

Matchings in connection with ground delay program planning

Michael Ball *
Geir Dahl†
Thomas Vossen ‡

June 16, 2008

Abstract

In this paper we analyze certain matching problems that arise in ground delay program planning. Ground delay programs are air traffic flow management initiatives put in place when airport arrival demand is expected to exceed arrival capacity for an extended length of time, e.g. 4 hours. Most of the problems we study can be modeled as assignment problems, where flights are assigned to arrival slots. In the context we analyze, however, these problems have important special structure, which allows us to develop special solution properties. In particular, solutions are measured both in terms of efficiency (delay minimization) and equity (delay distribution). We show that the theory of majorization provides a powerful tool in addressing solution equity. We consider problems with flight deletions and develop special solution properties and parametric methods.

Keywords: Ground delay programs, matching, majorization.

1 Introduction

In this paper we study certain problems related to the assignment of flights to airport arrival slots. These problems arise in the context of ground delay programs (GDPs), which are traffic flow management initiatives put in place in the U.S. when airport arrival demand is expected to exceed arrival capacity for an extended length of time, e.g. 4 hours; see Ball et al. [3] for background. GDPs are usually instituted in response to weather conditions at an airport, which substantially reduce visibility. Thus, this assignment applies only to the day in question and can rely on an existing published schedule giving arrival times for each flight. Flights will be assigned arrival slots no earlier than their scheduled arrival times so that, in the typical case of a strictly later assignment, the flight will be assigned an arrival delay. This delay will be converted to a departure delay (ground delay). The net effect is that flights will be delayed on the ground at their origin airport so that when they arrive at the destination airport they can immediately land, incurring little or no airborne delay. Some of our results also apply to a recently developed initiative for managing U.S. air traffic: the airspace flow program (AFP). In this case, a volume of congested airspace (flow constrained area – FCA) is isolated and all flights that have filed flight plans through that airspace during a time horizon of interest are identified. Slots are defined at the boundary of the FCA and a GDP-like assignment is carried out.

*Robert H. Smith School of Business and Institute for Systems Research, University of Maryland, College Park 20742, USA (mball@rhsmith.umd.edu)

†Centre of Mathematics for Applications, University of Oslo, Oslo, Norway (geird@ifi.uio.no)

‡Leeds School of Business, University of Colorado, Boulder, CO 80309, USA (Thomas.Vossen@Colorado.edu)

The basic inputs to the problems we study are a set of flights and a set of slots, together with certain information that restricts the flight-to-slot assignments. Generally, these problems can be modeled and solved as simple flight-to-slot assignment problems. Our goal in this paper is to investigate the special structure of these problems in order to produce insights into the corresponding structure of the solutions and their properties. Where equitable assignments are of interest, simple assignment problem models may not be able to adequately represent the problem and our deeper analysis produces the needed solution properties. Additionally, in some cases, we are able to give very simple, e.g. greedy, solution algorithms.

There are two perspectives relative to solving these problems. First, the air traffic service provider (the ATSP in the U.S. is the Federal Aviation Administration (FAA)) is responsible for managing the airspace safely, while limiting delays and maximizing throughput. In addition, the airspace user (ASU's consist of airlines and general aviation users) has its own internal objectives, generally related to minimizing overall delays for its flight subset. The ATSP must consider issues related to equitably balancing resources among the ASU's, whereas each ASU seeks to optimize its own interests. Whereas the ATSP must allocate all available slots to all flights (when possible), the ASU will formulate a problem of reallocating the slots it has been given among its own flights.

Section 2 gives the underlying model we analyze; most importantly, it defines the flight allocation graph, which is a structured bipartite graph that represents our problem. Section 3 describes Glover's Algorithm and Modified Glover's Algorithm, which are used extensively in finding solutions of interest in the flight allocation graph. Section 4 gives solution properties of interest to us. It presents definitions and results related to both delay minimization and equity, the two solution criteria of interest to us. Our results generally fall into two categories. In Section 5, we treat the case where all flights are allocated a slot. This is generally the assumption for modeling ATSP planning of a GDP since an ATSP cannot cancel a flight, i.e. the ATSP must allocate some (possibly very late) airport arrival slot to each flight. The models of Section 6, explicitly consider the problem of selecting which flights should be allocated a slot and which should not. We say that a flight not allocated a slot is *deleted*. Such problems apply when an ASU plans its response to a GDP and it considers flight cancelations, which correspond to flights not allocated any slot. In addition, when an ATSP is planning an AFP, it has the prerogative of not allocating slots to certain flights – in this case, it is not canceling the flight, rather it is just prohibiting the flight from passing through a volume of airspace. The ASU would then have the choice of rerouting that flight around the FCA or canceling the flight. The results of Section 6 provide parametric solutions to flight deletion problems. These should be quite useful in decision support as it can be difficult to objectively evaluate a-priori the cost of deleting a flight vs the cost of additional delay to several others.

Our work uses as a starting point the work by Vossen and Ball [11]. Some of the results in Sections 3, 4 and 5 can be viewed as reinterpretation within a graph context of results in Vossen and Ball [11]. However, some important new ideas and results are provided, including the use of majorization theory and its importance in measuring equity, the relationship to Schur-convexity and its implications and use of fork matchings. All of the results on flight deletions given in Section 6 are new.

2 The model

The underlying model we consider consists of a bipartite graph $G = (V, E)$ with vertex set $V = F \cup S$ and edge set E . We call G the *flight allocation graph*. The vertex set is partitioned into the set of flights F and the set of arrival slots S . We let $F = \{f_1, f_2, \dots, f_m\}$ and $S = \{s_1, s_2, \dots, s_n\}$, and define the index sets $I = \{1, 2, \dots, m\}$ and $J = \{1, 2, \dots, n\}$. We also let $T(f_i)$ denote the

(original) scheduled arrival time of flight f_i ($i \in I$), and denote the start time of slot s_j by $T(s_j)$.

We assume that flights are ordered according to their scheduled arrival times so that

$$T(f_1) \leq T(f_2) \leq \dots \leq T(f_m).$$

Moreover, slots are ordered in the natural (time-based) way, that is,

$$T(s_1) \leq T(s_2) \leq \dots \leq T(s_n).$$

Every flight f_i ($i \in I$) has an earliest possible arrival time EAT_i which is no earlier than the scheduled arrival time $T(f_i)$, i.e., $T(f_i) \leq EAT_i$. We may have strict inequality here; for instance, a mechanical problem may cause a flight delay. Moreover, we assume that there is a maximum delay d_i , specified for each flight f_i , so that any flight that is delayed more than d_i will be deleted. Thus, for each flight f_i we can define its earliest slot $j(i)$ and latest slot $j'(i)$ as

$$j(i) = \min\{j \in J : T(s_j) \geq EAT_i\} \quad \text{and} \quad j'(i) = \max\{j \in J : T(s_j) \leq T(f_i) + d_i\}.$$

With these assumptions, a flight f_i may be allocated to a slot s_j if and only if $j(i) \leq j \leq j'(i)$. The edges in the bipartite graph G correspond to possible flight-slot allocations, so

$$E = \{[f_i, s_j] : j(i) \leq j \leq j'(i), i \leq m\}.$$

The bipartite flight allocation graph G has a special structure: for each $i \leq m$ the set of neighbor nodes of node f_i is an interval of consecutive nodes in S (with the mentioned ordering of S). The neighbors of a node s_j , however, may not be an interval since a flight with early scheduled arrival time may be more delayed than a flight with later scheduled arrival time. In Section 5.3 we consider the special case when such mixed delays do not occur and each node s_j is adjacent to an interval of nodes in F .

Our focus is on flight allocations and their delay properties. A flight *allocation* is a function that associates (some or all) flights with (permitted) slots in such a way that each flight is allocated to at most one slot and no more than one flight is allocated to a slot. Thus, an allocation corresponds to a matching in G .

In the course of our analysis we will consider various special cases of the model described.

No delay bounds: Here, $d_i = \infty$ for all i . This assumption is not unusual for all problem perspectives, particularly the ATSP perspective, since the ATSP generally seeks to put few restrictions on possible allocations.

Constant delay bounds: Here, $d_i = d$ for all i . This assumption is reasonable for certain ATSP problems, where the ATSP seeks to treat all ASU's equitably.

Weakly increasing $j'(i)$: Here, $i \leq k$ implies $j'(i) \leq j'(k)$. This is a generalization of the constant delay bounds assumption and clearly covers no delay bounds as well.

No internal delay allowance: Here, $T(f_i) = EAT_i$. This model is sometimes considered by ATSP, even though under this assumption, it is possible to produce infeasible flight-to-slot assignments. However, such allocations may be considered more equitable and are still useful to the ASU's since unusable slots can be traded for usable later slots using the GDP mechanisms (see Vossen and Ball [11, 12]).

Weakly increasing $j(i)$: Here, $i \leq k$ implies $j(i) \leq j(k)$. This is a generalization of the no internal delay allowance assumption that covers a broader class of inputs.

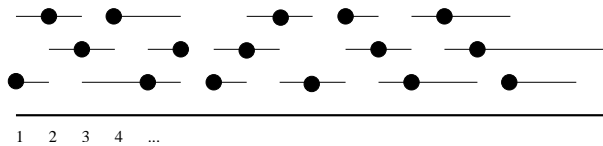


Figure 1: Glover's algorithm.

We refer to Vossen and Ball [11] for a detailed a description and analysis of ground delay program algorithms. The *Ration-By-Schedule* (RBS) algorithm gives an initial assignment based on the scheduled times of arrival. Due to flight cancelations and other flight delays, this schedule might contain infeasible flight assignments. To address this possibility, the *Compression* algorithm carries out exchanges of slots among airlines. Slots assigned to canceled flights that are otherwise unusable by the owner/airline are exchanged for later slots to which that airline can assign one of its flights.

