The Aircraft Sequencing Problem with Arrivals and Departures

by

Alp Muharremoglu

B.S., Industrial and Operations Engineering (1997)
The University of Michigan, Ann Arbor, MI

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Operations Research

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2000

Author .....
Department of Electrical Engineering and Computer Science
December 17, 1999

Certified by .....
Amedeo R. Odoni
Professor of Aeronautics and Astronautics and of Civil and Environmental Engineering
Thesis Supervisor

Accepted by .....
Cynthia Barnhart
Associate Professor of Civil and Environmental Engineering
Co-Director of the Operations Reseach Center

The Aircraft Sequencing Problem with Arrivals and Departures

by

Alp Muharremoglu

Submitted to the Department of Electrical and Computer Engineering
on December 17, 1999, in partial fulfillment of the
requirements for the degree of
Master of Science in Operations Research

Abstract

This thesis investigates the Aircraft Sequencing Problem (ASP) with Arrivals and Departures. The ASP is the problem of sequencing the arriving and departing aircraft on a single runway to minimize certain performance criteria. We focus on minimizing the total weighted delay. Both the theoretical aspects of the problem, and some practical issues are discussed. The static version of the problem is basically a scheduling problem with sequence dependent processing times and ready times, with the objective of minimizing total weighted delay. Exact algorithms for this problem are not fast enough for practical implementation. We give several algorithms that can be used both for the static and the dynamic versions of the problem. These algorithms are not exact solutions, however they are much faster than an exact algorithm and address some very important practical issues related to the ASP. Computational results from these algorithms are given. The computational results demonstrate that the potential benefits of using optimization in the sequencing of arrivals and departures in the Terminal Area are fairly significant. For example, the algorithm HWTW with $MPS = (0, 0)$ reduces delays by 40% compared to FCFS. Certain fairness and safety issues are addressed as well.

Acknowlegmets

I would like to thank my advisor, Prof. Amedeo R. Odoni for his support during the past two years. This research was partially supported by the Federal Aviation Administration (FAA) under the project "Advanced Concepts for Collaborative Decision Making (CDM)," award number SA1603JB and by the Charles Stark Draper Laboratory Inc., under Contract Number DLH-505328.

Thesis Supervisor: Amedeo R. Odoni
Title: Professor of Aeronautics and Astronautics and of Civil and Environmental Engineering

Contents

List of Figures

List of Tables

Chapter 1

Introduction

"Worldwide growth in air traffic, especially in the U.S., Europe and the Pacific rim requires more efficient utilization of airspace and airport operational resources. Limited airport capacity continues to be a major cause of costly delays."[9]

The airport is one of the integral parts of the air transportation system. With the increasing use of air transportation, the capacity of many major airports is being exceeded at congested periods. This causes delays for incoming or outgoing flights, which in turn causes more problems down the road when a certain aircraft or crew is late for the next leg. Currently, the airports and specifically runways is the principle bottleneck of the air transportation system. The reason is that the runways are the interface between the three dimensional airspace and a 'single file flow' traffic regime. [11] However, most airports were built within or near a city and financial, physical, environmental or political limitations usually make it impossible to expand the airport physically such that it can service more aircraft. For this reason, it is very important to utilize the existing airport capacity to its fullest.

Due to the requirements imposed by the FAA, aircraft within the terminal area of an airport have to maintain a certain longitudinal distance from each other. In addition, only a single aircraft is allowed on the runway at a time. These requirements translate into a minimum duration that has to pass between two movements on the runway. This minimum duration depends on the weight class of both the trailing and the leading aircraft. In other words, the sequence in which the aircraft use the runway affects many performance criteria, such as the average delay per aircraft. This suggests that optimization of the sequence of operations on the runway can potentially increase the efficiency of the airport, which is the main bottleneck

of the air transportation system. This in turn can have a significant positive influence on the whole system. In the literature, the problem of 'optimally' sequencing the arriving aircraft has been referred to as the Aircraft Sequencing Problem (ASP). The subject of this thesis is the sequencing of both arrivals and departures on a runway, hence the ASP with arrivals and departures.

The control of the traffic in the near terminal area of the airport has become a very complex task. The air traffic controller has to make several decisions in a relatively short time, such as deciding which aircraft should use which runways, in what sequence should the aircraft depart or land, what maneuvers should be executed, etc. As a consequence, it is very difficult for the controller to think about minimizing the delay by considering different sequences. If one wants to take advantage of the structure of the minimum separation requirements using optimization techniques, an automated real time scheduler is necessary to act as a decision tool for the controller. Such a scheduler has been designed for arriving aircraft as part of the Center TRACON Automation System (CTAS). However, CTAS incorporates very limited optimization of the different sequences. Practical constraints such as the dynamic nature of the problem and lengthy computation times are given as a reason for doing so [10]. In addition, the interaction of arrivals and departures may change the story if a runway is used for both types of operations at the same time.

Our goal in this thesis is to try to understand the potential benefits of optimizing the sequence of arriving and departing aircraft. To do so, we have investigated the theoretical nature of the problem and then developed several algorithms by trying to keep in mind as many practical issues as possible. We have restricted ourselves to the case when a single runway is used for both arrivals and departures. We hope that we have demonstrated the potential benefits of optimization and some of the trade-offs involved in this very difficult problem. In particular, we have shown that there are sequencing rules yielding considerable improvements on the efficiency of the system that do not require lengthy computation times.

The rest of this thesis is organized as follows: Chapter 2 introduces the Aircraft Sequencing Problem and some of the practical issues that are related to the ASP. In Chapter 3, we give a review of the existing literature on the problem. Chapter 4 gives two exact algorithms for static versions of the ASP, one for the problem without ready times, and one for the problem with ready times. Ideas about some dominance rules for general objective functions in the case where

we have ready times are also included in Chapter 4. In Chapter 5, we develop two heuristic algorithms for the ASP. The second of these heuristics is seen to perform better. In Chapter 6, the second algorithm of the previous chapter is modified to include the Constrained Position Shifting idea. Chapter 7 reports the numerical results obtained by the implementation of these algorithms. Chapter 8 concludes the thesis.

Chapter 2

The Aircraft Sequencing Problem with Arrivals and Departures

2.1 The Minimum Separation Requirements

We are concerned with the sequencing of arriving and departing aircraft on a single runway. The case where only arrivals are sequenced has been considered by several researchers in the past and some of their work will be mentioned in Section 3.

First, we discuss the minimum separation requirements between different movements on the runway. These requirements stem from the wake vortex safety rules determined by the FAA. These are minimum longitudinal distances that the aircraft have to maintain between each other. These distances depend on the sizes of the two aircraft in question. The aircraft are divided into categories with respect to their weight class: Heavy (H), large/medium (L/M) and small (S). The aircraft in the class L/M have widely varying characteristics and can be further subdivided into two subcategories, L and M. If we have arrivals and departures at the same time, we can view an arrival of class H as a different class than a departure of class H, which means that we will have a total of 8 aircraft categories. Using the speeds and the runway occupancy times of landing and departing aircraft and the wake vortex separation requirements, a minimum separation matrix in terms of seconds between each pair of categories can be calculated. Odoni [11] uses equations of motion to do this conversion between distances and time for arriving aircraft. Using a similar methodology, we have calculated a typical matrix that we shall use henceforth in this thesis, without loss of generality, which is given in Table 2.1. Basically, the numbers were calculated

so that at any time the airborne separation requirements are respected and at any time there is only one aircraft on the runway as required by the FAA.

Leading\Trailing	H_{Arvl}	L_{Arvl}	M_{Arvl}	S_{Arvl}	H_{Dept}	L_{Dept}	M_{Dept}	S_{Dept}
H_{Arvl}	96	146	182	195	70	70	70	70
L_{Arvl}	60	69	92	186	60	60	60	60
M_{Arvl}	60	69	82	175	55	55	55	55
S_{Arvl}	60	69	82	100	50	50	50	50
H_{Dept}	65	65	65	65	90	120	120	120
L_{Dept}	55	55	55	55	60	60	60	60
M_{Dept}	45	45	45	45	60	60	60	60
S_{Dept}	40	40	40	40	60	60	60	60

Table 2.1: The Minimum Separation Matrix

Note that this matrix does not satisfy the triangular inequality. If we define $t_{i,j}$ as the minimum separation when a type j aircraft is following a type i aircraft, the triangular inequality says that $t_{i,j} + t_{j,k} \geq t_{i,k}$ for all i, j, k. In the above matrix, this inequality is not satisfied for example when S_{Arvl} is k, H_{Arvl} is i and S_{Dept} is j, i.e. $70 + 40 \ngeq 195$. However, after a heavy aircraft lands, a minimum of 195 seconds have to pass before a small aircraft can land, whether or not another small aircraft took off in between. In the following sections, the algorithms we propose will initially assume that the triangular inequality is satisfied and afterwards we will explain how to modify the generated schedules such that the problems that this may cause will be avoided.

2.2 Mathematical Formulation

Consider a situation in which a number of aircraft have to be scheduled to land on or to depart from a single runway. The aircraft are categorized as belonging to N different classes. First, there is a categorization with regard to the size of the aircraft; for example, small, medium and large and heavy. Second, the aircraft are also distinguished as being either an arrival or a departure.

As explained in Section 2.1, there is a minimum separation requirement between different classes of aircraft. More precisely, the time between the starting epochs of two operations has to be greater than or equal to a certain threshold. Suppose that we want to use the runway for an operation of type j after using it for an operation of type i. Then for any feasible schedule, t_{ij} is the minimum time between the starting times of the two operations.

We denote k_l^i as the number of aircraft of type i that have not been serviced yet after the l^{th} aircraft in the schedule has been serviced. So, k_0^i is the number of aircraft of type i that are initially present. The total number of aircraft in the system is $T = \sum_{i=1}^{N} k_0^i$.

In addition, there is a "ready time" associated with each aircraft. This is the time when a particular aircraft becomes available for landing or for departure. We are given a vector $R_i = (r_i^1, r_i^2, ..., r_i^{k_0^i})$ for each type $i = 1, 2, ...N$, where r_i^j is the ready time of the j^{th} aircraft of type i. The aircraft are ordered within a type so that $r_i^1 \leq r_i^2 \leq r_i^3 ..., \leq r_i^{k_0^i}$.

A sequence $\lambda = (\lambda(0), \lambda(1), ...\lambda(T))$ is a vector, where $\lambda(l)$ is the type of the l^{th} aircraft in the sequence. $\lambda(0)$ is defined to be 0. For example, $\lambda(5) = 3$ means that the 5^{th} aircraft to be serviced by the runway is of type 3. $\widehat{\lambda} = \left(\widehat{\lambda}(0), \widehat{\lambda}(1), ...\widehat{\lambda}(T)\right)$ is another vector associated with the sequence λ, where $\widehat{\lambda}(l) = y$ means that the l^{th} aircraft in the sequence, which is of type $\lambda(l)$ is the y^{th} aircraft of type $\lambda(l)$ in the sequence. Given λ, $\widehat{\lambda}$ is uniquely determined, if we assume that an aircraft of a particular type that has an earlier ready time than another aircraft of the same type will be serviced earlier than the one with the later ready time, i.e. we will adhere to the First Come First Serve rule within each type. By an interchange argument, it is easy to show that any schedule which does not satisfy the above assumption cannot be optimal in the sense that we describe below.

