Journal of Advanced Transportation, Vol. 32, No. 2, pp. 176-189

A Network Model for Gate Assignment

Shangyao Yan
Chia-Ming Chang

In this research we developed a network model that will help the airport authorities assign flights to gates both efficiently and effectively. The model was formulated as a multi-commodity network flow problem. An algorithm based on the Lagrangian relaxation, with subgradient methods, accompanied by a shortest path algorithm and a Lagrangian heuristic was developed to solve the problem. The model was tested using data from Chiang Chiek-Shek Airport.

Introduction

The flight-to-gate assignment problem is important for the operation of an airport system. Due to deregulation in recent years, market demand has grown significantly for airline carriers in Taiwan. Recently, more new airlines (both domestic and foreign) are flying into Taiwan. More and more airport operations are being crowded into facilities which were not originally designed to handle this increased traffic volume. The result is a worsening level of service in terms of congestion and delay. It is to be expected that the demand will be further increased in the future. Although the physical expansion of airport facilities is the long-term solution, there are a number of other methods which could be considered for alleviating airport capacity constraints. One of the short-term solutions is to optimize the assignment of flights to existing gates.

In planning for an effective utilization of aircraft gates at an airport terminal, the airport authority usually considers the distance that a passenger is required to walk inside the terminal to reach his departure gate, the baggage claim area, or his connecting flight [Mangoubi and Mathaisel, 1985]. The typical traditional approach of the Taiwan airport authority to assign flights to gates is by-hand, without systematic analysis. For

Shangyao Yan and Chia-Ming Chang are at the National Central University, Taiwan.
Received July 1996; Accepted June 1997.

example, the Chiang Chiek-Shek airport authority collects the flight schedules around 6 PM, from every airline using the airport the next day. Then the staff, guided by experience, manually assigns all flights to the existing gates. Although the Chiang Chiek-Shek airport is not very congested at this time, the assignment of all the flights (around 214 flights a day) to all gates (22 available gates) is not efficient. In practice, it takes a couple of hours to find a feasible solution, not the optimal solution. Moreover, there is no way to evaluate the feasible one in terms of optimality. Thus an efficient and effective tool for the optimization of the flight-to-gate assignment problem is important, both now and in the future.

A few analytical models have been developed to help effectively assign daily flights to gates [Babic et al., 1984; Mangoubi and Mathaisel, 1985; and Vanderstraetan and Bergeron, 1988]. These models are normally formulated as zero-one integer programming models. The objective functions are usually the minimization of 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, and 2) no two aircraft may be assigned to the same gate concurrently. Because of this airport's configuration, some special constraints were also introduced; for example, if a large-type aircraft is assigned to a gate, then the same type of aircraft cannot be assigned to the neighboring gates [Vanderstraetan and Bergeron, 1988].

These models were solved using 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]. Optimal solution to large scale zero-one integer problem is generally difficult to solve. However, heuristics have been suggested by some researchers (for example, Mangoubi and Mathaisel, 1985) to deal with this problem. Since network models have been shown to be efficient for handling large-scale integer optimization problems [Yan and Young, 1996; Yan and Yang, 1996; and Yan and Lin, 1997], we introduce a new network formulation as well a solution algorithm for the optimization of gate assignment problems.

Since numerous factors may affect flight schedules in real time operations (for example, flight delays, early arrivals, or the congestion of ground facilities), the flight schedule more or less needs to be modified in real time. We note that the model developed by our research would be useful for airport authorities to plan flight-to-gate assignments in advance, rather than to help determine the assignment under schedule perturbation in real time. Although we believe that the model could be suitably modified in the future in order to assist the airport authorities to make real time assignments, the model can even now generate a good schedule which can serve as a basis for necessary, minor modification in real time operations.

It should be mentioned that [Wirasinghe and Bandara, 1990] introduced a related model that minimized the sum of the flight cost and the construction cost to determine the optimal number of gates in planning the airport configuration. This model however, cannot be used to assign flights to gates. We note that other than analytical models, there have been other recently developed approaches for solving the gate assignment problems. [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 assignment. These approaches are, however, not the same as what we introduce in this paper. The remainder of the paper is organized as follows; first, we introduce our model. The model is then formulated as a multi-commodity network flow problem and its solution is developed hereafter. Finally, a case study is performed to test the model in the real world.