3 Maximum Matchings and Glover's Algorithm

Consider the problem of finding a maximum matching in the flight allocation graph G . The bipartite graph G has a very simple structure: the neighbors of node f_i are $s_{j(i)}, s_{j(i)+1}, \dots, s_{j'(i)}$. Thus, the neighbors of f_i form an *interval* (consecutive nodes). A bipartite graph with this property is called *convex* on S . Glover [7] found a simple algorithm for finding a maximum matching in a bipartite convex graph (see also Asratian [1], section 5.3).

Glover's algorithm:

Step 1: Start with $M = \emptyset$.

Step 2: for $j = 1, 2, \dots, n$ choose, if any, a free neighbor f_i of s_j with $j'(i)$ minimal and add $[f_i, s_j]$ to M .

We also propose a modification, called *Modified Glover's Algorithm*, where in Step 2, we choose a neighbor node f_i with i minimal. We note that with the weakly increasing in $j'(i)$ assumption, this change represents a specialization of Glover's algorithm. Specifically, it is Glover's algorithm with a tie breaking rule for the case where multiple eligible neighbors i have the same value of $j'(i)$. While we allow for $T(f_{i'}) = T(f_{i''})$ for $i' \neq i''$, when the flight ordering is produced we assume a deterministic rule is used to break "ties", implying that the output is Modified Glover's Algorithm is unique (this is equivalent to assuming that the T_i are distinct). We note that Modified Glover's Algorithm can be viewed as a reinterpretation of ration-by-schedule (RBS) as defined in Vossen and Ball [11] and Wambsganss [13].

The correctness and efficiency of Glover's algorithm is given by the following Theorem (see Asratian [1] or Glover [7] for a proof.)

Theorem 1. *Given a bipartite graph $G = (F, S, E)$ that is convex on F , Glover's algorithm will find a maximum matching in G in $O(|V| + |E|)$ steps.*

An illustration of the algorithm is shown in Figure 1. The horizontal line at the bottom corresponds to S and each interval above corresponds to a vertex f_i and shows the interval $j(i), j(i)+1, \dots, j'(i)$. Each solid circle corresponds to an edge in the maximum matching found by Glover's algorithm.

We remark that a linear time algorithm for finding a maximum matching in a convex bipartite graph was described in Steiner and Yeomans [10].

Based on our earlier remarks, the following corollary immediately follows.

Corollary 2. *Given a flight allocation graph with $j'(i)$ weakly increasing. Then Modified Glover's Algorithm finds a maximum matching.*

We point out one further important property of Glover's Algorithm. A bipartite graph $G = (F, S, E)$ that is convex on S remains convex when nodes are deleted from S in reverse order, so that Glover's Algorithm finds a maximum matching for G restricted to all ordered subsets of S . Given a flight allocation graph, G , and $S_k = \{s_1, \dots, s_k\}$ we define G restricted to S_k to be the flight allocation graph obtained by deleting nodes $\{s_{k+1}, \dots, s_n\}$ and all adjacent edges. A matching restricted to S_k is defined in a similar way.

Corollary 3. *Let G be a flight allocation graph and let M be a matching obtained by Modified Glover's Algorithm. Then for any k , M restricted to S_k is the maximum matching for G restricted to S_k .*

This property will be quite useful when analyzing fundamental properties of delay-minimizing solutions in Section 4.1.

4 Solution Criteria and Properties

Both the ATSP and the ASU's have an obvious objective in any allocation: minimizing delays. However, there are certain nuances in both cases, that require more careful consideration of specific objective functions. The ATSP must allocate slots to multiple sometimes-competing entities. As such there is a significant concern with insuring equity in such allocations. From the ASU perspective, it is typically the case that the cost of operating a flight can grow non-linearly with delay so that simply minimizing total flight delay might not be the best policy. We will use two generic objectives: efficiency and equity. We will use total delay as a measure of efficiency; basic solution properties are given in Section 4.1. It can be difficult to get a simple generally accepted measure of equity. In Section 4.2, we will first discuss a strong solution concept and then define a related equity metric.

4.1 Properties of Delay-Minimizing Solutions

We now define some notation to define the delay associated with a matching M in G . We denote by $F(M)$ the flights covered by M and $S(M)$ the slots covered by M . M defines a mapping, also denoted by M , of $F(M)$ into J in the natural way where $M(i) = j$ whenever M contains the edge $[f_i, s_j]$. Recall that the scheduled time of f_i is $T(f_i)$ and the start time of slot s_j is $T(s_j)$ so that the delay $d^M(i)$ of flight f_i under the allocation M equals

$$d^M(i) = T(s_{M(i)}) - T(f_i),$$

and this delay is clearly nonnegative.

In order to define an efficiency (delay) measure that takes into account deleted flights, we associate with each deleted flight, a cost, Δ , measured in delay minutes. Thus, the cost associated with any allocation (matching) M is:

$$\begin{aligned} \hat{D}_\Delta(M) &= \sum_{i \in F(M)} d^M(i) + |F - F(M)|\Delta \\ &= \sum_{j \in S(M)} T(s_j) - \sum_{i \in F(M)} T(f_i) + |F - F(M)|\Delta \end{aligned}$$

The important property of this delay function is that the total delay depends only on the set of flights assigned to a slot ($F(M)$) and the set of slots assigned a flight ($S(M)$). Here we will describe properties for the case where $F(M) = F$. In Section 6, we investigate problems involving flight deletions.

When $F = F(M)$, in the expression for $\hat{D}_\Delta(M)$, $|F - F(M)|\Delta = 0$ and $\sum_{i \in F(M)} T(f_i)$ is constant so that minimizing $\hat{D}_\Delta(M)$ is equivalent to minimizing $\sum_{j \in S(M)} T(s_j)$. The underlying combinatorial problem can be stated as follows: we are given a weight $T(s_j)$ for each $s_j \in S$ and we wish to find a minimum weight subset S' of S such that there exists a perfect matching between S' and F . The problem of finding such an S' is the problem of finding a minimum weight basis in a transversal matroid (see Brualdi [4], Welsh [14]). This optimization problem can be solved via the following Greedy algorithm:

Greedy-Slot:

step 0: Let G be a flight allocation graph for which there exists a matching M with $F(M) = F$.

Set $S' = \emptyset$ and $j = 0$

step 1: Set $j = j + 1$. If there exists a matching between F and $S' \cup \{s_j\}$ that covers $S' \cup \{s_j\}$ then set $S' = S' \cup \{s_j\}$. If $|S'| = |F|$ then stop; otherwise, repeat step 1.

Matroid greedy algorithms (Edmonds [5], Welsh [14]) for finding minimum weight bases must add elements in order of increasing element weight. The order previously specified for the slots implies that this is carried out in step 1. We denote such a delay minimizing subset of S by $S^*(G)$. $S^*(G)$ can be simply characterized. To do so, we first define some notation. Given a set $\{1, 2, \dots, k\}$ and two subsets U_1 and U_2 of $\{1, 2, \dots, k\}$, we say that U_1 is *lexicographically* greater than or equal to U_2 , denoted by $U_1 \succeq_L U_2$ if $i' \in U_1$ where $i' = \min\{i : i \in U_1 \cup U_2 \text{ but } i \notin U_1 \cap U_2\}$. It can easily be seen that

$$S^*(G) \succeq_L S' \quad \text{for all } S' = S(M) \text{ for some matching } M.$$

This property leads directly to the following result.

Proposition 4. *Let M be an arbitrary maximum matching in G . Then*

- (i) $|S(M) \cap \{s_1, s_2, \dots, s_j\}| \leq |S^*(G) \cap \{s_1, s_2, \dots, s_j\}|$ for $1 \leq j \leq n$.
- (ii) $\max\{j : s_j \in S(M)\} \geq \max\{j : s_j \in S^*(G)\}$.

Property (i) implies that it is impossible to let more flights arrive up to any point in time than the number that arrive under an M^* with $S(M^*) = S^*(G)$. Property (ii), on the other hand, states that the total “time span” used under $S^*(G)$ is smallest possible. In other words, $\max\{j : s_j \in S^*(G)\}$ is the smallest number of slots needed for a full flight allocation to exist.

Greedy-Slot as presented could be implemented so that a matching between F and S' is always maintained. Thus, Step 2 could be implemented by finding a single augmenting path that requires $O(|E|)$ time giving overall complexity of $O(|E||V|)$. However, it follows directly from Corollary 3 that $S^*(G)$ is naturally produced by Glover’s algorithm:

Proposition 5. *Let G be a flight allocation graph for which there exists a matching M with $F(M) = F$ and let M be the matching produced by Glover’s algorithm. Then, $S(M) = S^*(G)$.*

We now wish to show a further property of delay minimizing matchings. We first assume that some flight can be matched to the first slot, s_1 (otherwise we could delete early slots without changing the basic problem structure).

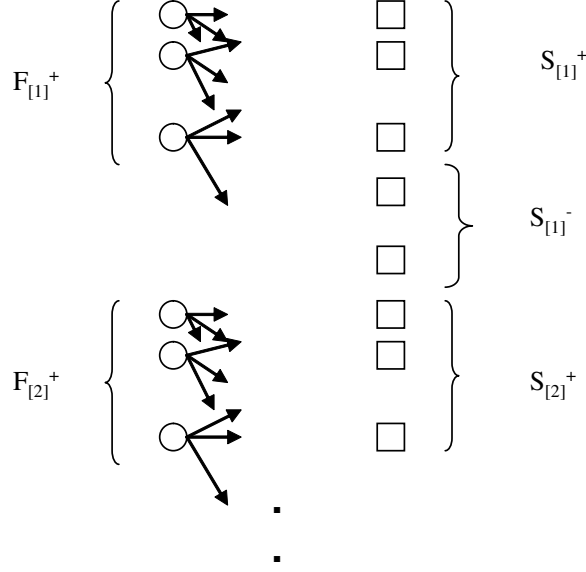


Figure 2: Graph Partition Associated with Delay Minimizing Matching.

