GROUND HOLDING STRATEGIES
FOR AIR TRAFFIC CONTROL

by

Mostafa Terrab

Ingénieur des Ponts et Chaussées (1979)
M.S., Massachusetts Institute of Technology (1982)

SUBMITTED TO THE DEPARTMENT OF
CIVIL ENGINEERING
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February, 1990
© Mostafa Terrab

The author hereby grants to MIT permission to reproduce and to distribute copies
of this thesis document in whole or in part.

Signature of Author......................

Department of Civil Engineering
February, 1990

Certified by.................................................................

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

Accepted by.................................

Ole S. Madsen
Chairman, Department Committee on Graduate Studies

MASSACHUSETTS INSTITUTE
OF TECHNOLOGY

JUN 15 1990

LIBRARIES
ARCHIVES
GROUND HOLDING STRATEGIES
FOR AIR TRAFFIC CONTROL

by

Mostafa Terrab

Submitted to the Department of Civil Engineering
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

ABSTRACT

Assigning "ground-holding" times to flights, i.e., determining how much to
delay the take-off of a particular aircraft headed to a congested part of the ATC
system, is becoming one of the most effective short term approaches to
diminishing the impact of airport delays.

This thesis focuses on developing algorithmic approaches to a fundamental
problem, the so-called Ground Holding Policy Problem (GHPP), in which flights
from many destinations are scheduled for arrival at a single destination airport for
which we expect some congestion.

We first address a deterministic version of the GHPP that assumes that the
evolution of the capacity (for receiving aircraft) at the destination airport can be
forecast exactly. Solution approaches to the deterministic case include minimum
cost flow algorithms, assignment algorithms, and a very fast algorithm that
exploits the special structure of the problem. The stochastic version of the
GHPP, which uses a probabilistic forecast for the evolution of the capacity of the
destination airport, is then investigated. An exact Dynamic Programming (DP)
solution to the stochastic GHPP is proposed. Since the time and space
complexities of the exact DP algorithm are prohibitive for solving problems of
realistic size, several heuristics are also developed. These heuristics include a
Limited Lookahead version of the exact DP, a "greedy" heuristic, and heuristics
that use the fast deterministic algorithm as a "building block".

The algorithms are tested in numerical examples. The results show that
significant savings in total delay costs can be expected from using algorithmic
approaches to the GHPP, even when these delays are distributed relatively
equitably among classes of users.

Formulations and directions for further research are proposed. They include
an Integer Programming formulation for a multi-airport version of the GHPP.

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

First and foremost this thesis owes its existence to Professor Amedeo R. Odoni who has provided constant encouragement and guidance throughout my stay at MIT. I wish to thank him specifically for introducing me to Operations Research and committing so much of his time to help me improve all aspects of this thesis. His uncompromising commitment to excellence in both teaching and research were a source of inspiration.

I am also especially indebted to Professor Alvin Drake who, with his special generosity, helped me in so many ways in the path of this thesis. The quality of his teaching and his total dedication to his students have left an indelible impact on me.

I would like to express my appreciation to the other members of my committee, Professor H. N. Koutsopoulos and Professor R. W. Simpson, for their interest in my work. My appreciation is likewise extended to Professor Richard L. De Neufville who generously guided my first steps as a Ph.D. candidate.

I would like to thank the C. S. Draper Laboratory for providing support for my research. My heartfelt appreciation goes to James Harrison, Owen Deutsch, and Milton Adams who have done everything to provide a congenial atmosphere for research and an ideal environment for the development and writing of this thesis. A special note of gratitude goes to Stephan Kolitz for his most needed help and encouragement during the final stages of this thesis. I am grateful to Rami Mangoubi for his friendship and constructive advice.

I wish to thank my mother for her selfless understanding and support throughout my extended student life.

My deepest gratitude goes to my wife, Fatiha, for her enduring patience and for always finding the strength to encourage me through difficult times. This thesis would not have been possible without her support. I thank my daughter, Nouzha, who helped keep the whole thing in perspective.

This thesis is dedicated to the memory of my father.
TABLE OF CONTENTS

1. INTRODUCTION ......................................................................................... 8
  1.1 Motivation .............................................................................................. 8
  1.2 General Problem Description ............................................................... 17
  1.3 Background .......................................................................................... 22
    1.3.1 Literature review .............................................................................. 22
    1.3.2 Approach in Practice ........................................................................ 23
  1.4 Preview of Thesis ................................................................................ 25
    1.4.1 Scope and Purpose ........................................................................... 25
    1.4.2 Organization of Thesis ..................................................................... 25

2. DETERMINISTIC CASE ............................................................................. 28
  2.1 Standard Formulations .......................................................................... 28
    2.1.1 Mathematical Model ........................................................................ 28
    2.1.2 Minimum Cost Flow ........................................................................ 31
    2.1.3 Assignment Problem ......................................................................... 33
  2.2 Fast Algorithm ...................................................................................... 35
3. PROBABILISTIC CASE: GENERAL DISCUSSION ................. 41

3.1 Capacity Forecast .............................................. 42

3.2 Mathematical Formulation ..................................... 51

3.3 Dynamic Implementation ...................................... 56

3.4 Cost Functions .................................................. 58

4. PROBABILISTIC CASE: ALGORITHMIC APPROACHES ... ... 61

4.1 Dynamic Programming ......................................... 61

4.1.1 Description ................................................... 61

4.1.2 Time/Space Complexity .................................... 68

4.2 Heuristics ....................................................... 72

4.2.1 Limited Lookahead .......................................... 73

4.2.2 Greedy Heuristic ............................................ 75

4.2.3 Equivalent-Capacity Heuristics ............................ 78

4.2.4 Equivalent-Policy Heuristics ............................. 79

4.2.5 Computation of Total Expected Costs ..................... 80

4.3 Dynamic Case ................................................... 84

4.3.1 Stochastic Dynamic Programming ......................... 85
4.3.2 A Simple Numerical Example ....................................... 91
4.3.3 Practical Considerations ........................................... 93
4.3.4 General Observations and Directions for Further Research ... 94

5. NUMERICAL EXAMPLES.................................................. 99

5.1 Introduction ............................................................. 99

5.2 Sample Problem Generation ......................................... 103
5.2.1 Sample Schedule Generation ...................................... 103
5.2.2 Cost Functions ......................................................... 106

5.3 Deterministic Case ..................................................... 110

5.4 Probabilistic Case ...................................................... 115
5.4.1 Description of Probabilistic Algorithms ...................... 115
5.4.2 Comparison of Heuristics ......................................... 121
5.4.3 General Cost Functions ............................................ 144
5.4.4 Comparison with Deterministic First-Come, First-Served Policies .................................................. 148
5.4.5 Forecast Systems ...................................................... 159
5.4.6 Value of Additional Landing Slots ............................. 164
5.4.7 Sensitivity of Cost Savings to Level of Utilization of Airport. . . 166

5.4.8 Sensitivity to Time Period Length........................................... 171

6. CONCLUSION ................................................................................. 177

6.1 Summary of Findings................................................................. 177

6.2 Further Research - Alternative Applications......................... 183

APPENDIX A: "Fully Dynamic" Formulation................................. 188

APPENDIX B: Proof of Dominating Capacity Case Property........... 190

APPENDIX C: Mathematical Programming Formulations.............. 198

APPENDIX D: Multi-Airport Formulations .................................... 232

APPENDIX E: Programs................................................................. 246

REFERENCES .................................................................................. 291
1. INTRODUCTION

1.1 Motivation

It has become clear in recent years that the air traffic system in the U.S. is suffering from a serious airport capacity problem\(^1\). It is estimated that direct operating costs for U.S. commercial airlines due to delays amounted to $2 billion in 1986\(^2\). The impact of these delays, however, goes far beyond these direct costs to airlines since a large proportion of passengers is also affected. This is not surprising if we consider that approximately 70% of all emplanements and deplanements in the U.S. take place in only 22 major airports (the 22 "pacing" airports in FAA terminology) most of which constitute the principal bottlenecks of the air traffic network (90% to 95% of delays can be attributed to congestion in these "pacing" airports). This situation is not likely to improve in the near future as demand for air transportation to and from major metropolitan areas is forecast to continue to grow. Furthermore, the problem is exacerbated by the use of the "hub and spoke" scheduling system by almost all major airlines. This system consists of using, for reasons of operational and economic efficiency, a small number of designated airports as transit points for a large proportion of all flights of a given airline, thereby generating a large number of "artificial" operations at these airports.

\(^1\) Since the summer of 1988 the capacity problem is being acutely felt in Europe too.

\(^2\) A recent article in the *Economist* (November 89) gives an estimate of total yearly delay costs for European airlines of $8 billion.
The air traffic network congestion problem can be addressed according to different time spans:

- **Long-term approaches** include the construction of additional airports (which typically requires 10 to 15 years from conception to operation), improved Air Traffic Control (ATC) technologies, additional runways at existing airports, and use of larger aircraft. Although several such airport improvement programs are under way in the U.S. and abroad, these approaches are generally very costly and may often be difficult to implement due to a lack of public support. This is particularly true for airport extension projects; these projects are usually needed in larger metropolitan areas where demand for air travel is high, but where resistance to such projects is also high due to their impacts on local communities.

- **Medium-term approaches** (6 months to a few years) are mostly administrative or economic in nature. They try to alleviate congestion by modifying the temporal pattern of aircraft flow through the ATC network, for example by imposing different user charges at different times of day or putting pressure on airlines to modify some of their scheduling practices, such as the use of the hub-and-spoke system, that tend to concentrate flights temporally and geographically.

- **Short-term approaches** have to deal with a given schedule and network capacity and are intended to mitigate the effects of unavoidable congestion through the control of the flow of aircraft. The time horizon for such flow control can vary from a few minutes to a whole day and the purpose is to best match the flow of aircraft to available capacity throughout the time horizon.
This thesis focuses on a particular macroscopic version of these short-term problems that consists of considering ground holds on some flights before departure. In a paper, that to our knowledge is the first systematic discussion of these problems, Odoni (1987) refers to the family of such short term approaches as the Flow Management Problem in Air Traffic Control (FMP). This paper also contains a detailed discussion of the problem central to this thesis and therefore, in order to motivate and define the problem, the following discussion will draw from it and will use (and redefine) the same terminology.

In order to describe the FMP we use a network representation of the ATC system that distinguishes four types of elements, as shown in Figure 1-1:

- Airway elements are represented by arcs in the network. These correspond to the physical paths that aircraft use.

- Airport elements are the sources and sinks for air traffic in the system; they are represented by nodes in the network.

- Waypoint elements are also represented by nodes in the network. They correspond to points in the network where airways intersect.

- Sector elements are defined to be a set of waypoints and segments of airways that is treated as a unit for air traffic control purposes.
Congestion can occur at any one of the elements of the network of Figure 1-1 when the capacity of these elements is reduced. If the capacity of each one of the elements of the network were known and did not change with time, there might be no delays in the ATC system since the flows could be adjusted to match exactly these known capacities. Delays occur because these capacities, in particular the airport capacities, can be greatly affected by conditions that change over time and may be difficult to predict. Airports constitute the principal "bottlenecks" of the ATC system. Weather conditions (visibility, precipitation, wind, cloud ceiling, etc...) determine the capacity of a given airport (in terms of the number of landings and take-offs that can occur during a given time span) since they affect minimum landing separation rules and can dictate the runway
configuration to be used. Furthermore, the reductions in airport capacity can be as high as 50% in some extreme weather conditions. Figure 1-2 shows the runway capacity profile for Boston-Logan Airport for the year 1987; it shows in particular that, for that year, the capacity of Logan Airport was reduced from a maximum of 130 operations per hour to less than 100 operations per hour more than 20% of the time, with a reduction to less than half that maximum capacity (60 operations per hour) happening 15% of the time.
There is a need to adjust the aircraft flows through the various elements of the ATC network on a short term basis using available information and forecasts concerning the capacities of airports and other elements of the ATC network. The FMP can then be defined as the problem of adjusting flows in the ATC network so as to minimize the cost of delays given the available information concerning the (present and future) status of the elements of the ATC network. Odoni (1987) distinguishes two types of such flow management actions. "Tactical" actions are to be exercised when the aircraft is already airborne and include:

- High altitude holdings, path stretching manoeuvres, or modifying en-route flight plans in order to avoid costlier low altitude delays.

- The control of en-route speeds to time the arrival of an aircraft at a congested point of the ATC network.

- The sequencing of aircraft for landing to maximize runway acceptance rate.

The above type of action is clearly microscopic in nature. These actions can help control the flow of aircraft through specific areas of the air traffic network but are limited in the extent of control they permit by the very fact that the aircraft is airborne and is therefore subject to fuel and safety constraints.

The other type of action is a macroscopic or "strategic" type of action with greater potential for regulating aircraft flows. It includes:

- The modification of flight plans of some flights before take-off in order to by-pass congested areas of the network.
- Delaying the departure time of some flights. These delays are referred to as "gate holds" or "ground holds" and correspond to delaying the actual departure time of an aircraft beyond its scheduled take-off time. These delays are to be taken before the aircraft starts its engines on the apron area (either at the gate or in a remote parking area) even if the aircraft is otherwise ready to taxi to the runway.

The first type of "strategic" action also has a limited potential since there are clearly limits to the scope of modifications of flight plans (e.g., fuel constraints); this type of action remains in the domain of actions of a "fine tuning" nature. The second type of action, on the other hand, provides greater flexibility in adjusting flows and can lead to greater savings in delay costs for the following two reasons:

- First, because our ability to control and regulate aircraft flows is greatly increased since we do not have to deal with some of the constraints (e.g. fuel constraint) of airborne control.

- Second, because important savings in delay costs as well as improved safety can be expected if we are able to "absorb" some of the delays on the ground. Savings in delay costs can be expected because ground-holding delays are less costly than airborne delays that involve fuel consumption as well as depreciation and maintenance costs.\(^3\)

Since, in general, it is not possible to predict airport capacities exactly at the times of departure of aircraft, the basic trade-off is between the cost of airborne delays that result from "optimistic" strategies, which impose little ground holds, and the costs of ground

---

\(^3\) Ground-holds are also preferable for Safety considerations.
holds from "pessimistic" strategies, which impose excessive ground holds and may result in unused capacity.

This type of strategic flow management action should therefore be considered as the essential component of a flow management system; it should, nevertheless, be complemented by the "tactical" actions mentioned above but those are only of a "fine-tuning" nature. It is also important to note that the FMP is not a short-term problem in the sense that it is to be addressed only until the longer term issues are resolved. It is, in fact, to be considered a "permanent" part of an efficient ATC system. The reason for this is that, as we have argued above, the FMP is motivated by the fact that the capacity condition in the ATC system can change over time and may be difficult to predict. This situation is not likely to be greatly affected by longer term actions (except, maybe, in terms of improvement in weather prediction technology) and there will always be a need to adjust flows in the short term to match available capacity for an efficient use of the ATC network.

Although the above classification of methods for tackling the FMP is useful for decomposing the problem, we should keep in mind that the boundaries between tactical versus strategic flow management are not clearly defined in practical situations. There are some obvious interdependencies, such as, for example, the fact that airport capacity depends not only on weather conditions but also on tactical actions such as sequencing of landing aircraft. A complete flow management system has to deal with interfacing the various approaches.

As is the case with airport capacities, capacities of airways, waypoints and sectors (in terms of the number of aircraft that can traverse these elements per unit time or occupy them at any given time) can also be reduced. In most cases, however, the remaining
capacities for these elements are greater than the actual traffic flows that use them. This suggests considering, as a first approximation, strategic flow actions when we assume that the only capacitated elements of the air traffic network are the airports. The resulting version of the FMP that deals exclusively with the trade-off between ground-holds and airborne delays for a network in which airports are the only congested elements is called the generic FMP in the paper by Odoni.

The main object of this thesis is a fundamental case of this problem in which we must decide on ground holds for flights from many destinations scheduled for arrival at a single congested airport. There are several motivations for looking at the single arrival airport case:

- There are instances for which this is a reasonable formulation. This is obviously true when only one airport of the network is expected to be congested. It is also applicable to cases for which several airports are expected to be congested but there is little traffic between these congested airports (or if the congestion is not affecting the inter-airport traffic on both ends simultaneously).

- It features and focuses on the most important aspect of the strategic FMP: the trade-off between ground-holding delays and airborne delays.

- We hope to be able to develop solution methods for the "full network" case based on solution methods for the single airport case.

We will refer to the single airport version of the generic FMP as the Ground Holding Policy Problem (GHPP) throughout this thesis. The following section describes the different formulations of the GHPP that will be considered in the thesis and discusses the assumptions of these formulations.
1.2 General Problem Description

The GHPP is concerned with timing the arrival of aircraft coming from several origin airports into a single arrival airport, Airport Z, and is described, referring to Figure 1-3, as follows:

(1) We consider arrival operations at a given airport Z during a time interval [0,T] for which we expect some amount of congestion.

(2) We are given a complete list \( F_1, F_2, \ldots, F_N \) of flights departing from a number of origin airports and scheduled to land at airport Z during \([0,T]\). For each flight \( F_i \), we know the scheduled departure and arrival times of \( F_i \); furthermore we assume that the travel times are deterministic and known in advance.

(3) For each \( t_2 \in [0,T] \) and any \( t_1 \geq t_2 \) we are given a probabilistic forecast for \( \text{CAP}(t_1|t_2) \), the arrival capacity of airport Z at time \( t_1 \).
The GHPP is then defined as the problem of finding, for each flight $F_i$, the "optimal" amount of ground hold to be imposed on flight $F_i$ so that the overall expected total delay cost (ground + airborne queueing delays) is minimized. A solution to the GHPP will be referred to as a Ground Holding Policy (GHP) in future discussions. If the queueing delays were predicable, we would "take" them entirely as ground delays before departure so as to minimize costs. Because of the uncertainty concerning the capacity of airport Z during $[0,T]$, however, we cannot predict the amount of queueing delays that each flight would incur if it left on time. The GHPP is therefore concerned with finding the amount of ground delay to impose on each flight so as to strike an
"optimal" balance between the costs of these (known) ground delays and future (unknown) airborne delay costs based on available information concerning the capacity of airport Z. The most general version of the GHPP has therefore the following characteristics:

(a) It is a **stochastic problem**. There are many circumstances for which it is not possible to exactly predict the capacity of an airport even a few hours in advance. The reason is that capacity depends strongly on local weather conditions which are subject to high uncertainty. In these circumstances we may have to settle for a **probabilistic** capacity forecast. The object of the GHPP is then to try to reach an optimal balance between (present-known) ground delays and (future-expected) airborne delays.

(b) It is a **dynamic problem**. In (3) we assumed that at any time $t_2$ for which we need to make a decision we have access to a whole family of random variables (a stochastic process) $\{\text{CAP}(t_1,t_2) : t_1 \geq t_2\}$. This formulation allows that, for any specific $t_1$, the forecast $\text{CAP}(t_1,t_2)$ can change with $t_2$.

(c) It is a **combinatorial problem**. Because the individual delay costs for aircraft can be very different, it is not enough to think of this problem solely in terms of controlling the flow of aircraft arriving at airport Z but we have to determine the "optimal" composition of these flows. The combinatorial aspect is evident if we consider the case for which we can predict the capacity of airport Z exactly (deterministic capacity forecast); we can consider this situation as one for which the stochastic and dynamic aspects have been "resolved" but it is obvious that large cost-savings can be expected from assigning individual aircraft to available capacity in an optimal manner.

A fundamental characteristic of all the formulations in this thesis is a discretization of the time $[0,T]$ into a number, $P$, of individual time periods $T_1, T_2, \ldots, T_P$. The
capacities $\text{CAP}(t_1|t_2)$ are translated into capacities $K_1, K_2, \ldots, K_P$ (which can be random variables), where $K_j$ is the arrival capacity for airport $Z$ during time period $T_j$ (the maximum number of landings that can take place at airport $Z$ during $T_j$). Within this framework, the GHPP is concerned not with determining an exact amount of ground delay for each flight but with timing the arrival time of individual flights to individual time periods. This leads to a macroscopic view of the situation particularly suited to the strategic nature of the GHPP. For example, it justifies the assumption that travel times are deterministic (see (2) above). Let us discuss the underlying assumptions of our formulation of the GHPP in view of its macroscopic nature:

- The GHPP assumes that the arrival capacities $K_1, K_2, \ldots, K_P$ of airport $Z$ are exogenous to our model. There is however an interaction between the actual capacities and the GHP resulting from solving the GHPP. The reason for this is that the acceptance rate of a given runway configuration during a given time period depends on the aircraft mix during that time period through a set of minimum separation rules between aircraft types. We can, however, assume that this interaction occurs at the microscopic level; the macroscopic nature of the model allows us to consider an average aircraft mix per time period and therefore to talk about capacities that, in effect, do not depend on the GHP.

- The GHPP assumes that we can determine arrival capacities separately from departure operations at airport $Z$. In some cases, when arrivals and departures share the same runways, there is an interaction between both types of operations. Again we can assume, given the macroscopic nature of the approach, that the split of total capacity between arrivals and departures is known and remains constant for each time period $T_i$.

- Finally, the model assumes implicitly that there is no departure congestion at the airports of origin so that a given GHP can be implemented regardless of conditions at
these origin airports. Including such considerations into the model is in the scope of multi-airport formulations of the GHP.

The probabilistic and dynamic aspects of this problem are difficult to deal with. The approach taken in this thesis is to decompose the problem. We first develop solution methods for the deterministic version. This is a purely combinatorial problem, as was pointed out earlier. Then we focus on a "static" probabilistic version of the problem which consists of determining ground holds based on a unique probabilistic forecast for the airport capacity. Finally we will use the results of the analysis of this "static" version to propose solution methods for the dynamic probabilistic version of the GHPP. Therefore, the different problems investigated in this thesis are:

- The deterministic GHPP. This corresponds to assuming that the capacities $K_1, K_2, \ldots, K_P$ are fixed and known. The GHPP is then concerned with assigning each flight $F_i$ to a time period $T_j$ so as to minimize a total ground delay cost (and not violate the capacity constraints).

- The static-probabilistic GHPP assumes that the capacities $K_1, K_2, \ldots, K_P$ are random variables and that, at some time $t$ before the earliest departure time of any of the flights $F_1, \ldots, F_N$, we are given a probability distribution $P_{K_1, \ldots, K_P}(t)$ that represents a forecast for these capacities and that is assumed to remain fixed after $t$.

- The dynamic-probabilistic GHPP also assumes that the capacities $K_1, K_2, \ldots, K_P$ are random variables. However we also assume that we have a model for the evolution of the probabilistic distribution $P_{K_1, \ldots, K_P}(t)$ with time; i.e., we assume that we have a family of distributions $P_{K_1, \ldots, K_P}(u)$ for all times $u$ after the earliest departure time.
1.3 Background

1.3.1 Literature Review

The existing literature dealing with the ATC flow management problem seems to be rather limited. One reason for this may be that network-wide ATC delays were not felt until 1986. As we mentioned above, Odoni’s paper (1987) seems to be the first systematic description of the flow management problem. It provides a mathematical formulation of the generic FMP and points to the need for analytical and algorithmic approaches to this problem. This paper also contains a discussion of the interplay between the technical and policy aspects of the FMP.

To our knowledge the only algorithmic approach dealing specifically with the GHPP is a paper by Andreatta and Romanin-Jacur (1987). In this paper the authors propose a Dynamic Program to solve a static-probabilistic version of the GHPP for which it is assumed that a destination airport experiences congestion during a single time period $T$. We will not describe this formulation in detail here since it is precisely this formulation that will be extended to multiple time periods in Section 4.1.

Even though algorithmic approaches are almost nonexistent there have been a number of computer-simulation based approaches to the FMP. In fact it is precisely such a simulation based approach that is utilized at the present time by the Federal Aviation Administration (FAA) to determine ground holds; we discuss this approach next.
1.3.2 Approach in Practice

The FAA is responsible for initiating and coordinating ground-holding strategies in the U.S. air traffic system. The FAA operates a Central Flow Control Facility (CFCF) in Washington, D.C. for this purpose; the CFCF is equipped with state-of-the-art information gathering capabilities through which it obtains access to regional and local weather data and forecasts as well as up-to-the-minute information on the status of virtually all airborne traffic in the U.S.

Part of the information the CFCF obtains from local control centers is a forecast of airport capacities for a given time period. This information is then used to determine ground holds for individual flights by:

- estimating (through what is essentially a deterministic simulation model) the air-delay that could be expected for each flight if it left at its scheduled departure time. Figure 1-4 illustrates this process.

- setting these ground holds equal to the "expected" airborne delay if the latter is not below a given threshold (typically the "threshold" may be set at 15 or 20 minutes).
This process can therefore be described as assigning flights to available capacity on a first-come first-served basis (i.e. flights with earlier scheduled arrival times are given priority) without considering any uncertainties in the capacity forecast. Assuming the capacity forecast used in this process is indeed accurate, this approach would minimize the total delay costs if all flights using airport Z had identical delay costs per unit of time (since it minimizes total aircraft-delay). It does not, however, consider cost
functions that reflect different operating costs for different sizes of aircraft. It also does not consider explicitly the uncertainty concerning the capacity forecast itself.

1.4 Preview of Thesis

1.4.1 Scope and Purpose

This thesis has two main goals:

- The primary goal is to demonstrate the need for analytical-algorithmic approaches to the GHPP. This will be done in the context of the single airport static-probabilistic version of the GHPP by developing optimization algorithms and implementing them on realistic sample problems to demonstrate the potential savings in costs.

- A second goal is to set the stage for further research on the GHPP. This will require clarifying the structure of the dynamic-probabilistic GHPP and determining, on the basis of this analysis, promising directions and formulations for tackling the problem in a dynamic context.

1.4.2 Organization of Thesis

The remainder of the thesis is organized as follows:

Chapter 2 is concerned with the deterministic version of the GHPP. We first identify two "standard" formulations: a network-flow formulation for which the optimal ground holding policy corresponds to the minimum cost flow in a capacitated network,
and an assignment formulation for which this optimal policy corresponds to the least cost
assignment of each flight to a landing slot. These "standard" formulations apply to the
most general case of the deterministic version for which (ground) delay costs can have
any functional shape. In section 2.2 we develop a fast algorithm that yields the optimal
ground holding policy when the delay cost functions satisfy certain "regularity"
conditions that are identified.

In Chapter 3 we look at several issues related to the probabilistic case. After a
discussion of capacity forecast systems, we develop a mathematical formulation of the
probabilistic GHPP that allows a precise definition of the static as well as the dynamic
versions of the problem. Finally we discuss implementation issues and the issue of delay
cost functions.

Chapter 4 presents several algorithmic approaches to the probabilistic case. We
first develop a Dynamic Program that extends the DP used by Andreatta and
Romain-Jacur (1987) for the single-period case to the multiperiod case of the static
probabilistic version of the SFMP. We then propose several heuristic algorithms, some
of which use the fast algorithm developed in Chapter 2 for the deterministic version as a
building block. Finally, in the light of the discussion in Chapter 3, we propose a
Stochastic Dynamic Program as well as some heuristics based on the "static" heuristics
for a simplified version of the dynamic-probabilistic version of the GHPP.

Chapter 5 illustrates the use of the fast deterministic algorithm and the "static"
probabilistic algorithms from Chapter 4 through several numerical examples.

Chapter 6 reviews the main contributions of the thesis with a view towards
further research. We first summarize the contributions of the thesis. Then, we show
how the mathematical programming formulation of Chapter 2 can be extended to deal
with a deterministic multi-airport case. We then conclude with a discussion of directions for further research and alternative applications.
2. **DETERMINISTIC CASE**

This short but important chapter is concerned with the deterministic version of the GHPP which assumes that future capacity at airport $Z$ can be predicted exactly. This assumption reduces the GHPP to a purely combinatorial problem for which standard solution methods are available. One of the motivations for looking at the deterministic case is that there are cases for which it is reasonable to assume that capacities can be forecasted with little error. This is the case for airports located in areas where there is little variability in weather conditions or for which changes in weather conditions are predictable and weather patterns remain stable for a long time period, once established.

The motivation for looking at the deterministic formulations goes, however, beyond cases for which the deterministic assumption is a reasonable one. We develop "deterministic" solution methods in the hope of being able to use some of them as components of probabilistic solutions. Indeed some of the algorithms we will develop in Chapter 4 for the static-probabilistic case use the fast algorithm of section 2.2 as a building block. Some of the most promising algorithms for the dynamic-probabilistic case will also be based on deterministic solution methods.

2.1 **Standard Formulations**

2.1.1 **Mathematical Model**

We consider arrival operations at a given destination airport (airport $Z$) during a time interval $[0, T]$ for which we expect some congestion. The interval $[0, T]$ is
subjected to several consecutive time periods $T_1, T_2, ..., T_P$ with corresponding fixed deterministic capacities $K_1, K_2, ..., K_P$. $N$ flights $F_1, F_2, ..., F_N$ are scheduled to land at airport $Z$ during these time periods. For each flight $F_i$ we know $P_i$ the flight's scheduled landing period ($P_i = 1, ..., P$) and the cost $C_{gi}(t)$ of delaying flight $F_i$ at time periods on the ground before take-off. We assume that all flights that were not able to land during one of the time periods $T_1, T_2, ..., T_P$ can do so during a final time period $T_{P+1}$ (i.e. we assume that $K_{P+1} = \infty$).

The objective is to find the ground holding policy $X_1, X_2, ..., X_N$, where $X_i$ is the number of time periods flight $F_i$ is delayed on the ground. The ground holding policy must be feasible (does not violate the capacities $K_i$) and minimize the total ground delay cost:

$$TC = \sum_{i=1}^{N} C_{gi}(X_i)$$

The underlying assumptions of this model are as follows:

1. Imposing a ground delay $X_i$ on flight $F_i$ will make that flight arrive at airport $Z$ during time period $T_{P_i + X_i}$. This requires that travel times be deterministic and that airport $Z$ be the only congested element of the air traffic network (i.e. no delays can occur at the origin airports or en route to airport $Z$).

2. We can determine in advance how available capacity at airport $Z$ will be allocated between arrivals and departures.

Both assumptions can in many cases be reasonably realistic given the macroscopic nature of the model. The assumption that airport $Z$ is the only congested element of the
network is the key one in virtually all the problems this thesis is concerned with. As was stated in the introduction, the approach taken in this thesis is to decompose the problem by looking at the single airport SFMP as a first approximation.

We define the assignment variables \( x_{ij} \) by \( x_{ij} = 1 \) if flight \( F_i \) is assigned to land during period \( T_j \), \( x_{ij} = 0 \) otherwise. The \( x_{ij} \)'s are only defined for \( j \geq P_i \).

We denote by \( C_{ij} \) the quantity \( C_{ij}(j-P_i) \), the cost of assigning flight \( F_i \) to land during time period \( T_j \). This quantity is also only defined for \( j \geq P_i \).

Using this notation the solution of the following integer program (IP) yields the optimal policy:

\[
\min \sum_{i=1}^{N} \sum_{j=P_i}^{P_i+1} C_{ij} x_{ij} \tag{1}
\]

subject to:

\[
\forall (i,j) : x_{ij} = 0 \text{ or } 1 \tag{2}
\]

\[
\forall i (=1,N) : \sum_{j=P_i}^{P_i+1} x_{ij} = 1 \tag{3}
\]

\[
\forall j (=1,P) : \sum_{i=1}^{N} x_{ij} \leq K_j \tag{4}
\]

The constraint matrix of this IP is totally-unimodular. We can therefore relax the integrality conditions and use, for example, the simplex method to solve it. In fact, as we will see in the next section, the solution of this IP corresponds to a minimum cost flow in
a capacitated network and we can therefore use faster specialized algorithms such as the Out-of-Kilter algorithm. We will also see that a slight reformulation of the network approach poses the problem as a classical assignment problem for which more specialized algorithms exist. Finally, we will see that an even faster algorithm can be developed when the cost functions $C_{gi}(X_i)$ satisfy certain conditions that we will identify in section 2.2.

2.1.2 Minimum Cost Flow

Figure 2-1 illustrates a capacitated network formulation of the problem (1)-(4) for the case of general costs functions $C_{gi}(t)$.
The numbers in brackets represent the costs per flow-unit associated with the corresponding arc. When no number is indicated this cost is assumed to be zero.

The letters u and l represent respectively the upper and lower limits for the flow on each arc. The default values are infinity for the upper bound and zero for the lower bound.

Each time period Tj is represented by an arc with the upper bound for the flow on this arc set to the capacity of the time period (u=Kj). Each flight Fi generates a node "flight i" in the above network. Each node "flight i" is connected to the nodes at the origin of the arcs representing all time periods with index ≥ Pi.
It is clear that the cost of any feasible flow through the network is:

$$
\sum_{i=1}^{N} C_{g_i}(j - P_i)
$$

where \( j \) is the index of the time period corresponding to the non-zero flow out of the node "flight \( i \)."

The optimal assignment therefore corresponds to the minimum cost flow through this network.

### 2.1.3. Assignment Problem

We can also develop a truly microscopic formulation and consider minimum separations between landing aircraft explicitly. Instead of considering the period arcs with upper bounds \( K_j \) to correspond to time intervals, with each interval having the capacity to serve several flights, we can view them as individual landing slots for single aircraft. The number of landing slots to be generated for each time period is equal to the forecasted capacity for that time period. The problem becomes a classical assignment problem, as illustrated by figure 2-2.
If $S_i$ is the scheduled arrival time for flight $F_i$ then the cost $C_{ij}$ of assigning flight $F_i$ to landing slot $L_j$ is given by: $C_{ij} = C_{gi}(t_j - S_i)$ if $t_j \geq S_i$

This can also be solved as a minimum cost flow problem if we introduce some dummy nodes and arcs as shown in figure 2-2. But to be able to use faster algorithms tailored to the assignment problem we have to make sure that the cumulative number of available landing slots does not exceed the number of flights scheduled at any point in time and that the cost matrix is complete. We accomplish this in the following manner:

Never create a landing slot at time $t$ if the cumulative number of scheduled arrivals at that time is lower than the cumulative number of already created landing slots. Doing
this ensures that the number of available assignments equals the number of flights and still produces an optimal solution since it is clear that no assignment will use more landing slots than the flights can fill according to the earliest schedule. We also need to complete the cost matrix; if we index the times for the landing slots as \( t_1, t_2, \ldots, t_N \) we can do so by setting \( C_{ij} = C_{Gi}(t_N - S_i) + 1 \), for \( t_j < S_i \).

This level of detail in the analysis seems to be unwarranted. Still the assignment formulation is justifiable in the sense that it produces a solution that is just as good as the one produced by the time-period formulation and yet can utilize specialized algorithms for the assignment problem.

The time complexity of the Hungarian Algorithm which solves the assignment problem is \( O(N^3) \) where \( N \) is the number of flights (or equivalently the number of landing slots). Faster algorithms have been developed for the assignment problem, but we are typically dealing with time periods of several hours and therefore a number of (arriving) flights as high as several hundred or even a thousand. It is therefore reasonable to try to develop much faster algorithms specifically tailored to our problem. In the following section we will develop such an algorithm that takes advantage of special forms for the cost functions \( C_{Gi}(t) \).

### 2.2 Fast Algorithm

We utilize the mathematical formulation developed in section 2.2.1 and denote by \( \partial_j C_{Gi} \) the marginal cost of delaying flight \( F_i \) during time period \( T_j \), i.e.,

\[
\partial_j C_{Gi} = C_{Gi}(j+1-P_i) - C_{Gi}(j-P_i).
\]
A simple algorithm can be developed when the cost functions $C_{gi}(t)$ satisfy the following two conditions:

For any pair of flights $(F_i, F_k)$:

(i) if $\partial_j C_{gi} > \partial_j C_{gk}$ for some time period $T_j$, then for any $m > j$,

$$\partial_mC_{gi} > \partial_mC_{gk},$$

and

(ii) if $\partial_j C_{gi} = \partial_j C_{gk}$ for some time period $T_j$, then for any $m > j$,

$$\partial_mC_{gi} = \partial_mC_{gk}.$$

Intuitively, these conditions say that if it is more costly to delay flight $F_i$ than flight $F_k$ during time period $T_j$, then it never becomes cheaper to delay flight $F_i$ rather than $F_k$ later on. These conditions can therefore be viewed as regularity condition on the cost functions and are reasonable assumption for the particular context under consideration.

In following discussions we will refer to conditions (i) & (ii) as regularity conditions.

The following notation will be used to describe the very fast algorithm that can be used when the regularity conditions apply:

$$E_j = \text{set of indices of flights eligible for operation during time period } T_j \text{ under the optimal policy.}$$

$$OP_j = \text{set of indices of flights that actually operate during time period } T_j \text{ under the optimal policy.}$$
The following algorithm, which will be referred to throughout the remainder of this thesis as the "Fast Algorithm", determines the optimal policy $X_1, X_2, \ldots, X_N$:

**step 1: initialize**

$E_0 = \emptyset$

$OP_0 = \emptyset$

**step 2: for** $j=1$ **to** $P$ **do:**

$E_j = \{1 \leq i \leq N \mid P_i = j\} \cup E_{j-1} \setminus OP_{j-1}$

$OP_j = \{i \in E_j \text{ with the first } K_j \text{ highest } c_{j,i}\}$

**step 3: for** $i=1$ **to** $N$ **do:**

if $\exists j \in [1...P]$ such that $i \in OP_j$, then set $X_i = j - P_i$

if no such $j$ exists, then set $X_i = P + 1 - P_i$

The algorithm consists of, for each time period $T_j$, ordering candidate flights for landing according to their marginal cost of delay, and allowing the $K_j$ flights with highest cost to land. The complexity of the algorithm is therefore $O(PN\ln(N))$ where $N$ is the total number of flights and $P$ is the number of time periods. The above algorithm is the one that is implemented in Chapter 5 and the (Fortran) code of which is listed in Appendix E.
However, it has been subsequently pointed out to me\textsuperscript{1} that a faster ($O(N \ln N)$) algorithm yields the same (optimal) policy; this algorithm consists of first ordering all flights according to their final marginal costs, $\partial P + 1 C_{g_i}$, and then assigning these flights in that order to available capacity for each time period\textsuperscript{2}. This observation will play an essential role when we will need to compute the airborne delays associated with a given ground holding policy in the probabilistic case. (This will be discussed in more detail in section 4.2.5; several algorithms used in Chapter 5 to compute the airborne delay costs will use this observation.) Nevertheless the correctness of the argument to follow is clearer and more intuitive if we adhere to the 3-step description given above.

The correctness of the algorithm follows intuitively from the conditions imposed on the cost functions; they insure that when we consider candidate flights for a time period these flights can be unambiguously ordered according to their potential costs for this and for all future time periods. Therefore we can decide in each time period which flights to operate given the available capacity during that time period. A formal proof of the correctness of the algorithm requires more notation but it also helps understand why the regularity conditions are needed.

\textsuperscript{1} Private communication with A. R. Odoni.

\textsuperscript{2} The savings in computational time are likely to be barely noticeable in practice since the algorithm implemented for the numerical examples of Chapter 5 already runs in less than one second for the largest instances of the problem.
Proof of Fast Algorithm:

We define \( A_j \) to be the set of flights available for landing during time period \( T_j \);
\[ F_i \in A_j \iff P_i \geq T_j. \]

Let \( P \) be a given ground holding policy. We denote by \( L^j_P \) the set of flights
assigned to land during time period \( T_j \) according to policy \( P \).

Assume there is a flight \( F_k \in L^j_P \) and a flight \( F_i \in A_j \) but \( F_i \notin L^j_r \) for all \( r < j \) such
that
\[ \partial_j C_{gi} > \partial_j C_{gk} \quad (5). \]

To prove the correctness of the algorithm it is enough to show that policy \( P \) is not
optimal. We show this by considering the policy obtained from \( P \) by switching the roles
of \( F_i \) and \( F_k \); we call this policy \( Q \). More precisely:

\[ F_i \in L^j_Q, \text{ and } \forall h \neq (i,k) \; [F_h \in L^j_P] \Rightarrow [F_h \in L^j_Q]. \]
Furthermore let \( m \) be such
that \( F_i \in L^j_m \). Then we set \( F_k \in L^j_m \) (we can do this since \( F_k \in L^j_P \) implies \( F_k \in A_j \) which implies \( F_k \in A_m \) since \( m > j \). The feasibility of policy \( P \) implies that of policy \( Q \)
since they schedule exactly the same number of flights in each period.

The costs of policies \( P \) and \( Q \) differ only in the costs for flights \( F_i \) and \( F_k \). We
have to compare \( CP = C_{gi}(T_m-P_i) + C_{gk}(T_j-P_k) \) and \( CQ = C_{gi}(T_j-P_i) + C_{gk}(T_m-P_k) \):

\[
CP = \sum_{l=P_i}^{j-1} \partial_l C_{gi} + \sum_{l=J}^{m-1} \partial_l C_{gi} + \sum_{l=P_k}^{j-1} \partial_l C_{gk} \\
CQ = \sum_{l=P_i}^{j-1} \partial_l C_{gi} + \sum_{l=P_k}^{j-1} \partial_l C_{gk} + \sum_{l=j}^{m-1} \partial_l C_{gk}
\]
\[ \text{CP - CQ} = \sum_{i=j}^{m-1} (\partial_j c_{g_i} \cdot \partial_j c_{g_k}) \]

Therefore CP - CQ > 0 by (5) and conditions (i)&(ii), and policy Q is better than P.

Almost all the numerical examples in this thesis will assume that the marginal costs are given by:

\[ \partial_j c_{g_i} = c_i(1+\alpha)^{j-P_i}, \text{ where } c_i \text{ is the cost of delaying flight } F_i \text{ for one time period on the ground.} \]

Let us verify that these cost functions satisfy the regularity conditions:

Assume that, for two given flights \( F_i \) and \( F_k \), we have \( \partial_j c_{g_i} > \partial_j c_{g_k} \) for some time period \( T_j \). This translates into \( c_i(1+\alpha)^{j-P_i} > c_k(1+\alpha)^{j-P_k} \). Now, for any \( m>j \), we can multiply both sides of the inequality by \( (1+\alpha)^{m-j} \) yielding: \( c_i(1+\alpha)^{m-P_i} > c_k(1+\alpha)^{m-P_k} \) or \( \partial_m c_{g_i} > \partial_m c_{g_k} \).

The coefficient \( \alpha \) will play an important role in these numerical examples since, as we will see in Chapter 5, the magnitude of \( \alpha \) affects the distribution of delays among classes of aircraft. (For now we can simply note that setting \( \alpha \) to 0 corresponds to assuming linear cost functions which tends to assign a disproportionate amount of delays to smaller aircraft; on the other hand, a very high \( \alpha \) will result in assigning aircraft to available capacity on a first come first served basis as is usually done under present practice.) The coefficient \( \alpha \) will be referred to as the "Ground Cost Increase Coefficient" in the remainder of this thesis. For an interpretation of \( \alpha \) we note that

\[ \alpha = \frac{\partial_j c_{g_i} - \partial_{j-1} c_{g_i}}{\partial_{j-1} c_{g_i}}, \] which is the relative increase in cost due to holding a flight on the ground for an additional hour.
3 PROBABILISTIC CASE: GENERAL DISCUSSION

The initial motivation for a probabilistic version of the GHPP is to allow for a more realistic model. In practice, there is much uncertainty on the capacity of airports even a few hours into the future and large errors can occur with any deterministic prediction. Considering that the flight times of a large proportion of aircraft using a particular airport are generally of several hours and that we have to decide on a ground hold, at the latest, by the departure time of a flight, only a probabilistic forecast for capacities could be available in practice. In this case we must try to minimize an expected delay cost.

The probabilistic formulation is also closely related to the dynamic nature of the problem as we will want to take advantage of additional weather information as it is received, to diminish some of the uncertainty concerning the capacities $K_1, K_2, \ldots, K_P$ for time periods $T_1, T_2, \ldots, T_P$. Thus, we will need to extend the formulation of the problem to be able to introduce new concepts related to its dynamic nature.

The purpose of this chapter is to clarify the structure of the probabilistic formulation. We begin, in Section 3.1, by looking at the capacity forecast mechanism and related implementation issues. Then, in Section 3.2, we extend the deterministic mathematical formulation into two probabilistic formulations: a "static" formulation and a "dynamic" formulation. In Section 3.3, we investigate some practical aspects of the probabilistic formulation. Finally, we examine the issue of cost functions in Section 3.4.

Approaches toward solving specific probabilistic version of the GHPP will be described in detail in Chapter 4.
3.1 Capacity Forecast

The inherent uncertainty in the evolution of weather patterns makes the issue of airport capacity forecasting a difficult one. It is beyond the scope of this thesis to address this issue in any detail but it is essential to give a precise meaning to a number of concepts relevant to capacity forecasting which are directly related to the probabilistic formulation of the GHPP. In doing so we will also be able to show how optimization techniques can interact with capacity forecast systems to yield a complete flow management system; we will also able to propose a minimal forecast system that will have enough of the attributes of more sophisticated systems to serve later as a basis for illustrating the basic concepts.

Major variations in the capacity of a given airport over time, given a fixed air traffic control technology and a fixed airport layout, can be related almost exclusively to the variations of the weather parameters that affect directly the aircraft flow rate into the airport. Even though other time-dependent variables such as the fleet mix have an impact on this flow rate through minimum landing separation requirements, that impact is only felt at the microscopic level and we can, given the level of detail of our analysis, consider the flight mix to be uniform through time. It is therefore reasonable to assume for our purposes that there is an unambiguous relationship between the measurements that constitute a weather observation, to be precisely defined later, and the capacity of airport Z.

The probabilistic formulation assumes that future capacity of the airport can be described as a random variable or more precisely, for the multiperiod case at least, as a stochastic process. We want to show that such a formulation is not only more realistic
than a deterministic description but is dictated by the nature of the problem. On one hand, we will see how a capacity forecast can be naturally formulated as a stochastic process. On the other hand, the nature of the weather system, particularly within the local framework we are dealing with, often makes the probabilistic formulation inevitable because of the weather system's inherent chaotic character which makes it computations irreducible.

We assume that the set of variables that go into the description of a weather forecast is complete. By this we mean that there is a clear, well-known and well-defined, one-to-one correspondence between the state of the weather at any time t, as described by this set of variables, and the capacity of airport Z at that time. How this total capacity is arrived at considering the split between landings and take-offs, or how the different landing separations between different aircraft types are taken into account is not the subject of this discussion. Instead, we assume that there exists a fixed set of rules that allows such computations. Our definition of the state of the weather is therefore such that we can translate unambiguously a weather forecast into a landing capacity forecast so that these notions are assumed to be equivalent and will be used interchangeably in the remainder.

In order to complement optimization methods such as those described in the next chapter we need a Capacity Forecast System (CFS). The basis of such a forecast system is a sequence of observations intended to capture the evolution of parameters relevant to a capacity forecast for the time periods preceding time t at which we make that forecast. We define the meaningful backward scope to be the number of time periods before t whose observed conditions have any bearing on capacities after t. More precisely, this backward scope can be measured as the number of time periods we have to go back until no statistical dependence remains between the measurements of the parameters and the
capacity at time t. Similarly, we can define a **meaningful forward scope** to be the number of time periods after t for which weather observations at time t are relevant. We will denote these numbers respectively b and f\(^1\). Therefore, we have to extend the discretization of the time axis to time periods before the first congested time period T\(_1\).

We consider a particular instance of the problem related to the operations of destination airport Z during a congested time span that has been subdivided into P time periods T\(_1\), T\(_2\), ..., T\(_P\) with a final time period T\(_P+1\) with infinite capacity. Each time period T\(_i\) represents a half-open time interval \([t_i, t_{i+1})\) of fixed length PERL = \(t_{i+1} - t_i\), for all \(i = 1, ..., P\). The origin of the time axis (\(t=0\)) is determined so that the earliest scheduled departure time of any flight scheduled to arrive at airport Z during T\(_1\), T\(_2\), ..., T\(_P\) (i.e. between \(t_1\) and \(t_{P+1}\)) falls in the interval \([0, \text{PERL}]\). We denote with \(s\) the (negative) index of this time period \((T_s = [0, \text{PERL}); t_s = 0 \text{ and } t_{s+1} = \text{PERL})\) thereby extending the discretization of the time axis to negative indices to define **generalized time periods** T\(_s\), T\(_{s+1}\), ..., T\(_{2}\), T\(_{-1}\), T\(_0\), T\(_1\), T\(_2\), ..., T\(_P\), T\(_{P+1}\) (see Figure 3-1). For each flight F\(_i\) we can therefore determine D\(_i\), the index of the time period during which flight i is scheduled for departure. For any time t between 0 and \(t_{P+1}\) we can then define the corresponding generalized time period T\(_t\) to be the time period that time t falls into according to the extended discretization. (i\(_t\), the index of this time period, is such that t belongs to the half-open interval \([t_{i_t}, t_{i_t+1})\).

\(^1\) This definition implies that b\(=\)f. However, it will be convenient to use distinct symbols for the purposes of the discussion that follows.
It is useful to define a hypothetical Perfect Capacity Forecast System (PCFS) that will play a central role in our discussion and against which all other forecast systems will be measured. This PCFS corresponds to the state of perfect information in which we assume that all parameters relevant to capacity forecasting are taken into account and measured. The basic components of this perfect information state are therefore individual complete observations \( \mathcal{O}_i \) for each time period \( T_i \). Each individual complete observation \( \mathcal{O}_i \) consists of the complete set of measurements for time period \( T_i \); this set could conceivably include measurements of airport capacity, measurements of local as well as global weather parameters (wind velocity and direction, barometric pressure, temperature, visibility etc...), time of the day, etc... A complete observation \( \mathcal{O}(t) \) at time \( t \) is a sequence of individual complete observations for the \( b \) time periods before and including \( T_{b+1}, \mathcal{O}_{b+1}, \mathcal{O}_{b-1}, \ldots, \mathcal{O}_{b+1} \). Associated with this complete observation is a

\[ \text{Figure 3-1} \]

\[ ^2 \text{In the sense that no parameter relevant to capacity forecasting has been left out.} \]
probabilistic forecast of the capacities, $K_{t+1}, K_{t+2}, \ldots, K_{t+f}$, for the $f$ individual time periods after $T_t$. We will assume that $f > s + P$ so that a forecast for $K_{t+1}, K_{t+2}, \ldots, K_{t+f}$ will always include a forecast for $K_1, K_2, \ldots, K_P$, i.e. for all the time periods of interest.

