Computers & Industrial Engineering 93 (2016) 171–191

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

MIP-based heuristics for solving robust gate assignment problems

Chuhang Yu, Dong Zhang *, H.Y.K. Lau

Department of Industrial and Manufacturing Systems Engineering, The University of Hong Kong, Hong Kong

ARTICLE INFO

Article history:
Received 17 January 2015
Received in revised form 15 December 2015
Accepted 16 December 2015
Available online 21 December 2015

Keywords:
Gate assignment problem
Robustness
Linear transformation
MIP-based heuristics

ABSTRACT

This paper considers the problem of robust gate assignment. Three factors having significant impact on gate assignment are considered: schedule robustness, facility and personnel cost during tows, and passenger satisfaction level. To precisely evaluate passenger satisfaction level, especially for transfer passengers, a model with quadratic terms is formulated. The quadratic model can exactly represent the traveling distance of transfer passengers, which cannot be precisely considered by traditional approximate models. However, the quadratic model is rather difficult to solve. Therefore, we then transform the model to an equivalent MIP model which is proven to be more efficient than the linearized models proposed in previous literature. In addition, in order to handle large size instances, we developed four different algorithms including diving, local branching, and relaxation induced neighborhoods (RINS), which are popular algorithms for solving general MIP models, together with a new algorithm which hybridizes the strength of RINS and diving. Extensive experiments are performed to compare the performance of the proposed algorithms.

1. Introduction

During the past thirty years, the volume of air transport has almost doubled, which makes major international airports, especially those hub airports congested. There are three major scarce resources in a typical hub airport: runway, taxiway and gate, which cannot be increased in a short timeframe due to physical constraints, government legislation as well as huge investment. Among these three resources, the assignment of gates is the toughest one in hub airports in Asia, as compared with those in US. Table 1 lists six of the ten largest airports in the world; three of them are in US while the other three are in Asia. The last column indicates that on average the airlines take fewer passengers per movement in US than those in Asia. As the first column indicates that they serve comparable number of passengers annually, we can conclude that US airlines use smaller aircrafts and provide more frequent flights with much shorter turn-around time, while Asia airlines use larger aircrafts and stay at the hub airport for a relative longer time. The reason may lie in that if we consider the ratio of domestic flight to the number of international flight in North America, we can see that airlines operated in Asia have a much higher percentage of international flight, than by the US. International flights are flown using larger aircraft and in general, required longer parking time at an airport due to the service schedule. As a result, in US, the large hub airports usually have more than 3 or 4 runway on average. However, these runways are usually crowded and used to its maximum capacity. On the other hand, the longer turn-around time forces an aircraft in Asia to occupy longer time at gates, which makes the gate resources a scarce resource. In this paper, we focus on the problem of gate shortage faced by the Asia airports. Due to the long parking time of some aircrafts in Asia hub airports, towing aircrafts to remote gates may be necessary to leave more space for incoming aircrafts.

The gate assignment problem (GAP) is a scheduling problem that schedules all the activities of the aircraft at an airport, including arrival, departure and parking, to satisfy the related constraints. Typically, the arrival and departure activities of the same aircraft are assigned to the same gate. However, in some situations, the buffer time between arrival and departure is very long, we treat this time duration as parking time. The aircraft may be assigned to a different gate or parking apron area during parking time, to leave convenient gate for other aircraft to embarking and disembarking passengers. In this paper, the arrival, departure and parking activities of an aircraft are separately treated to efficiently utilize the gate resource. That means, we may tow the arrival aircraft to another gate to leave space for incoming aircraft, if the parking time is long enough. Considering the movement of aircrafts will cause additional costs and inconvenience to the facility and other aircraft. Therefore, it is necessary to minimize the tow frequency.

* Corresponding author. Tel.: +852 95190366.
E-mail addresses: ych1102@gmail.com (C. Yu), zd.pony@gmail.com (D. Zhang), hyklau@hku.hk (H.Y.K. Lau).

http://dx.doi.org/10.1016/j.cie.2015.12.013
0360-8352/

Table 1
Air transportation data comparison between USA and Asia (Belobaba, Odoni, & Barnhart, 2009).

	Annual passengers (million)	Annual aircraft movement (thousand)	Passenger per movement
Atlanta	89.4	994	90
Chicago	76.2	928	82
Tokyo	66.7	332	201
Los Angles	61.9	681	91
Beijing	53.7	400	134
Hong Kong	47	305	154

Disturbances always occur during the flight schedule operation. They may come from adverse weather, system malfunction, or other unexpected events. If the arrival flights come earlier or departure flights leave later than the original schedule, there will be gate conflict. The gate conflict time is defined as the waiting time of arrival flight until the occupying aircraft leaves the gate. Robustness of the gate assignment is defined as the capability of the schedule to absorb the minor disturbances and avoid long gate conflict. There are various ways to measure the robustness of a gate schedule, such as minimization of the number of buffer time with duration under a predefined value, and minimization of the gap between the maximum slack time and the minimum slack time to make the buffer time evenly distributed. One of the most widely adopted assumption is that the flight delay are uniformly distributed and therefore the gate conflict is linearly related to the buffer time. However, in reality, the delay time distribution is more like a log-normal distribution. Moreover, the occurrence of delay may be related to the local weather condition, airport infrastructure such as runway and taxiway condition and capacity of the airport. To make better robust schedule for the specific target airport, the robust schedule can be generated based on its historical flight data. In this paper, we analyze the operation statistics in Hong Kong International Airport (HKIA) and to show that the flight delay is close to log-normal distribution. Based on this flight delay distribution, the expected gate conflict is approximately exponential to the scheduled buffer time between successive flights being assigned the same gates. The assignment robustness, which is measured by the expected gate conflict, is maximized by appropriately scheduling the buffer time and assigning the parking activities to parking apron to leave more free space. However, the movement of aircraft at the airport due to gate change, which is called tow cost, will lead to additional costs to airport. Therefore, there is a trade-off between the tow cost and schedule robustness.

Among the passengers, the transfer passengers are the most sensitive to the traveling time at the airport, therefore, the transfer traveling time is also included in this paper. To consider the transfer time of transfer passengers, the walking distance from the arrival gate to departure gate is used as a measurement index. As both the arrival gate and departure gate are unknown, the gate assignment problem or GAP is modeled as a quadratic model in this study. The long computation time induced by the quadratic objective makes the model impractical. To avoid the extensive computation time, some previous research approximated the distance by using the transfer distance as the average distance from one gate to every other gate. However, the deviation of the result from the exact solution is not negligible according to our computational experiments. In this study, we successfully transformed the quadratic model into an equivalent Mixed Integer Programming (MIP) model, which improves the solving speed extensively. To solve the larger size problems, we further adapt the-state-of-art general MIP solving methods, as well as propose a math-heuristic named Variable Reduced Neighborhood Search (VRNS) which exploits the strengths of current existed MIP-based heuristics. In the computational experiments, we compare these algorithms and analyze the suitable algorithms for each data type.

Therefore, the main contributions of our study are as follows:

(1) The quadratic formulation is transformed to a novel equivalent mix-integer programming formulation, which make the general MIP solving methods applicable to solve our GAP.
(2) This study is the first one to adopt three general MIP solving methods, namely relaxation induced neighborhoods (RINS), diving and local branching to solve gate assignment problems. In addition, a new math-heuristic variable reduce neighborhood search (VRNS) is proposed based on RINS and diving and integrate their advantages to solve the multi-objective GAP. Comparisons are made between the methods for solving GAP.
(3) Managerial implications are provided by analyzing numerical results

This paper is organized as follows: Section 2 gives a literature review on current research of gate assignment problem and related methodologies. In Section 3, we formally define the problem and reformulate it as an MIP model based on the quadratic one. In Section 4, we present the algorithms for solving the multi-objective gate assignment problem. Section 5 discusses the computational results and Section 6 gives the conclusion.

2. Literature review

There are extensive previous research related to gate assignment problems. Firstly, the previous problem and models are presented as well as their solution methods in Section 2.1. However, after linear transformation, the traditional methods are no longer suitable for solving our problem. We suggest solving the model by general MIP solving methods. The literature review related to the popular general MIP solving methods are shown in Section 2.2.

2.1. Gate assignment problems and previous solution methods

We categorize the previous research in the gate assignment problem into three categories according to three different optimizing perspectives: airport, airline and passenger.

2.1.1. Passenger

Passenger satisfaction is one of the earliest research objectives considered in literature, as it is one of the most important factors that have impact on the airport revenue. Passenger satisfaction is commonly measured by three indices: walking distance for arrival and departure passengers, transfer walking distance, and waiting time for flight connection. Babic, Teodorovic, and Tošic (1984) is among the first to develop models to minimize the passenger walking distance. In their work, only departure and arrival passengers are considered. They proposed a backtracking version branch and bound with an acceleration constraint to solve this problem. A small instance consists of 5 gates and 9 airplanes are tested. For transfer passengers, the arrival gate and departure gate are both to be decided, making the problem intractable. Based on Babic et al. (1984), Mangoubi and Mathaisel (1985) included transfer passengers. To simplify the problem, the transfer passengers walking distances are determined from a uniform probability distribution of all integrate walking distances. Bihr (1990) solved an approximate problem by fixing arrivals for transfer passenger so as to simplify the formulation as an 0–1 linear programming. Braaksma (1977), Zhang, Cesarone, and Miller (1994) also worked on passenger walking distances. One of the first paper modeling the transfer distance as quadratic objective was (Xu & Bailey,

2001), which formulated the problem as a quadratic assignment problem and reformulated it as 0–1 integer program. A Tabu Search (TS) was proposed to solve the problem. Passenger waiting time was regarded as the objective in Yan and Tang (2007), which considered the stochastic situations. A heuristic is developed that imbedded in planning and real-time stages. Yan and Huo (2001) considered both walking distance and waiting time for transfer passengers, however, to simplify the model, they approximate transfer distance by using the average distance assuming a uniform distribution, and formulated the problem as 0–1 integer program. They used weighting method, column generation, simplex method and branch and bound. Apart from these two objectives, Hu and Di Paolo (2007) added the baggage transport distance into the objectives. The authors proposed a genetic algorithm (GA) which used relative position between aircrafts as chromosomes, and they implemented a new uniform crossover to avoid infeasible chromosomes. In this research, we use quadratic exact expression to optimize the transfer distances and show in experimental results that it is meaningful and much more representative than the approximate linear expressions. Moreover, we transform the quadratic programming model into an equivalent MIP model. The major difference between our model and the one proposed by Xu and Bailey (2001) lies in that we largely reduced the number of decision variables and improve the solution speed. The comparison will be made in the computational experiments.

2.1.2. Airline

For airlines, there are two primary objectives: (1) arriving at and departing from the preferred gate and (2) the level of on-time operation. Flight-gate preference usually comes with other objectives. Drexl and Nikulin (2008) proposed a simulated annealing algorithm to solve the multi-objective which is composed of minimizing un-gated flights, minimizing total passenger walking distance, maximizing flight-gate preference scores. Dorndorf (2002) also considered a multi-criteria problem, one of the objectives is to maximize flight-gate preference. Apron gates are usually the last option to be assigned and are always regarded as special dummy gates in which there are no time overlapping restrictions, examples could be seen in Drexl and Nikulin (2008), Ding Lim, and Rodrigues (2004a, 2004b) and Ding, Lim, Rodrigues, and Zhu (2005), these flights are regarded as unassigned or un-gated. Flight delay or cancelation will lead to great loss to airlines, so on-time operation is rather important for airlines. Therefore, more of recent attention is focused on this issue. A stochastic uncertainty occurred on an aircraft will not only affect the downstream flights of this aircraft, but also the following aircraft assigned to the gate. A pioneering paper that addressed flight delay is by Wirasinghe and Bandara (1990). In this work, a closed-form solution is provided for obtaining the optimum number of gates given the peaking of the aircraft arrival rate. Yan and Tang (2007) proposed to integrate assignments and reassignments in the planning stage and developed a heuristic approach to deal with the model. Another approach to deal with delay issue is to make robust schedules in the planning period. In this case, the objective is to distribute buffer time among flights assigned to the same gate in expectation of less overlapping occurrence. Hassounah and Steuart (1993) planned buffer time and found it can improve the punctuality. Minimization of maximum slack time and minimum slack time are used as objective to make the buffer time evenly distributed (Bolat, 1999, 2000). A branch and bound algorithm is proposed by them to solve small size problem and a heuristic to solve large problems. Dorndorf et al. tried to make robust schedule by minimizing the number of buffer time with duration under a predefined value (Dorndorf, Jaehn, & Pesch, 2008). Recently, Kim and Feron (2013) and Kim, Feron, Clarke, and Marzuoli (2013) maximized the robustness of gate assignments by minimizing the duration of gate conflict. Another two objectives, passenger transit time and aircraft taxi time are also considered. A Tabu search algorithm is adopted to solve this problem. In this paper, we try to minimize the gate conflict caused by log-normal distributed flight delay.

2.1.3. Airport

The owner or manager of airports usually concern the facility and personnel cost such as towing cost. A towing occurs when the aircraft changes its position between different gates. Jo, Jung, and Yang (1997) tried to minimize the tow by assigning the arrival and departure of the aircraft to the same gate. The tow cost in their work is measured by the tow time between gates. Two heuristics were deployed: one is to first assign the best-fit flight to the available time intervals; the other is to first assign the larger size aircraft. Dorndorf (2002) and Dorndorf et al. (2008) measured tow cost by calculating the total number of the tow activity. The authors then transforms the problem, which considered three objectives, into an equivalent Clique Partitioning Problem and solves it using an ejection chain algorithm. In this paper, in order to efficiently utilize the scarce gate resources, we separately treat the arrival, parking and departure activities of an aircraft at the airport when assign a gate. To minimize the inconvenience and cost brought by the movement of aircraft to the facility and staff, we try to minimize the tow frequency. In the computational study, we analyze the trade-off relationship between gate conflicts and tow frequency.

In this paper, we model the gate assignment problem from all three perspectives mentioned above with weighting parameters to reflect the relative cost related to each perspective. The aim is to minimize the total cost with respect to all three aspects. Specifically, we consider tow cost, robustness and walking distance in the gate assignment problem. These factors are incorporated which indicate the benefit and cost of airports, airlines and passengers respectively.

2.2. Popular MIP-based heuristic approaches for general MIP problems

In this paper, in order to improve the solving speed, we transform the quadratic model into an equivalent MIP model. The structure of this MIP model is not appropriate to apply the previous algorithms designed for gate assignment problems. Therefore, we propose to adopt the MIP-based heuristics which are designed to solve general MIP models for solving our GAP.