Denote by $INT[j, j']$ the sequence of consecutive integers between two integers j and j' with $j \leq j'$. Note that $S^*(G)$ can be written as a sequence of intervals:

$$INT[j_1 = 1, \hat{j}_1], INT[j_2, \hat{j}_2], \dots, INT[j_q, \hat{j}_q],$$

where for each ℓ , $\hat{j}_\ell + 1 \leq j_{\ell+1} - 1$ so that $[\hat{j}_\ell + 1, j_{\ell+1} - 1]$ is also an interval, which we call a “gap”. We denote by $S_{[\ell]}^+$ the ℓ th interval in the partition of $S^*(G)$ and $S_{[\ell]}^-$ the next adjacent gap. Thus, $S^*(G) = S_{[1]}^+ \cup S_{[2]}^+ \cup \dots \cup S_{[q]}^+$ and $S = S_{[1]}^+ \cup S_{[1]}^- \cup S_{[2]}^+ \cup S_{[2]}^- \cup \dots \cup S_{[q]}^+ \cup S_{[q]}^-$.

We now claim that in any delay minimizing matching M , for any ℓ , it must be the case that all of the f_i matched to a slot in $S_{[\ell]}^+$ must have $j_\ell \leq j(i) \leq \hat{j}_\ell$. Clearly, $j(i) \leq \hat{j}_\ell$ otherwise f_i would not be adjacent to a slot node in $S_{[\ell]}^+$. However, additionally $j_\ell \leq j(i)$ because, otherwise a matching with a lexicographically smaller $S(M)$ could be obtained by matching some flight in $F(S_{[\ell]}^+)$ to a slot in the gap $S_{[\ell-1]}^-$. Thus, the slot partition of $S^*(G)$ leads naturally to a flight partition as well. That is, if we define $F_{[\ell]}^+ = \{f_i : j_\ell \leq j(i) \leq \hat{j}_\ell\}$, we see that any delay minimizing maximum matching can be partitioned into a set of maximum matchings between each of $S_{[\ell]}^+$ and $F_{[\ell]}^+$. Figure 2 illustrates this structure.

4.2 Equitable Allocation

Many challenges exist in measuring equity of allocations. On the other hand, there is a rich history related to fair allocation schemes and much experience to build upon (see Young [15]). One specific challenge is that it can be difficult to develop simple linear expressions (objective functions) since the relative performance of various players is usually an important consideration. We introduce a partial order on the set of allocations that allows comparison of equity levels.

4.2.1 Majorization

We now introduce the concept of majorization and show that it provides an effective tool for analyzing equity in delay allocation. Majorization has been found to be very useful in the study of different inequalities in mathematics and statistics. The classic reference in this area is the book by Marshall and Olkin [9].

Majorization is an ordering (actually a preorder) of pairs of vectors. It reflects that the components of one vector are more evenly distributed than the components in the other vector. One says that a vector $x = (x_1, x_2, \dots, x_n)$ is *weakly majorized* by $y = (y_1, y_2, \dots, y_n)$ if the following inequalities hold:

$$\sum_{j=1}^k x_{[j]} \leq \sum_{j=1}^k y_{[j]} \quad (k \leq n).$$

Here $x_{[j]}$ denotes the j th largest component in x , so $x_{[1]} \geq x_{[2]} \geq \dots \geq x_{[n]}$. We write $x \prec_w y$ if x is weakly majorized by y . If the equation holds and also $\sum_j x_j = \sum_j y_j$, then one says that x is *majorized* by y and denote this by $x \prec y$. A basic result from majorization theory is a theorem by Hardy, Littlewood and Pólya [8] which says that $x \prec y$ if and only if $x = yA$ for some $n \times n$ doubly stochastic matrix. Recall that A is called doubly stochastic when it is nonnegative and all row and column sums are 1.

Majorization ideas provide a natural way to consider equity in delay allocations. Generally, when delays must be imposed it is desirable to distribute those delay evenly among the affected parties. In this case the ‘‘affected parties’’ could either be defined to be individual flights or ASU’s. Here we base our analysis on individual flights; we leave analysis of the ASU (airline) case to a future paper. In this section we continue to assume that $F = F(M)$ so that M is of size m . The delay associated with a matching M may be organized into the *delay vector*

$$d^M = (d_1^M, d_2^M, \dots, d_m^M).$$

Each delay vector d^M is nonnegative and has length m . We are interested in the mapping $M \rightarrow d^M$. Hereafter, by a matching in G we shall mean a maximum matching, i.e., a matching of size m , and we let \mathcal{M} denote the set of all such matchings. We now define a certain ordering on \mathcal{M} .

Definition 6. *Let $M, N \in \mathcal{M}$. We say that M is majorized by N , and denote this by $M \prec N$, provided that $d^M \prec d^N$, i.e., the delay vector d^M is majorized by the delay vector d^N . The notion of weak majorization is defined similarly, i.e., $M \prec_w N$ whenever $d^M \prec_w d^N$.*

This majorization ordering is a *preorder* on \mathcal{M} , i.e., the following holds for all $M, N, K \in \mathcal{M}$:

- $M \prec M$ (*reflexive*),
- if $M \prec N$ and $N \prec K$, then $M \prec K$ (*transitive*).

Weak majorization satisfies the same properties. The motivation for our definition is that it seems natural to prefer the matching M to another matching N when the delays of flights under the allocation M is more evenly distributed than under N . This is based on the assumption that there are no preferences among the flights, so ideally one would like equal delays on the flights, and clearly also this delay to be smallest possible.

We are interested in matchings that are small in this (weak) majorization order. Thus we would like to find algorithms that produce majorization-minimal matchings. In this connection a certain interchange operation is useful.

Theorem 7. Let $N \in \mathcal{M}$ and let $1 \leq i < k \leq m$. Assume that $j(i) \leq M(k) < M(i)$. Let N be obtained by interchanging i and k , meaning that $N(i) = M(k)$, $N(k) = M(i)$, and $N(r) = M(r)$ ($r \neq i, k$). Then $M \in \mathcal{M}$ and $M \prec N$.

Proof. First, N is also a maximum matching in G as $|M| = |N|$ and both $[f_i, s_{M(k)}]$ and $[f_k, s_{M(i)}]$ are edges in E because $j(i) \leq M(k)$. Then the i th and k th components of the delay vectors d^M and d^N are given by

$$\begin{aligned} d_i^M &= T(s_{M(i)}) - T(f_i), & d_k^M &= T(s_{M(k)}) - T(f_k), \\ d_i^N &= T(s_{M(k)}) - T(f_i), & d_k^N &= T(s_{M(i)}) - T(f_k). \end{aligned}$$

Therefore, d^N is obtained from d^M by subtracting $\delta := T(s_{M(i)}) - T(s_{M(k)})$ from the i th component and adding δ to the k th component. Note that $\delta > 0$ as $M(i) > M(k)$ implies that $T(s_{M(i)}) > T(s_{M(k)})$. Moreover

$$\begin{aligned} d_i^M - d_k^M &= (T(s_{M(i)}) - T(f_i)) - (T(s_{M(k)}) - T(f_k)) = \\ &\delta + (T(f_k) - T(f_i)) > \delta, \end{aligned}$$

as $T(f_k) > T(f_i)$ (recall that $k > i$). Let $\lambda = (d_i^M - d_k^M - \delta) / (d_i^M - d_k^M)$. Then $0 < \lambda < 1$. Define the $m \times m$ matrix $A = [a_{ij}]$ as follows: $a_{ii} = a_{kk} = \lambda$, $a_{ik} = a_{ki} = 1 - \lambda$, $a_{rr} = 1$ for all $r \leq m$, $r \neq i, k$, and all other entries of A are zero. This matrix is doubly stochastic. Moreover, a simple calculation now shows that

$$d^M = d^N A.$$

Thus, by the mentioned theorem by Hardy, Littlewood and Pólya, we conclude that $d^N \prec d^M$. So $M \prec N$ as desired. \square

The content of the theorem is that, under the mentioned assumptions, the interchange will make the delays of flights f_i and f_k more equal (that is, the difference between the delays of flights f_i and f_k decreases) while other delays remain unchanged. These interchanges can be inferred based solely on the structure of the graph G (via the mapping $i \rightarrow j(i)$ and $i \rightarrow j'(i)$), so that the information about the original scheduled arrival times T_i ($i \leq m$) is not necessary. It is interesting to note that the non-existence of such exchanges is given as a general characteristic of equitable allocations (see Young [15], pg 77).

Theorem 7 provides an approach to improving a given matching from the perspective of majorization (and equity). It would then be natural to ask whether a globally "best" perfect matching exists. We call a matching $M^* \in \mathcal{M}$ such that $M^* \prec N$ for all $N \in \mathcal{M}$ an *M-best matching*. In Section 5 we will investigate conditions under which an M-best matching exists. Before addressing this question, we develop relationships between majorization and Schur-convexity.