The PCFS could be constructed as follows:

- Given some appropriate discretization of the possible capacity values\(^3\) we have a finite number of capacity scenarios for the evolution of the capacities $K_{t+1}, K_{t+2}, \ldots, K_{t+f}$. Assume that we have identified $D$ such scenarios that we will denote $\mathcal{X}^1, \mathcal{X}^2, \ldots, \mathcal{X}^D$, each scenario $\mathcal{X}^i$ being a sequence of capacities $\mathcal{X}^i = (K_{1i}, K_{2i}, \ldots, K_{fi})$. In this context a probabilistic capacity forecast is a $D$-tuple $(p_1, p_2, \ldots, p_D)$ where $p_i$ is the probability of occurrence of scenario $\mathcal{X}^i$ ($\sum p_i = 1$). We note that if we chose in advance to limit the number of scenarios that would be considered in practice to a predetermined number which is less than the one obtained from a finest grain discretization, we would have to use some statistical method for data clustering (minimizing a given statistical measure of error) to partition this finest grain set into a coarser collection with the predetermined cardinality.

- We also assume that some appropriate statistical data clustering method has been utilized to identify a predetermined number, $L$, of possible complete observations $\mathcal{CO}^1, \mathcal{CO}^2, \ldots, \mathcal{CO}^L$ (each $\mathcal{CO}^j$ is a set of $b$ individual observations $\mathcal{CO}^j = (\mathcal{CO}_{1j}, \mathcal{CO}_{2j}, \ldots, \mathcal{CO}_{bj})$).

---

\(^3\) The greatest number of cases would be obtained if we use the finest grain discretization that consists of one-unit-increments of capacity.
- We assume that we have also maintained a data base of the history of capacity evolutions at airport Z given complete observations. This data base associates with each possible observation $\mathcal{O}_j$ a frequency of occurrence $p_j$. This data base also associates a probabilistic forecast $(p_{1j}, p_{2j}, ..., p_{DJ})$ with each observation $\mathcal{O}_j$ by simply keeping count of the frequencies of occurrence of each capacity scenario when observation $\mathcal{O}_j$ was made. In this context, at any time $t$, $p_j$ is the probability of scenario $\mathcal{X}_i = (K_{t+1}, K_{t+2}, ..., K_{t+t})$ occurring when the observations for the $b$ time periods before and including $T_1$ correspond to the complete observation $\mathcal{O}_j$. Although the perfect system we are describing is hypothetical, so that the numbers $(p_{1j}, p_{2j}, ..., p_{DJ})$ are not available, it is easy to see how such an inference system can be implemented and updated on the basis of less than complete observations. We will assume that this is how any forecast system is built.

We referred to the above forecast system as perfect in the following sense: Assume that in the future we are able to recognize a given observation $\mathcal{O}_j$ a large number $N$ of instances. If we keep count of the number of times $N_i$ among $N$ that any given capacity scenario $\mathcal{X}_i$ occurs, then we have the property that $\frac{N_i}{N} \to p_j$ as $N \to \infty$.

We could have defined our hypothetical perfect forecast system as one that, given a complete observation, would yield a deterministic forecast. We have refrained from doing so for an important reason: we believe that such a system is more than hypothetical; it is impossible and misleading. The main use of our perfect system is to provide a reference for evaluating real systems such that as the precision of real forecast systems improves we want the performance of these systems to approach asymptotically that of the perfect system. There is convincing evidence towards the fact that this asymptotic behavior is not deterministic but can only be described probabilistically. A
major part of the argument has to do with the chaotic nature of weather patterns, particularly in a local framework. A second argument is based on the theoretical (absolute) limitations of computational power; this kind of argument, setting limitations on the size and speed of physically possible computations, is usually based on the theory of quantum mechanics. Any realistic set of equations that can be used to model the interactions among weather parameters is likely to be an enormously complex, highly nonlinear set of differential equations leading to a chaotic solution behavior. The complexity (in the information-theoretical sense\(^4\)) of these computations coupled with the absolute limitations in computational speed are likely to be such that no deterministic forecast will ever be possible. (This corresponds to the case where running a solution to these equations on the fastest computers takes as much time as observing the events unfold in real time.)

Our hypothetical perfect forecast system is therefore a special one, in the sense discussed above. Any other forecast system can be defined relatively to this perfect forecast system by assuming that the number of parameters that go into an individual observation \(\Sigma\) associated with this system is a subset of the complete set of parameters that go into any complete individual observation \(\mathcal{O}\).

Let us see now how we can define and evaluate any given forecast system on the basis of the PCFS. A consequence of the fact that a forecast system other than the perfect forecast system does not consider complete observations is that it cannot differentiate between complete observations that share the same values for the subset of parameters that go into \(\Sigma\). It will therefore produce a forecast that aggregates all these cases and that

---

\(^4\) The size of the smallest algorithm that can be used to solve the equations.
carries less information. Let us see how this occurs more precisely and assume that a
given forecast system makes an (incomplete) observation $\Omega^i = (\Omega_1^i, \Omega_2^i, \ldots, \Omega_q^i)$ to
which correspond a set of complete observations $C\Omega_{i1}, C\Omega_{i2}, \ldots, C\Omega_{iQ}$ that share the
same values for the subset $\Omega$ (i.e., the event $\Omega^i$ is the union of the disjoint events
$C\Omega_{i1}, C\Omega_{i2}, \ldots, C\Omega_{iQ}$; $\Omega^i = \bigcup_{j=1}^Q C\Omega_{ij}$). The conditional probability of occurrence
of the complete observation $C\Omega_{ij}$, given $\Omega^i$, $q^i$, is given from the $p^{ik}$'s defined
previously ($p^{ik}$ is the a priori probability of making the observation $C\Omega_{ik}$) by:

$$ q^i = \text{Prob}(C\Omega_{ij} | \Omega^i) = \frac{P(C\Omega_{ij} \cap \Omega^i)}{P(\Omega^i)} $$

$$ \Rightarrow \quad q^i = \frac{\frac{P(C\Omega_{ij})}{Q}}{\sum_{k=1}^Q \frac{P(C\Omega_{ik})}{Q}} = \frac{p^{ij}}{\sum_{k=1}^Q p^{ik}} $$

Recall that according to the PCFS, each complete observation $C\Omega_{ij}$ has an
associated forecast ($p_{1ij}$, $p_{2ij}$, ..., $p_{Qij}$) that is based on actual frequencies of occurrence of
the different capacity scenarios. When our imperfect system will have come across a
large number of instances of observation $\Omega^i$ it will have produced an imperfect forecast
since it will have seen the capacity scenario $\chi^h$ occur with frequency

$$ P(\chi^h | \Omega^i) = \frac{P(\chi^h \cap \Omega^i)}{P(\Omega^i)} = \sum_{j=1}^Q P(\chi^h | C\Omega_{ij}) \cdot P(C\Omega_{ij} | \Omega^i) = \sum_{j=1}^Q p_{hij} \cdot q^i $$

There is an alternative way to evaluate a less than perfect forecast system relatively
to the perfect system. It consists of introducing a margin of error associated with a
forecast system in the following way:

We assume that a given forecast system making an observation $\Omega$ is subject to
error in measuring the value of some parameters. In this case, assuming that $C\Omega^i$ is the
complete observation that the PCFS would have made in the same situation, there is a
probability \( p_{ij} \) that the given forecast system mistakes \( O_j \) for \( O_i \) when it makes its observation and infers the wrong forecast \( (p_{1j}, p_{2j}, \ldots, p_{Dj}) \) instead of \( (p_{1i}, p_{2i}, \ldots, p_{Di}) \) (therefore \( p_{ii} \) is the probability that the forecast system is right given \( O_i \)). With this formulation it is possible to compute the cost of this error by computing the cost of the ground holding policy associated with the first (wrong) forecast if the second forecast materializes and comparing this cost with the cost of using the optimal GHP associated with the (correct) first forecast. This is the method we will use in a numerical example in section 5.3.4.

Let us now examine a couple of particular forecast systems that will play a role in future discussions.

The no-information forecast is the one that corresponds to an empty set of observations \( (O = \emptyset) \). This system therefore merely consists of the database that keeps track of all capacity scenarios along with their relative frequencies of occurrence. All other forecast systems are versions of this system which are conditional on some set of observations as we now illustrate with the next more sophisticated forecast system.

The capacity-based forecast system is the one that corresponds to a set of individual observations reduced to one element: the airport capacity observed in each time period \( (O = \{K\}) \). This is the simplest non-trivial forecast system; it is in fact obtainable from the database associated with the above no-information forecast by keeping track of, for each forecast of the next \( f \) capacities, the capacity of the preceding \( b \) time periods.

The result is a conditional pmf for \( f \) future capacities given \( b \) past capacities \( P(K_{t+1}, K_{t+2}, \ldots, K_{t+f} | K_t, K_{t-1}, \ldots, K_{t-b+1}) \). It is also easy to see how this crude minimal forecast system can be greatly improved upon in practice by supplementing the capacity information with weather information resulting in forecasts of the form.
We build on the extension of the discretization of the time axis from the previous section (see Figure 3-1) and we define a **Situation Function** to be a function \( S \) with domain \([0,t_{P+1}]\). The value of the function \( S \) at any time \( t \) is called a **situation at time \( t \)** and is a pair \( S(t) = (s_{at(t)}, P_{K}(t)) \) defined as follows:

- The first element of the pair is a function \( s_{at(t)} \) with domain \( FL \), the set of all flights scheduled at airport \( Z \) during \( T_{1}, T_{2}, \ldots, T_{p} \) (\( FL = \{F_{1}, F_{2}, \ldots, F_{N}\} \)), and range \( \{0,1,2\} \). The value of this function for a given flight \( F_{i} \), \( s_{at(t)}(F_{i}) \), indicates whether, at time \( t \), flight \( F_{i} \) is still on the ground at the airport of origin (\( s_{at(t)}(F_{i}) = 0 \)), is airborne between the airport of origin and airport \( Z \) (\( s_{at(t)}(F_{i}) = 1 \)), or has already landed at airport \( Z \) (\( s_{at(t)}(F_{i}) = 2 \)). Note that if a flight \( F_{i} \) is still on the ground at time \( t \) (\( s_{at(t)}(F_{i}) = 0 \)) we can determine whether it has been delayed or it is not eligible for take off since we know \( D_{t} \) the index of the time period during which it is scheduled for departure.

- The second element of the pair is a joint probability mass function \( P_{K}(t) = P_{K_{R_{t}}}, \ldots, P_{K_{P}}(t) \) for the capacities \( K_{R_{t}}, K_{R_{t+1}}, \ldots, K_{P} \) of the "remaining" time periods at time \( t \), \( T_{R_{t}}, T_{R_{t+1}}, \ldots, T_{p} \). The index \( R_{t} \) is defined to be 1 if \( t \) is such that the index \( I_{t} \) is less than or equal to 0, or to be the index \( I_{t} \) if \( I_{t} \) is greater than or equal to 1 (see also Figure 3-1). Note that a forecast system (as described in the previous section) determines such a joint probability mass function \( P_{K}(t) \) through an observation \( \Theta(t) \) at time \( t \).

A **Static Generalized Ground Holding Policy** (SGGHP) can be viewed as a sequence of values of the first component function of the situation function.
\([\text{stat}_0, \text{stat}_{t_i+1}, ..., \text{stat}_{t_p}, \text{stat}_{t_p+1}]\)\(^6\) that satisfy certain logical constraints of feasibility. (For example, we could not have \(\text{stat}_t(F_i) > 0\) for \(t\) less than the scheduled departure time of \(F_i\), or \(\text{stat}_t(F_i) > \text{stat}_{t'}(F_i)\) for \(t' > t\).) The appellation "static" will be justified shortly.

A _static optimization at time \(t\) _is an algorithm that associates to a history of situations up to time \(t\), \([S(u) = \{\text{stat}_u, P_K(u)\}_0 \leq u \leq t}\), a SGGHP after time \(t\) \([\text{stat}_{t+1}, \text{stat}_{t+2}, ..., \text{stat}_{t_p}, \text{stat}_{t_p+1}]\).\(^7\) Note that such a SGGHP can influence ground holds (\(\text{stat}_t\) changes from 0 to 1) as well as airborne delays (\(\text{stat}_t\) changes from 1 to 2). (This is why we refer to this as a "generalized" ground holding policy.)

We also note that, in general, we need the full history of situations up to time \(t\) if we are using costs as a criterion for evaluating ground holding policies since, for example, future delay costs for an airborne flight could depend on the ground hold imposed on that flight before take-off.\(^8\) Therefore the SGGHP \([\text{stat}_{t+1}, \text{stat}_{t+2}, ..., \text{stat}_{t_p}, \text{stat}_{t_p+1}]\) is

\[\quad\]

\(^6\) According to our approach of discretizing the time axis the values of the function \(\text{stat} \) at the time instants \(0, t_i + 1, ..., t_p + 1\) determine completely the results of a ground holding strategy in terms of costs.

\(^7\) We start the sequence at \(t_{i+1}\) since we will assume that it is too late at time \(t\) to affect events of time period \([t+t_{i+1}\). In other words, we assume that the scope of any optimization starts in the next time period.

\(^8\) Again, according to our approach, it is sufficient to consider the sequence \([\text{stat}_{t_i}, \text{stat}_{t_i+1}, ..., \text{stat}_{t_i-1}, \text{stat}_{t_i}]\) to be able to compute future costs.
obtained (as the result of the optimization at time t) by considering a joint probability mass function $P_{K_{R_1},...,K_P}(t)$ and minimizing expected delay costs.

We used the term "static" in the above definitions because, at any time $t$ at which we apply the optimization, the SGGHP can be expressed as a fixed set of assigned ground delays, $\{X_1, X_2, ..., X_N\}$, where $X_i$ is the ground delay imposed on flight $F_i$, that only depend on $P_K(u)$ for $u \leq t$. Suppose now that at time $t$, in addition to a capacity forecast $P_K(t)$, we have access to a model of the updating of the capacity forecast with time. A forecast system constitutes such a model since it can be thought of as a function that associates to an observation at time $t$, $\mathcal{O}(t)$, a probabilistic forecast $P_{K_{R_1},...,K_P}(t)$.

We can then define a Dynamic Ground Holding Policy (DGHP) to be a set of ground delays, $\{X_1(\mathcal{O}(t_{P_1})), X_2(\mathcal{O}(t_{P_2})), ..., X_N(\mathcal{O}(t_{P_N}))\}$, where the ground delay for each flight $F_i$ now depends on the observation made at time $t_{P_i}$, the beginning of the time period during which flight $F_i$ is scheduled to take-off. In Section 4.3 we develop a Stochastic Dynamic Program that can be used to find an optimal DGHP.

It is important to note, however, that a DGHP can only be optimal if we assume that the ground holds cannot be revised once they have been determined. In practice, this assumption corresponds to a situation for which once the ATC system has issued a ground hold directive for a flight it cannot revise this ground hold. A "fully dynamic" GHP, on the other hand, is one for which the ground hold for a flight $F_i$ is expressed not as a single variable $X_i(\mathcal{O}(t_{P_i}))$ but rather as a sequence of zero-one variables $\{x_i(\mathcal{O}_j); \text{for all } j \geq P_i\}$ where, for example, $x_i(\mathcal{O}_j) = 1$ indicates that flight $F_i$ is allowed to take-off during time period $T_j$ if observation $\mathcal{O}_j$ is made. This expresses the fact that, in a "fully dynamic" situation, we take advantage of additional information after each time period. Appendix A contains a mathematical formulation of the "fully dynamic" GHPP.
The dynamic version of the probabilistic GHPP is obviously a difficult one to deal with. As was stated in Chapter 1, the approach taken in this thesis is to decompose the problem; we will therefore concentrate on algorithmic solutions to the static version in the hope of being able to extend these solutions to the dynamic version. In the next chapter we will see that we can indeed propose solution approaches to the dynamic version built on "static" algorithms. We will however limit the numerical examples, which are the subject of the Chapter 5, to the static case. One reason is that an essential component of the dynamic version is the model for the updating of information with time; the implementation of the "dynamic" algorithms we will propose cannot be illustrated without this component. We will still discuss the dynamic version in this and subsequent chapters. The following Section 3.3, for example, is devoted to issues related to the dynamic aspect.

3.3 Dynamic Implementation

The simple capacity-based forecast system described in Section 3.1 shows clearly that (partial) additional information is available after each time period in the form of the capacity observed for that time period resulting in a new conditional pmf (a new forecast) for the future capacities. This is in fact true for any forecast system since such a system would necessarily include the capacity information among the variables making up an individual observation associated with it. It might however be impractical to assume that we are in a position to take advantage of this additional information, as it becomes available, to update a given holding policy after each time period. Such a scenario however is certainly conceivable and desirable; it requires the automation and integration of three processes:
(i) A background information gathering and updating system. This system is responsible for keeping track of all the relevant information concerning the status of all flights scheduled at airport Z. Namely, at any time t, it compiles a history of status functions \( \{ \text{status}_t \}_{0 \leq t} \) as defined in the previous section. From such a status history the system determines all cost functions that will serve as input for the optimization loop ((iii) below). This is likely to be the most costly component of our complete system. The backbone of such an information gathering and updating system already exists in the Advanced Traffic Management System (ATMS) being developed by the FAA. This sophisticated traffic monitoring system collects real-time information about flights at different regional centers which is relayed through satellites. It gathers detailed data such as aircraft identification, destination, altitude, speed and position. It has the ability to update this database every few minutes; it can also sort the information according to pre-specified criteria and uses interactive graphics computer screens for displaying it. It is easy to see how such a system can be built upon by, for example, linking it to an airline reservation system to make available more precise information such as aircraft load factors, number of transferring passengers on board a given flight, etc.... This information could be used to assign more accurate costs to flights on a real time basis. It is even conceivable that airlines would be willing to provide crew and aircraft scheduling information if they knew that in exchange for this information the ATC system would be more responsive to their scheduling needs and constraints. This scenario is likely to result in a bargaining and control process to monitor the validity of the information provided by airlines as these will learn to utilize (and take advantage of) the system. But the net result would be a desirable constant dialogue between the airlines and the FAA that promotes the (valid) idea that some central planning is necessary to use efficiently common limited resources.
(ii) A forecast system that queries the database compiled by the first system to issue a probabilistic forecast on demand. This system is therefore responsible for determining at any time $t$ the second ingredient of the situation function defined in Section 3.1, namely $P_K(t)$.

(iii) An optimization system that considers the information from the first system together with the forecast issued by the second system, namely a history of situations $\{S(u) = \{status, P_K(u)\}\}_{0 \leq u \leq t}$ and a trajectory of the system up to time $t$, to output an updated GGHP. The results of this optimization loop that we choose to implement are in turn fed back into the background information updating system to update the values of cost variables, status variables, etc.

Such an integrated system could conceivably operate continuously and should allow manual override of all its components. It is important to note that this system affects both the ground holds as well as the airborne holds imposed on flights. It would, therefore, have to be complemented by and interfaced with whatever landing sequencing system is used by the control tower of airport Z.

3.4 Cost Functions

The issue of assigning costs to individual flights is a delicate one. The major difficulty stems from the fact that each airport Z is likely to be utilized by many different users (the different airlines and general aviation) which makes the issue of equitability of delay assignment, or "distributive effects", an important one.
Dealing with these distributive effects boils down, in one way or another, to determining a "fair" allocation of available capacity at airport Z among users. In the case of a deterministic capacity forecast this can be done, for example, by subdividing the future (known) airport capacity among the airlines in proportion to the level of utilization of airport Z by these airlines. The probabilistic case can be treated by considering each possible capacity case (for any given probabilistic forecast) separately and allocating that capacity among users as would be done in the deterministic case. The result is a probabilistic forecast for the capacity allocated to each user. The effect of such capacity allocation schemes is to generate a number of ground holding policy subproblems, each subproblem involving a single user.

There may be no need however for such extreme decomposition of the problem if the usage patterns of all the users are "identical", meaning that all users (all airlines) have an almost identical "fleet mix" as well as an almost identical proportion of long-haul versus short-haul flights using airport Z. One of the reasons for this was touched upon in Chapter 2 and will be confirmed by the numerical examples in Chapter 5: the assignment of ground holds that minimizes "direct" monetary delay costs (fuel + crew + depreciation costs) tends to penalize smaller aircraft by (naturally) imposing a disproportionate amount of delays on them. The other reason is one related to the dynamic character of the probabilistic formulation: in order to take full advantage of the pattern of resolution of uncertainty though time, a "dynamic" algorithm will tend to penalize short-haul flights (see also Chapter 4). Now, if all airlines using airport Z have approximately the same aircraft-mix and ratio of long-haul to short-haul flights a GHP that minimizes direct operating costs will not discriminate against any particular airline.
Suppose now that this is not the case; for example suppose that some airlines have a large proportion of small aircraft using airport Z. A way to insure that aircraft with lower delay costs are not kept on the ground a disproportionate amount of time would be, as we saw in Chapter 2, to use cost functions of the type \( C_i(1+\alpha)D \) (where \( D \) is the delay imposed on flight \( F_i \)) with a large enough cost increase coefficient \( \alpha \). Another situation for which we might want to take corrective action is the case in which some airlines operate a large proportion of short-haul flights; we will see in Chapter 4, when we discuss the dynamic aspect of the probabilistic formulation, that there are ways to compensate for the inherent bias of a "dynamic" optimization algorithm against short-haul flights.

It is implicit in the above discussion that it is acceptable to deal with distributive effects at the "airline level" in the sense that we have not worried about distributive issues within the set of flights of a given airline. We therefore have to investigate to what extent it is acceptable to assign delays within a given airline according to direct costs. In order to do that, we can assume that we are dealing with a single airline using airport Z (or equivalently that we are dealing with one of the component subproblems resulting from an acceptable allocation of capacity among airlines). It is also reasonable to assume that the management of that airline can evaluate its preference with respect to delay scenarios (this only requires that, faced with two alternative scenarios of delays, the management can either decide which one is preferable or decide that they are equivalent scenarios). Then it is a classical result that, if that preference relation satisfies some (reasonable) consistency axioms, there exists a utility function on the space of possible delay scenarios that represents the preference relation. In this case the approach of minimizing an expected utility is justified. Furthermore, since we are dealing with a situation for which the experiment is repeated a large number of times (every time we have to assign ground
holds), it can be argued that the utility functions to be used should reflect closely the
direct costs of delays. This idea can be illustrated by the following simple setting. A
person is asked about the minimal amount of monetary compensation $M$ that he would
require for accepting to play a lottery as follows: a fair coin is tossed, if the outcome is
heads the person loses $1,000, if it is tails no loss is incurred. If the lottery is to be
played only once the (risk averse) person would probably ask for a compensation $M$ that
exceeds $500, the expected monetary loss. In this case the utility values of the outcomes
of the lottery have to be different from their monetary values ($-1,000$ and $0$) to reflect
this preference. If, on the other hand, the person is assured that the lottery will be played
a large number of times, say 100 times, that person is likely, because the variance of the
average loss is diminished, to accept a compensation close to actual expected monetary
loss even though he might be risk averse. This situation is quite similar to the case of a
single airline facing the possibility of delays if the goal is to minimize operating costs in
the long run.
4. PROBABLISTIC CASE: ALGORITHMIC APPROACHES

In this chapter we propose a number of exact or heuristic algorithms for the probabilistic GHPP. The algorithms developed in the first two sections are concerned with the static version of the probabilistic case. We first develop a Dynamic Programming (DP) algorithm that yields the exact (static) solution to the problem but that suffers from exponential (in the number of time periods) time and space complexities. The second section proposes some heuristics for the same static probabilistic formulation; some of these heuristics use the fast algorithm developed for the deterministic case as a building block. These heuristics will be used for the computational examples of Chapter 5. The final section of the chapter is concerned with the dynamic version of the probabilistic problem. After showing how we can extend the formulation and the DP to deal with the dynamic problem we show how the heuristic approaches to the static case, and therefore the fast deterministic algorithm as well, can be used to build heuristics for the dynamic case.

4.1 Dynamic Programming

4.1.1 Description

The Dynamic Program (DP) we will develop is an extension of the DP used by Andreatta and Romanin-Jacur (1987). As was stated in the introductory chapter the Andreatta/Romanin-Jacur formulation assumes that congestion occurs at the destination airport Z during a single time period for which we consider a probabilistic forecast $p_X(k)$
for the capacity $K$. We will extend this formulation to cases for which we have a probabilistic forecast for the capacities $K_1, K_2, \ldots, K_p$ of several consecutive time periods $T_1, T_2, \ldots, T_p$.

A major characteristic of the following DP formulation and of the heuristics to follow is that they determine ground holds for aircraft in a way that is insensitive to the time of departure of each individual flight. This corresponds to the case we referred to in Chapter 3 as the static probabilistic case. The implications of this restriction will be fully discussed in section 4.3.4 when we discuss the dynamic version; it suffices to state now that this static version will not only provide a good basis for discussing the dynamic version but, as stated earlier, will provide the basis for solution methods for the dynamic version as well.

Another major characteristic of the DP formulation is that it assumes the existence of a fixed landing priority rule for aircraft arriving at airport $Z$. To discuss this and formulate the DP algorithm we need to recall and augment some of the notation used in Chapter 3:

We consider landing operations at an arrival airport $Z$ during consecutive time periods $T_1, T_2, \ldots, T_p$. $N$ flights, $F_1, F_2, \ldots, F_N$, are scheduled to land during $T_1, T_2, \ldots, T_p$. For each flight we define:

- $C_{g_i}(x) =$ cost of delaying flight $F_i$ for $x$ time periods on the ground prior to departure.

- $C_{a_i}(x, y) =$ cost of delaying flight $F_i$ for $y$ time periods in the air when it has already been delayed $x$ time periods on the ground (for a total delay of $x+y$ time periods).
- $P_i =$ index of the earliest possible landing time period (scheduled arrival) for flight $F_i$.

At some time $t$ before the earliest scheduled take-off time of any of the $N$ flights we have available a probabilistic forecast $P_{K_1, K_2, ..., K_P}$ for the capacities $K_1, K_2, ..., K_P$ of the $P$ time periods $T_1, T_2, ..., T_p$. The static probabilistic ground holding policy problem is defined as finding the vector $(X_1^*, X_2^*, ..., X_N^*)$, where $X_i^*$ is the ground hold imposed on flight $F_i$, that minimizes the total expected delay cost.

An exact optimal solution to the above static-probabilistic version of the ground-holding strategy problem can be found through a Dynamic Program under the assumption that a fixed landing priority rule $\Pi_1, \Pi_2, ..., \Pi_N$ is imposed on the flights $F_1, F_2, ..., F_N$. By a fixed priority rule we mean that, if two flights $F_i$ and $F_j$ are candidates for landing during the same time period, $F_i$ will not be cleared for landing before $F_j$ if $\Pi_i < \Pi_j$.

For the remainder of this section we assume that the flights have been reordered so that $\Pi_{i+1} > \Pi_i$ for all $i$.

We note that the assumption of a fixed priority rule is not particularly restrictive in practice for two reasons:

(i) Air traffic controllers tend to use such rules in practice when sequencing flights for landing. Priority by scheduled time of arrival is an example of a landing priority rule used in practice.

(ii) If we use (ground and airborne) delay cost functions that satisfy the regularity conditions (see section 2.2) then we can impose such a fixed priority rule with no loss of optimality (i.e. we can find a priority rule such that the solution of the Dynamic Program

63
is in fact an optimal solution for the general formulation with no landing-priority constraint.) The reason for this is that, as will be demonstrated in section 4.2.5, the "optimal" landing strategy problem (i.e. finding the sequencing of landing aircraft that minimizes total air delay cost for a given capacity case and a given ground holding policy) can be formulated as a deterministic ground holding problem. But we saw in section 2.2 that, given "regular" cost functions, the optimal solution to this problem consists of determining an "optimal" ordering for assigning flights to available capacity. Furthermore this "optimal" ordering does not depend on the airport capacities. This "optimal" ordering translates into a fixed landing priority rule for landing aircraft when airborne cost functions satisfy the regularity conditions. Also note that for the single time period case such an "optimal" landing strategy always exists; it corresponds to ordering flights according to the cost of airborne (one time period) delay. (The same observation is made by Andreatta and Romanin-Jacur (1987).)

The DP algorithm is based on two observations:

A) The expected (ground plus air) delay cost for a flight \( F_i \), \( C_i(X_i; H_i) \), depends only on two quantities:

(i) \( X_i \) = the ground hold for flight \( F_i \).

(ii) the vector \( H_i = (H_i^1, H_i^2, ..., H_i^P) \) where \( H_i^j \) represents the number of flights with priority greater than \( H_i \) that are assigned to arrive during time-period \( T_j \).

B) The "optimal" (minimum) value of the expected delay costs for the first \( i \) flights \( (F_1, F_2, ..., F_i) \), \( G_i(H_i) \), depends only on \( H_i \) and is given by the following recursion formula:

\[
G_i(H_i) = \min_{X_i} \left\{ C_i(X_i; H_i) + G_{i-1}(H_{i-1}(H_i; X_i)) \right\}
\]  

(2.1)
where the functional relationship $\text{Hi-1}(\text{Hi},X_i)$ is defined as follows:

- For time period $T_{P_i+X_i}$ to which flight $F_i$ is reassigned set

  $$\text{Hi-1}_{P_i+X_i} = \text{Hi}_{P_i+X_i} + 1.$$  This expresses the fact that the number of flights with higher priority than flight $F_{i-1}$ scheduled during time period $T_{P_i+X_i}$ has to be increased by one unit if flight $F_i$ is reassigned to that time period.

- For all other time periods $T_R$ with $R \neq P_i + X_i$ set

  $$\text{Hi-1}^R = \text{Hi}^R,$$  expressing the fact that the number of flights with higher priority than flight $F_{i-1}$ scheduled during time period $T_R$ does not change if $F_i$ is reassigned to another time period.

To see why A) is true consider the two terms that make up $C_i(X_i, \text{Hi})$:

$$C_i(X_i, \text{Hi}) = C_{g_i}(X_i) + \sum_{j=1}^{C} C_{a_i}(X_i, Y_{i,j}) p_j$$

(2.2)

The first term, $C_{g_i}(X_i)$, corresponds to the cost of delaying flight $F_i$ $X_i$ time periods on the ground. The second term, $\sum_{j=1}^{C} C_{a_i}(X_i, Y_{i,j}) p_j$, is the expected airborne-delay cost for flight $F_i$. It is computed by finding, for each possible capacity case $\mathcal{X}^j = (K_1^j, K_2^j, ..., K_P^j)$ of the probabilistic capacity forecast $P_{K_1,K_2,...,K_P}$, the airborne delay $Y_{i,j}$ for flight $F_i$ corresponding to that case. More precisely, the parameters of equation (2.2) are defined as follows:

- $C =$ the number of capacity cases distinguished in the probabilistic forecast $P_{K_1,K_2,...,K_P}$. In the notation of Section 3.1, the $C$ capacity cases (or scenarios) are $\mathcal{X}^1, \mathcal{X}^2, ..., \mathcal{X}^C$, each case $\mathcal{X}^i$ being a sequence of capacities $\mathcal{X}^i = (K_1^i, K_2^i, ..., K_P^i)$. 

65
• $p_j = \text{the probability of capacity case } \mathcal{K}_j \text{ occurring. } (\sum_{j=1}^{C} p_j = 1)$

• $k_{ji} = \text{index of the actual landing time period for } F_i \text{ for the } j\text{th capacity case. If we denote by } L_{jh} \text{ the number of flights with higher priority than } F_i \text{ that are still waiting for landing at the beginning of time period } T_h \text{ (for the } j\text{th capacity case), we can compute } k_{ji} \text{ through the following formula:}$

$$k_{ji} = \min \{ m \mid m \geq P_i + X_i \text{ and } H_i^m + L_{j m-1} < K_{mi} \}, \text{ where we set } L_{j0} = 0$$

and we compute the $L_{jh}$'s recursively through:

$$L_{jh} = \max \{ H_i + L_{j h-1} , 0 \}.$$

These relationships express the fact that flight $F_i$ will only be able to land during the first time period after $T_{P_i+X_i}, T_{k_{ji}}$, for which the cumulative capacity is such that all flights with priority higher than $\Pi_i$ that are scheduled before $T_{k_{ji}}$ have been able to land.

• $Y_{ji} = \text{airborne delay for flight } F_i \text{ when capacity case } K_{1i}, K_{2i}, \ldots, K_{pi}$ materializes. $Y_{ji}$ is given by the formula:

$$Y_{ji} = k_{ji} - P_i - X_i.$$

We observe that, since the computation of $k_{ji}$ depends only on $H_i$, $Y_{ji}$ is also only a function of $H_i$. This justifies the notation $C_i(X_i, H_i)$ used in formula (2.2). More detailed descriptions of these calculations will be given in Chapter 5 when we discuss the computer subroutines used for computing the expected delay costs.

The DP that determines the ground holds $X_1^*, X_2^*, \ldots, X_N^*$ that minimize the total expected cost $\sum_{i=1}^{N} C_i(X_i, H_i)$ consists of two passes:
-Pass1 goes through all flights in ascending order (of priority) and finds, for all Hi, $X_i(H_i)$ that minimizes $G_i(H_i)$ according to the recursive formula (2.1).

-Pass2 goes through all flights in descending order (of priority) and uses the results of Pass1 to retrieve the optimal ground holds $X_1^*$, by computing successively the $H_i^*$'s that correspond to this optimal solution.

More precisely, the algorithm consists of the following steps:

**Step 1: initialization**

$G_0(H_0) = 0$ for all $H_0$

let $i = 1$

go to Step 2

**Step 2:**

for all $H_i$ do:

$$X_i(H_i) = 0$$

$$G_i(H_i) = C_i(0,H_i) + G_{i-1}(H_{i-1},0)$$

for $X = 1$ to $P-P_i+1$ do:

if $C_i(X,H_i) + G_{i-1}(H_{i-1}(H_i,X)) < G_i(H_i)$ then

$$X_i(H_i) = X$$

and $G_i(H_i) = C_i(X,H_i) + G_{i-1}(H_{i-1}(H_i,X))$

else continue (with: for $X = 1$...)

go to Step 3

**Step 3:**

if $i = N$ go to Step 4

else $i = i+1$; return to Step 2
Step 4:

Compute the optimal policy $X_1^*, X_2^*, ..., X_N^*$ as follows:

$X_N^* = X_N(\emptyset)$ (since $H_N^* = \emptyset$)

$X_{N-1}^* = X_{N-1}(H_{N-1}^* [\emptyset, X_N^*])$

$\vdots$

$X_{i-1}^* = X_{i-1}(H_{i-1}^* [H_i^*, X_i^*])$

$\vdots$

$X_1^* = X_1(H_1^* [H_2^*, X_2^*])$

END

The only assumption made in the above DP is that of the existence of a fixed landing priority rule. The DP provides an exact solution to the static probabilistic formulation even if we do not assume "regular" cost functions. Unfortunately, we are constrained in the use of this DP algorithm both by the time and the space complexity of Step 2. Let us examine these complexities in detail.

4.1.2 Time/Space Complexity

For the purposes of discussion let us define:

$N = \text{the total number of flights considered.}$

$P = \text{the number of time periods.}$
M = the maximum airport capacity for a single time period. For our purposes it is reasonable to assume that M is a number of order \( \frac{N}{P} \) \( O(\frac{N}{P}) \).\(^1\)

C = the number of capacity cases considered in the probabilistic formulation.

The main computational burden occurs during Pass1 of the algorithm where, for each flight and for each \( H_i \), we have to find \( X_i(\{H_i\}) \) that minimizes \( G_i(\{H_i\}) \). The cardinality of the state space for \( H_i \) is \( (M+1)^P \). There are, on the average, \( \frac{P+1}{2} \) steps involved in the search of \( X_i(\{H_i\}) \) for a given \( H_i \). Each one of these steps involves considering each capacity case separately and searching, on the average, \( \frac{P+1}{2} \) time periods for the actual landing period for the flight. The total time complexity of the Dynamic Program is therefore:

\[ O[ N \cdot C \cdot (P+1)^2 \cdot (M+1)^P ] . \]

The dominating factor in this expression is \( (M+1)^P = \left( \frac{N}{P} + 1 \right)^P \). We will therefore focus our analysis on this term. Considering that the number of flights, the length of the total time span considered, and the number of capacity cases are exogenous variables determined by the problem at hand and not amenable to modification, the only parameter we can use to control the time complexity is the number of time periods \( P \). We also know that \( \left( \frac{N}{P} + 1 \right)^P \) is an increasing function of \( P \) that tends to \( \exp(N) \) as \( P \to \infty \).

It is therefore natural to try to reduce \( P \). The extent to which we can reduce \( P \) is,

\[ \]

\[ ^1 \text{In fact } M \text{ is very likely to be close to } \frac{N}{P} \text{ if we assume that the schedule at airport } Z \text{ reflects roughly the airport's capacity. This is true at most congested airports. Since we are concerned here with computational complexity the magnitude of the multiplicative constant is of no importance.} \]
however, dictated by the level of detail to which we need to conduct the optimization; it does not seem reasonable to choose time periods of duration of more than one hour.

A typical instance of the problem could involve several hundred flights and a total time span of several hours (with a corresponding high number of time periods); the practicality of the exact DP approach is therefore limited and we will have to consider limited lookahead policies which will be described in the next section. Before doing this, however, it is important to note that our analysis of time complexity assumes that we are running the DP algorithm on a single-processor sequential machine. It should be observed that the algorithm actually consists of a large number of independent steps that could theoretically be carried simultaneously on separate processors. More precisely:

We observe that, for a fixed flight index \( i \), each of the \( (M+1)^P \) computations of \( X_i(H_i) \) in Step 2 of the algorithm depend only on the results of the same computations for flight index \( i-1 \). They could therefore be carried out simultaneously (on \( (M+1)^P \) parallel processors). Now each one of these computations is \( O(C(P+1)^2) \); the total computational complexity of the fully parallel algorithm is therefore \( O(NC(P+1)^2) \). If less than \( (M+1)^P \) processors are available, say \( Q \) processors, the complexity becomes \( O(NC(P+1)^2 \frac{(M+1)^P}{Q}) \). This type of parallelism at the algorithmic level\(^2\) is the easiest to implement: the algorithm itself has an intrinsically high degree of parallelism which can be directly translated into any of its implementations\(^3\).

\(^2\) As opposed to parallelism at the operational level, i.e., one that depends on the particular implementation of the algorithm.

\(^3\) This is also the case, for example, of the matrix multiplication algorithm.
It should also be noted that, if we assume a reasonably "stable" schedule for landings at airport Z, there is still another (complementary) way to get around the time complexity limitation: preprocessing. The idea is to carry out the first three steps of the algorithm "off-line" for each possible probabilistic forecast scenario and each possible landing schedule and store the resulting values of $X_i(H_i)$. In this way the only "real time" computation would consist, for a given day, of retrieving the results that correspond to the particular schedule and forecast of that day. Obviously the feasibility of this method depends on the "stability" of the schedule at airport Z and also on the "stability" of weather patterns as measured by the number of capacity scenarios that can occur at airport Z. Of course the "off-line" computations can be updated to account for changes in schedule when these become significant.

The storage space complexity of the DP algorithm is also an obstacle. It is $O(N(M+1)^P)$ corresponding to the storage requirement for the values $X_i(H_i)$ to be used by Step 4. However we observe that virtual memory techniques can also help with that respect. They consist of periodically storing intermediate results in disk-storage space thereby freeing the "active" memory (RAM) for computations. This process can occur at different stages of the computation as needed. For example, we could include instructions to store the values of $X_i(H_i)$ after each pass through Step 2 of the DP algorithm into a file (thereby creating one such file for each flight); in this way each call to Step 2 can use the same physical RAM space. If the space requirement for $X_i(H_i)$ is too large, even for a single index $i$, we can simply incorporate file-writing instructions earlier in Step 2; this method theoretically removes any space limit, albeit at the expense of time complexity since we are adding (a possibly large number of) instructions to Step 2 of the algorithm. The only other modification of the DP algorithm needed is to include instructions in Step 4 to retrieve the values of $X_i(H_i)$ from the corresponding files as
needed. Even though we have described precisely here how we can get around the space complexity by *explicitly* adding instructions to the algorithm it should be noted that some operating systems allow this type of memory management to be hidden from the programmer. In this case the original algorithm does not have to be modified by the programmer\(^4\) and can work as is, regardless of the available RAM space.

### 4.2 Heuristics

A typical instance of the GHPP could involve several hundred flights and a time span of several hours (with a corresponding high number of time periods \(P\)). As suggested in Section 4.1.2 the practicality of the exact DP approach is therefore limited and we have to consider heuristics for solving the probabilistic version of our problem. We have identified four groups of heuristics for evaluation:

- **Limited Lookahead Heuristics** based on the DP algorithm. The basic idea is to decompose the original problem involving \(P\) time periods into a sequence of subproblems, each involving a smaller number of time periods, for which we can use the DP algorithm.

- **Greedy Heuristics** which consist of finding the "optimal" ground hold for each flight individually in the order of a fixed landing priority rule. (Different landing priority rules lead to different ground holding policies.)

---

\(^4\) This type of memory management will be available on microcomputers shortly; the new operating system for the Macintosh\(^\text{TM}\), announced for the beginning of 1990, will include it.
The two groups of algorithms above require that a fixed landing priority rule be in effect. In contrast, the two groups we describe now can be applied to problems for which no such rule is in effect.

- The idea behind Equivalent Capacity Heuristics is to reduce the problem to a single instance of an "equivalent" deterministic problem. This is accomplished by trying to find, for each probabilistic capacity forecast, a single "equivalent" capacity case that best captures the "optimal" trade-off between ground delays and airborne delays.

- Equivalent Policy Heuristics, on the other hand, try to capture this "optimal" trade-off by solving a number of deterministic problems, corresponding to the different capacity cases distinguished in the probabilistic forecast $P_{K_1, K_2, \ldots, K_P}$, and finding an "optimal" mix of the resulting ground holding policies.

Most of the above heuristics compute a ground holding policy without computing the total expected delay costs associated with this policy. In order to compare the performance of these heuristics we will need a method for computing these costs; this issue will be the subject of Section 4.2.5.

### 4.2.1 Limited Lookahead

Limited lookahead consists of subdividing the set of time periods into smaller sets (of 3 time periods, for example) and running the DP algorithm on each subproblem consecutively, taking into account the flights "left over" from the previous subproblem (i.e. those landing in the "infinite capacity" final period relative to the previous subproblem). The motivation for doing this is a reduction in time complexity as well as
storage space requirements. Assume we use $Q$ such subproblems of $R$-time periods each ($P=QR$). The time complexity of each subproblem is $O[N(C(R+1))^2(M+1)^R]$ to be applied $Q$-times, leading to a total time complexity of $O[QNC(R+1)^2(M+1)^R]$ to be compared to $O[N(C(P+1))^2(M+1)^P]$ for the original DP. In terms of memory requirement, the $Q$ subproblems can use the same physical memory space (since they are applied in sequence) which reduces the space complexity from $O[N(M+1)^P]$ to $O[N(M+1)^P]$. The implementation goes as follows:

1. Apply an $R$-time-period version of the DP described in the previous section to the first $R$ time periods $T_1, T_2, ..., T_R$, for all flights scheduled to land at airport $Z$ during $T_1, T_2, ..., T_R$, using $P_{K_1, K_2, ..., K_R}$, and assuming $K_{R+1} = \infty$. The resulting assignments of flights to time periods $T_1, T_2, ..., T_R$ are final; flights assigned to time period $T_{R+1}$ are "carried over" to the next subproblem.

2. Apply the same size DP to time periods $T_{R+1}, T_{R+2}, ..., T_{2R}$, for all flights scheduled during $T_{R+1}, T_{R+2}, ..., T_{2R}$ as well as any flights that were assigned to time period $T_{R+1}$ by the previous sub-optimization. Here we use $P_{K_{R+1}, K_{R+2}, ..., K_{2R}}$ and we assume $K_{2R+1} = \infty$. Again, any assignment to time periods $T_{R+1}, T_{R+2}, ..., T_{2R}$ is final while the flights assigned to time period $T_{2R+1}$ are "carried over" to the next subproblem.

3. Continue this sequence of sub-optimizations until the final sub-problem which corresponds to time periods $T_{P-R+1}, T_{P-R+2}, ..., T_P$. All assignments are now final.

This limited lookahead strategy will be implemented using $R=3$ in Chapter 5. Figure 4-1 illustrates the process (with $R=3$).
Finally we note that, as is the case for the general DP formulation, Limited Lookahead Heuristics assume that a fixed landing priority rule is in effect and can be used with general cost functions.

4.2.2 Greedy Heuristic

Greedy Heuristics also work with general cost functions and assume that a fixed landing priority rule is in effect. The basic idea for these heuristics comes from observation A) of section 4.1.1, namely that, if we have a fixed landing priority rule, the
expected cost for flight $F_i$, $C_i(X_i, H_i)$ depends only on $X_i$ and the status of flights of higher priority represented by the vector $H_i$. Note therefore that to each priority rule corresponds a different heuristic.

Suppose that flights have been indexed so that $\Pi_{i+1} > \Pi_i$ for all $i$. By definition, the flight with highest priority, $F_N$, is such that $H_N = \emptyset$. Therefore, the expected (ground+air delay) cost for flight $F_N$, $C_N(X_N, \emptyset)$, does not depend on the status of any other flight; it depends only on the ground hold $X_N$ that we impose on it. Thus, we can find the "optimal" ground hold $X_N^*$ that leads to the lowest expected delay cost for $F_N$. Once we have $X_N^*$, we can compute $H_{N-1}^*[\emptyset, X_N^*]$ through the procedure outlined in Section 4.1.1 and we can therefore also compute, for each possible ground hold $X_{N-1}$, the expected cost for flight $F_{N-1}, C_{N-1}(X_{N-1}, H_{N-1}^*[\emptyset, X_N^*])$. Again this allows us to find the "optimal" ground hold $X_{N-1}^*$ for flight $F_{N-1}$ as the one that yields the lowest expected cost. This procedure is repeated until we have computed the "optimal" ground hold for the flight with lowest landing priority, $F_1$.

The Greedy Heuristic therefore computes ground holds by minimizing the expected cost for each flight individually, starting with the flight with highest priority. More precisely, the $N$ steps are:

**Step 1:**

Find $X_N^* = \arg \min C_N(X_N, \emptyset)$.

Compute $H_{N-1}^*[\emptyset, X_N^*]$ using the procedure outlined in section 4.1.1.

**Step 2:**

Find $X_{N-1}^* = \arg \min C_{N-1}(X_{N-1}, H_{N-1}^*[\emptyset, X_N^*])$. 

76
Compute $H_{N-2}^*(H_{N-1}^*, X_{N-1}^*)$.

\[ \vdots \]

- **Step $N+2:i$**

Find $X_{i-1}^* = \arg\min C_{i-1}(X_{i-1}, H_{i-1}^*, [H_i^*, X_i^*])$.

Compute $H_{i-2}^*(H_{i-1}^*, X_{i-1}^*)$.

\[ \vdots \]

- **Step $N$**

Find $X_1^* = \arg\min C_1(X_1, H_1^*[H_2^*, X_2^*])$.

We note that a major component of this algorithm is the computation, for a given flight $F_i$, of the expected cost associated with a given ground hold, $X_i$, and a given vector $H_i$. We saw in Section 4.1.1 that this computation consists of looking at each possible capacity case in $P_{K_1,K_2,...,K_P}$ to find the actual landing time (and therefore the airborne delay) for flight $F_i$ corresponding to that capacity case. This involves a search through, at most, $P+1$ time periods. The time complexity of this procedure is therefore $O[NCP(P+1)^2]$.

Note that, if we use "regular" (ground and air-delay) cost functions and use the resulting "optimal" landing priority rule, the Greedy Heuristic translates into a "myopic" strategy consisting of going after the highest possible marginal cost reduction at each step of the algorithm (thereby the appellation "Greedy"). The reason for this is that the "optimal" landing priority rule associated with a set of "regular" cost functions is such
that the flights with highest delay costs are given priority; therefore the Greedy Algorithm goes after the highest possible gains from the outset. All the numerical examples in Chapter 5 that use this algorithm will use "regular" cost functions. It is therefore the particular algorithm that corresponds to the "optimal" landing priority rule that will be referred to as the "Greedy Heuristic" in the remainder of the thesis.

In Chapter 5 we will see that this approach performs as well as the Limited Lookahead DP when we assume "regular" cost functions.

The next two heuristics are intended to take advantage of the Fast Algorithm developed for the deterministic version of the problem in section 2.2.

4.2.3 Equivalent-Capacity Heuristics

The strategy for this family of heuristics is to reduce the probabilistic problem to a deterministic one by reducing the probabilistic forecast \( P_{K_1,K_2,\ldots,K_P} \) to a single set of "equivalent" deterministic capacities \( EK_1,EK_2,\ldots,EK_P \). The first such heuristic that comes to mind is the Expected-Capacity Heuristic which consists of using \( EK_1, EK_2, \ldots, EK_P \), where \( EK_k \) is the expected capacity for time period \( T_k \) computed from \( P_{K_1,K_2,\ldots,K_P} \) in the following manner:

\[
EXP{K_k} = \sum_{j=1}^{C} K^j_k \cdot P_{K_1,k_2,\ldots,K_P}
\]

This Expected-Capacity Heuristic will be implemented in Chapter 5. It is important to realize that this heuristic is "blind" to the magnitude of airborne-delay costs; it determines ground delays based solely on ground costs. Since airborne-delay costs are typically higher than ground costs we can expect this heuristic to yield solutions that are
too "optimistic" in the sense that they do not impose enough ground delays to minimize the total expected cost. In Chapter 5, we will investigate another heuristic in this same family that is intended to compensate for this bias: the Weighted-Capacity Heuristic consists of using "equivalent" capacities \( WK_1, WK_2, \ldots, WK_P \) where:

\[
WK_k = \frac{C}{\sum_{j=1}^{C} K^j_k \cdot w_j} \quad \text{and} \quad \left( \sum_{j=1}^{C} w_j = 1 \right)
\]

The weights \( w_j \) allow hedging against unfavorable capacity cases by giving more weight (compared to the actual probabilities \( P_{K^1, K^2, \ldots, K^P} \)) to these cases. Any algorithm that can solve the deterministic version of the Ground Holding Problem can be used as an Equivalent Capacity Heuristic, once the "equivalent" capacities have been selected. For the case of general cost functions, we can use either the minimum cost flow formulation or the assignment formulation. The time complexity of the algorithm is then equal to the complexity of the deterministic algorithm used. For the case of "regular" cost functions the Fast Algorithm can be used and the time complexity of the Equivalent Policy Heuristic is \( O[N \ln(N)] \).

### 4.2.4 Equivalent-Policy Heuristics

The strategy for this family of heuristics is to compute ground holding policies for each capacity case \( K^1, K^2, \ldots, K^P \) separately (using any deterministic algorithm) and to

---

5 The extreme case is to set \( w_e = 1 \) for the lowest capacity case, \( e \), resulting in a very "pessimistic" strategy. This policy can be optimal if the airborne-delay costs are very high.
aggregate these policies into a single one. Again the most obvious such heuristic is the Expected-Policy Heuristic which consists of the following:

1. For each capacity case $K_{j1}, K_{j2}, ..., K_{jp}$, compute the deterministic policy $(X_{j1}, X_{j2}, ..., X_{jN})$, where $X_{j1}$ would be the optimal ground hold for flight $F_i$ if capacity case $K_{j1}, K_{j2}, ..., K_{jp}$ were to materialize.