The Model

Before formulating the problem, suppose that we have the following information: 1) the layout of the terminal area and the number of gates, 2) the 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 the arriving, departing and transferring passengers, 5) the walking distances for each gate, including the distance for an arriving passenger, a departing passenger, or a transferring passenger.

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. 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]. Thus, the transferring distance is equal to the average distance from one gate to every other gate. As mentioned by Mangoubi and Mathaisel (1985), the uniform distribution assumption may not be realistic. Attractive gates, which are close to the main hall, tend to be clustered together, a transfer passenger's walking distance between these gates is shorter. Note that the use of such an assumption to derive the transferring distance is justifiable [Mangoubi and Mathaisel, 1985].

We suggest using a time-space network, as shown in Figure 1, to formulate the assignment of a flight or a pair of flights (an arrival flight and a departure flight), because it is natural to represent conveyance routings in the dimensions of time and space [Yan et al., 1995]. Three type of flights are considered in the gate assignment problem; (a) an arriving flight, (b) a departing flight, (c) a pair of connecting flights served by an airplane. If two connecting flights (case c) are separated by a short time (for example, less than three hours) and served by the same airplane, then these two flights are usually assigned to the same gate, so that dragging the airplane away from and back to the gate in a short time is avoided. As a result, we will have similar multiple networks, each corresponding to a flight, or a pair of flights.

In Figure 1, the horizontal axis represents the gate locations, while the vertical axis represents the time duration. The network contains a period from a starting time to an ending time. The starting time for a flight or a pair of flights is when the gates are ready for use. The starting time for case (a) or (c) is equal to the flight's arrival time minus half of the clear time. The clear time is defined as a safe gap between two continuous airplanes using the same gate. The starting time for case (b) is equal to the departure time minus the time required for investigating aircraft before departure, fueling, passenger/baggage boarding, etc., minus half of the clear time. The ending time is when the gates are released from the use of the associated aircraft and ready for use again. The ending time for case (a) is equal to the flight's arrival time, plus passenger/baggage deplaning time, plus half of the clear time. The starting time for case (b) or (c) is equal to the flight's departure time plus half of the clear time.

There are two dummy nodes in each network, the initial and the final nodes. Each node, excluding the two dummy nodes, represents a specific gate at a specific time. Each arc represents an activity for an airplane. The three types of arcs are described below:

<u>**entering arc**</u>: an entering arc is added, from the initial node, to all available gates, at the starting time. If the specific aircraft type is excluded from some gates, or if the associated airline is excluded from using some gates

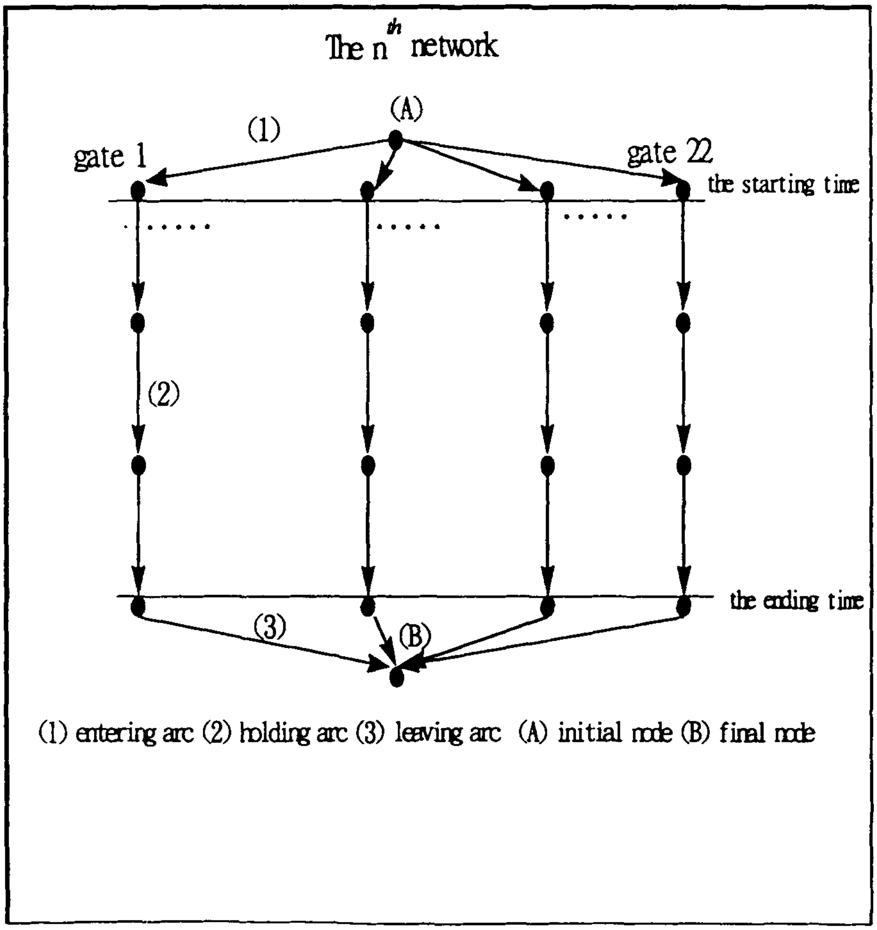
The n^th network
(A)
(1)
gate 1
gate 22
the starting time
(2)
the ending time
(3)
(B)
(1) entering arc (2) holding arc (3) leaving arc (A) initial node (B) final node