We are given a weight ω_i associated with each type of aircraft. The value of a schedule λ is:

$$Z_1^{\lambda} = \sum_{l=1}^{T} \omega_{\lambda(l)} \cdot d_l^{\lambda} \tag{2.1}$$

where,

$$d_l^{\lambda} = c_l^{\lambda} - r_{\lambda(0)}^{\widehat{\lambda}(0)} \tag{2.2}$$

is the delay absorbed by the l^{th} aircraft in the sequence when sequence λ is used. c_l^{λ} is the starting time of the service of the l^{th} aircraft under the sequence λ, and given λ these numbers can be recursively calculated by using the following system of equations:

$$\begin{aligned} c_0^{\lambda} &= 0; \\ c_l^{\lambda} &= c_{l-1}^{\lambda} + \max\left[t_{\lambda(l-1)\lambda(l)}, r_{\lambda(l)}^{\widehat{\lambda}(l)} - c_{l-1}^{\lambda}\right] \text{ for } l = 1, 2, ...T \end{aligned} \tag{2.3}$$

The problem is to find a feasible schedule λ, that minimizes Z_1^λ. The feasibility conditions for λ are that the minimum separation requirements are satisfied and that no aircraft is scheduled to use the runway before its ready time. The separation requirements follow from the recursive definition of c_l^λ's. The ready time requirement corresponds to the delays being nonnegative.

Also note that a schedule minimizes the total weighted delay if and only if it also minimizes the weighted sum of the starting times c_l^λ, since these two objective functions differ only by a constant.

To summarize, the problem that we want to solve is a static optimization problem; to find a schedule for a given set of aircraft with ready times, that minimizes the total weighted delay of all aircraft. The minimum time between each operation is sequence dependent.

2.3 Practical Issues

The static problem described above is closely related to the real life problem of sequencing landings and takeoffs on a single runway. However, in real life, many practical issues have to be considered. Here, we will list some of these issues and in the rest of the thesis, address them in the algorithms we propose.

1. The real life problem is not a static problem. We do not have a given number of aircraft whose ready times we know in advance, but are informed about the arrival or departure requests as time advances. Any practical algorithm has to make decisions with the information at hand and proceed, i.e. develop a sequence in a dynamic manner, (see Chapter 5).

2. Safety and fairness considerations need to be addressed. For this reason, a methodology such as the Constrained Position Shifting approach has to be implemented, (see Chapter 6).

3. The algorithm must be fast enough to be usable in practice. Erzberger [10] states that "...in practice a response within 10-15 seconds has been found to be acceptable and qualifies as real time performance," (see the running time results in Chapter 7).

4. The sequencing decisions must be finalized (i.e. cannot be modified further) a specified time in advance. We refer to this time as the "Freeze Horizon" (FH). FH can, in practice,

be different for arriving aircraft and for departing aircraft. Our algorithms can be used to incorporate consideration of the FH. We can apply the algorithms to determine the moments that are FH before the aircraft actually start using the runway. After these moments, the corresponding movement's position in the sequence will be "frozen".

5. The generated schedules have to respect the triangular inequality. This issue is addressed in Section 6.3.

Chapter 3

Literature Review

The Aircraft Sequencing Problem for arrivals has been considered by many researchers. Dear [1] was first to investigate this problem in depth. He noted that there are potential benefits in sequencing runway arrivals and used an algorithm that enumerates the sequences of a small number of arriving aircraft. He assumed that the ready times of all aircraft were zero, i.e. that all aircraft are ready to use the runway at any time. He was also the first one to introduce the Constrained Position Shifting (CPS) idea. The idea behind CPS is that the absolute difference between the position of an aircraft in the FCFS sequence and the position of the aircraft in the sequence generated by the algorithm should not exceed a certain threshold value. We will elaborate more on this idea in Section 6. He contributed to the mathematical formulation of the problem, however his enumerative approach was too restrictive for computational purposes.

Psaraftis [2] continued this line of research, but used a dynamic programming formulation to solve two versions of the sequencing problem of arriving aircraft without ready times: one to minimize the makespan (i.e. completion time of the last aircraft in the sequence) and another to minimize the total delay. In [3], he generalized the ideas in his previous study and gave an algorithm for very general, yet additive cost functions. His algorithms were very efficient and exact, but they assumed that all the aircraft were ready to land at any time.

Recently, Bianco et al. [5] have developed a local search heuristic that is applied to the ASP. They consider arriving aircraft and the objective is to minimize the total completion time. Their formulation includes ready times for the aircraft. This formulation as it is given in the paper is a special case of our formulation, since we are trying to minimize total weighted completion time. However, the general ideas in their paper could be applied to the problem of minimizing

the total weighted completion time as well.

Venkatakrishnan et al. [13] did a study on the landings at Boston Logan Airport. The first part of their paper deals with the statistical analysis of the times between the landings of different types of aircraft. The second part proposes three algorithms for the ASP for arriving aircraft; one static and two dynamic algorithms. The static algorithm is a modification of Psaraftis' algorithm to include time windows, i.e. lower and upper bounds representing, respectively, the earliest and the latest times at which a particular aircraft can land. The two dynamic algorithms are essentially solving the static algorithm repetitively. The algorithms are tested using actual data from the Logan Airport and it is shown that in some cases a 30% improvement in average delay is possible.

The static version of the ASP is similar to the famous Traveling Salesman Problem (TSP) [6][7]. The similarity stems from the fact that we have a sequence dependent time between movements on the runway, like the distances between the cities in the TSP. There is extensive literature on the TSP and related problems. Among these, the most relevant work to the ASP belongs to Tsitsiklis [12]. In this paper, Tsitsiklis considers what he calls "B-TRPTW, The Traveling Repairman Problem with Time Windows" where the number of nodes is bounded by B. He defines the Traveling Repairman Problem as the modification of the TSP where the objective is to minimize the total delay. He shows that when there are only release times, this problem is solvable in $O(B^2 n^{B^2+1})$. He uses a dynamic programming algorithm which can be used to minimize total weighted delay as well. In this case, the formulation is equivalent to the formulation of the static ASP that we gave in Section 2.2. This showed that the static problem is polynomially solvable. However, in the case of the ASP, $B = 8$, so the algorithm runs in $O(n^{65})$. This is not fast enough for a practical implementation. In addition, the practical implementation requires a dynamic algorithm besides other practical considerations that were mentioned in Section 2.3.

Gilbo [8] does a study on the estimation and optimization of airport capacity. His approach is not to consider each flight separately but rather to look at the total number of arrivals and the total number of departures over periodic intervals. He uses optimization for dynamically determining capacities to be allocated to arrivals and to departures over a time horizon. [9] extends these results to the case where the airport capacity is stochastic. These papers are not directly related to the ASP, however we mention them because they deal with the interaction

between arrivals and departures in the airport.

The Center Tracon Automation System (CTAS) is a set of tools developed jointly by NASA Ames Research Center and by the Federal Aviation Administration (FAA). These tools generate advisories that assist controllers in handling aircraft from about 40 minutes of flying time to an airport, until they reach the final approach fix. Erzberger [4] explains the algorithms and principles employed within CTAS in detail. CTAS is one of the first practical implementations of an automated decision tool that deals with the sequencing of terminal area traffic. Even though this is very encouraging, the sequencing techniques used in CTAS are very limited, partly in an attempt to avoid lengthy computation times. We show that, with the computer power that is available today, one can do much better with regard to optimization and still ensure that running times will be within the acceptable short amount of time that a practical implementation mandates.

Chapter 4

Exact Algorithms

4.1 The Dynamic Programming Algorithm for the Aircraft Sequencing Problem without Ready Times

The problem we are going to analyze in this section is the scheduling of arrivals and departures on a single runway such that the total weighted delay is minimized. However, this problem differs from our formulation in Section 2.2 in that here we will not allow aircraft to have different ready times. Hence, we will assume that all aircraft are available to land or to depart.

Psaraftis [2] presents a dynamic programming algorithm to solve the problem of minimizing the total delay, but the same ideas can be used to minimize the total weighted delay. We shall explain below the details of this DP formulation, because it will be used in the algorithms that we shall develop in later sections.

We shall use the following notation:

N = the number of aircraft types.

ω_j = the weight associated with a type j aircraft.

t_{j_1,j_2} = the sequence-dependent separation time; this means that after an aircraft of type j_1 starts using the runway, at least t_{j_1,j_2} units of time have to elapse before an aircraft of type j_2 can start using the runway.

k_n^j = the number of aircraft of type j that have not been scheduled yet after n aircraft have already been scheduled; so, k_0^j is the initial number of aircraft of type j.

$T = \sum_{j=1}^{N} k_0^j$;i.e. the total number of aircraft.

L_n =the type of the last serviced aircraft, after n aircraft have already been scheduled.

The decision epochs are the times when an aircraft starts service. At any decision epoch, the necessary information to optimally schedule the remaining aircraft constitutes the state of the DP algorithm. Here, the necessary information is the number of remaining aircraft of each type plus the type of the last aircraft that was on the runway. So, the state can be written as:

$$\begin{aligned} x_n &= (L_n, k_n^1, k_n^2, ..., k_n^N) \\ x_n &\in X_n \text{ where } X_n = \{x_n : L_n \in \{1,2,...,N\}, \sum_{j=1}^{N} k_n^j = T - n, k_n^j \leq k_0^j \text{ for } j \in \{1,2,...,N\}\end{aligned} \tag{4.1}$$

where n is the state variable corresponding to the number of aircraft that have already been serviced.

$x_0 = (L_0, k_0^1, k_0^2, ..., k_0^N)$ = The initial state. L_0 can be an aircraft type $(1,2,...,N)$, in which case an aircraft of type L_0 has just started using the runway at time 0, or L_0 can be 0, which would mean that the runway is empty at time 0 and any aircraft can start using the runway immediately.

The decision at every service decision epoch is to choose what type of aircraft to serve next.

u_n = the type of the $(n+1)^{th}$ aircraft to be serviced.

$u_n \in U(x_n)$, where $U(x_n) = \{y \in \{1,2,...,N\} : k_n^y > 0\}$

The operation of the dynamic program (DP) can be described by the following set of equations:

$$\begin{aligned} x_{n+1} &= f(x_n, u_n) \\ f(x_n, u_n) &= (u_n, k_{n+1}^1, ..., k_{n+1}^N) \text{ for } n \in \{1,...,T\} \\ \text{where } k_{n+1}^j &= \begin{cases} k_n^j - 1 \text{ if } u_n = j \\ k_n^j \text{ if } u_n \neq j \end{cases} \quad \text{for } j \in \{1,2,...,N\} \end{aligned} \tag{4.2}$$

The incremental cost of making the decision u_n, when the state is x_n, can be written as follows:

$$g(x_n, u_n) = t_{L_n, u_n} \cdot \sum_{j=1}^{N} (k_n^j \cdot w_j) \qquad \text{for } n = 1,2,...,T-1 \tag{4.3}$$

$g(x_n, u_n)$ is the cost of scheduling an aircraft of type u_n to use the runway next given that we are in state x_n. This cost basically adds the weighted delay incurred by all aircraft during the current separation time to the objective function. x_T is simply the final state, meaning that all aircraft have been serviced.

We define the optimal value function, $J_n(x_n)$ as the minimum total weighted delay that will be incurred if we optimally schedule the aircraft, starting from state x_n. There is no termination cost, so that $J_T(x_T) = 0$. $J_0(x_0)$ is the minimum total weighted delay that would be incurred starting from the initial state, i.e. the optimal value of the problem is $J_0(x_0)$.

The DP recursions are:

$$\begin{aligned} J_T(x_T) &= 0 \\ J_n(x_n) &= \left\{ \min_{u \in U} \left[g(x_n, u_n) + J(f(x_n, u_n)) \right] \text{ for } x_n \in X_n \right. \end{aligned} \tag{4.4}$$

Note that n, the stage variable is equal to $\sum_{j=1}^{N} k_n^j$. In the implementation of the algorithm, it is easier to suppress the dependence on n and proceed lexicographically in the states (putting L_n at the end of the state vector). This way, any optimal value of a state is calculated before we will need it in the following recursions.

4.2 An Algorithm to Minimize the Makespan in the Presence of Ready Times

4.2.1 Problem Statement

The problem in this section is the scheduling of a given set of arriving or departing aircraft on a single runway so that the time until the last movement on the runway is minimized (in machine scheduling terminology, to minimize the makespan). The aircraft belong to N different categories and within each type, the aircraft are identical. Each aircraft has a ready time, meaning that this particular aircraft cannot land (or depart) before the corresponding ready time. There is a sequence-dependent separation time that has to elapse between two consecutive movements on the runway, meaning that the time the runway is kept busy depends both on the type of the aircraft that is currently being serviced and also on the type of the aircraft that was serviced

before this one.