MIP problems are typically solved using branch-and-bound or branch-and-cut algorithms. In the branch-and-bound scheme, a tree of continuous relaxations of the original MIP model is iteratively explored. At each node in the tree, two children nodes are built by imposing corresponding complementary bounds on an integer variable. Based on this concept, CPLEX is a state-of-the-art commercial software which is capable of solving MIP problems (Ibm, 2010). Although general MIP problems have NP-hard nature, many small-size MIP problems can be solved to optimality effectively using CPLEX. However, when the problem size grows larger, they are hard to be solved by CPLEX with reasonable time. In our problem, the problem with size larger than 40 cannot be solved within 2 h. For some hard instances, even a feasible solution is intractable. Recently, more and more research are focusing on designing local search heuristics exploiting the strong searching ability of CPLEX as black-box solver. These sub-problems are controlled by an external framework.

The local search approaches for general MIP problems could be divided into two branches: feasibility heuristic for finding feasible solutions for tight-constrained hard MIP and refining heuristic for improving current solutions. Feasibility Pump (FP), which was proposed by Fischetti et al., turns out to be very useful in finding a first feasible solution for hard problems (Fischetti, Glover, & Lodi, 2005).

The main idea of FP is to maintain two solutions, one $\tilde{x}$ is an integer but not feasible, the other $\bar{x}$ is a feasible solution to the linear relaxation of the original problem. Two solutions are updated iteratively until they become the same. Due to its effectiveness, more researches devoted to its further improvements. Based on their own work, Bertacco, Fischetti, and Lodi (2007) extended FP to solve both binary and general integer MIP problems and to drive a subsequent enumeration phase. A more clever rounding heuristic based on diving-like procedure and constraint propagation is proposed in Fischetti and Salvagnin (2009). Achterberg and Berthold (2007) improved the quality of first feasible solution obtained by considering the original objectives in addition to the distance between two solutions. The Feasibility Pump has been further extended to the nonlinear and nonconvex cases respectively in Bonami et al. (2009) and D'Ambrosio, Frangioni, Liberti, and Lodi (2012).

Recent popular local search heuristic for improving current solutions include local branching (LB), relaxation induced neighborhoods (RINS), and diving heuristics, among others. Local branching (Fischetti & Lodi, 2003) was proposed by Fischetti and Lodi to find good solutions early during the exact tree search. The main idea is to constrain the distance from candidate solution to the imcunbent solution, so as to reduce the searching space. Based on their own work, Fischetti, Polo, and Scantamburlo (2004) applied LB to solve telecommunication network design problems, which have the specific structure that the set of binary variables partitions naturally into two levels. A variable neighborhood search local branching in which the distance is allowed to vary in Hansen, Mladenović, and Urošević (2006). Rodríguez-Martín and Salazar-González (2010) implemented LB to solve the classical capacitated fixed-charge network design problem. The feasible solution obtained by Feasibility Pump is used by Fischetti and Lodi to serve as an initial solution to the subsequent LB procedure (Fischetti & Lodi, 2008).

RINS (Danna, Rothberg, & Le Pape, 2005) was proposed by Danna et al. in the hope of building good neighborhood by exploiting the information contained in the continuous relaxation and incumbent feasible solutions. A similar approach, Distance Induced Neighborhood Search (DINS) Ghosh (2007) is proposed by Ghosh, in which soft fixing is adopted rather than hard fixing. A diving heuristic is a depth-first heuristic search. The main idea of the diving heuristic is to round some values obtained by solving linear relaxation to integer and solve MIP recursively.

In this paper, in order to precisely describe the transfer passengers' satisfaction level, improve robustness of obtained schedule and reduce airport personnel cost, a network flow model with quadratic objectives is formulated. Based on this quadratic formulation, we proposed an equivalent MIP model. To our knowledge, only one research has transformed the quadratic objective into a linear one for GAP, which is proposed by Xu and Bailey (2001). The linearized model we proposed is different from the one proposed by Xu and Bailey that we introduce an auxiliary continuous variable ξ and largely reduce the number of binary decision variables and auxiliary constraints. Finally, to solve practical-size problems, we proposed an MIP-based heuristic, which uses the Commercial MIP solver CPLEX as a black-box to solve the problem. After linear transformation, the characteristic of our model is quite different from current scheduling models and it is hard to use current scheduling algorithms or traditional GAP algorithms to solve it. RINS, diving and local branching are very popular MIP-based heuristics for solving the general MIP models, especially those NP-hard ones. We expect they can achieve good performance in solving this gate assignment model. The proposed local search heuristic VRNS in this paper is inspired by RINS and diving heuristic. RINS, diving, and local branching are also adapted to solve our GAP. The comparison as well as the analysis between them is made in the computational study.

3. Problem formulation and models development

In general terms, the gate assignment problem (GAP) is to assign a gate for a set of aircraft activities. The activities here include arrival, parking and departure of an aircraft in an airport. The arrival and departure are also called flights. The parking means the duration between the arrival and departure. Fig. 1 illustrates the three activities of an aircraft at an airport. Suppose the aircraft arrives at a gate at 10:10. The time required for connecting the bridge, disembarking passengers and cabin cleaning is 30 min. Then this arrival activity starts from 10:10 and ends at 10:40. Instead of starting the departure flight immediately, the aircraft is waiting at the gate until 12:00. The departure flight served by this aircraft then starts at 12:00. The following 30 min represent the activity processing time for aircraft checking, catering refreshment, and passenger embarking. So the departure activity starts from 12:00 and ends at 12:30. Parking activity, which is the time duration the aircraft stays at the airport waiting for its next departure flight, starts from 10:40 and ends at 12:00. Note that the minimum buffer time for assigning successive aircrafts to the gate is included in the activity processing time. During the parking time period, the aircraft may be moved to a different gate or even at the parking apron. The parking apron is assumed to have large enough capacity for all the parking activities. Towing the aircraft to the parking apron during parking time may leave more gate buffer time for other flights, however, it will lead to additional tow costs.

The assignment is subjected to restrictions which include:

(1) Each flight activity (either arrival or departure) must be assigned to a gate.
(2) The parking activity can be assign to either a gate or parking apron.
(3) The time of activities which are assigned to the same gate cannot be overlapping.
(4) Gates are excluded for certain aircraft type.

To obtain better gate assignment, three factors are considered as the objective: (1) conflict cost measures the robustness of the schedule; (2) tow cost measures the staff and facility cost required when changing the position of aircraft, and (3) passenger transfer cost that measures the passengers' satisfaction level.

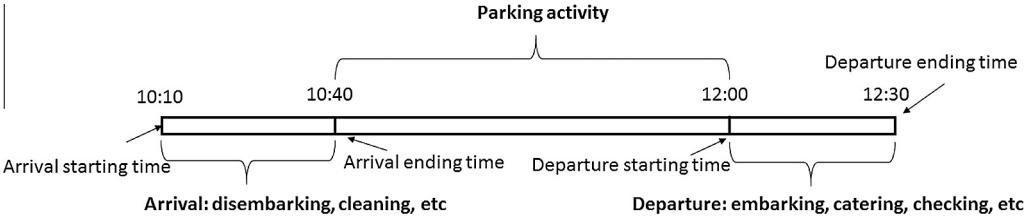

Fig. 1. Arrival, parking and departure activities of an aircraft.

The following denotations are used in the formulation:

Set:	
A	Set of arrival activities/flights
D	Set of departure activities/flights
P	Set of parking activities
F_k^i	Set of outgoing arcs of activity i in the network of gate k
G	Set of available (real) gates at the airport, which is indexed by k and l
N	Set of nodes which is indexed by n
S_i	Starting arcs connecting to activity i
Ω	Set of connecting arcs
Ω^k	Connecting arcs in the network of gate k
CCE	Set of cycle arcs
$Input_n$	Set of incoming arcs of node n
$Output_n$	Set of outgoing arcs of node n
Parameters:	
s_i	Starting time of activity i
e_i	Ending time of activity i
$\Delta t(i,j)$	The separation time between activity i and j, $\Delta t(i,j) = s_j - e_i$
cf_{ij}	The conflict cost between arrival activity i and departure activity j
$U(i)$	The successive activity following activity i, equals 0 if no activity follows
$SG(i,j)$	Equals 1 if activity i and activity j are not assigned to the same gate, equals 0 otherwise
m_{ij}	The number of transfer passenger between arrival flight i and departure flight j
w_{kl}	The traveling distance between two gates k and l
α, β, γ	The weighting parameter for three objectives respectively
c_{ij}	The cost of connecting arc between activity i and j in the network flow model
Decision variables:	
y_{ik}	Binary variable, equals 1 if activity i is assigned to gate k, equals 0 otherwise
x_{ijk}	Binary variable, equals 1 if activity i and j are successively assigned to gate k, equals 0 otherwise
ξ_{ik}	Auxiliary continuous variable in the MIP model

3.1. Robustness

In this research, a robust schedule means a lower expected waiting time of the new incoming aircraft for the occupying aircraft to leave the gate. This is measured by gate conflict. The gate conflict cost between two flights assigned to the same gate is defined as the overlap time between them. Referring to Fig. 2, suppose flight 1 and flight 2 are assigned to the same gate. In the original schedule (a), flight 1 is scheduled to arrive at gate at 11:00 and leave gate at 11:30, while flight 2 is scheduled to arrive at gate at 11:40 and leave gate at 12:05, and the separation time is 10 min. We suppose the duration of operation time at gate is fixed for both flight. There will not be any gate conflict between flight 1 and flight 2. However, the actual flight time may be different with the original one due to disruptions. In (b), flight 1 is delayed for 20 min and arrives at gate at 11:20 and supposed to leave gate at 11:50. When flight 2 arrives at the airport at 11:40, flight 1 is still occupying the gate, and then the gate conflict will be 10 min, as flight 2 has to wait for 10 min. In (c), flight 2 arrives at the airport at 11: 25, 15 min earlier than its original schedule time. As flight 1 is operated on schedule and will leave gate at 11:30, flight 2 has to wait for 5 min for flight 1 to leave the gate. In this case, the time of gate conflict is 5 min, which is from 11:25 to 11:30.

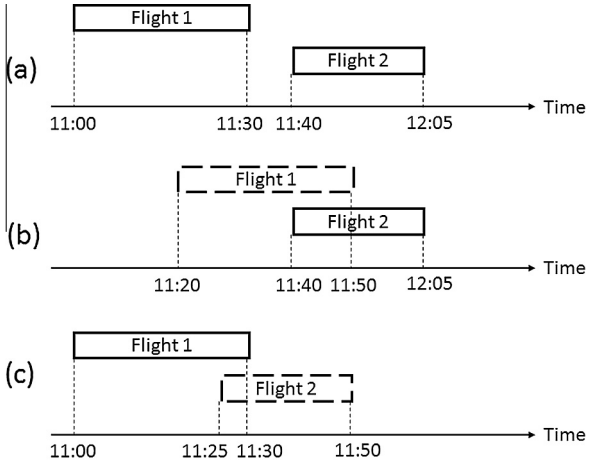

Fig. 2. Two situations causing gate conflicts.

In the original schedule, there should not be any such conflict, however, if the buffer time is not arranged appropriately, late departure or early arrival will break this balance easily. Referring to Fig. 3, it is assumed that all the flights are compatible with both gate 1 (G1) and gate 2 (G2). The upper figure (a) represents a non-robust schedule. The buffer time between flight 1 and flight 2 is too short that even minor time variations could cause gate conflict between them. If we switch the gates assigned for flight 2 and flight 4, the resulting assignment, which is shown in (b), would be more robust.

In order to minimize the expected conflict cost, we analyze the arrival and departure statistics and find the relationship between the expected conflict cost and separation time scheduled between two flights. The expected conflict cost is thus modeled as a function of separation time dispatched between two flights successively assigned to the same gates.

As an example, we analyze four consecutive day flight data of HKIA from 5 May to 8 May 2014 with 2191 arrivals and 2253 departures. The delay distribution of the arrival and the departure is listed in Table 2.

The arrival delay (minute) distribution and departure delay (minute) distribution are shown in Figs. 4 and 5, respectively.

Referring to the work of Kim (2013), we model the delay using a Log-normal distribution. Assuming the arrival delay and departure delay are independent, we can calculate the expected conflict and fit it to an exponential function of the separation time. After that, we obtained the approximate model using Mathematica as Eq. (1):

$$\mathrm{Exp}(cf_{ij}) = f(\Delta t(i,j)) \sim 15.6 * 0.966^{\Delta t(i,j)} \tag{1}$$

in which $\Delta t(i,j)$ denotes the scheduled separation time between two flights and $f(\Delta t(i,j))$ indicates that the expected conflict between activity i and j is a function of the separation time between them. Note that 15.6 and 0.966 are both problem-specific and should be determined by the historical data of the target airport. The curve is shown in Fig. 6. In the computational experiment, we use 15.6 and 0.966 for tests. We can approximate the conflict cost by relating it to the separation time between two flights that are successively assigned to the same gate. Thus, the gate conflict cost can be formulated as Eq. (2):

$$z_1 = \sum \mathrm{Exp}(cf_{ij}) = \sum_{i \in D} \sum_{j \in A, s_j > e_i} f(\Delta t(i,j)) \sum_{k \in G} x_{ijk} \tag{2}$$

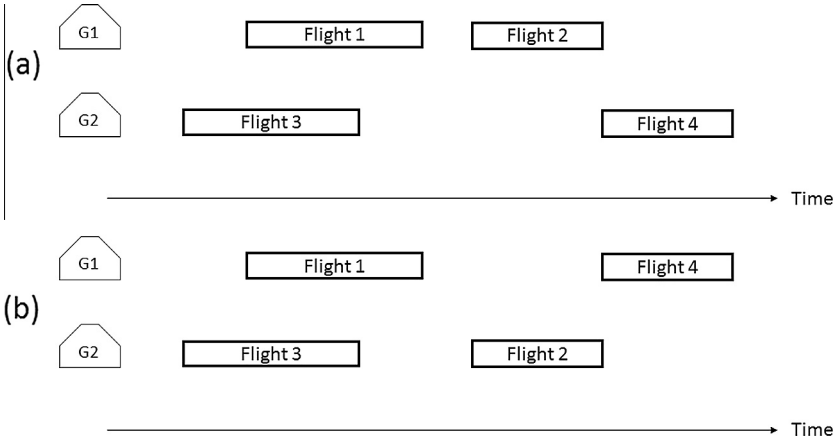

Fig. 3. An example of a non-robust schedule and the modified robust schedule.

Table 2
Delay distribution of HKIA flight data (5 May–8 May 2014).

