경제학석사학위논문

The truncation algorithm in the reassignment of landing slots

항공기운항시각 재조정을 위한 절단 알고리즘

2014년 2월

서울대학교 대학원

경제학부 경제학 전공

최 인 혁

The truncation algorithm in the reassignment of landing slots

지도교수 전 영 섭

이 논문을 경제학석사 학위논문으로 제출함.

2013년 10월

서울대학교 대학원

경제학부 경제학 전공

최 인 혁

최인혁의 경제학석사 학위논문을 인준함.

2013년 12월

위 원 장 김 진 우 (인)

부위원장 전 영 섭 (인)

위 원 주 병 기 (인)

Abstract

When an airport's preexisting landing schedule becomes inefficient mainly due to inclement weather, the Federal Aviation Administration (FAA) in the United States aims to create a new queue that does not waste airport landing slots, considering airlines' incentives. Although the incentive to report delays is satisfied by the compression algorithm currently used by the FAA, it fails to give airlines the incentive to report cancellations. This paper gives an alternative mechanism, called the truncation algorithm, that satisfies the two incentive conditions above. Further, we show that the truncation algorithm satisfies desirable properties such as non-wastefulness and individual rationality.

Keywords: Incentive, Algorithm, Airport
Student Number: 2011-23175

Contents

List of Figures

1 Introduction

When an airport's preexisting landing schedule becomes inefficient mainly due to inclement weather, the Federal Aviation Administration (FAA) in the United States aims to create a new queue that does not waste airport landing slots,[1] considering airlines' incentives. To be specific, when the number of utilizable runways or visibility is reduced by inclement weather, the FAA initiates a ground delay program, usually hours in advance, which usually proceeds in two steps. First, the number of landing slots at the affected airport is reduced. Second, considering further changes after the first step, the FAA performs the reassignment step by exploiting an algorithm. This paper focuses only on this reassignment step.[2]

Implementing the second step, the FAA should respect incentives in order to efficiently and fairly reassign airport landing slots among flights. Schummer and Vohra (2013) formalize two incentive conditions: the incentive to report delays, and the incentive to report cancellations. They show that the former is satisfied both by the compression algorithm (which has been used by the FAA since 1998) and by their proposed alternative (the trade-cycle algorithm), but that the latter is satisfied by neither of these two algorithms. Furthermore, they introduce the concept of the core in this environment, and demonstrate that the trade-cycle algorithm always produces an outcome in the core while the compression algorithm does not. However, it should be noted that, in practice, all resources (slots) basically belong to the planner (the FAA), not the airlines. Therefore, it is not natural to define and consider the concept of the core which is usually discussed in the literature. Instead, it would be more natural to weaken this condition to individual rationality.

From this perspective, the objective of the paper is to design an alternative mechanism that satisfies the two incentive conditions above, maintaining the desirable properties of the compression algorithm such as non-wastefulness and individual rationality. We call this algorithm the truncation algorithm because it "truncates" the waiting time, taking into account an initial slot ownership and landing schedule. To understand our algorithm more extensively, we compare the algorithm with others. For instance, the truncation algorithm does not always produce an outcome in the core while the trade-cycle algorithm does, as mentioned above.

More intriguingly, the truncation algorithm is very similar to the deferred acceptance algorithm introduced by Schummer and Abizada (2013), who actually ex-

[1] Landing slots (or airport slots) are rights assigned to an entity (airline) by government agency (mainly, an airport), who allows the slot owner (airline) to have the right to schedule a take-off or landing during a specific time period. Landing slots are typically allocated according to guidelines set down by the International Air Transport Association (IATA), which is available at the IATA's website: http://www.iata.org/policy/slots/Pages/slot-guidelines.aspx or (for download) http://www.iata.org/policy/slots/Documents/wsg-5.pdf.

[2] For details on a ground delay program, I refer the reader to Ball, Hoffman, and Vossen (2002) and Schummer and Vohra (2013).

tend the well-known deferred acceptance algorithm of Gale and Shapley (1962). In fact, given exogenously a fixed profile of priority orders of slots, the deferred acceptance algorithm also achieves the above two incentive conditions, satisfying non-wastefulness and individual rationality. However, since a priority order is exogenously determined, the outcome of the algorithm depends on the names of the airlines. In other words, their algorithm is not anonymous whereas the truncation algorithm is anonymous.

