AN AIRPORT GATE REASSIGNMENT PROBLEM WITH TWO CRITERIA

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

DURSEN DENİZ POYRAZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
INDUSTRIAL ENGINEERING

JULY 2022

Approval of the thesis:

AN AIRPORT GATE REASSIGNMENT PROBLEM WITH TWO CRITERIA

submitted by **DURSEN DENİZ POYRAZ** in partial fulfillment of the requirements for the degree of **Master of Science** i**n Industrial Engineering, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Esra Karasakal
Head of the Department, **Industrial Engineering** _____

Prof. Dr. Meral Azizoğlu
Supervisor, **Industrial Engineering, METU** _____

Examining Committee Members:

Prof. Dr. Gülser Köksal
Industrial Engineering, METU _____

Prof. Dr. Meral Azizoğlu
Industrial Engineering, METU _____

Prof. Dr. Ferda Can Çetinkaya
Industrial Engineering, Çankaya University _____

Assist. Prof. Dr. Banu Yüksel Özkaya
Industrial Engineering, Hacettepe University _____

Assoc. Prof. Dr. Mehmet Rüştü Taner
Industrial Engineering, TED University _____

Date: 05.07.2022

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name Last name : Dursen Deniz Poyraz

Signature :

ABSTRACT

AN AIRPORT GATE REASSIGNMENT PROBLEM WITH TWO CRITERIA

Poyraz, Dursen Deniz
Master of Science, Industrial Engineering
Supervisor : Prof. Dr. Meral Azizoğlu

July 2022, 79 pages

In this study, we consider an airport gate reassignment problem where the aircraft are already assigned to the gates and a disruption occurs at some of the gates. After the disruption, the aircraft are reassigned to the gates considering efficiency and stability measures. Our efficiency criterion focuses on the maximum utilization of the gates in terms of both the number of aircraft and the number of passengers in these aircraft. On the other hand, our stability criterion is concerned with remaining as close to the initial plan as possible.

We propose solution approaches for generating two extreme, extreme supported, and all nondominated objective vectors with respect to our efficiency and stability criteria. We also present an optimal decomposition rule that reduces the complexity of the solution. Our extensive experiments have shown the satisfactory behavior of our solution algorithms.

Keywords: Airport Gate Reassignment Problem, Nondominated Objective Vectors, Mixed Integer Linear Programming

ÖZ

İKİ KRİTERLİ HAVALİMANI KAPISI YENİDEN ATAMA PROBLEMİ

Poyraz, Dursen Deniz
Yüksek Lisans, Endüstri Mühendisliği
Tez Yöneticisi: Prof. Dr. Meral Azizoğlu

Temmuz 2022, 79 sayfa

Bu çalışmada, uçakların kapılara atandığı ve bazı kapılarda aksama olan bir havalimanı kapısı yeniden atama problemini ele alıyoruz. Arızanın akabinde, uçaklar verimlilik ve stabilite ölçüleri dikkate alınarak yeniden atanmaktadır. Verimlilik kriterimiz, uçak sayısı ve uçaklardaki yolcu sayısına göre kapıların maksimum kullanımı üzerine odaklanmaktadır. Öte yandan, stabilite kriterimiz ilk plana mümkün olduğunca yakın kalmayı dikkate almaktadır.

Verimlilik ve stabilite kriterlerimize göre iki ekstrem, destekli ekstrem ve tüm baskın amaç vektörleri üretmek için çözüm yaklaşımları öneriyoruz. Ayrıca çözümlerin karmaşıklığını azaltan bir optimal ayrıştırma kuralı sunuyoruz. Kapsamlı deneylerimiz, çözüm algoritmalarımızın başarılı sonuçlar verdiğini göstermektedir.

Anahtar Kelimeler: Havalimanı Kapısı Yeniden Atama Problemi, Baskın Amaç Vektörleri, Karışık Tamsayı Lineer Programlama

To my mother Gonca and my father İbrahim…

ACKNOWLEDGMENTS

I would like to take this opportunity to thank my brilliant supervisor, Prof. Meral Azizoğlu. I felt so lucky every step of our work, being guided by her wisdom and ideas. I aspire to become one day as passionate an academician and warm a mentor as she is. I also would like to thank the jury members for their valuable contributions and insights.

No words can be enough to thank the two of the best people I know: my parents. I thank them for their encouragement throughout this study, for their patience and support, and most importantly for their love. I will forever be indebted to these two beautiful souls.

One cannot live life as it was without thanking their smart and witty best friends. I consider myself fortunate to keep the company of such strong young women: Şehnaz Genç, Pelin Dayan, and Dijan Teymur. Thank you for making me laugh.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

The gates are important resources of airports whose efficient allocations are crucial in effective air transport operations. The gate assignment problem is to assign the aircraft having prespecified arrival and departure times to the available gates to optimize the prespecified objective. Several objectives are studied in the literature, including but not limited to maximizing the number of aircraft assigned to gates, maximizing the number of passengers assigned to gates, and minimizing the walking distances of the passengers. The aircraft that could not be assigned to the gates are assigned to the remote gate, so-called apron. The satisfactory solutions to almost all gate assignments problems avoid airport assignments to the apron, due to its remote nature.

The airport operations are prone to disturbances due to changes in the flight schedules, cancellation of flights, and gate shutdowns due to maintenance and breakdowns. As a result of disturbances, so-called disruptions, the optimal solution to any gate assignment problem may no longer be preferred or may even become infeasible to implement. Hence, a need arises for reassigning the aircraft to the gates.

We consider a gate reassignment problem where the aircraft are already assigned to the gates or to the apron and disruption that affects a subset of the gates is observed. After the disruption, the aircraft assigned to the disrupted gates should be shifted to the remaining available gates or the apron. This shift may also trigger some assignment changes for the aircraft of the non-disrupted gates to give room for the aircraft of the disrupted gates. Such adjustments which are referred to as reassignment should ensure efficiency en route to low airport operating costs and stability en route to low setup costs. We assume that the initial plan was efficient so

our new plan should have its objective function as an efficiency measure. Moreover, we assume that the preparations are already made according to the initial plan, hence the new plan should stay faithful to the initial one.

In our efficiency criterion, we focus on an airport's vital need of utilizing its gate resources most efficiently. The gates are used both by aircraft and by passengers. We define our efficiency criterion with two objectives in hierarchy. The first objective that goes into our efficiency criterion is the maximization of the number of aircraft assigned to gates. With this objective, the number of ungated aircraft is minimized, i.e., the gate utilization is maximized, thus a most efficient assignment plan is obtained in terms of apron related costs. The second objective that goes into our efficiency criterion is the maximization of the number of passengers assigned to gates. By assigning the aircraft with the higher number of passengers to a gate, a contribution to the minimization of passenger walking distance or to the general customer satisfaction, since a smaller number of passengers will be routed to the remote apron, is inherently being made. Therefore, with our efficiency criteria, we maximize primarily the number of aircraft assigned to gates and secondarily the number of passengers in these aircraft. We show that the gate assignment problem that minimizes our efficiency measure is solvable in polynomial time.

In our stability criterion, we look for a new plan which is the most similar to the initial one. After a disruption, initial plan may no longer be feasible. As the name suggests, with our stability concern, we focus on staying stable, i.e., the new plan should resemble the initial plan as much as possible, preserving most of the initial assignments that remain feasible. In the event of a disruption, first and foremost, we would like to reassign the already gated aircraft to a gate again. We believe that reassigning an already gated aircraft to apron, will give a high deviation from the initial plan in terms of similarity to the initial plan and also passenger discomfort. Secondly, we would like to focalize on the number of passengers in the already assgned set of aircraft. This consideration is in parallel with that of the secondary objective of the efficiency criterion. Lastly, from an opportunistic point of view, we would like to reassign the ungated aircraft to a gate whenever possible. With the

decreased apron usage, we would be reaping the benefits of increased efficiency. Thus, a most similar reassignment plan to the initial plan is obtained through considerations in three-fold: we maximize the number of aircraft reassigned to gates that were initially assigned to gates as the primary objective, the number of passengers in these aircraft as the secondary objective, and the number of aircraft reassigned to gates that were initially assigned to apron as the tertiary objective. We show that the gate assignment problem that minimizes our stability measure is NP-Hard.

Our performance measures are studied in a multicriteria context as an increase in the efficiency value would lead to a decrease in the stability value, and vice versa, hence very fruitful trade-off analysis could be made.

Recognizing this fact, we study several trade-off problems. First, a hierarchical optimization is considered such that the efficiency (stability) value is maximized while the stability (efficiency) value is kept at its optimal, i.e., maximal level. This approach returns two extreme points and is important for the decision makers who have a strong preference for one objective.

Second, we assume that the decision maker has a linear preference (utility) function that is expressed as a weighted combination of two measures. We produce the set of objective vectors each of which is optimal for a particular weight range. These objective vectors altogether form the extreme supported nondominated objective vectors.

Finally, we assume that the decision maker has an unknown utility function of efficiency and stability criteria. We produce the set of nondominated objective vectors, one of which is optimal for a particular nonincreasing utility function. Using this set, the decision maker can make a trade-off between a certain amount of increase in efficiency value and a certain amount of decrease in stability value, and vice versa.

We present a model-based optimization approach to generate the exact nondominated set of objective vectors and a heuristic approach to generate an approximate set of all nondominated objective vectors. The heuristic approach uses the similarities of the extreme nondominated objective vectors and produces approximate nondominated set of objective vectors.

All trade-off problems that we consider are NP-Hard as the single objective problem that minimizes our stability measure is NP-Hard.

We also develop a decomposition rule where the problem is decomposed into subproblems, where each of which is dealt with independently, and then their corresponding solutions are combined by a mathematical model. We observe that if one is faced with instances for which the decomposition rule can be applied, the exact nondominated objective vectors can be found considerably easier.

To the best of our knowledge, we propose the first exact approaches for the airport gate reassignment problem. Our experiments have shown that the exact approaches can be used to tackle the real-life instances with many aircraft and many gates. We make an application for the airports in the three largest cities in Turkey: İstanbul Airport in İstanbul, Esenboğa Airport in Ankara, and İzmir Adnan Menderes Airport in İzmir, namely.

The rest of the thesis is organized as follows. In Chapter 2, we review the related literature on assignment and reassignment problems. Chapter 3 defines our gate reassignment problem and gives basic mathematical models, where one is assignment based and the other is network based. Chapter 4 presents the solution approaches that are used to generate two extreme nondominated objective vectors, all extreme supported nondominated objective vectors, and all nondominated objective vectors. We report the results of our extensive experiments in Chapter 5. Chapter 6 concludes the study by pointing out the main conclusions and suggestions for future research.

CHAPTER 2

LITERATURE REVIEW

We give literature reviews on the airport gate assignment problem (AGAP) and airport gate reassignment problem (AGRP) in the following sections.

2.1 Gate Assignment Studies

Firstly, we give the most similar works to ours in this section. The two mathematical models presented in Chapter 3, namely Assignment Based Model and Network Based Model are similar to the works of Karsu et al. (2021) and Yan and Chang (1998), respectively, in which their models are developed for the Airport Gate Assignment Problem (AGAP), where we further develop them for the Airport Gate Reassignment Problem (AGRP).

Karsu et al. (2021) studied a gate assignment problem with two objectives: minimization of apron assignments and minimization of total passenger walking distance. They give exact and heuristic solution approaches. Their problem instances mimic the real-life airports in Turkey, namely Esenboğa Airport in Ankara and Atatürk Airport in İstanbul.

Yan and Chang (1998) studied an airport gate assignment problem where they formulated their model as a multi-commodity network flow problem. They defined their objective as minimization of total passenger walking distance and case studied an international airport in Taiwan.

In an AGAP review by Daş et al. (2020), many objective functions of this assignment problem are classified under three categories. The first category, passenger-oriented objective functions, consists of objectives such as minimizing total/average

passenger walking distance, minimizing passenger waiting/transit time, minimizing baggage transferring distance and minimizing some expected passenger discomfort.

The second category, airline/airport-oriented objective functions, includes objectives such as minimizing the number of un-gated aircraft, maximizing the total duration of aircraft assigned to gates, minimizing towing cost/number of towing moves, minimizing taxi time/related fuel consumption, minimizing aircraft waiting time for a gate, maximizing total flight-gate preferences, maximizing aircraft-gate size compatibility, maximizing potential airport commercial revenues, and maximizing the number of passengers at gates close to shopping facilities to increase potential revenues.

In the last category, robustness-oriented objective functions, we have minimizing idle times at gates at peak times, minimizing range/variance/expected variance of idle times, maximizing robustness by avoiding the assignment of two flights with low buffer times to the same gate, minimizing the expected number of gate conflicts/expected gate conflict cost/worst case gate conflict, and minimizing the absolute deviation of new gate assignment from a reference schedule.

Such a large variety of objective functions is employed in a multi-objective problem setting. In the literature, according to Daş et al. (2020), multi-objective AGAP is widely handled by using weighted sum approaches.

A multi-commodity flow model is developed by Wang et al. (2022) where they focus on two objectives: robustness and taxiing times. In their multi-commodity flow model, gates are represented as commodities. The multi-objective nature of their problem is managed using a linear scalarization parameter, $\alpha \in [0,1]$, which monitors the attention given to either objective, including foregoing an objective when set to either 0 or 1. They used real-life data from the Paris-Charles-de-Gaulle international airport in France.