Index	Arrival	Departure
Total flight	2191	2253
Mean/min	2.6	9.7
Median/min	−3	2
Max/min	1153	280
Min/min	−78	−60
SD/min	55.3	28.3

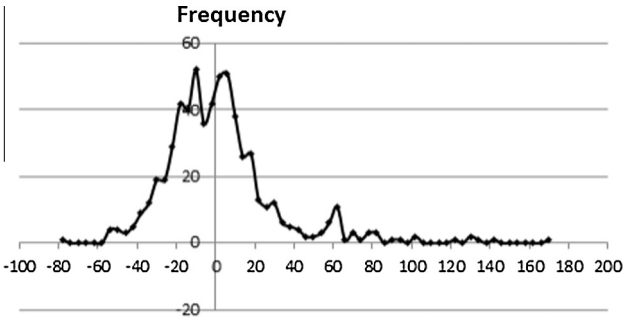

Fig. 4. Arrival delay distribution.

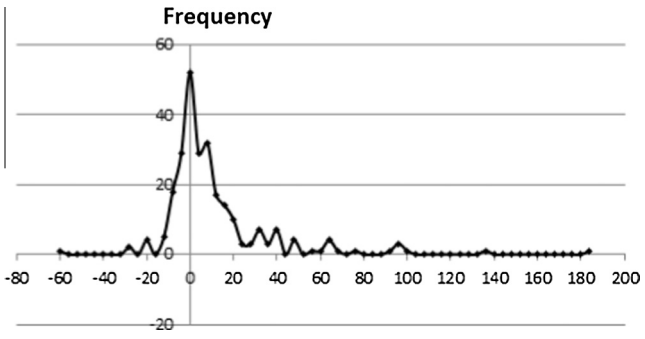

Fig. 5. Departure delay distribution.

This cost function for evaluating the robustness is compatible with our intuition: at the beginning, the curve is steep which means a small increase of separation time will improve the robustness between this pair of flights when the separation time is relatively short. As the separation time becomes longer, further increases have less impact on the robustness, as the separation time is long enough that minor variation will not cause a gate conflict.

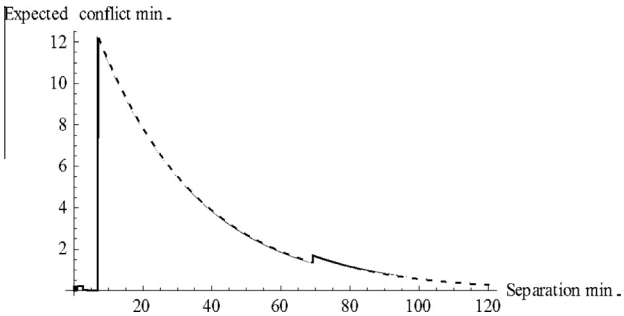

Fig. 6. Fitting an exponential curve for expected conflict.

3.2. Tows

As mentioned before, an aircraft has three activities at the airport, namely arrival, parking and departure. We assume the parking area has a large enough capacity to provide parking space for all the aircraft. Parking activity can be assigned to either a gate or the parking apron area. However, to use the facility to disembark and embark passengers, the arrival activity and departure activity should be assigned to the gate, using either a jet bridge gate or a remote gate.

During the time of parking activity, if the duration is short, then an aircraft is usually assigned to the same gate with its arrival and followed by its departure. However, when the gate resource is scarce, especially during the peak time, the occupation by the parking activity will make the schedule rather crowded. Therefore, for the parking activity with longer duration, it may be better to move it to the parking area. Moving the aircraft will lead to additional tow costs. It is a trade-off and requires deliberate balance.

Moreover, the arrival flight and departure flight of the same aircraft may have different preference to the gate. In robust gate assignment problem considered in this paper, the preference is reflected by the transfer distance. The two flights that have more transfer passengers between them will prefer to be assigned to the gates closer to each other. Therefore, the departure flight may be assigned to a different gate to the arrival flight. The parking activity could be either with the arrival or with the departure. Fig. 7 shows four possible assignments for three activities of an aircraft. In (a), the aircraft is assigned to gate 1 (G1) all the time. In (b), the arrival flight is assigned to gate 1 and parks until its departure. Then it is moved to gate 2 (G2) for departure. In (c), the aircraft arrives at gate 1 and is moved to gate 2 for parking and departure. In (d), the arrival is assigned to gate 1, then the aircraft is moved to

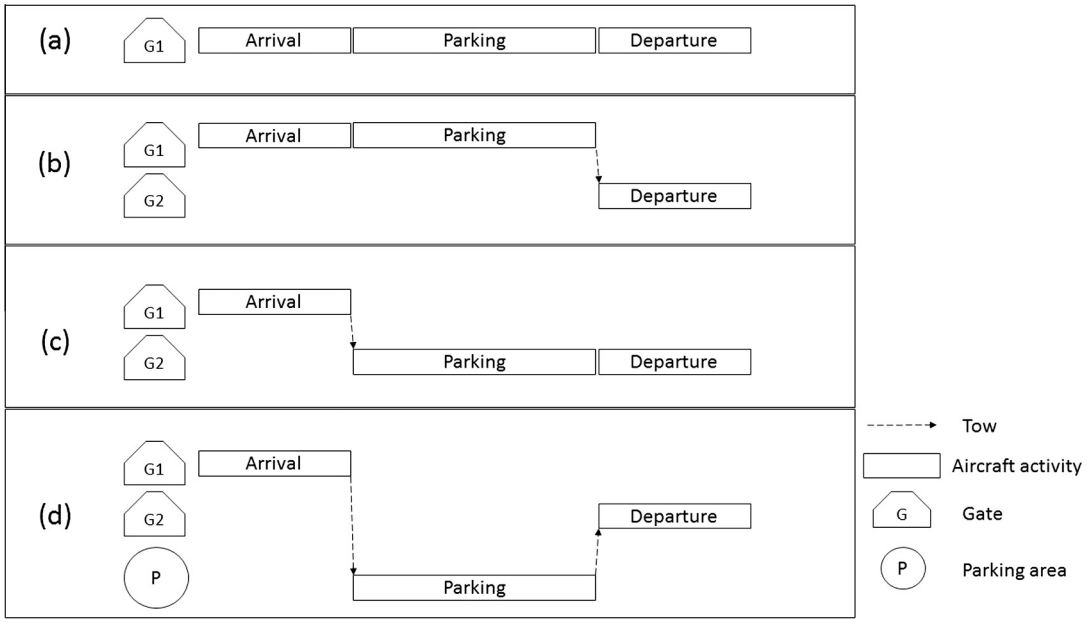

Fig. 7. Possible assignments for an aircraft at an airport.

the parking apron, and then assigned to gate 2 for departure. Each movement of aircraft needs external power provided by tractors to save the fuel cost. We call this movement a tow. In another word, if two successive activities are not assigned to the same gate, there will be a tow. When we tow the aircraft to another gate or the parking apron area, it will not only lead to additional personnel and facility cost, but may also affect the taxiway operations. The tow cost is calculated by the frequency that successive activities are not assigned to the same gate. There is zero tow, one tow, one tow, and two tows in (a), (b), (c), and (d) respectively.

According to this definition, we can formulate the tow cost as Eq. (3):

$$z_2 = \sum_{i \in A \cup P} SG(i, U(i)) \tag{3}$$

where $U(i) = j$, if j is the successive activity of i, $U(i) = 0$ if there is no successive activity following i. Specifically, for one aircraft, the parking activity is the successive activity of the arrival flight, while the departure activity is the successive activity of the parking. $SG(i,j) = 0$ if i and j are assigned to the same gate, $SG(i,j) = 1$ if they are assigned to different gates. In this research, we assume there is a parking apron area with unlimited capacity for parking activities only. Let P denotes the set of parking activities.

3.3. Transfer passengers

The transfer passengers' satisfaction is one of the most important factors that affect the airport's performance and revenue. Convenient transfer procedures and shorter traveling distance are beneficial for the transfer passengers to catch up the connecting departure flights especially when the connection time is short. The transfer distance is simultaneously determined by the arrival gate and departure gate, therefore, the cost for evaluating the service satisfaction of transfer passengers is decided by both two assignments of their connecting flights.

As a result, the transfer cost is formulated as a quadratic equation.

$$z_3 = \sum_{i \in A} \sum_{j \in D} \sum_{k \in G} \sum_{l \in G} m_{ij} w_{kl} y_{ik} y_{jl} \tag{4}$$

In general, the objective of GAP considered in this paper can be represented as the weighted summation of these three objectives, as illustrated in Eq. (5).

$$\min z = \alpha z_1 + \beta z_2 + \gamma z_3 \tag{5}$$

3.4. Trade-off between three objectives

In fact, since the gate resources are limited, three parts of objectives are counterparts to each other. Trade-off between each pair of them is illustrated subsequently.

3.4.1. Tow cost and robustness

Assigning the same gate to parking activity directly following arrival will save a tow while the parking activity will occupy the gate and thus make the assignment on the gates more crowded, less separation time, therefore longer expected gate conflict time. Fig. 8 illustrates this situation. The number next to the activity type denotes the aircraft code.

In the first situation (a), arrival 2 is tightly following departure 1 and they are both assigned to the gate 1 (G1). The separation time is short between activities. In the second situation (b), the parking 3 is moved to the parking apron and arrival 2 is assigned to gate 2 (G2). The separation time between flights in (b) is much looser and therefore leads to smaller expected conflict time but provokes three more tows. The trade-off relationship between tow cost and robustness will be explored in Section 5.5.

3.4.2. Robustness and transfer distance

Since the major attribute to robust schedule is assumed to be relatively long separation time between successive flights assigned to the same gate, allocating some flights from crowded gates to free gates is a natural choice. However, for hub airports, the distance between some gate pairs are long to each other so that assigning two flights to them will lead to inconvenience for passengers to catch up with their second flights. Fig. 9 shows an example. Suppose most of the transfer passengers who take Arrival 1 as their first flight will fly on Departure 1 and Departure 2. Gate 1 and Gate 2 are on two sides of the terminal building and therefore have long distance to each other. Assigning all three flights to the same

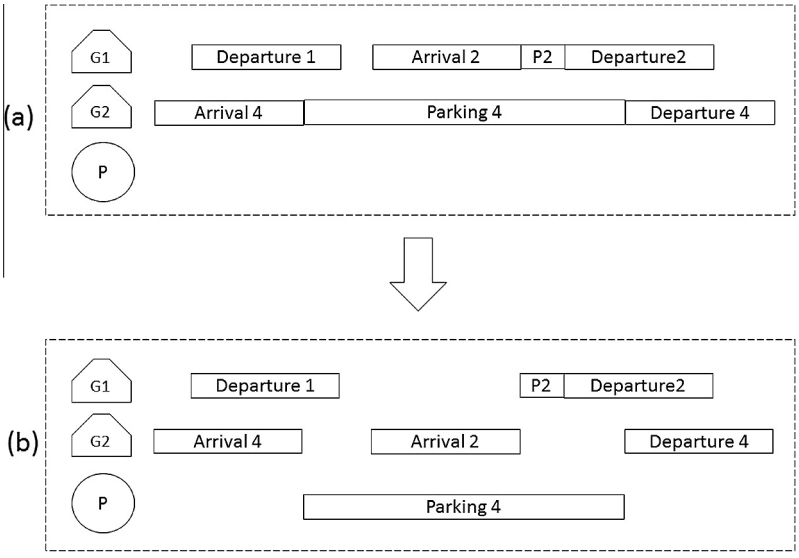

Fig. 8. Illustration of relationship between tows and separation time.

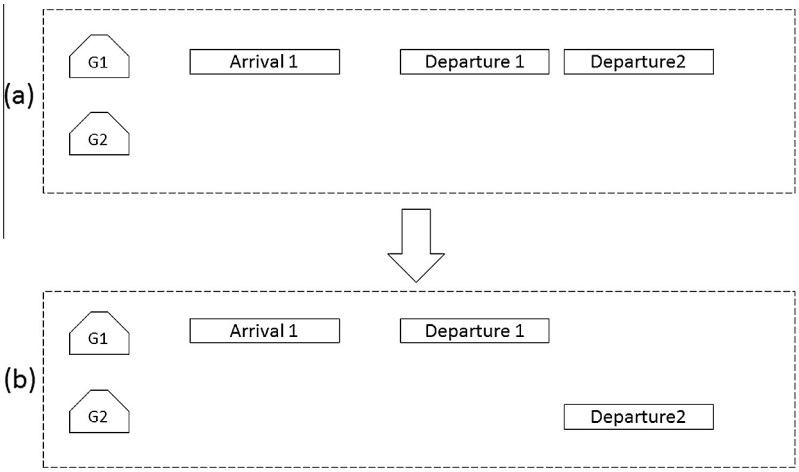

Fig. 9. Illustration of relationship between separation time and distance.

gate will definitely satisfy the transfer passengers with respect to traveling distance, however affect the robustness between Flights Departure 1 and Departure 2 (as shown in Fig. 9(a)). Allocating Departure 2 to Gate 2 will affect the transfer convenience, while improve the robustness of schedules, which indicates possible better on-time performance of future flights.

3.4.3. Transfer distance and tow cost

In some circumstance, pursuit of high passenger satisfaction with respect to travel distance requires more tows. In Fig. 10, the distance between Gate 1 and Gate 2 is long. Arrival 1 has many passengers who will take Departure 2 as second flight and Arrival 2 has many passengers who will take Departure 1 as second flight. Fig. 10(a) shows the schedule that no tow occurs. However, the transfer passengers have to walk long. In Fig. 10(b), Departure 1 and Departure 2 switch their gate, which leads to two tows but save transfer distances for passengers.

3.5. Network flow model with quadratic objective (M1)

The expressions for the objective functions considered above is hard to be formulated as a single mathematical model. Moreover, quadratic terms exist in both constraints and objectives. The model can be built as a network flow model that makes it easier to solve.

An example of a network built for one gate is illustrated in Fig. 11. The flight arcs are only constructed for the compatible gates.

The connecting arcs are only built between the activities with $\Delta t(i,j) \geqslant 0$. The cost of all the arcs except connecting arcs is 0. The set of connecting arcs is denoted as Ω. To set the costs on connecting arcs, we divide them into three groups:

$$\Omega_1 = \{(i,j)|(i,j) \in \Omega, i \in D, j \in A\}$$
$$\Omega_2 = \{(i,j)|(i,j) \in \Omega, j = U(i)\}$$
$$\Omega_3 = \Omega \setminus (\Omega_1 \cup \Omega_2)$$

The costs defined on the arcs are:

$c_e = \alpha f(\Delta t(i,j))$, *if* $e = (i,j) \in \Omega_1$,
$c_e = -\beta$, *if* $e = (i,j) \in \Omega_2$,
$c_e = 0$, *if* $e = (i,j) \in \Omega_3$.