Figure 1. The network model

(because those gates are dedicated to specified airlines), then entering arcs are not added to these gates. Each entering arc indicates that the airplane enters a gate. The arc cost for case (a) or (c) represents the walking distance for the arriving and transferring passengers, for the associated flight at the gate. The arc cost for case (b) is zero. The arc flow upper bound is one, meaning that at most one airplane enters the gate (i.e. at most one flight is assigned to the gate). The arc flow lower bound is zero, indicating that the gate is not assigned.

holding arc: a holding arc is added between two adjacent nodes at a gate. Each holding arc represents the holding of an airplane at a gate in a time window. The time window lies between two times at a gate. The arc cost is zero. The arc flow upper bound is equal to one, indicating that at most one airplane can be held, at this gate, during this time window. The arc flow lower bound is zero, meaning that a minimum of zero airplane is held at this gate in this time window.

leaving arc: a leaving arc is added, for every available gate, from the ending time to the final node. As above, if the specific aircraft type is excluded from some gates, or if the associated airline is excluded from using some gates, then leaving arcs are not added for these gates. Each leaving arc indicates that the airplane has left the gate. In other words, the gate is released from use by the associated airplane, after the ending time. The arc cost for case (a) is zero. The arc cost for case (b) or (c) represents the walking distance of the departing passengers for the associated flight at the gate. The arc flow upper bound is one, meaning that at most one airplane leaves the gate. If the arc flow lower bound is zero, it indicates that this gate is not used.

It should be noted that there is only one supply (one airplane) at the initial node and one demand at the final node in each network, which turns out to be a shortest path problem. To ensure that at most one airplane stays in each gate, at each time window, a set of bundle constraints should be added in the problem formulation: the sum of all arc flows, associated with a gate at a time window, should not be greater than one. Note that the holding arcs along a gate in each network may be suitably aggregated to make fewer arcs to reduce the number of side constraints. However, we did not do so because of the following reasons; 1) it turns to be complicated to construct the bundle constraints, 2) the algorithm we develop in Section 3 will not incur much difficulty or complexity in its solving. In particular, the relaxation of bundle constraints and the solving of the shortest path problems are very efficient due to the network structure.

The objective of this model is to "flow" all the airplanes in each network, at a minimum cost, which is equivalent to the minimization of total passenger walking distance. The integer program formulating the model is shown below;

$$Min \sum_{n \in M} \sum_{(i,j) \in A^n} c^n_{ij} x^n_{ij} \qquad (1)$$

$$s.t. \quad \sum_{j \in N^n} x^n_{ij} - \sum_{k \in N^n} x^n_{ki} = b^n_i \ , \quad \forall i \in N^n, \forall n \in M \qquad (2)$$

$$\sum_{n \in M} x^n_{ij} \leq 1 \quad , \quad \forall (i,j) \in L \qquad (3)$$

$$0 \leq x^n_{ij} \leq 1 \quad , \quad \forall (i,j) \in A^n, \forall n \in M \qquad (4)$$

$$x^n_{ij} \in Integer \quad , \quad \forall (i,j) \in A^n, \forall n \in M \qquad (5)$$

where;

n : the nth flight (or flight pair)

M : the set of all flights and flight pairs

N^n: the set of all nodes in the nth flight (or flight pair) network

A^n: the set of all arcs in the nth flight (or flight pair) network

b^n_i: the supply/demand of node i in the nth flight (or flight pair) network. It equals 1 for the initial node, -1 for the final node, and 0 for others.

