PERGAMON

Transportation Research Part A 35 (2001) 413–432

TRANSPORTATION RESEARCH PART A

www.elsevier.com/locate/tra

Optimization of multiple objective gate assignments

Shangyao Yan *, Cheun-Ming Huo

Department of Civil Engineering, National Central University, Chungli 32054, Taiwan, ROC

Received 13 April 1999; received in revised form 14 September 1999; accepted 4 October 1999

Abstract

This paper proposes a multiple objective model to help airport authorities to efficiently and effectively solve gate assignment problems. The model is formulated as a multiple objective zero–one integer program. To efficiently solve large-scale problems in practice, we used the weighting method, the column generation approach, the simplex method and the branch-and-bound technique to develop a solution algorithm. To test how well the model may be applied in actual operations, a case study regarding the operation of Chiang Kai-Shek (CKS) Airport was performed. The results show that the model could be useful for actual operations. © 2001 Elsevier Science Ltd. All rights reserved.

Keywords: Airport; Gate assignment; Multiple objective; Weighting method; Column generation

1. Introduction

Because of recent deregulation, market demand has significantly grown for air carriers in Taiwan. Many new airlines (both domestic and foreign) are flying into Taiwan. Since the airport facilities, particularly the gates, were not originally designed to handle this increased traffic volume, Taiwan's airports have become more and more crowded. The demand is expected to further increase in the future. As a result, the level of service is expected to worsen in terms of congestion and delay.

While the long-term physical expansion of airport facilities has been planned, this cannot solve near future airport congestion problems. One short-term solution, which can be considered to alleviate airport capacity constraints, is to optimize the assignment of flights to the existing gates. Unfortunately, the Taiwan airport authority's current approach is to assign flights to gates manually without system analysis, which makes it difficult to handle large-scale problems. Thus,

* Corresponding author. Tel.: +11-886-3-422-7151; fax: +11-886-3-425-2960.
E-mail address: t320002@sparc20.ncu.edu.tw (S. Yan).

an efficient and effective tool for the optimization of the flight-to-gate problem is important for Taiwan both now and in the future.

Researchers have developed a few analytical models to help assign daily flights more effectively to gates; for example, Babic et al. (1984), Mangoubi and Mathaisel (1985) and Vanderstraetan and Bergeron (1988). These models are normally formulated as zero–one integer programs. The objective functions are usually the minimization of either the total passenger walking distance or the number of off-gate events. Two classes of constraints are commonly required in the modeling:

1. every flight must be assigned to one and only one gate;
2. no two aircrafts may be assigned to the same gate concurrently.

Some special constraints have been also introduced because of an airport‘s configuration. For example, if a large-type of aircraft is assigned to a gate, then the same type of aircraft cannot be assigned to the neighboring gates (Vanderstraetan and Bergeron, 1988). Such models have been solved using either the simplex method (Mangoubi and Mathaisel, 1985), the branch-and-bound technique (Babic et al., 1984) or a problem-oriented heuristic (Vanderstraetan and Bergeron, 1988). Because optimal solutions to large-scale zero–one integer problems are generally difficult to solve, heuristics have been suggested by some researchers (for example, Mangoubi and Mathaisel, 1985) to deal with this problem.

To improve the solution efficiency of the past models, Yan and Chang (1998) recently introduced a new network model formulated as an integer multiple commodity network flow problem. They also developed a Lagrangian relaxation-based algorithm for the optimization of gate assignment problems. Although this model and the solution algorithm were shown to be more effective than Mangoubi and Mathaisel's (1985), they still need to be improved. First, though this model explored the merits of a network flow, to efficiently solve large-scale gate assignment problems, the convergence gap for the solution algorithm was still significant, more than 5%.

Second, the objective function was the minimization of the total passenger walking distance inside the terminal to reach their departure gate, the baggage claim area or a connecting flight. It did not consider flight delays usually used in practice, in the gate assignment process. We note that, in reality, arriving airplanes are temporarily held on aprons if there is no available gate and hence passengers are forced to wait on the airplane. Without considering flight delays in gate assignment, the model does not yield a feasible solution if there are more flights than gates in a time slot (particularly during peak hours). Thus, the total passenger waiting time on an aircraft may be another important criterion for gate assignment.

To improve Yan and Chang's (1998) model, we develop a new gate assignment model that is formulated as a multiple objective zero–one integer program. In practice, to efficiently solve the model for large-scale problems and to improve Yan and Chang's algorithms, we applied the column generation approach coupled with the simplex method, the branch and bound technique and the weighting method, to develop a solution algorithm.

Note that as well as analytical models, there have been some other recently developed approaches to solving gate assignment problems. For example, Hamzawi (1986) developed a simulation model to solve the gate assignment problem; Gosling (1990) and Su and Srihari (1993) developed different expert systems for aircraft gate assignments. These approaches are, however, not the same as those introduced in this paper. The rest of the paper is organized as follows. First, we introduce our model. Then we formulate the model as a multiple objective zero–one integer program. A solution algorithm is developed thereafter. Finally, we perform a case study to test the model in the real world.

2. The model

Airport congestion is a common problem in many countries. In short-term operations, if both the passenger walking distance and the waiting time can be evaluated together in the gate assignment process, then the assignment can better reflect practical needs than when evaluated independently. Here we develop a dual-objective gate assignment model, which considers the minimization of both the passenger walking distance and their waiting time. Before formulating the problem, suppose that we have the following information:

1. The layout of the terminal area and the number of gates.
2. Flight schedules indicating when each aircraft arrives at and departs from the airport.
3. The aircraft type associated with each flight.
4. The number of passengers for each flight, including arriving, departing and transferring passengers.
5. The buffer time, which is used for resolving minor delays that often occur in real-time operations. A buffer time is added between two continuous flights that are assigned to the same gate. For example, the buffer time is traditionally set as 30 min for international flights at Taiwan's Chiang Kai-Shek (CKS) airport.
6. The walking distances to each gate, including the distance for an arriving passenger, a departing passenger or a transferring passenger. In particular, the walking distance for an arriving passenger is the measured distance between the gate and the baggage claim area, the walking distance for a departing passenger is the distance between the check-in counter and the gate, and for modeling simplification, the distance for a transferring passenger is determined using a uniform probability distribution between one gate and all other gates (Mangoubi and Mathaisel, 1985). Consequently, the transferring distance is equal to the average distance from one gate to every other gate. Note that the uniform distribution assumption may not be realistic, as has been mentioned by Mangoubi and Mathaisel (1985). For example, the attractive gates close to the main hall tend to be clustered together, so a transfer passenger's walking distance between these gates is shorter. However, the use of such an assumption to derive the transferring distance is justifiable (Mangoubi and Mathaisel, 1985).