The reformulated model is given by Eqs. (6)–(12).

$$\text{Min } z' = \sum_{e \in \Omega} c_e x_e + \gamma \sum_{i \in A} \sum_{j \in D} \sum_{k \in G} \sum_{l \in G} m_{ij} w_{kl} y_{ik} y_{jl} \quad (6)$$

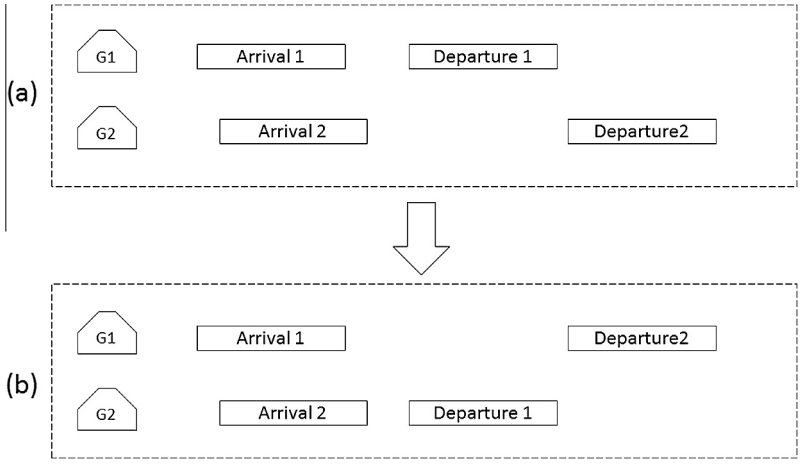

Fig. 10. Illustration of relationship between separation time and distance.

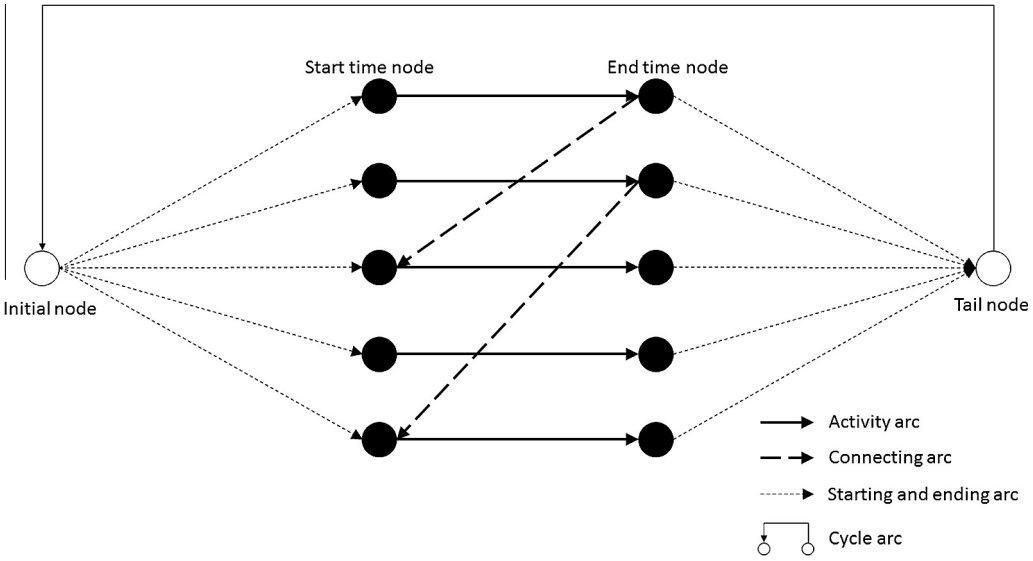

Fig. 11. A network for one gate.

$$\text{s.t.} \quad \sum_{e \in S_i} x_e = 1, \quad \forall i \in A \cup D \tag{7}$$

$$\sum_{e \in S_i} x_e \leqslant 1, \quad \forall i \in P \tag{8}$$

$$x_e \leqslant 1, \quad \forall e \in CCE \tag{9}$$

$$\sum_{e \in Input_n} x_e = \sum_{e \in Output_n} x_e, \quad \forall n \in N \tag{10}$$

$$\sum_{e \in F_k^i} x_e = y_{ik} (k \in G, i \in A \cup D) \tag{11}$$

$$x_e, y_{ik} \in \{0, 1\} \tag{12}$$

Objective (6) aims to minimize the weighted arc cost and transfer cost. In Proposition 1 presented in next paragraph, we will prove that the values between (5) and (6) differ only the constant $\beta|\Omega|$. That means if we solve (6)–(12), the solution of the original problem are naturally obtained. Constraint (7) ensures that exactly one gate is assigned to the arrival and departure activity. Constraint (8) makes sure that parking activity can be assigned either a gate or apron. Constraint (9) guarantees that each gate can only serve one activity list. Coupled with the procedure to constrain the feasible arcs, we can guarantee that no time overlapping occurs in a gate. Constraint (10) ensures flow conservation. Constraint (11) describes the relationship between two groups of decision variables. Note that x_e in formula (6)–(12) is defined on all feasible arcs, which means it

- not only includes the x_{ijk} in formula (2). x_{ijk} in formula (2) is defined based on the arcs that satisfy specific constraints: $i \in A, j \in D, s_j > e_i$,
- but also includes the x_{ijk} that satisfy $j = U(i)$. This part of decision variables are used to represent the formula (3), although they are not identical, which is why we gave Proposition 1 and its proof,
- and includes the remaining x_{ijk} that does not satisfy the above two types of constraints. Since we set the cost on this part of arcs as 0, they will have no significant effect on the model transformation.

Proposition 1. *The values between (5) and (6) differ only the constant $\beta|\Omega|$.*

Proof. First we assume the solution obtained by solving the network flow model has the objective value $z < +\infty$. Note that $\Omega_1 \cap \Omega_2 = \varnothing$, as there is no successive activity for departure activity of an aircraft. Since the second term is the same within the two models, we could focus on the first term $\sum_{e\in\Omega} c_e x_e$.

$$\sum_{e\in\Omega} c_e x_e + \gamma \sum_{i\in A}\sum_{j\in D}\sum_{k\in G}\sum_{l\in G} n_{ij} w_{kl} y_{ik} y_{jl} = \sum_{e\in\Omega_1} c_e x_e + \sum_{e\in\Omega_2} c_e x_e + \gamma z_3$$

$$= \sum_{k\in G}\sum_{e\in\Omega_1^k} c_e x_e + \sum_{k\in G}\sum_{e\in\Omega_2^k} c_e x_e + \gamma z_3$$

$$= \sum_{k\in G}\sum_{e\in\Omega_1^k} \alpha f(\Delta t(i,j)) x_e + \sum_{k\in G}\sum_{e\in\Omega_2^k} (-\beta) x_e + \gamma z_3$$

$$= \sum_{k\in G}\sum_{e\in\Omega_1^k} \alpha f(\Delta t(i,j)) x_e + (-\beta)\left|\left\{ e\in\Omega_2 \left| \sum_{k\in G}\sum_{e\in\Omega_2^k} x_e = 1 \right.\right\}\right| + \gamma z_3$$

$$= \sum_{k\in G}\sum_{e\in\Omega_1^k} \alpha f(\Delta t(i,j)) x_e + (-\beta)|\Omega_2|$$

$$- (-\beta)\left|\left\{ e\in\Omega_2 | \sum_{k\in G}\sum_{e\in\Omega_2^k} x_e = 0 \right.\right\}\left| + \gamma z_3 \right.$$

$$= \alpha z_1 + \beta z_2 + \gamma z_3 + (-\beta)|\Omega_2|$$

Therefore, solution z satisfying Eqs.(5), (7)–(12) could be obtained by solution z' from (6)–(12) plus $\beta|\Omega_2|$. □

Model 1 could be directly solved by mathematical solvers such as CPLEX. However, the computation time is too long for practical applications. This will be demonstrated in the numerical study in Section 5.2.

3.6. Linearization of quadratic objectives (M2)

The quadratic expressions in the objective are hard to solve even for small instances. Therefore, we further transform the quadratic objective in M1. Let UB_{ik} be the upper bound of the expression $\sum_{j\in D}\sum_{l\in G} n_{ij} w_{kl} y_{jl}$, so $0 \leqslant \sum_{j\in D}\sum_{l\in G} n_{ij} w_{kl} y_{jl} \leqslant UB_{ik}$ holds for all $i \in A, k \in G$.

Then we replace the objective (6) with (13) as well as adding constraints (14) and (15) to (7)–(12), and obtain (M3).

$$\text{Min} \sum_{e\in\Omega} c_e x_e + \gamma \sum_{i\in A}\sum_{k\in G} \xi_{ik} \tag{13}$$

$$\xi_{ik} \geqslant \sum_{j\in D}\sum_{l\in G} y_{jl} n_{ij} w_{kl} - UB_{ik} * (1 - y_{ik}),\ (i \in A, k \in M) \tag{14}$$

$$\xi_{ik} \geqslant 0 (i \in A, k \in M) \tag{15}$$

x_e, y_{jk} subject to (7)–(12).

Proposition 2. *(M1) and (M2) are equivalent in the sense that they have the same optimal solutions.*

Proof. The first expression in the objective of two models are the same, thus we focus on the second expression. Observe that each ξ_{ik} in (M2) is subject to two constraints:

$$\text{If } y_{ik} = 0 : \xi_{ik} \geqslant \sum_{j\in D}\sum_{l\in G} y_{jl} n_{ij} w_{kl} - UB_{ik}, \quad \xi_{ik} \geqslant 0$$

$$\text{If } y_{ik} = 1 : \xi_{ik} \geqslant \sum_{j\in D}\sum_{l\in G} y_{jl} n_{ij} w_{kl}, \quad \xi_{ik} \geqslant 0$$

The constraints on ξ_{ik} are thus reduced to:

$$\xi_{ik} \geqslant \begin{cases} 0 & \text{if } y_{ik} = 0 \\ \sum_{j\in D}\sum_{l\in G} y_{jl} n_{ij} w_{kl} & \text{if } y_{ik} = 1 \end{cases}$$

Since $\gamma \geqslant 0$, the optimal solution will always be achieved with equality holds, so the requirements can be written as:

$$\xi_{ik} = \begin{cases} 0 & \text{if } y_{ik} = 0 \\ \sum_{j\in D}\sum_{l\in G} y_{jl} n_{ij} w_{kl} & \text{if } y_{ik} = 1 \end{cases}$$

without affecting the optimal solutions. Substituting ξ_{ik} in (M2), we obtain:

$$\text{Min } z' = \sum_{e\in\Omega} c_e x_e + \gamma \sum_{i\in A, k\in G, y_{ik}=1}\ \sum_{j\in D}\sum_{l\in M} y_{jl} n_{ij} w_{kl}$$

Subject to the constraints (7)–(12).

This is just the same as (M1), which completes the proof. □

To our knowledge, there is only one MIP model previously proposed for equivalently transforming the quadratic objective in GAP considering the transfer passengers. It was proposed by Xu and Bailey (2001). In this research, M3 is created based on an adaptation from Xu and Bailey's model (Xu & Bailey, 2001). The major difference between M2 and M3 lies in the number of decision variables and auxiliary constraints. In M2 we introduce one decision variable and one auxiliary constraint for each $\langle i,k\rangle$, $i \in A$, $k \in G$. Totally the number of increment decision variables and constraints is $|A| * |G|$. In M3, the number of added decision variables and auxiliary constraints is $|A| * |D| * |G| * |G|$, as one decision variable and three auxiliary constraints are added for each $\langle i,j,k,l\rangle$, $i \in A$, $j \in D$, $k \in G$, $l \in G$. As a result, the size of M2 is smaller than that of M3. Comparisons of efficiency are made between M1, M2 and M3 in the numerical experiments in Section 5.2.

4. Algorithms development

To make the problem solvable, we transform the quadratic model into an equivalent MIP model. After the transformation, the characteristic of our model is different from current scheduling models and it is hard to use current scheduling algorithm to solve it. To deal with this problem, we propose to use general MIP solving algorithms. RINS, diving and local branching are very popular MIP-based heuristics for solving general MIP models, especially those NP-hard ones. We found that they achieve good performance in solving M2.

In this section, we adapt the diving heuristic, relaxation induced neighborhoods (RINS), and local branching (LB) to solve our GAP. The details of solving procedure are shown in Sections 4.3. These three algorithms are adopted for solving the MIP model of our GAP as they have shown good performance in solving general MIP problems.

Branch and bound is a traditional method to solve general MIP problems. Based on our preliminary computational results, pure branch and bound scheme is far slower than the branch-and-cut scheme imbedded in the CPLEX solver. Therefore, in the computational experiments, CPLEX solving M2 is used for a benchmark comparison.

In Section 4.4, we describe our proposed algorithm Variable Reduce Neighborhood Search (VRNS).

The denotations used in these algorithms are listed as follows:

φ	IP solution time limit/node solution time limit
Φ	Total solution time limit
ψ	Sub iteration limit
Ψ	Total iteration limit
TL	Time left
ρ_c	Candidate AP ratio
ρ_n	Node pool ratio
σ	Step size
dr	Diving ratio
dv_max	Maximum diversify times
k	Distance of candidate solution and incumbent solution
$\tilde{x}$	New MIP solution
$\bar{x}$	Solution obtained from Linear relaxation programming model (LP)
x^*	Best solution
AP	The set of decision variables that will be fixed to 1
UP	The set of decision variables that will be fixed to 0
UB	Upper bound
$bestUB$	The objective value of best solution
$first$	Return the first found feasible MIP solution

Observing that Constraint (11) ensure that fixing a variable y_{ik} to one has the effect of fixing a large number of other variables $y_{ik'}, x_e, \quad \forall j \in N,\ k' \in G,\ k' \neq k$, to zero. So the resulting sub-MIP is typically much easier than the original problem. Therefore, in the following context, the decision variables that we choose to fix the value (either to 1 or 0) only include y_{ik}. For general denotation, we denote the decision variables as x_j. In some of the algorithms, we use AP to denote the set of decision variables fixed to 1, and UP to denote that fixed to 0.

