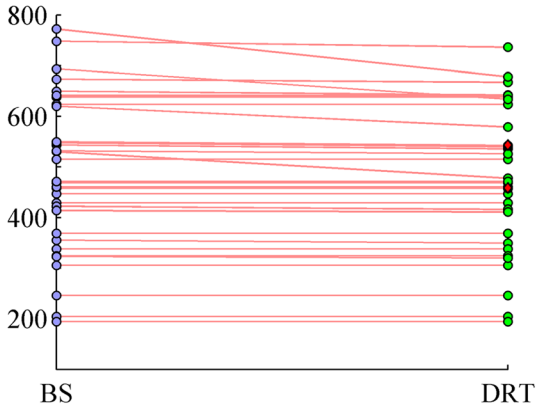

Figure 16. Comparison of planning results between dynamic routing and timing and basic search under the same time window constraints.

twice the cost as that of the BS algorithm, while the expansion times of the latter is apparently less than twice as many as the former, we notice that the actual solution time of both algorithms is very close.

5.5.2. Search ability

To compare the search ability for the shortest time trajectory, we display the planning results of the earlier experiment in Figures 14 and 15.

Figure 14 shows the results of the DRT and LS algorithm, plotted in a manner with the horizontal and vertical axes being the time-cost of the resulting trajectories for the DRT and LS algorithm, respectively. Both algorithms are able to find out a feasible trajectory for each aircraft. We can see that one data point lies out of the bisecting line, indicating equal time-cost for both algorithms. So we further examine this exceptional point by recalculating the trajectory using the DRT algorithm under the same time window constraints as the LS algorithm. This time, the results are identical for both algorithms. After a further examination of the time window constraints, we find that the previous difference is due to different time window constraints arising from trajectories assigned to previously planned aircraft. This is possible because the two algorithms may find different trajectories with the same time-cost. The results for the other two data sets are similar, although the numbers of different trajectories vary for different data sets.

As mentioned in Section 4.1, the BS algorithm may fail to find out existing trajectories because of the utilization of the ORS. This is observed in Figure 15, where the BS algorithm fails to find out existing trajectories for two aircraft (denoted by the red diamond symbol). We also notice that there are more points lying out of the bisecting line than in Figure 14. To further investigate the search ability of the two algorithms, we recalculate the trajectory using the DRT algorithm for each of the aforementioned aircraft (including the two cases where BS failed) under the same time window constraints as in the BS experiment. The results are presented in Figure 16, where the left and right data columns represent the results using BS and DRT, respectively. It is observed that DRT is able to find out feasible trajectories in the two cases where BS failed (the results are denoted again by the red diamond symbol). Additionally, we notice that under the same time window constraints, DRT can find better trajectories for several other aircraft. Analogous results are observed for the other two data sets.

6. CONCLUSION

This paper studies the taxiway routing and timing problem for surface trajectory-based operations. A new modeling approach is proposed to improve the modeling fidelity of taxiway structures. Based on a partition, the presented model reserves the inner structure of each zone, especially for intersections. This makes the spatial description of the taxi trajectories much more straightforward and precise compared against typical network models. This paper also puts forward a network model of a higher

DOI: 10.1002/atr

abstract level upon the partition model to simplify the taxiway structure description and reduce the expansion times of the presented dynamic routing and timing algorithm.

With the taxiway partition model, this paper uses zone control rules to avoid taxi conflicts between aircraft. It uses time window constraints to implement the zone control rules and extends the time window concepts to deal with the different characteristics of lanes and intersections. The dynamic routing and timing algorithm can be naturally implemented with the sequential planning framework to find the shortest time taxi trajectories for a group of aircraft in a predefined order. It respects the occupancy of each zone by aircraft with higher priorities via the time window constraints and makes sure that zone control rules are never violated by any aircraft during the planning stage.