The assignment of flights to gates is daily. A non-home base airplane, that arrives late and departs the next day (usually in the early morning), is typically assigned to two gates, one for today's arrival and the other for tomorrow's departure. Both assignments are independent. However, if these two gates are not the same, if the airplane could be held overnight at the arrival gate without affecting the assignment of other flights the following day, then the airplane could stay at this gate until its departure. On the other hand, if holding the airplane overnight at the arrival gate will affect the assignment of other flights the following day, then this aircraft should be moved to some space (for example, a temporary apron) until it is moved to its designated departure gate.

To evaluate gate assignment flight delays, as shown in Fig. 1, we use a time window to represent a flight's gate assignment. In particular, Fig. 1(a) shows the time window for an on-time flight gate

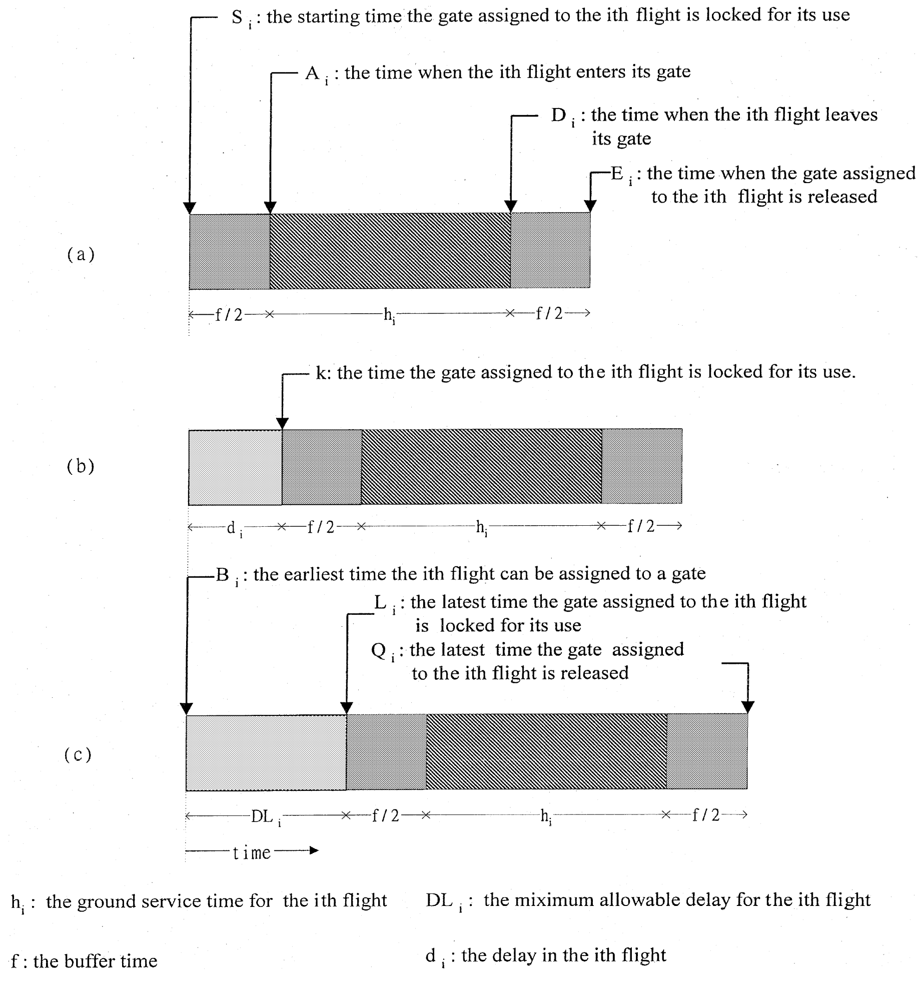
S i : the starting time the gate assigned to the ith flight is locked for its use
A i : the time when the ith flight enters its gate
D i : the time when the ith flight leaves its gate
E i : the time when the gate assigned to the ith flight is released
(a)
f / 2
h i
f / 2
k: the time the gate assigned to the ith flight is locked for its use.
(b)
d i
f / 2
h i
f / 2
B i : the earliest time the ith flight can be assigned to a gate
L i : the latest time the gate assigned to the ith flight is locked for its use
Q i : the latest time the gate assigned to the ith flight is released
(c)
DL i
f / 2
h i
f / 2
time
h i : the ground service time for the ith flight
DL i : the miximum allowable delay for the ith flight
f : the buffer time
d i : the delay in the ith flight

Fig. 1. The time window for a flight during gate assignment.

assignment. The time window contains half the buffer time, a ground service time and the other half of the buffer time. The ground service time for a flight is from the time when the aircraft serving this flight enters the gate, to the time when the aircraft leaves the gate.

Specifically, the ground service time for an arriving flight is from its arrival time to the time when all its passengers and packages have deplaned. The ground service time for a departing flight is from the time that the airplane is prepared for this flight (including investigation time before departure, fueling, passenger/package boarding) to the actual departure time. The ground service time for a transferring flight-pair (an arrival flight and a departure flight) is from the arrival time of the arriving flight to the departure time of the departing flight. Note that the two flights in a flight-pair are shortly connected, usually within one hour in current CKS Airport operations.

In Fig. 1(a), the start for the time window (S_i) represents the starting time when the gate is locked for the ith flight use. The start of the ground service time (A_i) represents the time when the aircraft that serves the ith flight enters the gate. The end of the ground service time (D_i) denotes the time when the aircraft that serves the ith flight leaves the gate. The end of the time window (E_i) represents the time when the gate assigned to the ith flight is released.

If an aircraft is held on the ground for waiting, the time window for the flight is changed to that shown in Fig. 1(b). In particular, a delay is added at the beginning of the normal time window. The start of the normal time window (k) represents the starting time when the gate is locked for the ith flight. To prevent a long waiting time for passengers, which is not practical in actual operations, we set a maximum allowable gate assignment waiting time and add this to the beginning of the normal time window, as shown in Fig. 1(c). Note that in practice the length of the maximum allowable waiting time can be adjusted based on the user's needs. The start of the entire time window (B_i) represents the earliest time when the flight can be assigned to a gate. The start of the normal time window (L_i) indicates the latest time when the gate assigned to the flight is locked for use. The end of the entire time window (Q_i) represents the latest time when the gate assigned to the ith flight is released. Note that normal time windows for flights assigned to a gate should not overlap.