4.1. Diving heuristic

The main idea of a diving heuristic is as follows: firstly an Linear programming (LP) relaxation is solved and we fix the variables to 1 whose values are equal to 1 in the obtained LP solution. We denote the set of variables that are fixed to 1 as assign pool (AP). On the contrary, the set of variables that are fixed to 0 is denoted as unassign pool (UP). Then the simplified MIP sub-problem is solved by CPLEX. If no integer solution is obtained within a predefined time limit, then LP relaxation is further solved and we fix the variables whose values are greater than $1-\sigma$, where $\sigma(\sigma > 0)$ is a threshold. The MIP is again solved and if infeasibility is detected, σ is reduced in next LP. The diving heuristic used for comparison is adapted from the one used in Liang and Chaovalitwongse (2012). The main procedures are:

(1) Solve MIP model.
   a. If optimal solution $\tilde{x}$ is obtained. if $c^T\tilde{x} < bestUB$, update the best solution and bestUB, else do nothing. GO to (2).
   b. If proven infeasible, move the last added variable of AP to UP, set TL to φ, $first$ to false, go to (1).
   c. If a feasible solution obtained, if $c^T\tilde{x} < bestUB$, update the best solution and bestUB, else do nothing. set TL to φ, $first$ to false, go to (2).
   d. If no solution is obtained, set TL to $total\ time - elapsed\ time$, $first$ to false, go to (1).
(2) Solve LP, add the largest $\bar{x}_i$ which is not in AP to AP. If the obtained solution is integer, or achieve time limit, end; else go to (1).

The details are shown in Fig. 12.

4.2. Relaxation induced neighborhoods (RINS)

The key idea of RINS is to devise a sub-MIP as neighborhood of current solution at a node of branch-and-bound tree. The variables having the same value in the continuous relaxation and incumbent feasible solution are fixed to their current value, and the remaining variables are further solved. It is based on the possibility that the neighborhood generated by fixing the same variables in feasible solution and linear relaxation solution contain better solutions. We use RINS directly by setting the parameters in CPLEX as specified in Danna et al. (2005). We set $f = 100$ and $nl = 1000$, as suggested in their work. f denotes the frequency we apply RINS, and nl is the node limit.

4.3. Local branching (LB)

The spirit of basic LB is to modify the branching rule by defining a k-opt neighborhood and searching within this neighborhood. Given an incumbent solution $\bar{x}$, k-opt neighborhood is constrained as Eq. (16):

$$\Delta(x,\bar{x}) := \sum_{j\in\bar{S}}(1-x_j) + \sum_{j\in B\setminus\bar{S}} x_j \leqslant k \tag{16}$$

where B denotes the binary variable set and $\bar{S}$ denotes the subset of binary variables that are set to 1 in the incumbent solution $\bar{x}$. The search space is partitioned into k-opt neighborhood of incumbent solution and the remaining part. The remaining part is constrained by applying $\Delta(x,\bar{x}) > k$. When an optimal solution for the left branch (k-opt neighborhood) is found, the other branch is further split into two parts and continue searching the corresponding left branch.

The local branching we adopted is the enhanced version of local branching (LB) from Fischetti and Lodi (2003). The details are shown in Fig. 13.

4.4. Variable reduce neighborhood search (VRNS)

Instead of fixing the value of these variables in AP to 1 as we did in the Diving heuristic, in the proposed algorithm, AP denotes the set of potential decision variables to be fixed to 1, $y_{ik} \in AP$ indicates we potentially assign activity i to gate k. Similarly, UP denotes the set of potential decision variables to be fixed, $y_{ik} \in UP$ indicate we potentially prevent activity i to be assigned to gate k. This idea is borrowed from RINS in the hope that the neighborhood generated by fixing the same variables in feasible solution and linear relaxation solution may contain better solutions. Moreover, the AP is too small for handling such large number of decision variables if we put into it only the variables having same value, i.e. value one, in feasible solution and linear relaxation. Therefore, we borrow the concept in diving heuristic that also include the variables having value near one in LP solution. Specifically, we have two candidate set: RINS set $RAP = \{j | j \in B, \overline{x_j} \geqslant 0.5, \widehat{x_j} = 1\}$ and Diving set $DAP = \{j | j \in B, \overline{x_j} \geqslant 0.5\}$. B denotes the set of all binary variables, in our problem, it include all y.

From our preliminary computational results, we find that diving heuristics perform better in large problems, while RINS performs better in relatively smaller problems. Therefore, we sort the elements in both sets and set a parameter dr to control the combination of variables chosen from RAP and DAP to form AP. Large dr lead to higher percentage of elements from DAP.

The main procedure consists of four components:

(1) Fix $x_j = 0$ *for* $j \in UP$, $UB = \infty$. Get a feasible solution by solving MIP and return the first feasible integer solution $\hat{x}$, set upper bound $UB = c^T\hat{x}$, go to (2).

```
Function Diving(Φ, φ,x*);
    x* := undefined; UP = null, AP = null; TL = φ; first = false
    Repeat
        Solve MIP(TL, first, AP, UP, x̃);
    1.  If(stat ="opt_sol_found")then
            If(c^T x̃ < bestUB) then bestUB := c^T x̃; x* := x̃ endif;
            Break;
        Endif;
    2.  If(stat ="proven_infeasible")then
            Move the last added variable of AP to UP
            TL = φ; first = false
            Continue;
        Endif;
    3.  If(stat ="feasible_sol_found")then
            If(c^T x̃ < bestUB) then bestUB := c^T x̃; x* := x̃ endif;
            TL = φ; first = false
        Endif;
    4.  If(stat ="no_feasible_sol_found")then
            TL := Φ − elapsedTime; first = true
            Continue;
        Endif;
        Solve LP(AP,UP, x̄); find x̄_j = max_{i∈B,i∉AP} x̄_ι,add j to AP
    Until (elapsed_time > Φ) or (x̄ is integer);
End.
```

Fig. 12. Pseudo-code of the diving algorithm.

(2) Solve LP to obtain linear solution $\bar{x}$, and according $\bar{x}$ and $\hat{x}$ and *dr*, find *AP*, go to (3).
(3) Choose a subset *subAP* of *AP*, fix $x_j = 1$ *for* $j \in$ *subAP* and $x_j = 0$ *for* $j \in$ *UP*, solve MIP within upper bound *UB* and predefined node-time-limit.
   e. If another solution is obtained, update $\hat{x}$ and go to (2).
   f. If no solution is obtained, choose another subset *subAP*, go to (3).
   g. If no solution is obtained when maximum node-iteration is achieved, go to (4).
(4) Add all elements in *AP* to *UP*, clear *AP*, go to (1) unless maximum iteration is achieved.

The detailed algorithm is shown in Fig. 14.

5. Computational results

All the experiments are performed on a PC with an i7-2620M 2.70 GHz processor. The mathematical models are solved using the commercial software CPLEX version 12.1.0, while the algorithms are implemented in Java 6. The test data sets are generated in a similar way to Ding et al. (2004b), except that we add parking activity in addition to the arrival and departure. The details of data generation is described in Section 5.1. In Section 5.2, we compare the efficiency of M1, M2 and M3. In Section 5.3, the MIP-based heuristics are run to solve our GAP. In Section 5.4, we compare the linear approximation to transfer distance with the quadratic representation of transfer distance, and show that quadratic representation is necessary to precisely evaluate the transfer passengers' experience. Section 5.5 shows the trade-off relationship between tow frequency and expected gate conflict.

5.1. Test-data generation

The test data is generated in a way similar to that in Ding et al. (2004b) with the details described in the following paragraph. The major difference lies in that we treat three continuous activities, namely arrival flight, parking activity and departure flight of an aircraft independently rather than one single activity. In our problem formulation, these three activities can be assigned to different gates rather than one gate when an aircraft is ground.

For arrival activities, s_i^a (start time of arrival activity of aircraft i) is randomly generated in the interval $[10i, 10i + 7]$, and the e_i^a (end time) is generated in the interval $[s_i^a + 20, s_i^a + 40]$, given that passenger disembarking and cleaning take around half an hour depending on the aircraft type.

For parking activities, s_i^p (start time of parking activity of aircraft i) = e_i^a, e_i^p (end time) is randomly generated in three different intervals. For "frequent" set (with prefix "fre"), it is set as $[s_i^p + 10, s_i^p + 60]$, which presents the set with short parking time. In another word, an aircraft arrives at the airport and depart again in a short turnaround time. For "stay" set (with prefix "sta"), it is set as $[s_i^p + 120, s_i^p + 300]$, which presents the set with long parking time, which means aircrafts will park for longer time at the airport.

```
Function LocalBranching(k, Φ, φ,dv_max,x*);
    rhs := bestUB := UB := TL := +∞; x* := undefined;
    opt := first := true; dv := 0; diversify := false;
    Repeat
        If(rhs < ∞) then add the local branching constraint Δ(x, x̄) ≤ rhs endif;
        TL := min{TL, Φ − elapsed_time};
        stat := MIP_SOLVE(TL, UB, first, x̃);
        TL := φ;
    1.  If(stat ="opt_sol_found")then
            If(c^T x̃ < bestUB) then bestUB := c^T x̃; x* := x̃ endif;
            If(rhs ≥ +∞) return(opt);
            Reverse the last local br. constraint into Δ(x, x̄) ≥ rhs + 1;
            diversify := first := false; x̄ := x̃; UB := c^T x̃; rhs := k
        Endif;
    2.  If(stat ="proven_infeasible")then
            If(rhs ≥ +∞) return(opt);
            Reverse the last local br. constraint into Δ(x, x̄) ≥ rhs + 1;
            If(diversify) then UB := TL := +∞; dv := dv + 1; first := true endif;
            rhs := rhs + ⌈k/2⌉; diversify := true
        Endif;
    3.  If(stat ="feasible_sol_found")then
            If(rhs < +∞) then
                If(first) then
                    Delete the last local br. constraint Δ(x, x̄) ≤ rhs
                Else
                    Replace the last local br. constraint Δ(x, x̄) ≤ rhs by Δ(x, x̄) ≥ 1
                Endif
            Endif;
            REFINE(x̃);
            If(c^T x̃ < bestUB) then bestUB := c^T x̃; x* := x̃ endif;
            diversify := first := false; x̄ := x̃; UB := c^T x̃; rhs := k
        Endif;
    4.  If(stat ="no_feasible_sol_found")then
            If(diversify) then
                Replace the last local br. constraint Δ(x, x̄) ≤ rhs by Δ(x, x̄) ≥ 1
                UB := TL := +∞; dv := dv + 1; rhs := rhs + ⌈k/2⌉; first := true
            Else
                Delete the last local br. constraint Δ(x, x̄) ≤ rhs;
                rhs := rhs − ⌈k/2⌉
            Endif;
            diversify := true
        Endif;
        Until (elapsed_time > Φ) or (dv > dv_max);
        TL := Φ − elapsed_time; first := false;
        stat := MIP_SOLVE(TL, bestUB, first, x*);
        opt := (stat = "opt_found") or (stat = "proven_infeasible");
        Return (opt)
    End.
```

Fig. 13. Pseudo-code of the local branching algorithm.

For "random" set (with prefix "ran"), it is set as $[s_i^p + 10, s_i^p + 300]$. Some parking activities in this set have short parking time, while the others have long parking time. The parking time duration and corresponding prefix are shown in Table 3.

For departure activities, s_i^d (start time of departure activity of aircraft i) = e_i^p, e_i^d (end time) is generated in the interval $[s_i^d + 20, s_i^d + 40]$, representing the time for food preparation, checking and refueling the aircraft, and embarking passengers, etc.

Function VRNS(data of activity, passenger, gate; $\varphi; \Phi; \Psi;\ \psi; \pi; \rho_c; \rho_n; dr$)

$bestUB = UB = \infty;\ S =$ size of activities×AP ratio;
Solve the MIP to obtain the first feasible solution as an initial solution $\hat{x}$, set $x^* = \hat{x},\ UP = null,\ AP = null$
While (iteration<total iteration & time <Φ)
  $E = 0$, Solve the LP to get $\bar{x}$, obtain AP according $\bar{x}$ and $\hat{x}$ and dr
  $Improve = false$
  $node\_time = \varphi$
  While (iter<node iteration)
    Randomly choose $subAP \in AP$, fix $x_j = 1 for\ j \in subAP,\ x_j = 0 for\ j \in UP$
    Solve $MIP\ (UB, node\ time)$
    If (new solution $\tilde{x}$ found):
      If ($c^T\tilde{x} < bestUB$):
        $x^* = \tilde{x};\ bestUB = c^T\tilde{x}$
      End
      $\hat{x_j} = \tilde{x}; UB = c^T\tilde{x}; Improve = true$
      Break
    End
  End
  If ($Improve == false$):
    $subAP = null,\ UB = \infty, UP = UP + AP$
    Fix $x_j = 0 for\ j \in UP,\ node\_time = \infty$
    Solve $MIP(UB, node\_time)$ to obtain first feasible solution $\tilde{x}$
    $\hat{x_j} = \tilde{x}; UB = c^T\tilde{x}$
  End
End

Fig. 14. Pseudo-code of the VRNS algorithm.

Table 3
Prefix of data name.

Prefix	Characteristic of data set
fre	The parking time is within [10,60]
ran	The parking time is within [10,300]
sta	The parking time is within [120,300]

Table 4
Suffix of data name.

Suffix	Characteristic of data set
sca	Scarce gate resource
suf	Sufficient gate resources

The transfer passenger number between activity i and j, n_{ij} is generated in the interval $[1, 20]$ if $i \in A, j \in D$ and $s_j > e_i$. $n_{ij} = 0$ otherwise.

The gate distance matrix is symmetric with w_{ij} being generated in the interval $[5, 20]$.

Two types of gate resources are generated: scarce gate resource (with suffix "sca") and sufficient gate resources (with suffix "suf"). The measure and criteria to judge whether the gate resource is scarce are as follows. We use greedy algorithm to obtain an initial solution. If one fewer gate is given, then the initial solution obtained for the problem in this group is infeasible. For example, if only one gate is given for Fig. 8(a), there will not be any feasible solution. If two gates are given, a feasible solution could be obtained. Therefore, in this instance, the initial activities set with two gates given will be a "sca" data instance. For larger instances, if the gate resource is scarce, there should be a lot of parking activities being assigned to parking apron in the initial solution, and very possible the similar case in the final solution. For the sufficient gate resources set, firstly, in the initial solution obtained by greedy algorithm, there should be no activities assigned to the parking apron. In addition, to prevent too many empty gates (which is unusual in reality), we ensure that the initial solution exploits every gate (no empty gate) except parking apron. In fact, for this group, in the final solution, the activities is likely to only occupy part of the gates, leaving some of them empty. The characteristics of gate resources and corresponding suffix is shown in Table 4.

The input contains three parts: activities with their information; distance map between each pair of gates; and the set of transfer passenger pair showing the number of passenger, their arrival and departure activity (flight). The output is supposed to be the gate assignment of all the activities.

The objective coefficients set in Sections 5.2 and 5.3 are $\alpha = \beta = \gamma = 1$. In Section 5.4, $\alpha = \beta = 0,\ \gamma = 1$. In Section 5.5, these coefficients are varied with different weighting vector to see the relationship between tows and robustness.