4.2.2 Schur-convexity and the *OPTIFLOW* model

Theorem 7 may be used to establish inequalities involving delay vectors. A function $\phi : \mathbb{R}_+^m \rightarrow \mathbb{R}$ is called *Schur-convex* (on \mathbb{R}_+^m) if $\phi(x) \leq \phi(y)$ whenever $x, y \in \mathbb{R}_+^m$ and $x \prec y$. Schur-convexity is discussed in detail in Chapter 3 of Marshall and Olkin [9] and several classes of Schur-convex functions are established. An important class of Schur convex functions consists of functions ϕ given by

$$\phi(x) = \sum_{j=1}^m f(x_j),$$

where $f : \mathbb{R} \rightarrow \mathbb{R}$ is a convex function. Actually, a theorem of Schur, and of Hardy, Littlewood and Pólya [8, 9], says that $x \prec y$ if and only if $\sum_{k=1}^m f(x_k) \leq \sum_{k=1}^m f(y_k)$ for *all* convex functions

$f : \mathbb{R} \rightarrow \mathbb{R}$. A similar theorem (see also Marshall and Olkin [9]) holds for weak majorization: $x \prec_w y$ if and only if $\sum_{k=1}^m f(x_k) \leq \sum_{k=1}^m f(y_k)$ for all nondecreasing convex functions $f : \mathbb{R} \rightarrow \mathbb{R}$. When these results are combined with Theorem 7, one obtains the following result.

Corollary 8. *Let $f : \mathbb{R} \rightarrow \mathbb{R}$ be a nondecreasing convex function. Let M be a maximum matching in G where i and j are out of order for some pair i, j and let N be defined as in Theorem 7; let M^* be an M -best matching (if one exists). Then*

$$\sum_{k=1}^m f(d_k^{M^*}) \leq \sum_{k=1}^m f(d_k^N) \leq \sum_{k=1}^m f(d_k^M).$$

More generally, if $\phi : \mathbb{R}_+^m \rightarrow \mathbb{R}$ is nondecreasing and Schur-convex, then $\phi(d^{M^*}) \leq \phi(d^N) \leq \phi(d^M)$.

Some examples of Schur convex functions that may be of interest in this connection are:

1. $\phi(x) = \sum_j f(x_j)$ where $f(x) = (x - \alpha)^+$ for $\alpha \geq 0$;
2. $\phi(x) = \sum_j f(x_j)$ where $f(x) = x^\alpha$ for $\alpha \geq 1$;
3. $\phi(x) = \sum_{k=1}^s x_{[k]}$ for some positive integer s ;
4. $\phi(x) = \sum_{k=1}^m (x_j - \bar{x})^2$ where $\bar{x} = (1/m) \sum_j x_j$.

In example 1 the inequalities in Corollary 8 involve the sum of delays above the threshold value α . And in example 2 one wants to avoid long delays by using the terms d_j^α . In example 3 one sums the s largest delays and in example 4 we have the variance of the delay vector.

A basic model for finding flight allocations is the OPTIFLOW model introduced in Ball et al. [2] and further investigated in Vossen and Ball [11]. In this model, the allocation of flight is formulated as an assignment problem where the cost of assigning a flight f_i to slot s_j is defined as

$$c_{ij} = w_i(T(s_j) - T(f_i))^{1+\epsilon}.$$

Consider now the case when all weights are one, $w_i = 1$ ($i \in M$), so there are no preferences among the flights. Let M be a maximum matching in G . Its incidence vector $x = \chi^M$ is then feasible in the OPTIFLOW model and the corresponding value of the objective function is

$$\sum_i \sum_j c_{ij} \chi_{ij}^M = \sum_i c_{i, M(i)} = \sum_i (d^M(i))^{1+\epsilon}.$$

The function $f(x) = x^{1+\epsilon}$ is nondecreasing and convex, so the objective function is Schur-convex. Thus Corollary 8 may be applied and, in particular, we conclude that the M -best matching is optimal in the OPTIFLOW model. Thus, the development above casts light on the structure of an optimal solution of the OPTIFLOW model in the case with $w_i = 1$. Moreover, the partial ordering of delay vectors given by majorization shows additional properties of the relationship between different flight allocations.

5 Allocations That Cover All Flights

In this section we investigate allocations that cover all flights, i.e. matchings with $F(M) = F$. Section 5.1 treats the case of a general flight allocation graph. While delay-minimizing matchings can be found by solving an appropriate assignment model, we show that in general an M -best matching does not exist. In Section 5.2 we treat the case of weakly increasing $j'(i)$ and show that modified Glover's algorithm always produces an M -best matching. We then analyze a particularly elegant solution for a special case: fork matchings (Section 5.3). Section 6 will cover the case with flight deletions.

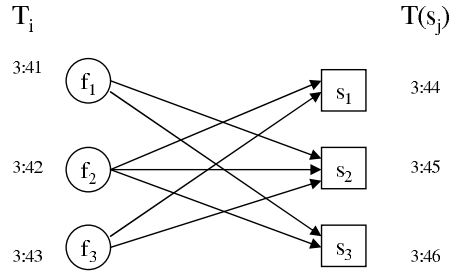


Figure 3: Flight allocation graph with two matchings that cannot be improved using 2-by-2 exchanges: $M_1 : \{[f_1, s_3], [f_2, s_2], [f_3, s_1]\}, d_{M_1} = (5, 3, 1)$; $M_2 : \{[f_1, s_3], [f_2, s_1], [f_3, s_2]\}, d_{M_2} = (5, 2, 2)$; $M_3 : \{[f_1, s_2], [f_2, s_3], [f_3, s_1]\}, d_{M_3} = (4, 4, 1)$.

5.1 General Flight Allocation Graphs

For the case of a general flight allocation graph we can find a delay-minimizing perfect matching by finding $S^*(G)$ as described in Section 4 and then finding any perfect matching between F and $S^*(G)$. Ideally, one would like to find such a perfect matching that is M-Best. However, the example given in Figure 3 provides a 3-by-3 flight allocation graph, G^c , for which no M-Best perfect matching exists. G^c contains three perfect matchings. Each of M_2 and M_3 can be obtained from M_1 by executing a 2-by-2 exchange as described in Theorem 7. The ordered delay vectors for M_1 , M_2 and M_3 are $(5, 3, 1)$, $(5, 2, 2)$ and $(4, 4, 1)$ respectively. Note that $M_2 \prec M_1$ and $M_3 \prec M_1$ but M_2 and M_3 are non-comparable with respect to majorization. It is also clear that no majorization-improving exchanges can be executed on either M_2 or M_3 . In that sense, these two matchings represent “local optima”.

5.2 Allocations Produced by Modified Glover’s Algorithm

We now show that Modified Glover’s algorithm produces an M-best matching for the case of weakly increasing $j'(i)$.

Theorem 9. *Let G be a flight allocation graph with $j'(i)$ weakly increasing in i and such that there exists a matching that covers all flights. Let M^* be the matching produced by Modified Glover’s algorithm. Then*

$$M^* \prec_w M \quad \text{for all } M \in \mathcal{M},$$

i.e. M^ is an M-best matching.*

Proof. Let $M \in \mathcal{M}$ be an arbitrary matching that covers F but that was not produced by Glover’s algorithm. Then we know for some j , $M(i) = j$ and there exists an i' with $j'(i') < j'(i)$ and $M(i') > j$. Since $j'(*)$ is weakly increasing, it follows that $i' \leq i$ and if the assignments of i and i' are interchanged to produce a new matching $M' \in \mathcal{M}$ then, by Theorem 7, we have that $M' \prec_w M$. Now, this process can be repeated until the M^* produced by Modified Glover’s Algorithm is obtained. By the transitivity of \prec_w , it follows that $M^* \prec_w M$. Since we can start this process with an arbitrary M and always end up with the M^* produced by Modified Glover’s Algorithm, it must be that M^* is M-best. \square

Corollary 10. *Let G be a flight allocation graph with $j'(i)$ weakly increasing in i and such that there exists a matching that covers all flights. Then G contains an M-Best matching.*

We note that the “core” inputs to ground delay programming planning problems are the $T(f_i)$ and $T(s_j)$. On the other hand, $j'(i)$ can be viewed as a type of “control knob” that can be set based on various criteria. This Corollary and the counter-example given in Section 5.1 suggest that, in order to guarantee the existence of equitable allocations, any such criteria should insure that $j'(i)$ is weakly increasing.

5.3 A Special Case: Fork Matchings

In this section we consider a special case of our general model where we have that both $j'(i)$ and $j(i)$ are weakly increasing functions of i . The domain of relevance of these assumptions was discussed in Section 2. We denote this special case by *FORK*. A bipartite graph with color classes U and W is called *doubly convex* if it is convex on both U and W (after suitable ordering of the nodes).