Using h_i, DL_i, f, d_i and the calculations shown in Fig. 1, we can determine all possible time windows assigned for every flight. For example, given a flight i, the earliest time window (that is without delay, $d_i = 0$) permitted for the assignment of flight i is from B_i to $B_i + h_i + f$. The latest time window (that is with the maximum allowable delay, $d_i = DL_i$) is from L_i $(= B_i + DL_i)$ to Q_i $(= L_i + h_i + f)$. The time window with delay d_i is from $B_i + d_i$ to $B_i + d_i + h_i + f$. Note that the starting time of any possible time window assigned for flight i is between B_i and L_i.

The model is formulated as follows:

$$\text{Min } Z_1 = \sum_{i=1}^{M}\sum_{j=1}^{N}\sum_{k=B_i}^{L_i}(c_{ij} - x_{ijk}), \tag{1a}$$

$$\text{Min } Z_2 = \sum_{i=1}^{M}\left(P_i\left(\sum_{j=1}^{N}\sum_{k=B_i}^{L_i}kx_{ijk} - B_i\right)\right), \tag{1b}$$

s.t.

$$\sum_{j=1}^{N}\sum_{k=B_i}^{L_i}x_{ijk} = 1 \quad \forall i, \tag{1c}$$

$$\sum_{i\in F_j}\sum_{k\in H_{is}}x_{ijk} \leqslant 1 \quad \forall j,\ \forall S, \tag{1d}$$

$$x_{ijk} = 0 \text{ or } 1 \quad \forall i,\ \forall j,\ \forall k, \tag{1e}$$

where

i	the ith flight (or a flight pair that has to be assigned to the same gate, below we call it a flight for simplification)
j	the jth gate
k	the kth time point
M	the set of all flights
N	the set of all gates

Z_1	total passenger walking distance
Z_2	total passenger waiting time
P_i	total number of passengers on the ith flight
B_i	the earliest time that the ith flight can be assigned to a gate
L_i	the latest time that the ith flight should be assigned to a gate
c_{ij}	total walking distance for passengers on the ith flight assigned to the jth gate
x_{ijk}	the decision variable; 1 if the ith flight is assigned to the jth gate at the kth time; 0 otherwise
F_j	the set of all flights that can be assigned to the jth gate
H_{is}	the set of all permitted times assigned for the ith flight, where the resulting time windows will cover the sth time, $H_{is} = \{k \mid B_i \leqslant k \leqslant L_i,\ k \leqslant s \leqslant k + h_i + f\}$

The model is now a multiple objective zero–one integer program. Objective (1a) represents the minimum total passenger walking distance. Objective (1b) represents the minimum total passenger waiting time. Since the waiting time for a passenger on the ith flight is equal to

$$d_i = \sum_{j=1}^{N}\sum_{k=B_i}^{L_i} kx_{ijk} - B_i$$

and there are P_i passengers on the ith flight, the total passenger waiting time is equal to

$$Z_2 = \sum_{i=1}^{M} P_i\left(\sum_{j=1}^{N}\sum_{k=B_i}^{L_i} kx_{ijk} - B_i\right).$$

Constraint (1c) denotes that every flight must be assigned to one and only one gate. Constraint (1d) ensures that at most one aircraft is assigned to every gate in every time window. In particular, constraint (1d) shows that for the sth time the sum of all permitted assignments of all flights, whose time windows cover the sth time, to the jth gate has to be less than or equal to one, that is,

$$\sum_{i\in F_j}\sum_{k\in H_{is}} x_{ijk} \leqslant 1 \quad \forall j,\ \forall s.$$

Constraint (1e) indicates that all the decision variables are either zero or one.

Note that the number of constraints (1d) is significantly large if all gates and all times are counted. For example, if there are 24 gates (as for current CKS Airport operations) and 228 times (assume that 5 min is used as a check time point in a 19 h operating day); then the total number of constraints (1d) is $24 \times 228 = 5472$. Certainly, if the number of gates is increased in the future, then the number of constraints will also be increased. As in Mangoubi and Mathaisel (1985), many of these constraints may be redundant. Because deleting all redundant constraints using a complete enumeration approach is time-consuming, we have developed a scanning approach with the notations shown in Fig. 1 to efficiently "catch" all the effective constraints.

In should be noted that the model might not return a feasible solution within a given maximum allowable waiting time (DL_i) if there are too many flights crowded into a time slot. To deal with this, the user may increase the maximum allowable waiting time (DL_i) if possible, to make a feasible solution more likely. If there is no feasible solution with the adjustment of the maximum allowable waiting time, this means that the number of currently available gates is not enough to service the scheduled flights. The airport authorities may thus consider reallocating airport time slots (or landing\taking-off flight quotas) for airlines using this airport or design temporary gates to use for short-term operations. In long-term operations, more gates may be constructed to relief the congestion.

3. Solution algorithms

In this section we develop a solution algorithm to solve the multiple objective zero–one integer program which is characterized as NP-hard in terms of optimization (Garey and Johnson, 1979). We first apply the weighting method to transfer the multiple objective problem into several single objective problems. Then we apply the column generation approach to develop an efficient algorithm to solve the single objective problem. Finally, we propose two methods to efficiently solve single objective problems with various weighting vectors.

3.1. Solution method for multiple objective functions

In general, multiple objective mathematical programs and their solution methods can be classified into four categories (Hwang, 1979). Of these, the weighting method is more popular and is usually applied in situations when the decision maker provides his/her preference information before the problem is solved. Since the airport authority can always specify preference in advance, we suggest using the weighting method to handle problems in this research. However, whether or not there are more suitable methods, can be a direction of future research.

Using the weighting method, the user can set two weights on the basis of his/her preference for two objective functions, then transfer the two objective functions into a weighted objective function. In our research a weighting vector is used to represent a combination of two weights. The user can set the various weighting vectors, then solve for the optimal solution for each one. Note that the optimal solution for a weighting vector is a non-dominated solution of the multiple objective problem. For example, the user can set weighting vectors such as: $\{1,0\}$, $\{0.9,0.1\}$, $\{0.8,0.2\},\ldots,\{0.1,0.9\}$ and $\{0,1\}$, then solve for 11 non-dominated solutions for reference.