This problem is identical to the problem explained in Section 2.2, except for its objective function. The objective function in this case is to minimize the makespan instead of minimizing the total weighted delay. In certain situations, such an objective function might be suitable. For example, the controller may be trying to clear up the runway as soon as possible for some important future aircraft that are expected. Even though this objective function is additive, we cannot use the algorithm in Psaraftis [3], since that paper did not model the ready times.

The algorithm that we propose has a running time that grows exponentially in the number of aircraft types, but once the number of aircraft types is fixed, then the algorithm runs in a time bounded by a polynomial in the number of aircraft.

4.2.2 Formulation

There are N aircraft types with k_0^i aircraft belonging to type i. Each aircraft has a ready time and the ready time of the j^{th} aircraft of type i is r_i^j. If an aircraft of type i was processed last, then an aircraft of type j cannot use the machine until after t_{ij} time units after the last aircraft started. The initial aircraft processed is assumed to be of type L_0. The objective is to find a schedule for all aircraft such that the total time to service all the aircraft on the machine will be minimized.

We shall use dynamic programming to solve this problem. To completely describe the state of the system we need the following state vector (in the following, for notational simplicity, we suppress the dependence on n, the number of aircraft that were scheduled so far):

$$x = (t, L, k^1, k^2, ..., k^N) \tag{4.5}$$

where,

t : The current time.

L : The type of the aircraft that was serviced last.

k^i $(i = 1, 2, ..., N)$: The remaining number of aircraft of type i.

The decision epochs are the moments when an aircraft starts the movement (landing or takeoff). The decision at each decision epoch is $u \in \{1, 2, ..., N\}$, the type of the next aircraft to be serviced. If $u = i$, then the aircraft that has the earliest ready time out of all the aircraft belonging to type i that have not been serviced yet will be the next aircraft to use the runway.

This approach implicitly assumes that we are going to adhere to the FCFS rule within each type. But since aircraft within each type are identical, it is easy to show that this is indeed optimal.

Using this state definition and appropriate cost functions, a dynamic programming formulation could be written for any objective function (assuming that the cost functions are additive). But since the state vector includes t, the current time as a component, the DP network is not likely to experience any aggregation and would almost be a tree. The reason is that two different subsequences that serve the same number of aircraft from each type will most likely take different amounts of time, because of the asymmetries in the separation matrix. Hence, the state space would be huge and we would in essence be enumerating all the possible sequences when going through the DP recursion. Yet, the inclusion of the time in the state is necessary because we have to be able to know which aircraft are available or which aircraft will be available in how much time.

However, the case of minimizing the makespan is different and rather special. The observation we make is that the value associated with a state is the time it took to reach that state. Consequently, since information about the current time is available as the value of a certain state, this piece of information does not have to be repeated as a component of the state vector as well. This enables us to eliminate t from our state definition and we end up with $x = (L, k^1, k^2, ..., k^N)$ as our state vector. As a result of this modification, the size of the state space is dramatically decreased. For example, now an initial sequence of $1, 2, 3$ takes us to the same state as an initial sequence of $2, 1, 3$, whereas these two initial sequences would most likely take us to different states if the time was included in the state, because the sequences are most likely to take different amounts of time due to the sequence dependent separation times.

We now proceed to solve this problem via dynamic programming. Note that in this setting, since at every decision epoch we need to know which aircraft are available (and hence the current time) a backward recursion is not possible. Hence, we will use forward recursion. The optimal value function can be defined as $J(x)$, the earliest time the state x can be reached. The initial state is $x_0 = (L_0, k_0^1, k_0^2, ..., k_0^N)$. The total number of aircraft is $T = \sum_{i=1}^{N} k_0^i$. The DP recursions are as follows:

$$J(x_0) = 0 \tag{4.6}$$

$$
\begin{aligned}
J(x) &= \left\{ \min_{u \in U} [g(x,u) + J(f(x,u))] \text{ for } x \in X \right. \\
X &= \left[x : k^i \in \{0, 1, ..., k_0^i\} \text{ for all } i, \sum_{i=1}^{N} k^i < T, \; L \in \{1, 2, ..., N\} \right] \\
U &= [u : k^u + 1 \leq k_0^u] \\
f(x,u) &= (L', k^{1\prime}, ..., k^{N\prime}) \\
\text{where } k^{i'} &= \left\{ \begin{array}{l} k^i + 1 \text{ if } u = i \\ k^i \text{ otherwise} \end{array} \right. \text{ for } i = 1, 2, ..., N \\
\text{and } L' &= u
\end{aligned}
$$

By now, we have formulated everything except for the cost function $g(x,u)$. In words, this cost function has to be equal to the time it would take to service the next aircraft of type u if we are currently in state x. Mathematically:

$$
\begin{aligned}
g(x,u) &= \max \left[t_{uL}, \min_{j \in J_u} (r_u^j - J(f(x,u)))^+ \right] \qquad (4.7) \\
&\text{where } J_u \text{ is the set of aircraft of type } u \text{ that have not been serviced yet} \\
\text{and } (a)^+ &= \left\{ \begin{array}{l} a \text{ if } a > 0 \\ 0 \text{ otherwise} \end{array} \right.
\end{aligned}
$$

The maximization is between the minimum separation requirement and the time until the next aircraft of type u will be ready. For example, if the minimum separation between a type 1 and a type 3 aircraft is 50, we are at a decision instant where the a type 1 aircraft just used the runway, and the next aircraft of type 3 will be ready in 70 time units, then the aircraft of type 3 can be scheduled at 70 time units after the current time. However, if the aircraft will be released in 30 time units, then the earliest time it can be scheduled is 50 time units after the current time to satisfy the separation requirement.

The minimum makespan is the minimum of the values of the states that have $k_i = 0$ for $i = 1, 2, ..., N$, i.e.:

$$
J^* = \min_{L \in \{1,...,N\}} J(L, 0, 0, ...0) \qquad (4.8)
$$

The recursions can be carried out by moving lexicographically downwards, starting from the

initial state. The optimal schedule can also be retrieved by simply keeping track of the control made at each recursion.

The running time of the algorithm is of the order of $N^2 \prod_{i=1}^{N}(1+k_0^i)$. There are $N \cdot \prod_{i=1}^{N}(1+k_0^i)$ states and N controls have to be examined at each state. If N is fixed, this is a polynomial function of the number of aircraft of each type.

4.3 Dominance Rules for More General Objective Functions in the Presence of Ready Times

The previous section dealt with minimizing the makespan, whereas our main problem of Section 2.2 was to minimize the total weighted delay. In this section, we will lay out some ideas about what can be done if we have a general additive objective function; the total weighted delay being a special case of this.

As mentioned in Section 4.2, when we have time as a component of the state vector, the state space is very large and we need to find a way to either aggregate some states or find some dominance rules between states so that the execution of a DP could be practical. Our approach will be to try to find dominance rules among states that have all components identical except the time and the objective function value. In other words, we are going to compare states where L and the k^i's are equal, i.e. the same aircraft have been serviced, but in a different sequence, the last aircraft being of the same type. Let's call each set of states with such a property "a node set" meaning the set of states corresponding to a node in another DP network without a time component in the state. This way, we partition the state space into node sets. The special feature about the makespan objective is that the optimal value function and the time are the same and a state with a higher value (time) can be eliminated. This means that in each "node set", a single state dominates all the other states and we end up with a polynomial number of states. If we have a general objective function, it is not always clear which state in a node set could dominate which other states.

Suppose we have two states, $x = (t, L, k_1, k_2, ..., k_N)$ and $x' = (t', L, k^1, k^2, ..., k^N)$, i.e. these two states are in the same node set. Each state has a value; $J(x)$ and $J(x')$. In general we do not know which state will lead us to a sequence with lower cost. There are four different possibilities:

1. $J(x) \leq J(x')$ and $t \leq t'$

2. $J(x) > J(x')$ and $t > t'$

3. $J(x) \leq J(x')$ and $t > t'$

4. $J(x) > J(x')$ and $t \leq t'$

If we have case 3 and we look into the future from states x and x', the optimal scheduling of the remaining aircraft is a smaller version of our original problem. The difference between these two subproblems is that the new ready times of the remaining aircraft differ by a constant, $t-t'$. The subproblem originating from state x has the same set of aircraft as the one originating from state x'. But since $t > t'$, the new ready times of the latter are all greater than the corresponding ready times of the previous subproblem. This in turn means that the latter subproblem has an optimal value that is greater than or equal to the optimal value of the previous subproblem. However, all the completion times are increased by $t-t'$ as well. In addition, since $J(x) \leq J(x')$, we arrive at the condition that in Case 3, x dominates x' if $J(x) - C \cdot (t - t') \leq J(x')$ (where $C = \sum_{i=1}^{N} k^N$ if the objective is to minimize the total delay. Note that C can be changed depending on the objective function, e.g. $C = \sum_{i=1}^{N} w_i \cdot k_i$ when the objective is to minimize the total weighted delay.)

If we have case 4, the reverse argument gives a symmetric condition.

If we have cases 1 or 2, note that case 2 can be seen as a special case of case 1 if we rename the states. So, let's focus on case 1. We have $J(x) \leq J(x')$ and $t \leq t'$. Suppose that we are at state x and let the runway idle for $t' - t$ time units. Also suppose that the increase in the objective value due to this idling is smaller than $J(x') - J(x)$. In this case, we can eliminate the state x'. In other words, if we are at a state where by simply idling we can reach another state and still have a lower objective value, then our current state dominates the latter state. More precisely: If we have case 1 and $J(x) + C \cdot (t' - t) \leq J(x')$ then x' is dominated by x.

To get an intuitive idea of how often dominance will occur within a node set, it is useful to think geometrically. In Figure 4-1, each point represents a state belonging to a certain node set (i.e. all states in the figure belong to the same node set). The increasing lines have a slope of C and the decreasing lines have a slope of $-C$. The dominance rules that we described above tell us that only the three states that are at the corners of the lines will survive, the others are dominated by them. In particular, looking at the left most point, the two lines that emanate from it represent the dominance conditions that were given. Any state that is above the two

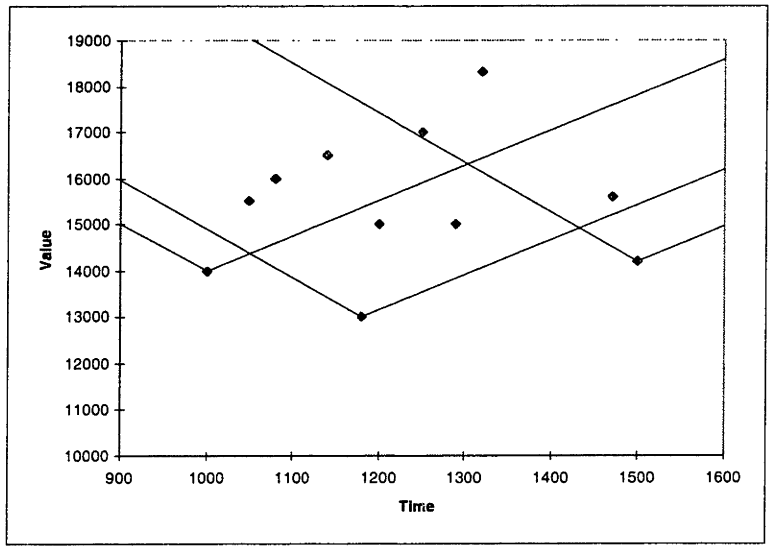

Figure 4-1: Dominance in a "Node Set"

lines that emanate from a state are dominated by that state. Hence, in Figure 4-1, five states are dominated by the left most state and it itself is not dominated by any state.