Lemma 11. *In the FORK situation the graph G is doubly convex.*

Proof. To prove the result we must take an arbitrary j and show that the neighbors of s_j form an interval. For such a j , let f_{i^o} and f_{i^*} be the neighbors with min and max i values respectively and let $f_{i'}$ be an arbitrary flight node with $i^o \leq i' \leq i^*$. To prove the result we must show that $f_{i'}$ is a neighbor of s_j . Since i^o and i^* are neighbors we have:

$$j(i^o) \leq j \leq j'(i^o); \quad j(i^*) \leq j \leq j'(i^*),$$

and since $j(i)$ and $j'(i)$ are weakly increasing we have

$$j'(i^o) \leq j'(i'); \quad j(i') \leq j(i^*).$$

Putting these together we have:

$$j(i') \leq j(i^*) \leq j \leq j'(i^o) \leq j'(i'),$$

which implies $f_{i'}$ is adjacent to s_j and the result follows. \square

We want to find a maximum matching in G that covers all flights. Since the graph is convex we may use Modified Glover’s algorithm and the results of the previous section apply. However, the very special doubly convex structure of G makes it possible to determine a maximum matching explicitly. We should note that the construction we now describe assumes that a matching covering all flights exists. Specifically, this construction is not guaranteed to find a matching covering all flights for any doubly convex graph. The following terminology will be convenient. Let $i \in I$ and let t_i be the largest nonnegative integer such that $j(i+k) \leq j(i) + k \leq j'(i+k)$ for all $0 \leq k \leq t_i$. Thus E contains the edges

$$[f_i, s_{j(i)}], [f_{i+1}, s_{j(i)+1}], \dots, [f_{i+t_i}, s_{j(i)+t_i}].$$

This edge set is called a *fork* and it is denoted by $F(i)$. Moreover, we define $I(i) = \{i, i+1, \dots, i+t_i\}$. Let F^* be the following union of forks

$$F^* = F(i_1) \cup F(i_2) \cup \dots \cup F(i_p),$$

where $i_1 = 1$, $i_t = |F(i_1)| + \dots + |F(i_{t-1})| + 1$ ($2 \leq t \leq p$) and $m \in I(i_p)$. Then F^* is a maximum matching in G and we call F^* the *fork matching*. Moreover, F^* is the matching that would be produced by the application of Modified Glover’s Algorithm in the FORK situation. Thus, we have

Theorem 12. *In the FORK situation, F^* has the M^* properties given in Theorem 9 and Proposition 4.*

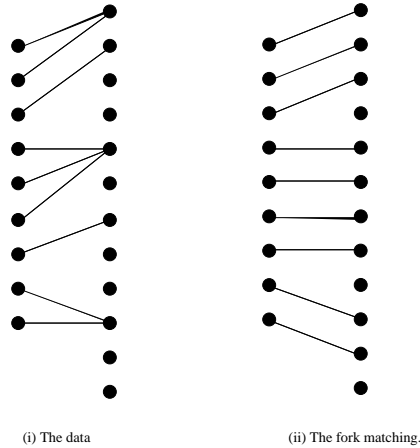


Figure 4: The edges $[f_i, s_{j(i)}]$ and the fork matching.

Example 1. Figure 4 shows an example with $m = 9$, $n = 12$. Part (i) shows the edges $[f_i, s_{j(i)}]$ ($i \leq m$) and part (ii) shows the fork matching F^* .

Consider again the fork matching F^* . It uses the slots $S(F^*)$. The slots $S - S(F^*)$ are not used because the “next flights” cannot arrive early enough. Thus, at the start of each fork in F^* one has caught up with the delays and is waiting for the next group to arrive. If there are not too many cancelled flights, one may very well have a situation where F^* consists of a single fork.

Fork matchings exhibit a high degree of structure, which allows them to be easily constructed, and which may lead to further useful properties.

6 Models with Flight Deletions

In this section we consider problem instances where the structure of G and the cost function $\hat{D}_\Delta(M)$ are such that minimizing $\hat{D}_\Delta(M)$ implies that some flights must be deleted, i.e. $F \neq F(M)$. We consider two cases. Note from the definition of $\hat{D}_\Delta(M)$ that when Δ is sufficiently large, minimizing $\hat{D}_\Delta(M)$ implies that $|F(M)|$ is maximized. This case is treated in Section 6.1 and the more general case is treated in Section 6.2. Section 6.3 extends these results to the case of flight-specific flight deletion costs.

6.1 Optimally Deleting the Minimum Number of Flights

In this section we treat of the problem of finding an M that minimizes $\hat{D}_\Delta(M)$ assuming that Δ is “large”. We further assume that there does not exist a matching M with $F(M) = F$ so that some flights must be deleted. However, for sufficiently large Δ , a solution that minimizes $\hat{D}_\Delta(M)$ must minimize $|F - F(M)|$, i.e. maximize $|F(M)|$. Thus, we can restrict our attention to maximum matchings in G and the optimization problem becomes:

$$\begin{aligned} \min \quad & \sum_{s_j \in S(M)} T(s_j) - \sum_{f_i \in F(M)} T(f_i) \\ \text{s.t.} \quad & M \text{ is a maximum matching in } G. \end{aligned}$$

In Section 4.2 we characterized a delay-minimizing $S(M) = S^*(G)$ for the case where F could be covered by a matching. It is clear that this analysis carries over to the case where we wish to cover an $F' \subset F$:

Proposition 13. *Let G be a flight allocation graph and $F' \subset F$. Then, if M' is a matching with $F(M') = F'$ that minimizes $\hat{D}_\Delta(M)$ then $S(M') \succeq_L S(M)$ for all matchings M with $F(M) = F'$.*

It is clear from the form of $\hat{D}_\Delta(M)$ for this case that, just as we seek to minimize $\sum_{s_j \in S(M)} T(s_j)$, we also seek to maximize $\sum_{f_i \in F(M)} T(f_i)$. It is also clear that the same arguments that imply Proposition 13 imply:

Proposition 14. *Let G be a flight allocation graph and $S' \subset S$. Then, if M' is a matching with $S(M') = S'$ that minimizes $\hat{D}_\Delta(M)$ then $F(M') \preceq_L F(M)$ for all matchings M with $S(M) = S'$.*

Here, lexicographically less than or equal to (\preceq_L) is defined in an analogous way to \succeq_L .

We extend the definition of $S^*(G)$ to be the lexico-maximum S with $S = S(M)$ for some maximum matching M and define $F^*(G)$ to be the lexico-minimum F with $F = F(M)$ for some maximum matching M . Now it is clear that if there exists a maximum matching M^* with $S(M^*) = S^*(G)$ and $F(M^*) = F^*(G)$ then M^* would minimize $\hat{D}_\Delta(M^*)$. In fact, such an M^* does exist as we now show. The essential result is given by the following proposition.

Proposition 15. *Let G be a flight allocation graph. Let $F' = F(M')$ for some maximum matching M' . Then, there exists a maximum matching M^o such that $F' = F(M^o)$ and $S(M^o) = S^*(G)$.*

Proof. Let M^+ be the maximum matching with $F' = F(M^+)$ and such that $S(M^+)$ is lexicographically maximum. Now in order to prove the result by contradiction, we assume that $S(M^+) \neq S^*(G)$. Let $s_{j'}$ be the minimum slot index where $S(M^+)$ and $S^*(G)$ differ, i.e. $s_{j'} \in S^*(G)$ but $s_{j'} \notin S(M^+)$, and for all $j < j'$, $s_j \in S^*(G) \implies s_j \in S(M^+)$. Now note that there can be no $f \in F - F'$ adjacent to a slot in $S - S(M^+)$, otherwise M^+ could trivially be increased in cardinality. Further, no f matched to a slot s_j with $j > j'$ can be adjacent to a slot s_j with $j \leq j'$, otherwise, by the interval properties of neighbors, such an f would be adjacent to $s_{j'}$ and M^+ could be trivially adjusted to create an M^{++} with $S(M^{++}) \succeq_L S(M^+)$. Now we know there is a matching \hat{M} , that covers all slots in $S^*(G) - \{s_{j'+1}, \dots, s_n\}$. By the above arguments this matching does not use any f matched to a slot s_j with $j > j'$ in M^+ . Thus, this matching can be combined with those elements of M^+ adjacent to s_j with $j > j'$ in M^+ to obtain a matching of cardinality one greater than $|M^+|$. This contradicts that M^+ was maximum and the proof is complete. \square

Our main result now follows.

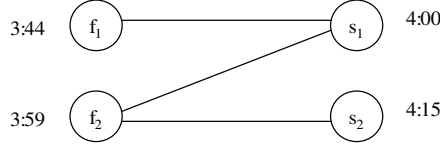
Theorem 16. *Let G be a flight allocation graph. Then, there exists a maximum matching M^* with $S(M^*) = S^*(G)$ and $F(M^*) = F^*(G)$. Furthermore, M^* minimizes $\hat{D}_\Delta(M^*)$.*

It follows from Proposition 5 that Modified Glover's algorithm produces a matching M with $S(M) = S^*(G)$. Once $S^*(G)$ is found, the following greedy algorithm can be used to find $F^*(G)$ and M^* .

Greedy-Flight:

step 0: Let G be a flight allocation graph and assume that $S^*(G)$ has been found. Set $F^* = \phi$ and $i = m + 1$.

step 1: Set $i = i - 1$. If there exists a matching, M^* , between $S^*(G)$ and $F^* \cup \{f_i\}$ that covers $F^* \cup \{f_i\}$ then set $F^* = F^* \cup \{f_i\}$. If $|F^*| = |S^*(G)|$ then stop; otherwise, repeat step 1.



$\Delta = 30$
 Max matching solution: f_1-s_1, f_2-s_2 ; tot delay = $16+16 = 32$.
 Min delay solution: $f_1 - \text{cancel}; f_2-s_1$; tot delay = $30 + 1 = 31$.

Figure 5: Min Cost Matching that is not Maximum.

Greedy-Flight can be implemented by maintaining a matching between F^* and $S^*(G)$. Thus, Step 2 needs only search for a single augmenting path, which requires $O(|E|)$ time. The overall complexity then becomes $O(|E||V|)$.

It can easily be seen that the structure of delay minimizing matching identified in Section 4.1 applies in this more general setting. That is, a delay minimizing maximum matching M^* can be partitioned into delay minimizing maximum matchings between each of $S_{[\ell]}^+$ and $F_{[\ell]}^+$. Note that in this case it can be that $|S_{[\ell]}^+| < |F_{[\ell]}^+|$. We should note that in general (when Δ is not large) total cost is not minimized by a maximum matching. This is demonstrated by the example in Figure 5.

6.2 A Parametric Approach to Flight Deletions

This section introduces a somewhat different approach to flight deletions. Specifically, we describe a procedure that determines a delay-minimizing allocation *given that exactly L flights have to be deleted*. The resulting procedure may be viewed as a parametric optimization problem, and our objective is to determine how the allocation changes if L is increased.