2. Use the following "expected" ground hold for flight $F_i$:

$$EX_i = \sum_{j=1}^{C} X_{j1} \cdot P_{kj1, kj2, ..., kjp}$$

The same observation as in section 4.2.3 about the "optimistic" character of this policy can be made. We can, however, use a similar strategy (a "Weighted-Policy Heuristic") to try to compensate for this, namely use "adjusted" weights $w_j$ instead of the actual probabilities $P_{kj1, kj2, ..., kjp}$ in the computation of the final ground holds:

$$WX_i = \sum_{j=1}^{C} X_{j1} \cdot w_j$$

The time complexity of the algorithms in this family is $O(CN \ln(N))$ if regular cost functions are used.

Finally, note that the Weighted-Capacity Heuristic is equivalent to the Weighted-Policy Heuristic when we set one of the weights to 1 (and all others to 0).

### 4.2.5 Computation of Total Expected Costs

An important side effect of the Dynamic Program of section 4.1.1 is that, at the end of Step 4, the total expected cost of the "optimal" ground holding policy is available.
It corresponds to the quantity $G_N(\Omega)$. To see why this is true we have to go back to observation B) of section 4.1.1 which states that the "optimal" (minimum) value of the expected delay costs for the first $i$ flights $(F_1, F_2, ..., F_i)$, $G_i(H_i)$, depends only on $H_i$. Since flight $F_N$ is the flight with highest landing priority, we have $H_N = \Omega$ and the total expected delay cost corresponding to the "optimal" ground holding strategy is indeed $G_N(\Omega)$.

In the case of the Greedy Heuristic, the total expected delay cost, $TC(X_1^*, X_1^*, ..., X_N^*)$, corresponding to the "Greedy" ground holding policy $(X_1^*, X_1^*, ..., X_N^*)$, is given by $TC(X_1^*, X_1^*, ..., X_N^*) = \sum_{i=1}^{N} C_i(X_i^*, H_i^*)$. This total cost is also easily obtained as a direct "by-product" of the Greedy Heuristic since the individual costs $C_i(X_i^*, H_i^*)$ are computed at each step of the algorithm.

The situation is different for the Limited Lookahead, Equivalent-Capacity, and Equivalent-Policy Heuristics. For these heuristics the total expected delay cost has to be computed in a separate subroutine as we now explain.

All the heuristics presented in this Chapter work with general cost functions. Most of the numerical examples of Chapter 5 assume, however, that the airborne delay cost functions satisfy the "regularity" conditions of Section 2.2. More specifically, we will assume that $C_{ai}(x, y)$, the cost of holding flight $F_i$ $y$ time periods in the air, if it has been held $x$ time periods on the ground before take-off, is such that the marginal cost of delaying flight $F_i$ during time period $T_j$ ($j \geq P_i + x$) in the air, $\partial_j C_{ai}$, is given by:

$$\partial_j C_{ai} = C_{ai}(x, j-P_i+x+1) - C_{ai}(x, j-P_i+x) = K C_i(1+\alpha)^{x-1}(1+\beta)^j P_i x+1$$

When we assume that $\alpha = \beta$ (as is done in many numerical examples of Chapter 5), we get $\partial_j C_{ai} = K C_i(1+\beta)^j P_i$ in which case $K$ can be interpreted as a
multiplicative coefficient intended to reflect the ratio between the direct airborne operating costs of aircraft and the direct costs of keeping them on the ground. (In the numerical examples of Chapter 5 we will use a value for K between 2 and 3.)

The coefficient $\beta$ plays, for airborne delay costs, the same role as does $\alpha$ for ground delay costs. It will be referred to as the "airborne Cost Increase Coefficient" in the remainder of this thesis. We note that

$$\beta = \frac{\frac{\partial j}{\partial i} C_{ai} - \frac{\partial j}{\partial i} C_{ai-1}}{\frac{\partial j}{\partial i} C_{ai}},$$

which is the relative increase in cost due to holding a flight in the air for an additional time period.

To compute the total expected delay cost associated with a given ground holding policy $(X_1, X_2, \ldots, X_N)$, $TC(X_1, X_2, \ldots, X_N)$, we have to determine, for each capacity case that can materialize, the airborne delays that will be imposed on each flight. We will follow the notation of Section 4.1.1 and denote the airborne delay for flight $F_i$, when capacity case $\mathcal{K}_j = (K_1, K_2, \ldots, K_P)$ of the probabilistic capacity forecast $P_{K_1, K_2, \ldots, K_P}$ materializes, by $Y_i^j$. The expected cost for flight $F_i$, $EC_i$, is then given by

$$EC_i(X_1, X_2, \ldots, X_N) = C_{gi}(X_i) + \sum_{j=1}^{C} C_{ai}(X_i, Y_i^j) p_j$$

In order to compute the $Y_i^j$'s we will distinguish two cases in our numerical example:

Case 1: In some numerical examples we assume that a fixed landing priority rule is in effect. In this case the computation of the $Y_i^j$'s proceeds as follows:

We start with flight $F_N$ and set $HN = 0$. Then we compute $HN+1[0, X_N]$ using the procedure indicated in section 4.1.1. We go down the list of flights in descending order.
of priority to compute all the vectors $H_i$ ($i=1,N$) that correspond to the ground holding policy $(X_1,X_2,...,X_N)$.

For each flight $F_i$, we then find the index $k_j$ of the actual landing time period for that flight when the $j^{th}$ capacity case materializes through

$$k_j = \min \{ m \mid m \geq P_i + X_i \text{ and } H_i^m + \bar{L}_{m-1} < K_{m,j} \}. \quad Y_{i,j} \text{ is given by:}$$

$$Y_{i,j} = k_j - P_i - X_i$$

Finally the total expected cost $TC(X_1,X_2,...,X_N)$ is given by:

$$TC(X_1,X_2,...,X_N) = \sum_{i=1}^{N} \left[ C_{gi}(X_i) + \sum_{j=1}^{C} C_{ai}(X_i,Y_{i,j}) p_j \right]$$

**Case 2:** In other numerical examples we assume that we are allowed to determine the airborne delays in an "optimal" manner. The objective is then, given a ground holding policy $(X_1,X_2,...,X_N)$, to find, for each capacity case $Y_j = (K_{1,j},K_{2,j},...,K_{p,j})$ that materializes, the airborne delays $(Y_{1,j},Y_{2,j},...,Y_{N,j})$ that minimize the total expected airborne delay cost. But this corresponds precisely to the formulation of a deterministic Ground Holding Policy problem for which the deterministic capacities are $(K_{1,j},K_{2,j},...,K_{p,j})$, the "scheduled" arrival periods are $(P_1'=P_1+X_1,P_2'=P_2+X_2,...,P_N'=P_N+X_N)$, and the cost of delaying flight $F_i$ during time period $T_j$ is $\partial j C_{ai} = K C_i (1+\alpha)^{X_i-1}(1+\beta)^{j-P_i}X_i^{1+1}$. If we set $C_i'=KC_i(1+\alpha)^{X_i-1}(1+\beta)$ this cost can be rewritten as

$$\partial j C_{ai} = C_i'(1+\beta)^{j-P_i}. \quad \text{Thus, in this case it is clear we are dealing precisely with the type of deterministic problem discussed in Chapter 2 (for which we had}$$

$$\partial j C_{gi} = C_i'(1+\alpha)^{j-P_i}). \quad \text{The Fast Algorithm developed in Section 2.2 therefore yields}$$
the "optimal" airborne delays. But then, as was noted in that section, the optimal airborne delay policy corresponds simply to assigning flights to available capacity according to an "optimal" landing priority rule which does not depend on which capacity case materializes. This rule is obtained by ordering flights according to $p_{p+1} C_i$. Once this landing priority rule is found, we can go back to the situation in Case 1 and we can proceed in a similar fashion for the computation of total expected delay costs. It is precisely this method that is used in the numerical examples of Chapter 5.

The fact that this "optimal" landing priority rule does not depend on the capacity case that materializes is essential to the practicality of this rule: Air traffic controllers do not know in advance which capacity case will materialize for all time periods. Yet they can follow the rule "blindly" and be assured that the outcome is optimal in terms of total airborne delay costs. In practice, however, there is a limit to how long we can keep a given flight airborne; this constraint can be modeled by using a large value for $\beta$. In fact, as was observed in Chapter 2, a (very large) value of $\beta$ can be found for which the priority rule corresponds to the assignment of flights to available capacity on a first-come first-served basis. This observation will be used in the numerical examples of Chapter 5 to simulate present practices.

4.3 Dynamic Case

We stated in the introductory chapter that the basic approach taken in this thesis is to decompose the Strategic Flow Management Problem into problems of increasing complexity in order to build a solution method for the fundamental dynamic probabilistic (single arrival-airport) case. So far we have developed a fast algorithm for the deterministic case and several approaches (some based on this fast algorithm) for the
static-probabilistic case. This set of algorithms will be investigated further in Chapter 5 when we look at some numerical examples. It is, however, necessary at this stage to consider the dynamic probabilistic case if only to show that the work done so far does indeed move in the direction of developing algorithmic solutions for this case. We will in fact sketch out several "dynamic" algorithms that could be developed on the basis of "static" algorithms; we should however emphasize that a complete examination of the dynamic case is not the object of the present work (we will not investigate it computationally for example).

4.3.1 Stochastic Dynamic Programming

The Dynamic Program of section 4.1.1 expresses the ground holding policy for flights $F_1, \ldots, F_N$ as a sequence of ground delays $(X_1, \ldots, X_N)$. The following simple two-flight example shows that in order to take full advantage of the information carried by a probabilistic forecast system\(^6\) we also need to take into account the departure time of each flight $F_i$; in this case the optimal ground holding policy has to be expressed as a rule in which the ground hold $X_i$ depends on the history of capacity up to the time period $T_i$, where $t$ is the time of departure of flight $F_i$ (using the notation developed in Chapter 3).

Consider two flights $F_1$ and $F_2$ scheduled to land at airport Z during a single time period with capacity $K$. We assume that $K$ is probabilistic and can take on two values as shown below:

\[^6\] As defined in Chapter 3.
Flight $F_1$ is a long haul flight with a scheduled departure time $t_1$ and costs $Cg_1(1) = $1,000 and $Ca_1(1) = $2,000. Flight $F_2$ is a short haul flight with scheduled departure time $t_2 > t_1$ and costs $Cg_2(1) = $1,200 and $Ca_2(1) = $2,400. Furthermore we assume that the uncertainty in $K$ is resolved at time $t_2$ so that 70% of the time we know at time $t_2$ that $K=1$ and 30% of the time we know that $K=2$.

![Diagram](image)

**Figure 4.2**

If we consider only a "static" formulation of this single-time-period problem at time $t=t_1$, i.e., if we consider only the probabilistic forecast $P_K$ at $t=t_1$, it is not difficult to see that the "static" solution corresponds to keeping $F_1$ one time period on the ground and sending $F_2$ to airport $Z$ on time. The total expected cost of this policy is $1,000.

Now suppose we want to find the optimal dynamic solution; this corresponds to taking into account the dynamic nature of the problem and recognizing explicitly at time $t_1$
that the uncertainty in \( K \) will be resolved at time \( t_2 \) (note that this information is not available from a probabilistic forecast \( P_K \)). It is not difficult to see that in this case the optimal solution is to send \( F_1 \) on time and decide on what to do with flight \( F_2 \) only at time \( t_2 \) for an expected cost of \((0.7 \times 1,200 + 0.2 \times 0) = 840\). We have used the assumption that \( K \) is deterministic at time \( t_2 \) only to illustrate the point more vividly; the same argument could be made by using any properly selected conditional probabilistic forecast at \( t_2 \).

This example shows that a "static" capacity forecast \( P_{K_1,\ldots,K_P} \) (\( P_K \) for the above example) does not provide a model for the updating of information. A capacity forecast \( P_{K_1,\ldots,K_P} \) can be thought of as providing a model for the updating of information but only for departure times that are more that \( b \) (the "meaningful backward scope"—see Section 3.1) time periods after the beginning of time period \( T_1 \); furthermore it provides only partial information since it corresponds solely to the observation of capacities (which, in some cases, could be misleading since local weather conditions are not observed). We now develop a Stochastic Dynamic Program (SDP) inspired by the previous DP that takes into consideration the dynamic nature of the problem.

We assume that the probabilistic capacity forecasts are obtained from the "capacity based" forecast system described in Chapter 3. This means that the distribution of capacities after any time \( t \), \( P_{K_{t+1},\ldots,K_t} \), depends only on the observations \( K_{t-b},\ldots,K_t \) (where, following the notation developed in Chapter 3, \( f \) and \( b \) are the forward and backward scope and the index \( I_t \) is the index of the time period during which time \( t \) falls). In the following, we will assume that \( f \) is large enough to allow a probabilistic forecast \( P_{K_{t+1},\ldots,K_t} \) at any time \( t \) later than the earliest departure time of any flight among \( F_1,\ldots,F_N \). (We can always assume this to be true since, as was described in Chapter 3, the data used for building a forecast always contain the necessary information.) It is important to note that we assumed the "capacity based" forecast
system only to simplify the notation; the discussion applies to any forecast system if we replace the observation of capacities $K_{l-b},...,K_l$ by observations $\sigma_{l-b},...,\sigma_l$ that would include other parameters (such as local weather parameters). In fact one should not use the "capacity based" forecast system in practice since, as we pointed out in Section 3.1, it ignores potentially important weather data.

Note the difference with the situation in which we assume that only a pmf $P_{K_1,K_2,...,K_p}$ is available. In this case the solution is a static one since this formulation does not allow us to model the updating of information through time. A forecast system, on the other hand, extended to enough time periods before $T_1$ can be viewed as a mapping that associates, at any time $t$, a probabilistic forecast $P_{K_{l+1},...,K_1,K_2,...,K_p}$ to an observation of the state of the system. This situation is dynamic since it contains a model for the updating of information over time that covers the earliest scheduled times of departure.

To describe the SDP we assume that the flights $F_1,...,F_N$ are indexed in such a way that flight $F_i$ is scheduled to depart later than flight $F_{i+1}$. Furthermore we assume, for the moment, that a fixed landing priority rule is implemented such that $\Pi_i < \Pi_{i+1}$. This rule corresponds to giving landing priority to long-hauls.

---

7 As will be shown later the algorithm applies to any landing priority rule.
Figure 4.3

We want to develop a ground holding rule where the optimal ground hold $x_i^*$ for any flight $F_i$ is given as a function of observed values for the relevant capacities up to and including the capacity of the time period corresponding to its scheduled departure time, $T_{D_i}$, by $x_i^* = x_i^*(K_{D_i-b},...,K_{D_i})$. Note how the ground holding solution now takes into account the resolution of uncertainty with time: it assigns a ground hold based on the current observation of the weather situation. The following Stochastic Dynamic Program gives this optimal rule:

We denote by $OC_i(H_i,K_{D_i-b},...,K_{D_i})$ the optimal cost for the first $i$ flights given we make the observation $K_{D_i-b},...,K_{D_i}$ during and before time period $T_{D_i}$. Then $OC_i$ is obtained from $OC_{i-1}$ through the following recursion formula:
\[ OC_i(H_i, K_{D_i-1}, \ldots, K_{D_i}) = \]
\[ \min_{x_i} \{ C_i(x_i, H_i, K_{D_i-1}, \ldots, K_{D_i}) + \]
\[ \sum_{j: \text{all cases } i \rightarrow i-1} p_j \{ OC_{i-1}(H_{i-1}(H_i, x_i), K_{D_{i-1}-1}, \ldots, K_{D_{i-1}}) \} \]
The following Stochastic Dynamic Program is based on this recursion formula:

1) Start with \(i=1\) and for each \(H_1\) and \(K_{D_1:b},...,K_{D_1}\) compute \(x_1^*(H_1,K_{D_1:b},...,K_{D_1})\) that minimizes \(C_1(x_1,H_1,K_{D_1:b},...,K_{D_1})\).

2) Use the recursion formula to compute all the \(x_i^*(H_i,K_{D_i:b},...,K_{D_i})\)'s for all values of \((H_i,K_{D_i:b},...,K_{D_i})\).

Now we can compute the optimal ground holding strategy associated with any system trajectory. We start with the earliest departure (flight \(F_N\)) and compute the ground delay for \(F_N\) as \(x_N^*(0,K_{D_N:b},...,K_{D_N})\) where \(K_{D_N:b},...,K_{D_N}\) are the capacities actually observed at airport \(Z\) for the \(b\) time periods before \(F_N\)'s scheduled time of departure. Then at the time of departure of \(F_{N-1}\) we compute \(x_{N-1}^*(H_{N-1}(0,x_N^*),K_{D_{N-1}:b},...,K_{D_{N-1}})\) as a function of the observed capacities, and so on until flight \(F_1\). In this manner we have a ground holding policy that is adjusted to the evolution of the state of the system.

### 4.3.2 A Simple Numerical Example

Let us consider again the two-flight example of the previous section. The following figure shows how we extend the formulation to deal with the dynamic aspect: we extend the discretization of the time axis to the time period that includes the scheduled time of departure of flight \(F_2\) (note that the indexing of flights is reversed relative to the
previous section; flight F₁ has landing priority over flight F₂). For the sake of argument we assume that flight F₂ is scheduled to depart during (extended⁸) time period T₀.

![Diagram of flight schedules]

**Figure 4.4**

Applying the above stochastic dynamic program yields the optimal ground hold in the following manner:

---

⁸ In the terminology of Chapter 3.
We start with flight $F_2$ and compute $x_2^*(H_2,K_0)$ for all values of $H_2$ and $K_0$ where $H_2$ is the number of flights with higher priority than $F_2$ scheduled during $T_1$. We get:

- $x_2^*(0,1) = 0; \quad C_2(x_2^*,0,1) = 0$.

- $x_2^*(1,1) = 1; \quad C_2(x_2^*,1,1) = 1,200$.

- $x_2^*(0,2) = 0; \quad C_2(x_2^*,0,2) = 0$.

- $x_2^*(1,2) = 0; \quad C_2(x_2^*,1,2) = 0$.

Now we compute $x_1^*(0,K_{P_1})$ that minimizes:

$$C_1(x_1,0,K_{P_1}) + [p(K_0=1) \cdot C_2(x_2^*(H_2(0,x_1),1),H_2(0,x_1),1) +$$

$$p(K_0=2) \cdot C_2(x_2^*(H_2(0,x_1),2),H_2(0,x_1),2)]$$

It is easy to see that this optimal ground hold is zero for an overall total expected cost of: $p(K_0=1) \cdot C_2(x_2^*,1,1) + p(K_0=2) \cdot C_2(x_2^*,1,2) = 840$.

**4.3.3 Practical Considerations**

Both the time and space complexity of this SDP are greater than those of the original DP since we now have to compute and store values for all possible capacity cases $K_{D_i-b},\ldots,K_{D_i}$. The time complexity is $O[N(C(P+1)^2(M+1)^{P_{S_b}+1})]$ and the space complexity is $O[N(M+1)^{P_{S_b}+1}]$, where $S$ is the number of values that capacities for
individual time periods can take-on. However the same observations as in Section 4.1.2 for reducing these complexities are also valid, namely:

- The SDP consists of highly independent steps making parallel processing a desirable and promising approach.

- If the schedule and capacity configurations are stable a significant amount of preprocessing can be made corresponding to having the values $x_t^*(H_i, K_{D_{-b}}, \ldots, K_{D_i})$ stored in memory. What is left to do in real time is to retrieve the values that correspond to the particular trajectory of the system (the observed capacities of time periods) during the day in question.

- Finally, limited lookahead is also possible. Limited lookback can also be applied consisting of reducing the number of time periods we use to compute $x_t^*$. We could use for example the observation that corresponds only to the time period in which flight $F_i$ is departing in which case we use $x_t^*(H_i, K_{D_i})$. Note that this does not mean that we assume that $b=0$ but only that we do not use all the information carried by the forecast system.

4.3.4 General Observations and Directions for Further Research

Once a forecast system is available, a "dynamic" version of the Greedy algorithm of section 4.2.2 can be developed. Again, for the sake of presentation, we assume that this forecast system is the "capacity-based" forecast system; it is however easy to see that the algorithm can and should be used with a more complete forecast system. The first two steps of the algorithm are as follows:
- for each of the possible capacity values for $K_{DN-b},...,K_{DN}$ for the $(b+1)$ time periods before and including $T_{DN}$ (scheduled departure time for flight $F_N$) find $\gamma^*_N[K_{DN-b},...,K_{DN}]$, the "optimal" ground hold to be imposed on flight $F_N$ if the observed state of the system is $(K_{DN-b},...,K_{DN})$, that minimizes the expected cost for flight $F_N$ using the corresponding probabilistic capacity forecast $P_{K_{DN-b},...,K_{DN}}$. (This expected cost does not depend on the status of other flights since $F_N$ is the flight with highest landing priority.)

- for each of the possible capacity values for $K_{DN-b},...,K_{DN}$ and for $K_{DN-1-b},...,K_{DN-1}$ find $X_{N-1}^*[X_N^*[K_{DN-b},...,K_{DN},K_{DN-b},...,K_{DN},K_{DN-1-b},...,K_{DN-1}]]$ that minimizes the expected cost for flight $F_{N-1}$ using $P_{K_{DN-b},...,K_{DN},K_{DN-1-b},...,K_{DN-1}}$ given $X_N^*[K_{DN-b},...,K_{DN}]$. (Note that there will generally be much overlap between the indices in $K_{DN-b},...,K_{DN}$ and those in $K_{DN-1-b},...,K_{DN-1}$; in fact in most cases we will have $D_N = D_{N-1}$.) Again this can be done since the cost for flight $F_{N-1}$ depends only on the status of flight $F_N$.

This procedure is repeated until flight $F_1$. Notice that the number of capacity cases to be considered tends to increase exponentially as we go down the list of flights; there is however an upper bound on this number determined by the "meaningful backward scope" $b^9$. If the algorithm is used with a more complete forecast system, it

---

9 Furthermore one can apply a "limited lookback" version of this algorithm for which one considers only a limited (less than $b$) number of time periods before $T_D$ (i.e., for which we do not use all the information available) thereby reducing the complexity of the algorithm.
suffices then to replace, as was the case with the Stochastic Dynamic Program, the observations $K_{D_i-b},...,K_{D_i}$ by $\sigma_{D_i-b},...,\sigma_{D_i}$.

In the description of both the Stochastic Dynamic Program and the above "dynamic" Greedy algorithm we have assumed that the landing priority $\Pi$ was such that earlier flights had landing priority. This was done so we would be able to express the conditional probabilities $P_{K_{D_i-1-b},...,K_{D_i-1},K_{D_i-b},...,K_{D_i}}$ in their most natural form (since $D_i \leq D_{i-1}$); this, however, places no restriction on the applicability of the algorithms since Bayes’ Rule allows us to compute these conditional probabilities even when $D_i \geq D_{i-1}$.

The two flight example of the previous section indicates clearly the bias inherent in a dynamic approach in favor of long-haul flights; for most values of delay costs and capacity forecasts, flight $F_1$ will be allowed to take-off on time. In fact one could have proceeded by ignoring flight $F_2$ altogether, determining a ground hold for $F_1$ based solely on the (static) probabilistic forecast $P_{K_1}$ (resulting correctly in assigning no-ground hold in this particular case), then determining the ground hold time for flight $F_2$ based on the "residual" capacity forecast (obtained by subtracting one unit to $K_1$ in this particular case). This suggests the following heuristic approach to the dynamic problem:

- separate the flights $F_1,F_2,...,F_N$ into $Q$ groups of flights $G_1,G_2,...,G_Q$ based on their scheduled time of departure; group $G_1$ corresponding to the earliest departure times (say between times $s_1=0$ and $s_2$) and group $G_Q$ to the latest departure times (between times $s_Q$ and $t_{p+1}$, end time of the time span $T_1,T_2,...,T_p$; see Figure 4.5).
Figure 4.5

- use a static probabilistic algorithm to assign ground holds on flights of group $G_1$ using a probabilistic forecast $P_{K_1,K_2,...,K_p|\sigma_{I_1-b},...,\sigma_{I_1}}$. We use $X_1(\sigma_{I_1-b},...,\sigma_{I_1})$ to denote this assignment of ground holds.

- for all possible values of $\sigma_{I_2-b},...,\sigma_{I_2}$ and of $\sigma_{I_3-b},...,\sigma_{I_3}$ use a static probabilistic algorithm to assign ground holds on flights in group $G_2$ given the global assignment $X_1(\sigma_{I_1-b},...,\sigma_{I_1})$. This is accomplished by considering the probabilistic forecast $P_{K_1,K_2,...,K_p|\sigma_{I_1-b},...,\sigma_{I_1},\sigma_{I_1-b},...,\sigma_{I_1}}$ and reducing the values of the capacities $K_1,K_2,...,K_p$ according to $X_1(\sigma_{I_1-b},...,\sigma_{I_1})$. (Again note that there will generally be some overlap in the indices $I_{s1-b},...,I_{s1}, I_{s2-b},...,I_{s2}$.)

- repeat this procedure until group $G_Q$ is processed.

Since this approach to the dynamic problem is based on successive applications of static solutions we can use "static" heuristics, including those based on the Fast Algorithm, to build "dynamic" heuristics. The quality of the solutions obtained obviously depends on the choice of parameters such as $Q$ and the length of time periods. In fact one can "control" the amount of "dynamism" of the formulation by setting $Q$ to different values. When $Q = 1$, for example, we are dealing with the most "static" case
corresponding to a single application of a static probabilistic algorithm. At the other extreme when we use $Q = N$ we are dealing with the Stochastic Dynamic Program.

Finally we should note that both the Greedy algorithm and the Stochastic Dynamic Program above are not the most "dynamic" possible formulations of the problem. They in fact correspond to the case for which once we decide upon a ground hold for each aircraft we are committed to this value. A "fully dynamic" formulation would state the problem not as one of determining a ground delay $X_i$ for each flight $F_i$ but rather as one of determining for each time period $T_j$, after the scheduled departure time of flight $F_i$, whether to allow that flight to take-off or not, depending on the state of the system at that time as encapsulated by the sequence of observations $\mathcal{O}_{j-b,\ldots,\mathcal{O}_j}$. The solution to this "fully dynamic" problem is therefore expressed as a matrix of 0,1 numbers $[X_{ij}(\mathcal{O}_{j-b,\ldots,\mathcal{O}_j})]^{10}$.

\[10\] Note how difficult it would be to implement such a "fully dynamic" solution.
5.** NUMERICAL EXAMPLES**

5.1 **Introduction**

The purpose of this chapter is to illustrate, through numerical examples, some of the key points of the algorithmic approaches proposed in Chapter 4. For this purpose it is useful to be familiar with some of the computational solutions implemented in the examples to follow since they reflect important underlying assumptions, as will become evident when we discuss different methods for computing the expected air costs associated with a given ground holding policy in the case of probabilistic forecasts. We will therefore include, throughout the chapter, descriptions of the program modules used at a level of detail adequate to capture the essential features of the computations. This description can also serve as a guide for understanding the functional connections of the subroutines which are listed in Appendix E. The subroutines used were written in FORTRAN77 and implemented on a MACIIcx™.

Section 5.2 describes how the sample problems to which the different optimization algorithms are applied are generated. It shows how the parameters of the problem, such as the scheduled arrival time of each flight or the delay-cost functions for each flight, are set.

Section 5.3 deals with the deterministic case. The numerical example assumes that the capacity of airport Z can be forecasted exactly for the day in question. The fast algorithm of Section 2.2 is applied and the results are compared with the results from a simple first-come first-served approach (described in Section 1.2.2). The experiment assumes "regular" cost functions. The results show that significant savings in total delay
costs can be obtained and that the distribution of delays among aircraft classes can be controlled through the cost increase coefficient $\alpha$.

Section 5.4 deals with the static-probabilistic version of the Ground Holding Policy Problem:

Section 5.4.1 describes the fundamental algorithms used in the probabilistic experiments. In particular we show how the heuristic algorithms of Section 4.2 are implemented. These include the Greedy Heuristic (GR), the Limited Lookahead Heuristic (LL), the Expected-Capacity Heuristic (EK), and the Expected-Policy Heuristic (EX). We also provide a description of the subroutines used for computing the total expected costs related to a given ground holding policy.

Section 5.4.2 compares the performance of the above algorithms. Because of the time complexity of the Limited Lookahead Heuristic, we run an initial experiment that involves optimization only over a 2-hour time span. The experiment shows that the lowest expected costs are obtained with LL and GR; the total expected cost for EK is approximately 10% higher and that for EX approximately 30% higher than the total cost obtained for GR. The second experiment uses a version of EK (WEK) that allows weights for the capacity cases that are different from their probabilities, resulting in an aggregate capacity that is different from the expected capacity. The best results are obtained when only a few capacity cases (3 cases) are considered in the probabilistic capacity forecast and one of these cases is given a 1-weight (in this case the WEK Heuristic performs as well as GR). When four capacity cases are considered in the probabilistic forecast, the WEK Heuristic still performs well when we set the weight of one of the capacity cases to 1 (although not as well as GR, the relative difference in total costs being only 5%). Finally, Section 5.4.2 concludes with an interpretation of these
results that draws on properties of the single time period probabilistic GHPP (these properties are proved in Appendix B).

In Section 5.4.3 we use a 12-flight experiment with 3 time periods and we run the exact Dynamic Program and three heuristics (GR, EK, and EX) assuming general cost functions. The Dynamic Program solution (which we know is the optimal solution) is 20% better than the solutions from the heuristics. (When we assume regular cost functions on the same limited example then GR performs as well as the optimal DP and the costs for the two other heuristics are 20% higher).

Section 5.4.4 compares the GHP's obtained from using algorithmic approaches with GHP's that assign flights to capacity on a first-come first-served (FCFS) basis. We use the Greedy and EK heuristics to compare with a simulation of FCFS policies on a realistic 9-hour problem. We use the expected capacity as a basis for simulating FCFS policies and implement two algorithms: one outputs the first-come first-served policy (PFCFS), the other outputs the same policy but assigns no ground holds to flights with less than 20-minutes expected delay (TRESH).

- A first set of experiments shows that there is a significant saving in total expected costs (50%) for both GR and EK, even though PFCFS and TRESH use an optimal landing strategy. In fact even the no-ground-delay strategy is preferable since it allows more opportunity to use the optimal landing sequence.

- In a second set of experiments we implement a first-come first-served landing strategy for PCFCF and TRESH as well as for the no-ground-delay strategy. In this case the savings in cost are even larger (cost for GR = (cost for PCFCF)/3).
- In a third set of experiments we implement a version of GR that allows an air-cost increase coefficient (ALPHAL) that can be different from the ground cost coefficient (ALPHA). In this way we can also simulate first-come first-served for landings by using a very large coefficient ALPHAL. The results show that this modified GR still performs significantly better than simplistic approaches (30% cost savings).

Section 5.4.5 contains two experiments related to forecast systems. The first experiment shows how we can estimate the cost of error in the probabilistic forecast (as defined in Section 3.1); this information could be used, for example, to compare the cost of investing in better weather forecast systems with potential benefits. The second experiment examines the impact of the number of capacity cases distinguished in the probabilistic forecast of capacity. We use a capacity forecast with nine capacity cases and compute the ground holding policy using GR. Then these nine cases are clustered into only three cases and GR is again used to compute the approximate ground holding policy. The cost of this policy is computed using the nine original capacity cases and compared with the (necessarily lower) cost of the original policy. The increase in cost from clustering the capacity cases is never greater than 5%.

Section 5.4.6 shows how we can compute the value of additional landing slots. This information could be used, for example, in evaluating the benefits of better ATC technologies. The main result here is that the marginal benefit of an additional landing slot for a given day is sensitive to the time of day (with a much higher value for a landing slot during peak demand, as expected) but not very sensitive to the particular capacity forecast for the day.
In Section 5.4.7 we illustrate how the delay cost savings per scheduled flight afforded by the use of optimization methods increases with the level of congestion of the airport.

Finally, Section 5.4.8 shows that the quality of the ground holding solution obtained from the optimization algorithms is fairly insensitive to the length of the individual time periods used in the discretization of the time axis with obvious consequences in terms of the complexity of the algorithms that can be used.

5.2 Sample Problem Generation

5.2.1 Sample Schedule Generation

The computational examples presented in Sections 5.3 and 5.4 are based on a hypothetical situation constructed as follows:

We consider operations at a given airport $Z$ during a given day according to a schedule that resembles the operation profile at Boston Logan airport during a typical day, using 1987 data. Since 95% of the total daily operations occur between 7-am and 11-pm we restrict our analysis to this time span. The inputs to the problem are:

- A total number of landings for each hour of the day. A random number generator determines the scheduled landing time during each hour for each individual flights using a uniform distribution. (This would be equivalent to simulating the instants of Poisson arrivals, given the number of arrivals per hour.)
We have 3 types of aircraft distinguished on the basis of their ground delay costs. The random number generator assigns to each flight one of the three categories according to a prespecified flight mix. For each aircraft type i we are given the cost of holding an aircraft of this type one hour on the ground. We will denote this cost by CG(i,1). More generally the cost of holding an aircraft of type i J hours on the ground is denoted by CG(i,J). In most of the numerical examples we have assumed CG(1,1) = $400, CG(2,1) = $1,200 and CG(3,1) = $2,000 and the flight mix is 40% of aircraft type 1, 40% of aircraft type 2, and 20% of aircraft type 3.

We have actually set up a more general-purpose environment for generating sample schedules and problems for single airports:

The sample schedule and the basic parameters of the sample problem used to evaluate the optimization subroutines are set up by a sequence of three subroutines GEN, DISCRET, and SETCOSTS. These subroutines read the inputs for the sample problem from a set of input files (REQUES, BOSARR, TYPFIL and COSTS) the contents of which can be directly modified to generate different sample problems.

Subroutine GEN generates the sample schedule at airport Z for a given time span. It reads the input file REQUES which contains the start-hour (SHOUR) and the end-hour (EHour) of the time span as well as an initial seed for the random number generator (SEED). Subroutine GEN also reads another input file BOSARR which contains the total daily number of flights (TOTNUM) arriving at airport Z\(^1\) as well as an array (HPER(I); I=1,...,24) that represents the hourly distribution of the arrival demand throughout the

\(^1\)Boston in our case.
day (HPER(I) is the percentage of TOTNUM scheduled to arrive between hours I-1 and I). A uniform distribution is used to generate, for each hour between SHOUR and EHR, scheduled landing times. Since TOTNUM and HPER are given, this is equivalent to simulating the instants of Poisson arrivals. Seven types of aircraft can be distinguished on the basis of their delay costs. Subroutine GEN also reads input file TYPFIL that contains an array (TYPAR(i); i=1,...,7) representing the flight mix for airport Z (TYPAR(i) is the percentage of flights of type i operating at airport Z). The random number generator is also used to assign to each flight generated one of the types according to this flight mix.

The outputs of subroutine GEN are a total number of flights (TOTFL) scheduled to land between SHOUR and EHR and, for each flight i, a scheduled landing time BTIME(i) and an aircraft type BTYPE(i).

Subroutine GEN also reads (from the input file REQUES) the input PERL representing the length (in minutes) of the individual time periods\(^2\) corresponding to the partition of the total time period SHOUR-EHR.

Subroutine DISCRET utilizes the outputs of GEN to assign to each flight i its scheduled landing period LSPER(i).

Subroutine SETCOSTS reads the input file COSTS which contains, for each flight type i, the cost CG(i,1) of holding aircraft of that type one hour on the ground before take-off.

\(^2\) PERL can be 5,10,15,20,30 or 60 minutes in our particular implementation.
In summary, the sequence of subroutines GEN, DISCRETE and SETCOSTS sets up all the parameters of a sample problem associated with our formulation involving the discretization of the time axis. The remainder of this section shows how the costs of delaying aircraft for several time periods are computed.

5.2.2 Cost functions

We assume, for most of our numerical examples, that the ground holding cost functions are such that \( \Delta J(i) \), the cost of holding an aircraft of type \( i \) a \( J \)th hour on the ground, is given by:

\[
\Delta J(i) \equiv CG(i,J) - CG(i,J-1) = CG(i,1)(1+\alpha)^{J-1},
\]

where \( CG(i,J) \) is the cost of holding type \( i \) aircraft \( J \) consecutive hours on the ground.

For an interpretation of \( \alpha \) we note that:

\[
\alpha = (\Delta J(i)-\Delta J_{-1}(i))/\Delta J_{-1}(i),
\]

which is the relative increase in cost due to holding a flight on the ground for an additional hour. This cost increase coefficient \( \alpha \) (variable ALPHA in the Fortran programs) is an input of the sample problem and is also read from the input file COSTS. The numerical examples will show that the magnitude of \( \alpha \) has a considerable impact on the solution of the problem particularly on the distribution of ground holding times among aircraft types.

Subroutine SETCOSTS computes the costs \( CG(i,J) \) which, given our assumptions, are given by:
\[
\text{CG}(i,J) = \text{CG}(i,1)((1+\alpha)^{J-1})/\alpha \quad \text{if } \alpha \neq 0, \text{ and} \\
\text{CG}(i,J) = J \cdot \text{CG}(i,1) \quad \text{if } \alpha = 0
\]

The latter case corresponds to linear cost functions.

The hourly costs computed above cannot be used directly in our approach. The time period \text{SHOUR}-\text{EHOURL} has been subdivided into \text{PERN}=(\text{EHOURL}-
\text{SHOUR})/\text{PERL} individual time periods and we need costs that correspond to this finer subdivision. To compute these we also assume that there is a constant relative increase in the cost of holding any aircraft for an additional time period that we will note \(\theta\). Therefore we assume that \(C(i,m) - C(i,m-1) = C(i,1)((1+\theta)^{m-1})\), where \(C(i,m)\) represents the cost of holding an aircraft of type \(i\) for \(m\) time periods on the ground.

It is easy to show that \(C(i,1)\) and \(\theta\) can be computed from the inputs \(\text{CG}(i,1)\) and \(\alpha\) by:

\[
\theta = (1+\alpha)^{1/Q} - 1 \quad \text{where } Q \text{ is the number of time periods per hour (} Q = 60/\text{PERL}), \text{ and} \\
C(i,1) = \text{CG}(i,1) \cdot \frac{\theta}{\alpha}
\]

We can then compute \(C(i,K)\) the cost of holding aircraft type \(i\) \(K\) time periods on the ground by:

\[
C(i,K) = C(i,1) ((1+\theta)^{K-1})/\theta \quad \text{if } \alpha \neq 0 (\Leftrightarrow \theta \neq 0) \text{ and} \\
C(i,K) = K \cdot C(i,1) \quad \text{if } \alpha = 0 (\Leftrightarrow \theta = 0).
\]

Figure 5-1 shows the shape of a typical cost function.
It is possible to discuss the meaning of the coefficient $\alpha$ at this stage even before looking at the numerical examples. The case $\alpha = 0$ corresponds to linear costs implying that the cost of holding any flight for a single time period further on the ground does not depend on how long it has been held previously. In this case we expect that an efficient utilization of available capacity will always favor the same type of aircraft, namely the
higher cost type to the detriment of less costly types. At the other extreme the case where \( \alpha \) is very large will tend to favor flights on the basis of how long they have already been held on the ground. In fact it is easy to see that for \( \alpha \) above a certain threshold value it becomes optimal to assign available capacity to flights on a first-come first-served basis. This happens when \( \alpha \) is high enough so that for any pair of aircraft types \( i,j \) the following holds:

\[
C(i,2) > C(j,1)
\]

which requires that:

\[
\theta > \max_{i,j} \frac{C(j,1)}{C(i,1)} - 1
\]

These expectations are confirmed by the numerical examples to follow. Higher values of \( \alpha \) correspond to a widespread distribution of ground holds among aircraft types. We can therefore also interpret \( \alpha \) as a measure of the distribution of ground holds among aircraft types and indeed use \( \alpha \) explicitly as a parameter of the optimization to distribute these ground holds according to some prespecified criterion. For example we can make sure that no aircraft of type 1 is delayed more than \( K \) time periods if no aircraft of type 2 is delayed at all, by choosing \( \alpha \) such that \( \theta \) satisfies \( C(1,1) (1+\theta)^K > C(2,1) \).

Finally we note that these cost functions satisfy the regularity conditions of Section 2.2 and therefore that the fast algorithm described in that section does yield the optimal solution relative to a particular \( \alpha \). We will also see in section 5.3 that such cost functions allow a fixed landing priority rule that is optimal\(^3\).

\(^3\) In fact several such optimal rules can be found.
5.3 Deterministic Case

We utilize a sample problem generated as described above as a basis for evaluating the fast deterministic algorithm as well as illustrating the distributive effects mentioned above.

We assume that a fixed deterministic hourly forecast for the landing capacity at airport \( Z \) is available. The input file BOSKAP contains an array \( \text{KAP}(i) \ (i = 1, \ldots, 24) \) that represents this forecast for a whole day. The subroutine BREACKAP\(^4\) takes as input this array \( \text{KAP} \) and breaks up the hourly capacities into capacities corresponding to the individual time periods for each hour \( I \), between \( \text{SHOUR} \) and \( \text{EHOUR} \), according to the following strategy:

- If the hourly capacity \( \text{KAP}(I) \) and the length of the individual time periods \( \text{PERL} \) are such that \( \text{KAP}(I) \) is a multiple of \( \text{PERH} = 60 / \text{PERL} \) then \( \text{KAP}(I) \) is evenly distributed among the \( \text{PERH} \) time periods for hour \( I \).

- Otherwise we find \( K \) the greatest integer less than \( \text{KAP}(I) \) that is a multiple of \( \text{PERH} \), distribute evenly \( K \) among the individual time periods, then distribute the remaining \( \text{KAP}(I) - K \) capacity as evenly as possible. For example if \( \text{KAP}(I) = 52 \) and \( \text{PERL} = 5 \) the resulting distribution of capacity among the twelve time periods in the hour would be \( 5 + 4 + 4 + 5 + 4 + 5 + 4 + 5 + 4 + 5 = 52 \).

\(^4\) This subroutine is listed with the probabilistic subroutines in Appendix D.
Subroutine BPOL(DELAY,KAP,LSPER) implements the fast algorithm described in Section 2.2 and outputs the optimal GHP for the sample problem. It takes for inputs a particular schedule represented by the array LSPER, a forecasted capacity represented by the array KAP, and returns an array DELAY(i) (i =1,...,TOTFL). The time complexity of BPOL is, using the notation developed in Chapter 3, O(PNlnN).

Subroutine FCFS(DELAY,KAP,LSPER) determines ground holds on a first-come first-served basis.

Subroutine OKALG\(^5\) is based on the network formulation described in Section 2.1 and utilizes the Out of Kilter algorithm to determine an optimal ground hold strategy. Subroutine GENOKA generates the representation of the capacitated network corresponding to the GHPP that is fed to subroutine OKALG for optimization.

We know that, given the 'regular' cost functions described above and assumed for most of these numerical examples, the total ground holding costs for BPOL and OKALG are the same (both algorithms yield an optimal solution) although the actual ground holds may differ for some particular aircraft. Subroutine OKALG was used on a sample problem involving only 50 flights and took about 30 minutes to run. In contrast subroutine BPOL gave the optimal solution in less than one second. For this reason future experiments will not use OKALG; we still provide a listing of it in the appendix.

The following numerical example is intended to illustrate the impact of different values of \(\alpha\) on the costs as well as on the distribution of ground holds among aircraft

types when subroutine BPOL is used. For cost comparison, these results are contrasted with those obtained on the same sample problem from FCFS\textsuperscript{6}. Furthermore, even when \(\alpha \neq 0\) the costs we compare are computed assuming linear costs (\(\alpha = 0\)). These "equivalent" linear costs LC(i,K) are computed in the subroutine SETCOSTS.

Figure 5-2 shows the relationship of demand to capacity assumed in this numerical example. Three types of aircraft are distinguished on the basis of their ground costs. We assume CG(1,1) = $400, CG(2,1) = $1,200 and CG(3,1) = $2,000. The assumed flight mix is 40\% of aircraft type 1, 40\% of aircraft type 2, and 20\% of aircraft type 3 (TYPAR(1) = 40, TYPAR(2) = 40 and TYPAR(3) = 20).

\textbf{Figure 5-2}

\textsuperscript{6} The results from FCFS do not depend on the value of \(\alpha\) but only on the schedule.
Table 5-1 summarizes the results obtained for different values of $\alpha$. Along with the optimal ground holding policy, we also determine in each case the ground delays that would be imposed if a first-come first-served policy were used. The costs $C_{\text{opt}}$ and $C_{\text{fcfs}}$ represent respectively the cost of the optimal solution determined by the algorithm and the cost from the first-come first-served policy. In table 5-1 the numbers under the columns with heading N15 represent, for each aircraft type, the number of flights delayed less than 15 minutes, the numbers under N30, the number of flights delayed between 15 and 30 minutes, and so on until N2UP which corresponds to the number of flights delayed more than 2 hours. The numbers under heading FL/TYP represent the total number of flights for each type of flight and those under TOT-DEL represent the total flight-delay (measured in flight-hours) incurred by each flight type. It is clear from these results that if linear costs ($\alpha$=0) are used to determine the ground holding policies, the algorithm will generally give priority to type 2 and 3 aircraft. This observation raises questions of equitability since the same category of users (namely aircraft type 1 in this case) will always be penalized. The results also show that this state of affairs can be remedied, if a less than optimal solution is accepted, through a judicious choice of $\alpha$.

It is nonetheless clear that, even for high values of $\alpha$ (see, e.g. the results for $\alpha$=4) major total cost savings can be achieved through an optimization approach as compared to a first-come first-served approach. Moreover, through a judicious choice of $\alpha$, significant savings can be achieved without penalizing excessively any particular category of users.
\[ \alpha=0 \quad C_{\text{opt}}= 76,130 \quad ; \quad C_{\text{fcfs}}= 191,190 \]

<table>
<thead>
<tr>
<th>TYPE</th>
<th>FL/ TYPE</th>
<th>TOT- DEL</th>
<th>N15</th>
<th>N30</th>
<th>N45</th>
<th>N60</th>
<th>N90</th>
<th>N120</th>
<th>N2UP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>263</td>
<td>181</td>
<td>67</td>
<td>64</td>
<td>39</td>
<td>12</td>
<td>29</td>
<td>52</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>222</td>
<td>3</td>
<td>222</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>129</td>
<td>0</td>
<td>129</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

\[ \alpha=2 \quad C_{\text{opt}}= 88,397 \quad ; \quad C_{\text{fcfs}}= 191,190 \]

<table>
<thead>
<tr>
<th>TYPE</th>
<th>FL/ TYPE</th>
<th>TOT- DEL</th>
<th>N15</th>
<th>N30</th>
<th>N45</th>
<th>N60</th>
<th>N90</th>
<th>N120</th>
<th>N2UP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>263</td>
<td>166</td>
<td>67</td>
<td>64</td>
<td>39</td>
<td>12</td>
<td>62</td>
<td>19</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>222</td>
<td>17</td>
<td>187</td>
<td>26</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>129</td>
<td>0</td>
<td>129</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

\[ \alpha=4 \quad C_{\text{opt}}= 106,600 \quad ; \quad C_{\text{fcfs}}= 191,190 \]

<table>
<thead>
<tr>
<th>TYPE</th>
<th>FL/ TYPE</th>
<th>TOT- DEL</th>
<th>N15</th>
<th>N30</th>
<th>N45</th>
<th>N60</th>
<th>N90</th>
<th>N120</th>
<th>N2UP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>263</td>
<td>149</td>
<td>67</td>
<td>64</td>
<td>39</td>
<td>17</td>
<td>76</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>222</td>
<td>29</td>
<td>165</td>
<td>26</td>
<td>31</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>129</td>
<td>6</td>
<td>116</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

*Table 5.1*
5.4 Probabilistic Case

In this section we utilize the subroutines for sample schedule generation for evaluating the different probabilistic approaches described in Chapter 4. We now assume that we have a probabilistic forecast of the evolution of airport Z's hourly landing capacities. Furthermore we assume that we are given, in addition to ground holding costs, airborne costs for each flight. In most of our numerical examples we will assume that the cost of holding an aircraft of type i an m<sup>th</sup> time period in the air before landing is a multiple of \([CG(i,g+m) - CG(i,g+m-1)]\), where \(g\) is the number of time periods the aircraft has already been held on the ground before take-off. Also, we assume that the multiplicative coefficient is the same for all types of aircraft; the last item of the input file COSTS corresponds to this multiplicative air cost coefficient KAY. (Note that this corresponds to the airborne delay cost functions discussed in Section 4.2.5 where we take \(\beta = \alpha\).)

5.4.1 Description of Probabilistic Algorithms

We will assume that the probabilistic capacity forecast consists of a few capacity cases each associated with a probability of occurrence. The last items in input file REQUEST correspond to the number of the capacity cases (CASES) and their respective probabilities KPROB(I) (I=1,...,CASES). For the following numerical examples this probabilistic forecast will consist of three capacity cases (CASES=3). Input files KAP1, KAP2 and KAP3 contain these hourly capacity forecasts.
Expected-Capacity Heuristic:

Subroutine EKPOL implements the Expected-Capacity Heuristic described in section 4.2.3. It reads the contents of files KAP1, KAP2 and KAP3 and computes the expected capacity for each hour I according to the following formula:

$$\text{EKAP}(I) = \text{NINT} \left[ \sum_{j=1}^{\text{CASES}} \text{KPROB}(j) \times \text{PKAP}(j,I) \right]$$

where NINT is a function that returns the integer nearest to its argument and PKAP(j,I) is the element of the array PKAP that corresponds to the capacity of hour I for the j\textsuperscript{th} capacity case (this value is read from the file KAPj).

These expected capacities are then fed to subroutine BPOL which returns the Ground Holding Policy EKDELAY(i) (i = 1,..,TOTFL). This GHP therefore corresponds to the optimal ground delays if we assume that the actual capacities will be equal to EKAP.

Expected-Policy Heuristic:

Subroutine EXPOL implements the Expected-Policy Heuristic described in section 4.2.4. It calls subroutine BPOL for each capacity case and then averages the resulting GHP's to obtain EXDELAY(i) (i=1,..,TOTFL). If we note DELAY(i,j) the ground hold for flight i obtained for capacity case j then:

$$\text{EXDELAY}(I) = \text{NINT} \left[ \sum_{j=1}^{\text{CASES}} \text{DELAY}(i,j) \times \text{KPROB}(j) \right].$$

Note that the time complexity of EXPOL is CASES times that of EKPOL.
We also note that these two heuristics are insensitive to the difference in magnitude between airborne costs and in the ground costs as reflected in the coefficient KAY in the sense that the GHP obtained from these heuristics does not depend on KAY. The consequences of this will be investigated in our numerical examples.