This paper is placed between two bodies of literature: the game theoretic literature on matching, and the operations-oriented literature. First, our model can be seen as a restricted form of the college admissions model of Gale and Shapley (1962) or a generalization of the housing market model of Shapley and Scarf (1974). Since in our model landing slots do not have preferences but airlines do, our model complements the school choice model of Abdulkadiroğlu and Sönmez (2003) where students (landing slots) have preferences whereas colleges (airlines) do not. Other related literature is that of operations and transportation. Vossen and Ball (2006a) suggest a linear programming approach in order to minimize total airline costs. Vossen and Ball (2006b) regard the compression algorithm as a process of barter exchange. Note that in the operations literature the concept of incentives is not formalized as rigorously as in the matching literature. Instead, the literature considers a range of optimization problems. See Wambsganss (1997) for a historical perspective.

The paper is organized as follows. Section 2 is a preliminary section defining landing schedules and algorithms. In Section 3, we introduce the truncation algorithm, and explain how it works through some examples. In Section 4, we discuss properties of the truncation algorithm, while Section 5 compares the algorithm with others. Finally, Section 6 concludes with some remarks.

2 Preliminaries

Let $\mathcal{A}$ be a finite set of airlines, whose elements are typically denoted by $A, B \in \mathcal{A}$. Each airline $A \in \mathcal{A}$ has its own finite set of flights F_A. We regard F_A as the set of airline A's flights that have not been canceled so far. We denote by $F = \cup_{A \in \mathcal{A}} F_A$ the set of all airlines' flights.

There is a finite ordered set of slots $S = \{1, 2, 3, ..., |S|\}$, whose generic elements are denoted by $s, t \in S$. Since the slot labels have ordinal meanings, $s < t$ for $s, t \in S$ implies that slot s is earlier than slot t. Note that in our model the number of all slots considered always outnumbers the number of all flights considered (i.e., $|F| < |S|$), and that each flight requires the use of only one slot that cannot be shared.

We denote by $e_f \in S$ the earliest feasible arrival time of flight $f \in F$. Let $\mathbf{e}$ be the list of earliest feasible arrival times of all flights in F. Clearly, each flight $f \in F$ must be assigned to a slot no earlier than its earliest feasible arrival time e_f. In other

words, it is not feasible for flight $f \in F$ to be assigned to slot $j \in S$ with $j < e_f$.

A *landing schedule* is a function $\Lambda : F \to S$ mapping flights to slots such that (i) for any flight $f \in F$, $\Lambda(f) \geq e_f$ (i.e., each flight is assigned to one of its feasible slots), and (ii) for any two flights $f, f' \in F$, $f \neq f'$ implies $\Lambda(f) \neq \Lambda(f')$ (i.e., distinct flights are assigned to distinct slots).

Although a landing schedule implicitly indicates which airlines have ownership of the slots occupied by flights, it does not specify the ownership of unoccupied (vacant) slots. Thus, we introduce the concept of a slot ownership that completely describes which airlines holds which slots. A *slot ownership* is a function $\Omega : S \to \mathcal{A}$ which should satisfy the following condition: for any flight $f \in F$, if $f \in F_A$ for any airline $A \in \mathcal{A}$, then $\Omega(\Lambda(f)) = A$.

An *assignment* is a pair of functions (Λ, Ω). We interpret (Λ^0, Ω^0) as an *initial* assignment which could be inefficient owing to the location of unoccupied slots. A *problem* to reassign airport landing slots is summarized by $P = (S, \mathcal{A}, (F_A)_{\mathcal{A}}, \mathbf{e}, \Lambda^0, \Omega^0)$. An *algorithm* φ takes a problem as input and generates a (new) landing schedule as output (i.e., $\varphi(P) = \Lambda$). In what follows, we assume that each airline has the right to swap its own flights within its own slots.

We now turn our attention to airlines' preferences. In our model, we assume that an airline is made better off only if it moves at least one of its flights up in a (new) landing schedule while it moves no others down. Formally, given two landing schedules Λ and Λ', we say that airline $A \in \mathcal{A}$ *strictly prefers* Λ to Λ' if (i) for any flight $f \in F_A$, $\Lambda(f) \leq \Lambda'(f)$, and (ii) for at least one flight $f \in F_A$, $\Lambda(f) < \Lambda'(f)$. If a landing schedule Λ' adjusts Λ by simultaneously moving some of airline A's flights to earlier slots but moving others later, we say that Λ' is *preference-incomparable* with Λ.