L: the set of all time windows for all gates

c^n_{ij}: arc (i,j) cost in the nth flight (or flight pair) network. If arc (i,j) is an entering arc or a leaving arc, then c^n_{ij} denotes all walking distance for passengers in the nth flight (or flight pair); if arc (i,j) is a holding arc, then c^n_{ij} is zero. c^n_{ij}, which is a constant, is calculated for every flight (or flight pair) before optimization of the model.

x^n_{ij}: arc (i,j) flow in the nth flight (or flight pair) network. x^n_{ij} is the decision variable in the model.

The model is a typical multi-commodity network flow problem. Objective (1) of this model is to "flow" all node supplies to all node demands in each network, at a minimum cost. Constraint (2) is the flow conservation constraint at every node in each network. Constraint (3) ensures that at most one aircraft stays at every gate in every time window. Constraint (4) ensures that all arc flows are within their upper and lower bounds in each network, and constraint (5) ensures that all arc flows are integers in each network.

Solution Methods

The multi-commodity network flow problem is characterized as an NP-hard problem [Garey and Johnson, 1976]. Our research suggests using a Lagrangian relaxation with subgradient methods (LRS) for an approximation of the near-optimal solutions in order to avoid a long computation time. The LRS is known for its fast convergence and efficient allocation of memory space [Fisher, 1981; Yan and Young, 1996; Yan and Yang, 1996; and Yan and Lin, 1997]. By dualizing the bundle constraints with a Lagrangian multiplier, we produce a Lagrangian problem that is easy to solve and whose objective is a lower bound on the optimal value of the original problem. We then use a Lagrangian heuristic to find a feasible solution whose objective is an upper bound on the optimal value of the original problem. To reduce the gap between the upper bound and the lower bound, we modify the Lagrangian multiplier using a subgradient method, then solve for another lower and upper bound. Better bounds are updated. The process is repeated until the gap converges to within 1% of the error, or the number of iterations reaches 300.

We note that in many tests in the case study addressed later, the gaps did not lessen after 100 iterations. In particular, the best gap is around 4%. The process in each test is terminated according to the number of iterations. Therefore, we set the number of iterations at 300, indicating that the gap will not further converge after 300 iterations. In practice, users of the model may try to set their own iteration limits after some pre-testing. Similarly, users may set their own converging gaps at other than 1%. The detailed application of the LRS is summarized below in three parts.

(1) A lower bound for each iteration: The bundle constraints are relaxed by a non-negative Lagrangian multiplier and are added to the objective function, resulting in a Lagrangian problem which includes |M| shortest path problems. The shortest path problems can be easily solved using the label correcting algorithm [Ahuja et al., 1993]. The optimal objective value can be proven to be the lower bound of the original problem [Fisher, 1981].

(2) An upper bound for each iteration: A Lagrangian heuristic is developed to find a feasible solution (an upper bound of the optimal solution), from the lower bound solution (typically an infeasible solution), for each iteration. The Lagrangian heuristic proceeds as follows;

Step 1: Sort the flights and the flight pairs at the lower bound solution in the order of decreasing path distance. Note that the arc costs in the lower bound solution are modified by Lagrangian relaxation.
$k = 1$.

Step 2: Solve the shortest path for the kth flight (or flight pair). Set every arc cost along the shortest path to be a very large number (e.g. 9999999999).
Step 3: if k = |M|, stop the algorithm; else k = k + 1, go to Step 2.

Note that if any shortest path distance in Step 2 is equal to a very large number, then it means that there does not exist a feasible solution in this iteration. Such a situation, however, was not found in the case study presented in Section 4. It should be mentioned that the Lagrangian heuristic never performs worse than the heuristic [Mangoubi, 1984]. In particular, the heuristic [Mangoubi, 1984] can be shown to be equivalent to the first iteration of the Lagrangian heuristic (i.e. when the Lagrangian multipliers are zero).