We assume that both $j'(i)$ and $j(i)$ are weakly increasing functions of i . Note that, according to Lemma 11, this implies that the flight allocation graph G is doubly convex. A general model for this problem is as follows.

P(L):

$$z_P(L) = \min \sum_{i \in I} T(f_i) y_i + \sum_{ij \in E} T(s_j) x_{ij}$$

s.t.

$$y_i + \sum_{j:ij \in E} x_{ij} \geq 1 \quad (i \in I) \quad (1)$$

$$\sum_{i:ij \in E} x_{ij} \leq 1 \quad (j \in J) \quad (2)$$

$$\sum_{i \in I} y_i \leq L \quad (3)$$

$$x_{ij} \geq 0, \quad y_i \geq 0, \quad (4)$$

where $x_{ij} = 1$ iff flight f_i is assigned to slot s_j , and $y_i = 1$ iff flight f_i is canceled. For notational simplicity we write $ij \in E$ in place of $[f_i, s_i] \in E$. We note that any optimal solution will satisfy the first constraint at equality, however, an inequality is used to simplify the development of dual

solutions. The objective function follows since

$$\begin{aligned} \sum_{ij \in E} (T(s_j) - T(f_i))x_{ij} &= \sum_{ij \in E} T(s_j)x_{ij} - \sum_{i \in I} (1 - y_i)T(f_i) \\ &= \sum_{i \in I} T(f_i)y_i + \sum_{ij \in E} T(s_j)x_{ij} - \sum_{i \in I} T(f_i), \end{aligned}$$

where the last term represents a constant. While the resulting optimization problem can be solved efficiently, our focus here is on the development of a greedy procedure that provides insight into structural properties of the resulting allocations.

We furthermore assume that the structure of G is such that $P(L)$ is feasible for any $L \geq 0$, that is, we assume there exists a matching M such that $F(M) = F$. If this is not the case, we can start by optimally deleting the minimum number of flights together with the unused slots using the algorithm of Section 6.1 and then addressing $P(L)$ using the greedy procedure to be described in the next section. The correctness of this approach follows from the following Theorem.

Theorem 17. *Given a flight allocation graph, G , for any L , there exists a solution to $P(L)$ such that i) $\{f_i \in F : y_i = 0\} \subset F^*(G)$ and ii) $\{s_j \in S : \sum_i x_{ij} = 1\} \subset S^*(G)$.*

Proof. Let M^* be a delay minimizing maximum matching and $F^*(G)$ and $S^*(G)$ be defined as before. We start by proving ii). Suppose we have a solution, M^o , to $P(L)$ that covers a slot not in $S^*(G)$ and let s_i be the earliest such slot. Using the solution structure described in Section 4.1 and illustrated in Figure 2, we can see that s_i must be in some gap $S_{[\ell]}^-$; further it is clear that the flight covering that slot must be contained in $F_{[\ell']}^+$ for some $\ell' \leq \ell$. Now if we take the union of M^* and M^o , we obtain a union of cycles and alternating paths. s_i must be the end of an alternating path since it was not covered in M^* . That alternating path will either end in a slot node in a) $S_{[\ell']}^+$ or b) a flight node in $F_{[\ell']}^+$ not covered by M^* . In case a) we can improve M^o by performing an interchange that preserves $F(M^o)$ but exchanges s_i for a slot in $S_{[\ell']}^+$, which would lead to a lower delay solution – a contradiction to the optimality of M^o relative to $P(L)$. In case b) we could perform an interchange that would increase the cardinality of M^* contradicting that it was a maximum matching. This completes the proof of ii).

Given that ii) is true, in order to prove i), we take a solution to $P(L)$ and let $S' \subset S^*(G)$ be the corresponding set of covered slots. We know from our previous analysis that finding the best covering flight set is a matroid optimization problem so that this set, say F' is a max weight basis (flight set such that there is a matching between F' and S'). Thus, if one considers the transversal matroid associated with the original slot set, $S^*(G)$, then F' is a max weight basis in the transversal matroid associated with S' a subset of $S^*(G)$. It follows from basic properties of transversal matroids (see Edmonds and Fulkerson [6]) that F' can be extended to a max weight basis of the original transversal matroid. That is F' can be extended to an optimal flight set $F^*(G)$ (max weight basis) relative to the entire graph G^* . \square

6.2.1 A Fork Refinement

Before constructing a greedy procedure that solves $P(L)$, we first refine the FORK characterization discussed in Section 5.3. As in the earlier case, this construction assumes that a matching covering all flights exists (if not appropriate flights must be deleted first as described in the previous section). We start with some notation: let $I(j) := \{i \in I : j(i) = j\}$, and define $r(j) (j \in J)$ recursively as

$$r(1) := |I(1)|, \quad r(j+1) := (r(j) - 1)^+ + |I(j+1)|.$$

Intuitively, $r(j)$ may be interpreted as the number of flights available for slot j , i.e., those that can use the slot and have not been assigned a previous slot. Note that $r(j)$ is generally not monotone in j . In addition, we note that the definition of $r(j)$ may be used to provide an alternate characterization of forks, by changing the condition $j(i+k) \leq j(i) + k$ to $r(j(i) + k) > 0$.

To define *refined* forks, let $i \in I$ and let t_i be the largest nonnegative integer such that

1. $r(j(i) + t_i) = 1$, and
2. $r(j(i) + t) > 1$ for all $0 \leq t < t_i$.

The edge set $F(i)$ is defined as

$$F(i) := \{[f_i, s_{j(i)}], [f_{i+1}, s_{j(i)+1}], \dots, [f_{i+t_i}, s_{j(i)+t_i}]\},$$

and is called a *refined* fork. As before, we define $I(i) = \{i, i+1, \dots, i+t_i\}$, $J(i_k) = \{j(i), j(i)+1, \dots, j(i)+t_i\}$. Furthermore, we let F^* be the following union of forks

$$F^* = F(i_1) \cup F(i_2) \cup \dots \cup F(i_p),$$

where $i_1 = 1$, $i_k = |F(i_1)| + \dots + |F(i_{k-1})| + 1$ ($2 \leq k \leq p$) and $m \in I(i_p)$. Again, F^* is a maximum matching in G , and we call F^* the *refined* fork matching. Observe that any fork consists of a number of consecutive refined forks, for which $j(i_k) + t_{i_k} + 1 = j(i_{k+1})$. Intuitively, a refined fork may be viewed as a deletion opportunity (or, using Collaborative Decision-Making terminology, a bridging sequence; see Vossen and Ball [11]) since the deletion of flight f_{i_k} will allow all other flights in the refined fork to be moved up a slot.

To illustrate the benefits of this refinement, we now show the optimality of the refined fork matching by constructing a corresponding dual solution. The explicit construction of a dual will provide a solution for arbitrary values of L .

We can define the dual under consideration is as follows:

D(L):

$$\begin{aligned} z_D(L) = \max \quad & \sum_{i \in I} u_i - \sum_{j \in J} v_j - Lw \\ \text{s.t.} \quad & \\ u_i \quad & \leq T(s_j) + v_j \quad ([i, j] \in E : i \in I) \\ u_i \quad & \leq T(f_i) + w \quad (i \in I) \\ u_i, v_j, w \geq & 0. \end{aligned}$$

To determine an optimal solution for the dual problem $D(0)$ we first define for each refined fork $k \in \{1, \dots, p\}$

$$T^{\max}(k) = T(s_{j(i_k)+t_{i_k}}) \quad , \text{ and } \quad T^{\min}(k) = T(f_{i_k}).$$

Then, an optimal dual solution is

$$\begin{aligned} u_i &= T^{\max}(k) \quad , i \in I(i_k), k \in K; \\ v_j &= T^{\max}(k) - T(s_j) \quad , j \in J(i_k), k \in K \\ &= 0 \quad , \text{ otherwise;} \\ w &= \max_{k \in \{1, \dots, p\}} (T^{\max}(k) - T^{\min}(k)). \end{aligned}$$

It is easy to verify that the solution is dual feasible, and that $z_P(0) = z_D(0)$.

6.2.2 A Greedy Procedure for Flight Deletions

We now describe an algorithm for determining an allocation that solves $P(L)$ for arbitrary values of L . The procedure starts with an allocation in which no flights are deleted. Subsequently, the procedure repeatedly deletes a flight until L flights have been deleted. At each iteration, the procedure selects the first flight of a refined fork whose corresponding bridging sequence yields the maximum reduction in delay. The resulting procedure can be outlined as follows.

Greedy-Flight-Parametric:

step 0: Let G be a flight allocation graph. Let $l = 0$, and let the set of deleted flights be $C(l) = \emptyset$. Let $M(0)$ be the refined fork matching that solves $P(0)$.

while $l < L$ **do:**

step 1: For each refined fork $k \in \{1, \dots, p(l)\}$ in $M(l)$, let $T^{\max}(k) = T(s_{j(i_k)+t_{i_k}})$ and let $T^{\min}(k) = T(f_{i_k})$. Furthermore, let

$$\begin{aligned} w(l+1) &= \max_{k \in \{1, \dots, p(l)\}} (T^{\max}(k) - T^{\min}(k)), \\ k^* &= \arg \max_{k \in \{1, \dots, p(l)\}} (T^{\max}(k) - T^{\min}(k)). \end{aligned}$$

step 2: Cancel the first flight in refined fork k^* , and let

$$\begin{aligned} C(l+1) &:= C(l) \cup \{f_{i_{k^*}}\}, \\ M(l+1) &:= M(l) - F(i_{k^*}) \\ &\quad \cup \{[f_{i_{k^*}+1}, s_{j(i_{k^*})}], [f_{i_{k^*}+2}, s_{j(i_{k^*})+1}], \dots, [f_{i_{k^*}+t_{i_{k^*}}}, s_{j(i_{k^*})+t_{i_{k^*}}-1}]\}, \\ l &:= l+1. \end{aligned}$$

Observe that the Greedy-Flight-Parametric procedure requires $O(Lm)$ steps, since both steps 1 and step 2 may use $O(m)$ steps. The correctness of the procedure is shown in the following theorem.