**Greedy Heuristic:**

Subroutine GREEDY implements the Greedy algorithm of section 4.2.2. It returns GRDELAY(i) (i=1,...,TOTFL) the corresponding GHP.

The Greedy algorithm assumes that a fixed landing priority rule exists. Under these conditions, as was observed in Section 4.1.1, the total expected cost for a given flight depends only on the ground hold imposed on that flight and a vector Hi that represents the status of all flights with higher landing priority. The function AIRCOST(i,x,HIGHER) computes the total (ground + air) expected delay cost for a flight i, given the ground delay x and an array HIGHER(j) (j = 1,...,PERN) that represents, for each time period j, the number of flights with priority higher than flight i which are assigned to arrive\(^7\) at airport Z during that time period, by implementing the

\[ ...

\]

\(^7\) A flight scheduled to arrive during a given time period may not be able to land during the same time period.
procedure described in Section 4.2.5 (Case 1). The time complexity of AIRCOST is $O(C(P+1))$.

The subroutine GREEDY consists of a series of calls to function AIRCOST as shown in the following schematic description of GREEDY:

**step 1: initialize**

$$\text{HIGHER}(j) = 0 \text{ for all } j = 1,\ldots,\text{PERN}$$

$$\text{GRCOST} = 0$$

**Step 2 : For each flight i in descending order of landing priority do:**

get $$\text{GRDELAY}(i) = \arg\min_x \{ \text{AIRCOST}(i,x,\text{HIGHER}) \}$$

$$\text{GRCOST} = \text{GRCOST} + \text{AIRCOST}(i,\text{GRDELAY}(i),\text{HIGHER})$$

$$\text{HIGHER}(\text{LSPER}(i) + \text{GRDELAY}(i)) =$$

$$\text{HIGHER}(\text{LSPER}(i) + \text{GRDELAY}(i)) + 1$$

LSPER(i) is the element of an array LSPER that represents the scheduled landing period for flight i.

Subroutine GREEDY calls AIRCOST for each flight and searches possibly $P$ time periods. Its complexity is therefore $O(PNC(P+1))$.

\[8 \text{ C = CASES, P = (EHOUR-SHOUR)/PERL} \]
Computation of Total Expected Delay Costs:

Since GREEDY computes GRDELAY by minimizing the expected delay for each flight in descending order of priority (i.e. starting with the highest priority flight), the total expected cost of the policy, GRCOST, is available when GREEDY has finished computing GRDELAY. This is not the case for subroutines EKPOL and EXPOL; we will therefore need subroutines that can be used to compute the total expected delay costs associated with an arbitrary GHP.

Functions AGCOST(DELAY) and PICOST(DELAY,PRIO) are subroutines that compute total expected costs. They both assume that a fixed landing priority rule PRIO(i) (i = 1,...,TOTFL) exist, where PRIO(i) represents the landing priority of flight i. They however differ completely on how they actually compute expected costs; in function AGCOST the fixed priority rule remains implicit as can be seen from the list of inputs.

Function PICOST computes the total expected cost of a ground delay strategy by considering each capacity case separately and going through all flights in descending order of priority to compute the actual landing time period for each flight. The resulting air costs are then averaged over the different capacity cases weighted by the respective probabilities to obtain the total expected air cost for each flight. Function PICOST therefore implements the Case 1 procedure described in Section 4.2.5 in a way similar to the computation of expected delay costs in the GREEDY subroutine. We note that the

---

9 The flight with highest priority has PRIO = TOTFL
only assumption required for using function PICOST is that a fixed landing priority rule is in effect.

Function AGCOST, on the other hand, works only with "regular" airborne delay cost functions. As was demonstrated in Section 4.2.5-Case 2, under these conditions, finding the "optimal" airborne delays corresponds to solving an "equivalent" deterministic GHFP. AGCOST computes the expected delay costs based on these remarks. As we have seen in section 5.3, subroutine BPOL takes for inputs a schedule represented by the array LSPER(i) (i = 1,...,TOTFL) as well as a capacity forecast KAP(i) (i = 1,...,EHOUR-SHOUR). It outputs DELAY(i) (i= 1,...,TOTFL) which is the ground delay that minimizes the cost. In order to use BPOL in the context of air delays we input LSPER(i) + GDELAY(i) instead of LSPER(i), where GDELAY(i) is the ground hold for flight i, and we interpret the output DELAY as air delays corresponding to the materialization of the capacity case KAP. Function AGCOST implements this strategy by calling BPOL to compute the air delays and therefore the costs for each capacity case and then computes the total expected cost as the weighted average of these costs.

Limited Lookahead Heuristic:

Subroutine DPL3 implements a 3-time period lookahead dynamic programming heuristic. The general limited lookahead heuristics have been discussed in section 4.2.1 where we saw that the time complexity of a K-time period lookahead is, using the notation developed in that section , O[ PC(K+1)^2(M+1)^K].

As mentioned in that section, if we assume the availability of (M+1)^K parallel processors, this time complexity could be reduced to O[ PC(K+1)^2]. But for a single
processor sequential machine, which was used for our numerical examples, we are mainly constrained by the number of time periods we can allow for the sub-problems in the limited lookahead formulation. We are also constrained, through the magnitude of M, in the length of the individual time periods. (The time complexity increases if, for a fixed number of lookahead periods, we increase the length of these time periods.) Several computational trials with different values of K and P (and thereby M) have shown that we are constrained to limited lookaheads of three time periods of five minutes each for our particular hardware implementation.

The space complexity of the K-time period lookahead is $O(N(M+1)^K)$ which also severely limits the size of the limited lookahead for our particular implementation.

5.4.2 Comparison of Heuristics

This section contains two experiments. The first experiment is intended to compare the Limited Lookahead Heuristic with other heuristics (the Greedy Heuristic, the Expected-Capacity Heuristic, and the Expected-Policy Heuristic). The second experiment looks at the Equivalent-Capacity Heuristics. We conclude this section with an interpretation of the results.

Experiment with Limited Lookahead Heuristic:

In this experiment we look at the performance of some of the heuristics outlined above. We are limited, in choosing the total time span as well as the size of the individual
time periods we can consider, by both the time and space complexity of the Limited Lookahead Dynamic Program.

The space complexity $O[N(M+1)^K]$ of the Limited Lookahead limits severely the number of lookahead periods $K$. To see this let us consider a problem with PERL = 10 minutes, a maximum possible hourly capacity of 60 landings per hour and involving a total of several hundreds of flights (say 500) through a typical day. The memory space requirement is approximately $500.10^K$ bites of random access memory (RAM). For computers with only a few megabytes of memory available we cannot implement limited lookahead of more than three time periods. This is why we use $K=3$ in DPL3.

It is also clear how we are limited in terms of the time complexity$^{10}$ $O[P(N/P+1)^3]$ of DPL3 by the total number of flights and thereby by the total time span we can consider. The impact of the choice for the length of individual time periods, which determines $P$ ($P = (EHOUR-SHOUR)/PERL$), is more subtle. We have to consider a trade-off between the term $P$ which decreases with PERL, and the term $(N/P+1)^3$ which increases with PERL. It is this last term that dominates for large values of $P$. To illustrate this trade-off consider a sample problem of one hour involving 50 flights. The function $P(50/P + 1)$ is shown in Figure 5-3. We see that the minimum complexity is obtained for values of $P$ between 75 and 125 which corresponds to values of PERL of 4 to 2 minutes. Larger values of PERL, for example 20 minutes, corresponding to $P=12$, would lead to a much higher running time. We picked $PERL = 5$ minutes (the smallest reasonable length for a time period) in the following numerical examples.

---

$^{10}$ Obtained by substituting $K=3$ in the time complexity of the K-time period limited lookahead.
Subroutines EKPOL, EXPOL, GREEDY, and DPL3 are run on the following (small) sample problem:

- We consider operations at airport Z between 7-pm and 9-pm (SHOUR=19, E_HOUR=21) according to the schedule used for the deterministic case (Figure 5.2) that assumes respectively a demand of 45 flights and 32 flights during these hours. We use PERL=5 and ALPHA = 2.

- The flight mix is assumed to be 33% of aircraft type 1, 33% of aircraft type 2, and 34% of aircraft type 3. The first hour ground holding costs are assumed to be respectively $400, $1,200, and $2,000.
- The probabilistic forecast consists of three capacity cases (KAP1, KAP2, and KAP3). Figure 5-4 illustrates these capacity forecasts.

![Hourly capacity diagram](image)

**Figure 5-4**

Tables 5-2 and 5-3 show the results, in terms of total ground costs and total expected (ground + air) costs, from several runs of the algorithms on this sample problem. Throughout these examples ALPHA is fixed at 2. We vary only the probabilities of the different capacity cases (KPROB(1), KPROB(2), and KPROB(3)) and the air cost coefficient KAY. The rows labeled "NO DELAY" correspond to a policy for which no ground delays are imposed (all flights are assumed to take-off on time).
The total expected delay costs are computed using the subroutine AGCOST described in Section 5.4.1. The total cost associated with each row of the following Tables 5-2 and 5-3 corresponds therefore to an "optimal" landing strategy which minimizes expected airborne delay costs.

<table>
<thead>
<tr>
<th>KAY=2.5</th>
<th>KAY=2.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>KPROB(1)=0.3</td>
<td>KPROB(1)=0.4</td>
</tr>
<tr>
<td>KPROB(2)=0.5</td>
<td>KPROB(2)=0.4</td>
</tr>
<tr>
<td>KPROB(3)=0.2</td>
<td>KPROB(3)=0.2</td>
</tr>
<tr>
<td><strong>GROUND COSTS</strong></td>
<td><strong>TOTAL COSTS</strong></td>
</tr>
<tr>
<td>GREEDY</td>
<td>6,606</td>
</tr>
<tr>
<td>DPL3</td>
<td>6,606</td>
</tr>
<tr>
<td>EKPOL</td>
<td>6,045</td>
</tr>
<tr>
<td>EXPOL</td>
<td>6,849</td>
</tr>
<tr>
<td>NO DELAY</td>
<td>0</td>
</tr>
</tbody>
</table>

*Table 5-2*

<table>
<thead>
<tr>
<th>KAY=3</th>
<th>KAY=3.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>KPROB(1)=0.3</td>
<td>KPROB(1)=0.3</td>
</tr>
<tr>
<td>KPROB(2)=0.5</td>
<td>KPROB(2)=0.5</td>
</tr>
<tr>
<td>KPROB(3)=0.2</td>
<td>KPROB(3)=0.2</td>
</tr>
<tr>
<td><strong>GROUND COSTS</strong></td>
<td><strong>TOTAL COSTS</strong></td>
</tr>
<tr>
<td>GREEDY</td>
<td>6,606</td>
</tr>
<tr>
<td>DPL3</td>
<td>6,606</td>
</tr>
<tr>
<td>EKPOL</td>
<td>6,045</td>
</tr>
<tr>
<td>EXPOL</td>
<td>6,849</td>
</tr>
<tr>
<td>NO DELAY</td>
<td>0</td>
</tr>
</tbody>
</table>

*Table 5-3*
These numerical results show that the Limited Lookahead Heuristic does not perform better than the Greedy Heuristic. We can identify two reasons for this. One concerns the limitations imposed by the particular implementation on the number and the size of the time periods of lookahead (K = 3). The other is that the cost functions we used do not use the full power of the Dynamic Programming approach. This point is illustrated in section 5.4.3 when we will see that indeed Dynamic Programming performs better than any other approach when we do not assume "regular" cost functions.

Our results also show that both GREEDY and DPL3 perform consistently better than the other two heuristics EKPOL and EXPOL. GREEDY and DPL3 yield policies that have the same total cost (although they can be different policies as the ground costs in the right-hand side case of Table 5-2 indicates). As we mentioned above, the fact that these two heuristics perform almost identically is probably due to the fact that we use "regular" cost functions in these experiments; the fact that policies with different ground costs can lead to the same total cost will be explained at the end of this section at which point we will also be able to explain why some costs in Table 5-2 and in Table 5-3 are identical even though they correspond to experiments with different parameters KAY and KPROB. The certainty equivalence heuristic EKPOL comes close to GREEDY (or DPL3) on several examples when KAY is 2.5. The gap increases with KAY as Table 5-3 shows. This was to be expected since, as mentioned earlier, EKPOL is indifferent to the differential in air-borne costs versus ground costs, as measured by KAY.

The remaining experiments, when they involve "regular" cost functions, will not use DPL3 since its time complexity is forbidding for problems of the size we want to consider (several hours duration) and since the Greedy Heuristic seems to perform just as well in these conditions. We should however keep in mind that the Dynamic
Programming approach is likely to outperform the other heuristics for problems with more complex cost functions, if parallel processors and virtual memory techniques are available to allow limited lookahead of more meaningful scope. The experiment with general cost functions (Section 5.4.3) is intended to support this point, albeit on a very restricted example of the probabilistic formulation.

The next experiment investigates the Equivalent-Capacity Heuristics introduced in Section 4.2.3.

**Experiment with Equivalent-Capacity Heuristics:**

Subroutine WEKPOL is similar to subroutine EKPOL except that the ground holds are computed by using an equivalent deterministic capacity case that corresponds to a weighted average of the probabilistic capacity cases, where the weights can be different from the probabilities of the capacity cases and can be adjusted at will.

**EXPERIMENT WITH THREE PROBABILITY CASES:**

The following Figure 5-5 shows the three capacity cases used:
The following tables show the results for a sample problem in which:

- KPROB(1) = 0.3; KPROB(2) = 0.4; and KPROB(3) = 0.3.

- The air cost coefficient is first set to KAY = 2, and then increased to KAY = 3.

We assume linear cost functions (ALPHA = 0).
Subroutine WEKPOL is run with different weights. The quantities WPROB(1), WPROB(2), and WPROB(3) correspond to these weights. The total costs are computed using subroutine AGCOST.

<table>
<thead>
<tr>
<th>KAY=2</th>
<th>KAY=2</th>
</tr>
</thead>
<tbody>
<tr>
<td>KPROB(1)=0.30, WPROB(1)=0.40</td>
<td>KPROB(1)=0.30, WPROB(1)=0.00</td>
</tr>
<tr>
<td>KPROB(2)=0.40, WPROB(2)=0.50</td>
<td>KPROB(2)=0.40, WPROB(2)=1.00</td>
</tr>
<tr>
<td>KPROB(3)=0.30, WPROB(3)=0.10</td>
<td>KPROB(3)=0.30, WPROB(3)=0.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>GROUND COSTS</th>
<th>TOTAL COSTS</th>
<th>GROUND COSTS</th>
<th>TOTAL COSTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>GREEDY</td>
<td>62,100</td>
<td>130,740</td>
<td>62,100</td>
<td>130,740</td>
</tr>
<tr>
<td>EKPOL</td>
<td>76,900</td>
<td>143,620</td>
<td>87,300</td>
<td>173,350</td>
</tr>
<tr>
<td>WEKPOL</td>
<td>96,900</td>
<td>146,260</td>
<td>62,100</td>
<td>130,740</td>
</tr>
<tr>
<td>NO DELAY</td>
<td>0</td>
<td>176,700</td>
<td>0</td>
<td>251,740</td>
</tr>
</tbody>
</table>

Table 5-4
<table>
<thead>
<tr>
<th>KAY=3</th>
<th>KAY=3</th>
</tr>
</thead>
<tbody>
<tr>
<td>KPROB(1)=0.30, WP:OB(1)=0.40</td>
<td>KPROB(1)=0.30, WP:OB(1)=0.50</td>
</tr>
<tr>
<td>KPROB(2)=0.40, WP:OB(2)=0.40</td>
<td>KPROB(2)=0.40 WP:OB(2)=0.40</td>
</tr>
<tr>
<td>KPROB(3)=0.30, WP:OB(3)=0.20</td>
<td>KPROB(3)=0.30, WP:OB(3)=0.10</td>
</tr>
<tr>
<td><strong>GROUND</strong></td>
<td><strong>TOTAL</strong></td>
</tr>
<tr>
<td><strong>COSTS</strong></td>
<td><strong>COSTS</strong></td>
</tr>
<tr>
<td>GREEDY</td>
<td>62,100</td>
</tr>
<tr>
<td>EKPOL</td>
<td>76,900</td>
</tr>
<tr>
<td>WEKPOL</td>
<td>93,100</td>
</tr>
<tr>
<td>NO DELAY</td>
<td>0</td>
</tr>
<tr>
<td><strong>GROUND</strong></td>
<td><strong>TOTAL</strong></td>
</tr>
<tr>
<td><strong>COSTS</strong></td>
<td><strong>COSTS</strong></td>
</tr>
<tr>
<td>GREEDY</td>
<td>62,100</td>
</tr>
<tr>
<td>EKPOL</td>
<td>76,900</td>
</tr>
<tr>
<td>WEKPOL</td>
<td>66,200</td>
</tr>
<tr>
<td>NO DELAY</td>
<td>0</td>
</tr>
</tbody>
</table>

**Table 5-5**

We see from Table 5-4 (left side) that, when KAY = 2, WEKPOL does not perform as well as the Greedy Heuristic if we simply increase the weight of the unfavorable capacity cases. In fact EKPOL performs better than WEKPOL in this case. However we see that when we assume KAP2 (by making its weight = 1) WEKPOL performs identically to GREEDY. The same observation is true when KAY is increased.
to 3 (Table 5-5). This interesting phenomenon will be explained at the end of this section when we interpret the results in light of some properties of the single time period case.

Table 5-6 corresponds to the case where KAP1 and KAP2 are equiprobable. We note that both the certainty equivalence corresponding to KAP1 (WPROB(1) = 1) and that corresponding to KAP2 (WPROB(2) = 1) yield the same total expected cost. They differ naturally on the total cost of the ground holds they impose on the flights. The fact that different GHP's can lead to the same total cost may seem surprising at first; we will however explain this at the end of this section.

<table>
<thead>
<tr>
<th>KAY=2.5</th>
<th>KAY=2.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>KPROB(1)=0.40, WPROB(1)=0.00</td>
<td>KPROB(1)=0.40, WPROB(1)=1.00</td>
</tr>
<tr>
<td>KPROB(2)=0.40, WPROB(2)=1.00</td>
<td>KPROB(2)=0.40, WPROB(2)=0.00</td>
</tr>
<tr>
<td>KPROB(3)=0.20, WPROB(3)=0.00</td>
<td>KPROB(3)=0.20, WPROB(3)=0.00</td>
</tr>
<tr>
<td>GROUND COSTS</td>
<td>TOTAL COSTS</td>
</tr>
<tr>
<td>GREEDY</td>
<td>62,100</td>
</tr>
<tr>
<td>EKPOL</td>
<td>93,100</td>
</tr>
<tr>
<td>WEKPOL</td>
<td>62,100</td>
</tr>
<tr>
<td>NO DELAY</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>KAY=2.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>KPROB(1)=0.40, WPROB(1)=0.50</td>
</tr>
<tr>
<td>KPROB(2)=0.40, WPROB(2)=0.50</td>
</tr>
<tr>
<td>KPROB(3)=0.20, WPROB(3)=0.00</td>
</tr>
<tr>
<td>GROUND COSTS</td>
</tr>
<tr>
<td>GREEDY</td>
</tr>
<tr>
<td>EKPOL</td>
</tr>
<tr>
<td>WEKPOL</td>
</tr>
<tr>
<td>NO DELAY</td>
</tr>
</tbody>
</table>

**Table 5-6**
Table 5-7 corresponds to the case where all capacity cases are assumed to be equiprobable. If KAY is 3, using WPROB(2) = 1 yields the same total costs as GREEDY. When KAY is increased to 4, we have to use WPROB(1) = 1. This phenomenon was also observed in the results of Tables 5-4 and 5-5.
<table>
<thead>
<tr>
<th>KAY=3</th>
<th>KAY=3</th>
</tr>
</thead>
<tbody>
<tr>
<td>KPROB(1)=0.33, WPROB(1)=0.00</td>
<td>KPROB(1)=0.33, WPROB(1)=0.50</td>
</tr>
<tr>
<td>KPROB(2)=0.33, WPROB(2)=1.00</td>
<td>KPROB(2)=0.33, WPROB(2)=0.50</td>
</tr>
<tr>
<td>KPROB(3)=0.33, WPROB(3)=0.00</td>
<td>KPROB(3)=0.33, WPROB(3)=0.00</td>
</tr>
<tr>
<td><strong>GROUND COSTS</strong></td>
<td><strong>TOTAL COSTS</strong></td>
</tr>
<tr>
<td><strong>GROUND COSTS</strong></td>
<td><strong>TOTAL COSTS</strong></td>
</tr>
<tr>
<td>GREEDY</td>
<td>62,100</td>
</tr>
<tr>
<td>EKPOL</td>
<td>76,900</td>
</tr>
<tr>
<td>WEKPOL</td>
<td>62,100</td>
</tr>
<tr>
<td>NO DELAY</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>62,100</td>
</tr>
<tr>
<td></td>
<td>76,900</td>
</tr>
<tr>
<td></td>
<td>111,900</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>KAY=4</th>
<th>KAY=4</th>
</tr>
</thead>
<tbody>
<tr>
<td>KPROB(1)=0.33, WPROB(1)=0.00</td>
<td>KPROB(1)=0.33, WPROB(1)=1.00</td>
</tr>
<tr>
<td>KPROB(2)=0.33, WPROB(2)=1.00</td>
<td>KPROB(2)=0.33, WPROB(2)=0.00</td>
</tr>
<tr>
<td>KPROB(3)=0.33, WPROB(3)=0.00</td>
<td>KPROB(3)=0.33, WPROB(3)=0.00</td>
</tr>
<tr>
<td><strong>GROUND COSTS</strong></td>
<td><strong>TOTAL COSTS</strong></td>
</tr>
<tr>
<td><strong>GROUND COSTS</strong></td>
<td><strong>TOTAL COSTS</strong></td>
</tr>
<tr>
<td>GREEDY</td>
<td>176,500</td>
</tr>
<tr>
<td>EKPOL</td>
<td>76,900</td>
</tr>
<tr>
<td>WEKPOL</td>
<td>62,100</td>
</tr>
<tr>
<td>NO DELAY</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>176,500</td>
</tr>
<tr>
<td></td>
<td>76,900</td>
</tr>
<tr>
<td></td>
<td>176,500</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

**Table 5-7**

We conclude that, if the number of capacity cases is low, the weighted certainty equivalence (WEKPOL) can be made to perform better than the simple certainty equivalence (EKPOL) through a successful choice of the weights of the various capacity cases. The weights that correspond to the low cost solutions are usually those that assign a weight of 1 to the most likely capacity case. We saw however that, in the case of equiprobable capacity cases (Table 5-7), it is no longer clear what the "optimum" equivalent capacity case is since it depends on other parameters such as KAY. We also expect that this ambiguity will increase when the number of capacity cases increases as
we now demonstrate. After the next experiment we will appeal to some results for the single time period case to help us interpret some of the results presented in this section.

EXPERIMENT WITH FOUR CAPACITY CASES:

In Tables 5-8 and 5-9 we added a fourth capacity case (KAP4) to the above situation, such that KAP4 represents a capacity of 35 flights per hour for the entire time span considered (3-pm to midnight).

We start with four equiprobable cases:
<table>
<thead>
<tr>
<th></th>
<th>GROUND COSTS</th>
<th>TOTAL COSTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>GREEDY</td>
<td>102,300</td>
<td>148,680</td>
</tr>
<tr>
<td>EKPOL</td>
<td>87,300</td>
<td>153,550</td>
</tr>
<tr>
<td>WEKPOL</td>
<td>176,500</td>
<td>176,500</td>
</tr>
<tr>
<td>NO DELAY</td>
<td>0</td>
<td>231,940</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>GROUND COSTS</th>
<th>TOTAL COSTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>GREEDY</td>
<td>102,300</td>
<td>148,680</td>
</tr>
<tr>
<td>EKPOL</td>
<td>87,300</td>
<td>153,550</td>
</tr>
<tr>
<td>WEKPOL</td>
<td>62,100</td>
<td>155,600</td>
</tr>
<tr>
<td>NO DELAY</td>
<td>0</td>
<td>231,940</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>GROUND COSTS</th>
<th>TOTAL COSTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>GREEDY</td>
<td>102,300</td>
<td>148,680</td>
</tr>
<tr>
<td>EKPOL</td>
<td>87,300</td>
<td>153,550</td>
</tr>
<tr>
<td>WEKPOL</td>
<td>35,200</td>
<td>179,140</td>
</tr>
<tr>
<td>NO DELAY</td>
<td>0</td>
<td>231,940</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>GROUND COSTS</th>
<th>TOTAL COSTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>GREEDY</td>
<td>102,300</td>
<td>148,680</td>
</tr>
<tr>
<td>EKPOL</td>
<td>87,300</td>
<td>153,550</td>
</tr>
<tr>
<td>WEKPOL</td>
<td>97,300</td>
<td>149,920</td>
</tr>
<tr>
<td>NO DELAY</td>
<td>0</td>
<td>231,940</td>
</tr>
</tbody>
</table>

**Table 5.8**

The costs of WEKPOL and also those of EKPOL are close to the costs of greedy, but no "equivalent" capacity case exists as was the case for the experiment involving three capacity cases (even for equiprobable cases). This is true even if the four capacity cases are not equiprobable, as we show next:
We conclude that as the number of capacity cases we use to model the problem increases, even if they are not equiprobable, no single case "dominates" as was true for three capacity cases. However we also notice that EKPOL yields solutions with costs close to those from GREEDY.

We now turn to an interpretation of these results.
Interpretation of Results:

In order to interpret the above results we need to look at the following properties of the single time period version of the static-probabilistic GHPP.

Consider a single time period version of the GHPP for which $N$ flights, $F_1, ..., F_N$, are scheduled to land at airport $Z$ during a single time period $T$, with capacity $K$ (a random variable). The GHPP consists of determining, given a probabilistic forecast $P_K(k)$ for the capacity $K$, for each flight $F_i$, whether $F_i$ should be sent on time to airport $Z$ (in which case it may experience an airborne delay of at most one time period) or kept one time period on the ground (in which case it will be able to land without airborne delay during the time period following time period $T$).
We note by $C_{g_i}$ the cost of delaying flight $F_i$ one time period on the ground and by $C_{a_i}$ the cost of delaying it one time period in the air. We assume that the flights $F_1, ..., F_N$ have been indexed such that $C_{a_{i+1}} \geq C_{a_i}$.

It is obvious that, in these conditions, there exists an "optimal" landing priority rule which consists of giving landing priority to flights with higher indices (i.e., $\Pi_{i+1} > \Pi_i$ for all $i=1, ..., N$). We will assume that this landing priority rule is in effect in the remainder of this discussion of the single time period GHPP.
We will denote by $C$ the number of capacity cases, $K_1, K_2, ..., K_C$, that the probabilistic forecast $P_K(k)$ distinguishes; also, we denote by $p_j$ the probability $P(K_j) = \frac{C}{\sum_{j=1}^{C} p_j}$. We will assume that these capacities have been indexed such that $K_1 < K_2 < ... < K_C$ and that $N > K_1$ (otherwise the solution to the GHPP is trivial: it consists of not delaying any flight). We also assume that there is more than one capacity case ($C > 1$), otherwise the solution to the GHPP is equally trivial (it consists of sending exactly $K_1$ flights on time).

A Ground Holding Policy (GHP) $P$ will be denoted by $P = \begin{cases} F_{i_1}, F_{i_2}, ..., F_{i_R}, & \\
F_{j_1}, F_{j_2}, ..., F_{j_{N-R}} & \end{cases}$

where $F_{i_1}, F_{i_2}, ..., F_{i_R}$ are the flights that are sent on time to airport $Z$ according to policy $P$, and $F_{j_1}, F_{j_2}, ..., F_{j_{N-R}}$ are the flights that are kept on the ground according to policy $P$. The total expected cost of a policy $P$ will be denoted $TC(P)$. A GHP is said to be optimal if there exists no other GHP with strictly lower cost (there could be several optimal GHP's).

The first two claims to follow assume that $N \geq K_C$. Claim 1 is an interesting property of any GHP in this case. Claim 2 shows that, when $N \geq K_C$ (i.e. under congested conditions), there always exists an optimal GHP which sends exactly $K_j$ flights on time to airport $Z$, for some $j=1,...,C$. This is referred to as the "dominating capacity" property. Claim 3 shows that the "dominating capacity" property holds even when $N < K_C$, except, obviously, when the optimal GHP does not delay any flight and $N \neq K_j$ for all $j=1,...,C$.

In this section we simply state the three claims. The proofs of these claims can be found in Appendix B.
**Claim 1:**

If we assume that $C_{ai} = k \cdot C_{gi}$ for all $i=1,...,N$, where $k$ is a coefficient that does not depend on the index $i$, and if we assume that $N \geq K_C$, then:

There exists a GHP $Q$ that sends exactly $K_j$ aircraft on time to airport $Z$, for some $j = 1,...,C$, such that $TC(Q) \leq TC(P)$, for any GHP $P$.

An immediate consequence of Claim 1 is that there is an optimal GHP that sends exactly $K_j$ aircraft on time, for some $j=1,...,C$. The following Claim 2 indicates how to find such an optimal GHP:

**Claim 2:** "Dominating Capacity" Property of Optimal Solution - Congested Conditions:

If we assume that $C_{ai} = k \cdot C_{gi}$ for all $i=1,...,N$, where $k$ is a coefficient that does not depend on the index $i$, and if we assume that $N \geq K_C$, then:

(1) if there exists no index $u$ ($1 \leq u \leq C$) such that $k \cdot (p_1 + p_2 + ... + p_u) = 1$, then:

(i) if $k \cdot p_1 > 1$, the policy $P$ that sends the $K_1$ "most costly" flights defined by:

$$P = \begin{cases} F_{N-K_1+1},...,F_N \\ F_1,...,F_{N-K_1} \end{cases}$$

is optimal.
(ii) else, there must exist an index, s, such that: \( k(p_1 + p_2 + \ldots + p_s) < 1 \)

and \( k(p_1 + p_2 + \ldots + p_{s+1}) > 1 \) (since we know that \( k(p_1 + p_2 + \ldots + p_C) = k > 1 \)),

and the policy P defined by:

\[
P = \begin{cases} 
F_{N-Ks+1}+1, \ldots, F_N \\
F_1, \ldots, F_{N-Ks+1}
\end{cases}
\]

is optimal.

(2) if there exists an index s such that \( k(p_1 + p_2 + \ldots + p_s) = 1 \), then both policies

\[
P = \begin{cases} 
F_{N-Ks+1}+1, \ldots, F_N \\
F_1, \ldots, F_{N-Ks}
\end{cases}
\quad \text{and} \quad 
Q = \begin{cases} 
F_{N-Ks+1}+1, \ldots, F_N \\
F_1, \ldots, F_{N-Ks+1}
\end{cases}
\]

are optimal. (Note that the index s cannot be C since \( k(p_1 + p_2 + \ldots + p_C) = k > 1 \); therefore it makes sense to talk about \( K_{s+1} \).) Furthermore any policy S "in-between" is also optimal, where a policy S "in-between" P and Q is given by:

\[
S(m) = \begin{cases} 
F_{N-Ks-m+1}+1, \ldots, F_N \\
F_1, \ldots, F_{N-Ks-m}
\end{cases} \quad \text{for some m, } 0 \leq m \leq K_{s+1} - K_s.
\]

Claim 3: "Dominating Capacity" Property of Optimal Solution - General

Conditions:

If we assume that \( C_i = k - C_i \) for all \( i = 1, \ldots, N \), where \( k \) is a coefficient that does not depend on the index \( i \), then:

(1) if there exists no index \( u \) (\( 1 \leq u \leq C \)) such that \( k(p_1 + p_2 + \ldots + p_u) = 1 \), then:

(i) if \( kp_1 > 1 \), the policy
\[ P = \begin{cases} F_{N-K'1+1}, \ldots, F_N \\ F_1, \ldots, F_{N-K'1} \end{cases} \]

that sends the \( K'1 \) "most costly" flights on time, where \( K'1 \) is defined by: \( K'1 = \min\{K1, N\} \), is optimal.

(ii) else, there must exist an index, \( s \), such that: \( k(p_1 + p_2 + \ldots + p_s) < 1 \)
and \( k(p_1 + p_2 + \ldots + p_{s+1}) > 1 \) (since we know that \( k(p_1 + p_2 + \ldots + p_C) = k > 1 \)),
and the policy \( P \) defined by:

\[ P = \begin{cases} F_{N-K's+1}, \ldots, F_N \\ F_1, \ldots, F_{N-K's+1} \end{cases} \]

where \( K's+1 = \min\{K_{s+1}, N\} \), is optimal.

(2) if there exists an index \( s \) such that \( k(p_1 + p_2 + \ldots + p_s) = 1 \), then both policies

\[ P = \begin{cases} F_{N-K's+1}, \ldots, F_N \\ F_1, \ldots, F_{N-K's} \end{cases} \quad \text{and} \quad Q = \begin{cases} F_{N-K's+1}, \ldots, F_N \\ F_1, \ldots, F_{N-K's+1} \end{cases} \]

where \( K's = \min\{K_s, N\} \) and where \( K's+1 = \min\{K_{s+1}, N\} \), are optimal.

Furthermore any policy \( S \) "in-between" is also optimal, where a policy \( S \) "in-between" \( P \) and \( Q \) is given by:

\[ S(m) = \begin{cases} F_{N-K's-m+1}, \ldots, F_N \\ F_1, \ldots, F_{N-K's-m} \end{cases} \quad \text{for some} \ m, 0 \leq m \leq K's+1-K's. \] (Note that if \( N \leq Ks \) \( \Rightarrow m=0 \) there is only one policy (policy \( P = \) policy \( Q = \) policy \( S(0) \)) "in-between".)

These claims show that for the single time period case, when \( N \geq K_C \), the solution to the probabilistic GHPP *always* corresponds to the solution of a deterministic GHPP where the capacity is set to one of the capacities of the probabilistic forecast. We refer to this particular capacity as the "dominating" capacity case.
Tables 5-8 and 5-9 show that this is not true for the multiple time period case (either table is a counter-example). All the experiments, however, show that a "Dominating Capacity" phenomenon still occurs in the sense that we can find some "dominating" capacity case that either performs as well as the Greedy Heuristic or comes very close to the performance of the Greedy Heuristic.

Furthermore this "dominating" capacity case can be found in all the experiments that assume only three capacity cases if we use the criteria of Claim 3. Let us examine this point using the notation of Claim 3:

- Table 5-5: we have $k \cdot p_1 = 3 \times .3 = .9 < 1$ and $k \cdot (p_1+p_2) = 3 \times .4 = 1.2 > 1$. This suggests using capacity case KAP2 as the "dominating" case. It is indeed this case that performs as well as the Greedy Heuristic.

- Table 5-7 (bottom): we have $k \cdot p_1 = 4 \times .33 = 1.32 > 1$. This suggests KAP1 as the "dominating" case. It is precisely KAP1 that performs as well as the Greedy Heuristic in that experiment.

- Table 5-2 (right-hand side): we have $k \cdot p_1 = 2.5 \times .4 = 1$. This suggests that, according to Case 2 of Claim 3, many policies "in-between" KAP1 and KAP2 can yield the same total cost. This is precisely what we observe in this experiment where two policies with different ground costs ($6,720$ and $6,970$) have the same total expected cost $12,013$. Intuitively, the reason for that is that, since $k \cdot p_1 = 1$, the expected air delay cost for many flights (those that are delayed if the capacity turns out to be $K_1$) equals the cost of delaying these flights on the ground. This also explains why we find the same total cost of $12,013$ in Table 5-3 even though different values for KAY and KPROM are used.
the cost 12,013 is a ground cost in Table 5-3. This "equivalent in-between" policies phenomenon is even more pronounced in Table 5-6:

- Table 5-6: here we also have $k \cdot p_1 = 2.5 \times .4 = 1$. We see that GHP's that differ greatly by their ground delay costs (62,100; 111,900; and 176,500) yield the same total expected delay cost 176,500.

In experiments with four probability cases, however, we cannot use the criteria motivated by the single time period case any more. Both in Table 5-8 and Table 5-9 the "dominating" capacity case (the capacity case that comes the closest to the performance of the Greedy Heuristic) is KAP4. In both cases, though, applying the criterion would have led to using KAP2. (In Table 5-8 for example $k \cdot p_1 = 2.5 \times .25 = .725 < 1$ and $k \cdot (p_1 + p_2) = 2.5 \times .5 = 1.25 > 1$).

5.4.3 General Cost Functions

This set of experiments is designed to show that the exact Dynamic Programming approach does yield better results than the Greedy approach on problems that do not use "regular" cost functions. In order to do this we need to consider more general cost functions. We are still restricted, however, in the size of the sample problem we can construct by the particular hardware available for these computations. We will therefore construct a limited size problem as follows:

- We consider operations at airport Z between 2-am and 5-am (SHOUR=2, EHour=5), and we set PERL = 60 to be able to use an exact 3-time period Dynamic Program. In fact we use a subroutine identical to DPL3 to implement this exact Dynamic Program but we stop at one lookahead. The arrival demand during these three hours is
assumed to be five flights, four flights, and four flights respectively. Figure 5-7 shows the three capacity cases assumed for these examples.

![Graph showing hourly capacity from 2am to 5am with three capacity levels KAP1, KAP2, KAP3.]

Figure 5-7
The first experiment uses 'regular' cost functions with ALPHA set to 2 and runs all four algorithms, GREEDY, DPL3, EKPOL, and EXPOL. Table 5-10 shows the results for different probabilities of capacity cases and different values for KAY. We note that for the case of a relatively favorable capacity forecast all algorithms perform identically (and therefore all yield an optimal solution). The reason for this is that, because the function NINT used by EKPOL (see Section 5.4.1) rounds-off the results to the nearest integer, the expected capacity used by EKPOL is precisely KAP2. But KAP2 is also the "dominating" capacity in this case (we have \( k \cdot p_1 = 2.5 \times .3 = .75 < 1 \) and \( k \cdot (p_1 + p_2) = 2.5 \times .8 = 2 > 1 \)). Therefore, in this case, EKPOL is equivalent to WEKPOL with WPROB(2)=1 which we know to yield good results with three capacity cases. When we deal with a less favorable forecast and a higher KAY, however, only GREEDY performs as well as DPL3. The reason for this is that now the "dominating" capacity case is no longer KAP2 but has shifted to KAP1 (since \( k \cdot p_1 = 3 \times .4 = 1.2 > 1 \)); this is confirmed by the fact that GREEDY yields a GHP that results in no airborne delays.

<table>
<thead>
<tr>
<th>KAY=2.5</th>
<th>KAY=3</th>
</tr>
</thead>
<tbody>
<tr>
<td>KPROB(1)=0.3</td>
<td>KPROB(1)=0.4</td>
</tr>
<tr>
<td>KPROB(2)=0.5</td>
<td>KPROB(2)=0.5</td>
</tr>
<tr>
<td>KPROB(3)=0.2</td>
<td>KPROB(3)=0.1</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td><strong>GROUND COSTS</strong></td>
<td><strong>TOTAL COSTS</strong></td>
</tr>
<tr>
<td>GREEDY</td>
<td>4,800</td>
</tr>
<tr>
<td>DPL3</td>
<td>4,800</td>
</tr>
<tr>
<td>EKPOL</td>
<td>4,800</td>
</tr>
<tr>
<td>EXPOL</td>
<td>4,800</td>
</tr>
<tr>
<td><strong>NO DELAY</strong></td>
<td>0</td>
</tr>
</tbody>
</table>

*Table 5-10*
The second experiment differs from the first in that subroutine SETCOSTS is no longer used to compute the costs CG(i,m). Instead, these costs are directly read (for i = 1 to 3 and m = 0 to 3) from a new input file HOURCOSTS allowing us to use arbitrary cost functions. We assume linear costs for type 1 and type 3 aircraft, with hourly ground costs of respectively $800 and $4,000 for any hour (linear cost functions). We assume however that type 2 aircraft have non-linear costs such that the three costs are $1,600, $8,000, and $8,000 per hour delay, respectively. The intended interpretation for type 2 aircraft costs is that they can be used to model flights that should not be more than one hour late. A fixed landing priority rule is assumed such that type 2 aircraft have the highest priority followed by type 3 and finally type 1 aircraft. Landing Priorities are assigned within each class of aircraft according to the schedule, flights with earlier scheduled arrivals having higher priorities. Table 5 shows the results that confirm our earlier statements about the superiority of the Dynamic Programming approach over the other algorithms when general cost functions are involved. Again the reason that EKPOL yields the same GHP as GREEDY is that the "dominating" capacity case is KAP2 which corresponds also to the expected capacity (after rounding-off the results).
\[
\begin{array}{l|c|c}
\text{GROUND COSTS} & \text{TOTAL COSTS} \\
\hline
\text{GREEDY} & 4,800 & 22,080 \\
\text{DPL3} & 14,400 & 18,720 \\
\text{EKPOL} & 4,800 & 22,080 \\
\text{EXPOL} & 4,800 & 22,080 \\
\text{NO DELAY} & 0 & 25,440 \\
\end{array}
\]

Table 5-11

5.4.4 Comparison with Deterministic First - Come, First - Served Policies

This set of experiments is designed around a situation of more realistic size. We consider operations at airport Z between 3-pm and midnight (SHOUR=15, EHOUR=24) with a flight mix similar to that of Boston-Logan airport (40% of aircraft type 1, 40% of aircraft type 2, and 20% of aircraft type 3). The ground costs are CG(1,1) = $400, CG(2,1) = $1,200, and CG(3,1) = $2,000.

Subroutine PFCFS is used to develop policies that use first-come first-served (FCFS) sequencing and essentially disregards probabilities. It is almost identical to the deterministic subroutine FCFS(DELAY,KAP,LSPER) (see Section 5.3) except that KAP is replaced by EKAP computed for subroutine EKPOL. PFCFS therefore consists of simulating a first-come first-served policy using, as a capacity forecast, the expected values of the hourly probabilistic forecasts.
As was described in Section 1.2.2, a threshold value is imposed on ground holds in practice. Subroutine TRESH is a variant of subroutine PFCFS that implements such a threshold value of 20 minutes for ground holds (i.e., the GHP computed by TRESH is precisely the GHP computed by PFCFS except that ground holds of less than 20 minutes are set to zero).

Figure 5-8 shows the three cases of the probabilistic capacity forecast assumed for all the numerical examples to follow. The demand profile is the same as that assumed for the numerical examples of the deterministic case and is shown in dashed lines in the same figure.
Table 5-12 shows the results of three runs of the algorithms when we vary the probabilities of the capacity cases shown above as well as the air cost coefficient KAY. The costs are taken to be linear (ALPHA = 0). The NO DELAY row corresponds to a GHP that does not assign any ground delays.
<table>
<thead>
<tr>
<th></th>
<th>KAY=2</th>
<th></th>
<th>KAY=4</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>KPROB(1)=0.4</td>
<td>KPROB(2)=0.4</td>
<td>KPROB(1)=0.4</td>
<td>KPROB(2)=0.4</td>
</tr>
<tr>
<td></td>
<td>KPROB(3)=0.2</td>
<td></td>
<td>KPROB(3)=0.2</td>
<td></td>
</tr>
<tr>
<td>GROUND</td>
<td>TOTAL</td>
<td>GROUND</td>
<td>TOTAL</td>
<td></td>
</tr>
<tr>
<td>COSTS</td>
<td>COSTS</td>
<td>COSTS</td>
<td>COSTS</td>
<td></td>
</tr>
<tr>
<td>GREEDY</td>
<td>61,900</td>
<td>152,300</td>
<td>174,900</td>
<td>174,900</td>
</tr>
<tr>
<td>PFCFS</td>
<td>243,300</td>
<td>309,220</td>
<td>243,300</td>
<td>375,140</td>
</tr>
<tr>
<td>TRESH</td>
<td>226,700</td>
<td>300,660</td>
<td>226,700</td>
<td>374,620</td>
</tr>
<tr>
<td>EKPOL</td>
<td>92,900</td>
<td>160,900</td>
<td>92,900</td>
<td>228,900</td>
</tr>
<tr>
<td>NO DELAY</td>
<td>0</td>
<td>203,520</td>
<td>0</td>
<td>407,040</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>KAY=6</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>KPROB(1)=0.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>KPROB(2)=0.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>KPROB(3)=0.2</td>
<td></td>
</tr>
<tr>
<td>GROUND</td>
<td>TOTAL</td>
<td></td>
</tr>
<tr>
<td>COSTS</td>
<td>COSTS</td>
<td></td>
</tr>
<tr>
<td>GREEDY</td>
<td>174,900</td>
<td>174,900</td>
</tr>
<tr>
<td>PFCFS</td>
<td>243,300</td>
<td>441,060</td>
</tr>
<tr>
<td>TRESH</td>
<td>226,700</td>
<td>448,580</td>
</tr>
<tr>
<td>EKPOL</td>
<td>92,900</td>
<td>296,900</td>
</tr>
<tr>
<td>NO DELAY</td>
<td>0</td>
<td>610,560</td>
</tr>
</tbody>
</table>

**Table 5-12**

It is clear from the above results that the relative advantage of TRESH (compared to PFCFS) disappears for large values of KAY. The reason is that the threshold policy does not address the case where air delays are undesirable. If we took KAY=1, for example, it is clear that TRESH will always outperform PFCFS since the flights that took-off on time, and would have been delayed according to PFCFS, are ready to take advantage of available capacity. But in this case it is also clear that the no-delay policy is
optimal since, by taking $KAY = 1$, we are in fact indifferent between ground and air delays.

Another interesting observation from the results of the above table is that the NO DELAY policy performs better than either TRESH or PFCFS. The reason for this is that TRESH and PFCFS hold the most costly flights (type 2 and type 3 aircraft) on the ground. Therefore what is gained in terms of future air costs is partly lost in terms of ground costs as is evident from Table 5-12. This is also due to the fact that we compute the expected air costs using AGCOST for all the policies. The implicit assumption in doing this is that, even if the relative costs of different flights are not considered when deciding on the ground delays, the air traffic controllers at airport Z sequence flights optimally for landing. Therefore the no-delay policy gives more opportunities of "doing the right thing" at the destination airport compared with policies from TRESH or PFCFS.

This last observation is also the reason why TRESH performs surprisingly well compared to PFCFS even though we are dealing with a rather pessimistic forecast which should make a smaller amount of ground holds less advantageous. It would be wrong therefore to attribute the better performance of TRESH to the supposition that it somehow accounts for the uncertainty in forecast when it is due simply to the fact that, by allowing more flights to show up at airport Z, it gives more opportunity to save costs in sequencing these flights optimally for landing. The performance of TRESH compared to either GREEDY or EKPOL should convince us that this is not a good overall policy. We also see that if we increase the value of $KAY$ above 4, PFCFS catches up with and outperforms TRESH.

Our assumptions so far about "optimal" sequencing at landing do not reflect present practices of air traffic controllers. Flights are generally sequenced for landing on
a first-come first-served basis in actual circumstances. The next set of experiments is intended to reflect more accurately a first-come first-served policy through the use of a subroutine FCOST which is almost identical to PICOST except that the "optimal" fixed landing priority (which depends on ALPHA ) is replaced by the landing priority rule that orders flights according to their time of arrival at airport Z. The following Table 5-13 contains the results of this experiment. The heading TOTAL OPT. COSTS corresponds to the total expected (ground + air) costs computed assuming an "optimal" landing priority rule, the heading TOTAL FCFS COSTS corresponds to the same costs computed using the first-come first-served landing policy described above.
### Table 5-13

<table>
<thead>
<tr>
<th>GROUND COSTS</th>
<th>TOTAL OPT. COSTS</th>
<th>TOTAL FCFS COSTS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GREEDY</strong></td>
<td>61,900</td>
<td>152,300</td>
</tr>
<tr>
<td><strong>PFCFS</strong></td>
<td>243,300</td>
<td>309,220</td>
</tr>
<tr>
<td><strong>TRESH</strong></td>
<td>226,700</td>
<td>300,660</td>
</tr>
<tr>
<td><strong>NO-DELAY</strong></td>
<td>0</td>
<td>203,520</td>
</tr>
</tbody>
</table>

---

### Table 5-13

<table>
<thead>
<tr>
<th>GROUND COSTS</th>
<th>TOTAL OPT. COSTS</th>
<th>TOTAL FCFS COSTS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GREEDY</strong></td>
<td>35,200</td>
<td>71,808</td>
</tr>
<tr>
<td><strong>PFCFS</strong></td>
<td>156,000</td>
<td>184,670</td>
</tr>
<tr>
<td><strong>TRESH</strong></td>
<td>137,700</td>
<td>171,490</td>
</tr>
<tr>
<td><strong>NO-DELAY</strong></td>
<td>0</td>
<td>75,328</td>
</tr>
</tbody>
</table>
\[
\begin{align*}
\text{ALPHA} &= 2 \\
\text{KAY} &= 2 \\
\text{KPROB}(1) &= 0.4 \\
\text{KPROB}(2) &= 0.4 \\
\text{KPROB}(3) &= 0.2 \\
\end{align*}
\]

<table>
<thead>
<tr>
<th>GROUND COSTS</th>
<th>TOTAL OPT. COSTS</th>
<th>TOTAL FCFS COSTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>GREEDY</td>
<td>96,788</td>
<td>616,200</td>
</tr>
<tr>
<td>PFCFS</td>
<td>276,310</td>
<td>671,450</td>
</tr>
<tr>
<td>TRESH</td>
<td>265,820</td>
<td>668,710</td>
</tr>
<tr>
<td>NO-DELAY</td>
<td>0</td>
<td>668,040</td>
</tr>
</tbody>
</table>

**Table 5-14**

The results in Table 5-13 show that, as expected, TRESH does not perform better than PFCFS with respect to total FCFS (First-Come First-Served) costs. We also note that the NO-DELAY policy now leads to larger expected costs. Table 5-13 also shows that we have to consider extremely favorable capacity forecasts and use a very small KAY to see TRESH perform slightly better than PFCFS; but then the NO-DELAY policy becomes preferable.

Table 5-14 shows that for ALPHAL = 2 we still obtain significant (\(\sim 30\%\)) savings in total expected costs from using GREEDY and an "optimal" landing strategy. In fact we can implement landing strategies according to a cost increase coefficient different from the one used to determine the ground holds. Let us denote this cost coefficient as ALPHAL (ALPHA Landing). The reason we might want to consider this is that for the case of unfavorable capacity forecasts, when air delays are large, a small cost increase coefficient ALPHAL leads to a situation where some flights are held in the air for large amounts of time. The coefficient ALPHAL could be used, as is done for ground delays, to control the distribution of air delays; in this case large values of ALPHAL would yield
a more equitable distribution of delays. It is easy to see that the strategy that minimizes the maximum air delay (over all flights) is the first come first served landing policy which is equivalent, as was observed for the case of ground holds, to some large enough value of ALPHAL. We can expect that for a value of ALPHAL = 0 some of the airborne holds imposed on some of the flights are unrealistic if we consider fuel and other constraints. But this can also be expected for a first come first served policy in unfavorable weather conditions. In fact it is not unusual that when such a situation occurs in practice, air traffic controllers are forced to divert some flights to neighboring air fields. As was done with ALPHA, the value of ALPHAL can be adjusted to redistribute airborne delays among classes of aircraft.