To simplify the calculation, we unify the units of both objective functions. In particular, we divide the passenger distance by the average passenger walking speed. Thus, both units are time units. Using the weighting method and the unit transformation, the two objective functions are transformed into a weighted objective function as shown in formula (2) or (3):

$$\text{Min } Z = \alpha\sum_{i=1}^{M}\sum_{j=1}^{N}\sum_{k=B_i}^{L_i}(c_{ij}x_{ijk})/S_{\text{ave}} + (1-\alpha)\sum_{i=1}^{M}\left(P_i\left(\sum_{j=1}^{N}\sum_{k=B_i}^{L_i}kx_{ijk} - B_i\right)\right), \tag{2}$$

$$\text{Min } Z' = \sum_{i=1}^{M}\sum_{j=1}^{N}\sum_{k=B_i}^{L_i}(c_{ij}x_{ijk})/S_{\text{ave}} + (1/\alpha - 1)\sum_{i=1}^{M}\left(P_i\left(\sum_{j=1}^{N}\sum_{k=B_i}^{L_i}kx_{ijk} - B_i\right)\right), \tag{3}$$

where α is the weight for the passenger distance, $0 \leqslant \alpha \leqslant 1$ and S_{ave} is the average passenger walking speed. To further simplify the calculation, we transform formula (3) to formula (4) or (5), where ω represents the relative importance of the waiting time to the walking distance for passengers. For example, when ω equals 1, it means that the importance of the waiting time and the walking distance is equal. When ω is decreased, it means that the relative importance of the waiting time to the walking distance is decreased.

$$\text{Min } Z' = \sum_{i=1}^{M}\sum_{j=1}^{N}\sum_{k=B_i}^{L_i}(c_{ij}x_{ijk})/S_{\text{ave}} + \omega\sum_{i=1}^{M}\left(P_i\left(\sum_{j=1}^{N}\sum_{k=B_i}^{L_i}kx_{ijk} - B_i\right)\right). \tag{4}$$

$$\text{Min } Z' = \sum_{i=1}^{M}\sum_{j=1}^{N}\sum_{k=B_i}^{L_i}\left(\frac{c_{ij}}{S_{\text{ave}}} + \omega P_i k\right)x_{ijk} - \omega\sum_{i=1}^{M}P_iB_i. \tag{5}$$

Note that when ω equals 0, the weighted objective function, $\text{Min } Z'$ becomes the first objective function, $\text{Min } Z_1$, that is the minimization of the total passenger walking distance; when ω approaches infinity, the weighted objective function, $\text{Min } Z'$, reduces to the second objective function, $\text{Min } Z_2$, that is the minimization of the total passenger waiting time. This means that our model can be simplified to two single objective decision models, and is therefore useful for those preferring only one of the two objective functions in practice.

Given several weighting vectors, the user can create several single objective problems whose solution algorithm will be addressed in Section 3.2. In practice, after solving for non-dominated solutions, the decision maker can choose his/her preferred solution from these. To provide clearer information for these non-dominated solutions, we can transform their objective values into relative percentages as follows (Hwang, 1979):

$$P_{ij} = \frac{Z_{ij} - Z_i^{\min}}{Z_i^{\max} - Z_i^{\min}} \times 100\% \quad \forall i,\ \forall j, \tag{6}$$

where

i	the ith objective function
j	the jth weighting vector
P_{ij}	the relative percentage of the ith objective function value for the jth weighting vector
Z_{ij}	the ith objective function value for the jth weighting vector
$Z_i^{\min}$	the minimum of the ith objective function values for all weighting vectors
$Z_i^{\max}$	the maximum of the ith objective function values for all weighting vectors

3.2. Solution method for single objective problems

The problem sizes faced here are typically large. For example, if there are 24 gates, 145 flights/flight pairs (for current CKS Airport operations) and 4 delay choices, then the number of total variables will be $22 \times 145 \times 4 = 13{,}920$ which is typically large in terms of the optimization of NP-hard problems. Note that the number can be further increased in the future because the Taiwan government is making efforts to develop Taiwan as an air-hub in East Asia. Since the network model proposed by Yan and Chang (1998) does not resolve the significant convergence gap during their solution process, their method is not appropriate for solving our problems, which are even larger. After considering that there may be an enormous number of variables in our problem now, and in the future, we apply the column generation approach (for example, see Ahuja et al., 1993) to develop a solution algorithm, to efficiently solve the problem.

Note that the column generation approach has been applied in other fields to solve problems with a large number of variables. For example, Lavoie et al. (1988) applied a column generation approach to the solution of crew scheduling problems formulated as a set covering problem. Desrochers and Soumis (1989) applied a column generation approach to solve urban transit crew scheduling problems. However, there has not yet been an application of the column generation approach to the solving of gate assignment problems.

Before developing the algorithm, we note that the column generation approach has been conventionally used for solving linear programs, whose optimal solutions may not be integers. Although the constraint matrix of a gate assignment problem is relatively sparse compared to general linear programs, the linear optimal solution may not be an integer solution (and a non-integer solution is not even a feasible solution), especially when some constraints are tight (for example, when flights are crowded into peak hours). Thus, other approaches, for example, the branch and bound technique (for example, see Sheffi et al., 1990), need to be incorporated to solve for feasible solutions. The steps of the algorithm are as follows:

Step 1: Use a heuristic to generate enough initial columns (variables). Construct the master problem using the initial columns. Let $B=\{\}$, where B is a set of variables that can help improve the objective value of the master problem when they are added to the master problem (let us call these variables "improvable" variables).

Step 2: Solve the master problem using the simplex method.

Step 3: Modify the cost parameters in the sub-problem (see below) using the simplex dual variables obtained from the optimal simplex tableau in step 2.

Step 4: Solve the sub-problem and add improvable variables to B.

Step 5: If $B=\{\}$, then go to step 6; otherwise, add the improvable variables in B to the master problem and let $B=\{\}$, return to step 2.

Step 6: If the optimal linear solution is an integer, then stop, otherwise, go to step 7.

Step 7: Apply the branch-and-bound technique coupled with the column generation approach to find the optimal integer solution.

We describe the three major issues embedded in the above algorithm, specifically the master problem and the sub-problem in Section 3.2.1, the initial heuristic in Section 3.2.2 and the resolution of non-integer solutions in Section 3.2.3.

3.2.1. The master problem and the sub-problem

The master problem is a linear-relaxed version of the original problem with limited columns. We solve the master problem using the simplex method, then check whether we should add more columns to the master problem to improve the objective function value. In particular, the reduced costs for columns not considered in the master problem can be used for the evaluation (Hillier and Lieberman, 1995). To price out the best columns not in the master problem, we construct a sub-problem using the cost of every variable, and the dual variables obtained from the optimal simplex tableau. The reduced cost of every variable can be simply calculated using the sensitivity analysis theory (Hillier and Lieberman, 1995).

If the reduced cost of a variable is less than zero, then this variable is expected to help improve the objective value of the master problem if the variable is added to the master problem. As mentioned above, we call this variable an improvable variable. Otherwise, the variable will not help improve the objective value. In other words, it will remain a non-basic variable if it is added to the master problem. Note that there may be more than one improvable variable not considered in the master problem. In the sub-problem, after many tests, we choose at most one improvable variable with the greatest negative reduced cost for each flight. As a result, B contains at most a number of improvable variables equal to the number of flights in each run.