Finally, given the preference domain, we say that an algorithm is *manipulable* if an airline, by misreporting the feasible arrival times of its flights, can achieve a (new) landing schedule that it strictly prefers to the one achievable by honestly reporting feasible arrival times.

3 The truncation algorithm

In this section we introduce the truncation algorithm, denoted by φ^T, and show that (through example) it cannot be manipulated by delaying the cancellation announcement.[3] For a description of the truncation algorithm, see Figure 1.

The key idea behind the truncation algorithm is to shorten the waiting time of each flight (measured by $\Lambda(f) - s$ in Steps 2 and 3) while respecting an initial assignment (λ^0, Ω^0) (reflected in Steps 2 and 3 where a slot allocation highly depends on which airline initially has the given slot and to which slot each flight is initially

[3]The next section provides the formal proof; see Theorem 2.

Step 0	Initialize the current assignment as $\Lambda = \Lambda^0$ and $\Omega = \Omega^0$.
Step 1	If $S = \emptyset$, end the algorithm at the assignment (Λ, Ω). Otherwise pick the earliest slot $s \in S$ and declare it active.
Step 2	Let A denote the airline that owns s. Check whether airline A has a flight $f \in F_A \subset F$ such that $e_f \leq s$. If so, denote by F'_A the set of such flights, let $f = \text{argmin}_{f \in F'_A}\{\Lambda(f) - s\}$, denote by t its slot $\Lambda(f)$, and go to Step 4. Otherwise go to Step 3.
Step 3	Check whether *any* airline has a flight $f \in F$ such that $e_f \leq s$. If so, denote by F''_A the set of such flights, let $f = \text{argmin}_{f \in F''_A}\{\Lambda(f) - s\}$, denote by t its slot $\Lambda(f)$, and go to Step 4. Otherwise remove slot s from S, and return to Step 1.
Step 4	Move flight f from slot t to slot s: set $\Lambda(f) = s$ and set $\Omega(s)$ equal to the airline of flight f. Remove s, f from S, F, respectively, and return to Step 1.

Figure 1: The truncation algorithm

assigned). In other words, the algorithm "truncates" the waiting time, taking into account an initial slot ownership and landing schedule.

One of the notable distinctions from the compression algorithm[4] currently used by the FAA is that the truncation algorithm does not trade a slot that an airline owns but cannot use to another airline in exchange for some other slot. In addition, new assignments according to the truncation algorithm are determined in order from the earliest slot to the latest, and do not change once determined in previous rounds. The following example shows how the algorithm proceeds step by step.

EXAMPLE 1: *The initial assignment* (Λ^0, Ω^0) *is described by the table below. The first and second columns in the table list all slots and the airline who initially owns each slot. For example, airline* B *initially has slots 1, 3, and 7 (i.e.,* $\Omega(1) = \Omega(3) = \Omega(7) = B$*). The third column contains information as to which slot each flight is initially assigned, which airline owns each flight, and those flights' earliest feasible arrival times. For instance, airline* A *has flight* f_4 *whose earliest feasible arrival time is 2 but is initially assigned to slot 4 (i.e.,* $f_4 \in F_A$*,* $e_{f_4} = 2$*, and* $\Lambda^0(f_4) = 4$*).*

[4]See Schummer and Vohra (2013) for a description of the compression algorithm.

Slot	Airline	A f_4	B f_7	C f_5	C f_6
1	B				
2	C				
3	B				
4	A				
5	C				
6	C				
7	B				

In Step 1 of the truncation algorithm, slot 1 is first declared active. Since slot 1 initially belongs to airline B but flight f_7 (the only flight airline B has) cannot be assigned to it, flight f_5 is assigned to slot 1 in Step 3 of the algorithm. Similarly, flight f_4 is assigned to slot 2, and slot 3 remains vacant. This process updates the original assignment to the following table, where the last two columns show updated information.