The following experiment is intended to illustrate this point by looking at the distribution of air delays when different values of ALPHAL are used. Subroutine LANPI is used to compute the landing priority (LANPRIO) that corresponds to ALPHAL. The function PICOST is called with the arguments PICOST(GRDELAY,LANPRIO) to compute total expected cost associated with ALPHAL. Note that, even though the airborne delays are set based on ALPHAL (through LANPRIO), the function PICOST use ALPHA to compute ground costs as well as airborne delay costs; this allows us to compare costs. We use linear costs (ALPHA = 0) in this numerical example and we should expect the costs of using ALPHAL ≠ 0 to be higher than the cost of the base case (ALPHAL = 0) since the base case corresponds to the optimal landing strategy. Table 5-15 shows the results for different values of ALPHAL. The base case corresponds to the first case of Table 5-13 (KAY = 2, KPROB(1) = KPROB(2) = 0.4, KPROB(3) = 0.2). The different landing strategies are relative to the ground holding strategy from GREEDY. The TOTAL COSTS correspond for GREEDY to the optimal landing strategy.
costs, and for PFCFS and NO-DELAY to the first-come first-served-landing strategy costs.

\[
\text{ALPHA} = \begin{array}{cccc}
\text{GROUNDDS} & \text{COSTS} & \text{TOTAL} & \text{COSTS} \\
\text{GREEDY} & 61,900 & 152,300 & 269,100 & 293,420 & 326,220 \\
\text{PFCFS} & 243,300 & 415,940 & \\
\text{NO-DELAY} & 0 & 529,520 & \\
\end{array}
\]

Table 5-15

The value of \text{ALPHA} = 12 corresponds to first come first served for landing (increasing \text{ALPHA} beyond that value does not change the landing sequence). We note that even for that value of \text{ALPHA}, (which does not differentiate among aircraft) the total savings in costs compared to PFCFS are still sizeable. The following Figures 5-12 show the resulting distributions of airborne delays among aircraft types. We note that for \text{ALPHA} = 12 the distribution for aircraft type 1 does not match those for the other types as might be expected at first. The reason for this is that the ground delays have been computed using the subroutine GREEDY and a linear cost which tends to delay aircraft of type 1 exclusively. The result of this is that type 1 aircraft show up during uncongested periods whereas type 2 and 3 aircraft have been allowed to take off on time and show up during periods that, for some capacity cases, can be congested. This explains the relative distributions of air delays shown in the figures.
Figure 5.9
5.4.5 Forecast Systems

Cost of errors in forecasts:

In chapter 3 (Section 3.1) we discussed the use of the probabilistic formulation for evaluating the impact of better weather information on the overall cost of delays as well as a means for evaluating the benefits of improved forecasting technologies for a particular situation. We refer back to this discussion and set up a simplified situation in the following manner:

- We assume that the perfect state of information is embodied in only two observations O₁ and O₂.

- Associated with these two observations are two (probabilistic) forecasts F₁ and F₂ with respective likelihoods p₁ and p₂ (=1 - p₁).

We want to compare two forecast systems S₁ and S₂ with respective accuracies a₁ and a₂ (aᵢ is the probability that system Sᵢ will correctly predict Fᵢ when confronted with observation Oᵢ). In order to do that we compute the cost of using S₁ and that of using S₂. The cost of using a forecast system was defined to be the cost of implementing that system compared to the cost of the (hypothetical) perfect information forecast with accuracy a₁ =1. We recall that a perfect system can only forecast a probabilistic capacity. As was fully discussed in section 3.1, this is due to the inherent chaotic nature of local weather patterns and the (absolute) limitations of computational devices, which may never permit perfectly accurate deterministic forecasts.

Since we are computing costs relative to the perfect information system, the cost of using a forecast system reduces to the cost of predicting the wrong Fᵢ. One of the
components of this cost would be, for example, the total expected cost of implementing the solution associated with forecast F₁ (SOL₁) when forecast F₂ realizes, minus the total expected cost of implementing the solution associated with F₂; we will note this cost of using SOL₁ given F₂ by C₁₂. Similarly C₂₁ represents the cost of using solution SOL₂, associated with forecast F₂, when F₁ realizes. The following probability tree summarizes the situation for forecast system Sᵢ (i=1,2).

![Probability Tree Diagram]

**Figure 5.10**

The cost of system Sᵢ is therefore \((1 - aᵢ)(p₁C₂₁ + p₂C₁₂)\).
Assume a situation where forecast system $S_1$ is used. A new, more accurate, forecast system $S_2$ ($a_2 > a_1$) is available and we need to evaluate the benefits of switching to $S_2$ (in order to compare these benefits to the additional cost of $S_2$ for example). These benefits are computed as $(a_2 - a_1)(p_1C_{21} + p_2C_{12})$. The problem therefore reduces to computing the $C_{ij}$s.

To illustrate the use of the GREEDY algorithm for computing the $C_{ij}$s, consider the following situation with forecasts $F_1$ and $F_2$:

- $F_1$ corresponds to the situation of Figure 5-5.

- $F_2$ is such that KAP3 is 40 flights for all hours, KAP2 is 35 flights for all hours, and KAP1 is 30 flights for all hours.

We compute $C_{12}$ in the following manner:

- subroutine GREEDY is applied to $F_2$ to compute the cost of using the right forecast. This cost is: $152,300.

- subroutine GREEDY is used with $F_1$ to compute the ground holding strategy (DELAY1) if $F_1$ is selected by mistake (i.e., if $F_2$ is the right forecast and our forecast system says $F_1$).

- function AGCOST (or PICOST) is used on DELAY1 with capacity forecast $F_2$ to compute the cost of using the wrong forecast. This cost is: $163,340.

- $C_{12}$ is the difference $163,340 - 152,300 = 11,040$. 

161
Capacity forecast clustering:

We now turn our attention to another topic related to capacity forecast systems: we investigate how the quality of the solution is affected by the number of capacity cases we use in the probabilistic forecast of capacity.

It was mentioned in chapter 3 that one practical approach to capacity forecasting is clustering of capacity cases. In this method several capacity cases $k_1, k_2, \ldots, k_m$ are aggregated into fewer cases $K_1, K_2, \ldots, K_q$ (with $q < m$) such that some measure of error is minimized\(^{11}\). The major advantage of clustering capacity forecasts is to diminish the computational complexity of the optimization algorithms. We should however realize that some level of clustering is necessary anyway for our problem, simply because we are dealing with a discrete formulation and there is a limit to the level of resolution we can reasonably consider. In this numerical example we investigate the impact of such clustering by considering the following situation:

- We assume that $m = 9$. The corresponding capacity cases $KAP1, \ldots, KAP9$ ($k_i = KAPi$) are such that $KAP1$, $KAP2$, and $KAP3$ correspond to the cases of Figure 5-8. The remaining capacity cases are constructed from these three base cases as follows: $KAP4$ and $KAP5$ are obtained from $KAP1$ by subtracting two flights per hour for all hours across the time span considered (3pm to midnight) for $KAP4$, and adding two flights for $KAP5$. $KAP6$ and $KAP7$ are based on $KAP2$ in the same way ($KAP6 = KAP2 - 2$; $KAP7 = KAP2 + 2$). $KAP8$ and $KAP9$ are based on $KAP3$ ($KAP8 = KAP3 - 2$; $KAP9 = KAP3 + 2$).

\(^{11}\) Using the least squares method for example.
- We assume \(q = 3\) and we take \(K_1 = \text{KAP1}, K_2 = \text{KAP2},\) and \(K_3 = \text{KAP3}\) (a natural selection considering the way we built the capacity cases around these base cases).

We consider two examples for evaluation. The following probability trees show the probabilities for the different capacity cases in each example. These two examples are therefore constructed such that a natural way to cluster the capacity cases is to restrict the forecast to KAP1, KAP2, and KAP3 with respective probabilities 0.4, 0.4, and 0.2.

**Figure 5-11**
We use subroutine GREEDY to obtain a ground holding policy (SOL1) if we assume only the capacity cases KAP1, KAP2, and KAP3. In order to compare with the results that would have been obtained with the complete-nine capacity-case forecast we run GREEDY with the complete set of capacities to obtain an "optimal" solution and we compare the cost of this solution with the expected cost of SOL1 assuming the complete scenario. We do this for both capacity examples. The results are shown in the following table.

<table>
<thead>
<tr>
<th>EXAMPLE 1</th>
<th>EXAMPLE 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GROUND COSTS</td>
</tr>
<tr>
<td>GREEDY</td>
<td>88,700</td>
</tr>
<tr>
<td>SOL1</td>
<td>61,900</td>
</tr>
</tbody>
</table>

Table 5-16

We note from the above table that adopting SOL1 (that corresponds to the clustered forecast) leads to higher total costs than the solution obtained by assuming the complete capacity forecast as shown in the probability trees. The gap in costs is larger for example 2 as can be expected from observing the probability trees. The differences in costs remain however very small.

5.4.6 Value of Additional Landing Slots

The applicability of our problem to landing slot pricing was discussed in Chapter 3. The same approach can be used for evaluating the benefits of increased
capacity and therefore serve as a tool for investment decisions (better ATC, more physical capacity). We utilize the sample problem developed for the comparison with first-come first-served policies (Figure 5-8) and use subroutine GREEDY to determine the Ground Holding Policies. The goal is to compare the total cost obtained from a given forecast with the cost obtained from a capacity forecast that differs from it by one landing slot for a given hour.

The first result is obtained by subtracting one unit from the forecast for hour 6-pm to 7-pm (i.e. using 39, 34, and 29 instead of 40, 35, and 30 for the three capacity cases). The probabilities of the capacity cases are assumed to be 0.4, 0.4, and 0.2. The total expected cost turned out to be $154,420 instead of $152,300 for the full capacity case, an additional cost of $2,120 corresponding to the (marginal) cost of a landing slot during that hour. The same experiment was run with a much more pessimistic capacity forecast assuming the same level of capacities but with probabilities 0.7, 0.2, and 0.1. The result was a slight increase of the marginal cost of the same landing slot to $2,200.

The second result was obtained from the same sample problem but concerned an off-peak landing slot: 9-pm to 10-pm (with probabilities 0.4, 0.4, and 0.2). The resulting cost for the landing slot was $900 compared to the previous $2,120 for the 6 to 7 slot.

These results show that the cost of a landing slot is likely to be more sensitive to demand characteristics (peak conditions) as they vary with the time of day, rather than to the specific capacity forecast of the day in question. This has obvious implications for the feasibility of marginal cost pricing. It indicates that setting prices according to time of day is enough to capture roughly the marginal cost of a landing slot. (A price that would have
to vary with the weather conditions would be more difficult to implement.) Clearly,
further investigation of this topic is needed, if one is to draw any firm conclusions.

5.4.7 Sensitivity of Cost Savings to Level of Utilization of Airport

This experiment is designed to estimate the savings in costs at different utilization
levels when we use the Greedy Heuristic to determine the GHP to be implemented. We
consider operations at airport Z between 7-am and midnight under different levels of
utilization. The experiment assumes a schedule that resembles the actual operation profile
at Boston Logan Airport and displays two peak-demand periods, one morning-peak
between 7-am and 9-am, and one afternoon-peak around 6-pm. This schedule is shown
in Figure 5-12.
The experiment consists of running Greedy on sample problems with different levels of utilization (different total numbers of flights during the 17-hour time span). For each level of utilization, we compute the total expected cost from Greedy and compare with the cost under a simulation of first-come first-served delay policy using the most
likely capacity case as well as with the cost of the no-delay policy. We then evaluate how the cost savings change with this level of congestion.

Figure 5.13 shows the three capacity cases assumed; the probabilities of the cases KAP1, KAP2, and KAP3 are taken to be respectively 0.3, 0.4, 0.3. This demand profile is expressed in terms of hourly percentages of operations and is assumed to remain fixed for each experiment. Approximately 90% of total operations at Logan Airport occur within the 17-hour time span over which we do our analysis. In order to vary the level of utilization of the airport we simply change the total number of landings scheduled during the whole day, keeping the hourly percentages fixed. The low demand time period 9-pm to midnight is added to allow for the landing of all flights that cannot land during the high demand time-period when unfavorable weather conditions prevail. The total number of landings between 7-am and 9-pm under the best conditions is approximately 700 flights corresponding to the maximum capacity of airport Z. We will define the level of utilization of airport Z to be the ratio of the total number of flights scheduled in that time span and 700, that maximum capacity.
Figure 5.13

Figure 5.14 shows the results for different levels of utilization using the demand profile of Figure 5.12. The curve labeled MLTRESH corresponds to using a first-come first-served delay policy with a delay threshold of 15-minutes (no computed delays of less than 15 minutes are implemented) assuming a deterministic capacity corresponding to the most likely capacity case (KAP2). All algorithms were run using a 10-minute period length, linear cost functions, and an air-cost coefficient $KAY = 2$. The total expected costs are computed for all algorithms assuming a first-come first-served landing strategy.
It is evident from these results that the benefit from implementing a ground holding policy obtained using the Greedy Heuristic increases with the level of utilization of the airport. Figure 5-15 illustrates this more vividly by showing the average expected cost savings per flight as a function of the level of utilization of the airport, where the average expected cost savings is defined to be the ratio between the difference of the total
expected cost from MLTRESH and the total expected cost from GREEDY and N, the total number of flights.

![Graph showing average cost savings in $ vs. level of utilization of airport Z.](image)

**Figure 5.15**

### 5.4.8 Sensitivity to Time Period Length

This experiment is intended to analyze the sensitivity of the solution to the length of the individual time periods (PERL) used in the discretization of the time axis. There are two motivations for doing this: one is that the time complexity for all the algorithms associated with this formulation is a decreasing function of PERL, the other that the
feasibility of some of the methods suggested for future research, such as integer
programming approaches, is greatly increased if we can use a large PERL\textsuperscript{12}. We use the
Greedy algorithm to compute the ground delays in the following experiment.

The following approach is taken to evaluate the quality of a ground holding
solution obtained using a given period length PERL:

1. We determine the GHP obtained from running the Greedy Heuristic on a
sample problem using PERL for the discretization of the time axis.

2. We convert the delays of this GHP from multiples of PERL into multiples of 5
minute (this is always possible since the period lengths we consider are themselves
multiples of 5 minutes (10, 15, 20, 30, or 60 minutes)). We set PERL = 5 minutes and
use the subroutine AGCOST to compute the cost of that GHP.

3. We compare this cost with the (generally lower) cost of the solution obtained if
we use PERL = 5 minutes from the start.

The same probabilistic forecast is used for each step. We expect the cost
computed in step 2) to be generally greater than the cost with PERL = 5 computed in step
3) since the 5 minute solution corresponds to the greatest level of precision. The question
is therefore to determine how the magnitude of this difference is affected by our choice of
PERL.

\begin{enumerate}
\item
\item
\item
\end{enumerate}

\textsuperscript{12} This is true not only because the size of the IP decreases significantly with PERL but also
because it becomes reasonable to relax the integrality constraints if PERL is above a given level.
We conduct the experiments on a sample problem identical to that used in Section 5.4.4. The total time span is 9 hours (from 3pm to midnight) and the probabilistic forecast is illustrated in Figure 5-11 in that section.

A first set of experiments consists in considering three different schedules generated as described in Section 5.2 with three different seeds for the random number generator (same hourly demand profile). We compute the GHP for values of PERL of 5, 10, 15, 20, 30, and 60 minutes. The cost of each GHP obtained with a period length greater than 5 minutes is then recomputed using a 5-minute period length and compared with the cost of the original 5-minute solution. Table 5-17 shows the results for the three different seeds in terms of this recomputed cost.

<table>
<thead>
<tr>
<th>PERIOD LENGTH</th>
<th>SEED1</th>
<th>SEED2</th>
<th>SEED3</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>162,690</td>
<td>169,310</td>
<td>161,380</td>
</tr>
<tr>
<td>10</td>
<td>170,410</td>
<td>178,050</td>
<td>169,350</td>
</tr>
<tr>
<td>15</td>
<td>171,670</td>
<td>181,170</td>
<td>172,970</td>
</tr>
<tr>
<td>20</td>
<td>176,130</td>
<td>180,690</td>
<td>174,630</td>
</tr>
<tr>
<td>30</td>
<td>173,560</td>
<td>182,370</td>
<td>175,270</td>
</tr>
<tr>
<td>60</td>
<td>180,150</td>
<td>186,520</td>
<td>177,270</td>
</tr>
</tbody>
</table>

**Table 5-17**

The above Table 5-17 shows that the difference in total expected costs remains relatively small as PERL is made larger. This relative difference in costs never exceeds 10% for period length of less than one hour. The run time of the algorithm on the other
hand varies from about 40 seconds for PERL = 30 to 470 seconds (8 minutes) for PERL = 5.

Using 5-minute periods may correspond to an unnecessary level of detail considering the uncertainty in travel time. It is therefore reasonable to expect that randomness in travel time will attenuate this difference in cost even further in real situations. The second set of experiments is intended to illustrate this point in the following manner: we use the same 3-step approach except that we introduce some uncertainty in travel times when we compute the costs in steps 2) and 3). This is done by adding, for each flight F_i, a random term to the time period to which F_i is assigned according to the GHP when we compute the costs in steps 2 and 3. In the experiment we assume that this random term is a random variable that can take-on three values, +2 time periods (10 minutes late arrival), -2 time periods (10 minutes early arrival), and 0 time periods (on-time arrival), each with probability one-third. Table 5-18 shows the results of this experiment on the same sample problem.
<table>
<thead>
<tr>
<th>PERIOD LENGTH</th>
<th>SEED1</th>
<th>SEED2</th>
<th>SEED3</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>176,950</td>
<td>179,030</td>
<td>173,340</td>
</tr>
<tr>
<td>10</td>
<td>177,910</td>
<td>175,440</td>
<td>174,090</td>
</tr>
<tr>
<td>15</td>
<td>178,510</td>
<td>175,620</td>
<td>172,670</td>
</tr>
<tr>
<td>20</td>
<td>180,830</td>
<td>175,690</td>
<td>174,610</td>
</tr>
<tr>
<td>30</td>
<td>180,040</td>
<td>174,840</td>
<td>172,270</td>
</tr>
<tr>
<td>60</td>
<td>182,150</td>
<td>179,250</td>
<td>171,370</td>
</tr>
</tbody>
</table>

**Table 5-18**

The differences between costs are now very small as expected. In fact the total expected cost *decreases* in some instances when we use larger time periods. We conclude that for all practical purposes the quality of the solution does not depend on what period length we use within the range of this analysis. We should point out however that it may not be practical to use one-hour periods since this would imply that the ground holds have to be multiples of one hour. It seems therefore that if we want to use this formulation to determine those ground holds directly we should use period lengths of 20 or 30 minutes. We should not however exclude the results from one-hour time periods; a lot can be learned if we interpret the results not directly as ground delays for individual flights but rather as an "optimal" number of flights per hour. In this case the solution in terms of ground holds can be determined in a second step by allocating individual flights to this "optimal" number in order to minimize the ground holds (a deterministic problem for which we can use the fast algorithm); in this way we avoid imposing delays of one-hour "chunks" that are insensitive to individual times of departure.
As was mentioned earlier, this analysis is also relevant to the integer programming formulation since the size of the mathematical program depends heavily on PERL. In this case it is reasonable to consider one-hour time periods. The reason for this is that we can in this case consider relaxing the integrality constraints and then apply some deterministic method, such as the one suggested in the previous paragraph, to refine the solution. If this is done, we can use decomposition methods (where sub-problems can be solved using the fast algorithm as is indicated in Chapter 6) to solve the resulting global linear program which has a reasonable size. Note also that further size reductions can be obtained if we can group the aircraft using airport Z into a few categories since this significantly reduces the number of arcs in the networks.
6. CONCLUSION

The main purpose of this thesis was to address the deterministic and static-probabilistic versions of the GHPP. We were able to develop several solution methods for the deterministic GHPP. Optimal solutions to the deterministic GHPP were obtained efficiently through a very fast algorithm that can be used when certain natural conditions are imposed on the delay cost functions. The static-probabilistic GHPP was also solved optimally through a dynamic program; in addition, we developed heuristic algorithms for the static-probabilistic GHPP, some of which use the fast deterministic algorithm as a "building block" for a very efficient solution.

In this chapter we give a summary of the main findings of the thesis and suggest topics for further research as well as alternative applications.

6.1 Summary of Findings

At the outset of this work we investigated the single airport deterministic GHPP. Two "standard" formulations, a minimum-cost-flow in a capacitated network formulation and a minimum-cost-assignment formulation, were proposed. Any minimum-cost-flow or least-cost-assignment algorithm associated with these standard formulations solves the GHPP optimally in the case of general cost functions. Unfortunately, the size of a typical problem limits the practicality of these "standard" solution methods. (However, the integer program associated with the network formulation could play a key role in further research, as will be shown in the next section.) A very fast algorithm, which basically
consists of finding an "optimal" priority rule for assigning available capacity to aircraft, solves the GHPP optimally when the delay cost functions satisfy certain (reasonable) "regularity" conditions.

After a general discussion of some issues related to the probabilistic formulation in Chapter 3, several algorithms were developed for the static-probabilistic single airport case. Among these a Dynamic Program (DP) was developed that yields the "optimal" ground holding policy when a fixed landing priority rule exists for landings at the destination airport. This Dynamic Program suffers, however, from exponential (in the number of time periods) space and time complexities; nonetheless, if several parallel processors and memory management methods are available, this approach can become feasible. Next a computationally more efficient "greedy" algorithm was proposed that also assumes the existence of a fixed landing priority rule. This so-called Greedy Heuristic is not guaranteed to yield an "optimal" solution. Another heuristic, the Limited Lookahead Heuristic, which consists of decomposing a given instance of the GHPP into a sequence of smaller size subproblems and applying the exact DP to each subproblem consecutively, was also developed. Finally we proposed two classes of heuristics for which the basic strategy is to reduce a given instance of the probabilistic GHPP to "equivalent" deterministic GHPP's. Equivalent-Capacity Heuristics do this by aggregating the capacity cases distinguished in the probabilistic capacity forecast into a single capacity case and solving the corresponding deterministic GHPP. Equivalent-Policy Heuristics, on the other hand, consist of solving a number of deterministic GHPP's, each one corresponding to one of the capacity cases distinguished in the probabilistic capacity forecast, and then aggregating the resulting GHP's into a single GHP. These last two classes of heuristics can be applied to problems for which we do not necessarily assume the existence of a fixed landing priority rule.
All the above algorithms can work with general cost functions. However, significant savings in computational complexity can be obtained with the Equivalent-Capacity and Equivalent-Policy Heuristics if we use "regular" cost functions since, in this case, we can use the very fast algorithm devised for the deterministic case as a "building block". On the other hand, the exact DP, the Limited Lookahead DP, and the Greedy Heuristic have computational complexities that do not depend on the type of cost functions that are used. This can become an important consideration for further research since some "downstream" effects (such as a limit on the delay time we can impose on a given aircraft) can be modeled through sharp increases in the individual cost functions beyond a given threshold.

In Chapter 5, we investigated the deterministic and the static-probabilistic algorithms experimentally. Experiments with the deterministic Fast Algorithm show that large savings in total delay costs can be obtained with well-designed GHP's. They also show that we can treat users relatively equitably and still achieve significant savings in total delay costs. Experiments with the probabilistic algorithms show that, when we assume "regular" cost functions, the Greedy Heuristic and the Limited Lookahead DP perform well compared to the Expected-Capacity and Expected-Policy Heuristics (these yield GHP's with costs that are 10% and 30% higher, respectively). The Limited Lookahead DP was however much slower than the Greedy Heuristic which ran in less than one minute on a microcomputer for realistic instances of the problem involving several hundred flights. Further experiments show that, when the number of capacity cases used in the probabilistic forecast is low (three cases or less), an Equivalent-Capacity Heuristic which corresponds precisely to one of the capacity cases of the probabilistic capacity forecast performs surprisingly well.
The interpretation of this result led us to investigate a similar behavior for the single time period case. We showed that, when congestion occurs during only a single time period and when the airport is operating under congested conditions, an optimal GHP for the single time period case can always be found that consists of solving an "equivalent" deterministic problem that corresponds to one of the capacity cases of the probabilistic forecast. Although this result does not extend to the multiperiod case, as is shown by one of the experiments where no "equivalent" capacity case performs as well as the Greedy Heuristic, one can nevertheless interpret the experimental results for the Equivalent-Capacity Heuristics as extensions of this "dominating" capacity case behavior to the multiperiod case when the probabilistic capacity forecast consists of three capacity cases or less. When the probabilistic capacity forecast involves four cases, the experiments still show that an "equivalent" capacity case can be found that performs reasonably well (although not as well as the Greedy Heuristic). This result is confirmed by an experiment (Capacity Forecast Clustering in Section 5.4.5) that shows that the increase in total delay costs resulting from using a GHP that is obtained from deliberately reducing the number of capacity cases distinguished in a given capacity forecast is relatively small.

Another experiment in Chapter 5 shows that when we assume general cost functions the exact DP approach can outperform the Greedy Heuristic as can be expected. Because of the time/space complexity of the exact DP, this could only be demonstrated on a sample problem of limited size. However, this result shows that the exact DP approach should be considered for further research, particularly if a faster computer and virtual memory techniques are available.
Further experiments in Chapter 5 were intended to evaluate, as was done in the deterministic case, potential savings in total (expected) delay costs from using algorithmic approaches in the probabilistic case. These experiments used the Greedy Heuristic (since it performs as well as any other heuristic) and compare the results with assigning aircraft to capacity on a first-come first-served basis using a deterministic capacity forecast. The results show that significant savings in total expected delay costs are still obtained even when the delays are distributed relatively equitably among classes of users (by using a first-come first-served landing priority rule, for example). Furthermore, an experiment (Section 5.4.7) showed that the average cost savings increase as the airport becomes more congested.

Some numerical experiments were designed to show the use of the probabilistic formulations for planning purposes. One of these experiments showed how one can go about evaluating the potential benefit of increasing the reliability of the capacity forecast mechanism in use. The other showed how to compute the marginal benefit of an additional landing slot. Such results can be used to compare the benefits of investments in forecast mechanisms and ATC technologies with their costs.

Finally, another numerical experiment shows that the quality of the GHP obtained from optimization methods is not very sensitive to the size of the period length used in the discretization of the time axis (for period length of up to one hour) when we take the uncertainty in travel times into account. This has desirable implications in terms of the complexity of all the algorithms considered in this thesis and, in particular, in terms of the feasibility of an Integer Programming approach to the multi-airport case that is developed in the next section.
After extending the probabilistic formulation with a model for the updating of
information over time (in Chapter 3) we were able to propose, in Chapter 4, several
solution methods for the dynamic-probabilistic single airport case. A Stochastic Dynamic
Program (SDP) inspired by the "static" DP was developed. This SDP yields the
"optimal" GHP when we assume that a fixed landing priority rule is in use and when we
impose the condition that ground holds cannot be revised once determined. However,
this SDP too suffers from exponential time and space complexities but can also benefit
from parallel processing and virtual memory techniques. We also proposed "dynamic"
heuristics based on "static" solution methods and for which the fast deterministic
algorithm can also be used as a "building block". These algorithms, when complemented
with a model for the updating of information, could be investigated computationally as
part of further research.

The results of this thesis therefore allow us to feel optimistic about the usefulness
of algorithmic approaches to the GHPP. The success of such approaches will ultimately
depend on how well they can be integrated into the existing ATC system. In Chapter 3
we sketched a possible scenario for such a complete implementation. Success will also
depend on ability to treat users relatively equitably while achieving aggregate cost
reductions. On this count too we feel optimistic; some of our numerical examples show
that significant cost savings occur even when ground and airborne delays are redistributed
fairly equitably among classes of aircraft.

Another issue likely to determine the practical success of algorithmic approaches
is their ability to consider "downstream" effects. By "downstream" effects we mean
situations in which delaying the arrival time of a given aircraft at airport Z has
implications that go beyond the direct cost of this delay because, for example, the same
aircraft is used for another flight originating at airport Z with a short turnaround time. Situations like these can be modeled through a sharp increase in delay cost beyond a certain level of total (ground+air) delay. The resulting cost functions are likely, however, to violate the "regularity" conditions and thus algorithms based on the Fast Algorithm can no longer be used. On the other hand the Greedy and Limited Lookaheed Heuristics could be used since, as mentioned previously, the time complexities of these heuristics are not affected by the shape of cost functions (the practicality of the latter heuristic still depends on the availability of parallel processors).

An alternate way to tackle the "downstream" effects is to use a multi-airport formulation of the GHPP. Appendix D proposes a deterministic multi-airport formulation.

6.2 Possible Extensions and Alternative Applications

The following issues have not been investigated in depth and are important areas for further work:

- Experiments with general cost functions. As mentioned above, some "downstream" effects can be modeled through delay cost functions. The desirability of using alternative cost functions could be explored through numerical examples using some of the algorithms developed in this thesis.

- Investigation of the dynamic aspect of the GHPP. This issue includes several important topics such as how and when GHP's should be revised and should possible revisions and capacity forecast updates be explicitly considered in the GHPP formulation (since this tends to penalize short haul flights). The dynamic aspect of the GHPP is
therefore closely related to implementation issues which should also be addressed by further research.

- Investigations of network extensions of the single airport GHPP. As mentioned previously, Appendix D points to one possible direction for tackling the multi-airport GHPP.

In consequence, we propose the following outline for further research:

- Use some of the algorithms developed in Chapter 4 with general cost functions in an experimental setting similar to the one used in Chapter 5.

- The same experimental setting, complemented with a model of a dynamic probabilistic forecast, can be used to investigate the "dynamic" probabilistic algorithms of Section 4.3.

Appendix C contains an informal discussion of a number of mathematical programming formulations of the single airport probabilistic GHPP that could lead to "dynamic" solutions to the GHPP. In particular, we develop a zero-one IP for the single airport probabilistic case. This is done approximately in the following manner:

- We consider each capacity case of the probabilistic capacity forecast and generate the IP that corresponds to that (deterministic) case as is done in Section 2.1.1.

- We aggregate each one of these separate IP's into a global IP that minimizes an expected cost. This is done by using an objective function that is the weighted sum of the individual objective functions and adding some consistency constraints (that reflect the
fact that the assignments of aircraft to time periods are done before knowing which capacity case materializes)¹.

The "dynamic" IP solutions to the GHPP developed along these lines could then be compared with some of the "dynamic" algorithms of Section 4.3.

• Investigate multi-airport probabilistic formulations. This could be done by extending the single airport probabilistic IP formulations to multi-airport formulations as is done in Appendix D for the deterministic case.

The probabilistic GHPP can be viewed as a "decision timing" problem. The central planner is faced with timing a sequence of decisions, each decision corresponding to allowing an individual flight to take-off, for which the trade-off is between the cost of delaying the decision excessively (ground delays—"pessimistic" GHP's) and an unknown (stochastic) cost of making the decision prematurely (expected airborne delays—"optimistic" GHP's). There are many problems which can be described as "decision timing" problems and that could benefit from solution methods for the GHPP. We mention a few problems here; the parenthetical remarks are intended to show how each problem is related to the GHPP:

• The Inventory Management Problem (IMP), which consists of the proper timing of the production by a given firm of several items to maximize receipts from sale minus inventory costs. The uncertainty is in the demands for each item (airport capacities) and the trade off is between "pessimistic" production strategies (excessive

---

¹ The resulting IP is highly block angular in structure and can therefore take advantage of the availability of very fast deterministic solution methods.
ground holds) which may result in lost demand (lost capacity) and "optimistic" production schedules (small ground delays) which may result in excessive inventory costs because of low demands (airborne delay costs because of low capacity). We should note here that the GHPP has more structure than the IMP since the different aircraft are competing for the same physical capacity. This gives a combinatorial dimension to the problem and makes it more difficult to deal with. Such a combinatorial aspect could be introduced in the IMP if we assumed that the different items produced are also competing for the same demand. This can happen if the firm is producing items that are substitutable as is the case, for example, for a company that manufactures different versions of the same basic microcomputer. The following problem can also be viewed in the context of a company producing the same basic item.

- An important part of an airline Yield Management Problem (YMP) consists of determining, for a given flight, when to close the reservation system to bookings for each type of tickets (Coach seat, First Class seat, Business Class seat, etc...) sold on that flight. The uncertainty here is on the demand for each type of ticket and the trade-off is between "optimistic" strategies that start refusing reservations for low price seats too early (small ground delays) in the hope of getting more bookings for higher price seats and "pessimistic" strategies that do not keep enough seats available for possible high price seats bookings (excessive ground delays) and may result in lost sales (unused airport capacity). The combinatorial dimension is present in this problem since the different ticket classes are competing for the same overall seating capacity.

Finally, it should be noted that all the above problems are short term versions of a longer term (complementary) investment problem which consists:
- for the IMP, of determining the level of overall production (the number of production units to invest in) that strikes an "optimal" balance between the long term inventory costs and the benefits of sales over the long term, given the stochastic variations in short term demands.

- for the YMP, of determining which aircraft to operate for the given flight such that the total seat capacity of the aircraft strikes an "optimal" balance between the cost of operating the aircraft and actual receipts from ticket sales over the long term, given the stochastic variations in demand for tickets on that flight.

The GHPP also allows a longer term formulation which, given the stochastic description of the changes in capacity of airport Z from day to day, consists of determining the overall level of operations at airport Z that strike an "optimal" balance between the amount of (unavoidable) delays that result from this variability in capacity and the benefits resulting from operating airport Z. This longer term problem can be though of determining an "optimal" schedule of operations at airport Z. A mathematical programming formulation of this problem is proposed at the end of Appendix C.

Finally, it should be stressed that, as is the case for all the other related problems mentioned above, the short term version of the GHPP is a problem that remains relevant to every day operations and that has to be solved on a continuous basis, even if the schedule of operations at airport Z corresponds to an "optimal" schedule. In this sense, the GHPP we have considered in this thesis is the "dynamic" version of another fundamental scheduling problem. Further research in this area could therefore also consider longer term scheduling issues.
APPENDIX A: "Fully Dynamic" Formulation

This appendix contains a mathematical formulation of the "fully dynamic" GHPP mentioned in Section 3.2.

We build on the notation used in Chapter 3 and define a trajectory of the system to be a function \( \mathcal{T}: t \rightarrow \mathcal{O}(t) \) from \([0, t_{P+1}]\) to the set of all possible observations. A trajectory of the system up to time \( t \) is defined to be the restriction, \( \mathcal{T}|_{[0,t]} \), of a trajectory function \( \mathcal{T} \) to the interval \([0,t]\); a Static Ground Holding Policy up to time \( t \) is defined to be a sequence \( \{\text{stat}_0, \text{stat}_{t+1}, ..., \text{stat}_{\Pi_t}\} \). In this context, the following definitions for the dynamic version of the problem can be made:

A Dynamic Generalized Ground Holding Policy (DGGHP) is a function that, at any time \( v \), associates a Static Generalized Ground Holding Policy up to time \( v \) \( \{\text{stat}_0, \text{stat}_{t+1}, ..., \text{stat}_{\Pi_v}\} \) to each possible trajectory \( \mathcal{T}|_{[0,v]} \) of the system up to time \( v \). Furthermore we require this DGGHP to be "consistent" in the sense that, for any non-negative number \( v' \), if \( \mathcal{T}|_{[0,v]} \) and \( \mathcal{T}|_{[0,v'+v]} \) are two trajectories that "agree" on \([0,v]\) then the corresponding SGGGHP must also agree up to time \( v' \). In this context a dynamic optimization at time \( t \) is an algorithm that associates to a history of situations up to time \( t \), \( \{S(u) = \{\text{stat}_u, \text{PK}(u)\}\}_{0 \leq u \leq t} \), and a trajectory of the system up to time \( t \) a DGGHP that is defined on the appropriate subset of the set of trajectories (the subset that corresponds to all the trajectories that have the same restriction on \([0,t]\)). A DGGHP is to be applied in the following way:
- for each time period $T_j$ after $T_l$ we measure the state of the system, represented by a trajectory up to time $t_{j-1}$, and we extract, from the DGGHP, the SGGHP that correspond to that state of the system.

- for each flight $F_i$ still on the ground or still airborne at the beginning of time period $T_j$ we read off the status, $\text{stat}(F_i)$, for that flight from the SGGHP. If this status is equal to 0, keep $F_i$ on the ground for the duration of time period $T_j$. If time period $T_j$ is the first time period such that this status is equal to 1, allow the flight $F_i$ to take off. If time period $T_j$ is the first time period such that this status is equal to 2, allow flight $F_i$ to land.
APPENDIX B: Proof of "Dominating Capacity Case" Property.

This appendix contains proofs of the three claims of Section 5.4.2.

Proof of claim 1:

Consider a policy \( P = \{ F_{i_1}, F_{i_2}, \ldots, F_{i_R} \} \cup \{ F_{j_1}, F_{j_2}, \ldots, F_{j_{N-R}} \} \). We assume that the indexing is such that \( i_{j+1} > i_j \) and \( j_{i+1} > j_i \) for all \( i \) and \( j \). We consider three cases:

Case 1: \( R < K_1 \).

Set \( Z = K_1 - R \) and consider the following policy:

\[
Q = \{ F_{i_1}, F_{i_2}, \ldots, F_{i_R}, F_{j_1}, F_{j_2}, \ldots, F_{j_Z} \},
\]

that sends exactly \( K_1 \) flights on time to airport \( Z \). We can always do that since we assumed that \( N > K_1 \). We have:

\[
TC(P) - TC(Q) = \sum_{i=1}^{Z} C_{g_{ji}} > 0
\]

and therefore policy \( Q \) has a total expected cost strictly lower than that of policy \( P \).

Case 2: \( K_C \leq R \):

If \( R = K_C \), there is nothing to prove. If \( R > K_C \) the policy

\[
Q = \{ F_{i_1}, F_{i_2}, \ldots, F_{i_{K_C}}, F_{i_{K_C+1}}, \ldots, F_{i_R}, F_{j_{Z+1}}, \ldots, F_{j_{N-R}} \}
\]

which sends exactly \( K_C \) flights on time to airport \( Z \), is such that:
\[ TC(P) - TC(Q) = \sum_{i=K_C+1}^{R} C_{ai} > 0 \]

**Case 3:** \( \exists s \) such that \( K_s \leq R < K_{s+1} \): We consider two possibilities:

(i) \( k(p_1+p_2+...+p_s) > 1 \). Consider policy

\[ Q = \{ F_{i_{R-K_s+1}}, ..., F_{i_R}, F_{j_1}, F_{j_2}, ..., F_{j_{N-R}}, F_{i_1}, ..., F_{i_{R-K_s}} \} \]

which sends exactly \( K_s \) flights on time to airport \( Z \). We have:

\[ TC(P) = \sum_{u=1}^{N-R} C_{g_{ju}} + \sum_{j=1}^{S} [ \sum_{u=1}^{R-K_j} k \cdot C_{g_{iu}} ] \cdot p_j \] and

\[ TC(Q) = \sum_{u=1}^{N-R} C_{g_{ju}} + \sum_{u=1}^{R-K_s} C_{g_{ju}} + \sum_{j=1}^{S} [ \sum_{u=R-K_s+1}^{R-K_j} k \cdot C_{g_{iu}} ] \cdot p_j \Rightarrow \]

\[ TC(Q) - TC(P) = \sum_{u=1}^{R-K_s} C_{g_{ju}} - \sum_{j=1}^{S} [ \sum_{u=1}^{R-K_s} k \cdot C_{g_{iu}} ] \cdot p_j . \] We can rewrite this difference:

\[ TC(Q) - TC(P) = \sum_{u=1}^{R-K_s} C_{g_{ju}}(1-k(p_1+p_2+...+p_s)) \] which is a negative quantity since we assumed \( k(p_1+p_2+...+p_s) > 1 \). Therefore \( TC(Q) < TC(P) \).

The other possibility is:

(ii) \( k(p_1+p_2+...+p_s) \leq 1 \). In this case we consider policy
\[ Q = \begin{cases} F_{i_1}, F_{i_2}, \ldots, F_{i_R}, F_{j_1}, \ldots, F_{j_{K_{s+1}-R}} \\
F_{j_{K_{s+1}-R+1}}, \ldots, F_{j_N} \end{cases} \]

which sends \( K_{s+1} \) flights on time. We have:

\[
TC(Q) \leq \sum_{u=K_{s+1}-R+1}^{N-R} C_{g_{ju}} + \sum_{j=1}^{S} \left[ \sum_{u=1}^{R-K_j} k \cdot C_{g_{iu}} + \sum_{v=1}^{K_{s+1}-R} k \cdot C_{g_{jv}} \right] \cdot p_j .
\]

We have an inequality here, since the cost on the right hand side of the inequality assumes that we are not necessarily using an optimal landing strategy; it assumes that, when the capacity turns out to be less than \( K_{s+1} \), we do not allow any of the flights \( F_{j_1}, F_{j_2}, \ldots, F_{j_{K_{s+1}-R}} \) to land. (Finding the optimal landing priority rule would involve reordering the flights \( F_{i_1}, F_{i_2}, \ldots, F_{i_R}, F_{j_1}, F_{j_2}, \ldots, F_{j_{K_{s+1}-R}} \) in increasing order or delay cost; but an inequality is enough for the proof.)

We therefore have:

\[
TC(Q) - TC(P) \leq \sum_{j=1}^{S} \left[ \sum_{u=1}^{K_{s+1}-R} k \cdot C_{g_{iu}} \right] \cdot p_j - \sum_{u=1}^{K_{s+1}-R} C_{g_{ju}} .
\]

We can rewrite the inequality:

\[
TC(Q) - TC(P) \leq \sum_{u=1}^{K_{s+1}-R} C_{g_{ju}} \left[ k(p_1 + p_2 + \ldots + p_S) - 1 \right] \text{ which implies:}
\]

\[
TC(Q) \leq TC(P).
\]

This completes the proof of Claim 1.

In order to prove Claim 2 we need to prove the following properties of optimal policies:
Property 1: Any flight $F_i$ that is sent on time to airport $Z$ in an optimal policy satisfies:

$p(i)\cdot C_{a_i} \leq C_{g_i}$, where $p(i)$ is the probability that flight $F_i$ is delayed.

Proof: We proceed by contradiction and assume there exists an optimal policy $P$ such that, for some flight $F_i$ sent on time by $P$, we have $p(i)\cdot C_{a_i} > C_{g_i}$. Then the policy $Q$ that differs from $P$ by keeping that flight on the ground has total cost less than $P$ (because doing this cannot "hurt" any other flight and substracts $p(i)\cdot C_{a_i}$ from the cost), contradicting the assumption that $F$ is optimal.

Property 2: There exists an optimal policy $Q$ that keeps a number $R$ of the "cheapest" flights $F_1,\ldots,F_R$ on the ground.

Proof: Consider any optimal policy $P$ as follows:

$$P = \begin{cases} F_{i_1}, F_{i_2}, \ldots, F_{i_R} \\ F_{j_1}, F_{j_2}, \ldots, F_{j_{N-R}} \end{cases}$$

If $j_{N-R} = N-R$, then there is nothing to prove.

If $j_{N-R} \neq N-R$, then there exists an index $i_k$ such that $i_k < j_{N-R}$. Let us note $i = i_k$ and $j = j_{N-R}$.

Consider the policy $Q$ obtained from $P$ by switching the roles of flights $F_i$ and $F_j$ (i.e., according to policy $Q$, flight $F_i$ is kept on the ground and $F_j$ is sent on time). If $p(i)$ is the probability that $F_i$ is delayed in the air for policy $P$, then $p(i)$ becomes the probability that $F_j$ is delayed in the air for policy $Q$; We have:
TC(P) - TC(Q) = Cg_j + p(i)Ca_i - Cg_i + p(i)Ca_j = (1-p(i)k)(Cg_j - Cg_i)

But we have:

$p(i)k \leq 1$ by Property 1 and $Cg_j \geq Cg_i$ since index $j$ is greater than $i$. Therefore $TC(Q) \leq TC(P)$. But since we started with an optimal policy $P$, we must have:

$TC(Q) = TC(P)$.

Now we can proceed by switching flights in this manner until we obtain the policy $P'$ that keeps the first $N-R$ flights on the ground and has cost equal to $TC(P)$. Policy $P'$ is therefore optimal, proving Property 2. Note that the new optimal policy obtained through this procedure sends the same number of flights on time as does the optimal policy we started with.

We can now proceed with the proof of Claim 2:

**Proof of Claim 2:**

A direct consequence of Claim 1 is that there exists an optimal GHP that sends exactly $K_{s+1}$ flights on time for some $j=0,\ldots,C-1$. (We use index $s+1$ to keep the same notation as in the statement of Claim 2.)

The proof of Property 2 shows that the policy

$$P = \begin{cases} F_{N-K_{s+1}+1}, \ldots, F_N \\ F_1, \ldots, F_{N-K_{s+1}} \end{cases}$$

that sends the $K_{s+1}$ flights of highest indices $F_{N-K_{s+1}+1}, \ldots, F_N$ on time to airport $Z$ is also optimal.
There are two cases to examine:

**Case 1:** There is no index $u$ (1 ≤ $u$ ≤ $C$) such that $k \cdot (p_1 + p_2 + ... + p_u) = 1$, then:

(i) if $s = 0$, we must have $k \cdot p_1 > 1$. This can be shown by contradiction in the following way:

Assume that, on the contrary, we had $1 \leq p_1 \leq 1$. Then by (1) we must have $k \cdot p_1 < 1$ and the policy $P$ would not be optimal since we could send more than $K_1$ flights with a net reduction in total expected cost.

(ii) if $s \neq 0$, then there is a capacity case $K_s$. The expected cost of flight $F_{N \cdot K_s+1+1}$, $EC(F_{N \cdot K_s+1+1})$, is given by:

$$EC(F_{N \cdot K_s+1+1}) = k \cdot C_{G_{N \cdot K_s+1+1}} \cdot (p_1 + p_2 + ... + p_s),$$

which, according to Property 1 must be ≤ $C_{G_{N \cdot K_s+1+1}}$. Therefore we must have:

$$k \cdot (p_1 + p_2 + ... + p_s) ≤ 1.$$  

Since we cannot have equality according to (1) we must have $k \cdot (p_1 + p_2 + ... + p_s) < 1$. Furthermore we must also have $k \cdot (p_1 + p_2 + ... + p_{s+1}) > 1$ for otherwise policy $P$ would not be optimal. The reason for this is that, if we had $k \cdot (p_1 + p_2 + ... + p_{s+1}) < 1$, we could send more flights on time for a net gain in total expected costs.

The second case to examine is:

**Case 2:** If there is an index $u$ such that $k \cdot (p_1 + p_2 + ... + p_u) = 1$. (We know that, since $k > 1$, we have $u \neq C$. Also remember that we assumed $C \geq 2$) Again we have to examine two cases:
(2.1) Case $s = 0$. In this case we must have $kp_2 > 1$, since otherwise, if $kp_2 \leq 1$, policy $P$ would not be optimal (we could send $K_2$ flights on time for a net gain in total expected cost). We conclude that $u \leq 1$ and therefore that $u = 1$. In this case all policies of the form

$$S(m) = \begin{cases} \{F_{N-K_1-m+1}, \ldots, F_N\} & \text{for some } m, 0 \leq m \leq K_2 - K_1 \text{ are optimal.} \\ F_1, \ldots, F_{N-K_1-m}\end{cases}$$

The reason for this is that the expected cost for the $m$ additional flights $F_{N-K_1-m+1}, \ldots, F_{N-K_1+2}$ is exactly zero since we have $kp_1 = 1$.

(2.2) Case $s \neq 0$. We know that we have $k(p_1 + p_2 + \ldots + p_s) \leq 1$. Then, either $k(p_1 + p_2 + \ldots + p_s) = 1$ and $s = u$ and all policies of the form

$$T(m) = \begin{cases} \{F_{N-K_{s+1}-m+1}, \ldots, F_N\} & \text{for some } m, 0 \leq m \leq K_{s+1} - K_s \text{ are optimal, or} \\ F_1, \ldots, F_{N-K_{s+1}-m}\end{cases}$$

$k(p_1 + p_2 + \ldots + p_s) < 1$ and $s$ has to be equal to $u - 1$ (otherwise there would be a policy that sends $K_{s+2}$ flights on time with a lower cost) and all policies of the form

$$Y(m) = \begin{cases} \{F_{N-K_{s+1}-m+1}, \ldots, F_N\} & \text{for some } m, 0 \leq m \leq K_{s+1} - K_s \text{ are optimal.} \\ F_1, \ldots, F_{N-K_{s+1}-m}\end{cases}$$

In all cases (2.1) and (2.2) the policy

$$S(m) = \begin{cases} \{F_{N-K_{u+1}-m+1}, \ldots, F_N\} & \text{is optimal for any } 0 \leq m \leq K_{u+1} - K_u, \text{ completing} \\ F_1, \ldots, F_{N-K_{u+1}-m}\end{cases}$$

the proof of Claim 2.
Proof of Claim 3:

The following shows that Claim 3 is a special case of Claim 2.

If \( N \geq K_C \), there is nothing to prove.

If \( N < K_C \), add \( K_C - N \) "dummy" flights with zero ground cost (and therefore zero airborne delay cost). The new problem satisfies the conditions of Claim 2. Claim 3 corresponds to the interpretation of the results of Claim 2 ignoring the "dummy" flights.
APPENDIX C: Mathematical Programming Formulations

This appendix is an informal discussion of some mathematical programming formulations related to the single airport probabilistic problem.

C.1 Formulations for the GHPP

It is easy to see that the "ground holds only" formulation which has been used in this thesis applies also in the case of generalized ground holds (see Section 3.2) when we assume that the uncertainty in capacities $K_1, K_2, ..., K_P$ is fully resolved all at once at some time $t$ (before $t_1$): it suffices to replace the ground holding costs for the flights that are airborne (with stat = 1) with airborne delay costs. To illustrate this it is useful to consider the deterministic case again. Let us consider a situation where a deterministic forecast for the capacities $K_1, K_2, ..., K_P$ is available and we are considering an optimization instance at a time $t$ close to $t_1$. In this case it is likely that a large number of flights have already experienced some ground holds or are already airborne. The following figure is identical to Figure 2-1 except that the costs for certain arcs are different. Now there are three cases for assigning costs to assignment arcs:

\[ \text{\textsuperscript{1} Determined during a previous optimization instance for example.} \]
- Flights that have a scheduled departure time greater than t are such that their arc-costs start at $C_{gi}(0) = 0$. This is the case of flight i in Figure B-1. We will call these delays "type 1" delays in the future.