3.2.2. The initial solution

To improve solution efficiency, before starting to solve the master problem we refer to the heuristic by Yan and Chang (1998), developing a heuristic to generate enough columns to construct an initial feasible solution. The idea is to sequentially assign every flight (in decreasing order of the number of passengers on the flight) to its nearest available gate. Note that for multiple objective functions, we can focus on the first objective function (that is, assuming that there is no flight delay in the initial solution), and then solve for the initial columns.

Suppose that there is no available gate for a flight, we then delay this flight, within the maximum allowable waiting time, to find an available gate, then assign the flight to that gate. If there is still no available gate with such a delaying strategy, we assign this flight to an artificial gate whose passenger walking distance is infinite, meaning that the distance in the initial solution is infinite. Although we did not have such an assignment in the case study described later, this can theoretically insure an initial feasible solution. The steps of this heuristic are listed as follows:

Step 0: Sort all flights in decreasing order of the number of passengers on a flight. Let the first flight be $k = 1$.

Step 1: Let G_k be the set of all gates that are available to the kth flight when it arrives.

Step 2: If G_k is not empty, assign the kth flight to gate s ($s \in G_k$) which is the nearest gate for the kth flight. If G_k is empty, gradually delay the kth flight in order to find the first available flight and assign the kth flight to the gate. If there is still no available gate when the delay reaches the maximum allowable waiting time, assign the kth gate to an artificial gate at an infinite distance.

Step 3: If k is equal to the number of total flights, stop; otherwise, $k = k + 1$, return to step 1.

3.2.3. The resolution of non-integer solutions

If the linear optimal solution obtained is not an integer, we continue to apply the branch-and-bound technique coupled with the column generation approach to solve for the optimal integer solution. The steps are as follows:

Step 0 $(Z^* = Z_{\text{inc}}) : Z_{\text{inc}}$ is the best objective value from all the integer solutions previously obtained. Let the linear relaxed problem be a branched problem and save its optimal linear solution.

Step 1 (*Branching*): Select a branched problem with a minimum objective value that has not been fathomed. Choose a variable with a non-integer value from its optimal linear solution and generate two branched problems by defining the variable to be zero or one.

Step 2 (*Bounding*): For each branched problem, first optimally solve its master problem using the simplex method, then use the column generation technique to add improvable variables from its sub-problem to its master problem, until its linear optimal solution, Z, is found.

Step 3 (*Fathoming*): If any of the following conditions is satisfied, then fathom the branched problem:

1. The branched problem is infeasible.
2. $Z \geqslant Z^*$.
3. The optimal linear solution is an integer. If $Z < Z^*$, then $Z^* = Z$.

Step 4: If $Z^* \leqslant Z_{\text{linear}}$, stop. Z^* is the optimal integer solution, where Z_{linear} is the linear optimal objective value.

Step 5: If there are branched problems that have not been fathomed, go to step 1; otherwise, stop and Z^* is the optimal integer solution.

It should be noted that if the problem scale is too large, it could be too time-consuming to use the branch-and-bound technique (whose complexity is exponential) to solve for the optimal integer solution. In this case, the user can suitably reduce the decision accuracy by modifying the stopping criterion, $Z^* \leqslant Z_{\text{linear}}$ in step 4, to be $Z^* \leqslant (1 + e)Z_{\text{linear}}$, where e is the tolerable error or the user can develop his own heuristics to efficiently yield good near-optimal solutions in practice.

3.3. Repetitive solving for various weighting vectors

Here, in multiple objective programming, we have to solve a number of single objective problems with various weighting vectors. If there are n weighting vectors that need to be solved repetitively, then the simplest way to do this is to independently solve n single objective problems. However, since it may be very time-consuming to solve a single objective problem which is NP-hard, repetitive optimization of n single objective problems would be very inefficient. Although parallel computation could be applied to resolve such problems, not every user has devices available for parallel computation.

To efficiently solve single objective problems in a single-CPU computer, for various weighting vectors, using the column generation approach, we use the optimal basis, and the used columns obtained in the current problem as the initial basis, and the initial columns for the next problem. After many empirical tests we found that many columns for the optimal bases of the 2 weighting vectors were the same. Thus, if we started to solve the next problem using the current optimal basis, then we would begin from a good initial solution which is not usually “far” from the optimal basis of the following problem. Moreover, if we use columns used for the current problem, then we would not generate many more columns in the next problem. As a result, computation time for solving the following problem could be saved. Exploiting this idea, we propose two methods, shown in Fig. 2, to repetitively solve single objective problems with various weighting vectors.

Method A is as follows: Given a weighting vector, we first solve for the linear optimal solution and then the integer optimal solution. We then move to the next weighting vector and modify the cost parameters. Hereafter, we will use the optimal basis and columns used in the previous problem to start solving the current problem. Finally, we repeat the process until we solve all the problems. The steps for method A are as follows:

Step 0: Given a weighting vector, use the heuristic from Section 3.2.2 to find the initial columns.

Step 1: Use the simplex method to solve for the optimal linear solution of the single objective problem.

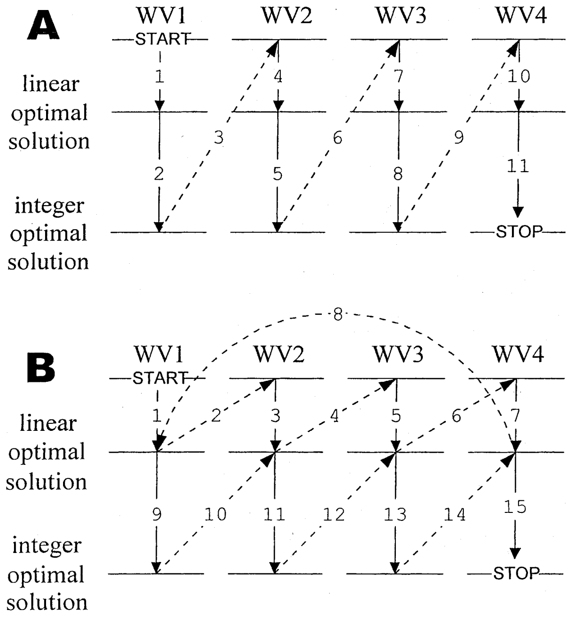

Fig. 2. Repetitive solving of single objective problems for various weighting vectors.

Step 2: Check whether the linear optimal solution is an integer? If yes, we have found the integer optimal solution for this weighting vector and may go to step 4; otherwise, go to step 3.

Step 3: Use the branch-and-bound technique and the column generation approach to solve for the integer optimal solution for this weighting vector.