If we could formally show that the number of non-dominated states within a node set is bounded by a polynomial in the number of aircraft, then we would have found a new polynomial time algorithm for the problem of scheduling groups of identical aircraft with ready times to minimize a general additive objective function. Even if there is no such bound, if in practice most of the states within a node set are dominated by other states, then the DP algorithm can be very fast. However, we have not investigated this matter thoroughly.

One way to address heuristically the problem is to assume that within each node set one state will dominate every other state in the same node set. In particular, we can assume that the state with the lowest value function $J(x)$ will dominate. This means that we do not have to keep time as a state component anymore, and we can keep track of time by simply keeping track of a single time value for each node set.

Chapter 5

Heuristic Algorithms for the ASP

We were able to solve the scheduling problem with the objective of minimizing the makespan in a polynomial number of iterations. But we have not given a complete algorithm for our main problem of Section 2.2 yet, which has an objective of minimizing the total weighted delay.

In Section 3, we mentioned that Tsitsiklis [12] showed that this problem can be solved to optimality in polynomial time. But, the proposed algorithm was not fast enough for practical use. In fact, as far as the Aircraft Sequencing Problem is concerned, an exactly optimal solution is not much better than a good suboptimal solution. One reason is that in the real world, the data we have is not exact either. For example, when we have a ready time for an arriving aircraft, this is usually our best estimate of when the aircraft will enter the terminal area plus some estimate of how long it will take the aircraft to reach the runway from there. But these are just estimates, and not exact values. In addition, many additional constraints have to be considered by the air traffic controller before fixing the schedules. Consequently, the exact optimal solution will almost never be possible to follow.

Aside from being fast, the practical algorithm has to be implementable in a dynamic fashion. This means that we need an algorithm that can make decisions with the information on hand as time goes by. These considerations and the complex nature of the scheduling problem have led us to thinking about heuristic ways to generate schedules that are possibly suboptimal but still "good schedules" with regard to the objective in mind.

In the following sections we shall develop two algorithms in detail, both of which can be used to dynamically sequence the aircraft.

5.1 The Greedy Heuristic

This is a very simple heuristic that makes use of the exact DP algorithm to minimize the total weighted delay in the absence of ready times. The decision instants are again the points in time when an aircraft starts using the runway. At every such instant, the heuristic looks at the set of aircraft that are already available to land or to take off and schedules them using the DP algorithm. Then, the first aircraft in the schedule will be the next aircraft to use the runway. At the moment this aircraft touches the runway, since some time has passed, the set of available aircraft has changed. Now, the heuristic schedules the new set of available aircraft, again by using the DP algorithm. This process is carried on until all aircraft have been scheduled.

A formal statement of the algorithm is as follows:

There are N aircraft types with k_0^i aircraft belonging to type i. The j^{th} aircraft of type i is C_i^j. Each aircraft has a ready time and the ready time of C_i^j is r_i^j. If an aircraft of type i was serviced last, then an aircraft of type j cannot use the runway until t_{ij} time units after the last operation started. At the beginning of the planning period, the last aircraft that was serviced is assumed to be of type L_0. The objective is to find a schedule for all aircraft such that the total weighted delay is minimized.

t : The current time

L : The type of the aircraft that was just serviced

$T = \sum_{i=1}^{N} k_0^i$;i.e. the total number of aircraft

$S \subset \{C_i^j : i = 1, 2, ..., N; j = 1, 2, ..., k_0^i\}$: The set of aircraft that are available to land or depart at the current time

$(G_1, G_2, ..., G_T)$: The final sequence generated by the Greedy Heuristic

Step 1) Set $t = 0$, $S = \emptyset$, $L_0 = L$, $a = 0$

Step 2) for $i = 1$ to $i = N$

for $j = 1$ to $j = k_0^i$

if $r_i^j \leq t$ then set $S = S \cup \{C_i^j\}$

end for

end for

Step 3) Apply the DP algorithm for minimizing the weighted delay in the absence of ready times using the aircraft in the set S. Suppose the algorithm generates the sequence $\left(C_{I(1)}^{J(1)}, C_{I(2)}^{J(2)}, ..., C_{I(|S|)}^{J(|S|)}\right)$.

Let $a = a + 1$

Step 4) Let $G_a = C_{I(1)}^{J(1)}$, $r_{I(1)}^{J(1)} = \infty$

Let $t = t + t_{L_o,I(1)}$

Let $L_0 = I(1)$

Let $S = S \backslash \{G_a\}$

Step 5) If $a = T$ stop, the schedule is complete; otherwise go back to Step 2.

We call this the greedy heuristic, because the runway never idles and never waits for a new aircraft when there are already other aircraft available, but chooses one of the aircraft whose ready times already passed to service next. However, this may not be optimal in many cases. For example, think of a situation where there are three classes of aircraft: small, medium and large. Also suppose that the minimum separation matrix is as follows:

leading\trailing	small	medium	large
small	75	75	72
medium	107	80	72
large	120	93	72

This means that after a small aircraft lands, 75 seconds have to pass before a medium aircraft can touch the runway. Now suppose that we have two aircraft to schedule. Suppose that the last serviced aircraft was a small size aircraft. The ready times and classes of the two aircraft to be scheduled are as follows:

Ready Times	0	1
Class	large	small

The greedy heuristic will look at the set of available aircraft at time 0, and schedule the large aircraft to land. This will occupy the runway for 72 seconds. By then, the small size aircraft will become available as well. Scheduling this aircraft will take another 120 seconds, so that the total time until both aircraft land will be 192 seconds. The total delay will be $72 + 191 = 263$. If we waited for 1 second instead and scheduled the small aircraft first, then this would take 75 seconds. Then, scheduling the large aircraft would take another 72 seconds, so the total time until both aircraft land will be 147 seconds. The total delay would be $74 + 147 = 221$ seconds. The system suffers a loss in this case, because the greedy heuristic was not patient enough to

wait 1 second for the small aircraft to arrive.

As stated, this algorithms seems like a static algorithm. However, it is clear to see that it can easily be used as a dynamic algorithm, since we can update the set of planes in our DP algorithm (the set of ready planes) at every decision epoch.

5.2 Heuristic with Time Windows

The exact dynamic programing formulation of the ASP with ready times takes into account all the aircraft that are scheduled to arrive in every iteration, but for this reason we have to include the time as a component of the state vector, and the state space becomes too large for us to carry out the DP recursions. On the other hand, the greedy algorithm takes only the currently available aircraft into account at every iteration so time can be left out of the state vector and the algorithm runs much faster. The downside is that the greedy nature of the algorithm disregards aircraft that are very soon to be available and may result in substantially suboptimal solutions. These are two extreme ways to handle the problem. This suggests an intermediate approach that incorporates the strengths of both algorithms. The result is the generation of the heuristic with time windows (HWTW).

The idea behind the HWTW is that at every iteration we take into account aircraft that are available at the current time as well as those that will be available within a certain time window into the future. This way, we avoid situations where the algorithm ignores aircraft that are to become available very soon. But the time window has to be chosen in a way such that time can still be left out of the state vector so that the running time of the algorithm is acceptable.

Basically, at every iteration, the algorithm takes into account a specified set of aircraft, schedules them using the DP recursions, and the first aircraft in this schedule is allowed to use the runway. After this operation is performed, the current time changes, a new set of candidate aircraft is generated and the algorithm continues. The set of aircraft that are taken into account at every iteration is chosen as follows: For each aircraft type, we compute the time it would take to start a movement of this type starting from the current time. This time may include the time we have to wait until an aircraft of the corresponding type will be available, or such an aircraft may already be available. The other consideration is the minimum separation time that has to pass to comply with safety regulations. Once we find this value for each type of aircraft, we take the minimum of them among all types and this becomes our time window. This

means that any aircraft that is already available or that will be available within the timeframe dictated by this window will be considered in the DP algorithm of this iteration. It may seem a little unclear why we chose such a window but the reason is quite intuitive. Our goal was to find a formulation where we did not have to include time as a component of the state vector. Looking at the definition of our window, one may notice that of whichever type the first aircraft in the optimal solution of the DP formulation for this subproblem turns out to be, after the first service, all the aircraft in our set are going to be available. To see this, note that we have chosen aircraft that will be available before $w = \min_i\{\max(r_i^{j^*}, t + t_{L,i})\}$. ($r_i^{j^*}$ is the ready time of the aircraft of type i that has the smallest ready time and that has not been serviced yet). After the first aircraft in the optimal solution of the DP at this stage, the time will be $\max(r_{i^*}^{j^*}, t + t_{L,i^*})$ for some i^*. However, since $w = \min_i\{\max(r_i^{j^*}, t + t_{L,i})\} \leq \max(r_{i^*}^{j^*}, t + t_{L,i^*})$ for any i^*, at this point, all the aircraft in the current DP formulation are available (otherwise, they would not be in the set since their ready time is after w). This means that after the first stage, we do not have to keep track of time anymore. As far as the first stage is concerned, we modify the incremental costs incurred at this stage to reflect the fact that some aircraft may not be ready yet. By doing this, we eliminate the need to use time as a component of the state vector in all stages. In fact, the modified incremental costs for the first stage will be precisely the values that we calculated at the beginning of the iteration to compute the window minus the current time. These are $\max(r_i^{j^*}, t + t_{L,i}) - t$ for each type

A formal description of the HWTW can be stated as follows:

There are N aircraft types with k_0^i aircraft belonging to type i. The j^{th} aircraft of type i is C_i^j. Each aircraft has a ready time and the ready time of C_i^j is r_i^j. If an aircraft of type i was serviced last, then an aircraft of type j cannot use the runway until t_{ij} time units have passed after the last movement started on the runway. At the beginning of the planning period, the last aircraft that was serviced is assumed to be of type L_0. The objective is to find a schedule for all aircraft such that the total weighted delay is minimized.

t : The current time

L : The type of the aircraft that was just serviced

$T = \sum_{i=1}^{N} k_0^i$, i.e. the total number of aircraft

$S \subset \{C_i^j : i = 1, 2, ..., N; j = 1, 2, ..., k_0^i\}$: The set of aircraft that are going to be available within our time window

$(G_1, G_2, ..., G_T)$: The final sequence generated by the HWTW

Step 1) Set $t = 0$, $S = \emptyset$, $L = L_0'$, $a = 0$

Step 2) for $i = 1$ to $i = N$

find $C_i^{j^*}$ such that $r_i^{j^*} = \min_{j=1,2,...,k_0^i} r_i^j$

end for

Step 3) Let $w = \min_i\{\max(r_i^{j^*}, t + t_{L,i})\}$

Step 4) for $i = 1$ to $i = N$

for $j = 1$ to $j = k_0^i$

if $r_i^j \leq w$ then set $S = S \cup \{C_i^j\}$

end for

end for

Step 5) Let $L_o = 0$, $t_{o,i} = \max(r_i^{j^*}, t + t_{L,i}) - t$. Solve the DP algorithm for minimizing the weighted delay without ready times using the aircraft in the set S. Suppose the algorithm generates the sequence $\left(C_{I(1)}^{J(1)}, C_{I(2)}^{J(2)}, ..., C_{I(|S|)}^{J(|S|)}\right)$

Step 6) Let $u = u + 1$

Let $G_a = C_{I(1)}^{J(1)}$, $r_{I(1)}^{J(1)} = \infty$

Let $t = t + \max(r_{I(1)}^{j^*}, t + t_{L,I(1)})$

Let $L = I(1)$

Let $S = S \backslash \{G_a\}$

Step 7) If $a = T$ stop, the schedule is complete; otherwise go back to Step 2.