- Flights that are already being held on the ground at time t are such that the cost of their first assignment arc is $C_{gi}(x+1)$ where x is the ground hold that has already been experienced. Flight j of Figure B-1 is an example. We will refer to these delays as "type 2" delays.

- Flights that are already airborne at time t are such that the cost of assignment arcs start at $C_{ak}(x,1)$, where x is the ground hold that has already been experienced. Flight k is the corresponding example in Figure B-1. We will call these airborne delays "type 3" delays in the remainder.
If, in the formulation corresponding to the above figure, we assume that the capacities are deterministic it becomes necessary to consider type 3 delays explicitly. We will see that it is preferable not to do so in the probabilistic case. Furthermore we have two options if, at time $t$, we decide to impose a type 3 delay on a given flight: we can either reduce the speed of the aircraft starting at time $t$ or we can defer the hold until the flight reaches the vicinity of airport $Z$. In the deterministic formulation it does not matter when we impose these airborne delays. For the probabilistic case, however, the superiority of the last option is evident since we acknowledge some uncertainty in the capacity forecast: not imposing airborne holds early allows us to take advantage of
favorable capacity cases. Moreover in the probabilistic case we must take airborne delays into account when computing expected costs associated with a given ground holding policy. These airborne delays are considered when we are confronted, during a given time period, with an actual capacity that is lower than the number of flights that show up during that time period. These are therefore delays implemented as holding patterns at the vicinity of airport Z. We will refer to them as "type 4 delays". But, as we argued above, the airborne delays determined from the optimization should also be imposed as late as possible; there is therefore no distinction between these type 3 and type 4 delays in the probabilistic formulation. The implication is that there is no need to consider type 3 (airborne) delays explicitly in the context of the probabilistic formulation. They will be taken care of implicitly through the computation of expected airborne delays. These remarks do not apply however to type 2 delays which should be considered explicitly.

As will be discussed in the next section the computation of these expected airborne delay costs corresponding to a given ground holding policy is one of the major difficulties associated with the probabilistic formulation which make it more than a trivial extension of the deterministic formulation.

The preceding discussion touched on one of the major difficulties associated with the probabilistic formulation. Whereas the deterministic formulation consists of a one-stage assignment problem, the probabilistic formulation involves a set of second-stage problems corresponding to sequencing arriving flights at airport Z for landing for each capacity case and revising ground holds. This set of second-stage problems may be far from trivial in the general case if no simplifying assumptions are made. In the following we will refer to this set of second-stage assignment problems simply as the second-stage problem. A full discussion of these issues will come later. It is enough to observe at this stage that even if we assume the existence of a fixed landing priority rule, which
simplifies greatly the determination of expected delays by reducing the second-stage problem to sorting flights arriving at airport Z according to that fixed rule before assigning them to available capacity, the computation of total expected costs associated with a given ground holding policy is, at best, time consuming. Furthermore there is no "short-cut" in figuring out the cost of a ground holding policy as a function of another ground holding policy that differs from the original one by the reassignment of only a few flights. Changing the assignment of only one flight, for example, can lead to changing the landing status of several other flights since it can affect both flights with lower landing priority as well as those with higher landing priority. The complexity of the second-stage problem not only reduces greatly the chances of finding an efficient\textsuperscript{2} exact solution to the probabilistic formulation but also excludes a whole class of heuristics, namely local-search heuristics. A necessary condition for the success of any local-search heuristic is to be able to quickly compute the cost of a solution given the cost of a "neighboring" solution. In the case of the famous Travelling Salesman Problem, for example, local-search methods such as Simulated Annealing or 2-opt or 3-opt local-search are successful because successive solutions differ in only a few arc costs. As we just pointed out this is not the case for the problem at hand.

The interplay between the first-stage and the second-stage problem is precisely illustrated by the following attempt to use Stochastic Programming techniques for an exact solution motivated by the integer programming formulation of Chapter 2.

\textsuperscript{2} Faster than complete enumeration.
C.1.1 Stochastic Programming - Multi-period Case:

In Chapter 2 we introduced the following integer programming formulation for the deterministic version:

\[
\min \sum_{i=1}^{N} \sum_{j \in P_i} C_{ij} x_{ij} \tag{1}
\]

subject to:

\[\forall (i,j) : x_{ij} = 0 \text{ or } 1 \] \tag{2}

\[\forall i (=1,N) : \sum_{j < P_i} x_{ij} = 0 \tag{3}\]

\[\sum_{j \geq P_i} x_{ij} = 1 \tag{3}\]

\[\forall j (=1,P) : \sum_{i=1}^{N} x_{ij} \leq K_j \tag{4}\]

If we exclude the integrality constraints (2) the matrix notation of this IP is:

\[
\min \text{ } C\mathbf{X} \\
\text{s.t. } \Delta \mathbf{X} \leq \mathbf{K} \\
\text{(This formulation includes (1), (3), and (4).)}
\]

We now assume that the right hand side $\mathbf{K}$ is a random vector that takes on a finite number of values (corresponding to the different capacity cases). Furthermore we assume that at time $t_1$, when the value of $K_1$ is known, the values of $K_2$, $K_3, ..., K_P$ also become known with certainty. We make this assumption to be able to formulate the problem as a two-stage linear programming problem which is sufficient to illustrate the
basic steps of this approach. We will indicate later how the same procedure can be
generalized to produce a multi-stage program in the most general case. We also note that
we made the simplifying assumption that the second-stage is only concerned with
airborne delays; the formulation assumes that at time \( t_1 \) the ground delays are not dealt
with any more. This assumption was made only to simplify the notation; the formulation
for the case where this assumption is relaxed goes through almost identically except that
we have to introduce a new set of variables corresponding to type 2 delays.

For any realization of \( K \), say \( K^0 \), and for any choice of a ground holding policy
\( X \), say \( X^0 \), we are left with a second-stage problem that corresponds to the assignment
of flights to available capacity to minimize air delay costs given \( X^0 \) and \( K^0 \). This second-
stage problem is identical to the deterministic formulation of the original ground holding
policy problem except that all ground holding costs are replaced by airborne costs. It
corresponds therefore to another integer program. This formulation is akin to the
classical two-stage stochastic linear (integer in our case) program with recourse. The
recourse action corresponds to the response of air traffic controllers to a random even:
that determines the value of \( K^0 \) and can be thought of as an adjustment of the original
solution \( X^0 \) to minimize penalty costs corresponding to further airborne delays. In some
instances integrating the formulation of the recourse action with the original mathematical
program yields another linear program. For example El Agizy [1] shows that the
Hitchcock-Koopman's transportation problem with uncertainty in demand is reducible to
a single linear program such that the constraint matrix remains totally-unimodular. We
will have no such luck here.

For illustration purposes we will assume that the vector \( K \) takes on only two
values \( K^1 = (K_1^1, K_2^1, \ldots, K_n^1) \) and \( K^2 = (K_1^2, K_2^2, \ldots, K_p^2) \) with probabilities \( p_1 \) and
\( p_2 \). Let us first consider the second-stage problem: we assume that \( X = (x_{ij} \mid i=1, \ldots, n; j=1, \ldots, p) \).
\(j=1,\ldots,P+1\) representing an assignment of each flight \(F_i\) to a time period \(T_j\) is given and that a capacity \(K = (K_1, K_2, \ldots, K_P)\) is realized. The second-stage problem is to find \(y = (y_{ij} \mid i=1,\ldots,N; j=1,\ldots,P+1)\), where \(y_{ij} = 1\) if \(F_i\) actually lands during \(T_j\) and \(y_{ij} = 0\) if not, that minimizes the total air-delay costs. We recognize another assignment problem \(Q(x, K)\):

\[
\min \sum_{i=1}^{N} \sum_{j=P_i}^{P+1} \frac{C_{ij}^x}{x} y_{ij}
\]

subject to:

\(\forall i (=1,N)\):

\(\sum_{j<P_i} y_{ij} = 0\)

\(\sum_{j\geq P_i} y_{ij} = 1\)

\(\forall j (=1,P)\):

\(\sum_{i=1}^{N} y_{ij} \leq K_j\)

where \(P_i^x\) is defined to be the time period during which flight \(F_i\) shows up for the first time for landing at airport \(Z\) according to the ground holding policy \(x\) (i.e., \(P_i^x = P_i + x_i\), where \(x_i\) is the ground delay imposed on \(F_i\) according to \(x\) and \(P_i\) is the index of its scheduled landing period). The cost coefficient \(C_{ij}^x\) corresponds to the airborne cost of assigning flight \(F_i\) to actually land during time period \(T_j\) given the ground hold it has already experienced according to \(x\) (i.e., \(C_{ij}^x = C_i(x_{ij} - P_i - x_i)\)). We also note that the assignment constraints, corresponding to constraints (3) of the original problem, have to be split into two sets of constraints to insure that no flight is assigned to land before it gets to airport \(Z\) according to \(x\).
The two second-stage problems \( Q(x, K^1) \) and \( Q(x, K^2) \) can now be combined with the first-stage problem to formulate the global mathematical program as follows:

find \((x, y^1, y^2)\), where \( y^1 \) and \( y^2 \) are the airborne-delay vectors corresponding respectively to \( Q(x, K^1) \) and \( Q(x, K^2) \), to solve:

\[
\min \left\{ \sum_{i=1}^{N} \sum_{j=P_i}^{P+1} C_{ij} x_{ij} + p_1 \left( \sum_{i=1}^{N} \sum_{j=P_i}^{P+1} C_{ai} y^1_{ij} \right) + p_2 \left( \sum_{i=1}^{N} \sum_{j=P_i}^{P+1} C_{aj} y^2_{ij} \right) \right\}
\]

subject to:

\( \forall \ i := (1, N) : \)

\[
\sum_{j < P_i} x_{ij} = 0 \quad ; \quad \sum_{j \geq P_i} x_{ij} = 1
\]

\[
\sum_{j < P_i} y^1_{ij} = 0 \quad ; \quad \sum_{j \geq P_i} y^1_{ij} = 1
\]

\[
\sum_{j < P_i} y^2_{ij} = 0 \quad ; \quad \sum_{j \geq P_i} y^2_{ij} = 1
\]

\( \forall \ j := (1, P) : \)

\[
\sum_{i=1}^{N} y^1_{ij} \leq K_j \quad ; \quad \sum_{i=1}^{N} y^2_{ij} \leq K_j
\]

Note that we would have had to add a new set of variables \( z^1 \) and \( z^2 \) to the above formulation if we wanted to include type 2 delays in \( Q(x, K^1) \) and \( Q(x, K^2) \).

Since the cost coefficients \( C_{ai} x_{ij} \), as well as the form of some of the constraints (through the quantities \( P_i x \)), depend on the solution \( x \), the resulting mathematical
program is a difficult non-linear integer program. This approach is however more successful in the case of a single time period as we demonstrate later. Note that we assumed that we could formulate the second stage problem all at once. This requires the knowledge of $K_1, K_2, \ldots, K_T$ at time $t_1$. In reality it is likely that when $K_1$ is known there still remains some uncertainty concerning the other capacities. If we make that simplifying assumption, however, we will see that we can formulate an integer program that is equivalent to the above non-linear program. We have included the non-linear program here because it is the simplest way to illustrate the basic step of the inductive argument that will allow us to indicate how we can build a non-linear program if the assumption of total disclosure of capacities at time $t_1$ is not valid (i.e. for the most general case of the problem).

The reason we want to include here a sketch of this recursive argument even though we do not actually construct the corresponding mathematical program is that it allows us to illustrate a major characteristic of the probabilistic version: the computation of the expected costs associated with a given ground holding policy, which is an essential element of any optimization method, depend not only on a capacity probability mass function but also on the assumptions we make concerning updates of the delay policy. The most complicated case corresponds to the assumption that optimal recourse action is considered after each individual time period. This is the case we will therefore describe; we will refer to this case as the full-recourse case. Other assumptions can be for example that optimal recourse action is considered only at time $t_1$ or that no optimal recourse action is considered at all. It is important to realize however that for any case other than the full recourse case we need to make some further assumption about the landing strategy utilized at airport $Z$, such as a fixed landing priority rule, to be able to compute expected
delay costs. In this sense recourse action has to be taken during each time period but this action is only optimal in the full recourse case.

The recursive argument that illustrates the full recourse case is as follows:

-Initial step: we can build a non-linear program that solves a problem where there is uncertainty only on the final capacity $K_P$ and where $K_1,K_2,...,K_{P-1}$ are known and fixed. This can be done by combining the integer programs that correspond to deterministic $K_1,K_2,...,K_P$ for each value of $K_P$ that $K_P$ can take on according to its probabilistic description in a way similar to the way subproblems $Q(x,K^1)$ and $Q(x,K^2)$ were combined to yield a non-linear program.

-Inductive step: for a given $Q$ we assume that a non-linear formulation exists for the case where $K_{P-Q+1},...,K_P$ are probabilistic and $K_1,K_2,...,K_{P-Q}$ are known and fixed (we call these order-$Q$ problems since they involve uncertainty on $Q$ capacities). We need to show that there exists a non-linear program for the case where $K_{P-Q},...,K_P$ are probabilistic and $K_1,K_2,...,K_{P-Q-1}$ are known and fixed (order-$(Q+1)$ problem). This is again done in the same manner by integrating the separate order-$Q$ programs (that we know we can build according to our inductive assumption) that correspond to all different values of $K_{P-Q}$ according to its marginal probability mass function (pmf) along with the associated conditional pmf's for $K_{P-Q+1},...,K_P$ into a global order-$(Q+1)$ non-linear program. Again the way we combined subproblems $Q(x,K^1)$ and $Q(x,K^2)$ indicates how this can be done.

This shows that non-linear programs exist for all orders of complexity. The exact solution corresponding to a given $P_{K_1,K_2,...,K_P}$ is obtained by building up the programs recursively. We illustrate the first two steps: we start with the order-1 formulations for all values of $K_1,K_2,...,K_{P-1}$ and associated $P_{K_1,K_1,...,K_{P-1}}$. From these subproblems we
build all the order-2 formulations for all values $K_1, K_2, ..., K_{P-2}$ and associated conditional pmf $P_{K_P, K_{P-1} | K_1, ..., K_{P-2}}$. It is evident that this is not a very practical approach; the size of the non-linear programs grows exponentially with each added time period. But we should note that we no longer assume that the capacities $K_1, K_2, ..., K_P$ become all known at once. We only assume that at the beginning of each time period $T_i$, when $K_1, K_2, ..., K_i$ are known we utilize the conditional pmf $P_{K_i+1, ..., K_P | K_1, ..., K_i}$ extracted from the forecast $P_{K_1, K_2, ..., K_P}$ to decide which flights to allow to land during $T_i$. This approach therefore does yield a least cost strategy given that the pmf $P_{K_1, K_2, ..., K_P}$ that we started with represents accurately the evolution of capacities.

Let us now see how we can develop the formulation for a single time period. This provides another opportunity to illustrate how subproblems can be combined to yield a global program.

C.1.2 Stochastic Programming - Single Period Case:

We consider a single time period with a capacity $K$ that is a random variable taking on two values $K_1$ and $K_2$ with respective probabilities $p_1$ and $p_2$. We are looking for an optimal ground holding policy $\mathbf{x} = (x_1, ..., x_N)$ such that $x_i = 1$ if flight $F_i$ is kept in the ground (for one time period) and $x_i = 0$ if it is allowed to take off on time. We note by $C_{a_i}$ and $C_{g_i}$ the cost of delaying flight $F_i$ one time period in the air and one time period on the ground respectively. Let us consider the second stage problem first:

Assume a ground holding policy $\mathbf{x}$ has been chosen and a capacity $K$ is realized. We define $\mathbf{y} = (y_1, ..., y_N)$ such that $y_i = 1$ if flight $F_i$ is delayed in the air after getting to
airport $Z$ on time and $y_i = 0$ if it is allowed to land on time. In this case the second-stage assignment problem $Q(x,K)$ is:

$$\min \sum_{i=1}^{n} C_{ai} y_i$$

subject to:

$$\sum_{i}(1 - x_i) + \sum_{i} y_i \leq K$$

$$y_i \leq 1 - x_i \quad \forall \ i (=1,...,N)$$

The first constraint represents the capacity constraint. The following set of $N$ constraints insures that only flights that took off on time ($x_i = 0$) are eligible for airborne delays (since we cannot have $y_i = 1$ if $x_i = 1$). This insures that the cost function contains only the costs of delayed flights.

The global problem is to find $(x, y^1, y^2)$, where $y^1 = (y_1^1, y_2^1, ..., y_N^1)$ corresponds to air-delays for the capacity case $K_1$ and $y^2 = (y_1^2, y_2^2, ..., y_N^2)$ for capacity case $K_2$, to solve:

$$\min \left\{ \sum_{i=1}^{N} C_{gi} x_i + p_1 \left[ \sum_{i=1}^{N} C_{ai} y_i^1 \right] + p_2 \left[ \sum_{i=1}^{N} C_{ai} y_i^2 \right] \right\}$$

subject to:

$$\sum_{i}(1 - x_i) + \sum_{i} y_i^1 \leq K_1$$

$$\sum_{i}(1 - x_i) + \sum_{i} y_i^2 \leq K_2$$

$$y_i^1 \leq 1 - x_i; \quad y_i^2 \leq 1 - x_i \quad \forall \ i (=1,...,N)$$

The constraint-matrix is of the form:
\[ A_N = \begin{pmatrix}
I_N & I_N & 0_N \\
I_N & 0_N & I_N \\
1 \ldots 1 & -1 \ldots -1 & 0 \ldots 0 \\
1 \ldots 1 & 0 \ldots 0 & -1 \ldots -1
\end{pmatrix} \]

where \( I_N \) is the identity matrix of order \( N \) and \( 0_N \) is the zero matrix of order \( N \).

This matrix is not totally unimodular since one of its codeterminants is \( \begin{vmatrix} 1 & 1 \\ 1 & -1 \end{vmatrix} = -2 \). The Dynamic Programming approach proposed by Andreatta-Jacur (described in the introduction and Chapter 4) for the case of a single time period and general cost functions is faster than the above integer program for solving this problem. We need to assume that (ground and airborne) delay costs are equal for all flights to formulate this integer program as a network flow; but in this case the problem has also a simple analytical solution as we will demonstrate later.

When we assume that all flights have the same linear cost function the network formulation developed for the deterministic case in Chapter 2 can be extended to the probabilistic case to yield an integer program (IP). This IP does not however correspond to a minimum cost flow in a network except in the single time period case. It is instructive to investigate these simpler formulations in detail as they cast some light on the structure of the general problem.

C.1.3 Extension of Network Formulation to the Probabilistic Case:

In this section we investigate the applicability of the network formulation developed for the deterministic case to the probabilistic case. We will see that in general we cannot formulate the probabilistic problem as a minimum cost flow in a network as was the case for the deterministic problem. The network approach is however useful in
motivating an integer programming (IP) formulation that yields an exact solution to the most general formulation when we assume that the uncertainty in capacities is resolved at time $t$. Interestingly enough it seems that it would have been difficult to come up with the IP formulation without the help of the network interpretation. We will use the same approach of trying to extend the network formulation to identify another IP that solves a version of the multi-airport case when we discuss directions for future research in Chapter 6. We will start our discussion from the most simple cases and relax the simplifying assumptions as we go by adding more logical constraints.

The network formulation breaks down in the probabilistic case for two main reasons. One has to do with the fact that we now have to add to the network arcs that correspond to airborne delays and that we cannot avoid some undesirable mixing of flow in these arcs unless we add some additional constraints that destroy the total-unimodularity of the constraint matrix. The other obstacle appears only in the multiperiod case and concerns the functional form of airborne delay costs. In order to illustrate these obstacles it is useful to start with the single time period case which isolates clearly the first difficulty.

**single period case:**

We consider the case where a total of $D$ flights with identical delay costs (denoted $C_g$ for the ground cost, and $C_a = k \cdot C_g$ for the airborne cost) are scheduled to land at airport $Z$ during the single time period. The capacity of that single time period is a
random variable $K$ that takes on 2 values $K_1$ and $K_2$ with probabilities $p_1$ and $p_2$. Since all flights are assumed to be identical in terms of costs the problem reduces to deciding how many of the $D$ flights should be allowed to take off on time. The following network illustrates the situation; the flow through the first arc (with cost $[0]$) out of the node labeled "D flights" associated to the minimum cost flow in this network corresponds to this optimal number. Here we use only two values for $K$ to illustrate better an interesting phenomenon of discontinuity of the solutions linked to the number of capacity cases we use in our probabilistic description of the capacity forecast. We will come back to this phenomenon in more detail in the numerical examples of Chapter 5.

![Network Diagram]

**Figure B-2**

A simple analytical solution exists for the above formulation. When all flights have identical costs the solution to the second-stage problem is trivial: it consists of

---

3 The generalization to an arbitrary number of values is trivial. We use two values to keep the presentation simple.
filling as much of the available capacity with any set of flights. The phenomenon we
referred to above is that the optimal number of flights that take off on time according to
the optimal solution is either 0, K1, K2, or D but cannot be a number inbetween. The
reason for this is that any solution that sends a number of flights on time strictly between
any of these values can be improved on by sending more flights up to the next highest
capacity value. It is easy to see that this policy yields a lower expected cost than the
original one: it suffices to consider each capacity case that can be realized separately and
note that it yields a lower cost for each one of these capacity cases. Therefore we have
only four values to check for the optimal solution: 0, K1, K2, and D. The optimal
solution corresponds to the first value of capacity, Kopt, among these four values such
that the expected (air) cost of sending the (Kopt+1)th flight on time is higher than the
ground cost Cg. The expected cost for the (Kopt+1)th flight is given by:

\[ E(\text{cost}(K_{opt}+1)) = p_1 \cdot E(\text{cost}(K_{opt}+1|K_1)) + p_2 \cdot E(\text{cost}(K_{opt}+1|K_2)), \]

Where the quantities E(\text{cost}(K_{opt}+1|K_i)) are either 0 or Ca depending on whether K_i is greater than
Kopt or not.

It is interesting to see where the network formulation breaks down if the delay
costs are no longer assumed to be equal. We therefore consider the same formulation
except that delay costs have to be indexed using the flight index, the costs C_{gi} and C_{ai}

\[ \text{\textsuperscript{4}} \] This assumes that D ≥ K2. Otherwise we need only check up to D.

\[ \text{\textsuperscript{5}} \] Since all landing priority rules are equivalent in the case of identical costs we can assume,
with no loss of generality, that the \textit{i}th flight has priority over the \textit{(i+1)}th flight. So when we speak
about the cost of the \textit{i}th flight sent on time we assume that it can only be landed at the \textit{i}th position.
referring to the ground and airborne delay costs for flight F₁. The difficulty is that one arc 

\[ p_{1Ca} \] 

of the type \( p_{1Ca} \) is no longer sufficient; we need one such arc for each flight 

with associated cost \( p_{1Ca_j} \). The following figure illustrates what would happen if we 

try to generalize the network approach to the case of two different flights F₁ and F₂.

![Diagram](image)

**Figure B.3**

As we examine the above figure it is clear that there are feasible flows that do not 

correspond to any ground holding policy. The flow indicated by thick lines in the 

following figure is an example of such a flow; this flow implies that flight F₁ is held on 

the ground and therefore will experience no air delay whereas it is the "aircost" arc
corresponding to flight $F_1$ that is utilized. ("aircost" arc refers to arcs with unit-flow costs $C_{ai}$.)

![Diagram](image)

**Figure B-4**

In order to correct this and make sure that feasible flows correspond to ground holding policies we have to complement the integer program corresponding to this capacitated network with additional constraints resulting in a constraint matrix that is no longer totally-unimodular. Such a constraint would be one that insures that the flow in the arc with cost $[p_1Ca_1]$ has to be lower or equal than that of the arc with cost $[0]$ out of the node "flight $F_1$". In conclusion we are left with another integer program similar to the one we identified when discussing stochastic programming methods.
The same problem occurs in the multiperiod case as we now indicate.

**Multi-period case:**

At the end of this section we will build an integer program that yields an exact solution for the most general case of the probabilistic formulation corresponding to arbitrary delay costs under the assumption that the uncertainty in the capacities $K_1, K_2, \ldots, K_P$ is entirely resolved at once at time $t_1$. We will start however with a simpler case where we assume that the cost functions are of the form $C_{ij}(x_i, y_i) = C_{ij}(x_i + y_i)$.

We assume that the capacity vector $K = (K_1, K_2, \ldots, K_P)$ can take on $k$ different values $(K^1, K^2, \ldots, K^k)$, where $K^i = (K_{1i}, K_{2i}, \ldots, K_{Pi})$ happens with probability $p_i$. For each realization $K^i$ of this probability vector, and under the assumption of complete disclosure of capacities at $t_1$, is associated a network illustrated in the following figure:
The number of arcs in this network is very large. Each node "flight i" generates one "aircost arc" (arcs with associated costs [Ca...]) for each time period after its scheduled landing period. It is possible in this formulation to allow non-linear costs by having, for a given flight, different costs for different time periods; but these costs can depend only on the index of the time period and therefore only on the cumulative delay $x_i + y_i$. As was the case for a single time period some feasible flows through this network do not correspond to ground holding policies. We have to add constraints that insure that when a flight is assigned to a time period none of its "aircost" arcs for the previous time periods is used. We also need to insure that if a flight has landed during a given time
period $T_j$ no "aircost" arc associated with that flight for any time period after $T_j$ is used.
Finally we need to insure that each aircost arc is used only by the corresponding flight.
This is accomplished by setting an upper bound $u=1$ on the aircost arcs (if we do not set $u=1$ the "aircost" arc with the lowest cost will be used by several flights in the minimum cost flow). We need to define additional notations to describe these constraints more precisely.

Let us denote by $x_{ij}^1$ the assignment variables for this network-1; we have $x_{ij}^1 = 1$ if flight $F_i$ is assigned to time period $T_j$ (i.e. scheduled to arrive at airport $Z$ during time period $T_j$), $x_{ij}^1 = 0$ otherwise ($x_{ij}^1$ represents the flow through the assignment arc from
the node "flight i" to the node corresponding to the jth time period; no such arc exists if
the index of the scheduled landing period for $F_i$ is larger than j). We also note $y_{ij}^1$ the
flow through the aircost arc corresponding to flight $F_i$ and time period $T_j$ ($y_{ij}^1 = 1$ if flight $F_i$ is held in the air during the jth time period, $y_{ij}^1 = 0$ otherwise).

Let $\mathbf{x}^1$ be the vector $\mathbf{x}^1 = (x_{11}^1, x_{12}^1, ..., x_{1P}^1, x_{21}^1, ..., x_{2P}^1, ..., x_{NP}^1)$. Let $\mathbf{y}^1$ be the vector $\mathbf{y}^1 = (y_{11}^1, y_{12}^1, ..., y_{1P}^1, y_{21}^1, ..., y_{2P}^1, ..., y_{NP}^1)$. Note that for each flight $F_i$ the first usable index $j$ in $x_{ij}^1$ and $y_{ij}^1$ corresponds to the scheduled landing period for that flight, which we have already denoted as $P_i$. We denote by $\mathbf{X}^1$ the complete decision vector $\mathbf{X}^1 = (\mathbf{x}^1, \mathbf{y}^1)$.

The first type of constraint is of the form: $\sum_{j=P_i}^{h} x_{ij}^1 \geq y_{ih}^1$, with one such constraint
for all $P_i \leq h \leq P$. This insures that no aircost arc with flow $y_{ij}^1$ is used before the first $x_{ij}^1$ that is non-zero. The reason we have to use a summation in the left hand side is that we do not want to force zero flow in the aircost arcs after that non-zero $x_{ij}^1$. 

219
The second type of additional constraint is of the form: \( y_{ij}^l \leq y_{i(j-1)}^l + x_{ij}^l \) for all \((P_i + 1) \leq j \leq P\). This insures that once one of the \(y_{ij}^l\)'s has gone to zero (i.e. the flight has landed) no other \(y_{ij}^l\) is used (set to 1) again. The inclusion of \(x_{ij}^l\) makes this constraint applicable only after flight Fi arrives at airport Z to allow for the first non-zero aircost arc if necessary.

To complete the description of the integer program associated with network l we denote by \(C_{g_{ij}}\) the ground cost of assigning flight Fi to (arrive at airport Z during) time period Tj and \(C_{a_{is}}\) the cost of delaying that flight in the air during Ts. It is now clear that in this formulation we are restricting the shape of airborne cost functions. The costs \(C_{a_{is}}\) cannot depend on the ground delay previously imposed; this corresponds to the case we identified when setting the mathematical formulation in section 3.2, namely the case where airborne delay costs depend only on the cumulative delay time \((C_{a_i(x_i,y_i)} = C_{a_i(x_i+y_i)})\). Later we will generalize the formulation to arbitrary cost functions. With these notations the lth integer program corresponding to the minimum cost flow through a network and the additional constraints identified above is:

\[
\begin{align*}
\min \{ & \sum x_{ij}^l C_{g_{ij}} + \sum y_{ij}^l C_{a_{ij}} \} \\
\text{subject to:} & \quad \Delta X^l \leq \Omega^l \\
& \forall \ i \ (=1,N): \\
& \forall h \geq P_i : \sum_{j=P_i}^{h} x_{ij}^l \geq y_{ih}^l \\
& \forall j \geq P_i : \quad y_{ij}^l \leq y_{i(j-1)}^l + x_{ij}^l \\
\end{align*}
\]
where matrices $\Delta$ and $Q^l$ correspond to the usual flow conservation constraints and lower and upper bound constraints for network-$l$. The matrix $\Delta$ (which is totally-unimodular) depends only on the network’s topology whereas the matrix $Q^l$ depends on the bounds for the arc flows; particularly it depends on $(K^l_1, K^l_2, \ldots, K^l_p)$, and therefore it has to be indexed by $l$. Let us note this $l$th IP:

$$\min \{ X^l \ C \} \text{ subject to:} B^l \ X^l \leq Q^l$$

where the matrices $B^l$ and $Q^l$ are obtained from the matrices $\Delta^l$ and $Q^l$ by adding the two last sets of constraints identified above.

Note that, if we use this IP in isolation, the solution will not use any aircost arcs and would set $x = 0$. We have to complement the $k$ network formulations associated with the $k$ capacity cases with additional constraints to insure that they relate to the same ground holding policy. This is done by adding the constraints $x^{\ast}_{ij} = x^u_{ij}$ for all $i,j,u,$ and $v$.

The global IP that minimizes the total expected cost is therefore:

$$\min \left\{ \sum_{l=1}^{k} \left[ X^l \ C \cdot p_l \right] \right\}$$

subject to:

$\forall l = \{1, \ldots, k\}$: $B^l \ X^l \leq Q^l$

$\forall (u,v) \in \{1, \ldots, k\}$ and $\forall (i,j) \in \{1, \ldots, N\}$: $x^{\ast}_{ij} = x^u_{ij}$
The interpretation of this last set of constraints is interesting: Their shadow costs correspond to the cost of uncertainty since without them the solution of the above IP will consists of ground holds exclusively. Without these constraints the problem is equivalent to k separate minimum cost flows since, as we noted before, the aircost arcs are never used in the solution of any of the k subproblems taken in isolation. Adding these constraints is equivalent to conceding that we do not know which of the k capacity cases will occur, the shadow cost on these constraint being the price we have to pay for not being able to adjust the ground holds to each situation. This brings us to an important point alluded to previously: the solution of this IP is optimal only if we assume that at time t₁, when the capacity K₁ becomes known, all the remaining capacities K₂,...,Kₚ also become known so that the air-holds obtained from the subproblems do indeed correspond to the air delays that will be imposed in reality. If this is not true only the nonlinear IP obtained using the stochastic programming approach yields the optimal solution to our problem since the recursive construction is equivalent to resolving the uncertainty gradually which is the case in reality.

We can now describe what is involved in generalizing this IP to arbitrary cost functions. If we assume that the airborne delay cost functions are general functions of the form \( C_{ai}(x_i, y_i) \) the above formulation fails as we pointed out earlier. To allow the cost to depend also on which assignment arc is used we have to add aircost arcs to each time period with a further index indicating which assignment arc is used and additional logical constraints to insure that only one set of aircost arcs is used. This leads us to the final and most general version of the IP as follows.

We will denote by \( C_{aijs} \) the cost of delaying flight \( F_i \) in the air during time period \( T_j \) given it was previously delayed on the ground so that it showed up at airport Z during
time period $T_s$. It is easy to see that this corresponds to the most general functional form for air delay costs: $C_a(x_i, y_i)$. (These costs are only defined for $s \geq j$, and $j \geq P_i$.) The cost $C_{a_{ijs}}$ is the unit flow cost associated with an aircost arc corresponding to time period $T_j$ with an associated flow that we note $y_{ijs}^1$ (we are dealing with the $l$th network, thereby the index $l$). We have to make sure that the set of aircost arcs indexed by $s$ can be used only if the assignment arc with flow $x_{is}^1$ is used; this is accomplished through constraints of the form $y_{ijs}^1 \leq x_{is}^1$ for all $i, j, s$. The first type of logical constraint from the original IP is now redundant; the second type of constraint becomes $y_{ijs}^1 \leq y_{i(j-1)s}^1$ for $j = s+1$ to $P+1$. To describe the whole IP the decision vector has to be extended to account for the additional aircost arcs ($s$-index); we will still denote it by $X^l$ but now we have: $X^l=(x^l, y_{l1}^1, \ldots, y_{ls}^1, \ldots, y_{lp}^1)$ where each $y_{ls}^1=(y_{ij}^1, y_{ij(j+1)}^1, \ldots, y_{ij(P+1)}^1)$. The final IP is:

$$\min \left\{ \sum_{i=1}^{N} x_{ij}^l C_{g_{ij}} + \sum_{l,j,s}^{l} y_{ijs}^1 C_{a_{ijs}} \right\}$$

subject to:

\[ A' X^l \leq Q'^l \]

\[ \forall \ i \ (=1, N), \ \forall \ l \ (=1, k), \ \forall \ j \ (=P_i, P+1), \]

\[ \forall \ s \geq j: \ y_{ijs}^l \leq x_{is}^l \]

and \[ \forall \ s \geq j: \ y_{ijs}^l \leq y_{i(j-1)s}^l \]

where the matrices $A'$ and $Q'^l$ are obtained from the matrices $A$ and $Q^l$ by adding the flow conservation constraints for the new aircost arcs.

Let us denote this IP by:

$$\min \{ X^l C' \} \text{ subject to: } B X^l \leq Q'^l$$

223
The global IP that minimizes the total expected cost for all k networks corresponding to the k capacity cases is:

\[
\min \left\{ \sum_{l=1}^{k} \left( X^l C^l \cdot p_l \right) \right\}
\]

subject to:

\[
\forall l \in \{1,\ldots,k\}: \quad B^l X^l \leq Q^l
\]

\[
\forall (u,v) \in \{1,\ldots,k\} \text{ and } \forall (i,j) \in \{1,\ldots,N\}: \quad x^u_{ij} = x^v_{ij}
\]

Note that this IP is significantly larger than the one where we assumed \( C_{ai}(x_i, y_i) = C_{ai}(x_i + y_i) \). However the reason we might want to allow the airborne costs to depend separately on \( y_i \) is to model such practical considerations as fuel constraints, e.g., by sharply increasing the costs for values of \( y_i \) that correspond to fuel usage that is close to fuel reserves. This is the only reason we can think of for allowing such flexibility in cost functions; there is otherwise no argument for having costs different from \( C_{ai}(x_i + y_i) \). The price we pay for such total flexibility is however high since the size of the IP is greatly increased. There is fortunately a way to model this particular fuel constraint consideration in the original (much smaller) IP by adding constraints that directly specify that the sum of the flows of air cost arcs used by any flight \( F_i \) has to be less than an upper bound that corresponds to the limit in fuel usage:

\[
\sum_{j=P_i}^{P+1} y_{ij} \leq FUEL_i \quad \text{where } FUEL_i \text{ is the number of time periods flight } F_i \text{ can remain airborne beyond its regular flight time without running}
\]
out of fuel. The resulting IP is much smaller that the final, most general IP described above and accomplishes the same thing.

Finally we note that each of the IP's we have developed is such that large subsets of the constraint set correspond to flows through networks. Therefore there exists fast solution methods for large portions of these IP's making decomposition methods particularly appropriate for consideration.

We now summarize the main findings of this section:

We started by presenting the stochastic programming approach in part because it illustrates vividly a major characteristic of the probabilistic formulation: in order to evaluate a given ground holding policy at a time $t$ we have to compute the expected airborne delay costs for that ground holding policy and in order to do so it is necessary to make some assumption about the timing of further instances of optimization (or recourse actions as we called them in the context of stochastic programming). Such an assumption could be, for example, that no further optimization is to be undertaken; even in this case we have to make some assumption about the landing strategy that will be utilized by air traffic controllers at airport Z and, in this case, we must assume a prespecified landing strategy that cannot depend on further events (such as first-come, first-served according to the order in which aircraft arrive in the terminal area of airport Z). In the case of the two-stage stochastic program the assumption was that at time $t_1$ we would proceed with a recourse optimization based on the value of $K_1$. It is important to realize that the only pmf we can use at time $t$ of optimization is the conditional pmf for $K_2,...,K_P$ given $K_1$ corresponding to $P_K(t)$. The "accuracy" of the expected costs computed, and thereby the quality of the solution implemented, therefore depends on the precision of the forecast system that determined $P_K(t)$. This is the issue that we turn to now; we will see that we
can use this observation to define a hypothetical perfect forecast system as the one that can allow us to determine the "true" expected cost of any given solution at time $t$. We put the words true and accuracy in quotes because the real costs associated with an optimization instance at time $t$ are deterministic by definition (they correspond to any particular realization of $K_1, K_2, \ldots, K_P$ after $t$). It is the purpose of the following sections to define precisely these concepts as they relate to the problem at hand and in doing so we hope to motivate a whole range of new concepts and terms that we feel are a major contribution of the probabilistic formulation.

**C.2 Planning Tool**

We will examine the application of the probabilistic formulation to a longer term planning problem and show how we can utilize some of the solution techniques investigated in this context. The long-term problem is that of determining an "ideal" schedule for airport $Z$ given a history of capacity variations. The purpose of this section is to show the applicability of some of the solution techniques examined so far to a new problem but also to show that this long-term problem bears the same relationship to the ground holding policy problem as this latter problem bears to the second stage recourse problem examined in the previous sections.

The question we want to tackle is that of establishing a schedule for the arrival demand at airport $Z$ given a history of landing capacities compiled through the years. We are shifting the scope of the analysis from the short-term adjustment of a given schedule to determining a schedule that will require a minimum adjustment to circumstances later. We are therefore no longer interested in ground holds versus airborne delays but we now consider delay costs as they relate to not being able to satisfy a given demand. We also
need to have a measure of the relative desirability of supply as a function of time of the
day to trade-off these benefits against the delay costs (otherwise the obvious solution is to
assign flights throughout the entire 24-hours even at times of low demand). As is the
case with the short term ground holding problem the solution depends on the assumptions
we make concerning how adjustments are made in the short term (in the case of the
ground holding policy these adjustments were referred to as the second-stage-recourse
problem; in the present case, these adjustments correspond to the GHPP ). To simplify
this presentation, and since this long term planning problem is not the central issue of the
thesis, we will confine ourselves to the case where we assume that no recourse action is
to be taken (i.e. we assume that no ground holding policy will be considered in the
future). It is however important to realize that we could refine the formulation to be
developed here to account for the case where we know that adjustments will also be made
through the implementation of ground holds.

Assuming that no ground holds will be considered in the future implies that we
need to consider only airborne delay costs in our formulation. We will extend the
network flow formulation to the probabilistic case. We consider that we are trying to
establish the "ideal" schedule at airport Z for a given day of the week and that we have
compiled a histogram for the landing capacities at airport Z and translated this histogram
into a pmf $P_K$ for the capacities of individual time periods. Note that this forecast system
is exactly the one we called the "no-information forecast" in Section 3.1 when we
discussed the issue of capacity forecast. We also assume that, based on past demand for
air transportation at airport Z, or some demand model for air transportation demand for
the region that is serviced by airport Z, we are able to determine an economic measure of
the utility of landing a flight during each time period for the day in question. Because of
the macroscopic scope of the problem it is not appropriate to differentiate among flights in
terms of utility or costs; for the same reason and in the light of our discussion of cost
functions in Section 3.4, it is sufficient to consider linear (air delay) cost functions in
which case we need only specify an average delay cost per period that we call Ca. We
will denote by \( U_i \) the utility of an arrival flight during time period \( T_i \) for the day in
question. The \( l^{th} \) network corresponding to the \( l^{th} \) capacity case of \( PK \) is shown in Figure
B-6. The purpose is to minimize total cost minus total utility. We will therefore label
some arcs with negative costs corresponding to the utilities. There are two options for
labelling the aircost arcs: one is to consider that the utilities used relate to the utilities of
scheduling flights to time periods, in which case it is enough to assign a cost \( Ca \) per unit
flow for these arcs; the other is to consider that we are given the utilities of actually
landing flights during the time periods, in which case we have to adjust the cost \( Ca \) to
reflect the change of landing period when a flight is held in the air before landing. This
latter alternative is the one illustrated in Figure B-6.

---

6 Demand patterns can vary form day to day.
As was the case for the ground holding policy problem the solution of this \( l \)th problem taken in isolation will send exactly \( K_l \) flights to time period \( T_i \). We need to consider the whole set of \( k \) networks and integrate each integer program into a global IP with additional constraints that specify that the assignment flows are equal. If we denote the \( l \)th IP: \( \min \{ (X_l, Y_l), C \} \) subject to: \( A(X_l, Y_l) \leq B_l \), where \( X_l = (x_1, x_2, ..., x_p) \) is the assignment vector corresponding to the arcs with costs \( [U_l] \) (\( x_i \) is the number of flights assigned to time period \( T_i \)) and \( Y_l \) represents the flow in the remaining arcs constituting network-1, the global IP is of the form:
\[
\min \sum_{l=1}^{k} p_l \cdot (x_l^l, y_l^l) \cdot c_l
\]

subject to: \( \forall l = \{1, ..., k\} \):

\[ A \cdot (x_l^l, y_l^l) \leq b_l \]

\[ \forall l, h = \{1, ..., k\} \quad x_l^l = x_h^l \]

where \( k \) is the number of probability cases distinguished by the pmf \( P_k \) and \( p_i \) is the probability associated with the \( i \)th case.

It is not difficult to see how we can now extend this formulation to account for the case where we know that ground holds will be considered to adjust the "ideal" schedule to a daily capacity forecast. For example we could assume that each morning we utilize a capacity forecast to determine ground holds. In this case we use our yearly forecast to determine the conditional pmf's to be used in this recourse problem (which again corresponds to the original problem considered in the previous sections) and a recursive argument identical to the one used when we considered the stochastic programming formulation would show that we end up with another nonlinear IP. Now we can make the further assumption that the capacity forecast is deterministic for each day we consider to reduce this problem to an integer program; the purpose of this discussion however is not to go so deeply into that formulation but to show that the longer term problem is strongly related to the short term problem we deal with in this thesis in the same way we have shown that the short term problem was related to what we called the second stage recourse problem. A resulting conclusion is that the probabilistic set up is as relevant to
the longer term problem of scheduling as it is to the ground holding policy problem and that a complete solution must integrate both aspects.

The second important conclusion is that the short term ground holding policy is in fact a problem that will persist even when an "ideal" schedule is implemented. This is why, in the introduction, we have avoided characterizing the ground holding policy problem as one that we have to deal with in the present until more long term solutions are available. The inherent uncertainty in airport capacity implies that we will always need to consider a trade-off between costly physical capacity and delays if we are concerned with providing an overall solution and this solution will always attempt to strike a balance between these two aspects as long as there is a cost associated with additional physical capacity.
APPENDIX D: Multi-Airport Formulations

The natural extension of the single airport problem investigated in this thesis is to consider "full network" formulations that include several airports. The main purpose of this appendix is to indicate how one can go about building Integer Programming multi-airport deterministic formulations.

D.1 An Extension with Two Airports

We consider a situation in which some congestion is expected at two airports, airport \( Z \) and airport \( Z' \), and several aircraft scheduled to land at airport \( Z \) are also used for flights from airport \( Z \) to airport \( Z' \). In this case, imposing a ground delay on one of this aircraft at its first airport of origin will not only affect its arrival time at airport \( Z \) but could also affect its arrival time at airport \( Z' \). These "downstream" effects can be directly modeled in an Integer Program (IP) that extends the single airport IP formulation of Section 2.1.1.
Figure C-1

Figure C-1 refers to a situation that we describe as follows:

- We consider operations at airport Z during P time periods, $T_1, T_2, ..., T_P$, with (known) deterministic capacities $K_1, K_2, ..., K_P$. 

233
• We also consider operations at airport \( Z' \) during \( P' \) time periods, \( T_{1}^{*}, T_{2}^{*}, \ldots, T_{P'}^{*} \), with (known) deterministic capacities \( K_{1}', K_{2}', \ldots, K_{P'}' \).

• \( N+M \) flights, \( F_{1}, \ldots, F_{N}, F_{N+1}, \ldots, F_{N+M} \), are scheduled to land at airport \( Z \) during these time periods. Furthermore, the \( M \) aircraft used to operate flights \( F_{N+1}, \ldots, F_{N+M} \) are to be used for flights \( H_{1}, \ldots, H_{M} \) which are scheduled to land at airport \( Z \) during time periods \( T_{1}^{*}, T_{2}^{*}, \ldots, T_{P'}^{*} \).

• \( N' \) flights, \( G_{1}, \ldots, G_{N'} \), that could originate from airports other than airport \( Z \), are also scheduled to land at airport \( Z' \) during time periods \( T_{1}^{*}, T_{2}^{*}, \ldots, T_{P'}^{*} \).

• For each flight \( F_{i} \) \((1 \leq i \leq N+M)\) we define:
  
  - \( P_{i} \), the index of the time period during which flight \( F_{i} \) is scheduled to land at airport \( Z \). \((1 \leq P_{i} \leq P)\)

  - \( C_{ij} \), the (ground delay) cost of reassigning flight \( F_{i} \) to land during time period \( T_{j} \) (note that \( C_{ip_{i}} = 0 \)).

  - \( x_{ij} \), the assignment variable for flight \( F_{i} \). We set \( x_{ij} = 1 \) if flight \( F_{i} \) is assigned to land during time period \( T_{j} \); \( x_{ij} = 0 \) otherwise. (These assignment variables, as well as the costs \( C_{ij} \), are only defined for \( P_{i} \leq j \leq P+1 \).)

• For each flight \( G_{i} \) \((1 \leq i \leq N')\) we define:

  - \( P'_{i} \), the index of the time period during which flight \( G_{i} \) is scheduled to land at airport \( Z' \). \((1 \leq P'_{i} \leq P')\)
- $C'_{ij}$, the cost of reassigning flight $G_i$ to land during time period $T_j$ ($C'_{iP'} = 0$).

- $x'_{ij}$, the assignment variable for flight $G_i$. We set $x'_{ij} = 1$ if flight $G_i$ is assigned to land during time period $T'_j$; $x'_{ij} = 0$ otherwise. (These assignment variables, as well as the costs $C'_{ij}$, are only defined for $P' \leq j \leq P'+1$.)

- For each flight $H_i$ ($1 \leq i \leq M$) we define:
  
  - $Q_i$, the index of the time period during which flight $H_i$ is scheduled to land at airport $Z'$. ($1 \leq Q_i \leq P'$)
  
  - $D_{ij}$, the cost of reassigning flight $H_i$ to land during time period $T'_j$ ($D_{ijQ_i} = 0$).

  - $y_{ij}$, the assignment variable for flight $H_i$. We set $y_{ij} = 1$ if flight $H_i$ is assigned to land during time period $T'_j$; $y_{ij} = 0$ otherwise. (These assignment variables, as well as the costs $D_{ij}$, are only defined for $Q_i \leq j \leq P'+1$.)

- Furthermore, since each flight $H_i$ is operated by the same aircraft as flight $F_{N+i}$, we have to make sure that flight $H_i$ is assigned to a time period that allows enough turnaround time for the aircraft. For this purpose, we denote by $L_{ij}$ the index of the latest time period during which flight $F_{N+i}$ could have landed at airport $Z$ and still allowed, given a minimum turnaround time for the aircraft, flight $H_i$ to land at airport $Z'$ during time period $T'_j$. 

235
With this notation the solution to the following zero-one IP corresponds to a minimum cost assignment of each flight to a time period:

\[
\text{Min} \left[ \sum_{i=1}^{N+M} \sum_{j=P_i}^{P+1} C_{ij} \ x_{ij} + \sum_{i=1}^{M} \sum_{j=Q_i}^{P+1} D_{ij} \ y_{ij} + \sum_{i=1}^{N} \sum_{j=P'_i}^{P'+1} C'_{ij} \ x'_{ij} \right]
\]

Subject to:

(1) Assignment Constraints:

\[
\sum_{j=P_i}^{P+1} x_{ij} = 1 \quad \text{for all } i = 1, \ldots, N+M
\]

\[
\sum_{j=P'_i}^{P'+1} x'_{ij} = 1 \quad \text{for all } i = 1, \ldots, N'
\]

\[
\sum_{j=Q_i}^{P+1} y_{ij} = 1 \quad \text{for all } i = 1, \ldots, M
\]

(2) Capacity Constraints:

\[
\sum_{i=1}^{N+M} x_{ij} \leq K_j \quad \text{for all } j = 1, \ldots, P
\]

\[
\sum_{i=1}^{N} x'_{ij} + \sum_{i=1}^{M} y_{ij} \leq K'_j \quad \text{for all } j = 1, \ldots, P'
\]

236
(3) "Turnaround Time" Constraints:

\[ \sum_{k=P_i}^{L_{ij}} x_{ik} \geq y_{ij} \quad \text{for all } i = N+1, \ldots, N+M, \text{ and for all } j = Q_i, \ldots, P'+1 \]

(4) Integrality Constraints:

\[ x_{ij} = 0 \text{ or } 1 \quad \text{for all } i = 1, \ldots, N+M, \text{ and for all } j = P_1, \ldots, P+1 \]

\[ x'_{ij} = 0 \text{ or } 1 \quad \text{for all } i = 1, \ldots, N', \text{ and for all } j = P'_i, \ldots, P'+1 \]

\[ y_{ij} = 0 \text{ or } 1 \quad \text{for all } i = 1, \ldots, M, \text{ and for all } j = Q_i, \ldots, P'+1 \]

Each "turnaround" constraint of the type \[ \sum_{k=P_i}^{L_{ij}} x_{ik} \geq y_{ij} \] reflects the fact that flight \(H_i\) cannot take-off to land at airport \(Z'\) during time period \(T_j\) if flight \(F_{N+i}\) has not landed at airport \(Z\) before \(T_{L_{ij}}\).