Slot	Airline	A f_4	B f_7	C f_5	C f_6	(Λ, Ω) Flight	(Λ, Ω) Airline
1	B					f_5	C
2	C					f_4	A
3	B					Vacant	B
4	A						
5	C						
6	C						
7	B						

Next, since airline A owns slot 4 but has no remaining flights to be assigned, flights f_6 and f_7 become candidates for slot 4. However, due to the fact that $\Lambda(f_6) - 4 < \Lambda(f_7) - 4$, the algorithm allocates slot 4 to flight f_6. Finally, f_7 is assigned to slot 5, and slots 6 and 7 remain vacant. Thus, the algorithm is completed with the following assignment. Note that the outcome below coincides with the outcome that the compression algorithm produces in this example.

		A	B	C		(Λ, Ω)	
Slot	Airline	f_4	f_7	f_5	f_6	Flight	Airline
1	B					f_5	C
2	C					f_4	A
3	B					Vacant	B
4	A					f_6	C
5	C					f_7	B
6	C					Vacant	C
7	B					Vacant	B

Now we consider one type of manipulation: the destruction of vacant slots. Suppose that an airline has flight f initially assigned to slot s, but that the airline has to cancel flight f. Then we can ask whether the airline has an incentive to report the cancellation or not. Clearly, the airline could make the slot useless by delaying the report sufficiently long, which probably leads to inefficiency. Thus, one should check whether an algorithm to reassign slots is vulnerable to this kind of manipulation. The following example illustrates that the compression algorithm is vulnerable to the destruction of slots, but the truncation algorithm is not. In fact, the truncation algorithm cannot be manipulated by delaying the cancellation announcement sufficiently long, as shown in the next section.

EXAMPLE 1: *(revisited) Suppose that airline B destroys slot 1 by delaying the cancellation announcement. Then the initial assignment and the outcome of the truncation algorithm can be summarized by the following table.*

		A	B	C		(Λ, Ω)	
Slot	Airline	f_4	f_7	f_5	f_6	Flight	Airline
2	C					f_5	C
3	B					f_4	A
4	A					f_6	C
5	C					f_7	B
6	C					Vacant	C
7	B					Vacant	B

Note that airline B cannot be better off despite the destruction of slot 1 since its flight is still assigned to slot 5. In fact, no airline can be better off by destroying one of its own slots under the truncation algorithm. However, under the compression

algorithm, airline B can make its flight f_7 occupy slot 4 by delaying the cancellation announcement, which demonstrates the fact that the compression algorithm is vulnerable to the destruction of slots.

4 Properties of the truncation algorithm

If an algorithm to reassign landing slots has more desirable properties than others, it would be reasonable to pick and adopt this algorithm instead of others. In our model, desirable properties can be classified into three categories: efficiency, stability, and incentive compatibility.

We first focus on the efficiency condition. Intuitively, we do not want to waste slots when reassigning them among airlines. This notion can be formalized by requiring an algorithm to produce a new landing schedule in which there is no flight and no slot such that both (i) a flight can feasibly move up to a slot, and (ii) the slot is vacant.

Definition An algorithm φ is *non-wasteful* if, for any problem $P = (S, \mathcal{A}, (F_A)_{\mathcal{A}}, \mathbf{e}, \Lambda^0, \Omega^0)$, there exists no flight $f \in F$ and no slot $s \in S$ such that both (i) $e_f \leq s < \varphi_f(P)$, and (ii) $\varphi^{-1}(s) = \emptyset$.

The stability condition is also significant. Before defining this concept formally, however, it should be noted that in practice all slots basically belong to the planner (the FAA), not the airlines, although we have already introduced a slot ownership as if each airline really owns its slots. Thus, it is not natural to define and consider the concept of the core which is usually discussed in the literature. Instead, we weaken the stability condition to individual rationality.

Definition An algorithm φ is *individually rational* if, for any problem $P = (S, \mathcal{A}, (F_A)_{\mathcal{A}}, \mathbf{e}, \Lambda^0, \Omega^0)$, no airline strictly prefers Λ^0 to $\varphi(P)$.

The proposition below shows that the truncation algorithm satisfies the (weak) efficiency and (weak) stability conditions, as do most other reasonable algorithms.