(3) Solution process: the LRS solution steps are listed below;
Step 0: Set the initial Lagrangian multiplier to zero.
Step 1: Solve the Lagrangian problem optimally using the label correcting algorithm to get a lower bound. Update the lower bound.
Step 2: Apply the Lagrangian heuristic to find an upper bound and update the upper bound.
Step 3: If the gap between the lower bound and the upper bound is within a 1% gap, or the number of iterations reaches 300, stop the algorithm.
Step 4: Adjust the Lagrangian multipliers using the subgradient method [Yan and Young, 1996]. Note that the subgradient method is an improvement on [Fisher, 1981] and [Camerini et al., 1976].
Step 5: Set k = k + 1. Go to Step 1.

Case Study

To test our model, we performed a case study regarding the operations of the Chiang Chiek-Shek Airport (an international airport) in Taiwan. There were 22 gates available at the airport. All data required in our model, mentioned in Section 2, were obtained from the airport authority. There were 35 arrival flights (case a), 41 departure flights (case b) and 69 flight pairs (case c) in one day's (Oct. 6, 1995) operation. Consequently, a total of 214 flights were tested. Note that each pair of flights (case c) are served by an airplane within three hours.

Several C programs and an automatic data process were developed for; (1) the analysis of the raw data, (2) the building of the model, (3) the development of the solution algorithm, and (4) the output of data. The case study was implemented on an HP715 workstation. Four scenarios were tested, with problem sizes of up to 31255 nodes and 60685 arcs. Scenario 1 was solved for the current 145 flights and flight pairs. To better understand the performance of our model, we randomly copied 5, 10 and 15 flights from the original flights, during the peak hours (8:00 AM - 10:00 AM), and then added them to the original flights in Scenarios 2 to 4, respectively. As a result, Scenarios 2, 3 and 4 were solved for 150, 155 and 160 flights and flight pairs, respectively.

To evaluate our model, we also used the heuristic [Mangoubi, 1984] to solve our problem. We note that the heuristic was found to be effective in [Mangoubi and Mathaisel, 1985] and was easy to implement in our own case study. The results are summarized in Table 1.

From Table 1, as the number of flights and flight pairs increase, the problem sizes also increase, meaning that more nodes and arcs are needed to construct a model for more flights. Note that the number of bundle constraints remains the same because these constraints are associated with the number of time windows, which is independent of the number of flights. As the number of flights and flight pairs increase, the computation times increase, meaning that a longer time is spent optimizing the assignment of more flights.. The computation times, which are all within 789 seconds (around 13 minutes), are obviously superior to those obtained by the current manual approach. Compared with the efficiency and the effectiveness of the current manual approach used by the airport authority, our algorithms are distinctly superior. As the number of flights and flight pairs increase, the objective values increase, meaning considering more walking distance for more passengers.

We also find that the model and algorithm developed in our research performed well, and could therefore be useful in practice. All problem instances converged to about a 5% gap in at most 789 seconds of CPU time. This shows that the Lagrangian relaxation-based algorithm could be efficient for solving the multi-commodity network flow problem. Although the convergence gaps are about 5% in all four scenarios, the objective values (the updated upper bounds) are better than that using Mangoubi's heuristic. In particular, the improvements for our model are; 244728 (3.46%), 362590 (4.87%), 299228 (3.89%), and 304897(3.77%), respectively.

Note that improvements of 3% to 5% seems to be marginal, however, the corresponding total walking distances (244728 m ~ 362590 m) are

Table 1. Results of all scenarios

number of flights and flight pairs	computation time (sec)	(1) objective (m)	convergence gap (%)	# nodes	# arcs	# bundle constraints	(2) Mangoubi's heuristic (m)	(2)-(1) (m)	[(2)-(1)]/(2) (%)
145	680	6822994	4.85	29454	57205	2319	7067722	244728	3.46
150	701	7082648	4.97	30016	58290	2319	7445238	362590	4.87
155	745	7394604	5.14	30087	58426	2319	7693832	299228	3.89
160	789	7772030	5.28	31255	60685	2319	8076927	304897	3.77

large. As mentioned above, Mangoubi's heuristic can be shown to be equivalent to the first iteration of the Lagrangian heuristic. Thus in practice, we can stop our algorithm at any iteration and produce better results than Mangoubi's. As a result, our model seems to be better than Mangoubi's in this respect.