The above zero-one IP is highly block-angular in structure. In fact the zero-one IP resulting from ignoring the "turnaround" constraints is equivalent to solving two independent GHPP's, one for each airport. This suggests that decomposition techniques are likely to be very useful since each subproblem corresponds to a minimum cost flow through a capacitated network (the constraints submatrix obtained by ignoring constraints (3) is totally unimodular). Furthermore, if we assume "regular" cost functions, we can
use the very fast algorithm introduced in chapter 2 in the decomposition techniques for an even faster overall solution\(^1\).

It is easy to see how one could extend this formulation to an arbitrary number of airports and aircraft with "turnaround time" constraints. In fact, one could also consider total (arrival and departure) capacities in the formulations by modifying the capacity constraints to include the assignment variables that reflect the departure time of each flight. (For example, if we define \(d_{ij}\) to be the index of the time period during which flight \(H_i\) will land at airport \(Z'\) if it took-off during time period \(T_j\) from airport \(Z\), we would modify the capacity constraints for airport \(Z\) to
\[
\sum_{i=1}^{N+M} x_{ij} + \sum_{i=1}^{M} y_{id_{ij}} \leq K_j,
\]
where \(K_j\) is now the total capacity for time period \(T_j\).) In this manner we can develop a "closed network" formulation that includes all origin and destination airport as well as flights between these airports. Section D.2 contains such a "closed network" formulation involving three airports.

It should also be noted that one can include other congested elements of the air traffic network in this model. For example, we could use the same formulation to model a congested sector of the network, interpreting the capacities not as landing capacities but as limits on total aircraft flow through the sector per time period.

\(^1\) If the fast algorithm is used we have to complement it with ways to compute the shadow costs that are necessary to proceed with the decomposition method. This can either be done by running the fast algorithm on several incremental problems or by inverting the constraint matrix once, whichever is faster.
D.2 Closed Network Formulation

This section contains a "closed network" formulation of the deterministic GHPP that assumes that the entire ATC network contains three airports and considers landing as well as take-off operations for all flights scheduled at these three airports during a 24-hour time span. Extending this formulation to an arbitrary number of airports presents no major difficulty but would have required a much more involved notation.

We consider operations at three airports, airport Z, airport Z', and airport Z'', during a 24-hour time span that has been subdivided into P time periods. We denote the time periods pertaining to each of the three airports by $T_1, T_2, \ldots, T_P$, $T'_1, T'_2, \ldots, T'_P$, and $T''_1, T''_2, \ldots, T''_P$, respectively. The (known) deterministic total (arrival + departure) capacities for these time periods are denoted by $K_1, K_2, \ldots, K_P$, $K'_1, K'_2, \ldots, K'_P$, and $K''_1, K''_2, \ldots, K''_P$, respectively.
Figure C-2 refers to a situation that we describe as follows:

- N flights, F₁,F₂,...,F₉, are scheduled to depart from airport Z to airport Z' and
- M flights, F₉₊₁,F₉₊₂,...,F₉₊₉, are scheduled to depart from airport Z to airport Z'' for each flight Fᵢ, we define:
- $P_i$ the index of the time period during which flight $F_i$ is scheduled to arrive at its destination airport.

- a zero-one variable $x_{ij}$ that is equal to 1 if flight $F_i$ is reassigned to a time period with index $j$, and equal to 0 otherwise.

- $C_{ij}$ the cost of reassigning flight $F_i$ to a time period with index $j$.

- $m_i$ the flight time of flight $F_i$ expressed in number of time periods (i.e., the index of the time period during which $F_i$ is scheduled to depart from airport $Z$ is $P_i - m_i$).

- $N'$ flights $F_1,F_2,...,F_{N'}$ are scheduled to depart from airport $Z'$ to airport $Z$ and $M'$ flights $F_{N'+1},F_{N'+2},...,F_{N'+M'}$ are scheduled to depart from airport $Z'$ to airport $Z''$. For each flight $F_i$, we define:

- $P_i'$ the index of the time period during which flight $F_i$ is scheduled to arrive at its destination airport.

- a zero-one variable $x'_{ij}$ that is equal to 1 if flight $F_i$ is reassigned to a time period with index $j$, and equal to 0 otherwise.

- $C_{ij}'$ the cost of reassigning flight $F_i$ to a time period with index $j$.

- $m_i'$ the flight time of flight $F_i$ expressed in number of time periods (i.e., the index of the time period during which $F_i$ is scheduled to depart from airport $Z'$ is $P_i' - m_i'$).

- $N''$ flights $F''_1,F''_2,...,F''_{N''}$ are scheduled to depart from airport $Z''$ to airport $Z$ and $M''$ flights $F''_{N''+1},F''_{N''+2},...,F''_{N''+M''}$ are scheduled to depart from airport $Z''$ to airport $Z'$. For each flight $F''_i$, we define:
- \( P''_i \) the index of the time period during which flight \( F''_i \) is scheduled to arrive at its destination airport.

- a zero-one variable \( x''_{ij} \) that is equal to 1 if flight \( F''_i \) is reassigned to a time period with index \( j \), and equal to 0 otherwise.

- \( C''_{ij} \) the cost of reassigning flight \( F''_i \) to a time period with index \( j \).

- \( m''_i \) the flight time of flight \( F''_i \) expressed in number of time periods (i.e., the index of the time period during which \( F''_i \) is scheduled to depart from airport \( Z'' \) is \( P''_i - m''_i \)).

- \( Q \) flights \( F_{k1}, F_{k2}, ..., F_{kQ} \) among \( F_1, F_2, ..., F_{N+M} \) are operated by aircraft that are used by other flights coming to airport \( Z \) from airport \( Z' \) or airport \( Z'' \). For each one of these flights \( F_{ki} \) we define:

- \( n_i \) the index of the flight that uses the same aircraft as \( F_{ki} \). (i.e., that flight is either \( F''_{ni} \) or \( F''_{ni} \)).

- \( R_i \) the index of the time period during which the flight that uses the same aircraft as \( F_{ki} \) is scheduled to land at airport \( Z \). (\( R_i \) is either \( P''_{ni} \) or \( P''_{ni} \))

- \( z_{nij} \) the zero-one assignment variable that corresponds to the flight that uses the same aircraft as \( F_{ki} \). (i.e., we either have \( z_{nij} = x'_{nij} \) or \( z_{nij} = x''_{nij} \)).

- \( L_i \) the minimum technical turnaround time for the aircraft that operates flight \( F_{ki} \) expressed in number of time periods.
• $Q'$ flights $F_{r1}F_{r2}...F_{rQ'}$ among $F_{1}'F_{2}'...F_{N'+M'}$ are operated by aircraft that are used by other flights coming to airport $Z'$ from airport $Z''$ or airport $Z$. For each one of these flights $F_{r1}$ we define:

  - $n_i'$ the index of the flight that uses the same aircraft as $F_{k1}$. (i.e., that flight is either $F_{n_i}''$ or $F_{n_i}''$.)

  - $R_i'$ the index of the time period during which the flight that uses the same aircraft as $F_{r1}$ is scheduled to land at airport $Z'$. ($R_i$ is either $P_{n_i}''$ or $P_{n_i}''$)

  - $z_{nij}'$ the zero-one assignment variable that corresponds to the flight that uses the same aircraft as $F_{r1}$. (i.e., we either have $z_{nij}' = x_{nij}$ or $z_{nij}' = x_{nij}'$.)

  - $L_i'$ the minimum technical turnaround time for the aircraft that operates flight $F_{r1}$ expressed in number of time periods.

• $Q''$ flights $F_{s1}''...F_{sQ''}$ among $F_{1}''F_{2}''...F_{N''+M''}$ are operated by aircraft that are used by other flights coming to airport $Z''$ from airport $Z'$ or airport $Z$. For each one of these flights $F_{s1}''$ we define:

  - $n_i''$ the index of the flight that uses the same aircraft as $F_{k1}''$. (i.e., that flight is either $F_{n_i}''$ or $F_{n_i}''$.)

  - $R_i''$ the index of the time period during which the flight that uses the same aircraft as $F_{s1}''$ is scheduled to land at airport $Z''$. ($R_i''$ is either $P_{n_i}'''$ or $P_{n_i}'''$)

  - $z_{n ij}''$ the zero-one assignment variable that corresponds to the flight that uses the same aircraft as $F_{s1}''$. (We either have $z_{n ij}'' = x_{n ij}''$ or $z_{n ij}'' = x_{n ij}''$.)
- $L''_i$ the minimum technical turnaround time for the aircraft that operates flight $F''_{s_i}$ expressed in number of time periods.

With this notation, the following zero-one IP finds, given the capacity constraints, the minimum cost assignment of each flight to a time period that does not violate the turnaround constraints:

$$\text{Min} \left\{ \sum_{i=1}^{N+M} \sum_{j=P_i}^{P+1} C_{ij} x_{ij} + \sum_{i=1}^{N'+M'} \sum_{j=P'_i}^{P'+1} C'_{ij} x'_{ij} + \sum_{i=1}^{N''+M''} \sum_{j=P''_i}^{P''+1} C''_{ij} x''_{ij} \right\}$$

Subject to:

(1) Assignment Constraints:

$$\sum_{j=P_i}^{P+1} x_{ij} = 1 \quad \text{for all } i = 1, \ldots, N+M$$

$$\sum_{j=P'_i}^{P'+1} x'_{ij} = 1 \quad \text{for all } i = 1, \ldots, N'+M'$$

$$\sum_{j=P''_i}^{P''+1} x''_{ij} = 1 \quad \text{for all } i = 1, \ldots, N''+M''$$

(2) Capacity Constraints: for all $j = 1, \ldots, P$

$$\sum_{i=1}^{N+M} x_{i(j+m_i)} + \sum_{i=1}^{N'} x'_{ij} + \sum_{i=1}^{N''} x''_{ij} \leq K_j$$

$$\sum_{i=1}^{N'+M'} x'_{i(j+m'_i)} + \sum_{i=1}^{N} x_{ij} + \sum_{i=1}^{M''} x''(N''+i)j \leq K'_j$$
\[ \sum_{i=1}^{N+M} x_{ij} \geq K_j \]

(3) "Turnaround Time" Constraints:

\[ \sum_{u \geq L_i} z_{ij}(j-m_{ki}-u) \geq x_{kj} \quad \text{for all } i = 1, \ldots, Q, \text{ and for all } j = R_i, \ldots, P+1 \]

\[ \sum_{u \geq L'_i} z'_{ij}(j-m'_{ri}-u) \geq x'_{rij} \quad \text{for all } i = 1, \ldots, Q', \text{ and for all } j = R'_i, \ldots, P+1 \]

\[ \sum_{u \geq L''_i} z''_{ij}(j-m''_{si}-u) \geq x''_{sij} \quad \text{for all } i = 1, \ldots, Q'', \text{ and for all } j = R''_i, \ldots, P+1 \]

(4) Integrality Constraints:

\[ x_{ij} = 0 \text{ or } 1 \quad \text{for all } i = 1, \ldots, N+M, \text{ and for all } j = P_i, \ldots, P+1 \]

\[ x'_{ij} = 0 \text{ or } 1 \quad \text{for all } i = 1, \ldots, N'+M', \text{ and for all } j = P'_i, \ldots, P+1 \]

\[ x''_{ij} = 0 \text{ or } 1 \quad \text{for all } i = 1, \ldots, N''+M'', \text{ and for all } j = P''_i, \ldots, P+1 \]

(Note that the variables denoted by \(z_{ij}, z'_{ij}, \) or \(z''_{ij}\) in the above IP are not additional variables but correspond to the proper \(x_{ij}, x'_{ij}, \) or \(x''_{ij}\) variables.)
APPENDIX E: Programs

This appendix contains the Fortran subroutines referred to in Chapter 5. Section E.1 contains the subroutines used for generating a sample problem and described in Section 5.2.1. Section E.2 contains the subroutines related to the deterministic GHPP and described in Section 5.3. Section E.3 contains the subroutines related to the probabilistic GHPP and described in Section 5.4.1.

E.1 Sample Schedule Generation

SUBROUTINE GEN

INTEGER SEED1, SEED2, MM, MTEMP
CHARACTER AIRFOR*3, FSTATU*3, INTFILE*10
CHARACTER FL(1:200)*8
CHARACTER APNAME(1:4)*3, STATUS(1:2)*3

INTEGER LINPI(1:2000)
CHARACTER FLIGHT(1:2000)*8
COMMON/FLSTUFF/TOTFL, BTIME, BTYPE, LSPER, FLIGHT, LINPI

INTEGER PORT, STAT, PERL, SHOUR, EHOUR, SEED, CASES, PERN, PERH
REAL KPROB(1:16)
COMMON/REQSTUFF/PORT, STAT, PERL, SHOUR, EHOUR, SEED, CASES,
1 PERN, PERH, KPROB

DATA APNAME/'BOS', 'LGA', 'ORD', 'XXX'/
DATA STATUS/'DEP', 'ARR'/

OPEN (UNIT=15, FILE='REQUEST', FORM='FORMATTED', STATUS='OLD',
1 ACCESS='SEQUENTIAL')
READ (15, '(I5)') PORT
READ (15, '(I5)') STAT
READ (15, '(I5)') PERL
READ (15, '(I5)') SHOUR
READ (15, '(I5)') EHOUR
READ (15, '(I5)') SEED

246
READ(15, '(I5)') CASES
DO (I=1,CASES)
READ(15, '(F5.2)') KPROB(I)
END DO
CLOSE(15)

PERL=5

AIRPOR=APNAME(PORT)
FSTATU=STATUS(STAT)

SEED1=SEED
TOTFL=0
DO 1000 I=1, (EHour-SHOUR)
CALL SCHED(AIRPOR,FSTATU,(SHOUR+I),SEED1,MM,FL)
DO 600 II=1,MM
FLIGHT(II+TOTFL)=FL(II)
600 CONTINUE
TOTFL=MM+TOTFL
SEED1=RANDI(1,1000)
1000 CONTINUE

DO (I=1,TOTFL)
WRITE(INFILE,'(A4)') FLIGHT(I)(2:5)
READ(INFILE,'(I4)') BTIME(I)
WRITE(INFILE,'(A1)') FLIGHT(I)(8:8)
READ(INFILE,'(I1)') BTYPE(I)
END DO

WRITE(9,'')
THE DATA GENERATED (BY GEN-SUB) IS FOR:
WRITE(9,'') APNAME(PORT), STATUS(STAT) // 'FLIGHTS'
WRITE(9,'') 'PERIOD LENGTH= ', ?PERL, ' ENDING AT ', DHOUR
,' SHOUR
RETURN
END

********************************************************************

SUBROUTINE SCHED (AIRPOR,FSTATU,HOUR,SEED3,MM,FL)

CHARACTER AIRPOR*3,FSTATU*3,CHOUR*2,DHOUR*2,FLHOUR*1
INTEGER HOUR,SEED3
CHARACTER FL(1:200)*8

REAL RANDU,RANDS
INTEGER RAND1,RANDY

INTEGER MM,JUNCK,TIMES(1:200)
INTEGER CATPER(1:4),TYPE(1:200)
CHARACTER FILNAM*6,OUTFILE*6,ORIGIN(1:200)*3,DEST(1:200)*3
CHARACTER APNAME(1:4)*3,J*3
CHARACTER INFILE*10,CTIME(1:200)*4,CTYPE(1:200)*1
REAL RMM

INTEGER LINPI(1:2000)
CHARACTER FLIGHT(1:2000) * 8
COMMON/FLSTUFF/TOTFL, BTIME, BTYPE, LSPER, FLIGHT, LINPI

INTEGER PORT, STAT, PERL, SHOUR, E_HOUR, SEED, CASES, FERN, PERH
REAL KPROB(1:16)
COMMON/REQSTUFF/PORT, STAT, PERL, SHOUR, E_HOUR, SEED, CASES,
1 PERN, PERH, KPROB

REAL TOTNUM, PERBOS, PERLGA, PERORD, HPER(1:24)
COMMON/DEMOSTUFF/TOTNUM, PERBOS, PERLGA, PERORD, HPER

INTEGER TYPAR(1:7)
COMMON/FMIXSTUFF/TYPAR

OPEN(UNIT=12, FILE=DEMFILE, FORM= 'FORMATTED', STATUS= 'OLD',
1 ACCESS= 'SEQUENTIAL')
READ (12, '(F10.2)') TOTNUM
READ (12, '(F10.2)') PERBOS
READ (12, '(F10.2)') PERLGA
READ (12, '(F10.2)') PERORD
DO (I=1, 24)
READ (12, '(F10.2)') HPER(I)
END DO
CLOSE (12)

OPEN(UNIT=14, FILE=FMIXFILE, FORM= 'FORMATTED', STATUS= 'OLD',
1 ACCESS= 'SEQUENTIAL')
DO (I=1, 7)
READ (14, '(I3)') TYPAR(I)
END DO
CLOSE (14)

WRITE (INFILE, '(I2)') HOUR
READ (INFILE, '(A2)') DHOUR
IF (HOUR.LT.10) THEN
    FHOUR=ADJUSTL(DHOUR)
CHOUR='0'//EHOUR
ELSE
    CHOUR=DHOUR
ENDIF

JUNCK=INT(RANDS(SEED3))
MM=MINT((TOTNUM*HPER(HOUR))/100)
WRITE(9,*) ' HOUR= ', (HOUR-1), ' - ', HOUR,
DO 500 I=1,MM
TIMES(I)=RANDI(0,59)
500 CONTINUE
CALL ARSORT(TIMES,MM)
APNAME(1)='BOS'
APNAME(2)='LGA'
APNAME(3)='ORD'
APNAME(4)='XXX'
CATPER(1)=PERBOS
CATPER(2)=PERLGA
CATPER(3)=PERORD
CATPER(4)=100-PERBOS-PERLGA-PERORD
DO 501 I=1,MM
TIMES(I)=((HOUR-1)*100)+TIMES(I)
IF (FSTATU='DEP') THEN
   ORIGIN(I) = AIRPOR
   DEST(I) = APNAME(RANTY(CATPER,4))
ELSEIF (FSTATU='ARR') THEN
   ORIGIN(I) = APNAME(RANTY(CATPER,4))
   DEST(I) = AIRPOR
ELSE
   WRITE(9,*) 'ERROR IN FSTATU, SHOULD BE DEP OR ARR'
ENDIF
TYPE(I) = RANTY(TYPAR,7)
501 CONTINUE
DO 504 I=1,MM
WRITE(INTFILE,'(I4)')TIMES(I)
READ(INTFILE,'(A4)')CTIME(I)
WRITE(INTFILE,'(I1)')TYPE(I)
READ(INTFILE,'(A1)')CTYPE(I)
IF (TIMES(I).LT.1000) THEN
   J=ADJUSTL(CTIME(I))
   FL(I)=FSTATU(1:1)//'O'//J//ORIGIN(I)(1:1)
   1 //DEST(I)(1:1)//CTYPE(I)
ELSE
   FL(I)=FSTATU(1:1)//CTIME(I)//ORIGIN(I)(1:1)
   1 //DEST(I)(1:1)//CTYPE(I)
ENDIF
504 CONTINUE
RETURN
END

******************************************************************************
SUBROUTINE DISCRET
INTEGER TSTART(1:500),TEND(1:500)
INTEGER CHOUR,NEXT,NUMF(1:500)
INTEGER TEGER LINPI(1:2000)
CHARACTER FLIGHT(1:2000)*8
COMMON/FILSTUFF/TOTFL, BTIME, BTYPE, LSPER, FLIGHT, LINPI

INTEGER PORT, STAT, Perl, SHOUR, EHOUR, SEED, CASES, PERN, PERH
REAL KPROB(1:16)
COMMON/REQSTUFF/PORT, STAT, Perl, SHOUR, EHOUR, SEED, CASES,
1 PERN, PERH, KPROB

REAL TOTNUM, PERBOS, PERLGA, PERORD, HPER(1:24)
COMMON/DEMSTUFF/TOTNUM, PERBOS, PERLGA, PERORD, HPER

INTEGER KAP(1:24)
COMMON/KAPSTUFF/KAP

CHARACTER*64 DEMFILE, CAPFILE, COSTFILE, FMIXFILE
COMMON/FILESTUFF/DEMFILE, CAPFILE, COSTFILE, FMIXFILE

OPEN (UNIT=18, FILE=CAPFILE, FORM='FORMATTED', STATUS='OLD',
1 ACCESS='SEQUENTIAL')
DO (I=1, SHOUR)
READ (18, ' (I4)') IDUMMY
END DO
DO (I=1, (EHOUR-SHOUR))
READ (18, ' (I4)') KAP(I)
END DO
CLOSE (18)

PERN= ((EHOUR-SHOUR) * 60) / PERL
PERH= 60 / PERL

NEXT= SHOUR * 100
OHHOUR= SHOUR * 100
DO 703 J=1, PERN
TSTART(J)= NEXT
TEND(J)= TSTART(J) + PERL - 1
IF ((TEND(J) - OHOUR(J)) .EQ. 59) THEN
NEXT= OHOUR+100
OHHOUR= OHOUR+100
ELSE
NEXT= TEND(J) + 1
ENDIF
703 CONTINUE

DO 555 I=1, TOTFL
DO 556 J=1, PERN
NUMF(J)= 0
IF ( (BTIME(I) .GE. TSTART(J)) .AND. (BTIME(I) .LE. TEND(J))) THEN
LSPER(I)= J
NUMF(J)= NUMF(J) + 1
ENDIF
ITEMP=0
DO (J=1,7)
DO (I=1,TOTFL)
IF(BTYPE(I) .EQ. (8-J)) THEN
LINPI(I)=TOTFL-ITEMP
ITEMP=ITEMP+1
ENDIF
END DO
END DO

RETURN
END

***************************************************************************
SUB SETCOSTS***************************************************************************
SUBROUTINE SETCOSTS
INTEGER NTEMP
REAL BETA, DELTA, THETA

INTEGER LINPI(1:2000)
CHARACTER FLIGHT(1:2000)*8
COMMON/FLSTUFF/TOTFL, BTIME, BTYPE, LSPER, FLIGHT, LINPI

INTEGER PORT, STAT, PERL, SHOUR, EHOUR, SEED, CASES, PERN, PERH
REAL KPROB(1:16)
COMMON/REQSTUFF/PORT, STAT, PERL, SHOUR, EHOUR, SEED, CASES,
1 PERN, PERH, KPROB

REAL CG(1:7,0:24), ALPHA, KAY, LC(1:7,0:500), C(1:7,0:500)
COMMON/COSTUFF/CG, ALPHA, KAY, LC, C

REAL TOTNUM, PERBOS, PERLGA, PERORD, HPER(1:24)
COMMON/DEMSUFF/TOTNUM, PERBOS, PERLGA, PERORD, HPER

CHARACTER*64 DEMFILE, CAPFILE, COSTFILE, FMIXFILE
COMMON/FILESTUFF/DEMFILE, CAPFILE, COSTFILE, FMIXFILE

OPEN(UNIT=16, FILE=COSTFILE, FORM='FORMATTED', STATUS='OLD',
1 ACCESS='SEQUENTIAL')
DO (I=1,7)
READ(16,'(F10.3)')CG(I,1)
END DO
READ(16,'(F10.3)')ALPHA
READ(16,'(F10.3)')KAY
CLOSE(16)

BETA=ALPHA+1
THETA=(BETA**((1/PERH))-1

DO (I=1,7)
CG(I,0) = 0.0
LC(I,0) = 0.0

LC(I,1) = CG(I,1)/PERH
END DO

DO (I=1,7)
  NTEMP=1
  DO (J=1,(EHOUR-SHOUR))
    DO (K=NTEMP,NTEMP+PERH-1)
      LC(I,K)=LC(I,1)
    END DO
    NTEMP=NTEMP+PERH
  END DO
END DO

DO 10 I=1,7
  C(I,0)=0.0
  IF (ALPHA.EQ.0) THEN
    C(I,1)=CG(I,1)/PERH
  ELSE
    C(I,1)=CG(I,1)*THETA/ALPHA
  ENDIF
  DO 11 J=1,(EHOUR-SHOUR)
  CG(I,J)=(BETA**(J-1))*CG(I,1)
11    CONTINUE
10 CONTINUE

DO 13 I=1,7
  NTEMP=1
  DO 14 J=1,(EHOUR-SHOUR)
    DO 15 K=NTEMP,NTEMP+PERH-1
      C(I,K)=((1+THETA)**(K-1))*C(I,1)
15      CONTINUE
    NTEMP=NTEMP+PERH
14 CONTINUE
13 CONTINUE

RETURN
END
E.2 Deterministic Subroutines

SUBROUTINE GENOKA (FLIGHT, MM, PERL, IEND, JEND, T1, T2, T3, NODES, ARCS)

CHARACTER FLIGHT(1:2000)*8, INTFILE*10, APORT*3
INTEGER PERL, PERN, PERH, NODES, ARCS, IEND, JEND, MM, SHOUR, EHOUR
DIMENSION IEND(5000), JEND(5000), T1(5000), T2(5000), T3(5000)
REAL T1, T2, T3
INTEGER PERKAP(1:500), ACOUNT, IND(1:12), GAMMA, L, K
INTEGER KAP(1:24), NEXT, OHOUR, TSTART(1:500), TEND(1:500), NTEMP
REAL CG(1:7, 0:24), C(1:7, 0:500), ALPHA, BETA, DELTA, THETA

OPEN (UNIT=15, FILE='REQUEST', FORM='FORMATTED', STATUS='OLD', 1
ACCESS='DIRECT', RECL=5)
READ (15, '(I5)', REC=4) SHOUR
READ (15, '(I5)', REC=5) EHOUR
CLOSE (15)

PERN = ((EHOUR-SHOUR)*60)/PERL
PERH = 60/P Erl

DO 603 I=1, MM
WRITE (INTFILE, '(A4)'), FLIGHT(I) (2:5)
READ (INTFILE, '(I4)'), BTIME(I)
WRITE (INTFILE, '(A1)'), FLIGHT(I) (8:8)
READ (INTFILE, '(I1)'), BTYPE(I)

ARCS = 0
NEX T = SHOUR*100
OHOUR = SHOUR*100
DO 703 J=1, PERN
TSTART(J) = NEXT
TEND(J) = TSTART(J) + PERL - 1
IF ((TEND(J) - OHOUR).EQ.59) THEN
NEXT = OHOUR+100
ELSE
NEXT = TEND(J) + 1
ENDIF

703 CONTINUE

DO 55 I=1, MM
DO 56 J=1, PERN
IF ((B TIME (I) .GE. T START (J)) .AND. (B TIME (I) .LE. T END (J))) THEN
LSPER(I) = J
ARCS = ARCS + PERN - LSPER(I) + 2 + 1
ENDIF

56 CONTINUE
55 CONTINUE
IF (FLIGHT(1) (1:1) .EQ. 'A') THEN
    IF(FLIGHT(1) (7:7) .EQ. 'B') THEN
        APORT='BOS'
    ELSEIF (FLIGHT(1) (7:7) .EQ. 'L') THEN
        APORT='LGA'
    ELSEIF (FLIGHT(1) (7:7) .EQ. 'O') THEN
        APORT='ORD'
    ELSE
        PRINT*, 'ERROR IN DESTINATION AIRPORT'; PAUSE
    ENDIF
ELSEIF (FLIGHT(1) (1:1) .EQ. 'D') THEN
    IF(FLIGHT(1) (6:6) .EQ. 'B') THEN
        APORT='BOS'
    ELSEIF (FLIGHT(1) (6:6) .EQ. 'L') THEN
        APORT='LGA'
    ELSEIF (FLIGHT(1) (6:6) .EQ. 'O') THEN
        APORT='ORD'
    ELSE
        PRINT*, 'ERROR IN DEPARTURE AIRPORT'; PAUSE
    ENDIF
ENDIF

OPEN(UNIT=18, FILE=APORT//'KAP', FORM='FORMATTED', STATUS='OLD', ACCESS='DIRECT', RECL=4)
DO 2010 I=1, (EHOUR-SHOUR)
    READ(18, '(I4)', REC=SHOUR+I) KAP(I)
    PRINT*, ' KAP(', SHOUR+I-1, '-', SHOUR+I, ') = ', KAP(I)
2010 CONTINUE
CLOSE(18)

NEXT=1
DO 2030 I=1, (EHOUR-SHOUR)
    GAMMA=0
    DO 2030 J=NEXT, NEXT+PERH-1
    PERKAP(J)=INT((KAP(I)/PERH)+0.001)
    GAMMA=GAMMA+PERKAP(J)
    2030 CONTINUE
    NEXT=NEXT+PERH
IF (GAMMA.LT.KAP(I)) THEN
    IF (PERL.EQ.5) THEN
        IND(1)=1; IND(2)=7; IND(3)=4; IND(4)=10
        IND(5)=2; IND(6)=8; IND(7)=5; IND(8)=11
        IND(9)=3; IND(10)=9; IND(11)=6; IND(12)=12
    ELSEIF (PERL.EQ.10) THEN
        IND(1)=1; IND(2)=4; IND(3)=2
        IND(4)=5; IND(5)=3; IND(6)=6
    ELSEIF (PERL.EQ.15) THEN
        IND(1)=1; IND(2)=3; IND(3)=2; IND(4)=4
    ELSEIF ((PERL.EQ.20) .OR. (PERL.EQ.30) .OR. (PERL.EQ.60)) THEN
        DO 2032 L=1, PERH
        IND(L)=L
        2032 CONTINUE
    ELSE
        PRINT*, 'ERROR IN PERIOD LENGTH INPUT'
END
PRINT*, 'MUST BE ONE OF 5, 10, 15, 20, 30 or 60'
PAUSE
ENDIF
DO 2031 K=1, (KAP(I)-GAMMA)
   PERKAP(((I-1)*PERH+IND(K))=PERKAP(((I-1)*PERH+IND(K))+1
2031   CONTINUE
ENDIF
2020 CONTINUE
DO 9000 I=1,7
   CG(I,0)=0
9000 CONTINUE
OPEN(UNIT=16, FILE='COSTS', FORM='FORMATTED', STATUS='OLD',
1 ACCESS='SEQUENTIAL')
DO 5600 I=1,7
   READ(16,'(F10.3)')CG(I,1)
5600 CONTINUE
READ(16,'(F10.3)')ALPHA
CLOSE(16)
BETA=ALPHA+1
THETA=(BETA**((1/PERH))-1
DO 10 I=1,7
   CG(I,0)=0.0
   C(I,0)=0.0
   IF (ALPHA.EQ.0) THEN
      C(I,1)=CG(I,1)/PERH
   ELSE
      C(I,1)=CG(I,1)*THETA/ALPHA
   ENDIF
10 CONTINUE
DO 11 J=1, (EHOUR-SHOUR)
   IF (ALPHA.EQ.0) THEN
      CG(I,J)=CG(I,1)
   ELSE
      CG(I,J)=(BETA**((J-1)))*CG(I,1)
   ENDIF
11 CONTINUE
DO 13 I=1,7
   NTEMP=1
   DO 14 J=1, (EHOUR-SHOUR)
      NTEMP=NTEMP+PERH-1
      IF (THETA.EQ.0) THEN
         C(I,K)=C(I,1)
      ELSE
         C(I,K)=((1+THETA)**(K-1)))*C(I,1)
      ENDIF
   14 CONTINUE
13 CONTINUE

DO 2201 I=1,7
DO 2301 K=1,PERN
C(I,K)=AIMT(C(I,K))
2301 CONTINUE
2201 CONTINUE

NODES=M+PERN+1+2
ARCS=ARCS+PERN+2
ACOUNT=1

DO 2000 I=M,M+1
   IEND(ACOUNT)=M+PERN+1+2 ; JEND(ACOUNT)=I
   T1(ACOUNT)=1.0 ; T2(ACOUNT)=1.0 ; T3(ACOUNT)=0.0
   ACOUNT=ACOUNT+1
DO 2001 J=1,PERN+2-LSPER(I)
   IEND(ACOUNT)=I ; JEND(ACOUNT)=M+LSPER(I)+J-1
   T1(ACOUNT)=0.0 ; T2(ACOUNT)=1.0 ;
   T3(ACOUNT)=T3(ACOUNT-1)+C(BTYPE(I),J-1)
   ACOUNT=ACOUNT+1
2001 CONTINUE
2000 CONTINUE

DO 2002 I=1,PERN
   IEND(ACOUNT)=M+I ; JEND(ACOUNT)=M+PERN+2
   T1(ACOUNT)=0.0 ; T2(ACOUNT)=REAL(PERKAP(I)) ; T3(ACOUNT)=0.0
   ACOUNT=ACOUNT+1
2002 CONTINUE

IEND(ACOUNT)=M+PERN+1 ; JEND(ACOUNT)=M+PERN+2
T1(ACOUNT)=0.0 ; T2(ACOUNT)=REAL(M) ; T3(ACOUNT)=0.0
ACOUNT=ACOUNT+1

IEND(ACOUNT)=M+PERN+2 ; JEND(ACOUNT)=M+PERN+3
T1(ACOUNT)=REAL(M) ; T2(ACOUNT)=REAL(M) ; T3(ACOUNT)=0.0

END
SUBROUTINE OKALG(I,J,T2,T1,FLOW,PI,T3,NODES,ARCS)

C

DIMENSION I(5000),J(5000),HI(5000),LO(5000),FLOW(5000),
1   PI(5000),COST(5000),T1(5000),T2(5000),T3(5000)
INTEGER   ARCS,NODES,FLOW,PI,COST,HI
REAL T1,T2,T3
LOGICAL INFES
100 FORMAT(2I10)
C
C READS THE TOTAL NUMBER OF NODES AND ARCS FOR THE PROBLEM
C

DO 1 M=1,ARCS
HI(M) = T2(M)+0.00001
LO(M) = T1(M)+0.00001
COST(M) = T3(M)+0.00001
1 CONTINUE
250 FORMAT(2I5,3F10.2)
C
C Initializes arc flows
C
DO 5 M=1,ARCS
5 FLOW(M) = 0
C
C Initializes PI values
C
DO 10 M=1,NODES
10 PI(M) = 0
C
C Calls the subroutine which evaluates the feasible netflow
C for each arc
C
CALL NETFLO(INFES,I,J,HI,LO,FLOW,PI,COST,NODES,ARCS)
IF(.NOT.INFES) WRITE(9,120)
C
C Prints the final summary report
C
WRITE(9,801)
801 FORMAT('1',12X,
X '****SOLUTION BY OUT-OF-KILTER ALGORITHM****')
WRITE(9,112)
112 FORMAT(/,20X,'FINAL SUMMARY REPORT ///</>
PRINT*, 'THE NUMBER OF NODES IS ',NODES
PRINT*, 'THE NUMBER OF ARCS IS ',ARCS
TCOST = 0.0
DO 140 M=1,ARCS
140 TCOST = COST(M)*FLOW(M) + TCOST
CONTINUE
WRITE(9,150) TCOST
150 FORMAT ///</ TOTAL PROJECT COST =',F10.2)
120 FORMAT(' SOLUTION INFEASIBLE')
WRITE(9,750)
750 FORMAT(/5X,'SENSITIVITY ANALYSIS ///</)
WRITE(9,753)
753 FORMAT(12X,'M',5X,'I',5X,'J',10X,'HI',10X,'LO',
X 10X,'FLOW',10X,'COST ///</)
DO 751 M=1,ARCS
IZ  = FLOW(M)
IF(IZ.EQ.0) GO TO 751
WRITE(9,752) M, I(M), J(M), HI(M), LO(M), FLOW(M), COST(M)
752 FORMAT(7X,3(2X,I4),4(3X,I10))
751 CONTINUE
RETURN
END

**********************************************************************
NETFLO**********************************************************************

SUBROUTINE NETFLO(INFES, I, J, HI, LO, FLOW, PI, COST, NODES, ARCS)

DIMENSION I(5000), J(5000), HI(5000), LO(5000), FLOW(5000),
1  PI(5000), COST(5000), NA(5000), NB(5000)

LOGICAL     INFES
INTEGER      A, AOK, C, COK, DEL, E, EPS, INF, LAB, N, NI, NJ, SRC, SNK,
X              FLOW, PI, NA, NODES, ARCS, I, J, COST, HI, LO, NB

C CHECK FEASIBILITY OF FORMULATION
C
INFES = .TRUE.
DO 10 A=1,ARCS
  IF(LO(A).GT.HI(A)) GO TO 39
10 CONTINUE
C
SET INF TO MAX AVAILABLE INTEGER
C
16 INF  = 999999
  AOK = 0
C
FIND OUT OF KILTER ARC
C
20 DO 21 A=1,ARCS
  IA = I(A)
  JA = J(A)
  C  = COST(A) + PI(IA) - PI(JA)
C
CHECKS THE CONDITIONS THE INDIVIDUAL ARCS ARE IN.
C
  IF((FLOW(A).LT.LO(A)).OR.
   X (C.LT.0.AND.FLOW(A).LT.HI(A))) GO TO 22
  IF((FLOW(A).GT.HI(A)).OR.
   X (C.GT.0.AND.FLOW(A).GT.LO(A))) GO TO 23
21 CONTINUE
C
NO REMAINING OUT OF KILTER ARCS
C
GO TO 38
22   SRC = J(A)
    SNK = I(A)
    E   = +1
    GO TO 24
23   SRC = I(A)
    SNK = J(A)
    E   = -1
    GO TO 24
24 DO 99 N=1,NODES

258
NA(N) = 0
NB(N) = 0
99 CONTINUE

IF((A.EQ.AOK).AND.(NA(SRC).NE.0)) GO TO 25

C ATTEMPT TO BRING OUT OF KILTER ARCS INTO KILTER

C AOK = A

DO 26 N=1,NODES
   NA(SRC) = IABS(SNK)*E
   NB(SRC) = IABS(AOK)*E
26  COK = C
    LAB = 0

DO 30 A=1,ARCS
   IA = I(A)
   JA = J(A)
   IF((NA(IA).EQ.0.AND.NA(JA).EQ.0).OR.
      (NA(IA).NE.0.AND.NA(JA).NE.0)) GO TO 30
   C = COST(A) + PI(IA) - PI(JA)
   IF(NA(JA).EQ.0) GO TO 28
   IF(FLOW(A).GE.HI(A).OR.
      (FLOW(A).GE.LO(A).AND.C.GT.0)) GO TO 30
   NA(JA) = I(A)
   NB(JA) = A
   GO TO 29
28 IF(FLOW(A).LE.LO(A).OR.
    (FLOW(A).LE.HI(A).AND.C.LT.0)) GO TO 30
   IA = I(A)
   NA(IA) = -J(A)
   NB(IA) = -A
29 LAB = 1

C NODE LABELED, TEST FOR BREAKTHRU

C IF(NA(SNK).NE.0) GO TO 33
30 CONTINUE

C NO BREAKTHRU

C IF(LAB.NE.0) GO TO 27

C DETERMINE CHANGE TO PI VECTOR

C DEL = INF

DO 31 A=1,ARCS
   IA = I(A)
   JA = J(A)
   IF((NA(IA).EQ.0.AND.NA(JA).EQ.0).OR.
      (NA(IA).NE.0.AND.NA(JA).NE.0)) GO TO 31
   C = COST(A) + PI(IA) - PI(JA)
   IF(NA(JA).EQ.0.AND.FLOW(A).LT.HI(A))
      DEL = MIN0(DEL,C)
   IF(NA(JA).NE.0.AND.FLOW(A).GT.LO(A))
      DEL = MIN0(DEL,-C)
31 CONTINUE

IF(DEL.EQ.INF.AND.(FLOW(AOK).EQ.HI(AOK).OR.FLOW(AOK).EQ.LO(AOK))) DEL = IABS(COK)
IF (DEL.EQ.INF) GO TO 39

EXIT, NO FEASIBLE FLOW PATTERN
CHANGE PI VECTOR BY COMPUTED DEL

DO 32 N=1,NODES
32 IF (NA(N).EQ.0) PI(N) = PI(N) + DEL

FIND ANOTHER OUT-OF-KILTER ARC

GO TO 20

BREAKTHRU COMPUTE INCREMENTAL FLOW

33 EPS = INF
NI = SRC
34 NJ = IABS (NA(NI))
A = IABS (NB(NI))
C = COST (A) - ISIGN (IABS (PI(NI) - PI(NJ)),NB(NI))
IF (NB(NI).LT.0) GO TO 35
IF (C.GT.0.AND.FLOW(A).LT.LO(A))
X EPS = MIN0 (EPS,LO(A) - FLOW(A))
IF (C.LE.0.AND.FLOW(A).LT.HI(A))
X EPS = MIN0 (EPS,HI(A) - FLOW(A))
GO TO 36
35 IF (C.LT.0.AND.FLOW(A).GT.HI(A))
X EPS = MIN0 (EPS,FLOW(A) - HI(A))
IF (C.GE.0.AND.FLOW(A).GT.LO(A))
X EPS = MIN0 (EPS,FLOW(A) - LO(A))
36 NI = NJ
IF (NI.NE.SRC) GO TO 34

CHANGE FLOW VECTOR BY COMPUTED EPS

37 NJ = IABS (NA(NI))
A = IABS (NB(NI))
FLOW(A) = FLOW(A) + ISIGN (EPS,NB(NI))
NI = NJ
IF (NI.NE.SRC) GO TO 37

FIND ANOTHER OUT OF KILTER ARC

AOK = 0
GO TO 20
39 INFES = .FALSE.
38 CONTINUE RETURN
END

******************************************************************************
SUBROUTINE BPOL FOR DETERMINISTIC CASE******************************************************************************

CHARACTER FLIGHT(1:2000)*8,INFILE*10,APORT*10
INTEGER BDELAY(1:2000)
INTEGER PERL,PERN,PERH,MM,SHOUR,SHOUR
INTEGER PERKAP(1:500),ACOUNT,IND(1:12),GAMMA,L,K
INTEGER KAP (1:24), NEXT, OHOUR, TSTART (1:500), TEND (1:500), NTEMP
REAL CG (1:7, 0:24), C (1:7, 0:500), ALPHA, BETA, DELTA, THETA

INTEGER TEMP, NE, K, OLPER (1:2000)
REAL TEMPCO (1:2000), TRESH

OPEN (UNIT=15, FILE='REQUEST', FORM='FORMATTED', STATUS='OLD',
  ACCESS='DIRECT', RECL=5)
READ (15, '(I5)', REC=3) PERL
READ (15, '(I5)', REC=4) SHOUR
READ (15, '(I5)', REC=5) EHOUR
CLOSE (15)

PERN = ((EHOUR-SHOUR)*60)/PERL
PERH = 60/PERL

NEXT = SHOUR*100
OHHOUR = SHOUR*100
DO 703 J=1, PERN
    TSTART (J) = NEXT
    TEND (J) = TSTART (J) + PERL - 1
    IF (TEND (J) - OHOUR) .EQ. 59 THEN
        NEXT = OHOUR + 100
        OHOUR = OHOUR + 100
    ELSE
        NEXT = TEND (J) + 1
    ENDF

    703 CONTINUE

DO 555 I = 1, MM
    DO 556 J = 1, PERN
        IF ((BTIME (I) .GE. TSTART (J)) .AND. (BTIME (I) .LE. TEND (J))) THEN
            LSPER (I) = J
        ENDF

    556 CONTINUE

    555 CONTINUE

    IF (FLIGHT (1) (1:1) .EQ. 'A') THEN
        IF (FLIGHT (1) (7:7) .EQ. 'B') THEN
            APORT = 'BOS'
        ELSEIF (FLIGHT (1) (7:7) .EQ. 'L') THEN
            APORT = 'LGA'
        ELSEIF (FLIGHT (1) (7:7) .EQ. 'O') THEN
            APORT = 'ORD'
        ELSE
            PRINT *, 'ERROR IN DESTINATION AIRPORT'; PAUSE
        ENDF
    ELSEIF (FLIGHT (1) (1:1) .EQ. 'D') THEN
        IF (FLIGHT (1) (6:6) .EQ. 'B') THEN
            APORT = 'BOS'
        ELSEIF (FLIGHT (1) (6:6) .EQ. 'L') THEN
            APORT = 'LGA'
        ELSEIF (FLIGHT (1) (6:6) .EQ. 'O') THEN
            APORT = 'ORD'
    END
ELSE
PRINT*, 'ERROR IN DEPARTURE AIRPORT'; PAUSE
ENDIF
ENDIF

OPEN(UNIT=18, FILE='APORT/''KAP', FORM='FORMATTED', STATUS='OLD',
1 ACCESS='DIRECT', RECL=4)
DO 2010 I=1, (EHOURL-SHOUR)
READ (18, '(I4)', REC=SHOUR+I) KAP(I)
2010 CONTINUE
CLOSE(18)

NEXT=1
DO 2020 I=1, (EHOURL-SHOUR)
   GAMMA=0
   DO 2030 J=NEXT, NEXT+PERH-1
   PERKAP(J)=INT((KAP(I)/PERH)+0.001)
   GAMMA=GAMMA+PERKAP(J)
2030 CONTINUE
   NEXT=NEXT+PERH
   IF (GAMMA.LT.KAP(I)) THEN
      IF (PERL.EQ.5) THEN
         IND(1)=1; IND(2)=7; IND(3)=4; IND(4)=10
         IND(5)=2; IND(6)=8; IND(7)=5; IND(8)=11
         IND(9)=3; IND(10)=9; IND(11)=6; IND(12)=12
      ELSEIF (PERL.EQ.10) THEN
         IND(1)=1; IND(2)=4; IND(3)=2
         IND(4)=5; IND(5)=3; IND(6)=6
      ELSEIF (PERL.EQ.15) THEN
         IND(1)=1; IND(2)=3; IND(3)=2; IND(4)=4
      ELSEIF ((PERL.EQ.20).OR. (PERL.EQ.30).OR. (PERL.EQ.60)) THEN
         DO 2032 L=1, PERH
            IND(L)=L
         2032 CONTINUE
      ELSE
         PRINT*, 'ERROR IN PERIOD LENGTH INPUT'
         PRINT*, 'MUST BE ONE OF 5,10,15,20,30 OR 60'
         PAUSE
      ENDIF
   DO 2031 K=1, (KAP(I)-GAMMA)
   PERKAP(((I-1)*PERH+IND(K))=PERKAP(((I-1)*PERH+IND(K))+1
2031 CONTINUE
ENDIF
2020 CONTINUE

DO 9000 I=1, 7
   CG(I,0)=0
9000 CONTINUE

OPEN(UNIT=16, FILE='COSTS', FORM='FORMATTED', STATUS='OLD',
1 ACCESS='SEQUENTIAL')
DO 5600 I=1, 7
   READ (16, '(F10.3)')CG(I,1)
5600 CONTINUE
READ (16, '(F10.3)')ALPHA
CLOSE(16)
BETA=ALPHA+1
THETA=(BETA**(1/PERH)) - 1

DO 10 I=1,7
   CG(I,0)=0.0
   C(I,0)=0.0
   IF (ALPHA.EQ.0) THEN
      C(I,1)=CG(I,1)/PERH
   ELSE
      C(I,1)=CG(I,1)*THETA/ALPHA
   ENDIF
   DO 11 J=1,(EHOUR-SHOUR)
   CG(I,J)=(BETA**((J-1)))*CG(I,1)
   11 CONTINUE
10 CONTINUE

DO 13 I=1,7
   NTEMP=1
   DO 14 J=1,(EHOUR-SHOUR)
      DO 15 K=NTEMP,NTEMP+PERH-1
      C(I,K)=((1+THETA)**(K-1))*C(I,1)
      15 CONTINUE
   NTEMP=NTEMP+PERH
14 CONTINUE
13 CONTINUE

DO 55 I=1,MM
   OLP(I)=PERN+1
55 CONTINUE

DO 66 J=1,PERN
   K=0
   TEMP=0
   DO 65 I=1,MM
      IF ((LSPER(I),LE,J).AND.(OLP(I),GE,J)) THEN
         K=K+1
         TEC(K)=C(BTYPE(I),J-LSPER(I)+1)
      ENDIF
65 CONTINUE
CALL ARSORT(TEC,K)

IF (PERKAP(J),GE,K) THEN
   DO 64 I=1,MM
      IF ((LSPER(I),LE,J).AND.(OLP(I),GE,J)) THEN
         OLP(I)=J
      ENDIF
64 CONTINUE
ELSE
   TRESH=TEC(K-PERKAP(J))
   DO 63 I=1,MM
      IF ((LSPER(I),LE,J).AND.(OLP(I),GE,J)
.AND. (C(BTYPE(I),J+1-LSPER(I)) \
.GT.TRESH)) THEN
  OLPST(J)=J \
  TEMP=TEMP+1 \
ENDIF
CONTINUE

DO 62 I=1,MM
IF((LSPER(I).LE.J).AND.(OLPER(I).GE.J) \
.AND. (C(BTYPE(I),J+1-LSPER(I)) \
.EQ.TRESH).AND. \
(TMP.LT.PERKAP(J))) THEN
  OLPST(J)=J \
  TEMP=TEMP+1 \
ENDIF
CONTINUE

FDNDF

CONTINUE

DO 78 I=1,MM
  BDELAY(I)=PERN+1-LSPER(I)
CONTINUE

DO 77 J=1,PERN
  DO 76 I=1,MM
    IF (OLPER(I).EQ.J) THEN
      BDELAY(I)=J-LSPER(I)
    ENDIF
    CONTINUE

END

******************************************FCFS******************************************
SUBROUTINE FCFS(FLIGHT,MM,BTIME,BTYPE,CDELAY,LSPER)

CHARACTER FLIGHT(1:2000) *8, INFILE*10, APORT*3
INTEGER CDELAY(1:2000)
INTEGER PERL, PERN, PERH, MM, SHOUR, EHOUR
INTEGER PERKAP(1:500), ACOUNT, IND(1:12), GAMMA, L, K

INTEGER KAP(1:24), NEXT, OHHOUR, TSTART(1:500), TEND(1:500), NTEMP
REAL CG(1:7,0:24), C(1:7,0:500), ALPHA, BETA, DELTA, THETA

INTEGER TEMP, NE,K,OLPER(1:2000)
REAL TEMPCO(1:2000), TRESH

OPEN(UNIT=15,FILE='REQUE', FORM='FORMATTED', STATUS='OLD', 
1 ACCESS='DIRECT', RECL=5)
READ(15,'(i5) ',REC=3) PERL
READ(15,'(i5) ',REC=4) SHOUR
READ(15,'(i5) ',REC=5) EHOUR
CLOSE(15)