The worst case running time of the whole algorithm is of the order $T * N^2 * (T/N + 1)^N$ which is equal to the worst case running time of the DP algorithm without ready times multiplied by the total number of aircraft. The quantity of interest is though, the worst case running time of one iteration of the algorithm, because an iteration of the algorithm corresponds to making a decision about what the next movement on the runway is going to be. The worst case running time of one iteration is $N^2 * (T/N + 1)^N$. But the worst case scenario is when all the aircraft become available around the same time, which is not something to worry about in the practical context. In our computational experiments, we have observed that for input streams that are likely to occur in the Air Traffic Control context, one iteration of the algorithm takes less than a second. Such a running time is certainly acceptable for a decision support tool to the controller. Also note that the algorithm can be run in a dynamic fashion, meaning that it does not require

all the data at the beginning. At each iteration, we only need to know the ready times and types of the aircraft that will become available within a short amount of time, typically a few minutes. As time passes we can take into account the newly arriving arrival and departure requests. Obviously, in this case the algorithm would not terminate after a specified number of aircraft but will be run throughout the time the runway is in operation.

Numerical results from this algorithm are given in Section 7.2.

Chapter 6

Algorithms with Constrained Position Shifting

6.1 Constrained Position Shifting

The HWTW addresses two important issues related to the practicality of the ASP algorithm. It can be run in a dynamic fashion and the running time is very fast. However, one can imagine scenarios where a certain aircraft is delayed indefinitely because new aircraft keep showing up that turn out to be selected to land prior to the aircraft already waiting. Aircraft belonging to a certain type may be favored or penalized because of particular separation requirements associated with that type. In other words, there are two more issues that we need to address: safety and fairness. Safety in the sense that no single aircraft should be delayed for too long and fairness in the sense that there should not be an excessive difference between the delays observed by different types of aircraft.

Dear [1] was the first to introduce the idea of Constrained Position Shifting (CPS) into the ASP to handle these two important concerns. He was working on the sequencing of arrivals only. The basic idea in CPS is that the absolute difference between the position of the aircraft in the First Come First Serve (FCFS) sequence and the position of the aircraft in the sequence generated by the algorithm should not exceed a Maximum Position Shift (MPS) number. For example, if an aircraft is in the 5^{th} position in the FCFS sequence and MPS is 1, this aircraft could only be the 4^{th}, 5^{th} or 6^{th} aircraft in the final sequence.

Our objective is to sequence arrivals and departures at the same time, so there are two

different ways to include the CPS constraints in the current context.

1. The first is to ignore the distinction between arrivals and departures as far as the MPS constraints are concerned, and use a single MPS number for both. As an example, if we have an arrival followed by a departure in the FCFS sequence and MPS is zero, the final sequence has to be identical to the FCFS sequence namely the arrival will be followed by the departure.

2. The second and probably the more natural way to incorporate the CPS idea is to consider arrivals and departures as different streams and make sure that MPS constraints are respected within each stream. Here, we can have two different MPS numbers, one for arrivals and the other for departures. If we consider the same example, with an arrival followed by a departure and both MPS values are equal to zero, then the final sequence 'arrival-departure' and the final sequence 'departure-arrival' satisfy the MPS constraints, because, the FCFS sequence for arrivals (or departures) is identical to the sequence for arrivals (or departures) in both of these sequences.

We have developed algorithms for both methods and numerical results for both are reported. From now on, when we talk about MPS numbers, we will give a single number for the first method (e.g. MPS=1) and a vector for the second method where the first component corresponds to the MPS value for arrivals and the second corresponds to MPS for departures (e.g. MPS=(1,2)). Now we proceed to the details of these two methods.

6.2 The Heuristics with Time Windows under CPS

6.2.1 Common MPS Value for Arrivals and Departures

Here, we have a single MPS number for arrivals and departures. We will modify the HWTW in such a way that the sequence generated will satisfy the MPS constraints in the sense of case 1 from the previous section. In fact, it is pretty straightforward to include these new constraints in the case where there are no ready times. Here, we only need to solve a Dynamic Program with the control space of the DP altered so that only sequences that satisfy the new requirements are allowed. The HWTW solves a DP at every iteration and assigns the first aircraft in the sequence generated by the DP to be the first aircraft to be allowed to use the runway. The

first and natural approach to model the MPS constraints in this case would be to limit the set of aircraft that can take the first position in the solution of the DP algorithm, since only this part of the DP solution will actually be implemented (the next aircraft will be determined by possibly taking some other aircraft into account as well). But, there is a problem with such an approach. If we make sure that at every iteration we satisfv the MPS constraints but do not consider the remaining aircraft, at some point the algorithm may have no choice that does not violate the constraints. The correct way to implement CPS is to ensure that the solution of every DP at every iteration satisfies the MPS constraints, not just the first aircraft. After a certain iteration, if no new aircraft is added to the set of aircraft with ready times within the window, for the set of the remaining aircraft, we know that a feasible solution with regard to the MPS constraints exists. Such a feasible solution is the sequence generated by the previous iteration, with the first aircraft in the sequence removed. If a new aircraft arrives and the set is expanded, this aircraft can be added to the end of the sequence from the previous DP to generate a feasible solution. This way, the algorithm will never generate a starting sequence that is not part of any final feasible solution, i.e. feasibility will be maintained throughout the algorithm. A formal description of the algorithm is (using the same notation as in HWTW):

Step 1) Set $t = 0$, $S = \emptyset$, $L = L_0^{'}$, $a = 0$, $M \equiv$ The number of maximum position shifts allowed.

Step 2) Let p_i^j be the position of aircraft C_i^j in the FCFS sequence.

Step 3) for $i = 1$ to $i = N$

find $C_i^{j^*}$ such that $r_i^{j^*} = \min_{j=1,2,...,k_0^i} r_i^j$

end for

Step 4) Let $w = \min_i\{\max(r_i^{j^*}, t + t_{L,i})\}$

Step 5) for $i = 1$ to $i = N$

for $j = 1$ to $j = k_0^i$

if $r_i^j \leq w$ then set $S = S \cup \{C_i^j\}$

end for

end for

Step 6) Let $L_o = 0$, $t_{o,i} = \max(r_i^{j^*}, t + t_{L,i}) - t$.

Let $y_u = \left|S \cap \{C_u^j : u = 1, 2, ..., N; j = 1, 2, .., k_0^u\right|$ (The number of aircraft of type u that are in the set S).

Solve the DP algorithm for minimizing the weighted delay without ready times using the aircraft in the set S. In the course of the algorithm, when considering a state $x = (t, L, k^1, k^2, ..., k^N)$ and a control $u \in U(x)$, where $U(x) = \{y \in \{1, 2, ..., N\} : k^y > 0\}$; Let $p = p_u^{j^*} + y_u - k^y$. (The position of the aircraft in question in the FCFS sequence). Let $p' = a + 1 + |S| - \sum_{i=1}^{N} k_i$ (The position of the aircraft in the final schedule if we make the decision u).

If $\left|p - p'\right| > M$, this control is infeasible.

Suppose the algorithm generates the sequence $\left(C_{I(1)}^{J(1)}, C_{I(2)}^{J(2)}, ..., C_{I(|S|)}^{J(|S|)}\right)$

Step 7) Let $a = a + 1$

Let $G_a = C_{I(1)}^{J(1)}$, $r_{I(1)}^{J(1)} = \infty$

Let $t = t + \max(r_{I(1)}^{j^*}, t + t_{L,I(1)})$

Let $L = I(1)$

Let $S = S \backslash \{G_a\}$

Step 8) If $a = T$ stop, the schedule is complete; otherwise go back to Step 2.

Notice that this algorithm is very similar to the HWTW except that at every state control pair that may be feasible, we have to make an additional check on the number of position shifts. In the worst case, these checks will add another multiple of T to the running time to see which aircraft is actually the next aircraft of a certain type, but a little care in bookkeeping reduces the additional burden to be a much smaller constant than T.

6.2.2 Different MPS Values for Arrivals and Departures

The need for including MPS constraints arose from practical concerns. One concern was that it may be difficult for the controllers to make drastic changes in the FCFS sequence, especially of arrivals, since such a situation would require overtakes among aircraft. The other concern is that a certain aircraft may find it unfair if an aircraft that was 10 positions behind that aircraft in the FCFS sequence is allowed to use the runway prior to that aircraft. Swapping an arrival and a departure in their FCFS sequence does not involve any overtakes. In addition, departures may accept priority being given to arrivals because of the safety considerations involved. As a consequence, we have included a new type of MPS constraint in the model, namely if an aircraft is an arrival, we keep track of its position among arrivals and enforce the MPS constraints within arrivals (true for departures as well). In cases where the arrivals may be sensitive to departures using the runway before them, the controller may choose to adjust the weights appropriately to

give priority back to arrivals.

The new algorithm is:

Step 1) Set $t = 0$, $S = \emptyset$, $S_A = \emptyset$, $S_D = \emptyset$, $L = L_0'$, $a_A = 0$, $a_D = 0, a = 0, M_A \equiv$The number of maximum position shifts allowed among arrivals, $M_D \equiv$the number of maximum position shifts allowed among departures

Let $A = \{i$:type i is an arrival$\}$

Let $D = \{i$: type i is a departure$\}$

Step 2) If C_i^j is an arrival

Let p_i^j be the position of aircraft C_i^j in the FCFS sequence of arrivals

If C_i^j is a departure

Let p_i^j be the position of aircraft C_i^j in the FCFS sequence of departures

Step 3) for $i = 1$ to $i = N$

find $C_i^{j^*}$ such that $r_i^{j^*} = \min_{j=1,2,...,k_0^i} r_i^j$

end for

Step 4) Let $w = \min_i\{\max(r_i^{j^*}, t + t_{L,i})\}$

Step 5) for $i = 1$ to $i = N$

for $j = 1$ to $j = k_0^i$

if $r_i^j \leq w$ then set $S = S \cup \{C_i^j\}$

If $C_i^{j^*}$ is an arrival

Let $S_A = S_A \cup \{C_i^j\}$

If $C_i^{j^*}$ is a departure

$S_D = S_D \cup \{C_i^j\}$

end for

end for

Step 6) Let $L_o = 0$, $t_{o,i} = \max(r_i^{j^*}, t + t_{L,i}) - t$.

Let $y_u = \left|S \cap \{C_u^j : u = 1, 2, ..., N; j = 1, 2, .., k_0^u\right|$ (The number of aircraft of type u that are in the set S).

Solve the DP algorithm for minimizing the weighted delay without ready times using the aircraft in the set S. In the course of the algorithm, when considering a state $x = (t, L, k^1, k^2, ..., k^N)$ and a control $u \in U(x)$, where $U(x) = \{y \in \{1, 2, ..., N\} : k^y > 0\}$, let $p = p_u^{j^*} + y_u - k^u$. (The position of the aircraft in question in the FCFS sequence).

If $C_i^{j^*}$ is an arrival

Let $p' = a_A + 1 + |S_A| - \sum_{i \in A} k^i$ (The position of the aircraft in the final schedule if we make the decision u).

If $\left|p - p'\right| > M_A$, this control is infeasible.

If $C_i^{j^*}$ is a departure

Let $p' = a_D + 1 + |S_D| - \sum_{i \in D} k^i$ (The position of the aircraft in the final schedule if we make the decision u).

If $\left|p - p'\right| > M_D$, this control is infeasible.

Suppose the algorithm generates the sequence $\left(C_{I(1)}^{J(1)}, C_{I(2)}^{J(2)}, ..., C_{I(|S|)}^{J(|S|)}\right)$

Step 7) Let $a = a + 1$

Let $G_a = C_{I(1)}^{J(1)}$, $r_{I(1)}^{J(1)} = \infty$

Let $t = t + \max(r_{I(1)}^{j^*}, t + t_{L,I(1)})$

Let $L = I(1)$

Let $S = S \backslash \{G_a\}$

If $C_{I(1)}^{J(1)}$ is an arrival

Let $a_A = a_A + 1$

If $C_{I(1)}^{J(1)}$ is an arrival

Let $a_D = a_D + 1$

Step 8) If $a = T$ stop, the schedule is complete; otherwise go back to Step 2.

This algorithm requires a little more computation than the previous one, since now we have to keep track of positions among arrivals and among departures.