Another two objective gate assignment problem is studied by Cai et al. (2019). In the study, they worked to minimize the total passenger walking distance and the total

robust cost of the gate assignment. They also put a limit on the number of aircraft that can be assigned to the apron. Moreover, compatibility related constraints such as gate sizes: small/large, gate and airline leasing contracts, and flight types: international/domestic are employed. They made an application for the Baiyun airport in Guangzhou, China.

Yu et al. (2017) simultaneously focused on both the robustness and some traditional costs: the expected conflict cost, tow cost, and passenger transfer distance. they designed an adaptive large neighborhood search with some novel multiple local search operators.

Liu et al. (2022) focused on gate utilization and running time of aircraft including parking time and taxi time. In their future research topics, a special focus on the number of passengers as a form of the objective function is placed, showing the less explored nature of this important measure.

2.2 Gate Reassignment Studies

Dorndorf et al. (2007) stated that the deterministic airport gate assignment problem can be modeled as a quadratic assignment problem as shown by Sahni and Gonzalez (1976) is NP hard. Dorndorf et al. (2007) further stated that even a small disruption at the beginning of the day often has severe results by the end of the day due to the knock-on effect.

Disruptions in the gate reassignment studies are not only limited to maintenance operations, flight and gate breakdowns, adverse weather conditions, emergency flights, flight earliness and delays, and flight cancellations but also include major incidents such as abnormal meteorological conditions and labor strikes of airport employees, which may result in temporary airport closures. The gate reassignment problem following such a major event is studied by Yan et al. (2009).

The literature regarding the gate reassignment problem widely consists of multi-objective studies.

Pternea and Haghani (2019) proposed hierarchical optimization for a gate reassignment problem to handle their selected multiple objectives: minimization of some costs of (i) flight assignment, (ii) successful passenger connections, and (iii) failed passenger connection. They gave several cost coefficient definitions, among which the most related to our work include number of the flights with gate changes, number of passengers with gate changes, number of flights assigned to remote gates but originally assigned to contact gates, number of passengers assigned to remote gates but originally assigned to contact gates.

Zhang and Klabjan (2017) define an efficient gate reassignment methodology for occurrences of disruptions. They come up with two multi-commodity network flow models, one of which for passenger connections, where each gate represents a commodity, and two heuristic algorithms since the reassignment model is NP hard. They handle the minimization of total flight delays, the number of gate reassignment operations, total passenger transfer distance, and the number of missed passenger connections with the weighted sum approach.

Dorndorf et al. (2012) studied the problem with the objectives: some assignment preference score, number of unassigned flights during overload periods, number of tows, some robustness measure, and the one most familiar to our work which is a deviation from a given reference schedule. With the last objective, they introduce a stability measure to their work while emphasizing that giving importance to staying faithful to the initial plan entails a more concise gate schedule which helps with the passenger satisfaction and is of convenience to the airport staff.

Yan et al. (2011) assumed to handle the uncertainty around aircraft arrival and departure times: that some flights which are closer to the time of planning reassignments tend to be more certain, and the further away they are, they become more stochastic. That's why they divided flights into the following two categories: deterministic flights and stochastic flights. They made an application to the Taiwan Taoyuan Airport in Taiwan.

Flight delays, whether in the form of early or tardy flight arrivals or in the form of tardy flight departures, in general, constitute the disruptions worked on by Tang et al. (2010), where they emphasized the crucial need of developing a framework for the gate reassignment problem, stating that the traditional manual flight reassignment method has too many shortcomings.

In their study, Deng et al. (2017) worked with multi-objectives that take into consideration the loss of passengers, cost of airport operating, and economic loss of airlines, in one criterion and for the other criterion, constructed a measure called the most important index of disturbance value to manage the deviation from the initial plan. They integrated two metaheuristics: the genetic algorithm and the ant colony algorithm to propose a two-stage hybrid method.

Wang et al. (2013) handled flight delays in two categories: certain delay time flights and uncertain delay time flights. For the former case, they aimed to minimize the apron and gate disturbance values and for the latter case, they aimed to minimize the gate disturbance value, time disturbance value, and some penalty value. They applied an ant colony-based heuristic to their model.

One of the most popular objectives in the gate assignment literature, minimization of the total walking distance of passengers, is also studied in Maharjan and Matis (2011). In this particular case, they considered the passengers who are either connecting to or originating from an airport where their boarding passes were issued before the reassignment of gates. They implemented their work for the operations of Continental Airlines at the George W. Bush Intercontinental Airport in Houston, Texas.

Further literature examples consist of Pternea and Haghani (2018) where they studied the gate reassignment problem with passenger connections, Gu and Chung (1999) where they implemented a genetic algorithm for the minimization of extra delay times, and Ali et al. (2019) where they proposed a passenger-centric model that minimizes the transit time of transfer passengers.

In this study we consider a gate reassignment problem with efficiency and stability criteria. Our efficiency criterion aims to maximize the number of aircraft assigned to gates and the number of passengers in these aircraft and our stability criterion maximizes the number of same gate assignments from the reference assignment plan and their number of passengers as well as the number of aircraft assigned to gates. To the best of our knowledge, there is no reported gate reassignment study that considers our efficiency and stability criteria simultaneously.

2.3 The Most Closely Related Studies

Our work differs from the existing literature based on selecting fairly unexplored multi-criteria, applying hierarchical optimization to handle said multi-criteria where the existing literature mostly applied the weighted-sum method, and proposing and comparing two mathematical models for the airport gate reassignment problem (AGRP). Aforesaid points besides, we give the mostly related studies to ours below.

Both Karsu et al. (2021) and Yan and Chang (1998) proposed exact approaches for the airport gate assignment problem (AGAP). While the former devised an assignment-based mathematical model, the latter formulated the problem as a multi commodity network flow. In spirit of these studies, we develop an Assignment Based Model in Section 3.2.1 and a Network Based Model in Section 3.2.2 for the airport gate reassignment problem (AGRP).

Furthermore, we define a multi-objective environment for AGRP and handle our objective functions through hierarchical optimization as proposed in Pternea and Haghani (2019). Although the logic behind using hierarchical optimization remains similar, we define widely different criteria for the AGRP than that of Pternea and Haghani (2019).

CHAPTER 3

PROBLEM DEFINITION AND MATHEMATICAL MODELS

We consider an Airport Gate Reassignment Problem (AGRP) with *n* aircraft, *m* gates and an apron. The elements of the problem are illustrated in Figure 3.1.

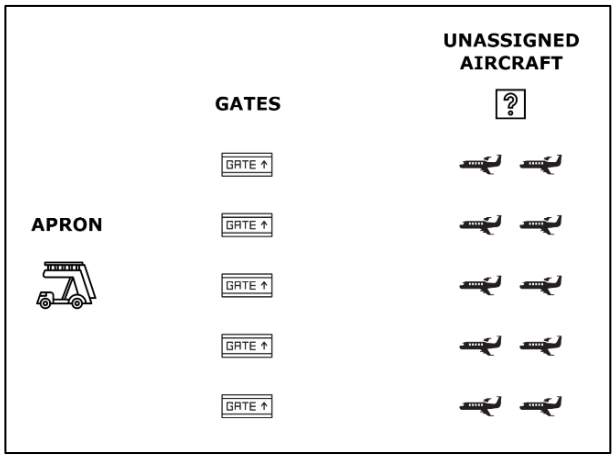

Figure 3.1. Illustration of Problem Elements

The apron is assumed to have infinite aircraft capacity, however, is too far away from the gates, hence is not favored by any reason. The aircraft that cannot be assigned to a gate, are assigned to apron. Each aircraft has a specified arrival time and a departure time. From its arrival time until its departure time, the aircraft is either at its assigned gate or at apron. This is compatible with the real-life application of renting the gates to airlines for fixed periods of time, i.e., time intervals. Moreover, each aircraft has a specified number of passengers who either have entered from the entrance point or transferred from other aircraft.

We assume that there is an initial plan that shows the assignment of each aircraft to either one of the gates or to the apron. We use the term ‘initial plan’ to refer to the interchangeably used terms in the literature: ‘current assignment’, ‘initial assignment’, and ‘reference assignment’. An initial plan is illustrated in Figure 3.2.

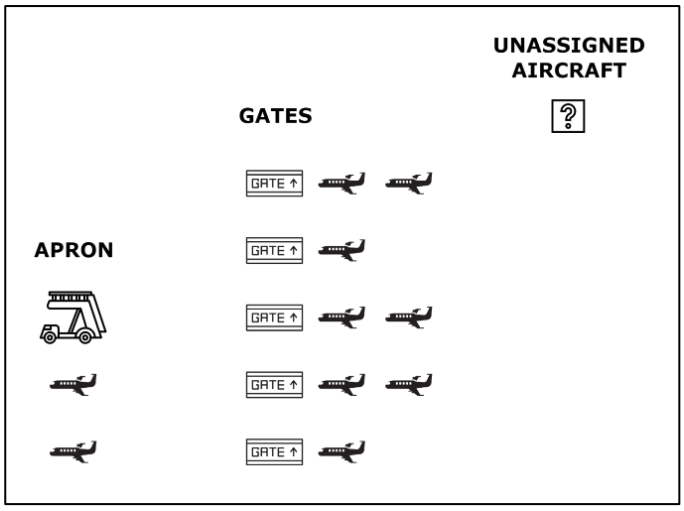

Figure 3.2. Illustration of Initial Plan

There is a disruption at the beginning of the planning horizon that affects a specified set of gates. This disruption may be due to several reasons such as breakdowns or maintenance operations, as previously sampled in Chapter 2, and makes the affected gates inoperable. We use the term ‘affected gates’ to refer to ‘disrupted gates’; and ‘affected aircraft’ to refer to ‘disrupted aircraft’. An illustration of a disruption is shown in Figure 3.3, where two gates are affected by the disruption.

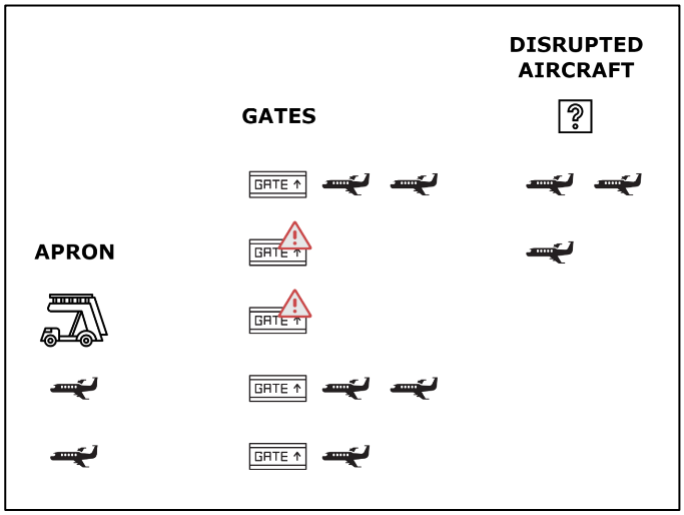

Figure 3.3. Illustration of a Disruption

After the disruption, a new plan is formed where the affected aircraft are assigned to one of the nonaffected gates or to the apron. The nonaffected aircraft may be reassigned to its initial gate or to any one of the nonaffected gates or to the apron. We use terms ‘new plan’ and ‘reassignment’ interchangeably.

We assume that there are no assignment restrictions, i.e., all aircraft can be assigned to one of the *m* gates. Furthermore, the arrival and departure times of the aircraft, the number of passengers and all other parameters that will be defined later, are known with certainty and not subject to any change. That is, the system we consider is deterministic and static.

The initial plan is known and found by the efficiency concerns of the decision makers. The efficiency and stability concerns define our performance measures, each of which is discussed next.

3.1 The Two Criteria

From an efficiency perspective and a stability perspective, we discuss our objective functions in this section.

3.1.1. Efficiency Criterion

From an airport and passenger satisfaction point of view, an assignment plan should meet the following requirements: (i) its number of aircraft assigned to gates should be as high as possible, and (ii) if there is a tie among multiple aircraft that are potentially competing against each other to be assigned to a gate, then the decision should be in favor of the one with the highest number of passengers. We call such an assignment plan *efficient* due to it having the least apron usage both by aircraft and their corresponding passengers.

The efficiency concern is turned into an objective function, E, with the direction of maximization. Thus, efficiency objective function is as follows:

$E1$ The primary objective that is number of aircraft assigned to gates

$E2$ The secondary objective that is number of passengers assigned to gates

E Efficiency objective function, $E = E1 + \varepsilon_E E2$ where ε_E is a sufficiently small number that gives priority to $E1$ and breaks the ties in favor of $E2$.

ε_E should be set small so that $E1$ value does not decrease even by one unit for the highest improvement of the $E2$ value. Note that, in addition to establishing the hierarchy between objective functions, the parameter ε_E, also performs some rescaling between the objective functions, where one objective is in units of aircraft and the other is in units of passengers. This follows,

$$E1 + \varepsilon_E E2_{min} \geq E1 - 1 + \varepsilon_E E2_{max} \quad (1)$$

where

$E2_{min} =$ the smallest possible value of $E2$,

$E2_{max} =$ the largest possible value of $E2$,

We define p_i as the number of passengers in aircraft i.