5.2. Comparison of M1, M2 and M3

In this comparison, experiments are conducted on the three models that are directly solved by CPLEX called by Java. The three models include quadratic model M1, MIP model M2, and the linearized model M3 adapted from the one proposed in Xu and Bailey (2001). The MIP model in Xu and Bailey (2001) is used to transform the quadratic objective of transfer passengers' walking time. However, in their study, this linearized model was not adopted for solving their GAP; instead, they proposed a Tabu search algorithm.

This test set consists of 6 small size "fre" inputs and 6 small size "sta" input. The results are presented in Table 5. The prefix "fre" and "sta" denotes the characteristic of the data set, as shown in Table 3. The first number denotes the number of activities involved, the second number denotes the number of gates involved. For example, fre9 $*$ 3 means the parking time in this data has short duration, and there are 9 activities and 3 gates. The column "value" shows the objective value and column "CPU" denotes the computation time.

In Table 5, we present the computational results of three models solving the 12 data sets. In Table 5, M1 and M2 represent the quadratic and MIP models we proposed respectively, M3 represents the linearized model adapted from the one proposed in Xu and Bailey (2001). The difference of the adapted model and the original one lies in that in addition to the transfer passengers, we also incorporate the objectives of robustness and tows.

From Table 5, we see that when the problem size is smaller than 15 $*$ 3, both "sta" set and "fre" set could be solved within 1 s for all the three models. When the size grows larger, the computation time, which is denoted as CPU in the tables, increases dramatically for M1. fre30 $*$ 5, sta27 $*$ 4, and sta30 $*$ 5 cannot achieve optimality within pre-set CPU time limit 7200 s (2 h). The CPU time of M3 grows not as fast as M1, but still much longer than M2 for data set with size larger than 24 $*$ 4. This is because the difference between the M3 and M2 lies in the decision variables and auxiliary constraints introduced by the linear transformation. The decision variables for transfer passengers in M3 are of size $|A| * |D| * |G| * |G|$, while the decision variables for transfer passengers in M2 are of size $|A| * |G|$. The introduced auxiliary constraints of M3 are of size $3 * |A| * |D| * |G| * |G|$, while the ones of M2 are of size $|A| * |G|$.

To sum up, linearization of the quadratic objective could largely reduce the CPU time. Moreover, the linear transformed method proposed in this research is proven more efficient than the one proposed in Xu and Bailey (2001). On the other hand, the solving efficiency on solving the problem with size 30 $*$ 5 is still far less than satisfactory in practical application, when problem size grows even larger, the computation time of CPLEX solving M2 directly is too large for practical applications. Therefore, we proposed to use MIP-based heuristics that exploits the strong capability of CPLEX.

5.3. Comparison of algorithms

In this section, we compare the performance of local branching, diving, RINS, VRNS and using CPLEX directly solving MIP model M2 to solve the data instances. The data used in this section is composed of 5 sets. Two middle size data, each one includes 10 problems with activity size in the range of [30,75]. There are three large size data with activity size in the range of [90,150]. The first one includes 5 "fre" problems. The second one includes 5 "sta" problems. The third one includes 5 "ran" problems. Practical size data in the range of [210,300] are further shown to prove the effectiveness of proposed algorithms.

Table 5
Comparison between M1 M2, and M3 using CPLEX.

Problem	M1		M2		M3	
Size	Value	CPU (s)	Value	CPU (s)	Value	CPU (s)
fre9 $*$ 3	216	0.11	216	0.06	216	0.12
fre15 $*$ 3	706.6	0.19	706.6	0.12	706.6	0.31
fre21 $*$ 4	950	15.1	950	0.33	950	0.83
fre24 $*$ 4	2895.1	69.1	2895.1	0.98	2895.1	23.9
fre27 $*$ 4	2774.8	684.6	2774.8	7.8	2774.8	293
fre30 $*$ 5	3311.3	7200	3285.3	59.1	3285.3	1028.4
sta9 $*$ 3	227.4	0.17	227.4	0.09	227.4	0.11
sta15 $*$ 3	1713.9	0.3	1713.9	0.3	1713.9	0.14
sta21 $*$ 4	5169.3	66.8	5169.3	0.67	5169.3	15.9
sta24 $*$ 4	2885.4	2636.8	2885.4	1.05	2885.4	44.4
sta27 $*$ 4	3436.6	7200	3436.6	1.4	3436.6	182
sta30 $*$ 5	3979.8	7200	3961.8	48.2	3961.8	1062.7

5.3.1. Heuristic parameter values setting

The computation time limit for all the algorithm is set as 2000 s. The detail of parameters for each algorithm is shown in Tables 6–9. For local branching, k controls the distance between current solution and potential better solutions. It is set as 18 in Fischetti and Lodi (2003). However, when problems become larger, a large k value makes the subproblem in each iteration hard to solve. We set k to 10 as it works best with the subproblem for most of the large problem instances both on solution quality and computation time. dv_max controls the diversification times. We set it to 7, as we found in the paper (Fischetti & Lodi, 2003) that most of best results are obtained within 7 diversifications. The node computation time of local branching and VRNS are set as 300, as we found it could give good results for most of the problems. For RINS, we set $f = 100$ and $nl = 1000$, as suggested in their work. The node time limit is set as 100 in Diving, since more iteration provide higher probability to generate better solution. The parameters set in VRNS are determined according to our preliminary experiments.

5.3.2. Results comparison

The results are listed in Tables 10–13. Tables 10 and 11 show the objective values as well as the computation time, as some of the results are obtained for short time. Table 12 shows only the objective values, as the time limit is set to 2000 s. All the algorithms work until the time limit. The time limit is set to 2 h (7200 s) for practical instances in Table 13. In addition, in Tables 10–13, we also directly solve model M2 using CPLEX to give a benchmark comparison, as shown in the last column M2. The number in bold denotes the cost obtained by the corresponding algorithm is the minimum.

First of all, for middle size instances, CPLEX can obtain best solution for about half of the instances in both Tables 10 and 11.

Table 6
Parameters of LB used in the numerical experiments.

LB	k	Φ	φ	dv_max
Large	10	2000	300	7
Middle	18			

Table 7
Parameters of RINS used in the numerical experiments.

RINS	Φ	f	nl
	2000	100	2000

Table 8
Parameters of diving used in the numerical experiments.

Diving	σ	Φ	φ
	0.05	2000	100

Table 9
Parameters of VRNS used in the numerical experiments.

VRNS	Φ	φ	ρ_c	ρ_n	dr
Large	2000	300	0.2	0.5	0.8
Middle					0.2

Table 10
Results of 10 middle size problems with scarce gate resource.

Size	VRNS		RINS		Diving		LB		M2	
$n * m$	Cost	CPU	Cost	CPU	Cost	CPU	Cost	CPU	Cost	CPU
30 * 3sca	**3505**	2	**3505**	2	**3505**	2	**3505**	2	**3505**	1
39 * 4sca	**6267**	36.6	6291	1000	**6267**	200	**6267**	2000	**6267**	915
42 * 4sca	**9176**	200	**9176**	73	**9176**	200	**9176**	2000	**9176**	928
48 * 4sca	**12,929**	146	13,036	957	12,996	135	12,932	2000	**12,929**	808
54 * 4sca	**20,244**	1000	20,951	1000	20,685	1070	20,353	2000	20,405	2000
57 * 5sca	**11,745**	2000	11,984	2000	12,461	602	11,796	2000	11,797	2000
60 * 5sca	13,704	2000	13,969	2000	14,934	900	13,563	2000	**13,366**	2000
66 * 5sca	15,337	2000	**15,152**	2000	15,402	2000	15,298	2000	15,366	2000
72 * 5sca	22,716	2000	22,733	2000	23,670	2000	**22,696**	2000	23,138	2000
75 * 5sca	22,774	2000	22,629	2000	23,697	2000	22,903	2000	**22,362**	2000

Table 11
Results of 10 middle size problems with sufficient gate resource.

Size	VRNS		RINS		Diving		LB		M2	
n * m	Cost	CPU	Cost	CPU	Cost	CPU	Cost	CPU	Cost	CPU
30 * 7suf	**3364**	170	**3364**	573	**3364**	199	**3364**	1782	**3364**	222
39 * 8suf	5619	2000	5641	2000	6283	700	5611	2000	**5598**	2000
42 * 8suf	**5794**	2000	6484	2000	7229	944	6571	2000	6382	2000
48 * 8suf	**7518**	2000	7834	2000	7621	1104	7721	2000	7723	2000
54 * 8suf	**9310**	2000	9730	2000	11,274	1105	11,025	2000	9871	2000
57 * 8suf	13,076	2000	14,131	2000	14,299	1107	**12,224**	2000	12,910	2000
60 * 8suf	12,841	2000	14,504	2000	14,854	788	12,770	2000	**12,179**	2000
66 * 8suf	15,217	2000	16,946	2000	18,729	2000	**15,147**	2000	16,226	2000
72 * 8suf	15,551	2000	16,039	2000	21,085	2000	17,742	2000	**15,519**	2000
75 * 9suf	**18,557**	2000	22,009	2000	20,555	2000	20,605	2000	20,141	2000

Table 12
Results of 15 large size problems.

Data name	VRNS Cost	RINS Cost	Diving Cost	LB Cost	M2 Cost
ran90 * 10	31,964	**30,173**	39,237	33,339	31,034
ran105 * 10	42,153	47,469	**41,573**	46,846	57,027
ran120 * 12	91,730	83,441	84,393	**81,967**	111,734
ran135 * 12	109,122	100,476	**93,842**	97,053	117,029
ran150 * 13	143,111	169,518	**140,895**	146,415	187,855
sta90 * 10	43,609	40,865	45,640	**40,010**	46,473
sta105 * 10	**62,337**	69,008	70,769	65,668	67,543
sta120 * 12	**70,562**	87,962	78,079	86,169	102,964
sta135 * 12	122,172	126,295	130,433	**119,861**	150,030
sta150 * 13	150,516	186,049	167,871	171,433	202,817
fre90 * 10	28,067	27,137	29,700	27,783	**26,860**
fre105 * 10	**47,336**	50,664	51,602	60,571	53,969
fre120 * 12	**63,189**	69,527	84,014	68,608	74,593
fre135 * 12	81,932	99,335	**81,521**	92,835	98,107
fre150 * 13	124,305	143,807	124,249	**123,476**	160,243

For these instances, although CPLEX obtains best solution, the ones obtained by other algorithms are quite close to them. On the other hand, CPLEX achieves best for only one instance in the set of large size instances.

For the instances with size smaller than 42, most of the algorithms could give optimal or near-optimal solution. Diving method achieves termination earlier (before preset 2000 s) when the problem size is small. This is because when the problem is small, fix a part of variables will make the remaining MIP problem very easy to solve. However, the quality of solutions produced is not satisfactory although it has the advantage of shorter computation time.

Diving method and local branching shows comparative performance for the large instances. That is also why we set the parameter *dr* to 0.8 in the large problems, compared to the 0.2 set in the middle size problems. In the preliminary experiments, we found that Diving method performs much better in large size data, so we set *dr* larger to include more elements from *DAP* in our candidate AP.

From Tables 10–12, we found diving algorithm performs better in solving larger instances than middle ones, when compared with RINS. This is because for middle instances, it converges too fast that it cannot achieve global optimum while other algorithms can find optimal or near optimal solutions. When problems grow larger, diving algorithm obtains better results. Fast convergence help diving algorithm find local optimum more efficient than other algorithms, although it may not be the global optimal. The basic principle of RINS is to fix the variables occurring in both feasible MIP solutions and linear solutions. When the problem size is large, there are very few common variables in feasible solution and linear ones (we denote this variables set as RINS set *RAP* in Section 4). Therefore, the guidance capability for searching better solutions based on information of current feasible solution and linear solution is much weaker.

When faced with practical size instances, LB shows advantageous performance. In each iteration, LB aims to find an optimal or near optimal solution within a constrained solution space, which surrounds the current best solution. The searching process is maintained in the neighbor of best solutions. For smaller instances, this will limit its searching capability. However, for practical application, it shows robustness with respected to solution quality. This is shown in Table 13.

The proposed algorithm VRNS performs well in large size problems because we add more diving variables (the variables with values near 1 in linear solutions) into the candidate AP to increase the exploration scope, making up the limitation of RINS. Not all the variables are fixed as this will avoid the solution being trapped in local optima prematurely. Instead, each time, we randomly choose one subset of the AP to explore. After several iterations, we assign all of them to zero to explore another solution space. For most of the instances, VRNS obtained the solution with quality between the solutions obtained by RINS and Diving. Therefore,

Table 13
Results of 15 practical size problems.

Data name	VRNS Cost	RINS Cost	Diving Cost	LB Cost	M2 Cost
fre210 * 16	263,075	290,765	268,091	**235,059**	292,805
fre225 * 16	377,136	385,454	377,387	**342,766**	392,973
fre240 * 18	417,489	416,033	416,660	**407,109**	428,139
fre270 * 20	536,527	541,998	**534,972**	538,839	546,177
fre300 * 22	693,456	699,374	693,456	**632,750**	698,567
sta210 * 16	423,523	427,531	423,560	**419,491**	454,853
sta225 * 16	505,567	492,845	505,567	**427,349**	516,394
sta240 * 18	560,010	533,855	560,010	**515,528**	551,695
sta270 * 20	739,864	719,477	739,864	**702,406**	727,433
sta300 * 22	**821,478**	912,547	**821,478**	858,543	911,331
ran210 * 16	415,831	409,039	419,855	**366,163**	420,311
ran225 * 16	383,749	378,047	394,383	**329,911**	401,684
ran240 * 18	507,679	507,015	507,679	**500,138**	521,037
ran270 * 20	740,107	648,683	740,107	**629,981**	666,175
ran300 * 22	823,090	865,030	823,090	**811,751**	870,894

VRNS could be regarded as the combination of RINS and Diving, which can be used to obtain acceptable results at least as good as one of the algorithms.

5.4. Comparison of different transfer distance solutions

Although many of the previous researches in GAP also considered transfer passengers, most of them are using approximate linear models. To simplify the solving process and build a linear model, some of them used uniform distribution to approximate the distances traveled by transfer passengers, some of them fixed either arrival gate or departure gate and only try to optimize the other end of the transfer gate. The target is to avoid the quadratic expressions and make the problem easier to solve. However, the capability of these methods to evaluate the actual distances is unexplored in previous research. Therefore, in this paper, experiments are performed to calculate the deviation between the exact results and approximate results, to show whether it is necessary to use the exact transfer distance, as we did in M1 and M2. In order to make the comparison clearer, we set coefficient $\alpha = \beta = 0,\ \gamma = 1$ in the objective function to focus on the passenger transfer distances only.