Proposition 1 *The truncation algorithm φ^T is non-wasteful and individually rational.*

Proof. Suppose that there exist flight $f \in F$ and slot $s \in S$ such that both $e_f \leq s < \varphi^T_f(P)$ and $(\varphi^T)^{-1}(s) = \emptyset$, given problem P. For slot s to be vacant according to the algorithm, it should be removed from S in Step 3. This clearly implies that there is no flight $f \in F$ such that $e_f \leq s$, leading to a contradiction.

It is straightforward to prove the remaining part of the proposition because the

algorithm regards $\Lambda^0(f)$ for each flight $f \in F$ as a *lower bound* that flight f could be assigned to. In other words, each flight f is never assigned to slot s later than $\Lambda^0(f)$ in the outcome of the algorithm. □

The following two theorems are the main result of the paper, regarding incentive compatibility of the truncation algorithm. The first theorem states that, under the truncation algorithm, an airline cannot gain by misreporting the feasible arrival times of its flights. Note that the compression algorithm also cannot be manipulated in this way, as demonstrated in Schummer and Vohra (2013).

Theorem 1 *The truncation algorithm φ^T cannot be manipulated by misreporting the feasible arrival times of its flights.*

Proof. This proof is very similar to the proof for Theorem 1 in Schummer and Vohra (2013). First we provide detailed proof for the case of misreporting a single flight's arrival time, after which we give a brief general argument for the case of misreporting multiple flights.

Suppose that flight f is assigned to slot s by honestly reporting e_f. Obviously $e_f \leq s$, and thus, misreport e'_f should belong to one of the following three cases:

(i) If $e'_f < e_f \leq s$, then the only way that this misreport can change the outcome of the algorithm is for flight f to be assigned to a slot earlier than e_f, which clearly makes the airline worse off.

(ii) If $e_f < e'_f \leq s$, then, when e'_f is reported, the outcome of the algorithm cannot make the airline strictly better off, compared to when e_f is reported. This is because, when e'_f is reported, flight f would miss the chance to move into an earlier slot.

(iii) If $e_f \leq s < e'_f$, then flight f would have to end up in a slot strictly later than s, since the algorithm never places a flight in a slot earlier than its reported earliest arrival time. This obviously makes its airline worse off.

Therefore, misreporting a single flight's time either puts the flight in a later slot, or does not affect the outcome. In either case, flight f's airline cannot reap any benefit from this misreporting.

More generally, suppose that airline A's flights actually have the earliest feasible arrival times $(e_f)_{f \in F_A}$, but the airline misreports them to be $(e'_f)_{f \in F_A}$ with $e'_f \neq e_f$ for at least one $f \in F_A$. Then consider the first round of the algorithm where operation would differ under the report of $(e'_f)_{f \in F_A}$ versus $(e_f)_{f \in F_A}$. Note that the difference must occur in Step 2 or Step 3, and that, thereby, some flight $g \in F_A$ would be placed into a later slot according to the previous argument regarding the

single case. Thus, regardless of the outcome of the other flights, airline A cannot benefit from this misreporting. □

The second result shows that the truncation algorithm gives airlines an incentive to honestly report their flights' cancellations. This is significant in that the compression algorithm currently used by the FAA fails to give the incentive, as illustrated in the previous example.

Theorem 2 *The truncation algorithm φ^T cannot be manipulated by delaying the cancellation announcement.*

Proof. First we note that delaying the cancellation announcement sufficiently long can be interpreted as assigning a dummy flight into an unusable vacant slot in an initial assignment. More formally, let f_D be a dummy flight of airline A, and s be an unusable vacant slot that belongs to A. Since $e_{f_D} = s$, it always holds that $\varphi^T_{f_D}(P) = s$. Further, for all the slots earlier than s, the outcome of the algorithm does not change even if airline A delays the cancellation announcement.

For the other slots later than s, the outcome of the algorithm could be altered. However, airline A who delayed the cancellation announcement cannot gain from this outcome because slot ownership is not affected by delaying the cancellation announcement, and $\Lambda(f) - s$ never changes for all $f \in F$ in Step 2 or Step 3. Rather, airline A could be worse off since one of other flights could lose the chance to otherwise use slot s. □

5 Comparison with other algorithms

There exist many other algorithms which could be used to reassign airport landing slots. Although we discussed the properties of the truncation algorithm in the previous section, we can understand the algorithm more extensively by comparing it with others. Thus, in this section, we briefly introduce two algorithms, and then compare them to the truncation algorithm.