Step 4: Check whether all the weighting vectors have been solved? If yes, stop the algorithm; otherwise, modify the cost parameters using the next weighting vector. Use the variables used in the current problem as the initial columns for the next problem and return to step 1.

Since the branch-and-bound technique for finding integer solutions may be very time-consuming, to prevent generating too many columns in resolving non-integer solutions we modify steps 2–4 of method A to develop method B. In particular, we repeatedly solve for the optimal linear solutions for all the single objective problems first. Using the variables used in linear optimization as the initial columns, we then apply the branch-and-bound technique to solve for the optimal integer solutions for all single objective problems. The steps in method B are similar to those in method A, except that steps 2–4 are modified as follows:

Step 2: Check whether the linear optimal solutions have been solved for all the weighting vectors? If no, modify the cost parameters using the next weighting vector. Use the variables used in the current problem as the initial columns for the next problem and return to step 1. If yes, go to step 3 and start from the 1st weighting vector.

Step 3: Use the branch-and-bound technique and the column generation approach to solve for the optimal integer solution for this weighting vector.

Step 4: Check whether the integer optimal solutions have been solved for all weighting vectors? If yes, stop the algorithm; otherwise, modify the cost parameters using the next weighting vector. Use the variables used in the current problem as the initial columns for the next problem and return to step 3.

4. Case study

To test how well the model may be applied in the real world, we performed a case study on CKS (an international) Airport operations in Taiwan. To build and to solve the model, we used FORTRAN 90, coupled with the XMP linear programming solver to develop all the necessary programs. The tests were performed on a Pentium 180 with 32 MB of RAM in the environment of Microsoft Windows 95. We first used CKS Airport operating data to build the mathematical model, then applied the solution algorithm to solve the problem. Finally, we performed a sensitivity analysis on the model.

4.1. Model testing and results

All data required for our model, as mentioned in Section 2, were obtained from the airport authorities. In particular, there were 20,341 departure passengers, 15,865 arrival passengers and 1951 transfer passengers in a test day. The types of aircraft operating at the CKS Airport include B747s, A300s, A340s, A320s, A330s, ATR72s, DC10s, L1011s, MD11s, B767s, B757s, B737s and B727s, which are practically classified as two types, large or wide, for gate assignment purposes. There were 24 gates in operation, of which two were temporary. Eight of the 24 gates were only available for wide types of aircraft, while the others were available for all types. There were 35 arrival flights (10 served by large aircraft), 41 departure flights (17 served by large aircraft) and 69 flight pairs (37 served by large aircraft) in a single day's operations. Consequently, a total of 214 flights were tested. Note that each pair of flights served by an airplane, were normally connected within 1 h. Therefore, in practice they had to be assigned to the same gate.

According the airport authority, the buffer time is traditionally set to be 30 min. Also, the ground service time for an arrival flight or a departure flight is set to be 30 min. The total passenger walking distance for each flight assigned to each gate was calculated according to the airport configuration, the number of passengers on a flight, and the manner addressed in Section 2. In the test we set the maximum allowable waiting time to be 30 min, in accordance with the airport authorities. Since the results for the various weighting vectors were similar, here we only show the one weighting vector, $(1/3, 1)$. To unify the units of two objective function values, we divided the passenger walking distance (unit: meters) by 1.0 m/s and transformed it into the passenger walking time (unit: seconds). The test results are shown in Tables 1–3.

Table 1 shows the problem size. There were 145 flights and 24 available gates. Since the maximum allowable waiting time is 30 min and there are four choices of waiting times, 0, 10, 20 and 30 min, the number of total variables is 13,920 ($= 145 \times 24 \times 4$). We used 5 min as a check time in a 19 h operating day. Thus, there are a total of 228 times in one day. Consequently, the number of total constraints is equal to 5617 ($= 145 + 24 \times 228$). As mentioned in Section 2, since there are usually many redundant constraints, we developed a searching algorithm to find all the

Table 1
Test problem scale

No. flights	No. available gates	No. variables	No. total constraints	No. effective constraints
145	24	13,920	5617	4057

Table 2
Test results

No. gates used	No. variables used	Variables used (%)	Initial solution, IS (s) ①	Optimal integer solution (s) ②	Optimal linear solution (s) ③	Computation time (s)	① − ② (s)	(① − ②)/② (%)	③ − ② (s)	(③ − ②)/② (%)
23	1355	9.73%	7513607.8	7295065.0	7293875.5	6402.2	218542.8	3.00	−1189.5	−0.016

Table 3
Objective values and computation times in different stages of optimization

	Objective value (s)	Computation time (s)	Percentage of accumulated computation time (%)
Initial solution	7513607.8	6.15	0.10
Best feasible integer solution in linear optimization	7300660.0	1109.15	17.32
Linear optimal solution	7293875.5	2154.31	33.65
Integer optimal solution	7295065.0	6402.20	100.00

effective constraints within 1.32 s. As a result, there are 4057 effective constraints, about 72.23% of all the constraints.

Table 2 shows some of the test results. In particular, 23 gates are used but a temporary gate is not used. Except when a flight is assigned to a temporary gate, all other flights are assigned to the 22 normal gates. There is only one flight waiting for 10 min. There are 1355 variables used, that is about 9.73% of all variables, showing that the column generation approach is efficient. In the optimal integer solution, the total passenger walking time is 21,624,995 s and the total passenger waiting time is 83,400 s. The weighted objective value, equal to 1/3 total walking distance plus the total waiting time, is 7,295,065.0 s which is 1189.5 s (about 0.016%) more than the weighted linear optimal objective. This shows that the duality gap is very small. In addition, the weighted objective value of the initial solution is 7,513,607.8 s, with a gap of 218,542.8 s (3.00%) from the optimal integer solution.

Table 3 shows the objective values and the computation times in different stages of optimization. In particular, it took 2154.31 s (about 33.65% of total computation time, 6402.2 s) to solve for the linear optimal solution. It took another 4247.89 s (about 66.35% of the total computation time) to solve for the optimal integer solution. It took 6.15 s (about 0.1% of the total computation time) to find the initial solution. The best feasible integer solution obtained in linear optimization was 7,300,660 s, which is 5595 s (0.077%) more than the optimal integer solution. The computation time used was 1109.15 s (about 17.32% of the total computation time). In this case, the solution process may be stopped when the linear optimal solution is solved. The error gap would then be insured to within 0.077% ($= 5595/7,293,875.5$).

It should be mentioned that for comparison we tried not using the column generation approach in the solution process. That is, we put all the variables into the master problem to start our algorithm. The result is that we terminated the computing process after 24 h without finding the integer optimal solution. This also shows that the column generation approach is very useful for solving large-scale problems.