Theorem 18. *The matching $M(L)$ obtained by the greedy flight deletion procedure yields an optimal solution to $P(L)$.*

Proof. The proof follows by construction of an optimal dual solution to $D(L)$. First, we let

$$\begin{aligned} w &:= w(L), \\ u_i &:= T(f_i) + w(L) \quad \text{for } i \in C(L), \\ v_j &:= 0 \quad \text{for } j \notin S(M(L)). \end{aligned}$$

To define a dual solution, we also need to define values for u_i ($i \in I - C(L)$) and v_j ($j \in S(M(L))$), corresponding to the flights and slots in each refined fork. We start by determining for each refined fork $k \in \{1, \dots, p(l)\}$ in $M(L)$,

$$T^S(k) = \max_{j \in J(i_k)} T(s_j), \quad T^C(k) = \max_{i \in C(L): j(i) \leq j(i_k)} u_i, \quad T^{\max}(k) = \max(T^S(k), T^C(k)),$$

and let

$$\begin{aligned} u_i &= T^{\max}(k) \quad , i \in I(i_k), 1 \leq k \leq p(L) \\ v_j &= T^{\max}(k) - T(s_j) \quad , j \in J(i_k), 1 \leq k \leq p(L). \end{aligned}$$

Observe that $T^{\max}(k) \leq T^{\max}(k')$ if $k < k'$.

To show that the resulting solution yields an optimal primal-dual pair, we first note that the complementary slackness conditions hold, that is,

$$y_i = 1 \quad \Rightarrow \quad u_i = T(f_i) + w,$$

$$x_{ij} = 1 \quad \Rightarrow \quad u_i = T(s_j) + v_j \Leftrightarrow T^{\max}(k) = T(s_j) + T^{\max}(k) - T(s_j),$$

where k is such that $i \in I(i_k)$ and $j \in J(i_k)$. Thus, what remains is to show that the resulting solution is dual feasible. Let us first consider the dual constraints

$$u_i \leq T(f_i) + w.$$

Clearly, these constraints are satisfied for $i \in C$. Thus, let $i \in I - C(L)$ with some some k such that $i \in I(i_k)$. Now, we can distinguish two cases:

1. $u_i = T^S(k)$. In this case, we have

$$T^S(k) \leq T(f_i) + w,$$

which holds since otherwise the deletion of f_i would have yielded a greater delay reduction than the flight that was deleted in iteration L .

2. $u_i = T^C(k)$. Let $T^C(k) = u_{i'}$ for some $i' \in C(L)$ such that $j(i) \leq j(i_k)$. In this case, we have

$$T^C(k) = u_{i'} = T(f_{i'}) + w \leq T(f_i) + w,$$

which holds since $T(f_{i'}) \leq T(f_i)$.

Next, we consider the dual constraints

$$u_i \leq T(s_j) + v_j \quad ([i, j] \in E).$$

To prove that these constraints are satisfied, we consider the following four cases:

1. $i \in C(L), j \in S(M(L))$ such that $j \geq j(i)$. Let k be such that $j \in J(i_k)$. In this case the constraint reduces to

$$T(f_i) + w \leq T^{\max}(k),$$

which holds by definition.

2. $i \in C(L), j \notin S(M(L))$ such that $j \geq j(i)$. In this case, the constraint reduces to

$$T(f_i) + w \leq T(s_j).$$

Suppose now that f_i was canceled at iteration $l (1 \leq l \leq L)$ and let $j(l)$ be the slot that was vacated. Clearly, the constraint holds if $j \geq j(l)$. Now suppose that $j < j(l)$. Let l' be the iteration in which slot s_j was vacated, and $i(l')$ be the corresponding flight that was canceled. Since $T(f_i) \leq T(f_{i(l')})$, we have

$$T(s_j) - T(f_i) \geq T(s_j) - T(f_{i(l')}) = w(l') \geq w(L) = w.$$

3. $i \in I - C(L), j \in S(M(L))$. Let k, k' be such that $i \in I(i_k)$ and $j \in J(i_{k'})$. Observe that $k \leq k'$. Substituting u_i and v_j then yields

$$T^{\max}(k) \leq T^{\max}(k'),$$

which holds since $T^{\max}(k)$ is ascending in k .

4. $i \in I - C(L), j \notin S(M(L))$ with some some k such that $i \in I(i_k)$. Now, we can again distinguish two cases:

(a) $u_i = T^S(k)$. In this case we have

$$T^S(k) \leq T(s_j),$$

which holds by definition.

(b) $u_i = T^C(k)$. Let $T^C(k) = u_{i'}$ for some $i' \in C(L)$ such that $j(i) \leq j(i_k)$. This case follows from case 2. □

It is relatively easy to see that the equity properties that apply to allocations that cover all flights (see Section 5) also carry over to the current situation. Similar to the approach taken in Section 4.2, we can organize the delays of any matching M of size $m - L$ into the delay vector $d^M = (d_1^M, d_2^M, \dots, d_{m-L}^M)$. The same notion weak majorization applies, and Theorem 12 can be extended as follows.

Theorem 19. *Let $M(L)$ be the allocation produced by procedure **Greedy-Flight-Parametric**. Then, $M(L)$ has the M^* properties given in Theorem 9 and Proposition 4.*

Observe that this result no longer applies if we consider the more general case in which the $j(i)$ are not weakly increasing. In this situation, however, one can easily transform the allocation $M(L)$ by repeatedly applying the pairwise interchanges introduced in Theorem 7 to the flights within a refined fork.

6.3 Flight Deletions under General $\hat{D}_\Delta(M)$

We now consider more general models where delay costs are associated with each deleted flight. In the first model we consider, there is an arbitrary cost Δ for each deleted flight. We show there is a correspondence with the parametric model discussed in Section 6.2. Specifically, we establish a mapping that associates with every value of Δ an L , such that the matching $M(L)$ found by the procedure **Greedy-Flight-Parametric** minimizes the total delay costs. To establish this result, we first state the optimization model for the cost minimization problem.

P(Δ):

$$z_P(\Delta) = \min \sum_{i \in I} (\Delta + T(f_i))y_i + \sum_{ij \in E: i \in I} T(s_j)x_{ij}$$

s.t. (1), (2), (4),

where, as before, $x_{ij} = 1$ iff flight f_i is assigned to slot s_j , and $y_i = 1$ iff flight f_i is canceled. Our main result is stated in the following theorem.

Theorem 20. *For any value $\Delta \geq 0$, let L be a non-negative integer such that*

$$w(L + 1) \leq \Delta \leq w(L),$$

where $w(l)$ corresponds to the delay reduction determined at each iteration of the greedy flight deletion procedure, and $w(0) := \infty$. Then, the matching $M(L)$ obtained by the greedy flight deletion procedure yields an optimal solution to $P(\Delta)$.

The proof follows by construction of a dual solution and is analogous to the proof of Theorem 18. Implicit in Theorem 20 is the property that for any non-negative integer L , there exists a Δ such that $M(L)$ is a solution to $P(\Delta)$. Another related property is that $w(L)$ is a non-increasing function of L . These are all special properties that are not true in general.

To conclude, we consider a more general model in which each deleted flight f_i may have a different cost Δ_i . We show that a variant of the procedure **Greedy-Flight-Parametric** yields a matching that minimizes the total delay costs. To establish this result, we first state the optimization model for this cost minimization problem.

P($\vec{\Delta}$):

$$\begin{aligned} z_P(\vec{\Delta}) = \min & \sum_{i \in I} (\Delta_i + T(f_i))y_i + \sum_{ij \in E: i \in I} T(s_j)x_{ij} \\ \text{s.t.} & (1), (2), (4). \end{aligned}$$

An algorithm that yields an allocation which solves $P(\vec{\Delta})$ is as follows. The procedure starts with an allocation in which no flights are deleted. Subsequently, the procedure repeatedly deletes the flight which yields the largest cost reduction until no such flights can be found. The resulting procedure can be outlined as follows.

Greedy-Flight-Delay:

step 0: Let G be a flight allocation graph. Let the set of deleted flights be $C = \emptyset$, and let M be the refined fork matching when no flights are deleted.

step 1: For each refined fork $k \in \{1, \dots, p\}$ in M , let $T^{\max}(k) = T(s_{j(i_k)+t_{i_k}})$. Furthermore, let

$$w = \max_{k \in \{1, \dots, p\}} \max_{i \in I(i_k)} (T^{\max}(k) - T(f_i) - \Delta_i),$$

and let k^*, i^* be such that $w = T^{\max}(k^*) - T(f_{i^*}) - \Delta_{i^*}$. Let j^* be such that $[f_{i^*}, s_{j^*}] \in M$.

Step 2: If $w \leq 0$, **stop**. Otherwise go to Step 3.

step 3: Cancel the flight f_{i^*} in refined fork k^* , and let

$$\begin{aligned} C & := C \cup \{f_{i^*}\}, \\ M & := M - F(i_{k^*}) \\ & \cup \{[f_{i_{k^*}}, s_{j(i_{k^*})}], \dots, [f_{i^*-1}, s_{j^*-1}]\} \\ & \cup [f_{i^*+1}, s_{j^*}], \dots, [f_{i_{k^*}+t_{i_{k^*}}}, s_{j(i_{k^*})+t_{i_{k^*}}-1}], \end{aligned}$$

and repeat Step 1.