PERN= ( (EHOUR-SHOUR) *60 ) / PERL

264
PERH=60/PERL

NEXT=SHOUR*100
OHHOUR=SHOUR*100
DO 703 J=1,PERN
  TSTART(J)=NEXT
  TEND(J)=TSTART(J)+PERL-1
  IF((TEND(J)=OHHOUR).EQ.59) THEN
    NEXT=OHHOUR+100
    OHHOUR=OHHOUR+100
  ELSE
    NEXT=TEND(J)+1
  ENDIF
703    CONTINUE

DO 555 I=1,MM
  DO 556 J=1,PERN
  IF((BTIME(I).GE.TSTART(J)).AND.(BTIME(I).LE.TEND(J))) THEN
    LSUPER(I)=J
  ENDIF
556    CONTINUE
555    CONTINUE

IF (FLIGHT(1)(1:1).EQ.'A') THEN
  IF(FLIGHT(1)(7:7).EQ.'B') THEN
    APORT='BOS'
  ELSEIF (FLIGHT(1)(7:7).EQ.'L') THEN
    APORT='LGA'
  ELSEIF (FLIGHT(1)(7:7).EQ.'O') THEN
    APORT='ORD'
  ELSE
    PRINT*, 'ERROR IN DESTINATION AIRPORT'; PAUSE
  ENDIF
ELSEIF (FLIGHT(1)(1:1).EQ.'D') THEN
  IF(FLIGHT(1)(6:6).EQ.'B') THEN
    APORT='BOS'
  ELSEIF (FLIGHT(1)(6:6).EQ.'L') THEN
    APORT='LGA'
  ELSEIF (FLIGHT(1)(6:6).EQ.'O') THEN
    APORT='ORD'
  ELSE
    PRINT*, 'ERROR IN DEPARTURE AIRPORT'; PAUSE
  ENDIF
ENDIF

OPEN(UNIT=18, FILE=APORT//'KAP', FORM='FORMATTED', STATUS='OLD',
  ACCESS='DIRECT', RECL=4)
WRITE(9, 6299)
6299   FORMAT(2X,'FROM', 3X,' TO', 5X,' CAPACITY/HOUR')/
  DO 2010 I=1, (OHHOUR-SHOUR)
  READ(18, '(I4)', REC=SHOUR+I) KAP(I)
2010  WRITE(9, 6300) SHOUR+I-1, SHOUR+I, KAP(I)
WRITE(9, 6300) SHOUR+I-1, SHOUR+I, KAP(I)
6300   FORMAT(2X,I4,3X,I4,5X,I13)
2010 CONTINUE
CLOSE(18)

WRITE(9,6401)

6401 FORMAT(2X,' HOUR ',5X,' CAPACITY BREAKDOWN ')
NEXT=1
DO 2020 I=1,(EHOUR-SHOUR)
   GAMMA=0
   DO 2030 J=NEXT,NEXT+PERH-1
      PERKAP(J)=INT((KAP(I)/PERH)+0.001)
      GAMMA=GAMMA+PERKAP(J)
   2030 CONTINUE
   NEXT=NEXT+PERH
IF (GAMMA.LT.KAP(I)) THEN
   IF (PERL.EQ.5) THEN
      IND(1)=1; IND(2)=7; IND(3)=4; IND(4)=10
      IND(5)=2; IND(6)=8; IND(7)=5; IND(8)=11
      IND(9)=3; IND(10)=9; IND(11)=6; IND(12)=12
   ELSEIF (PERL.EQ.10) THEN
      IND(1)=1; IND(2)=4; IND(3)=2
      IND(4)=5; IND(5)=3; IND(6)=6
   ELSEIF (PERL.EQ.15) THEN
      IND(1)=1; IND(2)=3; IND(3)=2; IND(4)=4
   ELSEIF (PERL.EQ.20).OR.(PERL.EQ.30).OR.(PERL.EQ.60)) THEN
      DO 2032 L=1,PERH
      IND(L)=L
   2032 CONTINUE
   ELSE
      PRINT*, 'ERROR IN PERIOD LENGTH INPUT'
      PRINT*, 'MUST BE ONE OF 5,10,15,20,30 or 60'
      PAUSE
   ENDIF
   DO 2031 K=1,(KAP(I)-GAMMA)
      PERKAP(((I-1)*PERH)+IND(K))=PERKAP(((I-1)*PERH)+IND(K))+1
   2031 CONTINUE
   ENDIF
WRITE(9,6402) SHOUR+I-1,SHOUR+I,1
   (PERKAP((I*PERH)-PERH+KK),KK=1,PERH)
6402 FORMAT(2X,I3,'-',I3,5X,12(2X,I2))
2020 CONTINUE

DO 9000 I=1,7
   CG(I,0)=0
9000 CONTINUE

OPEN(UNIT=16,FILE='COSTS',FORM='FORMATTED',STATUS='OLD',1
   ACCESS='SEQUENTIAL')
DO 5600 I=1,7
   READ(16,'(F10.3)')CG(I,1)
5600 CONTINUE
   READ(16,'(F10.3)')ALPHA
   CLOSE(16)

BETA=ALPHA+1
THETA=(BETA**(1/PERH))-1
DO 10 I=1,7
   CG(I,0)=0.0
   C(I,0)=0.0
   IF (ALPHA.EQ.0) THEN
      C(I,1)=CG(I,1)/PERH
   ELSE
      C(I,1)=CG(I,1)*THETA/ALPHA
   ENDIF
   DO 11 J=1,(EHOUR-SHOUR)
      CG(I,J)=(BETA**(J-1))*CG(I,1)
      CONTINUE
   11 CONTINUE
   DO 12 I=1,7
      NTEMP=1
      DO 13 J=1,(EHOUR-SHOUR)
         K=NTEMP,NTEMP+PERH-1
         C(I,K)=((1+THETA)**(K-1))*C(I,1)
         CONTINUE
         NTEMP=NTEMP+PERH
      13 CONTINUE
      CONTINUE
   12 CONTINUE
   DO 55 I=1,MM
      OLPER(I)=PERN+1
      CONTINUE
   55 CONTINUE
   DO 66 J=1,PERN
      K=0
      DO 65 I=1,MM
         AND.(K.LT.PERKAP(J))) THEN
            OLPER(I)=J
            K=K+1
         ENDIF
         CONTINUE
      66 CONTINUE
      CONTINUE
   65 CONTINUE
   DO 78 I=1,MM
      CDELAY(I)=PERN+1-LSPER(I)
      CONTINUE
   78 CONTINUE
   DO 77 J=1,PERN
      DO 76 I=1,MM
         IF (OLPER(I).EQ.J) THEN
            CDELAY(I)=J-LSPER(I)
         ENDIF
      76 CONTINUE
      CONTINUE
   77 CONTINUE
   END
E.3 Probabilistic Subroutines

SUBROUTINE DPL3(PRIO)

INTEGER PRIO(1:2000), ITER, ITNUM, I3START, KEND, ARRP
INTEGER*1 X(1:700, 0:6, 0:6, 0:6)
INTEGER HIGHER(1:500), NF1, NF2, NF3, F1, F2, F3
REAL INDCOST, G(0:6, 0:6, 0:6), CTEMP, CTEMP
LOGICAL STILL(1:2000)

INTEGER PKAP(1:16, 1:24)
COMMON/PKAPSTUFF/PKAP

INTEGER PORT, STAT, PRL, SHOUR, EHOUR, SEED, CASES, PERN, PERH
REAL KPROB(1:16)
COMMON/REQSTUFF/PORT, STAT, PRL, SHOUR, EHOUR, SEED, CASES,
1 PERN, PERH, KPROB

INTEGER LINPI(1:2000)
CHARACTER FLIGHT(1:2000) *8
COMMON/FLSTUFF/TOTFL, BTIME, BTYPE, LSPER, FLIGHT, LINPI

INTEGER EKDELAY(1:2000), EXDELAY(1:2000)
1 ,GREDelay(1:2000), L3DELAY(1:2000), D1DELAY(1:2000)
COMMON/DELSTUFF/EKDELAY, EXDELAY, GREDelay, L3DELAY, D1DELAY

REAL CG(1:7, 0:24), ALPHA, KAY, LC(1:7, 0:500), C(1:7, 0:500)
COMMON/COSTUFF/C, G, ALPHA, KAY, LC, C

ITNUM=4*(EHOUR-SHOUR)

JJJ=1

DO (I=1, TOTFL)
STILL(I)=.TRUE.
L3DELAY(I)=PERN+1-LSPER(I)
END DO

DO (ITER=1, ITNUM)
WRITE(9, *) ' ITERATION #', ITER
DO (F1=0, 6)
DO (F2=0, 6)
DO (F3=0, 6)
IF ((F1.GT.5).OR.(F2.GT.5).OR.(F3.GT.5)) THEN
G(F1, F2, F3)=1000000000.0
ELSE
G(F1, F2, F3)=0.0
END IF
END DO
END DO
END DO

268
CTEMP=0.0
CCTEMP=0.0

DO (J=1,TOTFL)
  DO (I=1,TOTFL)
    IF ((PRIOR(I).EQ.J).AND.(STILL(I)).AND.
1       (LSPER(I).LE.3*ITER)) THEN
  
  c find X(I,F1,F2,F3)
  WRITE(9,*)'FLIGHT',I

  IF (ITER.GT.1) THEN
    
    DO (KK=1,3*(ITER-1))
      HIGHER(KK)=0
    END DO

    DO (II=1,TOTFL)
      IF ((PRIOR(II).GT.J).AND.(.NOT.STILL(II))
1      .AND.((LSPER(II)+L3DELAY(II)).LE.3*(ITER-1)))
2      THEN
        DO (KK=1,3*(ITER-1))
          IF ((L3DELAY(II)+LSPER(II)).EQ.KK) THEN
            HIGHER(KK)=HIGHER(KK)+1
          END IF
        END DO
      END IF
    END IF
  END IF

  2200 CONTINUE
  END DO
END IF

DO (F1=0,5)
  DO (F2=0,5)
    DO (F3=0,5)
      HIGHER(3*(ITER-1)+1)=F1
      HIGHER(3*(ITER-1)+2)=F2
      HIGHER(3*ITER)=F3
      HIGHER((3*ITER)+1)=0

      IF (LSPER(I).GT.3*(ITER-1)) THEN
        ARRP=LSPER(I)
        KSTART=1
        KEND= (3*ITER)+1-LSPER(I)
      ELSE
        ARRP=3*(ITER-1)+1
        KSTART=3*(ITER-1)+2-LSPER(I)
        KEND= (3*ITER)+1-LSPER(I)
      END IF
      
      CTMP=INDCOST(I,ARRP,HIGHER,ITER)
      CALL SETFS(I,F1,F2,F3,NF1,NF2,NF3,ARRP-LSPER(I),ITER)
      CTMP=CTMP+G(NF1,NF2,NF3)
      X(I,F1,F2,F3)=KSTART-1
DO (K=KSTART,KEND)
CCTEMP=INDCOST(I, (LSPER(I)+K), HIGHER, ITER)
CALL SETFS(I, F1, F2, F3, NF1, NF2, NF3, K, ITER)
CCTEMP=CCTEMP+G(NF1, NF2, NF3)
IF (CCTEMP.LT.CTEMP) THEN
X(I,F1,F2,F3)=K
CTEMP=CCTEMP
END IF
END DO
END DO

G(F1,F2,F3)=CTEMP

ENDIF
END DO
END DO

NF1=0
NF2=0
NF3=0

DO (J=1,TOTFL)
DO (I=1,TOTFL)
IF ((P(RO(I)).EQ.TOTFL+1-J).AND.(STILL(I)).AND.
1 (LSPER(I).LE.3*ITER)) THEN
L3DELAY(I)=X(I,NF1,NF2,NF3)
F1=NF1
F2=NF2
F3=NF3
CALL SETFS(I,F1,F2,F3,NF1,NF2,NF3,L3DELAY(I),ITER)

IF((LSPER(I)+L3DELAY(I)).LT.((3*ITER)+1)) THEN
STILL(I)=FALSE.

WRITE(9,*), ' DP3 STEP=', JJJ
JJJ=JJJ+1

END IF
ENDIF
END DO
END DO
END DO

RETURN
END

**********SUB SETFS()****************
SUBROUTINE SETFS(I,F1,F2,F3,NF1,NF2,NF3,X,ITER)
INTEGER F1,F2,F3,NF1,NF2,NF3,X,F(1:3)
INTEGER LINPI (1:2000), ITER
CHARACTER FLIGHT (1:2000)*8
COMMON/FLSTUFF/TOTFL, BTIME, BTYPE, LSPER, FLIGHT, LINPI

F (1) = F1
F (2) = F2
F (3) = F3

DO (K = ((3*(ITER-1)) + 1), (3*ITER))
IF ((LSPER (I) + X) .EQ. K) THEN
F (K - (3* (ITER - 1))) = F (K - (3* (ITER - 1))) + 1
END IF
END DO

NF1 = F (1)
NF2 = F (2)
NF3 = F (3)

RETURN
END

**********FUNCTION INDCOST**************

REAL FUNCTION INDCOST (FLIGHT, ARRP, HIGHER, ITER)
INTEGER FLIGHT, ARRP, HIGHER (1:2000), ADEL (1:16), KTEMP
INTEGER OUTKAP (1:500), INKAP (1:24), LEFTOV, ITER
REAL AC (1:16)

INTEGER PORT, STAT, PERL, SHOUR, E HOUR, SEED, CASES, PERN, PERH
REAL KPROB (1:16)
COMMON/REQSTUFF/PORT, STAT, PERL, SHOUR, E HOUR, SEED, CASES,
1 PERN, PERH, KPROB

INTEGER LINPI (1:2000)
CHARACTER FLIGHT (1:2000)*8
COMMON/FLSTUFF/TOTFL, BTIME, BTYPE, LSPER, FLIGHT, LINPI

INTEGER PKAP (1:16, 1:24)
COMMON/PKAPSTUFF/PKAP

REAL CG (1:7, 0:24), ALPHA, KAY, LC (1:7, 0:500), C (1:7, 0:500)
COMMON/COSTUFF/CG, ALPHA, KAY, LC, C

HIGHER ((3*ITER) + 1) = 0
INDCOST = 0.0

DO (I = 1, CASES)
DO (J = 1, 24)
INKAP (J) = PKAP (I, J)
END DO

CALL BREACKAP (OUTKAP, INKAP)
OUTKAP ((3*ITER) + 1) = 2000
ADEL (I) = 0
KTEMP = 0
LEFTOV = 0

271
AC(I)=0.0
IF(ARRP.EQ.((3*ITER)+1)) THEN
  ADEL(I)=0
ELSE
  DO(J=1, (ARRP-1))
  IF(HIGHER(J).GT.OUTKAP(J)) THEN
    LEFTOV=LEFTOV+HIGHER(J)-OUTKAP(J)
  ELSE
    LEFTOV=MAX((LEFTOV-(OUTKAP(J)-HIGHER(J))), 0)
  ENDF
  ENDO
KTEMP=LEFTOV
DO(J=ARRP, ((3*ITER)+1);)
  IF (OUTKAP(J).GT.(HIGHER(J)+KTEMP)) THEN
    ADEL(I)=J-ARRP
    GO TO 2200
  ELSE
    KTEMP=KTEMP+HIGHER(J)-OUTKAP(J)
  ENDF
ENDO

2200 CONTINUE

IF(ARRP.GT.LSPEH(FLGHT)) THEN
  DO (K=1, (ARRP-LSPEH(FLGHT)))
  AC(I)=AC(I)+C(BTYPE(FLGHT),K)
  ENDO
ENDF

IF(ADEL(I).GT.0) THEN
  DO(K=(ARRP-LSPEH(FLGHT))+1), (ARRP-LSPEH(FLGHT)+ADEL(I)))
  AC(I)=AC(I)+(KAY*C(BTYPE(FLGHT),K))
  ENDO
ENDF

INDCOST=INDCOST+(AC(I)*KPROB(I))
END DO

RETURN
END

**********SUB GREEDY**************************

SUBROUTINE GREEDY (PI, GRCOST)
INTEGER PI(1:2000), HIGHER(1:500)
REAL ECOST(1:2000), RC, GRCOST, AIRCOST

INTEGER PKAP(1:16, 1:24)
COMMON/PKAPSTUFF/PKAP

INTEGER PORT, STAT, PERL, SHOUR, EHOUR, SEED, CASES, PERN, PERH
REAL KPROB(1:16)
COMMON/REQSTUFF/PORT, STAT, PERL, SHOUR, EHOUR, SEED, CASES,
1 PERN, PERH, KPROB

INTEGER LINPI(1:2000)
CHARACTER FLIGHT(1:2000)*8
COMMON/FLSTUFF/TOTFL, BTIME, BTYPE, LSPER, FLIGHT, LINPI

INTEGER EKDELAY(1:2000), EXDELAY(1:2000)
1, GRDELAY(1:2000), L3DELAY(1:2000), L1DELAY(1:2000)
COMMON/DELSTUFF/EKDELAY, EXDELAY, GRDELAY, L3DELAY, L1DELAY

REAL CG(1:7,0:24), ALPHA, KAY, LC(1:7,0:500), C(1:7,0:500)
COMMON/COSTUFF/C, ALPHA, KAY, LC, C

DO (K=1, (PERN+1))
   HIGHER(K)=0
END DO

GRCOST=0.0

DO (J=1, TOTFL)
   DO (I=1, TOTFL)
      IF (PI(I).EQ.(TOTFL+1-J)) THEN
         ECOST(I)=AIRCOST(I, LSPER(I), HIGHER)
         GRDELAY(I)=0
         DO (K=1, (PERN+1-LSPER(I)))
            RC=AIRCOST(I, (LSPER(I)+K), HIGHER)
            IF (RC.LT.ECOST(I)) THEN
               GRDELAY(I)=K
               ECOST(I)=RC
            ENDIF
         END DO
         HIGHER(LSPER(I)+GRDELAY(I))=HIGHER(LSPER(I)+GRDELAY(I))+1
         GRCOST=GRCOST+ECOST(I)
      ENDIF
   END DO
END DO

RETURN
END

************FUNCTION AIRCOST***************
REAL FUNCTION AIRCOST(FLIGHT, ARRP, HIGHER)
INTEGER FLIGHT, ARRP, HIGHER(1:2000), ADEL(1:16), KTEMP
INTEGER OUTKAP (1:500), INKAP (1:24), LEFTOV
REAL AC (1:16)

INTEGER PORT, STAT, PERL, SHOUR, EHOUR, SEED, CASES, PERN, PERH
REAL KPROB(1:16)
COMMON/REQSTUFF/PORT, STAT, PERL, SHOUR, EHOUR, SEED, CASES,
1 PERN, PERH, KPROB

INTEGER LINPI (1:2000)
CHARACTER FLIGHT (1:2000)*8
COMMON/FLSTUFF/TOTFL, BTIME, BTYPE, LSPER, FLIGHT, LINPI

INTEGER PKAP(1:16, 1:24)
COMMON/PKAPSTUFF/PKAP
REAL CG(1:7, 0:24), ALPHA, KAY, LC(1:7, 0:500), C(1:7, 0:500)
COMMON/COSTUFF/CG, ALPHA, KAY, LC, C

HIGHER(PERN+1)=0
AIRCOST=0.0

DO (I=1, CASES)
  DO (J=1, 24)
    INKAP(J)=PKAP(I, J)
  END DO

CALL BREACKAP(OUTKAP, INKAP)
OUTKAP(PERN+1)=2000
ADEL(I)=0
KTEMP=0
LEFTOV=0
AC(I)=0.0
IF (ARRP.EQ. (PERN+1)) THEN
  ADEL(I)=0
ELSE

  DO (J=1, (ARRP-1))
    IF (HIGHER(J).GT.OUTKAP(J)) THEN
      LEFTOV=LEFTOV+HIGHER(J)-OUTKAP(J)
    ELSE
      LEFTOV=MAX((LEFTOV-(OUTKAP(J)-HIGHER(J))), 0)
    ENDIF
  END DO

  KTEMP=LEFTOV
  DO (J=ARRP, (PERN+1))
    IF (OUTKAP(J).GT.(HIGHER(J)+KTEMP)) THEN
      ADEL(I)=J-ARRP
      GO TO 2200
    ELSE
      KTEMP=KTEMP+HIGHER(J)-OUTKAP(J)
    ENDIF
  END DO
  CONTINUE

2200
IF (ARRP.GT.LS PER(FLGHT)) THEN
  DO (K=1, (ARRP-LS PER(FLGHT)))
    AC(I)=AC(I)+C(BTYPE(FLGHT), K)
  END DO
ENDIF

IF (ADEL(I).GT.0) THEN
  DO (K=(ARRP-LS PER(FLGHT)+1), (ARRP-LS PER(FLGHT)+ADEL(I)))
    AC(I)=AC(I)+(KAY*C(BTYPE(FLGHT), K))
  END DO
ENDIF

AIRCOST=AIRCOST+(AC(I)*KPROB(I))
END DO
RETURN
END

******************************************************************************
SUBROUTINE ALPI
******************************************************************************
INTEGER MTEMP, ALPHAP(1:2000), IDUMMY(1:2000)
LOGICAL STILL(1:2000)
REAL TAR(1:2000)
CHARACTER ANS*1

INTEGER LINPI(1:2000)
CHARACTER FLIGHT(1:2000)*8
COMMON/FLSTUFF/TOTFL, BTIME, BTYPE, LSPER, FLIGHT, LINPI

INTEGER PORT, STAT, PRL, SHOUR, EHOUR, SEED, CASES, PERN, PERH
REAL KPROB(1:16)
COMMON/REALSTUFF/PORT, STAT, PRL, SHOUR, EHOUR, SEED, CASES,
1 PERN, PERH, KPROB

REAL CG(1:7, 0:24), ALPHA, KAY, LC(1:7, 0:500), C(1:7, 0:500)
COMMON/COSTUFF/CGL, ALPHA, KAY, LC, C

DO (I=1, TOTFL)
   TAR(I)=C(BTYPE(I), PERN+1-LSPER(I))
   STILL(I)=.TRUE.
END DO

CALL RRSORT(TAR, TOTFL)

MTEMP=1
DO (J=1, TOTFL) !FOR EACH PRIORITY
   DO (I=1, TOTFL) !FOR EACH FLIGHT
      IF (STILL(I)) THEN
         IF (C(BTYPE(I), PERN+1-LSPER(I)).EQ.TAR(TOTFL+1-MTEMP)) THEN
            ALPHAP(I)=TOTFL+1-MTEMP
            MTEMP=MTEMP+1
            STILL(I)=.FALSE.
         END IF
      END IF
   END DO
END DO

RETURN
END

******************************************************************************
FUNCTION AGCOST
******************************************************************************
REAL FUNCTION AGCOST(GDELAY)
INTEGER GDELAY(1:2000), ADELAY(1:2000)
INTEGER TEMPKAP(1:24), NEWLSPER(1:2000)
REAL COST, GCOST, ACOST, AACOST
INTEGER LINPI (1:2000)
CHARACTER FLIGHT (1:2000) * 8
COMMON/FLSTUFF/TOTFL, BTIME, BTYPE, LSUPER, FLIGHT, LINPI

INTEGER PORT, STAT, PERL, SHOUR, EHOUR, SEED, CASES, PERN, PERH
REAL KPROB (1:16)
COMMON/REQSTUFF/PORT, STAT, PERL, SHOUR, EHOUR, SEED, CASES,
PERN, PERH, KPROB

REAL CG (1:7, 0:24), ALPHA, KAY, LC (1:7, 0:500), C (1:7, 0:500)
COMMON/COSTUFF/CG, ALPHA, KAY, LC, C

INTEGER KAP (1:24)
COMMON/KAPSTUFF/KAP

INTEGER PKAP (1:16, 1:24)
COMMON/PKAPSTUFF/PKAP

AGCOST = 0.0
ACOST = 0.0
AACOST = 0.0
GCOST = COST (GDELAY)

DO (J=1, TOTFL)
NEWLSPER (J) = LSPER (J) + GDELAY (J)
END DO

DO (I=1, CASES)
DO (K=1, 24)
TEMPKAP (K) = PKAP (I, K)
END DO
CALL BPOL (ADELAY, TEMPKAP, NEWLSPER)

DO (II=1, TOTFL)
IF (.NOT. ADELAY (II).GT. 0) THEN
DO (K= (GDELAY (II)+1), (GDELAY (II)+ADELAY (II))
AACOST = AACOST + C (BTYPE (II), K)
END DO
ENDIF
END DO

AACOST = AACOST * KPROB (I)
ACOST = ACOST + AACOST
AACOST = 0.0
END DO
AGCOST = (KAY*ACOST) + GCOST

RETURN
END

FUNCTION PICOST
REAL FUNCTION PICOST (GDELAY, PI)
INTEGER GDELAY (1:2000), ADELAY (1:2000), PI (1:2000), IPER
INTEGER TEMPKAP (1:24), RESIKAP (1:500), NEWLSPER (1:2000)
REAL COST, GCOST, ACOST, AACOST
INTEGER LINPI (1:2000)
 CHARACTER FLIGHT (1:2000)*8
COMMON/FLSTUFF/TOTFL, BTIME, BTYPE, LSPER, FLIGHT, LINPI

INTEGER PORT, STAT, PERL, SHOUR, EHOUR, SEED, CASES, PERN, PERH
REAL KPROB (1:16)
COMMON/REQSTUFF/PORT, STAT, PERL, SHOUR, EHOUR, SEED, CASES,
PERN, PERH, KPROB

REAL CG (1:7, 0:24), ALPHA, KAY, LC (1:7, 0:500), C (1:7, 0:500)
COMMON/COSTUFF/CG, ALPHA, KAY, LC, C

INTEGER KAP (1:24)
COMMON/KAPSTUFF/KAP

INTEGER PKAP (1:16, 1:24)
COMMON/PKAPSTUFF/PKAP

PICOST = 0.0
ACOST = 0.0
AACOST = 0.0
GCOST = COST (GDELAY)

DO (I = 1, CASES)
  DO (J = 1, TOTFL)
    ADELAY (J) = PERN + 1 - (LSPER (J) + GDELAY (J))
  END DO

  DO (J = 1, 24)
    TEMPKAP (J) = PKAP (I, J)
  END DO

  CALL BREACKAP (RESIKAP, TEMPKAP)

RESIKAP (PERN + 1) = 2000
DO (K = 1, TOTFL)
  DO (J = 1, TOTFL)
    IF (PI (J) .EQ. (TOTFL + 1 - K)) THEN
      DO (IPER = (LSPER (J) + GDELAY (J)), (PERN + 1))
        PERIODS AFTER LSPER
        IF (RESIKAP (IPER) .GT. 0) THEN
          ADELAY (J) = IPER - (LSPER (J) + GDELAY (J))
          RESIKAP (IPER) = RESIKAP (IPER) - 1
          GO TO 3000
        ENDIF
      END DO
    END IF
  END DO
END IF

3000 CONTINUE !FLIGHTS J

277
END DO

DO (J=1, TOTFL)
  IF (ADELAY(J) .GT. 0) THEN
    DO (KK=(GDELAY(J)+1), (GDELAY(J)+ADELAY(J)))
      AACOST=AACOST+C(BTYPE(J), KK)
    END DO
  ENDIF
END DO

AACOST=AACOST*KPROB(I)
ACOST=ACOST+AACOST
AACOST=0.0
END DO

PICOST=GCOST+ (KAY*ACOST)

RETURN
END

SUBROUTINE BREACKAP(OUTKAP, INKAP)
INTEGER OUTKAP(1:500), INKAP(1:24), IND(1:12)
INTEGER OHOUR, NEXT, GAMMA

INTEGER PORT, STAT, PERL, SHOUR, EHOUR, SEED, CASES, PERN, PERH
REAL KPROB(1:16)
COMMON/REQSTUFF/PORT, STAT, PERL, SHOUR, EHOUR, SEED, CASES,
1 PERN, PERH, KPROB

OHOUR=SHOUR*100
NEXT=1
DO 2020 I=1, (EHOUR-SHOUR)
  GAMMA=0
  DO 2030 J=NEXT, NEXT+PERH-1
    OUTKAP(J)=INT ((INKAP(I)/PERH)+0.001)
    GAMMA=GAMMA+OUTKAP(J)
  2030 CONTINUE
  NEXT=NEXT+PERH
IF (GAMMA.LT. INKAP(I)) THEN
  IF (PERL.EQ.5) THEN
    IND (1)=1; IND (2)=7; IND (3)=4; IND (4)=10
    IND (5)=2; IND (6)=8; IND (7)=5; IND (8)=11
    IND (9)=3; IND (10)=9; IND (11)=6; IND (12)=12
  ELSEIF (PERL.EQ.10) THEN
    IND (1)=1; IND (2)=4; IND (3)=2
    IND (4)=5; IND (5)=3; IND (6)=6
  ELSEIF (PERL.EQ.15) THEN
    IND (1)=1; IND (2)=3; IND (3)=2; IND (4)=4
  ELSEIF ((PERL.EQ.20).OR. (PERL.EQ.30).OR. (PERL.EQ.60)) THEN
    DO 2032 L=1, PERH
      IND(L)=L
  2032 CONTINUE
  ELSE
    WRITE(9,*),'ERROR IN PERIOD LENGTH INPUT'
    WRITE(9,*),'MUST BE ONE OF 5,10,15,20,30 or 60'
ENDIF
}
PAUSE
ENDIF
DO 2031 K=1, (INKAP(I)-GAMMA)
       OUTKAP(((I-1)*PERH)+IND(K))=OUTKAP(((I-1)*PERH)+IND(K))+1
2031 CONTINUE
ENDIF
2020 CONTINUE

RETURN
END

******************************************************************************
SUBROUTINE EKPOL******************************************************************************
INTEGER EKAP(1:24)
REAL SKAP(1:24)
CHARACTER PKAPFILE*4, INTFILE*10, CI*2

INTEGER PKAP(1:16,1:24)
COMMON/PKAPSTUFF/PKAP

INTEGER PORT, STAT, PERL, SHOUR, EHOUR, SEED, CASES, PERN, PERH
REAL KPROB(1:16)
COMMON/REQSTUFF/PORT, STAT, PERL, SHOUR, EHOUR, SEED, CASES,
1 PERN, PERH, KPROB

INTEGER LINPI(1:2000)
CHARACTER FLIGHT(1:2000)*8
COMMON/FLSTUFF/TOTFL, BTIME, BTYPE, LSPE, FLIGHT, LINPI

INTEGER EKDELAY(1:2000), EXDELAY(1:2000)
1 ,GRDELAY(1:2000), L3DELAY(1:2000), L1DELAY(1:2000)
COMMON/DELSSTUFF/EKDELAY, EXDELAY, GRDELAY, L3DELAY, L1DELAY

DO (I=1, CASES)

IF (I.GT.10) THEN
       WRITE(INTFILE, '(I2)') I
       READ(INTFILE, '(A2)') CI
       PKAPFILE='KAP'/CI
ELSE
       WRITE(INTFILE, '(I1)') I
       READ(INTFILE, '(A1)') CI
       PKAPFILE='KAP'/CI
ENDIF

OPEN(UNIT=18, FILE=PKAPFILE, FORM='FORMATTED', STATUS='OLD',
1 ACCESS='SEQUENTIAL')
DO (J=1, SHOUR)
       READ(18, '(I4)') IDUMMY
       END DO
       DO (J=1, (EHOUR-SHOUR))
         READ(18, '(I4)') PKAP(I,J)
       END DO
       CLOSE(18)
       END DO
DO (J=1, (EHOUR-SHOUR))
SKAP(J)=0.0
DO (I=1,CASES)
SKAP(J)=SKAP(J)+(REAL(PKAP(I,J))*KPROB(I))
END DO
EKAP(J)=NINT(SKAP(J))
WRITE(9,*) J, PKAP(1,J), PKAP(2,J), PKAP(3,J), EKAP(J)
END DO
CALL BPOL(EKDELAY, EKAP, LSPER)
RETURN
END

**************************************************************************
SUBROUTINE EXPOL
**************************************************************************
REAL RDEL(1:2000)
INTEGER TEMPKAP(1:24), TEMPDEL(1:2000)

INTEGER PKAP(1:16,1:24)
COMMON/PKAPSTUFF/PKAP

INTEGER PORT, STAT, PERL, SHOUR, EHOUR, SEED, CASES, PERN, PERH
REAL KPROB(1:16)
COMMON/REQSTUFF/PORT, STAT, PERL, SHOUR, EHOUR, SEED, CASES, PERN, PERH, KPROB

INTEGER LINPI(1:2000)
CHARACTER FLIGHT(1:2000)*8
COMMON/FLSTUFF/TOTFL, BTIME, BTYPE, LSPER, FLIGHT, LINPI

INTEGER EKDELAY(1:2000), EXDELAY(1:2000)
1 ,GRDELAY(1:2000), L3DELAY(1:2000), L1DELAY(1:2000)
COMMON/DELSTUFF/EKDELAY, EXDELAY, GRDELAY, L3DELAY, L1DELAY

DO (J=1,TOTFL)
EXDELAY(J)=0
RDEL(J)=0.0
END DO

DO(I=1,CASES)
 DO (J=1,24)
 TEMPKAP(J)=PKAP(I,J)
 END DO
 CALL BPOL(TEMPDEL, TEMPKAP, LSPER)
 DO (J=1,TOTFL)
 RDEL(J)=RDEL(J)+(REAL(TEMPDEL(J))*KPROB(I))
 END DO
 END DO

DO (J=1,TOTFL)
EXDELAY (J) = NINT (RDEL (J))
END DO

RETURN
END

*******************************************************************************
SUB LICOST******************************************************************************

REAL FUNCTION LICOST (DELAY)

INTEGER LINPI (1:2000)
CHARACTER FLIGHT (1:2000) * 8
COMMON / FLSTUFF / TOTFL, BTIME, BTYPE, LSPER, FLIGHT, LINPI

REAL CG (1:7, 0:24), ALPHA, KAY, LC (1:7, 0:500), C (1:7, 0:500)
COMMON / COSTUFF / CG, ALPHA, KAY, LC, C

LICOST = 0.0
DO 20 I = 1, TOTFL
   IF (DELAY (I) .GT. 0) THEN
      DO 21 K = 1, DELAY (I)
         LICOST = LICOST + LC (BTYPE (I), K)
      21 CONTINUE
   ENDIF
20 CONTINUE

RETURN
END

*******************************************************************************
SUB COST******************************************************************************

REAL FUNCTION COST (DELAY)

INTEGER DELAY (1:2000)
CHARACTER FLIGHT (1:2000) * 8
COMMON / FLSTUFF / TOTFL, BTIME, BTYPE, LSPER, FLIGHT

REAL CG (1:7, 0:24), ALPHA, KAY, LC (1:7, 0:500), C (1:7, 0:500)
COMMON / COSTUFF / CG, ALPHA, KAY, LC, C

COST = 0.0
DO 20 I = 1, TOTFL
   IF (DELAY (I) .GT. 0) THEN
      DO 21 K = 1, DELAY (I)
         COST = COST + C (BTYPE (I), K)
      21 CONTINUE
   ENDIF
20 CONTINUE

RETURN
END
*************** subroutine BPOL FOR PROBABILISTIC CASE****

SUBROUTINE BPOL(BDELAY, BKAP, BLSPER)

CHARACTER INFILE*10, APORT*3, ANS*1
INTEGER PERKAP(1:500), ACOUNT, IND(1:12), GAMMA, L
INTEGER NEXT, OHHOUR, TSTART(1:500), TEND(1:500), n TEMP

INTEGER TEMP, NE, K, OLPER(1:2000), BDELAY(1:2000)
REAL TEMPCO(1:2000), TRESH

INTEGER BLSPER(1:2000), BKAP(1:24)

INTEGER LINPI(1:2000)
CHARACTER FLIGHT(1:2000)*8
COMMON/FLSTUFF/TOTFL, BTIME, BTYPE, LS PER, FLIGHT, LINPI

INTEGER PORT, STAT, PRL, SHOUR, EHHOUR, SEED, CASES, PERN, PERNH
REAL KPROB(1:16)
COMMON/REQSTUFF/PORT, STAT, PRL, SHOUR, EHHOUR, SEED, CASES,
1 PERN, PERNH, KPROB

REAL CG(1:7, 0:24), ALPHA, KAY, LC(1:7, 0:500), C(1:7, 0:500)
COMMON/COSTUFF/CG, ALPHA, KAY, LC, C

REAL TOTNUM, PERBOS, PERLGA, PERORD, HPER(1:24)
COMMON/DEMSTUFF/TOTNUM, PERBOS, PERLGA, PERORD, HPER

INTEGER TYPAR(1:7)
COMMON/FMIXSTUFF/TYPAR

CALL BREAKAP(PERKAP, BKAP)

DO 55 I=1, TOTFL
OLPER(I)=PERN+1
55 CONTINUE

DO 66 J=1, PERN
K=0
TEMP=0

DO 65 I=1, TOTFL

IF ((BLSPER(I).LE.J).AND.(OLPER(I).GE.J)) THEN
K=K+1
TEMPCO(K)=C(BTYPE(I),J-LS PER(I)+1)
ENDIF
65 CONTINUE

CALL RRSORT(TEMPCO, K)

IF (PERKAP(J).GE.K) THEN
DO 64 I=1, TOTFL
IF((BLSPER(I).LE.J).AND.(OLPER(I).GE.J)) THEN

282
OLPER(I)=J  
ENDIF  
64  CONTINUE  
ELSE  
TRESH=TEMPCO(K-PERKAP(J))  
DO 63 I=1,TOTFL  
   IF((BLSPER(I).LE.J).AND.(OLPER(I).GE.J)) THEN  
      IF(C(BTYPE(I),J+1-LSPER(I)).GT.TRESH) THEN  
         OLP(I)=J  
         TEMP=TEMP+1  
      ENDIF  
   ENDIF  
63  CONTINUE  
DO 62 I=1,TOTFL  
   IF((BLSPER(I).LE.J).AND.(OLPER(I).GE.J)) THEN  
      IF(C(BTYPE(I),J+1-LSPER(I)).EQ.TRESH).AND.  
         (TEMP.LT.PERKAP(J)) THEN  
         OLP(I)=J  
         TEMP=TEMP+1  
      ENDIF  
   ENDIF  
62  CONTINUE  
ENDIF  
66  CONTINUE  
DO 78 I=1,TOTFL  
   BDELAY(I)=PERN+1-BLSPER(I)  
78  CONTINUE  
DO 77 J=1,PERN  
   DO 76 I=1,TOTFL  
      IF (OLPER(I).EQ.J) THEN  
         BDELAY(I)=J-BLSPER(I)  
      ENDIF  
76  CONTINUE  
77  CONTINUE  
RETURN  
END  

**************************************************************************function RANDI LOWER,UPPER)************

INTEGER FUNCTION RANDI LOWER,UPPER)  
INTEGER LOWER,UPPER  
REAL RANDU  
RANDI = LOWER+INT((RANDU()) * (UPPER-LOWER+1))  
RETURN  

283
END

function RANDS(START)
REAL FUNCTION RANDS (START)
INTEGER START
REAL RANDU
INTEGER L,C,M
PARAMETER (L=1029,C=221591,M=1048576)
INTEGER SEED
SAVE SEED
DATA SEED / 10/

SEED = MOD(ABS(START),M)

function RANDU() ENTRY*****
ENTRY RANDU()
SEED = MOD(SEED*L+C,M)
RANDU = REAL(SEED)/M
RETURN
END

subroutine RANTY(*,*)
INTEGER FUNCTION RANTY(CATPER,CATNUM)
INTEGER CATPER(1:*),CATNUM,S,I,J,ELEM(1:100),Z
REAL RANDU

S=0
DO 333 I=1,CATNUM
   DO 334 J=1,CATPER(I)
      ELEM(S+J)=I
   CONTINUE
   S=S+CATPER(I)
333 CONTINUE
334 CONTINUE
Z=1+INT(100*RANDU())
RANTY=ELEM(Z)
RETURN
END

SUBROUTINE RRSORT
REAL TT(1:*)
INTEGER N,SIZET,T1

DO 333 NPASS = 1,SIZET-1
   DO 334 N = T1,NPASST-SIZET
      IF (TT(N).GT.TT(N+1)) THEN
         TEMP=TT(N)
         TT(N)=TT(N+1)
         TT(N+1)=TEMP
      ENDIF
   CONTINUE
334 CONTINUE
333 CONTINUE
RETURN
END

**********SUBROUTINE ARSORT(*,*)**********

SUBROUTINE ARSORT(TT,SIZETT)
INTEGER TT(1:*), TEMP, N, SIZETT, NPASS

DO 333 NPASS = 1, SIZETT-1
   DO 334 N = 1, SIZETT-NPASS
      IF (TT(N) .GT. TT(N+1)) THEN
         TEMP = TT(N)
         TT(N) = TT(N+1)
         TT(N+1) = TEMP
      ENDIF
   CONTINUE
334 CONTINUE
333 CONTINUE

RETURN
END

C ********** $$$$ BLOCK-DATA $$$$ **********

BLOCK DATA
C

INTEGER LINPI(1:2000)
CHARACTER FLIGHT(1:2000)*8
COMMON/FLSTUFF/TOTFL,BTIME,BTYPE,LSPER,FLIGHT,LINPI

CHARACTER*64 DEMFILE, CAPFILE, COSTFILE, FMIXFILE
COMMON/FILESTUFF/DEMFILE, CAPFILE, COSTFILE, FMIXFILE

INTEGER PORT, STAT, PERL, SHOUR, EHOUR, SEED, CASES, PERN, PERH
REAL KPROB(1:16)
COMMON/REQSTUFF/PORT, STAT, PERL, SHOUR, EHOUR, SEED, CASES,
1 PERN, PERH, KPROB

REAL CG(1:7,0:4),ALPHA, KAY, LC(1:7,0:500), C(1:7,0:500)
COMMON/COSTUFF/CG, ALPHA, KAY, LC, C

INTEGER KAP(1:24)
COMMON/KAPSTUFF/KAP

REAL TOTNUM, PERBOS, PERLGA, PERORD, HPER(1:24)
COMMON/DEMSSTUFF/TOTNUM, PERBOS, PERLGA, PERORD, HPER

INTEGER TYPAR(1:7)
COMMON/FMIXSTUFF/TYPAR

INTEGER PKAP(1:16,1:24)
COMMON/PKAPSTUFF/PKAP

INTEGER EKDELYA(1:2000), EXDELYA(1:2000)
1, GRDELYA(1:2000), L3DELYA(1:2000), L1DELYA(1:2000)
COMMON/DELSTUFF/EKDELY, EXDELY, GRDELY, L3DELY, L1DELY

DATA DEMFILE / 'BOSARR' /
DATA CAPFILE / 'BOSKAP'/
DATA COSTFILE / 'COSTS'/
DATA FMIXFILE / 'TYPFIL'/

END

************************************************** subroutine PFCFS **************************************************

SUBROUTINE PFCFS
INTEGER EKAP(1:24), PERKAP(1:500)
REAL SKAP(1:24)
CHARACTER PKAPFILE*4, INTFILE*10, CI*2
LOGICAL STILL(1:2000)
INTEGER PKAP(1:16,1:24)
COMMON/PKAPSTUFF/PKAP

INTEGER PORT, STAT, PERL, SHOUR, EHOUR, SEED, CASES, PERN, PERH
REAL KPROB(1:16)
COMMON/REQSTUFF/PORT, STAT, PERL, SHOUR, EHOUR, SEED, CASES,
1 PERN, PERH, KPROB

 INTEGER LINPI(1:2000)
CHARACTER FLIGHT(1:2000)*8
COMMON/FLSTUFF/TOTFL, BTIME, BTYPE, LSPER, FLIGHT, LINPI

 INTEGER EKDELAY(1:2000), TRDELAY(1:2000)
1, GRDELAY(1:2000), CFDELAY(1:2000), WKDELAY(1:2000)
COMMON/DELSTUFF/EKDELAY, TRDELAY, GRDELAY, CFDELAY, WKDELAY

DO (I=1,CASES)

 IF (I.GT.10) THEN
 WRITE(INTFILE, '(I2)') I
 READ(INTFILE, '(A2)') CI
 PKAPFILE='KAP'//CI
 ELSE
 WRITE(INTFILE, '(I1)') I
 READ(INTFILE, '(A1)') CI
 PKAPFILE='KAP'//CI
 END IF

OPEN(UNIT=18, FILE=PKAPFILE, FORM='FORMATTED', STATUS='OLD',
1 ACCESS='SEQUENTIAL')
 DO (J=1, SHOUR)
 READ(18 , '(I4)') DUMMY
 END DO
 DO (J=1, (EHOUR-SHOUR))
 READ(18, '(I4)') PKAP(I,J)
 END DO
 CLOSE(18)
 END DO
DO (J=1, (E_HOUR-S_HOUR))
SKAP(J)=0.0
DO (I=1, CASES)
    SKAP(J)=SKAP(J)+(REAL(PKAP(I,J))*KPROB(I))
END DO
EKAP(J)=NINT(SKAP(J))
END DO

CALL BREACKAP(PERKAP, EKAP)

DO (I=1, TOTFL)
    STILL(I)=.TRUE.
    CFDELAY(I)=PERN+1-LS PER(I)
END DO

DO (J=1, PERN)
    K=0
    DO (I=1, TOTFL)
        IF ( (STILL(I)).AND.(LS PER(I).LE.J)
           .AND.(K.LT.PERKAP(J)). THEN
            CFDELAY(I)=J-LS PER(I)
            STILL(I)=.FALSE.
            K=K+1
        END IF
    END DO
END DO

RETURN
END

FUNCTION FCOST()
REAL FUNCTION FCOST(GDELAY)
INTEGER GDELAY(1:2000), ADELAY(1:2000), PI (1:2000), IPER
INTEGER TEMPKAP(1:24), RESIKAP(1:500), NEWLS PER(1:2000)
REAL COST, GCOST, ACOST, AACOST

INTEGER LINPI(1:2000)
CHARACTER FLIGHT(1:2000)*8
COMMON/FLSTUFF/TOTFL, BTIME, BTYPE, LS PER, FLIGHT, LINPI

INTEGER PORT, STAT, PERL, SHOUR, E_HOUR, SEED, CASES, PERN, PERH
REAL KPROB(1:16)
COMMON/REQSTUFF/PORT, STAT, PERL, SHOUR, E_HOUR, SEED, CASES,
1 PERN, FNRH, KPROB

REAL CG(1:7,0:24), ALPHA, ALPHAL, KAY, LC(1:7,0:500), C(1:7,0:500)
COMMON/COSTUFF/CG, ALPHA, ALPHAL, KAY, LC, C

INTEGER KAP(1:24)
COMMON/KAPSTUFF/KAP

INTEGER PKAP(1:16,1:24)
COMMON/PKAPSTUFF/PKAP

FCOST = 0.0
ACOST = 0.0
AACOST = 0.0
GCOST = COST (GDELAY)

C THIS PART IS ADDED FROM PICOST.
C WE COMPUTE THE LANDING PRIORITY PI THAT CORRESPONDS TO FCFS
C ACCORDING TO GDELAY AND BTIME (WHICH IS WHAT WE WANT) SINCE
C THE FLIGHT INDEX ALREADY REFLECTS BTIME

K=TOTFL
DO (I=1,TOTFL)
   DO (J=1,(PERN+1))
      IF ( (LSPER(I)+GDELAY(I)).EQ.J ) THEN
         PI(I)=K
         K=K-1
      END IF
   END DO
END DO

C NOW WE RETURN TO THE CODE OF PICOST (OF COURSE WE CHANGE PICOST TO
FCOST)

DO (I=1,CASES)
   DO (J=1,TOTFL)
      ADELAY(J)=PERN+1-(LSPER(J)+GDELAY(J))
   END DO
   DO (J=1,24)
      TEMPKAP(J) = PKAP(I,J)
   END DO
   CALL BREATKAP (RESIKAP,TEMKPAP)
   RESIKAP(PERN+1)=2000
   DO (K=1,TOTFL)
      DO (J=1,TOTFL)
         IF (PI(J).EQ. (TOTFL+1-K)) THEN
            DO(IPER=(LSPER(J)+GDELAY(J)),(PERN+1))
               PERIODS AFTER LSPER
               IF (RESIKAP(IPER).GT.0) THEN
                  ADELAY(J)=IPER- (LSPER(J)+GDELAY(J))
                  RESIKAP(IPER)=RESIKAP(IPER)-1
                  GO TO 3000
               ENDIF
            END DO
         END IF
      END DO
   END DO
END DO

3000 CONTINUE
END DO
!PI'S K

DO (J=1,TOIL)
IF (ADELAY(J).GT.0) THEN
DO (KK=(GDELAY(J)+1), (GDELAY(J)+ADELAY(J)))
AACOST=AACOST+C(BTYPE(J),KK)
END DO
ENDIF
END DO

AACOST=AACOST*KPROB(I)
ACOST=ACOST+AACOST
AACOST=0.0
END DO

!CASES I

FCOST=GCOST+(KAY*ACOST)
RETURN
END

***********************************************************************LANPI********************************************************

SUBROUTINE LANPI(LANPrio)
INTEGER MTEMP,LANPrio(1:2000),IDUMMY(1:2000),X
LOGICAL STILL(1:2000)
REAL TAR(1:2000),FACTOR,THETAL

INTEGER LINPI(1:2000)
CHARACTER FLIGHT(1:2000)*8
COMMON/FLSTUFF/TOTFL,BTIME,BTYPE,LSPER,FLIGHT,LINPI

INTEGER PORT,STAT,PERL,SHOUR,EHOUR,SEED,CASES,PERN,PERH
REAL KPROB(1:16)
COMMON/REQSTUFF/PORT,STAT,PERL,SHOUR,EHOUR,SEED,CASES,
1 PERN,PERH,KPROB

REAL CG(1:7,0:24),ALPHA,ALPHAL,KAY,LC(1:7,0:500),C(1:7,0:500)
COMMON/COSTUFF/CG,ALPHA,ALPHAL,KAY,LC,C

THETAL=((1+ALPHAL)**(REAL(PERL)/60.0)) -1
WRITE(9,*) ' THETAL = ', THETAL

DO (I=1,TOTFL)
X=PERN+1-LSPER(I)
FACTOR=1+THETAL**X
TAR(I)=FACTOR*C(BTYPE(I),X)
STILL(I)=.TRUE.
END DO

CALL RRSORT(TAR,TOTFL)

MTEMP=1
DO (J=1,TOTFL) !FOR EACH PRIORITY
DO(I=1,TOTFL) !FOR EACH FLIGHT

289
IF (STILL(I)) THEN
    X=PERN+1-LSPER(I)
    FACTOR=(1+THETAL)**X
    IF((FACTOR*C(BTYPE(I),X)).EQ.1) THEN
        TAR(TOTFL+1-MTEMP)) THEN
        LANPrio(I)=TOTFL+1-MTEMP
        MTEMP=MTEMP+1
        STILL(I)=.FALSE.
    END IF
END IF
END DO
END DO

RETURN
END
REFERENCES