Simulations are designed to examine the computational performance of the presented algorithm. The first experiment explores the computation time with different airport layouts and shows promising results. The second experiment investigates the influence of heuristic measures on the computation time. The result indicates that the Euclidean distance outperforms the other two heuristics, but the advantage is not significant. The aforementioned results also reveal the main factor affecting the solution time of the algorithm—the average computation time for an expansion, which depends on the number of free time windows of each zone. In order to further investigate this feature, the next experiment compares the solution time before and after the removal of expired time windows and demonstrates that the computation time can be greatly reduced with such time window removal when the algorithm is continuously used. The average several milliseconds solution time also makes it possible for the algorithm to be used for real time decision support. Finally, the presented DRT algorithm is compared with existing approaches to show its advantages. The comparison with the BS algorithm presented in [36] demonstrates that the DRT has a much better search ability for existing trajectories at the cost of only small extra computation time. It can explore some routing patterns omitted by the BS approach. As a result, the DRT algorithm can successfully find out a trajectory when BS fails, and it can find out better trajectories in some other cases. The comparison with the LS algorithm presented in [25,44] shows both algorithms have the same search ability that they can find out existing shortest time trajectories, but the DRT algorithm demonstrates a much smaller computation time when the number of time windows is large.

However, important remaining problems still need to be solved to make the proposed method available for realistic operations. Specific problems with regard to the presented algorithm include how to make the partition of taxiways more reasonable, and the interaction of aircraft movement and the determination of the optimal planning priorities should also be further studied to get higher overall efficiency.

ACKNOWLEDGEMENTS

This work was partially supported by the Technology Project of Civil Aviation Administration of China (No. MHRD201124), National Natural Science Foundation of China (No. 61203170), China Postdoctoral Science Foundation funded project (No. 2013T60539), and Innovation Found of Jiangsu Province (No. KYLX_0291). The authors wish to thank Ángel Marín of Universidad Politécnica de Madrid for useful discussions. The authors would also like to thank the anonymous reviewers for their comments on the paper.

REFERENCES

1. Cheng VHL. Research progress on an automation concept for surface operation with time-based trajectories. in *Proceedings of the Integrated Communications*, *Navigation and Surveillance Conference*, Herndon, VA, 30 April–3 May 2007. IEEE; 1-13. DOI: 10.1109/ICNSURV.2007.384166.
2. Diffenderfer PA, Morgan CE. Surface conformance monitoring in the NextGen timeframe. in *Proceedings of the Integrated Communications*, *Navigation and Surveillance Conference* (*ICNS*), Herndon, VA, 10–12 May 2011. IEEE; I7-1-I7-11. DOI: 10.1109/ICNSURV.2011.5935293.
3. Lee H, Simaiakis I, Balakrishnan H. A comparison of aircraft trajectory-based and aggregate queue-based control of airport taxi processes. in *Proceedings of the 29th Digital Avionics Systems Conference*, Salt Lake City, UT, 3–7 October 2010. IEEE; 1.B.3-1-1.B.3-15. DOI: 10.1109/DASC.2010.5655526.