4.2. Sensitivity analysis

Below we perform a sensitivity analysis on the buffer time, ground service time, available gates and weighting vectors, which are essential inputs into the model. We also test the two methods for repetitively solving the various weighting vectors. Sensitivity analyses of other factors may be performed in a similar way in the future.

1. *Buffer time*: As mentioned previously, a buffer time between two continuous flights is useful for resolving minor flight delays in real-time operations. To evaluate the length of the buffer time in gate assignments, we tested 5 scenarios, 10, 20, 30, 40 and 50 min of buffer time. The results are shown in Table 4.

From Table 4, we find that the weighted objective values are positively correlated with the length of the buffer time. In particular, there is no delayed flight when the buffer time is set to be 10 min (scenario 1) and there is a flight delay of 30 min when the buffer time is set to be 50 min (scenario 5). The delay of other flights in all scenarios would be at most 10 min. The largest passenger waiting time, which happens in scenario 4, is 570,600 s. Note that increasing weighted objective values does not insure that the total passenger waiting time is increased, because the minimum weighted objective value is a combination of two objective values. The results also show

Table 4
Results due to changing the buffer times

Buffer time (min)	No. delayed flights			Total passenger waiting time, B (s)	Total passenger walking distance, A (s)	Weighted objective value $A/3+B$ (s)
	10 min	20 min	30 min			
10	0	0	0	0	21,107,000	7035666.6
20	1	0	0	132,000	21,069,193	7155064.2
30	1	0	0	83,400	21,634,995	7295065.0
40	1	0	1	570,600	20,700,092	7470630.6
50	3	0	0	354,600	21,704,825	7589541.6

that the length of the buffer time significantly influences the gate assignment process, so effectively reducing the length of the buffer time can improve the combined effects of passenger walking distance and waiting time. Therefore, a minimum but reasonable buffer time should be set for the user in practice.

2. *Ground service time*: The length of the ground service time always affects gate assignment. Since the ground service time is set to at least 30 min in practice, according to the airport authority, we tested 3 scenarios, with 30, 40 and 50 min ground service times. The results are shown in Table 5.

From Table 5, we find that the weighted objective values and the passenger waiting times are positively correlated with the length of the ground service times. In particular, except when there is a flight with a delay of 20 min, when the ground service time is set to be 50 min (scenario 3), delays for other flights in all scenarios are at most 10 min. The longest passenger waiting time, in scenario 3, is 583,200 s. The results also show that the length of the ground service time significantly influences the gate assignment. If the length of the ground service time can be effectively reduced, then the combined effects of both the passenger walking distance and the waiting time can be improved.

3. *Available gates*: In actual operations, some gates may be temporarily unavailable due to such incidences as equipment problems. To evaluate the effect of decreasing the number of available gates, we tested 5 scenarios, specifically with 19, 20, 21, 22 and 23 available gates. We also assumed the maximum allowable waiting time to be 60 min (instead of 30 min), to avoid infeasibility in gate assignment when the number of available gates is greatly reduced (for example, 19 gates). The results are shown in Table 6.

Table 6 shows that the number of available gates significantly influences the gate assignment. The weighted objective value is negatively correlated with the number of available gates. Sig-

Table 5
Results due to changing the ground service times

Ground service time (min)	No. delayed flights			Total passenger waiting time, B (s)	Total passenger walking distance, A (s)	Weighted objective value, $A/3+B$ (s)
	10 min	20 min	30 min			
30	1	0	0	83,400	21,634,995	7295065.0
40	2	0	0	270,000	21,324,571	7378190.2
50	2	1	0	583,200	20,543,363	7430987.8

Table 6
Results due to changing the number of available gates

No. of available gates	10 min	20 min	30 min	40 min	50 min	60 min	Total passenger waiting time, B (s)	Total passenger walking distance, A (s)	Weighted objective value, $A/3 + B$ (s)
19	5	3	2	1	2	0	3747600	11926324	7723041.2
20	2	2	2	1	1	1	2,101,200	16,214,883	7506161.0
21	2	2	0	1	0	0	1,116,600	18,805,606	7385135.4
22	1	0	0	1	0	0	625,200	20,101,264	7325621.4
23	1	0	0	0	0	0	83,400	21,634,995	7295065.0

nificant flight delays occur when the number of gates is greatly reduced. For example, when the number of available gates is 23, the total passenger waiting time is only 83,400 s. However, when the number of available gates is reduced to 19, the total passenger waiting time is greatly increased to 3,747,600 s. This indicates that delaying flights is an important strategy when there are not enough gates for gate assignment. Note that most gate assignment models in the past (for example, Yan and Chang, 1998, or Mangoubi and Mathaisel, 1985) did not include a delaying flight strategy.

4. *Weighting vector*: A weighting vector reflects the relative importance between two objective functions, the passenger walking distance and the waiting time. To evaluate the effects of various weighting vectors on gate assignments, we tested 4 scenarios, with weighting vectors of $(1/3, 1/10)$, $(1/3, 1/2)$, $(1/3, 1)$ and $(1/3, 10)$. The results are shown in Table 7 and Fig. 3. Other weighting vectors can be similarly analyzed but are not addressed here. Note that in Fig. 3, the blank columns represent the passenger waiting time, while the columns with slashed lines represent the passenger walking distance.

From Table 7, when the weighting vector is (1/3:10), there is no delayed flight. When the weighting vector is (1/3:1), there is a flight with a 10 min delay and a total waiting time of 83,400 s. When the weighting vector is (1/3:1/2), there are 4 flights with a 10 min delay, a flight with a 20 min delay and a total waiting time of 930,600 s. When the weighting vector is (1/3:1/10), there are 21 flights with 10 min delay, 5 flights with 20 min delay, one flight with a 30 min delay and a total waiting time of 5,403,600 s. These results indicate that when the importance of the passenger waiting time is relatively decreased, then the flight delay is relatively increased.