6.3 The Triangular Inequality

All the algorithms that we have discussed have assumed that the triangular inequality holds among the minimum separation requirements, i.e. $t_{i,j} + t_{j,k} \geq t_{i,k}$ for all i, j, k. However, even though these requirements do hold among arrivals and among departures, they fail to hold when we consider arrivals and departures together, as we have mentioned in Section 2.1. This creates a problem in the practical implementation of the algorithms. If we have two arrivals that have a certain minimum separation requirement, the same minimum time should still be between them even if the algorithm puts a departure in between. To overcome this difficulty, we modify the result that the algorithms generate in a way so that all the separation requirements will be

satisfied. In particular, we take the schedule that the algorithms generate and in cases where a separation requirement is violated, we stretch the time between the corresponding movements sufficiently to meet the requirements, while keeping the same sequence. The way to do this is to look at both the last arrival and the last departure and take the maximum of the two requirements. Since we are doing this after the sequence was generated and do not take this into account in the DP algorithms, this adds to the heuristic character to the algorithms. The hope however, is that such a case does not happen too frequently, so that the "stretched" schedule is still a reasonably good schedule. In our numerical results, we found this to be the case.

The concept of keeping track of both the last arrival and the last departure can be included in the DP algorithm as well. We would just need to enhance the state vector. The computational burden would increase and we are not sure whether the running time would still be acceptable or whether the additional benefit would be worth the extra effort, since we have not investigated this matter thoroughly. Our numerical results suggest that the benefits would not be substantial since the results after and before the "stretching" are not drastically different.

Chapter 7

Results

We started with the objective of investigating the sequencing of terminal area traffic and demonstrating the potential benefits and the trade-offs involved. With this objective in mind, we have developed several algorithms that might help us better utilize the capacity of the airport. In this section, numerical results indicating the performance of these algorithms will be reported.

7.1 Algorithms to be Compared

We have implemented four different ways of sequencing terminal area air traffic, to give an idea of how the current operation's performance compares with the performance of our algorithms .

7.1.1 First Come First Serve

This sequencing scheme is pretty self explanatory: the aircraft use the runway in increasing order of their ready times. We do not distinguish between arrivals and departures, we simply schedule an aircraft with an earlier ready time before an aircraft with a later ready time. Of course we make sure that all the separation requirements are satisfied. We understand that this may not reflect the current operations well, because under current practice, arrivals typically receive priority over departures. Nevertheless, we find it useful to report the results of this simplest of sequencing schemes.

7.1.2 FITG (Fill in the Gaps)

The main idea in this algorithm is that arrivals have priority over departures. Basically, a departure is not allowed to take off, if it is going to slow down any arrival. In some sense departures are scheduled into the gaps between the arrivals, hence the name. The exact description is: At every stage of the algorithm, we schedule a movement. If the aircraft with the smallest ready time is an arrival, it is allowed to land. If it is a departure, we compute the earliest time this operation can take place and add to this the minimum separation requirement that will need to be enforced if an arrival is to land after this departure. If there is any arrival that has a ready time smaller than this number that we computed, we hold the departure and let the arrival use the runway first. If not, the departure takes off and any arrival approaching the airport will not have to be slowed down. The algorithm proceeds the same way after each operation. The FCFS sequence among arrivals and among departures is maintained.

7.1.3 FITG2

This algorithm is a modified version of FITG. The difference is that we allow departures to take off if there is a backlog of them, even if we will have to slow down an arrival. More precisely, the procedure is the same as the above except that we shall schedule a departure if there are more than 10 departures and less than or equal to 5 arrivals waiting to use the runway, regardless of whether the departure will slow down an arrival or not. This algorithm is a more realistic version of FITG. Still, aircraft are scheduled in their FCFS order among arrivals and among departures.

7.1.4 ALTERNATE

A careful look at the minimum separation matrix reveals that the values corresponding to arrivals followed by departures and departures followed by arrivals are generally smaller than arrival-arrival and departure-departure values. Experienced controllers sometimes take advantage of this fact by trying to alternate between arrivals and departures. This is the main idea behind this sequencing scheme. Here, the algorithm takes turns between arrivals and departures. If the last operation was an arrival and there is at least one departure ready to take off, the departure with the lowest ready time is allowed to depart. If there is no departure ready to take off, the aircraft that has the lowest ready time is scheduled (whether it is a departure or not). A similar

procedure is applied when it is the arrivals' turn, i.e. when the last operation was a departure. The FCFS sequence among arrivals and among departures is maintained.

7.2 Computational Results

To study how our algorithms perform, we have implemented them and the four scheduling policies described above using C++. We have generated randomly 30 different arrival and departure streams to test. Both the arrival and departure streams were generated from a nonhomogeneous Poisson process and they reflect a period of three hours. The rate function of the arrival and departure streams are the same and is depicted in Figure 7-1. We tried to mimic a period of time during which demand for the airport increases, stays high for a while and then decreases again. Note that we have generated arrivals and departures using the same Poisson model, so when the rate is high, the airport is faced with approximately 64 movements per hour.

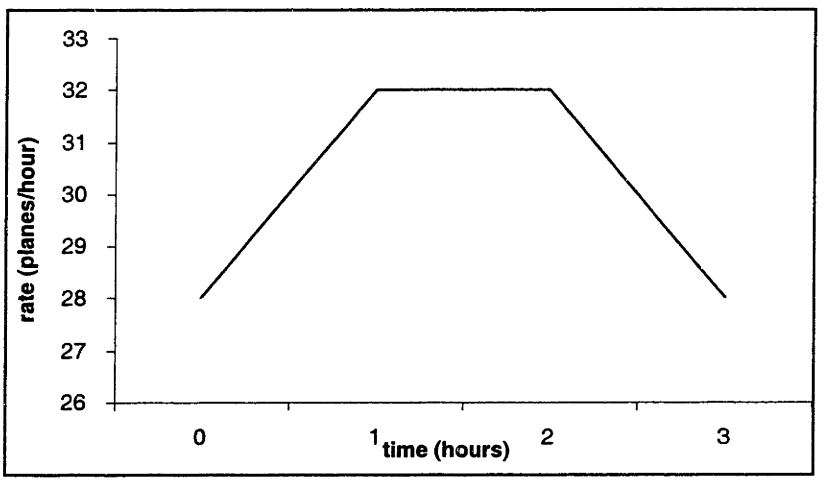

Figure 7-1: The Arrival Rate Used in the Nonhomogeneous Poisson Process (Same for Departures)

Each arrival and each departure was randomly chosen to be of a certain type. The probabilities for different types reflect the aircraft mix of the airport. The aircraft mix is shown in Table 7.1.

We have used three different sets of weights for different aircraft types to reflect different performance measures that may be of concern. First we set all the weights equal to one. This would correspond to measuring the total delay incurred by all the aircraft. Second, we used

Heavy	15%
Large	40%
Medium	35%
Small	10%

Table 7.1: Aircraft Mix

approximate values reflecting the number of passengers in each type. This would correspond to measuring the total delay incurred by all passengers.. The last set of weights we used is an approximate dollar value of operating the corresponding aircraft type. We have typical values for each type and the "on ground" values are taken to be 75% of the corresponding "in air" values. These different values are depicted in Tables 7.2 and 7.3.

	Aircraft Delay	Passanger Delay	Operating Cost
Heavy	1	300	4800
Large	1	150	1800
Medium	1	40	900
Small	1	4	240

Table 7.2: Weights for Arrivals

	Aircraft Delay	Passenger Delay	Operating Cost
Heavy	1	300	3600
Large	1	150	1380
Medium	1	40	660
Small	1	4	180

Table 7.3: Weights for Departures

As mentioned before, we have generated 30 different instances. We have run the following algorithms for all 30 instances.

FCFS: First Come First Serve

FITG: Fill in the Gaps (Schedule departures when there is a gap between arrivals)

FITG2: Fill in the Gaps 2 (Schedule departures when there is a gap between arrivals except when there are more than 10 departures waiting)

ALTERNATE: Alternate between arrivals and departures

HWTW: Heuristic With Time Windows (Without CPS)

MPS=X: Heuristic With Time Windows With CPS (CPS that does not distinguish between arrivals and departures, Maximum Position Shifts allowed is X). We have used $X = 1$ and

$X = 2$.

MPS=(X, Y): Heuristic With Time Windows With CPS (CPS within arrivals and within departures, Maximum Position Shifts allowed for arrivals is X, for departures is Y). We have used

$(X, Y) = (0, 0), (0, 1), (1, 0), (1, 1), (2, 0), (0, 2), (1, 2), (2, 1), (2, 2)$.

Note that when we use the last case with $(X, Y) = (0, 0)$, this means that the algorithm keeps the FCFS sequence among arrivals and among departures but combines these two sequences in a way that reduces delays.

7.2.1 Aircraft Delay

Table 7.4. gives the summary of the results that we obtained from running the algorithms on the 30 test problems when the Aircraft Delay weights are used, i.e. all aircraft are weighted equally. All the numbers are the average of the results obtained in the 30 problems. Column I of Table 7.4 reports the average of the Normalized Weighted Total Delays. Normalized weighted total delay means that we take the weighted total delay and divide it by the sum of all the weights, i.e. $\frac{\sum_{i=1}^{T} \omega_{\lambda(i)} \cdot d_i^{\lambda}}{\sum_{i=1}^{T} \omega_{\lambda(i)}}$ where λ is the sequencing algorithm used, ω are the weights and d are the delays incurred by the T aircraft. We have used such a normalization to avoid a disproportional contribution of test cases with a large number of aircraft to the average. In this case, since all the values are equal to 1, the normalized total weighted delay is the average delay incurred by the aircraft. Column II gives the percentage improvement of average delay over the delay in FCFS,. i.e. $\frac{(FCFS-ALGORITHM)}{ALGORITHM}$. A negative value means that the corresponding algorithm resulted in a higher mean delay than FCFS. Column III gives the average number of aircraft in arrival or departure strings. Basically, we first count the number of times the sequence switches from serving arrivals to serving departures or from departures to arrivals and divide the total number of aircraft by this number. The values obtained are reported in Column III. For example, if the sequence consisted of an equal number of arrivals and departures and if arrivals and departures were alternated at every step, this value would be equal to 1.

Figures 7-2, 7-3 and 7-4 show the same results graphically.