4. Stelzer E, Stanley RM, Shepley KK. Evaluating surface trajectory-based operations concepts through a human-in-the-loop simulation. in *Proceedings of the 30th Digital Avionics Systems Conference*, Seattle, WA, 16–20 October 2011. IEEE; 3D2-1-3D2-12. DOI: 10.1109/DASC.2011.6096058.
5. Cheng VHL, Sweriduk GD. Trajectory design for aircraft taxi automation to benefit trajectory-based operations. in *Proceedings of the 7th Asian Control Conference*, Hong Kong, 27–29 August 2009. IEEE; 99–104.
6. Atkin JAD, Burke EK, Greenwood JS, Reeson D. Hybrid metaheuristics to aid runway scheduling at London Heathrow airport. *Transportation Science* 2007; **41**(1): 90–106. DOI:10.1287/trsc.1060.0163.
7. Neuman UM, Atkin JAD. Airport gate assignment considering ground movement. *Computational Logistics. Springer Berlin Heidelberg* 2013; 184–198. DOI:10.1007/978-3-642-41019-2_14.
8. Marín ÁG. Airport taxi planning: Lagrangian decomposition. *Journal of Advanced Transportation* 2011; **47**(4): 461–474. DOI:10.1002/atr.175.
9. Smeltink JW, Soomer MJ. An optimisation model for airport taxi scheduling. in *Proceedings of INFORMS Annual Meeting* 2004.
10. Marín ÁG. Airport management: taxi planning. *Annals of Operations Research* 2006; **143**(1): 191–202. DOI:10.1007/s10479-006-7381-2.
11. Justin M, Zachary W, Sivakumar R, Waqar M. A mixed integer linear program for solving a multiple route taxi scheduling problem. in *Proceedings of AIAA Guidance*, *Navigation*, *and Control Conference*, 2010.
12. Clare GL , Richards AG. Optimization of taxiway routing and runway scheduling. *IEEE Transactions on Intelligent Transportation Systems* 2011; **12**(4): 1000–1013. DOI:10.1109/TITS.2011.2131650.
13. Pesic B, Durand N, Alliot JM. Aircraft ground traffic optimisation using a genetic algorithm. in *Proceedings of Genetic and Evolutionary Computation Conference*, 2001.
14. Deau R, Gotteland JB, Durand N. Runways sequences and ground traffic optimisation. in *Proceedings of the Third International Conference on Research in Air Transportation*, 2008.
15. Deau R, Gotteland JB, Durand N. Airport surface management and runways scheduling. in *Proceedings of the 8th USA/Europe Air Traffic Management R&D Seminar*, 2009.
16. García J, Berlanga A, Molina JM, Casar JR. Optimization of airport ground operations integrating genetic and dynamic flow management algorithms. *AI Communications* 2005; **18**(2): 143–164.
17. Gotteland JB, Durand N. Genetic algorithms applied to airport ground traffic optimization. in *Proceedings of the 2003 Congress on Evolutionary Computation*, 2003. 544–551. DOI: 10.1109/CEC.2003.1299623.
18. Gotteland JB, Durand N, Alliot JM. Handling CFMU slots in busy airports. in *Proceedings of the 5th USA/Europe Air Traffic Management R&D Seminar*, 2003.
19. Gotteland JB, Durand N, Alliot JM, Page E. Aircraft ground traffic optimization.in *Proceedings of the 4th USA/Europe Air Traffic Management R&D Seminar*, 2001. 04–07.
20. Herrero JG, Berlanga A, Molina JM, Casar JR. Methods for operations planning in airport decision support systems. *Applied Intelligence* 2005; **22**(3): 183–206. DOI:10.1007/s10791-005-6618-z.
21. Atkin JA, Burke EK, Ravizza S. The airport ground movement problem: past and current research and future directions. in *Proceedings of the 4th International Conference on Research in Air Transportation*, Budapest, Hungary, 1–4 June 2010. 131–138.
22. Roling PC, Visser HG. Optimal airport surface traffic planning using mixed-integer linear programming. *International Journal of Aerospace Engineering* 2008; **2008**(1): 1–11. DOI:10.1155/2008/732828.
23. Schüpbach K, Zenklusen R. Approximation algorithms for conflict-free vehicle routing. in *Algorithms–ESA 2011*. Springer Berlin Heidelberg: Berlin, 2011; 640–651. DOI: 10.1007/978-3-642-23719-5_54.
24. Lesire C. Iterative planning of airport ground movements. in *Proceedings of the 4th International Conference on Research in Air Transportation*, Budapest, Hungary, 2010. 147-154.
25. Ravizza S, Atkin JA, Burke EK. A more realistic approach for airport ground movement optimisation with stand holding. *Journal of Scheduling* 2014; **17**(5): 507–520. DOI:10.1007/s10951-013-0323-3.
26. Ravizza S, Atkin JAD, Maathuis MH, Burke EK. A combined statistical approach and ground movement model for improving taxi time estimations at airports. *Journal of the Operational Research Society* 2012; **64**(9): 1347–1360. DOI:10.1057/jors.2012.123.
27. Ravizza S, Chen J, Atkin JAD, Stewart P, Burke EK. Aircraft taxi time prediction: comparisons and insights. *Applied Soft Computing* 2014; **14**(2014): 397–406. DOI:10.1016/j.asoc.2013.10.004.
28. Atkin JAD, Burke EK, Greenwood JS. A comparison of two methods for reducing take-off delay at London Heathrow airport. *Journal of Scheduling* 2011; **14**(5): 409–421. DOI:10.1007/s10951-011-0228-y.
29. Atkin JAD, De Maere G, Burke EK, Greenwood JS. Addressing the pushback time allocation problem at Heathrow airport. *Transportation Science* 2013; **47**(4): 584–602. DOI:10.1287/trsc.1120.0446.
30. Burgain P, Feron E, Clarke JP. Collaborative virtual queue: benefit analysis of a collaborative decision making concept applied to congested airport departure operations. *Air Traffic Control Quarterly* 2009; **17**(2): 195–222.
31. Chen J, Stewart P. *Planning aircraft taxiing trajectories via a multi-ojective immune optimisation*. in *Natural Computation* (*ICNC*), *2011 Seventh International Conference on*, Shanghai, 26-28 July, 2011. IEEE; 2235 - 2240. DOI: 10.1109/ICNC.2011.6022587.
32. Ravizza S, Chen J, Atkin JAD, Burke EK, Stewart P. The trade-off between taxi time and fuel consumption in airport ground movement. *Public Transport* 2013; **5**(1-2): 25–40. DOI:10.1007/s12469-013-0060-1.
33. ter Mors A, Witteveen C. Plan repair in conflict-free routing. in *Next-Generation Applied Intelligence*. Springer Berlin Heidelberg: Berlin, 2009; 46–55. DOI: 10.1007/978-3-642-02568-6_5.