$E2_{max} = \sum_{i=1}^{n} p_i$ (all aircraft are assigned to gates).

$E2_{min}$ is found by collecting *m* aircraft having smallest p_i values as:

$E2_{min} = \sum_{i=1}^{m} p_{[i]}$ where $p_{[i]}$ is the i^{th} smallest p_i value (all gates are busy with one and only one aircraft).

Inequality (1) reduces to

$$\varepsilon_E E2_{min} \geq \varepsilon_E E2_{max} - 1$$

$$\varepsilon_E \leq \frac{1}{E2_{max} - E2_{min}}$$

Putting $E2_{min}$ and $E2_{max}$ into above expression, we find

$$\varepsilon_E \leq \frac{1}{\sum_{i=1}^{n} p_i - \sum_{i=1}^{m} p_{[i]}}$$

In our experiments, we set

$$\varepsilon_E = \frac{1}{\sum_{i=1}^{n} p_i - \sum_{i=1}^{m} p_{[i]} + 1} \quad (2)$$

Putting (2) into our objective function, we get

$$E = E1 + \frac{1}{\sum_{i=1}^{n} p_i - \sum_{i=1}^{m} p_{[i]} + 1} E2$$

To have integer value for E, we multiply it by $\sum_{i=1}^{n} p_i - \sum_{i=1}^{m} p_{[i]} + 1$ and get the following expression for our efficiency measure:

$$E = \left[\sum_{i=1}^{n} p_i - \sum_{i=1}^{m} p_{[i]} + 1 \right] E1 + E2$$

3.1.2. Stability Criterion

In the AGRP, the new plan that will be obtained after a gate disruption can surely have some reminiscence of the initial plan. Aside from our efficiency concern, we have another perspective from the stability side. We define our stability concern as staying faithful to the initial plan. Although the initial plan might become infeasible, per this concern, the number of gate assignments in the initial plan, the corresponding number of passengers and the number of gate assignments that were initially assigned to the apron are maximized in this very order. To clarify, we define below some new sets and objective functions.

S_1 Set of aircraft that are assigned to gates in the initial plan

S_2 Set of aircraft that are assigned to the apron in the initial plan

$I = S_1 \cup S_2$

$ST1$ Number of aircraft in S_1 assigned to their initial gates

$ST2$ Number of passengers of flights in S_1 assigned to their initial gates

$ST3$ Number of aircraft in S_2 assigned to gates

Our stability aim is primarily to maximize $ST1$. Among the optimal solutions of $ST1$, we prefer the one having maximum $ST2$. Hence, we first want to maximize $ST1 + \varepsilon_{ST1}ST2$, where ε_{ST1} is found as follows:

ε_{ST1} should be sufficiently small so that $ST1$ value does not reduce even by one unit for the highest improvement of the $ST2$ value. Accordingly,

$$ST1 + \varepsilon_{ST1}ST2_{min} \geq ST1 - 1 + \varepsilon_{ST1}ST2_{max} \quad (3)$$

where

$ST2_{min}$ = minimum possible $ST2$ value which we set to zero.

$ST2_{max}$ = maximum possible $ST2$ value

$ST2_{max}$ is the number of aircraft assigned to nonaffected gates in the initial assignment = $\sum_{m \in M}|SA_m|$ where $|SA_m|$ is the set of aircraft assigned to gate m in the initial plan.

Inequality (3) reduces to

$$\varepsilon_{ST1} \geq \varepsilon_{ST1}ST2_{max} - 1$$

$$\varepsilon_{ST1} \leq \frac{1}{ST2_{max}} = \frac{1}{\sum_{j=1}^{m}|SA_j|}$$

In our experiments we set ε_{ST1} to

$$\varepsilon_{ST1} = \frac{1}{\sum_{j=1}^{m}|SA_j|+1} \quad (4)$$

Putting (4) into our objective function, we get

$$ST1 + \frac{1}{\sum_{j=1}^{m}|SA_j| + 1}ST2$$

We multiply the above expression by $\sum_{j=1}^{m} |SA_j| + 1$ to get an integer value for the objective function as:

$(\sum_{j=1}^{m} |SA_j| + 1)ST1 + ST2 = ST_A$

Among the optimal solutions to ST_A, we prefer the one having maximum $ST3$ value. Hence, we want to maximize

$$ST = ST_A + \varepsilon_{ST2} ST3 \quad (5)$$

where ε_{ST2} is a sufficiently small number, which is found using the ideas of ε_E and ε_{ST1}. Similarly, parameters ε_{ST1} and ε_{ST2} perform rescaling between the objective functions as well as handling the hierarchy between them. This follows,

$ST_A + \varepsilon_{ST2} ST3_{min} \geq ST_A - 1 + \varepsilon_{ST2} ST3_{max}$

where $ST3_{min}$ is set to zero and $ST3_{max}$ is set to the number of aircraft assigned to apron in the initial assignment, i.e., $|SA_{m+1}|$.

Rearranging (5) with $ST3_{min} = 0$ and $ST3_{max} = |SA_{m+1}|$, we get $\varepsilon_{ST2} \geq \frac{1}{ST3_{max}} = \frac{1}{|SA_{m+1}|}$ and use $\varepsilon_{ST2} = \frac{1}{|SA_{m+1}|+1}$ in our experiments.

The overall stability measure is expressed as

$ST = ST_A + \frac{1}{|SA_{m+1}|+1} ST3.$

To get an integer valued objective function, we multiply the above expression with $|SA_{m+1}| + 1$.

The overall stability measure ST becomes

$ST = (|SA_{m+1}|+1)\left(\sum_{j=1}^{m} |SA_j| + 1\right)ST1 + (|SA_{m+1}|+1)ST2 + ST3.$

To sum up, our problem has the following two objective functions:

Max E

Max ST

In the next section, we discuss the mathematical models used to solve the multi-objective problem.

3.2 Mathematical Models

In this section, we define two mathematical models to the airport gate reassignment problem (AGRP) with efficiency and stability criteria.

Our efficiency measure maximizes the number of aircraft and then the number of passengers, assigned to gates. The stability measure tries to keep the assignments close to their initial plan counterparts.

The first mathematical model is a classical assignment-based model that is also used by Karsu et al. (2021) for a gate assignment problem. The second model is a network-based model that takes its spirit from the network-based model of Yan and Chang (1998) that is defined for the gate assignment problem. We next discuss the details of the models.

3.2.1. Assignment Based Model

The Assignment Based Model uses the following sets and parameters:

I Set of aircraft

K Set of gates (gates and apron)

n Number of aircraft ($|I|$)

m Number of gates ($|K| - 1$ gates, gate $m + 1$ is apron)

p_i Number of passengers in aircraft $i,\ i = 1, \dots, n$

a_i Arrival time of aircraft $i,\ i = 1, \dots, n$

d_i Departure time of aircraft $i,\ i = 1, \dots, n$

R Number of distinct a_i and d_i values, where $R-1$ is the number of time intervals

During $[a_i, d_i]$, aircraft i stays at the airport.

$\{ad_1, ad_2, \dots, ad_R\}$ Set of distinct a_i and d_i values in chronological order

During interval (ad_r, ad_{r+1}) there is no arrival or departure.

$$o_{ir} = \begin{cases} 1, \text{if aircraft } i \text{ is in the airport at interval } r, r = 1, \dots, R-1 \\ 0, \text{otherwise} \end{cases}$$

Stability related parameter, c_{ik}, is defined as:

$$c_{ik} = \begin{cases} 1, \text{if aircraft } i \text{ is assigned to gate } k \text{ in the initial plan} \\ i = 1, \dots, n \; k = 1, \dots, m+1 \\ 0, \text{otherwise} \end{cases}$$

The assignment decision variable is defined as:

$$x_{ik} = \begin{cases} 1, \text{if aircraft } i \text{ is assigned to gate } k \text{ in the new plan} \\ i = 1, \dots, n \; k = 1, \dots, m+1 \\ 0, \text{otherwise} \end{cases}$$

The constraint sets are as given below:

$\sum_{k=1}^{m+1} x_{ik} = 1$ $\quad i = 1, \dots, n$ (A)

$\sum_{i=1}^{n} o_{ir} x_{ik} \leq 1$ $\quad k = 1, \dots, m \; r = 1, \dots, R-1$ (B)

$x_{ik} \in 0$ or 1 $\quad i = 1, \dots, n \; k = 1, \dots, m+1$ (C)

Constraint (A) ensures each aircraft i is assigned to a single gate. The overlapping of aircraft is handled using binary parameter o_{ir}, where in constraint (B) aircraft that are in the system at the same time interval cannot be assigned to the same gate. Lastly, Constraint (C) states that the decision variable x_{ik} is binary.

We refer to the constraint sets (A), (B), and (C) as $x \in X_A$.

Efficiency measures are stated in their priority order:

$E1$ Max $\sum_{i=1}^{n} \sum_{k=1}^{m} x_{ik}$ (primary)

$E2$ Max $\sum_{i=1}^{n}\sum_{k=1}^{m} p_i x_{ik}$ (secondary)

$E1$ maximizes the number of gated aircraft, and $E2$ maximizes the corresponding number of passengers. The aggregate objective function, E_A is as follows:

Max E_A $\frac{1}{\varepsilon_E}\sum_{i=1}^{n}\sum_{k=1}^{m} x_{ik} + \sum_{i=1}^{n}\sum_{k=1}^{m} p_i x_{ik}$ where $\varepsilon_E = \frac{1}{\sum_{i=1}^{n} p_i - \sum_{i=1}^{m} p_{[i]}+1}$.

Karsu et al. (2021) showed that the problem of maximizing the number of aircraft assigned to gates (Max $\sum_{i=1}^{n}\sum_{k=1}^{m} x_{ik}$) is solved in polynomial time using a network flow model. The same network structure holds when the arc costs of '1', for the arcs that emanate from the node representing aircraft i, are replaced by $\frac{1}{\varepsilon_E} + p_i$. This follows, our efficiency problem can be solved in polynomial time.

Stability measures are stated in their priority order:

$ST1$ Max $\sum_{i=1}^{n}\sum_{k=1}^{m} c_{ik} x_{ik}$ (primary)

$ST2$ Max $\sum_{i=1}^{n}\sum_{k=1}^{m} p_i c_{ik} x_{ik}$ (secondary)

$ST3$ Max $\sum_{i|c_{im+1}=1}\sum_{k=1}^{m} x_{ik}$ (tertiary)

$ST1$ maximizes the number of aircraft assigned to their initial gate. $ST2$ maximizes the number of passengers assigned in these aircraft. $ST3$ maximizes the number of aircraft assigned to gates that were initially assigned to the apron.

The aggregate objective function, ST_A, is as follows:

Max ST_A $\frac{1}{\varepsilon_{ST1}\varepsilon_{ST2}}\sum_{i=1}^{n}\sum_{k=1}^{m} c_{ik} x_{ik} + \frac{1}{\varepsilon_{ST2}}\sum_{i=1}^{n}\sum_{k=1}^{m} p_i c_{ik} x_{ik} +$

$\sum_{i|c_{im+1}=1}\sum_{k=1}^{m} x_{ik}$ where $\varepsilon_{ST1} = \frac{1}{\sum_{i=1}^{n}\sum_{i=1}^{m} c_{ik}+1}$ and $\varepsilon_{ST2} = \frac{1}{\sum_{i=1}^{n} c_{im+1}+1}$.

As mentioned in Jaehn (2010) the problem of maximizing the total aircraft-gate preference score (Max $\sum_{i=1}^{n}\sum_{k=1}^{m} p_{ik} x_{ik}$) is NP-Hard. Our stability measure reduces to the total aircraft-gate preference score when $p_{ik} = \frac{1}{\varepsilon_{ST1}\varepsilon_{ST2}} c_{ik} + \frac{1}{\varepsilon_{ST2}} p_i c_{ik} + 1$. This follows, our stability problem can be solved in polynomial time.

Assignment Based Model can be summarized as:

Max E_A

Max ST_A

subject to $x \in X_A$

3.2.2. Network Based Model

Yan and Chang (1998) formulated the gate assignment problem as a multi-commodity network flow model. We use the basics of Yan and Chang (1998) model to define our network-based formulation. For the sake of completeness, we use the sets and parameters defined for the Assignment Based Model.

For each aircraft a network is presented. The network for aircraft i, NT_i, has a source node S_i and an end node T_i. There are $m + 1$ arcs departing from S_i, each arc representing the flow to a particular gate or apron.

Let t_i be the set of chronological time intervals from the set $\{ad_1, ad_2, \dots, ad_R\}$ where aircraft i is in the airport and let r_i be the number of time intervals from the set $\{ad_1, ad_2, \dots, ad_R\}$ where aircraft i is in the airport such that $t_i = t_1, \dots, t_{r_i}$. That is, in the chronological list of ad_r values, if $ad_{t_1} = a_i$ then $ad_{t_1+r_i-1} = d_i$.

We say that r_i many nodes are defined for each gate leg. Hence there are $r_i(m + 1) + 2$ nodes in NT_i.