Another interesting statistic is the distribution of delay among different aircraft types. Figure 7-5 gives pie charts showing the proportion of total delay coming from different aircraft categories, for four of the algorithms. On the legend, the first 4 types are arrivals, the last 4 are

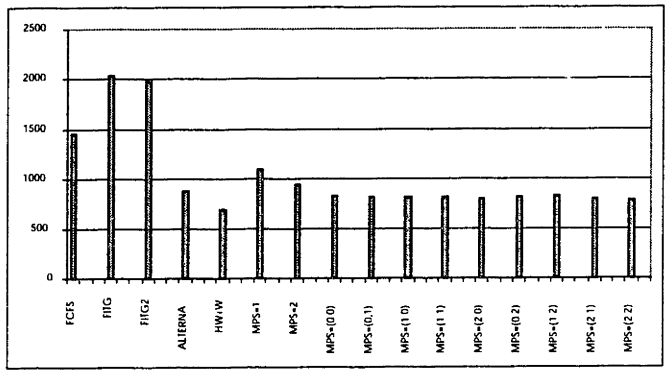

Figure 7-2: Normalized Total Weighted Delay (Aircraft Delay Case)

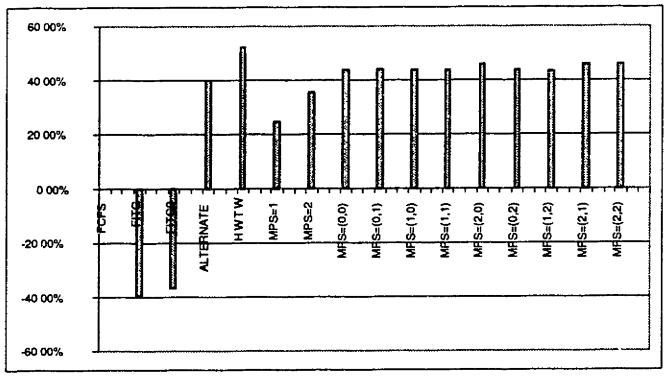

Figure 7-3: Improvement over FCFS (Aircraft Delay Case)

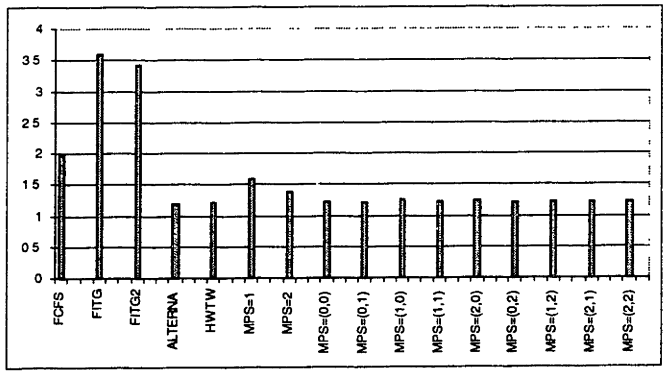

Figure 7-4: Average Length of Arrival and Departure Strings (Aircraft Delay Case)

	Column I	Column II	Column III
FCFS	1457.85	0.00%	1.982
FITG	2033.32	-39.47%	3.591
FITG2	1992.15	-36.65%	3.428
ALTERNATE	878.08	39.77%	1.191
HWTW	696.65	52.21%	1.210
MPS=1	1098.42	24.65%	1.592
MPS=2	939.66	35.54%	1.391
MPS=(0,0)	821.39	43.66%	1.229
MPS=(0,1)	819.37	43.80%	1.213
MPS=(1,0)	819.65	43.78%	1.253
MPS=(1,1)	820.67	43.71%	1.228
MPS=(2,0)	791.73	45.69%	1.241
MPS=(0,2)	820.30	43.73%	1.212
MPS=(1,2)	826.28	43.32%	1.219
MPS=(2,1)	790.34	45.79%	1.225
MPS=(2,2)	789.21	45.87%	1.215
Column I: Normalized Total Weighted Delay			
Column II: Percentage Improvement Over FCFS			
Column III: Average Number of aircraft in Arrival/Departure Strings			

Table 7.4: Summary of Results for Aircraft Delay Case

departures. (i.e., the top box represents arrivals of type H, the fifth box represents departures of type H).

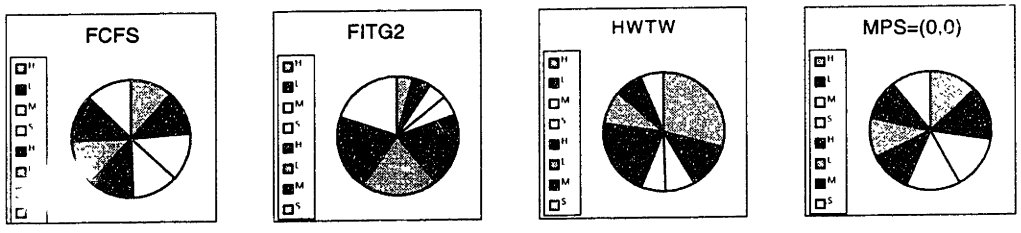

Figure 7-5: Distribution of Delay Among Aircraft Types (Aircraft Delay Case)

7.2.2 Passenger Delay

In this section, we report the results obtained from the experiments using the weights for the passenger delay objective function. (See Tables 7.2 and 7.3 for passenger delay weights). This means that the algorithm tried to minimize the average passenger delay.

	Column I	Column II	Column III
FCFS	1436.51	0.00%	1.982
FITG	2007.31	-39.74%	3.591
FITG2	1975.14	-37.50%	3.428
ALTERNATE	863.45	39.89%	1.191
HWTW	348.45	75.77%	1.369
MPS=1	1090.36	24.10%	1.727
MPS=2	1011.02	29.62%	1.539
MPS=(0,0)	815.43	43.24%	1.388
MPS=(0,1)	813.95	43.34%	1.367
MPS=(1,0)	828.93	42.30%	1.390
MPS=(1,1)	828.78	42.31%	1.361
MPS=(2,0)	793.27	44.78%	1.392
MPS=(0,2)	796.02	44.59%	1.347
MPS=(1,2)	809.04	43.68%	1.356
MPS=(2,1)	779.85	45.71%	1.367
MPS=(2,2)	765.65	46.70%	1.348
Column I: Normalized Total Weighted Delay			
Column II: Percentage Improvement Over FCFS			
Column III: Average Number of aircraft in Arrival/Departure Strings			

Table 7.5: Summary of Results for Passenger Delay Case

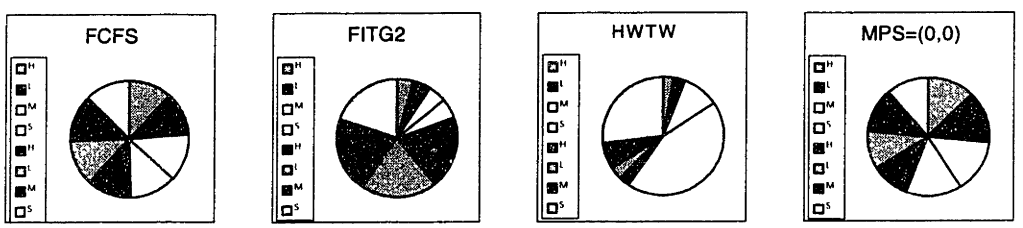

Figure 7-6: Distribution of Delay Among Aircraft Types (Passenger Delay Case)

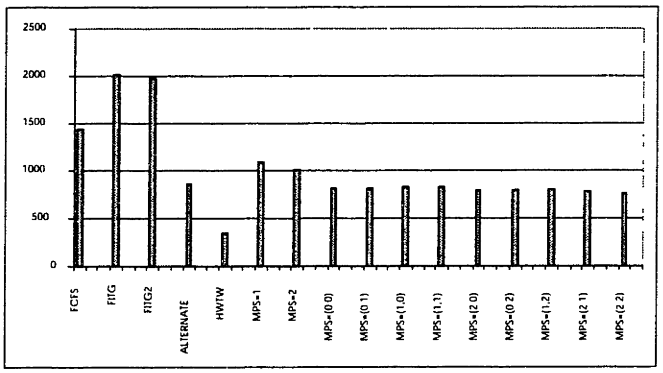

Figure 7-7: Normalized Weighted Total Delay (Passenger Delay Case)

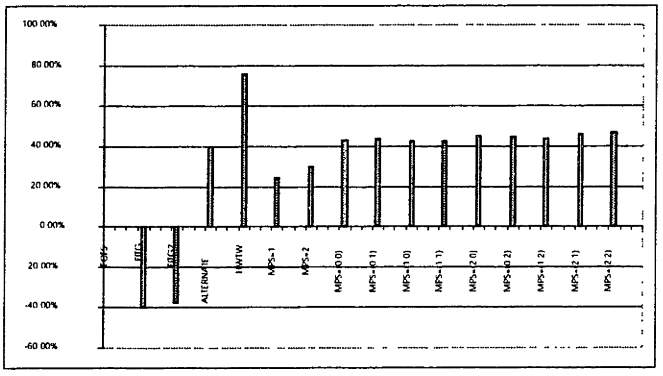

Figure 7-8: Percentage Improvement Over FCFS (Passenger Delay Case)

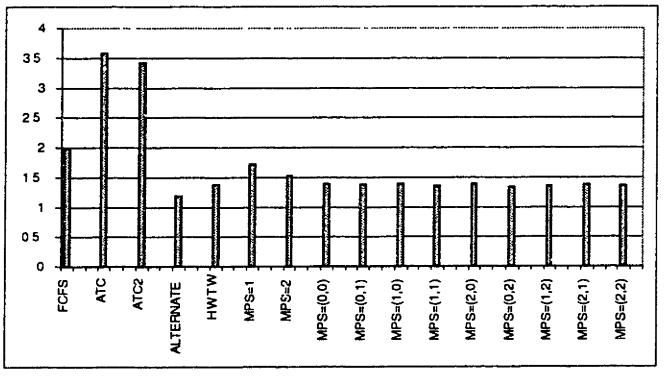

Figure 7-9: Average Length of Arrival or Departure Strings (Passenger Delay Case)

7.2.3 Operating Cost

In this section, we report the results obtained from the experiments using the weights for the operating cost objective function. (See Tables 7.2 and 7.3 for passenger delay weights). In this case, the algorithm tried to minimize the average operating costs of the aircraft.

	Column I	Column II	Column III
FCFS	1436.76	0.00%	1.982
FITG	1765.13	-22.85%	3.591
FITG2	1799.48	-22.25%	3.428
ALTERNATE	884.31	38.45%	1.191
HWTW	423.14	70.55%	1.301
MPS=1	1087.70	24.30%	1.701
MPS=2	1001.44	30.30%	1.509
MPS=(0,0)	810.06	43.62%	1.360
MPS=(0,1)	810.43	43.59%	1.344
MPS=(1,0)	826.65	42.46%	1.348
MPS=(1,1)	828.06	42.37%	1.337
MPS=(2,0)	786.62	45.25%	1.335
MPS=(0,2)	785.70	45.31%	1.337
MPS=(1,2)	818.11	43.06%	1.323
MPS=(2,1)	770.51	46.37%	1.325
MPS=(2,2)	753.24	47.57%	1.317
Column I: Normalized Total Weighted Delay			
Column II: Percentage Improvement Over FCFS			
Column III: Average Number of aircraft in Arrival/Departure Strings			

Table 7.6: Summary of Results for Operating Cost Case

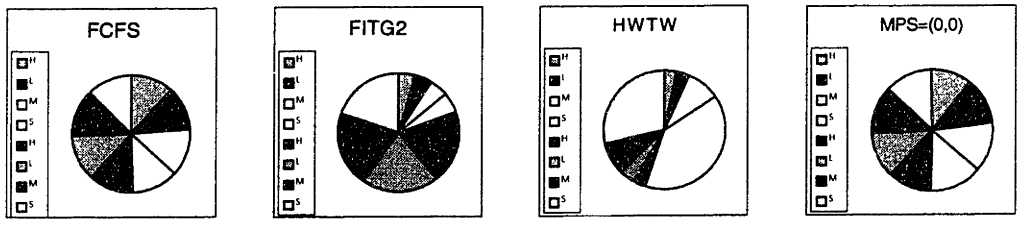

Figure 7-10: Distribution of Delay Among Aircraft Types (Operating Cost Case)

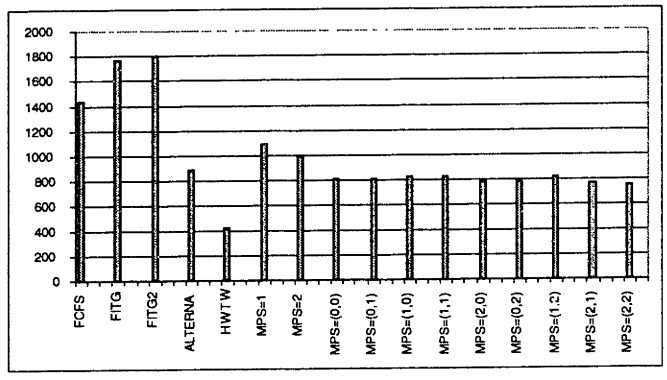

Figure 7-11: Normalized Total Weighted Delay (Operating Cost Case)

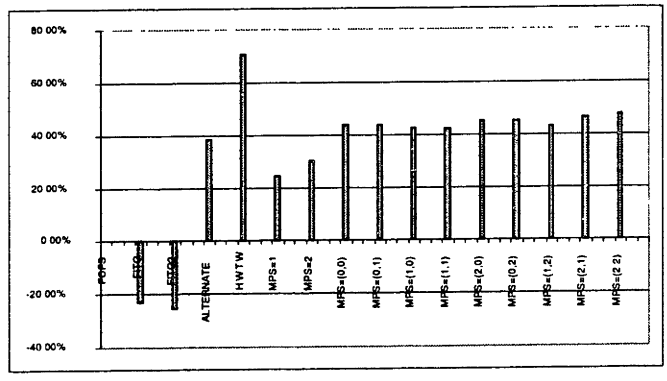

Figure 7-12: Percentage Improvement Over FCFS (Operating Cost Case)

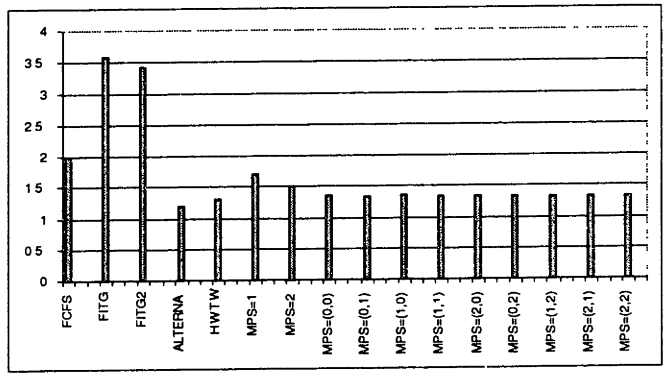

Figure 7-13: Average Length of Arrival or Departure Strings (Operating Cost Case)

7.2.4 Running Time

To qualify as real time performance, our algorithms have to run very fast. As mentioned before, even though the controllers would like an immediate response, a 5-10 sec. running time may qualify as real time performance [10]. The running time of our algorithms is reported in Table 7.7 (with Aircraft Delay Weights). The algorithms were programmed in C++ and were run on a Pentium 400 running Red Hat Linux.