The approximate method we adopt for comparison is from Yan and Huo (2001), in which the transfer distance is defined on each gate rather than between a gate pair. The distance is approximated as the average distance from the gate to all gates (including itself). We denote this model as Model 4 (M4). Note that although M2 and M4 are both linear models, M2 uses actual distance between each gate pair to achieve exact distance measurement while M4 adopts approximate distance.

We use fre9 * 3 as an example for illustrating the meaning of exact, approximate and actual cost.

In this example, there are 9 activities. Activities 1, 2, and 3 are arrival, parking and departure for aircraft 1; 4, 5, and 6 are for aircraft 2; 7, 8, and 9 are for aircraft 3. The starting time and ending time of each activity are shown in Fig. 15. There are three gates involved. The distances between each pair of gates are shown in Fig. 16. The parking apron is not shown in the figure. The number of transfer passengers is shown in Table 14.

In M4, the approximate transfer distance (aw) related to G1 is represented as the average distance between G1 and all gates (G1, G2 and G3), which is calculated as Eq. (17):

$$aw_{g1} = 1/3(dist_{g1g1} + dist_{g1g2} + dist_{g1g3}) = 9 \tag{17}$$

Similarly, the aw_{g2} is 7.1, aw_{g3} is 6.7.

The objective of M4 is Eq. (18):

$$obj_{M4} = \sum_{i\in A}\sum_{j\in D}\sum_{k\in G} m_{ij} aw_k y_{ik} \tag{18}$$

in which the transfer distance is calculated only for the arrival flights as well as its assigned gate. The constraints are the same as we presented previously.

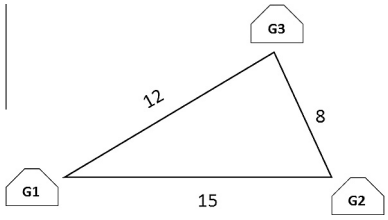

Fig. 16. Gate occupation time of 9 activities.

Table 14
The number of transfer passengers between arrival and departure activities.

Transfer number		Departure activity		
		3	6	9
Arrival activity	1	18	2	8
	4	0	12	12
	7	0	0	14

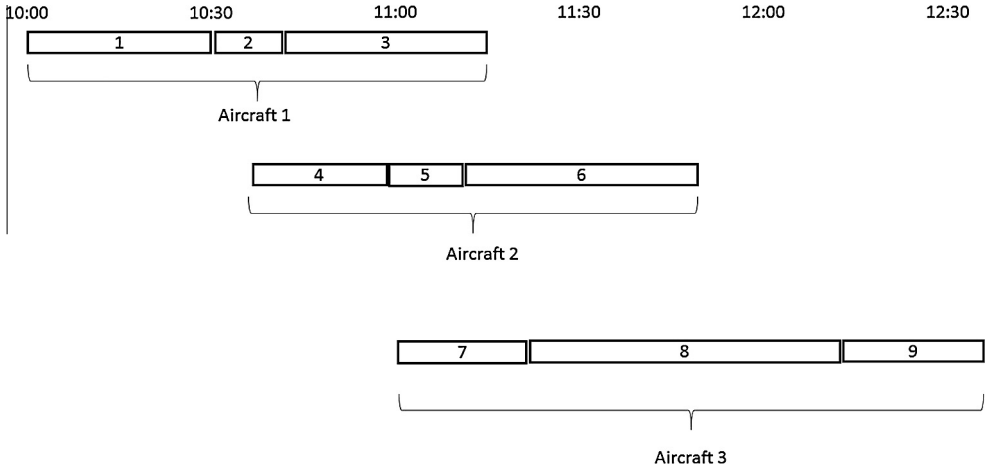

Fig. 15. Gate occupation time of 9 activities.

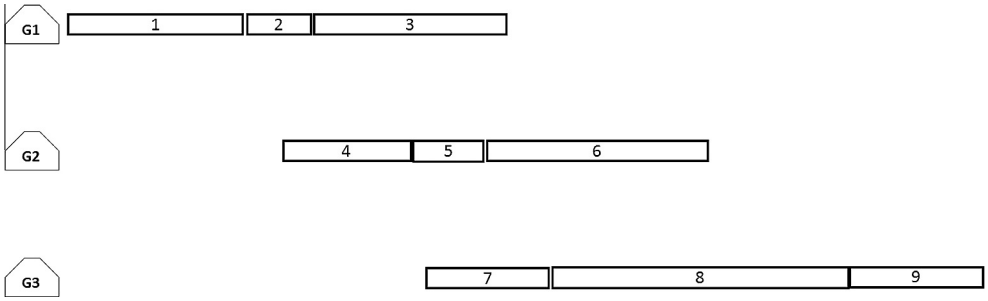

Fig. 17. Gate assignment obtained from M2.

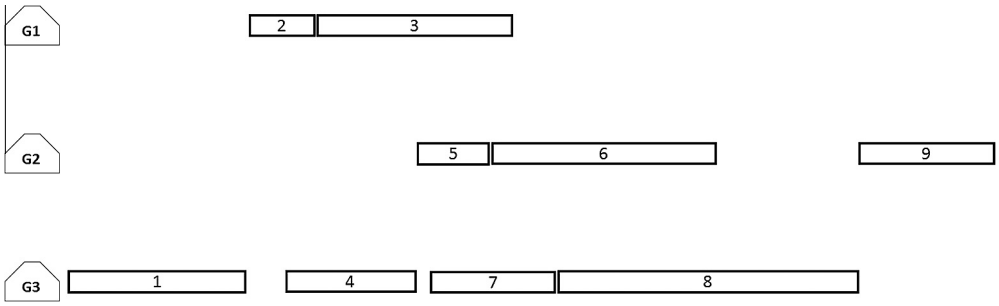

Fig. 18. Gate assignment obtained from Model 4.

To solve this instance by M2, the resulting gate assignment plan is as follows in Fig. 17. The objective value is denoted as exact. The exact cost of gate assignment is 225.

To solve this instance by M4, the resulting gate assignment plan is as follows in Fig. 18. The objective value of Eq. (18) is denoted as "approximate". The approximate cost of gate assignment in Fig. 18 is 440. However, this value is calculated based on the approximate distance *aw* of each gate. To obtain the actual transfer traveling cost of this assignment plan, we should re-calculate the travel cost by using the exact distance between each pair of gates. This cost is denoted as "actual". Note that we do not need to re-calculate the actual cost for the assignment in Fig. 17 because the obtained cost from M2 is already the actual cost of the solution. The actual cost of assignment in Fig. 18 is 600.

Apparently, the final assignments of two models are different. The column "with (1)" for deviation is calculated by the Eq. (19) to show that how worse the results obtained by M4 than M2.

$$Dev_1 = 100\% * (actual - exact)/exact \tag{19}$$

The column "with (2)" is calculated by the Eq. (20) to show the deviation that uses approximate cost to estimate the actual cost. The deviation is evaluated as the absolute value.

$$Dev_2 = 100\% * |(approximate - actual)|/actual \tag{20}$$

It is noted that the difference between approximate cost and actual cost for M4 (namely, $approximate - actual$) may be either positive or negative. As the approximate distance for a gate g must lie in $[0, fa_g]$, where fa is the distance between the gate and the farthest gate. In this instance, for gate 1 $aw_{g1} = 9$ and $fa_{g1} = 15$. Therefore, the actual cost may (theoretically) lie in $[0, fa * \sum m]$, where $\sum m$ denotes the sum of all transfer passengers and $fa = \max(fa_{g \in G})$. If the resulting assignment obtained by M4 allocates the same gate for most of the flights, then the actual distance is relatively small, since the traveling distance for the same gates is 0. On the other hand, if in the solution obtained by M4 allocates flights dispersedly, especially allocating the flight pair that has a lot of transfer passengers to two distanced gates, then the actual cost is relatively large compared to the approximate cost.

Table 15 shows the comparison results for solving 10 small instances. From Table 15, firstly, we find that the M4 consumes less computation time than the exact model. Although we succeeded in transferring the quadratic objective into linear one, the number of decision variables and constraints increases quadratically, making the model more difficult to solve. Second, we can see that the

Table 15
Comparison between results obtained by exact expressions and approximate expressions.

Data	Exact cost (1)		Approximate cost (2)		Actual cost	Deviation	
	Result	CPU (s)	Result	CPU (s)	Result	With (1) (%)	With (2) (%)
fre9 * 3	222	0.14	440	0.17	600	170.27	26.67
fre15 * 3	644	0.18	1000	0.17	930	44.41	7.53
fre21 * 4	856	0.36	2350.3	0.19	3236	278.04	27.37
fre24 * 4	2796	3.92	3844.8	0.2	4155	48.61	7.47
fre27 * 4	2677	12.21	4764	0.19	4839	80.76	1.55
sta9 * 3	200	0.03	581	0.19	308	54.00	88.64
sta15 * 3	1664	0.08	2630	0.17	2751	65.32	4.40
sta21 * 4	5094	1.95	6624	0.19	6553	28.64	1.08
sta24 * 4	2790	3.42	5515	0.17	6176	121.36	10.70
sta27 * 4	3304	6.26	6384	0.17	6894	108.66	7.40

Table 16
Middle size results due to changing the weighting parameter.

Data	0		0.5		1		10		100	
	Total	Conflict	Total	Conflict	Total	Conflict	Total	Conflict	Total	Conflict
n ∗ m	122.5	122.5	131.5	122.5	140.5	122.5	302.5	122.5	1922.5	122.5
ran30 ∗ 3sca	99.5	99.5	107.5	99.5	115.5	99.5	210.9	110.9	1110.9	110.9
ran39 ∗ 4sca	119.4	119.4	130.3	120.3	139.5	121.5	272.8	132.8	1532.8	132.8
ran42 ∗ 4sca	140.4	140.4	153.6	140.6	166.6	140.6	333.8	193.8	1593.8	193.8
ran48 ∗ 4sca	160.4	160.4	177.4	160.4	192.8	164.8	395	195	2195	195
ran54 ∗ 4sca	89.9	89.9	101.9	89.9	112.8	92.8	219.1	139.1	939.1	139.1
ran57 ∗ 5sca	109.9	109.9	122.9	110.9	134.7	112.7	266.1	146.1	1346.1	146.1
ran60 ∗ 5sca	123.1	123.1	137.2	123.2	150	126	271.3	171.3	1171.3	171.3
ran66 ∗ 5sca	131.9	131.9	146.5	133.5	159.3	136.3	296.2	196.2	1196.2	196.2
ran72 ∗ 5sca	130.6	130.6	149	131	165.2	133.2	322.6	202.6	1222.6	222.6

deviation (1) is rather large; some even over 100%, irrespective of problem size. It indicates the results obtained from the approximate model are much worse than the one obtained by the exact model. The deviation (2) is not as large as deviation (1), but the value of 10% on average also indicates the approximate costs based on average distances may not be a proper estimate of the actual costs. Note that the deviation calculated by Eq. (20) is evaluated by absolute value since this index is used to evaluate the deviation of actual cost (using actual distance between gate pairs and the assignment obtained by M4) to the objective cost obtained by M4 directly. As we explained in last paragraph, the approximate costs are larger than actual costs for some instances (fre15 ∗ 3, sta9 ∗ 3, sta21 ∗ 4) and smaller for other instances.

5.5. Relationship of tows and robustness

In Section 3.4, we indicate that appropriate arrangements of parking activities to parking apron could relieve the overcrowding situation in gate assignment, whereas this may induce some facility and personnel tow costs. In this Section, the experiments are conducted to check how unit tow cost effects the expected gate conflict cost and their distribution in the total costs.

The data sets used for this experiment consist of three parts. The first one includes 10 middle size "ran" and "sca" problems with activity size to be in the range of [30, 75]. We do not consider the data set with sufficient gate resource here because moving aircraft during parking period to parking apron is only meaningful when the gate resources are inadequate and the flight assignments are tight. If there are sufficient gate resources, the aircraft could find a new gate easily when its scheduled gate is occupied due to disruptions. The second one includes 4 large size "sta" problems. The third one includes 4 large size "ran" problems. We make comparison between the second and third data set to explore the effect of parking activities duration with respect to the relationship between conflict and tow costs.

To make the relationship clearer, we fix the coefficient $\alpha = 1,\ \gamma = 0$ and vary β to rule out the effect of the third objective (transfer distance) and focus on the evaluation of first two objectives (conflict cost and tow). We test 5 scenarios with $\beta = 0, 0.5, 1, 10, 100$. The results of 10 middle size problems are shown in Table 16.

Table 17 shows the rate of increase of the total cost ("ttl%") and conflict cost ("cfl%") as β increases compared to $\beta = 0$. Other values of β can be similarly analyzed but not addressed here. We extract four data to draw Fig. 19 to show the trend as the problem size grows.

In Table 16, column "total" is the total objective value, "conflict" is the cost of the first objective. We can see from most of the data samples that when β grows larger, conflict cost grows larger but growing not as fast as the total cost does. Because the total cost consists of two parts: tow cost and conflict cost, both of them are increased due to the change of β. In general, larger problems show a higher increase in conflict cost than smaller problems. The extreme example is data 30 ∗ 3sca, in which the conflict cost stays the same no matter how dramatically the β changes. The reason may be for small problems, there are few potential combinations of assignment. When unit tow cost grows larger, there are very limited spaces for re-assigning the activities. As a result, a large unit cost has minor impact on the conflict cost. The addition of total cost is the same to the addition of tow cost. On the other hand, for large problems, there are more activities and gates, when unit tow cost increases, other assignments that tend to make the assignment more crowded and more expected conflict may be discovered to relieve the effect of tow cost. The trend is proven in Fig. 19. Firstly, in all the cases, the proportion of tow cost is growing as unit tow cost increases. Although the conflict cost also increases as unit tow cost gets larger, the gate resource is so scarce that it does not grow as fast as the tow cost, so the proportion of conflict cost decreases. Second, the percentage of conflict cost decreases much more dramatically in smaller instances than larger ones. However, there are also some exception cases. For example, the data ran60 ∗ 5sca, when $\beta = 100$, the percentage of conflict cost is 10.9%, which is lower than both ran48 ∗ 4sca (12.2%) and ran75 ∗ 5sca (18.2%). After checking the data structure, we found that ran60 ∗ 5sca have fewer gate resources than the other two data. Even though β grows larger and the cost of tows are expensive, the gates have no more room for these parking activities and therefore lead to high total costs.

Table 17
Total cost and conflict cost variation due to changing the weighting parameter.