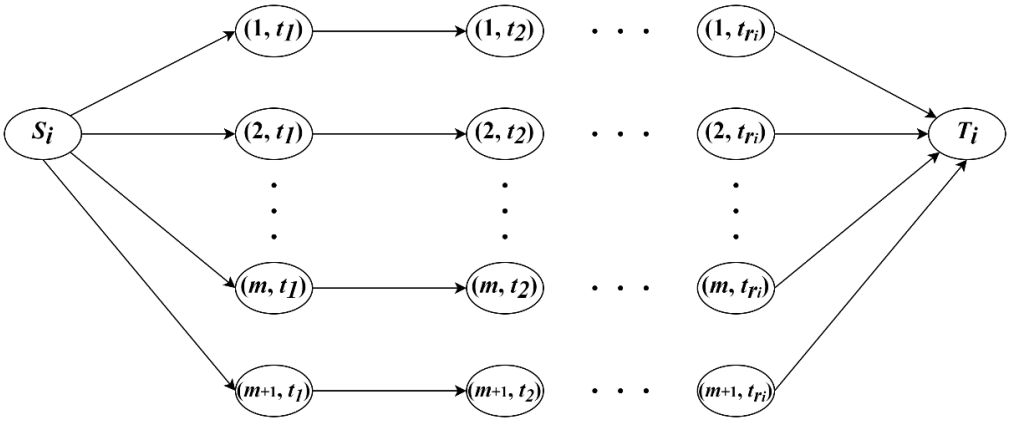

Figure 3.4. Network Representation of Aircraft i

For NT_i, we define node set, NS_i and arc set, AS_i as follows:

$NS_i = \cup_{k=1,\dots,m+1} NS_{ik} \cup S_i \cup T_i$ where $NS_{ik} = \{[k, t_1], \dots, [k, t_{r_i}]\}$

$$AS_i = \bigcup_{k=1,\dots,m+1} AS_{ik}$$

where $AS_{ik} = \{[S_i, (k, t_1)], \dots, [(k, t_{r_i-1}), (k, t_{r_i})], [(k, t_{r_i}), T_i]\}$

The basic parameter is the gain of each arc. The arc gains from node S_i to its neighbor node are defined as $EAC[S_i, (k, t_1)]$ and $SAC[S_i, (k, t_1)]$ where $k \neq m + 1$ for efficiency and stability measures, respectively. For both measures, all other arc gains are zero.

For an arc, let's show it by its starting node s and its ending node e.

The decision variable is defined as:

$$x^i_{(s,e)} = \begin{cases} 1, \text{if arc } (s, e) \text{ is selected} \\ \qquad i = 1, \dots, n \\ s \in \bigcup_{k=1,\dots,m+1} NS_{ik} \bigcup S_i \\ e \in \bigcup_{k=1,\dots,m+1} NS_{ik} \bigcup T_i \\ 0, \text{otherwise} \end{cases}$$

The constraint sets are defined as below:

$$\sum_{\substack{s\in\cup_{k=1,\dots,m+1} NS_{ik}\cup S_i \\ e\in\cup_{k=1,\dots,m+1} NS_{ik}\cup T_i}} x^i_{(s,e)} - \sum_{\substack{j\in\cup_{k=1,\dots,m+1} NS_{ik}\cup S_i \\ s\in\cup_{k=1,\dots,m+1} NS_{ik}\cup T_i}} x^i_{(j,s)} = \begin{cases} 1, & s = S_i \\ 0, & s \neq S_i, T_i \\ -1, & s = T_i \end{cases}$$

$$i = 1, \dots, n \text{ (D)}$$

$$\sum_{i=1}^{n} x^i_{(s,e)} \leq 1 \qquad \forall (s,e) \in \cup_{i=1,\dots,n} AS_i \text{ (E)}$$

$$x^i_{(s,e)} \in \{0,1\} \qquad i = 1, \dots, n,\ \forall (s,e) \in AS_i \text{ (F)}$$

Constraint sets (D) work as a classical flow balance constraint, ensuring an assigned aircraft be assigned to the same gate all throughout its present time intervals. An aircraft can only be assigned to a single gate as guaranteed by Constraint (E). Lastly, decision variable $x^i_{(s,e)}$ is binary as stated by Constraint (F). We hereafter refer to the constraint sets (D), (E), and (F) as $x \in X_N$. We now discuss the efficiency and stability measures for each aircraft i.

Efficiency Measure

Max E_N $\quad \sum_{i=1}^{n} \sum_{(s,e)\in AS_i} EAC[S_i, (k, t_1)]\, x^i_{(s,e)}$

where $EAC[S_i, (k, t_1)] = \frac{1}{\varepsilon_E} + p_i$.

Stability Measure

Max ST_N $\quad \sum_{i=1}^{n} \sum_{(s,e)\in AS_i} SAC[S_i, (k, t_1)]\, x^i_{(s,e)}$

where $SAC[S_i, (k, t_1)] = \frac{1}{\varepsilon_{ST1}\varepsilon_{ST2}} c_{ik} + \frac{1}{\varepsilon_{ST2}} p_i c_{ik} + c_{im+1}$ if $k = 1, \dots, m$

The Network Flow Based Model can be summarized as:

Max E_N

Max ST_N

subject to $x \in X_N$

A small-sized example is provided to illustrate a network for aircraft i, NT_i. In this example with $n = 5, m = 3$, let's assume the following sets and parameters:

Table 3.1 Example Network for an Aircraft: a_i, d_i, p_i

Aircraft, i	*Arrival time,* a_i	*Departure time,* d_i	*Passengers,* p_i
1	1	53	200
2	65	99	100
3	189	226	150
4	186	232	100
5	71	129	200

The distinct and chronological time intervals are derived as:

Table 3.2 Example Network for an Aircraft: r_t, ad_r

Interval, r_t	*Distinct time value,* ad_r
1	1
2	53
3	65
4	71
5	99
6	129
7	186
8	189
9	226

Then, the parameter o_{ir} is found as:

$$o_{ir} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

We demonstrate the r_i and t_i values below.

Table 3.3 Example Network for an Aircraft: r_t, t_i

Aircraft, i	*Number of time intervals,* r_i	*Time intervals,* t_i
1	2	{1, 53}
2	3	{65, 71, 99}
3	2	{226, 232}
4	3	{189, 226, 232}
5	3	{71, 99, 129}

So, $t_2 = \{ad_3, ad_4, ad_5\} = \{65, 71, 99\}$.

Let's calculate the ε_E to demonstrate arc gains on this network. To do so, we first find the $E2_{max}$ and $E2_{min}$.

$$E2_{max} = \sum_{i=1}^{n} p_i = 750$$

$$E2_{min} = \sum_{i=1}^{m} p_{[i]} = 100 + 100 + 150 = 350$$

$$\varepsilon_E = \frac{1}{\sum_{i=1}^{n} p_i - \sum_{i=1}^{m} p_{[i]} + 1} = \frac{1}{750-350+1} = \frac{1}{401}$$

Efficiency measure for aircraft 2, is found as below for $k = 1,2,3$.

$$EAC[S_2, (k, 65)] = \frac{1}{\varepsilon_E} + p_2 = 401 + 100 = 501.$$

Figure 3.5 demonstrates the network for aircraft 2.

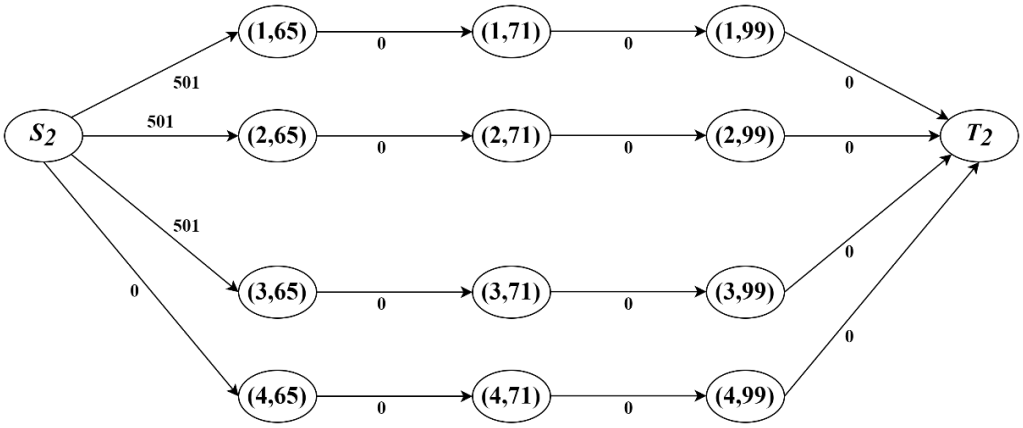

Figure 3.5. Example Network for Aircraft 2

CHAPTER 4

NONDOMINATED OBJECTIVE VECTORS

In this chapter, we discuss the nondominated objective vectors and their generation method. We define the nondominated objective vectors in Section 4.1. In Section 4.2 and Section 4.3, we define algorithms to generate extreme nondominated objective vectors. Section 4.4 discusses the generation of all nondominated objective vectors. In Section 4.5, a procedure to generate approximate nondominated objective vectors is proposed. In Section 4.6 and Section 4.7, we introduce an optimal decomposition rule and propose its heuristic implementation, respectively.

The problems that we work in Sections 4.3, 4.4 and 4.5 are all NP-Hard due to the NP-Hardness of our stability problem.

4.1 Nondominated Objective Vectors

A gate assignment solution r in $x \in X_A$ (or $x \in X_N$) is called efficienct if there is no other solution q in $x \in X_A$ (or $x \in X_N$) with in $E_q \geq E_r$ and $ST_q \geq ST_r$ with strict inequality holding at least once ($E_q > E_r$ and $ST_q \geq ST_r$ or $E_q \geq E_r$ and $ST_q > ST_r$). The associated objective vector (E_r, ST_r) is said to be nondominated objective vector (ndov). The solution q is dominated by solution r and the nondominated objective vector (E_q, ST_q) is dominated by the nondominated objective vector (E_r, ST_r).

4.2 Extreme Nondominated Objective Vectors

An efficient solution is called an extreme efficient solution if it has the largest objective function value for one objective.

A nondominated objective vector corresponding to an extreme efficient solution is called extreme nondominated objective vector. We discuss the generation of the nondominated objective vectors with the largest E and the largest ST values in subsections 4.2.1 and 4.2.2, respectively. We used the Assignment Based Model in the generation methods.

Our efficiency concern and stability concern make up two different perspectives to the problem. Hence, their resulting solutions are treated as the two extreme ends of a solution spectrum.

4.2.1 Extreme Nondominated Objective Vector with the Largest E Value

Consider the following problem:

Max E

subject to

$x \in X_A$

Let E^* be the optimal objective function value.

E^* is an upper bound on the efficiency values of all efficient solutions. However, any feasible solution with efficiency value of E^* is not necessarily efficient as there may exist another solution with a larger ST value. The solution having the maximum ST value among the solutions having efficiency value of E^* can be found using the following two-step procedure.

Procedure 1 Finding an Extreme Nondominated Objective Vector with the Largest E Value

Step 1. Solve the Max E subject to $x \in X_A$ problem. Let E_1^* be the optimal objective function value.

Step 2. Solve the following problem

Max ST

subject to

$E = E_1^*$

$x \in X_A$

Let ST_1^* be the optimal objective function value. Then, the first extreme nondominated vector is $(E, ST) = (E_1^*, ST_1^*)$.

An alternative model that delivers (E_1^*, ST_1^*) is given below:

Max $E + \varepsilon_E ST$

subject to

$x \in X_A$

where ε_E is a sufficiently small number.

ε_E should be small enough so that E value does not increase even one unit for the maximum value of ST.

This follows,

$$E + \varepsilon_E ST_{min} \geq E - 1 + \varepsilon_E ST_{max}$$

$$\varepsilon_E \leq \frac{1}{ST_{max} - ST_{min}}$$

We observe that ST_{max} can be so large, making the ε_E value so small, that the hierarchical relation between the objectives be lost due to rounding problems. Hence,

we use the two-step procedure to generate the efficient extreme nondominated objective vector.

4.2.2 Extreme Nondominated Objective Vector with the Largest *ST* Value

Consider the following problem:

Max ST

subject to

$x \in X_A$

In the optimal solution to the above problem, the initial plan should be implemented to its the maximum extent. To achieve this, we fix the aircraft that are not affected by disruptions to their initial gates. In doing so, the first and second parts of the stability objective are maximized. Hence, a great emphasis is put on fixing the aircraft that are not affected by disruptions. By keeping the initial plan for the aircraft, that were initially assigned to gates and are not affected by disruptions, we ensure that the new plan will be the most faithful one to the initial plan. To this end, we pre-process our Stability Model input and create a so-called "Reduced Problem" to run our model on. The steps to create a reduced problem are shown below.

Procedure 2 Creating a Reduced Problem for Stability Model

Step 1. Assign the aircraft that are not affected by disruptions to their initial gates

Step 2. From the remaining unassigned aircraft, find which ones can only be assigned to the apron, make the apron assignment for these "must-go" aircraft

Step 3. From the remaining unassigned aircraft, find which ones have a one-to-one relationship with an open time interval at a gate, meaning that an aircraft can only be assigned to a specific gate and this specific gate have no other possible unassigned aircraft for this open time interval. Make this one-to-one assignment.

Step 4. After making the pre-assignments, we have a partial assignment plan, and the remaining unassigned aircraft form the reduced problem that will be worked on.