We note that the convergence gap may be due to the inherent duality (relaxation) gap [Yan and Young, 1996]. To verify this, some form of enumeration procedure, such as the branch and bound technique, using the Lagrangian lower bound to help reduce the amount of concentration required, could be developed in the future.

Conclusions

Although the zero-one integer programming techniques have been applied to solve the gate assignment problems, they would not be efficient for handling large scale problems. In this paper, we developed a new network formulation and a solution algorithm as well, for the optimization of gate assignment problems, to help the airport authority to both efficiently and effectively plan the assignment of flights to gates. The network model is natural and it is easy to formulate the gate assignment problem. If more constraints are added, only a few nodes, links, or additional side constraints need to be modified, without changing the original network structure. Thus, we can handle more complicated and larger problems.

The model was formulated as a multi-commodity network flow problem. We developed a Lagrangian relaxation-based solution algorithm to solve the problem. A case study containing four scenarios was tested, with substantial problem sizes of up to 31255 nodes and 60685 arcs, on an HP715 workstation. All problem instances converged to about a 5% gap in at most 789 seconds of CPU time. The objective values were better than those using Mangoubi's heuristic. In particular, the improvements using our model are 244728 (3.46%), 362590 (4.87%), 299228 (3.89%), and 304897(3.77%) respectively. Our model is shown to be better than Mangoubi's. The results show that these models are both effective and efficient, and should be useful for actual operations.

To verify whether the convergence gap is due to an inherent duality (relaxation) gap, some form of enumeration procedure, such as the branch and bound technique, using the Lagrangian lower bound to help reduce the amount of concentration required, could be developed in the future. Since the case study was only for demonstration purposes at the current stage, evaluating the impact of applying the model to actual operations is left for future work.

Acknowledgments

This research was partially supported by a grant from the National Science Council of Taiwan. We thank the Civil Aeronautics Administration of Taiwan for providing the test data. We also thank the two anonymous referees for their valuable comments and suggestions.

References

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

Babic, O., Teodorovic, D. and Tosic, V. 1984. Aircraft Stand Assignment to Minimize Walking. Journal of Transportation Engineering 110: 55-66.

Fisher, M. L. 1981. The Lagrangian Relaxation Method for Solving Integer Programming Problems. Management Science 27: 1-18.

Garey, M. R. and Johnson, D. S. 1979. Computers and Intractability: A Guide to the Theory of NP-completeness, San Francisco: W.H. Freeman & Company.

Gosling, G. D. 1990. Design of an Expert System for Aircraft Gate Assignment. Transportation Research 24A: 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.

Mangoubi, R. S. 1984. Testing Gate Assignment Algorithms at Airport Terminals. B. S. thesis, Department of Mechanical Engineering, MIT.

Mangoubi, R. S. and Mathaisel, D. F. X. 1985. Optimizing Gate Assignment at Airport Terminals. Transportation Science 19: 173-188.

Su, Y. Y. and Srihari, K. 1993. A Knowledge Based Aircraft-Gate Assignment Advisor. Computers and Industrial Engineering 25: 123-126.

Vanderstraetan, G. and Bergeron, M. 1988. Automatic Assignment of Aircraft to Gates at A Terminal. Computers and Industrial Engineering 14: 15-25.

Wirasinghe, S. C. and Bandara, S. 1990. Airport Gate Position Estimation for Minimum Total Costs-Approximate Closed Form Solution. Transportation Research 24B: 287-297.

Yan, S., Bernstein, D. and Sheffi, Y. 1995. Intermodal pricing using network flow techniques. Transportation Research 29B: 171-180.

Yan, S. and Lin, C. 1997. Airline Scheduling for the Temporary Closure of Airports. Transportation Science 31: 72-82.

Yan, S. and Yang, D. 1996. A Decision Support Framework for Handling Schedule Perturbation. Transportation Research 30B: 405-419.

Yan, S. and Young, H. 1996. A Decision Support Framework for Multi-Fleet Routing and Multi-Stop Flight Scheduling. Transportation Research 30A: 379-398.