	Average (secs)	Maximum (secs)
HWTW	0.03	1.31
MPS=1	0.65	5.75
MPS=2	0.38	5.94
MPS=(0,0)	0.18	4.11
MPS=(0,1)	0.20	5.07
MPS=(1,0)	0.20	6.94
MPS=(1,1)	0.20	5.92
MPS=(2,0)	0.16	4.50
MPS=(0,2)	0.19	5.40
MPS=(1,2)	0.19	5.15
MPS=(2,1)	0.17	4.99
MPS=(2,2)	0.16	5.02

Table 7.7: Running Time

The times reported are the running time of an iteration of the algorithm, which is of relevance since every time the controller has to make a decision, this will be the time he/she would have to wait for. To speed up the algorithms, we have used a limit on the number of aircraft that can be taken into account every time the algorithm calls the dynamic program. This number was chosen to be 19. There is nothing special about this number, we chose it because it resulted at an acceptable running time and increasing this number beyond 19 did not improve the objective value substantially. Depending on the computational power at hand, this number can be increased. The way to do this is to experiment with some instances of the problem to find the highest number that will result in an acceptable running time. However, our computational experience indicates that after a point, the additional benefits of increasing this limit are marginal.

7.3 Discussion of the Results

The computational results clearly show that there is a significant potential benefit in the use of optimization for sequencing arrivals and departures in the terminal area. The FITG and FITG2 (departures scheduled when there is a gap) policies perform very poorly and even FCFS had a better performance than these. The ALTERNATE policy is quite good, which was to be expected due to the nature of the minimum separation matrix. The HWTW is the best in terms of minimizing total weighted delay. Depending on the weights used, the improvement over FCFS was 50%-75%. This was to be expected since HWTW incorporated no maximum position shift constraints. This very fact, however, makes the algorithm impractical. The algorithms that do not distinguish between arrivals and departures when enforcing MPS constraints (common MPS value) did not perform very well, but as noted before, the more realistic version of MPS constraints is when we enforce them separately among arrivals and among departures. These types of policies, even when $MPS = (0, 0)$, performed very well. $MPS = (0, 0)$ resulted in approximately a 43% improvement over FCFS in all three weight sets and gave 5%-8% better results than the simple ALTERNATE policy. In general, we can see that increasing the MPS values results in delay reductions, however since the algorithm is a dynamic algorithm and is not necessarily optimal, in some cases increasing the MPS values resulted in slightly higher delays.

There is another statistic to pay attention to. This is the average length of the arrival or departure strings number that was reported. Notice the similar shapes of Figure 7-2 and Figure 7-4. Essentially, there is a strong relationship between lower weighted delay numbers and shorter arrival/departure strings. This shows that a significant portion of the improvements may be due to the lower minimum separation requirements between arrivals and departures. This means that if we want to fully realize the benefits that are demonstrated in this thesis, the resulting sequences would switch between arrivals and departures more than the sequences produced by FCFS or FITG types of policies. Changing from arrivals to departures and vice versa may mean a more complex calculation for the controller in terms of the maneuvers required, so an automated decision aid in that area may complement the ideas in this research. On the other hand, we cannot attribute all the savings to more changes between arrival and departure strings. For example, $MPS = (0, 0)$ has less frequent changes between arrivals and departures than ALTERNATE and still performs 5%-8% better.

The distribution of the delay among different aircraft categories under different policies and

weights is important as well. We have seen that FCFS is very fair in this respect. FITG2 favors arrivals considerably and departures face long delays. HWTW has no MPS constraints, so it is not very fair towards different categories either. When the aircraft delay objective is used, the heavy aircraft are penalized. This is can be expected since the separation requirements when following a heavy aircraft are significantly stricter whereas the weights used do not give any more importance to a wide-body commercial jet (H) than a general aviation aircraft (S). Conversely, the passenger delay and the operating cost objectives put the burden on small aircraft, since the penalty of delaying small aircraft is much smaller. On the other hand, $MPS = (0, 0)$ is very fair throughout. It is worth noting that this algorithm can both achieve significant decreases in total weighted delay and be fair to different aircraft types.

As far as the running time is concerned, it can be seen in Table 7.7 that the algorithms we propose are fast enough for real time use. The average running times are so small that most of the time the controller can get an immediate response. Even the maximum values are a few seconds. For example, the maximum running time of an iteration for $MPS = (0, 0)$ in the 30 instances that we generated was 4.11 seconds.

Chapter 8

Conclusions

In this thesis, we have investigated the Aircraft Sequencing Problem with Arrivals and Departures. Both the theoretical aspects of the problem and some practical issues were discussed. Algorithms for sequencing arrivals and departures on a single runway were developed, while keeping in mind those practical issues. Computational results from these algorithms were given.

The static version of the problem is basically a scheduling problem with sequence dependent processing times and ready times, with the objective of minimizing total weighted delay. Exact algorithms for this problem are not sufficiently fast for a practical implementation. We have given several algorithms that can be used both for the static and the dynamic versions of the problem. These algorithms are not exact solutions, however they are much faster than an exact algorithm and address some very important practical issues related to the ASP.

The computational results demonstrate that the potential benefits of using optimization in the sequencing of arrivals and departures in the Terminal Area are fairly significant. For example, the algorithm HWTW with $MPS = (0, 0)$ reduces delays by 40% compared to FCFS. This algorithm keeps the FCFS sequence among arrivals and among departures. So, even by simply intertwining arrivals and departures in a clever way, the runway capacity can be utilized much more efficiently. This particular algorithm is also very fair to all types of aircraft.

The control of arrivals and departures in the terminal area of an airport is a very complex task. The sequencing is only one side of the coin. Determining the paths that the aircraft should take, while avoiding a close encounter of multiple aircraft is both a difficult and a very important task. Hence, it is understandable that many will be willing to sacrifice some efficiency in order to keep things simple for the controllers. However, the traffic at major airports is

increasing every day. We may soon find ourselves in a situation where we shall not be able to disregard any opportunities to improve on the existing situation. To actually realize any improvements through advanced sequencing techniques, an automated decision aid tool has to be designed. This tool has to help the controllers in both the sequencing of aircraft and also in other decisions such as determining conflict free paths. The tool has to be very user friendly and reliable. Precise procedures have to be designed to dictate the proper and safe use of this automated tool. Controllers have to be trained and the system needs to be tested by detailed simulation experiments involving humans. Hence, the implementation of an automated tool of this kind is certainly a challenging task. However we believe that the potential benefits that we demonstrated are significant enough and they at least show the need for a serious discussion about the possible use of an advanced sequencer in the terminal area.

A possible extension to this research is to model the ASP with multiple runways. This would require runway allocation decisions to be made in conjunction with sequencing decisions. One could also try to bring the model closer to reality by investigating and modeling the interaction between the controllers and the pilots, the ways an aircraft is delayed, the maneuvers that are executed in the terminal area etc. Another possible extension would be to try to include some probabilistic modeling while developing dynamic algorithms, such as having a probabilistic model of future aircraft types and arrival times.

Bibliography

[1] Dear, R.G.,"*The Dynamic Scheduling of Aircraft in the Near Terminal Area*," Ph.D. Thesis, Flight Transportation Laboratory Report R76-9, M.I.T., Cambridge, MA (1976)

[2] Psaraftis, H. N.,"*A Dynamic Programming Approach to the Aircraft Sequencing Problem*," Flight Transportation Laboratory Report R78-4, M.I.T., Cambridge, MA (1978)

[3] Psaraftis, H.N., "*A Dynamic Programming Approach for Sequencing Groups of Identical Jobs*," Operations Research, Vol. 28, No. 6, 1347-1359 (1980)

[4] Erzberger, H., "*Design Principles and Algorithms for Automated Air Traffic Management*," NASA Ames Center Technical Report, LS-200, (1995)

[5] Bianco, L., Dell'Olmo, P. and Giordani, S., "*Minimizing total completion time subject to release dates and sequence-dependent processing times*," Annals of Operations Research, 86: 393-415 (1999)

[6] Balas, E. and Toth, P.,"Branch and Bound Methods. In *The Traveling Salesman Problem* (Edited by E.L. Lawler, J.K. Lenstra, A.H.G. Rinooy Kan and D.B. Shmoys)," pp. 361-401 (1985)

[7] Reinelt, G.,"*The Traveling Salesman Problem, Computational Solutions for TSP Applications*," Springer-Verlag, Berlin Heidelberg (1994)

[8] Gilbo, E. P.,"*Airport Capacity: Representation, Estimation, Optimization*," IEEE Transactions on Control Systems Technology, Vol.1, No.3, (1993)

[9] Gilbo, E. P.,"*Optimization of Air Traffic Management Strategies at Airports with Uncertainty in Airport Capacity*," IFAC Transportation Systems, Chania, Greece (1997)

[10] Erzberger, H.*"Design Principles and Algorithms for Automated Air Traffic Management,"* NASA Report LS-200, (1995)

[11] Odoni, A.R., Rousseau, J. M., Wilson, N. H. M.,*"Models in Urban and Air Transportation,"* Handbooks in OR & MS, Vol. 6, Elsevier Science B.V. (1994)

[12] Tsitsiklis, J. N.,*"Special Cases of Traveling Salesman and Repairman Problems with Time Windows,"* Networks, vol.22, no.3, 263-282 (1992)

[13] Venkatakrishnan, C.S., Barnett, A. and Odoni, A.R., *"Landings at Logan Airport: Describing and Increasing Airport Capacity,"* Transportations Science, Vol. 27, No.3, (1993)

THESIS PROCESSING SLIP

FIXED FIELD: ill. _____ name _____

index _____ biblio _____

▶ COPIES: (Archives) Aero (Dewey) Eng Hum
Lindgren Music Rotch Science

TITLE VARIES: ▶☐ _____

NAME VARIES: ▶☐ _____

IMPRINT: (COPYRIGHT) _____

▶ COLLATION: 58 l

▶ ADD: DEGREE: _____ ▶ DEPT.: _____

SUPERVISORS: _____

NOTES:

cat'r: date:

page:

▶ DEPT: O.R. F58

▶ YEAR: 2000 ▶ DEGREE: S.M.

▶ NAME: MUHARREMOGLU, Alp