In a reduced problem, for each aircraft of the reduced problem, we define a network. This network has only arcs for the gates that the aircraft can be assigned. Note that an aircraft now has a set of eligible gates due to the fixed gates.

The resulting network-based model is as stated below:

n' Number of unfixed aircraft

$S(i)$ Set of gates that are eligible for aircraft i, $i = 1, \dots, n'$

The objective function is:

Max $ST + \in'_{ST} E$ where $ST = \frac{1}{\varepsilon_{ST1}\varepsilon_{ST2}} ST1 + \frac{1}{\varepsilon_{ST2}} ST2 + ST3$, $E = \frac{1}{\varepsilon_E} E1 + E2$, and $\in'_{ST} = \frac{1}{E_{max} - E_{min} + 1}$. E_{max} and E_{min} are found by considering only the unassigned aircraft as follows:

E_{max} E value under the assumption that all aircraft in the reduced problem are assigned to gates, $E_{max} = \frac{1}{\varepsilon_E} n' + \sum_{i=1}^{n'} p_i$.

E_{min} E value under the assumption that none of the aircraft in the reduced problem can be assigned to a gate, $E_{min} = 0$.

The constraint sets are

$\sum_{k \in S(i)} x_{ik} = 1 \qquad i = 1, \dots, n'$

$\sum_{i=1}^{n'} comp_{ir} x_{ik} \leq 1 \qquad \forall k \in \bigcup_{i=1,\dots,n'} S(i)\ r = 1, \dots, R-1$

$x_{ik} \in 0 \text{ or } 1 \qquad i = 1, \dots, n'\ \forall k \in \bigcup_{i=1,\dots,n'} S(i)$

Let (E', ST') be the optimal solution to the above model.

To find, (E_2, ST_2) vector, that is the nondominated vector for the second extreme nondominated objective vector, we consider the fixed aircraft as well. The resulting vector is found as:

$E_2 = E' + \frac{1}{\varepsilon_E} E1 + E2$ where $\frac{1}{\varepsilon_E} E1 + E2$ is the efficiency term of the fixed aircraft.

$ST_2 = ST' + \frac{1}{\varepsilon_{ST1}\varepsilon_{ST2}} ST1 + \frac{1}{\varepsilon_{ST2}} ST2 + ST3$ where $\frac{1}{\varepsilon_{ST1}\varepsilon_{ST2}} ST1 + \frac{1}{\varepsilon_{ST2}} ST2 + ST3$ is the stability term of the fixed aircraft.

We give a small example with $n = 10, m = 4$ with a disruption that closes two gates randomly, to demonstrate creating a reduced problem below.

Let's assume a problem with the following parameters and initial plan where a disruption causes gates 2 and 4 to be closed.

Table 4.1 Example of a Reduced Problem

Aircraft, *i*	*Initial assignment, k*	*Reduced problem*
1	1	Aircraft is not affected by the disruption, fix its gate.
2	**4** (closed)	Aircraft is affected by the disruption, no gate is available to this aircraft, it is assigned to apron.
3	**2** (closed)	Aircraft is affected by the disruption, only gate 3 is available to this aircraft, aircraft 9 is also affected and overlaps with aircraft 3, so it is in the reduced problem.
4	**2** (closed)	Aircraft is affected by the disruption, there are multiple gates available to this aircraft, it is in the reduced problem.
5	**4** (closed)	Aircraft is affected by the disruption, only gate 1 is available to this aircraft, there are no affected aircraft overlapping with aircraft 5, it is assigned to gate 1.
6	5 (apron)	Aircraft is not affected by the disruption, fix its gate.
7	**4** (closed)	Aircraft is affected by the disruption, only gate 3 is available to this aircraft, aircraft 3 is also affected and overlaps with aircraft 7, so it is in the reduced problem.
8	5 (apron)	Aircraft is not affected by the disruption, fix its gate.

Then, from Table 4.1, we see that a reduced problem is created with aircraft 3, 4, and 9, and gates 1 and 3.

4.3 Extreme Supported Nondominated Objective Vectors

An efficient solution is called supported efficient solution if it is optimal for the linear combination of E and ST, i.e., $wE + (1 - w)ST$ for any positive w. If the efficient solution does not optimize $wE + (1 - w)ST$ for all positive w, then it is non-supported efficient.

A supported efficient solution is called extreme supported efficient solution if it can be found by changing the value of w. A supported efficient solution is nonextreme supported efficient solution if it is on the linear combination of two extreme supported efficient solutions.

The nondominated objective vectors corresponding to supported, non-supported, extreme supported and nonextreme supported efficient solutions are referred to as supported, non-supported, extreme supported and nonextreme supported nondominated objective vectors, respectively.

We generate the nondominated objective vectors through the solutions of the following objective function:

$w\left(\frac{E-E_{min}}{E_{max}-E_{min}}\right) + (1 - w)\left(\frac{ST-ST_{min}}{ST_{max}-ST_{min}}\right)$ where (E_{max}, ST_{min}) is the extreme nondominated objective vector with the largest E value, whereas (E_{min}, ST_{max}) is the extreme nondominated objective vector with the largest ST value.

According to the above scaling, the weights would be more dispersed. However, the same extreme supported vectors without the scaling would be obtained. We rewrite our scaled objective as $wE_{scaled} + (1 - w)ST_{scaled}$ for the sake of simplicity.

We adapt the method used in Özlen and Azizoğlu (2009) to generate all nondominated objective vectors. As in Özlen and Azizoğlu (2009), we first generate two extreme nondominated objective vectors by setting $w = 0$ and $w = 1$. Then we find a range for w by solving the following relation:

$wE_{scaled}(1) + (1-w)ST_{scaled}(1) = wE_{scaled}(2) + (1-w)ST_{scaled}(2)$ where $E_{scaled}(i)$ is the E_{scaled} value of the i^{th} extreme nondominated objective vector and $ST_{scaled}(i)$ is the ST_{scaled} value of the i^{th} extreme nondominated objective vector.

Rearranging the terms in the equality, we get

$$w = \frac{ST_{scaled}(2) - ST_{scaled}(1)}{ST_{scaled}(2) - ST_{scaled}(1) + E_{scaled}(1) - E_{scaled}(2)}$$

For the above w, the extreme nondominated objective vectors have the objective function value, when the weight is in $[0, w)$, the first extreme point is favored. Otherwise, i.e., when the weight is in $(w, 1]$, the second extreme point is favored. The the following problem is solved to get the third extreme supported nondominated objective vector.

Max $wE_{scaled} + (1-w)ST_{scaled}$

subject to

$x \in X_A$

The optimal solution to the above problem is $(E_{scaled}(3), ST_{scaled}(3))$. We reorder the already found three extreme supported solutions so that $E_{scaled}(1) < E_{scaled}(2) < E_{scaled}(3)$ and $ST_{scaled}(1) < ST_{scaled}(2) < ST_{scaled}(3)$. Then, we solve the below relations:

$$w_1 = \frac{ST_{scaled}(2) - ST_{scaled}(1)}{ST_{scaled}(2) - ST_{scaled}(1) + E_{scaled}(1) - E_{scaled}(2)}$$

$$w_2 = \frac{ST_{scaled}(3) - ST_{scaled}(2)}{ST_{scaled}(3) - ST_{scaled}(2) + E_{scaled}(2) - E_{scaled}(3)}$$

In $[0, w_2], [w_2, w_1], [w_1, 1]$, $(1), (2)$ and (3) are the optimal solutions, respectively.

When a new schedule is added to the i^{th} order, then we find two weights w_t and w_{t+1} by forming two weight equation between S_t and S_{t+1}, such as w_1 and w_2.

The stepwise description of the procedure is stated below:

Procedure 3 Generating Extreme Supported Nondominated Objective Vectors

Step 0. Solve the model in section 4.2.1 to get the extreme nondominated objective vector with the largest E value. Let this optimal solution be (E_{max}, ST_{min}). Solve the model in section 4.2.2 to get the extreme nondominated objective vector with the largest ST value. Let this optimal solution be (E_{min}, ST_{max}).

Let k be the number of extreme supported efficient solutions.

Set $k = 1$

For a current solution (E, ST), define $E_{scaled} = \frac{E - E_{min}}{E_{max} - E_{min}}$ and $ST_{scaled} = \frac{ST - ST_{min}}{ST_{max} - ST_{min}}$.

Find the scalarized weight from the below relation:

$$w_1 = \frac{ST_{scaled}(2) - ST_{scaled}(1)}{ST_{scaled}(2) - ST_{scaled}(1) + E_{scaled}(1) - E_{scaled}(2)}$$

Step 1. Solve Max $w_1 E_{scaled} + (1 - w_1) ST_{scaled}$

subject to

$x \in X_A$

Let $\left(E_{scaled}(k), ST_{scaled}(k)\right)$ be the optimal solution.

If $\left(E_{scaled}(k), ST_{scaled}(k)\right)$ is one of the extreme solutions,

$\left(E_{scaled}(1), ST_{scaled}(1)\right)$ or $\left(E_{scaled}(2), ST_{scaled}(2)\right)$ then go to Step3.

Step 2. If $\left(E_{scaled}(k), ST_{scaled}(k)\right)$ is either $\left(E_{scaled}(k-1), ST_{scaled}(k-1)\right)$ or $\left(E_{scaled}(k-2), ST_{scaled}(k-2)\right)$, fix w_k, let $k = k + 1$, go to Step 1.

If $\left(E_{scaled}(k), ST_{scaled}(k)\right)$ is a new solution, then reorder the solutions, update w_k and w_{k+1} as follows:

$$w_k = \frac{ST_{scaled}(k+1) - ST_{scaled}(k)}{ST_{scaled}(k+1) - ST_{scaled}(k) + E_{scaled}(k) - E_{scaled}(k+1)}$$

$$w_{k+1} = \frac{ST_{scaled}(k+2) - ST_{scaled}(k+1)}{ST_{scaled}(k+2) - ST_{scaled}(k+1) + E_{scaled}(k+1) - E_{scaled}(k+2)}$$

If all w_k values are fixed then go to Step 3, else go to Step 1.

Step 3. Stop, all extreme supported solutions are generated.

The procedure returns an extreme supported nondominated objective vector at each iteration. The returned nondominated objective vector is either a new nondominated objective vector or an already known one. If it is one of the two extreme nondominated objective vectors at the first iteration, then we stop. If it is one of the other known nondominated objective vectors, then its weight is fixed.

We demonstrate Procedure 3 in an example with $n = 75, m = 10$ where a disruption closes two gates randomly. The numeric values of the extreme nondominated objective vectors for this instance are shown in Table 4.2.

Table 4.2 Example for Extreme Supported Nondominated Objective Vectors: $E_{max}, ST_{min}, E_{min}, ST_{max}$

The extreme nondominated objective vector with the largest E value, (1)	
E_{max}	17992
ST_{min}	114174
The extreme nondominated objective vector with the largest ST value, (2)	
E_{min}	16888
ST_{max}	154134

We calculate the scaled objective function values as shown below:

$$E_{scaled}(1) = \frac{E - E_{min}}{E_{max} - E_{min}} = \frac{17992 - 16888}{17992 - 16888} = 1$$

$$E_{scaled}(2) = \frac{E - E_{min}}{E_{max} - E_{min}} = \frac{16888 - 16888}{17992 - 16888} = 0$$

$$ST_{scaled}(1) = \frac{ST - ST_{min}}{ST_{max} - ST_{min}} = \frac{114174 - 114174}{154134 - 114174} = 0$$

$$ST_{scaled}(2) = \frac{ST - ST_{min}}{ST_{max} - ST_{min}} = \frac{154134 - 114174}{154134 - 114174} = 1$$

$$w_1 = \frac{ST_{scaled}(2) - ST_{scaled}(1)}{ST_{scaled}(2) - ST_{scaled}(1) + E_{scaled}(1) - E_{scaled}(2)}$$

$$w_1 = \frac{1-0}{1-0+1-0} = 0.5$$

Then, we solve the following model:

$$\text{Max } 0.5\frac{E-E_{min}}{E_{max}-E_{min}} + (1-0.5)\frac{ST-ST_{min}}{ST_{max}-ST_{min}}$$

subject to

$$x \in X_A$$

The optimal solution is found as $(E^*, ST^*) = (17242, 143982)$ which is said to be the second extreme supported nondominated objective vector when we reorder the extreme supported nondominated objective vectors. Then, we look for new ones employing new weights ranges.

We keep generating new nondominated objective vectors between the two found extreme supported nondominated objective vectors, as long as a new nondominated objective vectors are found in between. When no new nondominated objective vector is found between the two found extreme supported nondominated objective vectors, we move on the next pair of the found extreme supported nondominated objective vectors, thus generating all extreme supported nondominated objective vectors, reported in Table 4.3 and described by Figure 4.1.