34. Pschierer C, Sindlinger A, Schiefele J. Standardization of databases for AMDB taxi routing functions. in *Proceedings of International Society for Optics and Photonics on SPIE 7689*, *Enhanced and Synthetic Vision*, *768905-768905-9*, 2010. DOI: 10.1117/12.850112.
35. Tang X, An H, Wang C. Conflict-avoidance-oriented airport surface-taxiing guidance lights system model. *Journal of Guidance, Control, and Dynamics* 2012; **35**(2): 674–681. DOI:10.2514/1.54693.
36. ter Mors A, Witteveen C, Zutt J, Kuipers FA. Context-aware route planning. in *Multiagent System Technologies*. Springer Berlin Heidelberg: 2010; 138–149. DOI: 10.1007/978-3-642-16178-0_14.
37. Kunzi F. Reduction of collisions between aircraft and surface vehicles. *Transportation Research Record: Journal of the Transportation Research Board* 2013; **2325**(1): 56–62. DOI:10.3141/2325-06.
38. Landry SJ, Chen XW, Nof SY. A decision support methodology for dynamic taxiway and runway conflict prevention. *Decision Support Systems* 2013; **55**(1): 165–174. DOI:10.1016/j.dss.2013.01.016.
39. Smolic-Rocak N, Bogdan S, Kovacic Z, Petrovic T. Time windows based dynamic routing in multi-AGV systems. *IEEE Transactions on Automation Science and Engineering* 2010; **7**(1): 151–155. DOI:10.1109/TASE.2009.2016350.
40. Fanti MP. Event-based controller to avoid deadlock and collisions in zone-control AGVS. *International Journal of Production Research* 2002; **40**(6): 1453–1478. DOI:10.1080/00207540110118073.
41. Ho YC, Liao TW. Zone design and control for vehicle collision prevention and load balancing in a zone control AGV system. *Computers & Industrial Engineering* 2009; **56**(1): 417–-432. DOI: 10.1016/j.cie.2008.07.007.
42. Hart PE, Nilsson NJ, Raphael B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics* 1968; **4**(2): 100–107. DOI:10.1109/TSSC.1968.300136.
43. ter Mors A. The world according to MARP: multi-agent route planning. *PhD Dissertation*, Delft University of Technology, 2010.
44. Stenzel B. Online disjoint vehicle routing with application to AGV routing. *PhD Dissertation*, TU Berlin, 2008.