We first examine the trade-cycle algorithm[5] introduced by Schummer and Vohra (2013). They introduce this algorithm to guarantee a form of property rights, which is formalized by the concept of the core. A landing schedule Λ is a *core schedule* with respect to an initial assignment (Λ^0, Ω^0) if there is no other landing schedule Λ' and set of airlines $\mathcal{B} \subset \mathcal{A}$ such that (i) for all $f \in \cup_{A \in \mathcal{B}} F_A$, $\Omega^0(\Lambda'(f)) \in \mathcal{B}$, and (ii) each airline $A \in \mathcal{B}$ strictly prefers Λ' to Λ. As they demonstrate, the trade-cycle algorithm satisfies this condition. However, the truncation algorithm can produce a

[5]See Schummer and Vohra (2013) for a description of the trade-cycle algorithm. Note that the trade-cycle algorithm has the flavor of the top-trading cycle algorithm of Shapley and Scarf (1974).

landing schedule that is not in the core, as illustrated in the following example borrowed from Schummer and Vohra (2013).

EXAMPLE 2: *Consider the following initial assignment and the outcome of the truncation algorithm in the table.*

		A	B	C	(Λ, Ω)	
Slot	Airline	f_5	f_4	f_3	Flight	Airline
1	A				f_3	C
2	B				f_4	B
3	C				f_5	A
4	B				Vacant	B
5	A				Vacant	A

Note that the outcome of the truncation algorithm is not in the core because airlines A and B, by using only their own slots among themselves, can achieve the schedule in which they are each strictly better off (i.e., flights f_4 and f_5 can be assigned to slots 1 and 2, respectively).

We now concentrate on the deferred acceptance (DA) algorithm introduced by Schummer and Abizada (2013). In fact, they provide the DA with self-optimization[6] algorithm. However, whether any landing schedule is self-optimized or not is irrelevant when speaking of incentives in our model. Therefore, for a comparison with the truncation algorithm, we consider the DA (*without* self-optimization) algorithm described in Figure 2.[7] Note that, compared with the DA with self-optimization algorithm, the DA algorithm has only one adjustment: the omission of a self-optimization step.

Although the DA algorithm looks very different from the truncation algorithm, they are the same under certain circumstances. To be specific, if priority orders are determined *endogenously* by some function of parameters in the model, two algorithms could produce the same outcome. We give an example illustrating the relation between them.

EXAMPLE 1: *(revisited) Consider the following initial assignment with the profile of priority orders. The last two columns show the outcome of the truncation algorithm.*

[6]For the notion of self-optimization, see Schummer and Abizada (2013).

[7]For the DA algorithm's description, we need the following definitions. Let $\mathcal{P}(\mathcal{A})$ denote the domain of problems in which $\mathcal{A}$ is the existing set of airlines. For any positive integer $s \in \{1, 2, ..., |S|\}$, a *priority order* (on $\mathcal{A}$), $\succ_s$, is simply a linear order over the airlines $\mathcal{A}$. For $P \in \mathcal{P}(\mathcal{A})$, $(\succ_s)_{s \in S}$ is a *profile of priority orders*. Note that a priority order is *exogenously* given in the DA algorithm, as shown in Figure 2.

For a set of airlines $\mathcal{A}$ and a profile of priorities $(\succ_s)_{s\in S}$ on $\mathcal{A}$, the deferred acceptance algorithm with respect to $(\succ_s)_{s\in S}$ associates with each $P \in \mathcal{P}(\mathcal{A})$ the landing schedule computed with the following steps:

Step 0 For each $s \in S$, let $\succ'_s$ be the priority order over $\mathcal{A}$ that satisfies (i) if $\Omega^0(s) = A$, then A is ranked first in $\succ'_s$, and (ii) for any two airlines B and C such that $\Omega^0(s) \neq B$ and $\Omega^0(s) \neq C$, $B \succ'_s C$ is equivalent to $B \succ_s C$.

Step 1 Assign slot 1 to some flight $f \in F_A$ such that A is the highest-ranked airline in $\succ'_1$ that can feasibly use slot 1. Remove f from F. If no such flight exists, slot 1 permanently remains vacant.

Step 2 Assign slot 2 to some flight $g \in F_B$ such that, *subject to the removal of* f, B is the highest-ranked airline in $\succ'_2$ that can feasibly use slot 2. Remove g from F. If no such flight exists, slot 2 remains vacant.