Table 4.3 Example for Extreme Supported Nondominated Objective Vectors: weight range, E^*, ST^*

Extreme Supported Nondominated Objective Vector, r	*Weight (w) range*	E^*	ST^*
1	$w = 1$	17992	114174
2	$w = 0$	16888	154134
3	[0,1]	17242	143982
4	[0.5,1]	17842	134136
5	[0.5,0.91694]	17742	137466
6	[0.91694,1]	17942	127512

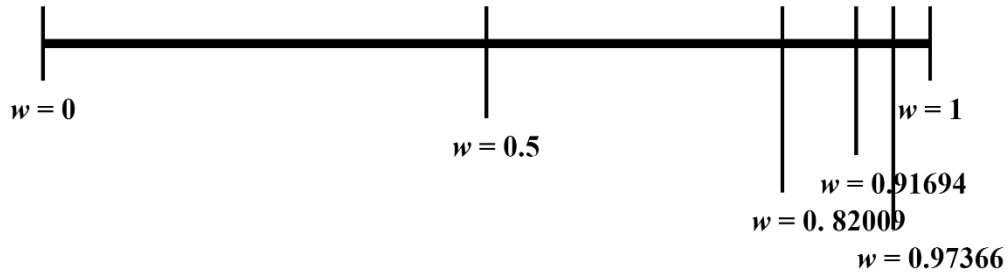

Figure 4.1. Example for Extreme Supported Nondominated Objective Vectors: explored weight ranges

As seen from Figure 4.1, weights between 0 and 0.5 did not produce any new extreme supported nondominated objective vectors. Then, we explored a new weight range, between 0.5 and 1, and found that $w = 0.91694$ produced a new extreme supported nondominated objective vector, after fixing it, we looked for a new one in the range 0.5 and 0.91694 and found that $w = 0.82009$ produced a new extreme supported nondominated objective vector and so on.

4.4 Generating All Nondominated Objective Vectors

Haimes et al. (1971) showed that an optimal solution to the following problem produces an efficient solution:

Max $E + \varepsilon_E ST$

subject to

$ST \leq k$

$x \in X_A$

or equivalently

Max E

subject to

$x \in X_A$

Let E^* be the optimal E value.

Max ST

subject to

$E = E^*$

$x \in X_A$

Using this result, we propose a procedure that generates a nondominated objective vector at each iteration. This procedure uses the fact that (E, ST) is in fact integer.

Procedure 4 Generating All Nondominated Objective Vectors

Input: Initial gate assignment plan, arrival and departure times of the aircraft

Step 0. Solve the following problem to find the extreme nondominated objective vector with the largest E value.

Max E

subject to

$x \in X_A$

Let $E(1)$ be the optimal E value.

Max ST

subject to

$E = E(1)$ and $x \in X_A$

$(E(1), ST(1))$ is the first nondominated objective vector. Set $r = 1$.

Step 1. Solve the following problem

Max ST

subject to

$ST \geq ST(r) + 1$

$x \in X_A$

If the problem is infeasible, go to Step 3.

Step 2. $r = r + 1$

Let $E(r)$ be the optimal solution.

Max ST

subject to

$E = E(r)$ and $x \in X_A$

Let $\mathrm{ST}(r)$ be the optimal solution. $(E(r), ST(r))$ is the r^{th} nondominated objective vector. Go to Step 1.

Step 3. Stop, all r nondominated objective vectors are generated.

Output: All nondominated objective vector and an optimal gate assignment plan corresponding to each vector.

Continuing with the example given in Section 4.3, we present all nondominated objective vectors of this example instance below.

Table 4.4 Example for All Nondominated Objective Vectors: r, E^*, ST^*

Ndov, r	E^*	ST^*	Ndov, r	E^*	ST^*
1	16888	154134	9	17442	140652
2	16988	150822	10	17742	137466
3	17088	150804	11	17792	134154
4	17142	144018	12	17842	134136
5	17188	147492	13	17892	130824
6	17242	143982	14	17942	127512
7	17288	144180	15	17992	114174
8	17342	140724			

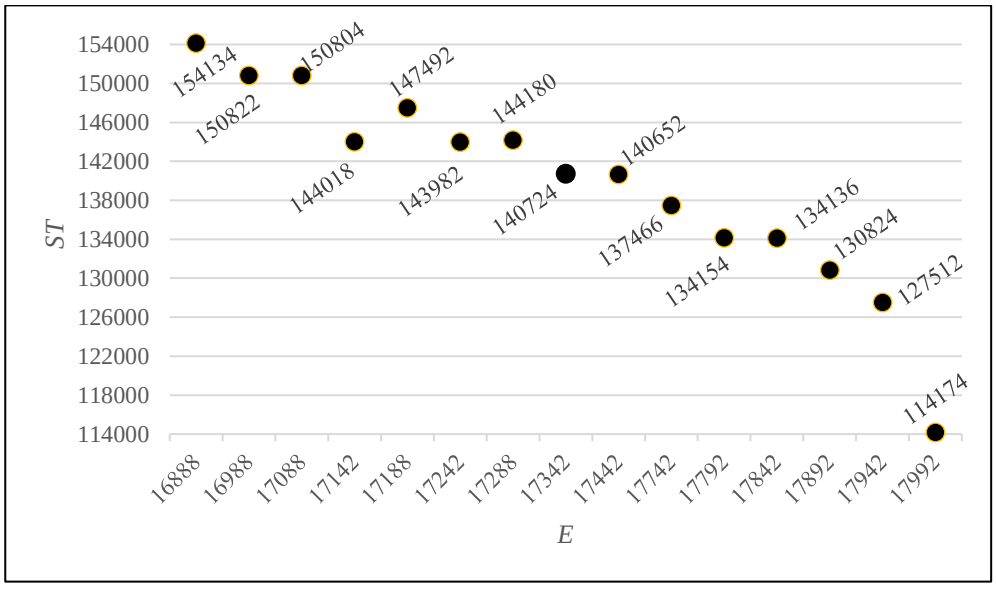

Figure 4.2. Example for All Nondominated Objective Vectors

The trade-off between the two objectives can clearly be seen in Figure 4.2.

4.5 Generating the Approximate Nondominated Objective Vectors

Our experiments have shown that the exact algorithm becomes computationally intractable when the number of aircraft and the number of gates get bigger. This rises a need for a heuristic procedure.

To get approximate set of nondominated objective vectors, we fix some aircraft at some gates and solve the reduced problem. In doing so, we check for the two extreme nondominated objective vectors, each of which obtained in reasonable times.

The similarity of assigned aircraft for each gate between the two extreme nondominated objective vectors is detected. If the aircraft assigned to a particular gate, say gate k, in the first extreme nondominated objective vector (the one having the largest E value) are also asisgned to gate k, in the second extreme solution (the one having the largest ST value), then we fix the aircraft of the latter extreme nondominated objective vector to gate k and reduce the problem by ignoring the fixed aircraft and gate k.

The idea behind these reductions is that if the two aircraft are assigned to the same gate in two extreme, i.e., the furthest, nondominated objective vectors, then it is very likely that those aircraft be assigned to gate k in the nondominated objective vectors in between.

Procedure 5 is the stepwise description of the heuristics procedure.

Procedure 5 Generating the Approximate Nondominated Objective Vectors

Step 0. Find two extreme nondominated objective vectors, $r = 1$ with the largest E and $r = 2$ with the largest ST.

Let S_{kr} be the set of aircraft assigned to gate k in extreme nondominated objective vector r

Let N be the set of all aircraft

Let M be the set of all gates

Set $k = 1$

Step 1. If S_{k2} is superset of S_{k1}, i.e., all aircraft in S_{k1} $are\ also\ in$ S_{k2} at gate k, then $N = N/\{S_{k2}\}$ and $M = M/\{k\}$

Step 2. If $k = m$, then go to Step 3.

$k = k + 1$

Go to Step 1

Step 3. For the reduced problem with gates in M and aircraft in N find all nondominated objective vectors using Procedure 4.

We give a small example with $n = 15, m = 5$ to demonstrate the reduction described in Procedure 5. Let's assume that due to a disruption, gate 4 is closed.

Table 4.5. shows the aircraft-gate assignments for the two extreme nondominated objective vectors.

Table 4.5 Example Reduced Problem for the Approximate Nondominated Objective Vectors

Extreme Solution	*m=1*	*m=2*	*m=3*	*m=4*	*m=5*
with the largest *ST*	1,2,4,13,15	6,8,9	11,12,14	closed	3,5,7,10
With the largest *E*	1,2,4,10,11	6,8,9	12,14		3,5,7,13,15

We see that the list of aircraft assigned to gate 2 in the extreme nondominated objective vector with the largest ST value is in fact a superset of the list of aircraft assigned to gate 2 in the extreme nondominated objective vector with the largest E value. Similarly, this also holds true for the case of gate 3. Then, we fix the assignments of the extreme nondominated objective vector with the largest ST value for gates 2 and 3 and do not include these gates in the reduced problem. A reduced problem is created for the remaining set of aircraft and gates, which consists of gates 1 and 5; and aircraft 1, 2, 3, 4, 5, 7, 10, 13, 15. Procedure 4 will be applied to this reduced problem that is clearly of smaller size.

4.6 Optimal Decomposition

A meaningful question regarding generating an assignment plan would be "What would happen if the problem could be decomposed?". Depending on how many, preferably equal-sized, small instances a problem consists of, obtaining an assignment for each small piece and combining them to get a full solution for the big problem might be a good idea to be explored.

The optimal decomposition of an AGRP would mean that the problem set at hand has time intervals where no aircraft occupies the gates, i.e., in such time intervals no aircraft is present in the system. Thus, at such time intervals, the problem could be decomposed where independent smaller-sized problems are obtained. What should follow would be to solve each decomposed problem and then combine the solutions to get a full assignment plan for the main problem.

We first find the set of nondominated objective vectors for each decomposed problem, using Procedure 4.

We let $A(v)$ be the set of nondominated objective vectors, $v = 1, \dots, V$.

$$A = \bigcup_{v=1}^{V} A(v)$$

EFF_u E value of solution u in A

STA_u STA value of solution u in A

We combine the solution by taking a solution from each decomposed problem and get a complete nondominated objective vector.

To form the complete vectors, we use two models. The models use the following decision variable

$$z_u = \begin{cases} 1 \text{ if solution } u \text{ is selected} \\ 0 \text{ otherwise} \end{cases} \qquad u \in A$$

The combined models are stated as:

$$(P_E) \text{ Max} \sum_{v=1}^{V} \sum_{u \in A(v)} EFF_u z_u$$

subject to

$$\sum_{v=1}^{V} \sum_{u \in A(v)} STA_u z_u \geq t$$

$$\sum_{u \in A(v)} z_u = 1 \qquad v = 1, \ldots, V$$

$$z_u = 0 \text{ or } 1 \qquad u \in A$$

Let EFF^* be the optimal solution.

$$(P_{ST}) \text{ Max} \sum_{v=1}^{V} \sum_{u \in A(v)} STA_u z_u$$

subject to

$$\sum_{v=1}^{V} \sum_{u \in A(v)} EFF_u z_u = EFF^*$$

$$\sum_{u \in A(v)} z_u = 1 \qquad v = 1, \ldots, V$$

$z_u = 0$ or 1 $\quad u \in A$

Let STA^* be the optimal solution.

(EFF^*, STA^*) is a nondominated objective vector once t is between the minimum and maximum stability values of all efficient solutions.

The minimum stability value is $ST_{LB} = \sum_{v=1}^{V} \min_{u \in A}\{STA_u\}$ and maximum stability value $ST_{UB} = \sum_{v=1}^{V} \max_{u \in A}\{STA_u\}$.

The following procedure gives the set of all nondominated objective vectors.

Procedure 6 Generating All Nondominated Objective Vectors with Optimal Decomposition Rule

Step 0. Find set $A(v)$ for all $v = 1, \dots, V$ by using Procedure 4.
Find ST_{LB} and ST_{UB}.
Set $t = ST_{LB}, r = 1$.

Step 1. Solve P_E and then P_{ST}.
The optimal solutions of P_E and P_{ST} (EFF^*, STA^*) gives the r^{th} efficient solution.

Step 2. If $STA^* < ST_{UB}$, then let $t = STA^* + 1, r = r + 1$ and go to Step 2, else stop, all r nondominated objective vectors are generated.

A small example with $n = 15, m = 5$ is given in Table 4.6 that is decomposable into three smaller problems $(r = 3)$.

Table 4.6 Example for Optimal Decomposition: a_i, $d_i, r = 3$,

Aircraft, i	*Arrival time,* a_i	*Departure time,* d_i	Aircraft, i	*Arrival time,* a_i	*Departure time,* d_i
1	41	72	9	163	193
2	33	78	10	145	176
3	55	95	11	280	339
4	52	97	12	300	344
5	14	58	13	286	343
6	138	189	14	277	317
7	174	228	15	287	319
8	165	206			

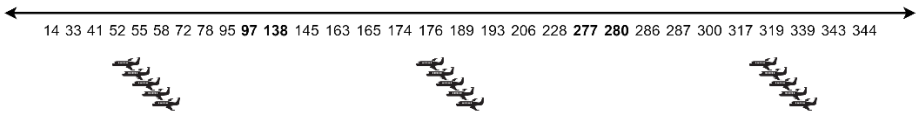

Figure 4.3. Example Time Intervals for a Decomposable Problem

As seen from Figure 4.3, there are no aircraft in the system in the boldface time intervals. Thus, we can decompose our problem into three smaller problems, solve them separately, and apply Procedure 6 obtain the solutions for the original problem.

4.7 Heuristic Implementation of Decomposition Algorithm

Our experiments have revealed a satisfactory behavior of our decomposition algorithm. The basic difficulty is that many instances do not have many intervals with no aircraft, hence the application of the decomposition algorithm may not be possible.