Data	0.5		1		10		100	
	ttl%	cfl%	ttl%	cfl%	ttl%	cfl%	ttl%	cfl%
30 ∗ 3sca	7.3	0.0	14.7	0.0	146.9	0.0	1469.4	0.0
39 ∗ 4sca	8.0	0.0	16.1	0.0	112.0	11.5	1016.5	11.5
42 ∗ 4sca	9.1	0.8	16.8	1.8	128.5	11.2	1183.8	11.2
48 ∗ 4sca	9.4	0.1	18.7	0.1	137.7	38.0	1035.2	38.0
54 ∗ 4sca	10.6	0.0	20.2	2.7	146.3	21.6	1268.5	21.6
57 ∗ 5sca	13.3	0.0	25.5	3.2	143.7	54.7	944.6	54.7
60 ∗ 5sca	11.8	0.9	22.6	2.5	142.1	32.9	1124.8	32.9
66 ∗ 5sca	11.5	0.1	21.9	2.4	120.4	39.2	851.5	39.2
72 ∗ 5sca	11.1	1.2	20.8	3.3	124.6	48.7	806.9	48.7
75 ∗ 5sca	14.1	0.3	26.5	2.0	147.0	55.1	836.1	70.4

We can also observe from the two tables that for the larger problems, the total cost increment ratio as β grows is smaller than that in the smaller size problems. As we mentioned in the previous paragraph, for large problems, the parking activities are more likely to find a gate position immediately following its arrival flight or immediately before its departure flight of the same aircraft by re-arranging other activities, thus saving some tows.

Table 18 shows the rate of increase of the total cost and conflict cost with different β values compared to $\beta = 0$. For the "sta" data

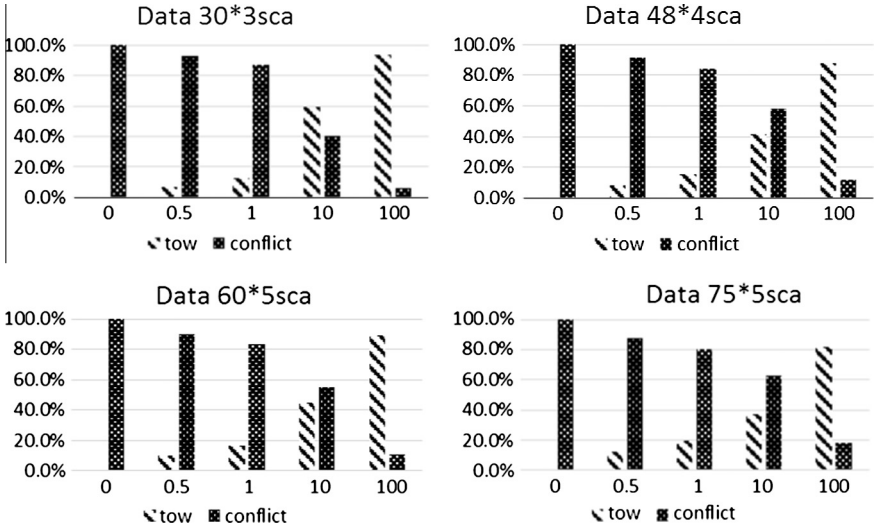

Fig. 19. The cost distribution of different size problems.

Table 18
Large size results: total and conflict cost variations due to changing weighting parameter.

Data	0.5		1		10		100	
	ttl%	cfl%	ttl%	cfl%	ttl%	cfl%	ttl%	cfl%
sta90 * 10	105.0	23.8	183.1	33.1	817.5	442.5	4192.5	442.5
sta105 * 10	67.1	6.3	124.1	12.7	792.9	185.3	4948.1	391.1
sta120 * 12	50.3	6.2	92.4	11.3	573.0	149.7	3477.4	302.8
sta135 * 12	32.4	1.0	61.8	7.8	443.1	112.4	2980.9	195.8
ran90 * 10	114.3	19.0	186.9	67.9	636.9	398.8	2779.8	398.8
ran105 * 10	17.0	0.0	32.5	6.0	139.5	63.9	820.0	63.9
ran120 * 12	35.0	1.7	68.2	1.7	447.6	132.9	2671.5	224.0
ran135 * 12	13.8	2.7	23.8	6.0	45.4	45.4	45.4	45.4

set, the total cost shows similar trend with previous middle cases that in larger problems the total cost increases less than that in smaller ones as unit tow cost grows. However, the percentage of conflict cost grows slower when problem size grows larger, which is different from the phenomena shown in Table 16. This may be due to the fact that the gate resources are not as scarce as reflected in the first data set. When unit tow cost grows larger, some parking activities are assigned to a gate instead of the parking apron to alleviate the influence of tow cost. This does not make schedules much more crowded, therefore the conflict cost does not increase significantly. For the "ran" data set, the phenomenon is less consistent with the problem size. Generally, the increasing unit tow cost has a more significant impact on "sta" set than "ran" set. Specifically, in "sta" set, the total cost and conflict cost increase more dramatically when unit tow cost increases. This may be due to that when unit tow cost increases, the solution tends to choose the one with fewer tows and the one that arranges the parking activities to the gate which is the same as its previous arrival flight or the following departure flight. The long parking time aircrafts will take up more gate resources and make the assignment more crowded than the random or short parking time ones. More crowded assignments in "sta" set means they have a higher gate conflict cost and therefore a higher total cost.

5.6. Managerial implications

5.6.1. Measurement of passenger transfer cost

First, previous research suggested using approximate distance instead of actual distance in order to improve computation efficiency. However, as shown in Section 5.4, evaluating the passenger transfer cost according to the exact transit distance between the arrival gate to the departure gate is necessary.

Second, as the walking distance directly affects the transit time, transfer passengers may miss the second connecting flight due to insufficient connection time and over-long walking distance. Considering this factor, origin or destination passengers are much less sensitive to the walking distance than transfer passengers. Therefore, the major focus of airport managers should be put on the walking distance of transfer passengers.

5.6.2. Maximizing robustness by reducing tow costs

The schedules are more robust to disruptions if arrival and departure activities could be separated and assigned to different gates. However, according to Section 5.5, as tow cost grows, gate swift will make the robustness "expensive" since it costs more aircraft moving fees. The tow cost includes personnel cost, facility cost, the impact on ramp area, etc. If managers can reduce tow costs, schedules with higher robustness can be obtained by incorporating appropriate number of gate swifts.

5.6.3. Selection of solution methods when determining schedules

The selection of solution methods should depend on the problem size. If a schedule is required on a whole day flights, efficient methods like LB, VRNS, diving and RINS should be selected. Among them, diving could obtain fast and acceptable solutions, which is especially suit for the situation that time for schedule is tight. LB and RINS could obtain better solutions when given sufficient time.

When only a subset of flight within a period is included, using CPLEX solving model M3 is a good selection.

6. Conclusion and future research

In this paper, we consider the gate assignment problem with three objectives: robustness, tow cost and transfer distance. Robustness is measured by the expected conflict time between aircraft schedules, tow cost is used to measure the facility and personnel cost when moving the aircraft during its parking in the airport, and transfer distance measures the passenger satisfaction level. The original problem is formulated as a quadratic model to precisely evaluate the transfer passengers' satisfaction level at the airport. The objective related to transfer passengers' walking distance or transit time is quadratic in nature, whereas, it is usually modeled as an approximate linear form for model convenience and solution efficiency. However, according to our numerical analysis, the quadratic representation is necessary to precisely evaluate the satisfaction level of transfer passengers. As the quadratic terms make the model hard to solve, we re-formulate the problem to an equivalent MIP model. Our experimental results show that this model is much more efficient than the original quadratic one as well as the linearized model previously proposed in literature. To solve larger problems with this general MIP model, we propose to adopt three proven and computation-efficient methods, diving, RINS and local branching. In addition, we integrate the advantages from Diving and RINS algorithms and propose a new algorithm VRNS. We tested these four algorithms in numerical experiments and analyzed their performances.

In addition, we also performed experiments to prove the importance of exact quadratic expressions compared to the approximate ones for transfer passengers' walking distance. The results show that quadratic terms are necessary to represent the transfer passengers' experience in the airport. Interesting relationships between robustness and tow cost are also shown in our experiments.

Future research will focus on the dynamic re-assignment of gates. In practice, due to the uncertainty related to flight schedule and airport condition, the occurrence of unexpected events may causes disruptions making the initial schedule infeasible. The reassignment problem requires highly computation efficient methods to meet the stringent time requirement. We therefore look into the development of an efficient algorithm for re-assigning the flights to gates in real-time.

References

Achterberg, T., & Berthold, T. (2007). Improving the feasibility pump. *Discrete Optimization, 4*(1), 77–86.

Babic, O., Teodorovic, D., & Tošic, V. (1984). Aircraft stand assignment to minimize walking. *Journal of Transportation Engineering, 110*(1), 55–66.

Belobaba, P., Odoni, A., & Barnhart, C. (2009). *The global airline industry* (Vol. 23). John Wiley & Sons.

Bertacco, L., Fischetti, M., & Lodi, A. (2007). A feasibility pump heuristic for general mixed-integer problems. *Discrete Optimization, 4*(1), 63–76.

Bihr, R. A. (1990). A conceptual solution to the aircraft gate assignment problem using 0, 1 linear programming. *Computers & Industrial Engineering, 19*(1), 280–284.

Bolat, A. (1999). Assigning arriving flights at an airport to the available gates. *Journal of the Operational Research Society*, 23–34.

Bolat, A. (2000). Procedures for providing robust gate assignments for arriving aircrafts. *European Journal of Operational Research, 120*(1), 63–80.

Bonami, P., Cornuéjols, G., Lodi, A., Margot, F., et al. (2009). A feasibility pump for mixed integer nonlinear programs. *Mathematical Programming, 119*(2), 331–352.

Braaksma, J. (1977). Reducing walking distances at existing airports. In *Airport forum*.

D'Ambrosio, C., Frangioni, A., Liberti, L., & Lodi, A. (2012). A storm of feasibility pumps for nonconvex MINLP. *Mathematical Programming, 136*(2), 375–402.

Danna, E., Rothberg, E., & Le Pape, C. (2005). Exploring relaxation induced neighborhoods to improve MIP solutions. *Mathematical Programming, 102*(1), 71–90.

Ding, H., Lim, A., & Rodrigues, B. (2004a). New heuristics for over-constrained flight to gate assignments. *Journal of the Operational Research Society, 55*(7), 760–768.

Ding, H., Lim, A., & Rodrigues, B. (2004b). Aircraft and gate scheduling optimization at airports. In *Proceedings of the 37th annual Hawaii international conference on system sciences, 2004*. IEEE.

Ding, H., Lim, A., Rodrigues, B., & Zhu, Y. (2005). The over-constrained airport gate assignment problem. *Computers & Operations Research, 32*(7), 1867–1880.

Dorndorf, U. (2002). *Project scheduling with time windows: From theory to applications; with 17 Tables*. Springer.

Dorndorf, U., Jaehn, F., & Pesch, E. (2008). Modelling robust flight-gate scheduling as a clique partitioning problem. *Transportation Science, 42*(3), 292–301.

Drexl, A., & Nikulin, Y. (2008). Multicriteria airport gate assignment and Pareto simulated annealing. *IIE Transactions, 40*(4), 385–397.

Fischetti, M., Glover, F., & Lodi, A. (2005). The feasibility pump. *Mathematical Programming, 104*(1), 91–104.

Fischetti, M., & Lodi, A. (2003). Local branching. *Mathematical Programming, 98*(1–3), 23–47.

Fischetti, M., & Lodi, A. (2008). Repairing MIP infeasibility through local branching. *Computers & Operations Research, 35*(5), 1436–1445.

Fischetti, M., Polo, C., & Scantamburlo, M. (2004). A local branching heuristic for mixed-integer programs with 2-level variables, with an application to a telecommunication network design problem. *Networks, 44*(2), 61–72.

Fischetti, M., & Salvagnin, D. (2009). Feasibility pump 2.0.. *Mathematical Programming Computation, 1*(2–3), 201–222.

Ghosh, S. (2007). DINS, a MIP improvement heuristic. In *Integer programming and combinatorial optimization* (pp. 310–323). Springer.

Hansen, P., Mladenović, N., & Urošević, D. (2006). Variable neighborhood search and local branching. *Computers & Operations Research, 33*(10), 3034–3045.

Hassounah, M. I., & Steuart, G. N. (1993). Demand for aircraft gates. *Transportation Research Record* (1423).

Hu, X.-B., & Di Paolo, E. (2007). An efficient genetic algorithm with uniform crossover for the multi-objective airport gate assignment problem. In *IEEE congress on evolutionary computation, 2007, CEC 2007*. IEEE.

Ibm, I. (2010). *CPLEX optimizer*.

Jo, G.-S., Jung, J.-J., & Yang, C.-Y. (1997). Expert system for scheduling in an airline gate allocation. *Expert Systems with Applications, 13*(4), 275–282.

Kim, S. H. (2013). *Airport control through intelligent gate assignment*.

Kim, S. H., & Feron, E. (2013). Numerical analysis of gate conflict duration and passenger transit time in airport. Available from arXiv:arXiv:1308.6217.

Kim, S. H., Feron, E., Clarke, J. P., Marzuoli, A. (2013). Airport gate scheduling for passengers, aircraft, and operation. Available from arXiv:arXiv:1301.3535.

Liang, Z., & Chaovalitwongse, W. A. (2012). A network-based model for the integrated weekly aircraft maintenance routing and fleet assignment problem. *Transportation Science, 47*(4), 493–507.

Mangoubi, R., & Mathaisel, D. F. (1985). Optimizing gate assignments at airport terminals. *Transportation Science, 19*(2), 173–188.

Rodríguez-Martín, I., & Salazar-González, J. J. (2010). A local branching heuristic for the capacitated fixed-charge network design problem. *Computers & Operations Research, 37*(3), 575–581.

Wirasinghe, S., & Bandara, S. (1990). Airport gate position estimation for minimum total costs—Approximate closed form solution. *Transportation Research Part B: Methodological, 24*(4), 287–297.

Xu, J., & Bailey, G. (2001). The airport gate assignment problem: Mathematical model and a tabu search algorithm. In *Proceedings of the 34th annual Hawaii international conference on system sciences, 2001*. IEEE.

Yan, S., & Huo, C.-M. (2001). Optimization of multiple objective gate assignments. *Transportation Research Part A: Policy and Practice, 35*(5), 413–432.

Yan, S., & Tang, C.-H. (2007). A heuristic approach for airport gate assignments for stochastic flight delays. *European Journal of Operational Research, 180*(2), 547–567.

Zhang, S., Cesarone, J., & Miller, F. (1994). A comparative study of an aircraft assignment problem at a large aircraft. *International Journal of Industrial Engineering, 1*(3), 203–212.