Step k Continue such assigning to each slot $k = 3, 4, \cdots$, until all flights are allocated.

Figure 2: The deferred acceptance algorithm

			A	B	C		(Λ, Ω)	
Slot	Airline	Priority order	f_4	f_7	f_5	f_6	Flight	Airline
1	B	$A \succ_1 B \succ_1 C$					f_5	C
2	C	$B \succ_2 C \succ_2 A$					f_4	A
3	B	$B \succ_3 A \succ_3 C$					Vacant	B
4	A	$C \succ_4 A \succ_4 B$					f_6	C
5	C	$B \succ_5 C \succ_5 A$					f_7	B
6	C	$C \succ_6 A \succ_6 B$					Vacant	C
7	B	$A \succ_7 B \succ_7 C$					Vacant	B

Note that the DA algorithm, in this example, produces the same schedule. However, if priority order $\succ_4$ changes into $A \succ_4 B \succ_4 C$, then the DA algorithm will produce a different schedule: flights f_6 and f_7 are assigned to slots 5 and 4, respectively.

Meanwhile, we rename airlines A to B, B to C, and C to A. The following table summarizes the result of this permutation. It can be easily checked that the landing schedule does not change according to the truncation algorithm while it does change according to the DA algorithm.

			A		B	C	(Λ, Ω)	
Slot	Airline	Priority order	f_5	f_6	f_4	f_7	Flight	Airline
1	C	$A \succ_1 B \succ_1 C$					f_5	A
2	A	$B \succ_2 C \succ_2 A$					f_4	B
3	C	$B \succ_3 A \succ_3 C$					Vacant	C
4	B	$C \succ_4 A \succ_4 B$					f_6	A
5	A	$B \succ_5 C \succ_5 A$					f_7	C
6	A	$C \succ_6 A \succ_6 B$					Vacant	A
7	C	$A \succ_7 B \succ_7 C$					Vacant	C

The last example above motivates us to consider the anonymity property. Let π be a bijection of $\mathcal{A}$ into itself denoting permutations of $\mathcal{A}$. Given any permutation π, let $F_{\pi(A)}$ denote a set of flights that airline $\pi(A) \in \mathcal{A}$ has, and $\pi\Omega(s)$ denote the airline that each $s \in S$ belongs to under the permutation. In addition, we define πP as follows: for any permutation π, $\pi P := (S, \mathcal{A}, (F_{\pi(A)})_{\mathcal{A}}, \mathbf{e}, \Lambda^0, \pi\Omega^0)$. Note that πP does not affect a profile of priority orders $(\succ_s)_{s \in S}$, which plays a critical role in differentiating the truncation algorithm from the DA algorithm.

Definition An algorithm φ is *anonymous* if, for any problem $P = (S, \mathcal{A}, (F_A)_{\mathcal{A}}, \mathbf{e}, \Lambda^0, \Omega^0)$ and any permutation π, it holds that $\varphi(P) = \varphi(\pi P)$.

Anonymity requires that an algorithm should not depend on the names of the airlines. Admittedly, this condition might not be significant in practice. However, by requiring anonymity, we can demand that a priority order should be determined endogenously, not exogenously, and thereby we can differentiate the truncation algorithm from the DA algorithm.

Proposition 2 *The truncation algorithm φ^T is anonymous.*

Proof. We first note that permutation π does not affect S and Λ^0, which implies that each value of $\Lambda(f) - s$ in Steps 2 and 3 would not change under π. Thus, even after the permutation, each flight $f \in F$ is also (according to φ^T) assigned slot s, to which f has been already assigned under problem P. The name of the airline who owns f and s, of course, may change. However, since $\varphi^T(\pi P)$ is a landing schedule that does not specify the ownership of slots, we can obtain the desired result. □

Finally, the table below summarizes the properties of each algorithm. One can easily check that the truncation algorithm satisfies all the properties except for the core condition.