In case there are no intervals with no aircraft, one may use our decomposition-based algorithm to get approximate nondominated objective vectors.

The following procedure may be applied to get such solutions.

Procedure 7 Generating Approximate Nondominated Objective Vectors with Heuristic Implementation of Decomposition Algorithm

Step 0. Find r intervals with minimum number of aircraft

Step 1. Let A be the set of aircraft that are present in at least one of the r intervals.

Step 2. Apply our decomposition algorithm for r individual subproblems and set of $N \backslash A$ aircraft

Step 3. For each nondominated objective vector, insert the aircraft in A.

The future research may point out development of efficient insertion and interchange mechanisms for Step 3.

CHAPTER 5

COMPUTATIONAL EXPERIMENTS

We design an experiment to test the performances of the algorithms. In Section 5.1, the data generation scheme is discussed and in Section 5.2, the performance measures are stated. Section 5.3 analyzes the result of our computational experiment.

5.1 Data Generation Scheme

We select the parameter m compatible with the layouts of airports in Turkey where we make a real-life application. The airports in the largest three cities in Turkey are İstanbul Airport in İstanbul, Esenboğa Airport in Ankara, and İzmir Adnan Menderes Airport in İzmir, and have about 40, 20, and 10 gates, respectively. So, in our experiments, we use m as 40, 20, and 10 gates.

As for the parameter n, we set the number of aircraft, starting from 50 and increase it increments of 25, for each m scenario.

For each aircraft i, arrival time a_i and departure time d_i are generated as stated in Karsu et al. (2021). According to this scheme, the following two sets are defined:

Set I $a_i \sim U[0,300]$

$d_i \sim U[0,30] + 30 + a_i$

Set II $a_i \sim U[0,150]$

$d_i \sim U[0,60] + 60 + a_i$

Set I and Set II represent low and high waiting instances, hence having low and high chances of apron assignments, respectively. Furthermore, the time unit used for the arrival and departure times is minutes. For example, when an aircraft has an arrival

time of 65, it means that this aircraft will be in the airport in the 65th minute of the planning horizon, which is compatible with the time intervals used in real-life.

For each aircraft i, number of passengers p_i are generated as follows.

$p_i \sim T(50,100,300)$ where T is the triangular distribution and 50 is the minimum number of passegers in an aircraft, 100 is the mode, and 300 is the maximum. Note that, we round this random number to an integer value.

We use an initial plan that is optimal for the efficiency measure. Then, we assume that disruptions occur at time zero and the affected gates do not become available thereafter. We define three types of disruption scenarios for our experiments, where affected gates are selected randomly.

Type I – one gate is affected by the disruption

Type II – one fifth of the gates are affected by the disruption

Type III – half of the gates are affected by the disruption

Disruption Type I depicts a small disruption where only a single gate is closed. Disruption Type II shows a more serious case where one fifth of the gates are affected. For the sake of completeness, Disruption Type III depicts the more severe incidents where half the gates become inoperable.

For each n, arrival and departure time set, and disruption type, 10 problem instances are randomly generated. We set a termination limit of two hours for the execution of each mathematical model.

MATLAB is used for random parameter generation and reduced problem creation efforts. All mathematical models and algorithms are developed using ILOG CPLEX Optimization Studio 20.1.0, and solved by CPLEX Optimizer 20.1.0. Furthermore, a computer with quad-core Intel(R) Core(TM) i7-10510U CPU @1.80GHz-2.30 GHz, 16 GB RAM, and Windows 11 is used. Reported CPU times are expressed in seconds.

5.2 Performance Measures

We report the average and maximum (worst case) CPU times for all procedures. We also include the statistics for the number of nondominated objective vectors.

To evaluate the performance of the heuristic algorithm that generates the approximate set of nondominated objective vectors, we first use P, the average of exact nondominated objective vectors found by the heuristic.

We let

$P = \frac{|ES \cap HS|}{|HS|} * 100$ where

ES exact set of nondominated objective vectors

HS approximate set of nondominated objective vectors

$|ES \cap HS|$ number of exact nondominated objective vectors found by heuristic

Hence, P is the percentage of $|ES \cap HS|$ in $|HS|$ (the number of exact nondominated objective vectors)

To evaluate the closeness of the non-exact solutions to their exact counterparts, we use the following two statistics, $D1$ and $D2$, defined in Czyzżak and Jaszkiewicz (1998). They define $D1$ and $D2$ as the average and maximum distance between the exact and heuristic nondominated objective vectors, respectively.

To find $D1$ and $D2$, we assume (E^r, ST^r) is in ES and (E^q, ST^q) is in HS, and calculate the ranges of E values, $R(E)$, and ST values, $R(ST)$, as follows:

$$R(E) = \max_{(E^r, ST^r) \in \mathrm{ES}} E^r - \min_{(E^r, ST^r) \in \mathrm{ES}} E^r$$

$$R(ST) = \max_{(E^r, ST^r) \in \mathrm{ES}} ST^r - \min_{(E^r, ST^r) \in \mathrm{ES}} ST^r$$

$D1$ and $D2$ measures are calculated as follows:

$$f((E^q, ST^q), (E^r, ST^r)) = \max\{0, \frac{1}{R(E)}(E^q - E^r), \frac{1}{ST(E)}(ST^q - ST^r)$$

$$D1 = \frac{1}{|ES|} \sum_{(E^r, ST^r) \in ES} \min_{(E^q, ST^q) \in \text{HS}} \{f\big((E^q, ST^q), (E^r, ST^r)\big)\}$$

$$D2 = \max_{(E^r, ST^r) \in ES} \left\{ \min_{(E^q, ST^q) \in \text{HS}} \{f\big((E^q, ST^q), (E^r, ST^r)\big)\} \right\}$$

In our computational efforts, higher percentage of nondominated objective vectors, P, will be preferred over lower P values. Furthermore, lower average and maximum distance between the exact and heuristic nondominated objective vectors, $D1$ and $D2$ respectively, will be preferred over higher $D1$ and $D2$ values.

5.3 Computational Results

In this section, we discuss the computational results for mathematical models in Chapter 3 and solution algorithms in Chapter 4. Note that an optimal initial plan under our efficiency criterion is used in the following computations.

5.3.1 Comparison of Assignment Based Model and Network Based Model

First and foremost, we compare the performances of mathematical models presented in Chapter 3, namely Assignment Based Model and Network Based Model. Since these two mathematical models are used to obtain feasible assignment plans, we test them on getting an initial assignment plan with our efficiency concern in mind. We make a comparison based on average and maximum CPU times where we run our models for Set 1 and Set 2 i.e., low and high apron usage scenarios, in Table 5.1 and Table 5.2, respectively, and several aircraft-gate pairings, before any disruption in the system.

Table 5.1 Comparison of Assignment and Network Based Models for Set 1

n	*m*	*CPU Time*			
		Assignment Based Model		*Network Flow Based Model*	
		Avg	*Max*	*Avg*	*Max*
50	10	0.16	0.22	0.99	1.61
	20	0.27	0.36	1.55	1.88
	40	0.45	0.53	3.66	4.77
75	10	0.31	0.47	1.87	2.72
	20	0.46	0.58	3.18	4.25
	40	0.84	1.33	6.94	7.95
100	10	0.51	0.92	3.35	4.31
	20	0.72	1.02	5.47	6.44
	40	0.90	1.08	12.15	13.06
125	10	1.00	1.59	4.88	5.52
	20	2.52	4.95	9.77	11.27
	40	1.25	1.30	20.39	22.19
150	10	1.32	1.73	6.12	7.42
	20	4.68	10.23	13.64	16.28
	40	1.71	1.92	25.97	28.80
175	10	1.84	2.80	11.13	14.50
	20	11.59	18.47	28.37	34.66
	40	3.43	5.50	49.92	67.45
200	10	2.31	2.98	9.23	10.11
	20	16.77	20.97	37.52	47.36
	40	6.61	8.98	62.47	70.28

Table 5.2 Comparison of Assignment and Network Based Models for Set 2

n	*m*	*CPU Time*			
		Assignment Based Model		*Network Flow Based Model*	
		Avg	*Max*	*Avg*	*Max*
50	10	0.21	0.41	0.89	1.00
	20	0.41	0.61	1.98	2.34
	40	0.75	0.83	5.19	6.11
75	10	0.38	0.44	1.85	2.09
	20	0.89	1.03	4.27	5.14
	40	1.83	2.16	10.35	11.84
100	10	0.59	0.72	3.24	4.94
	20	1.48	2.55	7.22	7.89
	40	3.15	3.80	19.00	20.52
125	10	0.84	0.92	4.46	4.70
	20	1.82	2.17	10.50	11.47
	40	5.17	6.25	29.30	32.14
150	10	0.99	1.11	6.09	6.94
	20	2.43	2.95	14.72	17.48
	40	6.16	7.58	43.27	53.95
175	10	1.26	1.47	12.66	13.84
	20	2.57	3.63	30.88	33.09
	40	8.26	9.69	77.32	94.20

Table 5.1 and Table 5.2 show that as the number of aircraft increases, so do the CPU times. This is due to the increased complexity with the increases in the problem size.

Table 5.1 demonstrates that as the number of gates increases, generally the CPU time also increases. However, for the larger number of aircraft, we observe that as the number of gates reaches 40, CPU times tend to decrease with the Assignment Based Model in the low apron usage scenario. Furthermore, Table 5.2 demonstrates that an increase in the number of the gates results in higher CPU times for both models in the high apron usage scenario. High apron usage inherently means that the problem setting is in need of more gates. Thus, as m increases, the solution space gets wider, taking longer to solve.

Strikingly, both Table 5.1 and Table 5.2 report lower CPU times for the Assignment Based Model than the Network Based Model. In some instances, the CPU time of the Network Based Model reaches up to ten times that of the Assignment Based Model. For example, in Table 5.1, when n is 200 and m is 40, the average CPU time for Assignment Based Model is reported as 6.61 seconds, whereas for Network Based Model, it is 62.47 seconds. We deduce that the Assignment Based Model is a better fit to our problem. Hence, we continue with the Assignment Based Model in our solution algorithms.

5.3.2 Extreme and Extreme Supported Nondominated Objective Vectors

We discuss the performance of the extreme and extreme supported algorithms given under Procedures 1, 2, and 3 in this section.

Table 5.3 and Table 5.4 report average and maximum CPU times for extreme nondominated objective vector with the largest E value, extreme nondominated objective vector with the largest ST value, extreme supported nondominated objective vectors, and average and maximum number of extreme supported nondominated objective vectors, for Set 1 and Set 2, respectively.

Table 5.3 Extreme Solutions (ES), Extreme Supported Solutions (ESS) for Set 1

Disruption Type	*n*	*m*	*ES-Efficiency*		*ES-Stability*		*ESS*			
			CPU Time				*Number of ESSs*		*CPU Time*	
			Avg	*Max*	*Avg*	*Max*	*Avg*	*Max*	*Avg*	*Max*
1	50	10	0.31	0.41	0.20	0.83	3.6	5	1.05	1.52
		20	0.39	0.47	0.39	0.84	1	1	0.78	1.23
	75	10	0.50	0.66	0.11	0.45	4.6	7	1.62	2.72
		20	0.79	1.41	0.35	0.66	1.4	3	1.43	2.52
	100	10	1.22	1.58	0.03	0.11	4.7	7	3.93	6.11
		20	2.37	3.58	0.31	0.89	4.3	6	5.92	9.36
		40	2.37	3.78	0.78	1.77	1	1	3.15	5.38
	125	10	2.10	2.55	0.04	0.11	4.5	7	5.51	8.05
		20	5.10	8.55	0.19	0.45	4.5	6	10.36	13.56
		40	3.13	3.58	1.26	3.41	1	1	4.39	6.66
	150	10	3.80	4.84	0.07	0.27	4.2	6	8.06	11.36
		20	10.62	16.83	0.15	0.39	6.3	8	23.99	38.67
		40	3.89	4.28	1.12	2.00	1	1	5.01	5.92
	175	10	5.02	7.05	0.12	0.38	4.9	7	10.74	17.53
		20	21.38	30.14	0.10	0.33	5.6	7	36.70	46.56
		40	10.34	15.22	1.90	3.19	2.1	3	16.17	23.42
	200	10	4.07	4.98	0.10	0.47	4.3	7	10.05	17.36
		20	22.87	30.61	0.21	0.67	5.6	8	45.57	55.48
		40	15.93	37.44	1.39	2.33	3.2	4	28.52	55.53
2	50	10	0.29	0.52	0.21	0.47	5.9	8	1.80	2.50
		20	0.39	0.45	0.56	1.28	1	1	0.95	1.66
	75	10	0.55	0.81	0.18	0.53	6.9	12	4.58	10.38
		20	0.88	1.06	0.61	1.22	4.2	7	4.16	5.50
	100	10	0.74	1.00	0.05	0.11	7	8	5.26	7.17
		20	2.32	3.02	0.60	0.88	11.3	16	18.68	29.36
		40	2.33	2.55	2.77	5.31	1	1	5.10	7.69
	125	10	1.52	1.92	0.06	0.19	6.8	9	7.35	11.80
		20	5.10	7.28	0.35	0.80	11.5	19	29.49	49.91
		40	3.36	3.77	3.80	7.08	1	1	7.16	10.23
	150	10	2.21	2.67	0.07	0.22	6.5	10	12.37	21.03
		20	8.10	11.91	0.13	0.28	14.6	18	52.71	68.92
		40	6.06	8.25	3.57	5.30	2.4	6	15.63	35.08
	175	10	3.83	4.50	0.08	0.28	7.6	11	13.81	19.77
		20	14.65	21.55	0.36	0.77	13	17	60.23	71.94
		40	24.97	48.19	3.08	5.13	10.4	18	95.67	217.11