Table 7
Results due to changing the weighting vectors

Weighting vector (walking distance: waiting time)	No. delayed flights			Total passenger waiting time, B (s)	Total passenger walking distance, A (s)	Weighted objective value, $A/3 + B$ (s)
	10 min	20 min	30 min			
1/3:1/10	21	5	1	5,403,600	5,418,250	7209683.4
1/3:1/2	4	1	0	9,30,600	19,051,651	7281150.2
1/3:1	1	0	0	83,400	21,634,995	7295065.0
1/3:10	0	0	0	0	21,893,038	7297679.2

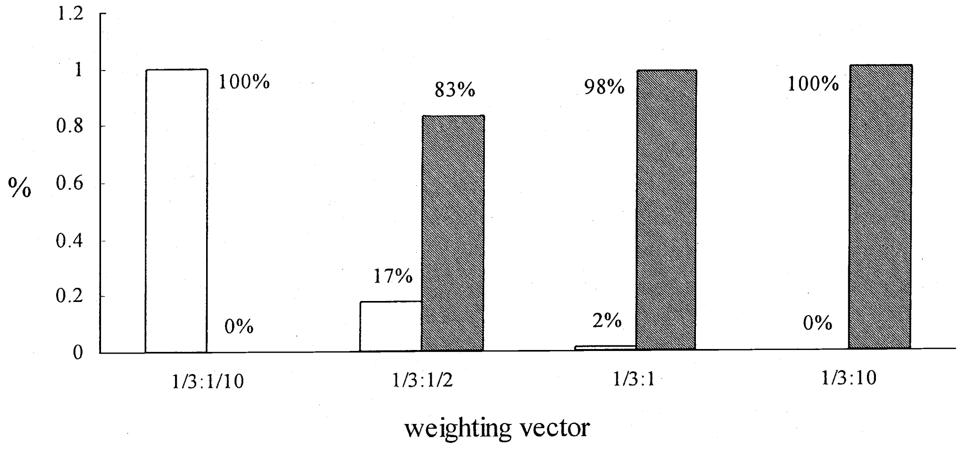

Fig. 3. Relative percentages of two objective function values for each weighting vector.

5. *Repetitive solving for various weighting vectors*: In section 3, we developed two methods, A and B, to repetitively solve for various weighting vectors. To evaluate the performance of these two methods, we tested them with the 4 weighting vectors used in the previous sensitivity analysis. We also solve each weighting vector independently (called method C) for comparison with the two methods. The results are shown in Table 8.

From Table 8, concerning the computation time for linear optimization, method A saves 63.58% (=(27504.34 − 10016.34)/27504.34) and method B saves 67.87% (= 27504.34 − 8837.05/27504.34) compared to method C, showing that methods A and B are efficient. Moreover, method B spends 11.77% (=(10016.43 − 8837.05)/10016.43) relatively less time than method A. We also found that method A used more variables than method B for each and all weighting vectors. Concerning the computation time for resolving non-integer solutions after linear optimization, methods A and B perform slightly better than method C. Consequently, for the total computation time, method B (46884.44 s) is more efficient than method A (50023.82 s), which is in turn more efficient than method C (66239.00 s).

As to the number of variables used in linear optimization, method A uses 1880 variables and method B uses 1867 variables. Both use many fewer variables than method C, that is, 5667 $(= 1759 + 1366 + 1278 + 1264)$ variables for the 4 weighting vectors. This implies that many variables used in solving for the 4 weighting vectors in method C are the same, and therefore method C is relatively inefficient. This also shows that methods A and B are efficient, effectively

Table 8
Results for repetitively solving for various weight vectors

	Computation time for linear optimization (s)	Total computation time (s)	No. variables used in linear optimization	No. total variables used
Method A	10016.43	50023.82	1880	2731
Method B	8837.05	46884.44	1867	2628
Method C	27504.34	66239.00	(1759, 1366, 1278, 1264)	(2271, 1684, 1355, 1245)

generating the initial columns to solve for the various weighting vectors. The analysis for integer optimization is similar and is not addressed here.

5. Conclusions

In this paper, we proposed a multiple objective integer programming model to assist airport authorities in gate assignment. We formulated the model as a multiple objective zero–one integer program, characterized as NP-hard in terms of optimization. To efficiently solve large-scale problems in practice, we use the weighting method, the column generation approach, the simplex method and the branch-and-bound technique to develop a solution algorithm. To efficiently solve for various weighting vectors, we also developed two methods.

To test how well the model and the algorithms may be applied to actual operations, we performed a case study concerning the operation of the CKS Airport. Many problem scenarios were tested, with substantial problem sizes of up to 25,375 variables and 5845 constraints. The model and the algorithms developed in this research performed well. The results were good and could be useful as a reference for both airport authorities and researchers. Finally, the column generation technique has first been applied in this research to gate assignment problems and multiple objective linear/integer programming. The results were impressive, showing that this method could be applied to similar problems or other fields in the future.

Acknowledgements

This research was supported by a grant (NSC-87-2211-E-008-014) from the National Science Council of Taiwan. We thank the Civil Aeronautics Administration of Taiwan for providing the test data. We also thank Professor Fu-Fwa Han for providing his valuable suggestions on our solution algorithm. Finally, we would like to thank the anonymous referees for their valuable comments and suggestions.

References

Ahuja, R.K., Magnanti, T.L., Orlin, J.B., 1993. Network Flows, Theory, Algorithms, and Applications. Prentice-Hall, Englewood Cliffs, NJ.

Babic, O., Teodorovic, D., Tosic, V., 1984. Aircraft stand assignment to minimize walking. Journal of Transportation Engineering 110, 55–66.

Desrochers, M., Soumis, F., 1989. A column generation approach to the urban transit crew scheduling problem. Transportation Science 23, 1–13.

Garey, M.R., Johnson, D.S., 1979. Computers and Interactability: A Guide to the Theory of NP-completeness. Freeman, San Francisco, CA.

Gosling, G.D., 1990. Design of an expert system for aircraft gate assignment. Transportation Research 24, 59–69.

Hamzawi, S.G., 1986. Management and planning of airport gate capacity: a microcomputer-based gate assignment simulation model. Transportation Planning and Technology 11, 189–202.

Hillier, F.S., Lieberman, G.J., 1995. Introduction to Mathematical Programming. McGraw-Hill, New York.

Hwang, C.L., 1979. Multiple Objective Decision Making – Method and Application. Springer, New York.

Lavoie, S., Minoux, M., Odier, E., 1988. A new approach for crew pairing problems by column generation with an application to air transportation. European Journal of Operational Research 35, 45–58.

Mangoubi, R.S., Mathaisel, D.F.X., 1985. Optimizing gate assignment at airport terminals. Transportation Science 19, 173–188.

Sheffi, Y., Yan, S., Teitelbaum, B., 1990. Optimal loading of transportation conveyances. The Annals of Society of Logistics Engineers 2, 8–25.

Su, Y.Y., Srihari, K., 1993. A knowledge-based aircraft-gate assignment advisor. Computers and Industrial Engineering 25, 123–126.

Vanderstraetan, G., Bergeron, M., 1988. Automatic assignment of aircraft to gates at a terminal. Computers and Industrial Engineering 14, 15–25.

Yan, S., Chang, C., 1998. A network model for gate assignment. Journal of Advanced Transportation 32, 176–189.