We note that the Greedy-Flight-Parametric procedure requires $O(m^2)$ steps, since the number of flights deleted is $O(m)$. The correctness of the procedure is shown in the following theorem.

Theorem 21. *The matching M obtained when the greedy flight deletion procedure terminates yields an optimal solution to $P(\vec{\Delta})$.*

Proof. The proof again follows by construction of an optimal dual solution to $D(L)$, and is analogous to the proof of Theorem 18 after we define

$$\begin{aligned} u_i & := T(f_i) + \Delta_i \quad \text{for } i \in C, \\ v_j & := 0 \quad \text{for } j \notin S(M). \end{aligned}$$

□

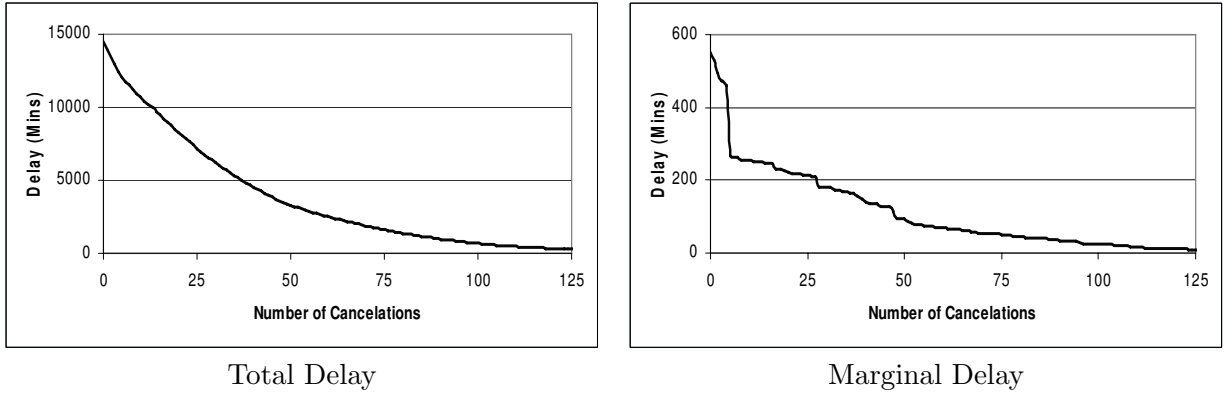


Figure 6: O'Hare Airport, Chicago.

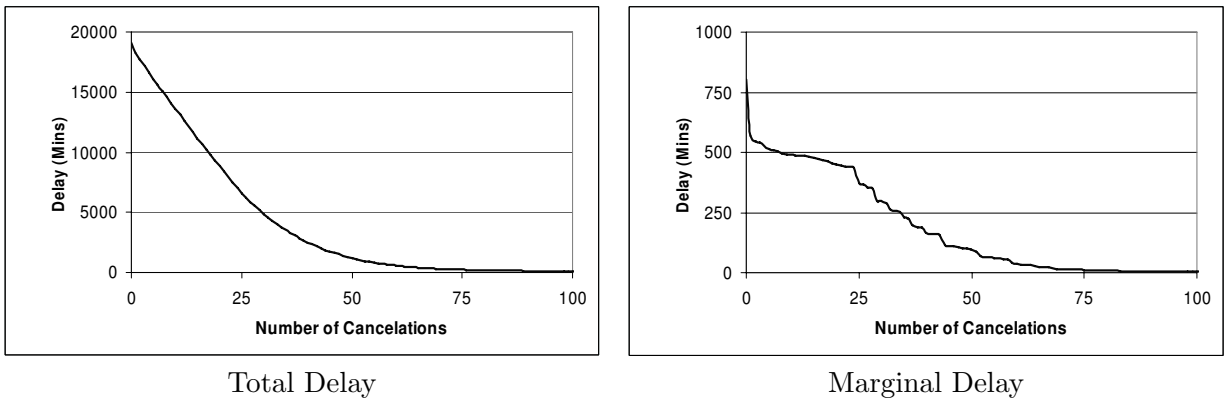


Figure 7: Logan Airport, Boston.

6.4 Empirical Results

Of course, each of the parametric problems $P(L)$, $P(\Delta)$, and $P(\vec{\Delta})$ can be solved as an assignment or transportation problem. The greedy iterative approach taken by Theorem 18, however, provides visibility into the structure of the solution and would readily allow a modeler or decision maker to adjust the solution or carry out a parametric analysis.

To illustrate this, we consider two case studies that are based on actual GDPs at Chicago O'Hare Airport (ORD) and Boston's Logan Airport (BOS). In the case of Chicago O'Hare Airport, we used data from a GDP that was implemented on January 26, 2001. This data set included 619 flights, while the number of slots in the initial 5-hour program period was 376. It is important to note, however, that the RBS algorithm creates extra slots past the program period to ensure that all flights are accommodated. As a result, the resulting flight allocation graph has 1238 nodes (619 each for flights and slots), while the earliest arrival times in our data set resulted in 262,636 edges. For Boston's Logan Airport, we used data from a GDP that was implemented on January 9, 2001, with 281 flights and 208 slots in the initial program period. In this case, the resulting flight allocation graph has 562 nodes and 71,645 edges.

For both of these cases, we consider the solutions found by the procedure **Greedy-Flight-Parametric** over a range of the number of flights cancellations L . The results are shown in Figure 6 and Figure 7. The graphs on the left in each figure illustrate how total delay decreases as a function of L . The graphs on the right show the marginal delay reduction for each value of L , and correspond to $w(L)$ as defined in the proof of Theorem 18. The RBS algorithm corresponds to the greedy procedure when L equals 0; since RBS does not allow flight cancellations, a direct comparison is

more difficult when $L > 0$. If cancellations are a possibility (i.e., in airspace flow programs), however, the graphs shown above would allow the service provider to evaluate the impact of different levels of flight cancellations. In addition, we note that the marginal delay graphs allow us to determine the optimal solutions to $P(\Delta)$. Consider, for example, the marginal delay graph in the case of Chicago O'Hare Airport. Since the marginal delay reduction equals 551 minutes when $L = 1$ (that is, $w(L) = 551$), Theorem 20 states that no flights will be canceled when $\Delta > 551$. Thus, the marginal delay graphs allow us to investigate the "implicit" delay cost associated with canceling flights.

7 Conclusions

In this paper we have investigated a class of models related to assigning flights to slots so as to achieve delay-related objectives. We have given simple, greedy algorithms for many of these models. These models produce solutions that not only minimize total delay objectives but that also satisfy strong properties related to the theory of majorization. These properties indicate that the solutions are highly desirable with respect to the equity of the distribution of delay among the flights. For this application equity is a second objective of importance that can be difficult to quantify using simple linear objective functions.

We investigate problems involving flight deletions and show useful structural properties for the solutions. In addition using simple, greedy algorithms we are able to produce parametric solutions to problems for varying values of the number of flights deleted or the deletion cost.

8 Acknowledgements

The work of the first author was supported by the National Center of Excellence for Aviation Operations Research (NEXTOR) under FAA Research Grant 96-C-001 and Contract DFTA03-97-D00004. Any opinions expressed herein do not necessarily reflect those of the FAA or the U.S. Department of Transportation.

References

- [1] A.S. Asratian, T.M.J. Denley and R. Haggkvist. 1998. *Bipartite graphs and their Applications*. Cambridge Univ. Press. Cambridge.
- [2] M. Ball, R. Dahl, L. Stone and T. Thompson. 1993. OPTIFLOW Build-I Design document, Version 1.5, *Optiflow Project Report to the FAA*, 1993.
- [3] Ball, M., C. Barnhart, G. Nemhauser, A. Odoni. 2006. Managing Air Traffic and Airline Operations for Schedule Reliability. *Handbook of Operations Research and Management Science: Transportation*, C. Barnhart and G. Laporte (eds), in press.
- [4] R.A. Brualdi. 1975. Transversal Theory and Graphs. *Studies in Graph Theory, Part I*, Fulkerson, D.R. (ed), The Mathematical Association of America, USA.
- [5] J. Edmonds. 1971. Matroids and the greedy algorithm. *Mathematical Programming* 1, 127-136.
- [6] J. Edmonds and D.R. Fulkerson. 1965. Transversals and matroid partition *J. Res. Nat. Bureau of Standards* 69B, 147-153.

- [7] F. Glover. 1967. Maximum matchings in a convex bipartite graph. *Naval Research Logistics* 14, 313–316.
- [8] G.H. Hardy, J.E. Littlewood and G. Polya. 1988. *Inequalities*. Cambridge University Press. Cambridge Mathematical Library. 2nd edition.
- [9] A.W. Marshall and I. Olkin. 1979. *Inequalities: Theory of Majorization and Its Applications*. Academic Press, New York.
- [10] G. Steiner and J.S. Yeomans. 1996. A linear time algorithm for maximum matchings in convex, bipartite graphs. *Comput. Math. Appl.* 31, No.12, 91-96.
- [11] T. Vossen and M. Ball. 2006. Optimization and mediated bartering models for ground delay programs. *Naval Research Logistics*, 53, 75–90.
- [12] T. Vossen and M. Ball. 2006. Slot trading opportunities in collaborative ground delay programs. *Transportation Science*, 40, 29–43.
- [13] M. Wambsganss. 1996. Collaborative decision making through dynamic information transfer *Air Traffic Control Quarterly*, 4, 107-123.
- [14] D.J.A. Welsh. 1976. *Matroid Theory* Academic Press/Harcourt Brace Jovanovich, London/New York.
- [15] H.P. Young. 1994. *Equity in Theory and Practice*. Princeton University Press, Princeton, NJ.