Property	Compression	Truncation	Trade-cycle	DA
Non-wastefulness	○	○	○	○
Individual rationality	○	○	○	○
Incentive to report delays	○	○	○	○
Incentive to report cancellations	×	○	×	○
Outcome in the core	×	×	○	×
Anonymity	○	○	○	×

6 Concluding remarks

To reassign airport landing slots more efficiently and fairly, we have introduced the truncation algorithm which is non-wasteful and individually rational, and proved that the algorithm gives airlines the incentives to report both delays and cancellations. Although the truncation algorithm does not always produce an outcome in the core, it satisfies anonymity.

Schummer and Abizada (2013) extends the model of Schummer and Vohra (2013) by explicitly considering the airlines' preferences. One of the remaining tasks is to check whether the truncation algorithm can be applied to the model considering the airlines' preferences explicitly. Another remaining task is to characterize the truncation algorithm, based on (i) individually rational monotonicity, which requires that, when an airline claims a subset of the set of its own slots, all airlines be weakly better off, (ii) population monotonicity, which requires that, when a new airline shows up to share the slots in the airport, all airlines be weakly worse off, or (iii) consistency, which requires that, when some airlines leave the airport with the assigned slots, the assignment of the other slots among the other airlines be consistent.[8] We leave these issues for future research.

[8]For the formal definitions of individually rational monotonicity, population monotonicity, and consistency, see Kojima and Manea (2010) and Kesten (2006).

References

[1] Abdulkadiroğlu, A. and T. Sönmez (2003) "School choice: a mechanism design approach," *American Economic Review* 93(3): 729-747.

[2] Ball, M., R. Hoffman, and T. Vossen (2002) "An analysis of resource rationing methods for collaborative decision making," Paper presented at Air Traffic Management (ATM) Workshop 2002, Capri, September: 22-26.

[3] Gale, D. and L. Shapley (1962) "College admissions and the stability of marriage," *American Mathematical Monthly* 69: 9-15.

[4] Kesten, O. (2006) "On two competing mechanisms for priority-based allocation problems," *Journal of Economic Theory* 127: 155-171.

[5] Kojima, F. and M. Manea (2010) "Axioms for deferred acceptance," *Econometrica* 78(2): 633-653.

[6] Schummer, J. and A. Abizada (2013) "Incentives in landing slot problems," *Working paper*.

[7] Schummer, J. and R. V. Vohra (2013) "Assignment of arrival slots," *American Economic Journal: Microeconomics* 5(2): 164-185.

[8] Shapley, L. and H. Scarf (1974) "On cores and indivisibility," *Journal of Mathematical Economics* 1: 23-37.

[9] Vossen, T. and M. Ball (2006a) "Optimization and mediated bartering models for ground delay programs," *Naval Research Logistics* 53(1): 75-90.

[10] Vossen, T. and M. Ball (2006b) "Slot trading opportunities in collaborative ground delay programs," *Transportation Science* 40(1): 29-43.

[11] Wambsganss, M. (1997) "Collaborative decision making through dynamic information transfer," *Air Traffic Control Quarterly* 4: 107-123.

국문초록

항공기운항시각 재조정을 위한 절단 알고리즘

서울대학교 대학원
경제학부 경제학 전공
최 인 혁

기상상황의 악화로 인하여 공항에서의 기존 착륙 일정이 비효율적으로 변할 경우 미국의 연방항공국(the Federal Aviation Administration)은 항공사들의 유인(incentive) 문제를 고려하는 가운데 공항의 착륙 슬롯(landing slot)이 낭비되지 않도록 착륙 일정을 재조정한다. 현재 미연방항공국은 공항에서의 착륙 일정 재조정을 위하여 항공사들에게 항공기의 지연을 공항에 정직하게 보고할 유인을 제공하는 압축 알고리즘(the compression algorithm)을 사용하고 있다. 그러나 압축 알고리즘의 경우 항공사들에게 항공기의 취소를 공항에 정직하게 보고할 유인을 제공하지 못한다는 문제점을 지니고 있다. 이에 본 논문에서는 항공사들에게 항공기의 지연 및 취소를 공항에 정직하게 보고할 유인을 모두 제공하는 절단 알고리즘(the truncation algorithm)을 새롭게 제시한다. 나아가 절단 알고리즘이 공항의 착륙 슬롯을 낭비하지 않음(non-wastefulness)은 물론 개별적 합리성(individual rationality)의 조건 역시 충족시킴을 보인다.

주요어: 유인(incentive), 알고리즘(algorithm), 공항(airport)
학 번: 2011-23175