Table 5.4 Extreme Solutions (ES), Extreme Supported Solutions (ESS) for Set 1 (cont'd)

Disruption Type	*n*	*m*	*ES-Efficiency*		*ES-Stability*		*ESS*			
			CPU Time				*Number of ESSs*		*CPU Time*	
			Avg	*Max*	*Avg*	*Max*	*Avg*	*Max*	*Avg*	*Max*
2	200	10	3.59	4.08	0.09	0.30	6.7	10	16.88	28.34
		20	18.13	20.97	0.33	1.28	12.7	17	81.50	111.67
		40	45.41	69.88	2.33	3.41	19.3	24	316.24	701.03
3	50	10	0.33	0.45	0.23	0.77	8.8	14	3.23	6.44
		20	0.52	0.66	0.84	1.37	3.3	5	2.33	3.22
	75	10	0.58	0.73	0.14	0.44	8.6	11	5.47	7.61
		20	0.93	1.13	0.79	1.61	10.7	16	10.58	16.91
	100	10	0.78	0.88	0.10	0.42	9.7	14	8.56	14.20
		20	1.76	2.17	0.45	0.83	15.6	20	25.06	34.81
		40	3.14	4.03	3.60	6.03	3.2	7	11.97	18.58
	125	10	1.10	1.47	0.14	0.63	9.1	13	12.68	17.42
		20	2.80	3.19	0.35	0.63	17.8	21	51.22	71.86
		40	7.18	10.70	4.82	6.69	9.8	15	58.28	98.39
	150	10	1.24	1.38	0.12	0.34	9.4	11	10.28	11.66
		20	3.73	4.14	0.28	0.84	17.9	20	50.75	56.83
		40	11.43	15.95	5.16	7.50	16	19	127.84	244.45
	175	10	2.43	3.00	0.08	0.38	9.7	15	17.36	27.84
		20	7.15	8.44	0.25	0.56	16.1	19	63.57	80.53
		40	35.40	45.08	4.96	8.09	24.2	28	344.99	608.61
	200	10	2.15	2.36	0.07	0.25	8.9	11	15.31	19.58
		20	5.60	6.17	0.16	0.41	17	21	71.47	108.80
		40	33.00	54.36	3.08	5.47	26.1	34	329.08	490.97

Table 5.5 Extreme Solutions (ES), Extreme Supported Solutions (ESS) for Set 2

Disruption Type	*n*	*m*	*ES-Efficiency*		*ES-Stability*		*ESS*			
			CPU Time				*Number of ESSs*		*CPU Time*	
			Avg	*Max*	*Avg*	*Max*	*Avg*	*Max*	*Avg*	*Max*
1	50	10	0.26	0.30	0.05	0.14	2.4	3	0.84	1.45
		20	0.53	0.61	0.07	0.17	2	3	1.30	2.05
	75	10	0.47	0.66	0.05	0.17	2.2	3	1.19	2.03
		20	1.29	1.83	0.15	0.66	2.3	3	2.38	3.31
	100	10	0.76	0.92	0.03	0.09	1.7	3	1.91	3.13
		20	1.75	2.23	0.05	0.19	2.3	4	4.32	6.69
		40	4.28	5.02	0.25	0.97	2.1	3	5.27	8.47
	125	10	1.05	1.16	0.07	0.20	2.2	3	3.04	3.73
		20	2.20	2.53	0.09	0.42	2.1	3	6.64	11.14
		40	6.04	7.69	0.17	0.42	2.3	3	7.90	11.25
	150	10	1.37	1.53	0.08	0.34	1.7	2	4.70	6.16
		20	2.89	3.08	0.10	0.25	2.5	3	12.83	19.83
		40	9.16	11.42	0.23	0.61	2.8	3	11.90	13.64
	175	10	2.02	2.36	0.06	0.27	2	3	5.00	6.23
		20	3.82	4.41	0.10	0.39	2.1	3	25.05	32.38
		40	13.42	15.33	0.27	1.08	2.4	4	26.00	48.58
2	50	10	0.27	0.34	0.04	0.14	3.2	5	1.09	2.00
		20	0.50	0.64	0.19	0.81	4.7	9	2.93	4.81
	75	10	0.48	0.67	0.07	0.25	2.8	4	1.61	3.09
		20	1.06	1.34	0.13	0.56	4.9	7	4.97	7.05
	100	10	0.69	0.78	0.06	0.25	2.9	4	2.13	3.42
		20	1.73	2.02	0.11	0.31	4.8	6	6.54	7.89
		40	4.09	4.97	0.39	1.08	7.6	12	19.00	28.39
	125	10	1.07	1.19	0.07	0.33	2.9	5	3.20	4.83
		20	2.04	2.17	0.06	0.20	5.1	7	11.05	14.81
		40	5.09	5.86	0.15	0.50	7.8	10	28.27	33.23
	150	10	1.58	2.66	0.08	0.38	2.5	4	3.82	4.72
		20	2.75	3.11	0.12	0.39	5.1	7	16.28	22.95
		40	7.02	7.77	0.34	1.11	8.6	12	42.32	62.45
	175	10	1.69	2.02	0.04	0.19	2.4	4	5.75	8.86
		20	3.25	3.91	0.14	0.38	4.3	7	27.29	33.42
		40	12.33	14.94	0.27	0.59	7.5	11	95.33	147.88

Table 5.6 Extreme Solutions (ES), Extreme Supported Solutions (ESS) for Set 2 (cont'd)

Disruption Type	*n*	*m*	*ES-Efficiency*		*ES-Stability*		*ESS*			
			CPU Time				*Number of ESSs*		*CPU Time*	
			Avg	*Max*	*Avg*	*Max*	*Avg*	*Max*	*Avg*	*Max*
3	50	10	0.25	0.28	0.08	0.30	4.1	5	1.37	1.81
		20	0.46	0.58	0.19	0.73	6.3	9	4.42	6.91
	75	10	0.47	0.56	0.04	0.11	3.7	6	2.14	3.61
		20	0.82	1.03	0.13	0.31	7.9	10	6.90	8.83
	100	10	0.61	0.67	0.04	0.14	4.2	6	3.37	4.31
		20	1.33	1.44	0.09	0.19	7.2	9	8.82	12.88
		40	3.03	3.97	0.42	0.94	11.6	14	32.36	49.47
	125	10	0.96	1.11	0.07	0.27	4.1	5	4.13	4.78
		20	1.91	2.25	0.08	0.23	6.9	9	13.24	18.20
		40	4.00	4.50	0.11	0.41	11.4	15	47.86	60.45
	150	10	1.23	1.42	0.05	0.23	4	7	4.71	8.16
		20	2.35	2.58	0.07	0.20	6.8	9	18.39	27.39
		40	5.38	6.14	0.13	0.44	11.4	16	65.88	91.42
	175	10	1.60	2.19	0.07	0.23	3.4	4	5.10	6.16
		20	2.90	3.17	0.07	0.25	6.8	11	21.97	36.78
		40	8.91	10.17	0.21	0.45	10.5	14	85.72	114.05

As expected, both tables demonstrate higher CPU times for the larger number of aircraft. Under both low and high apron usage scenarios, we observe that CPU times for finding the extreme nondominated objective vector with the largest ST value are smaller than those of the extreme nondominated objective vector with the largest E value. With this finding, the importance of using a reduced problem in Procedure 2, is emphasized. Especially, in instances with a larger number of aircraft, the difference between CPU times of the two extreme nondominated objective vectors is more dramatic, in some cases up to 40 times.

We also observe that CPU times for finding the extreme supported nondominated objective vectors are higher for Set 1 and it increases with the number of aircraft in the problem. Based on disruption types, the CPU times increase as the problem

becomes more complex. Disruption type 1 seems to be resulting in a less complex problem than its counterparts, disruption types 2 and 3 where more gates are closed.

Lastly, we observe that the average and maximum number of extreme supported nondominated objective vectors is higher in Set 1 as opposed to Set 2 which is compatible with higher CPU times. As expected, under low apron usage scenario, there exist more solutions compared to the more restrictive high apron usage scenario. To illustrate, when n is 150 and m is 20, the average number of extreme supported nondominated objective vectors is 14.6 (Disruption Type 2) and 17.9 (Disruption Type 3) for Set 1, whereas, for Set 1, it is 5.1 (Disruption Type 2) and 6.8 (Disruption Type 3).

5.3.3 All Nondominated Objective Vectors

Exact solutions are found using Procedure 4 from Chapter 4. We discuss the performance of the exact algorithm that generates all nondominated objective vectors. Table 5.5 and Table 5.6 report the average and maximum number of nondominated objective vectors, average and maximum CPU times, and average CPU time per nondominated objective vector for Set 1 and Set 2, respectively.

Table 5.7 All Exact Nondominated Objective Vectors for Set 1

Disruption Type	*n*	*m*	*Number of ndovs*		*CPU Time*		*Average CPU Time per ndov*
			Avg	*Max*	*Avg*	*Max*	
1	50	10	6.5	12	2.71	7.23	0.42
		20	1	1	0.51	0.83	0.51
	75	10	10.4	21	10.90	39.91	1.05
		20	1.6	3	1.34	2.45	0.84
	100	10	9.5	15	13.36	45.27	1.41
		20	7.1	14	14.89	30.66	2.10
		40	1	1	2.86	4.58	2.86
	125	10	9.2	18	18.84	61.80	2.05
		20	8.3	13	212.02	1671.61	25.54
		40	1	1	5.13	7.83	5.13
2	50	10	11.5	19	7.36	14.00	0.64
		20	1	1	0.75	1.08	0.75
	75	10	18.2	33	26.83	118.42	1.47
		20	6.5	11	8.99	22.69	1.38
	100	10	18	25	23.75	54.30	1.32
		20	32.8	69	452.11	1338.13	13.78
		40	1	1	4.14	4.81	4.14
	125	10	16.1	24	28.23	52.33	1.75
		20	45.2	72	1330.44	5985.94	29.43
		40	1	1	5.17	6.08	5.17
3	50	10	19.2	24	10.78	13.42	0.56
		20	4.2	9	3.33	7.23	0.79
	75	10	32.6	45	30.85	58.70	0.95
		20	26	57	59.60	143.33	2.29
	100	10	32.3	54	39.32	82.72	1.22
		20	52.4	74	589.38	2840.03	11.25
		40	4.3	10	16.84	40.27	3.92
	125	10	30.4	56	39.07	71.34	1.29
		20	76.3	103	792.51	1309.72	10.39
		40	20.8	33	623.93	2850.83	30.00

Table 5.8 All Exact Nondominated Objective Vectors for Set 2

Disruption Type	*n*	*m*	*Number of ndovs*		*CPU Time*		*Average CPU Time per ndov*
			Avg	*Max*	*Avg*	*Max*	
1	50	10	2.8	4	0.85	1.28	0.31
		20	2.8	5	1.75	2.91	0.63
	75	10	2.4	4	1.49	2.88	0.62
		20	2.5	4	2.98	4.48	1.19
	100	10	1.9	5	1.94	5.05	1.02
		20	2.7	7	5.23	11.36	1.94
		40	3.1	5	11.72	17.91	3.78
	125	10	2.9	6	4.38	11.38	1.51
		20	2.3	3	5.47	7.30	2.38
		40	2.7	4	13.98	22.56	5.18
2	50	10	4.2	6	1.78	2.88	0.42
		20	10	19	7.64	13.28	0.76
	75	10	3.6	6	2.18	4.13	0.61
		20	8	12	9.51	14.83	1.19
	100	10	3.4	6	3.12	6.08	0.92
		20	6.1	9	10.55	15.41	1.73
		40	28.4	48	118.32	258.56	4.17
	125	10	4.4	10	5.82	15.36	1.32
		20	7.6	10	15.01	18.23	1.98
		40	13.4	22	57.63	96.55	4.30
3	50	10	5.8	10	2.23	4.59	0.38
		20	12.3	21	8.93	16.06	0.73
	75	10	6	9	3.86	6.97	0.64
		20	13.4	22	12.57	24.61	0.94
	100	10	6.5	13	5.77	12.77	0.89
		20	11.2	16	16.81	26.98	1.50
		40	27.4	37	80.59	123.69	2.94
	125	10	6.3	8	6.91	9.42	1.10
		20	11.8	17	22.25	32.13	1.89
		40	25.3	30	91.52	107.95	3.62

From Table 5.5 and Table 5.6, we observe that the number of nondominated objective vectors is higher in Set 1 than that of Set 2. This is due to the high apron

usage nature of Set 2, there are not as many possible aircraft-gate assignments, hence the solution space is narrower.

Moreover, we observe that CPU times increase as n increases for both sets. This can be attributed to the increase in the complexity of models due to the increase in the problem size.

Set 1 instances are harder to solve than Set 2 instances. This is consistent with the higher number of nondominated objective vectors of Set 1. For example, under disruption type 2, when n is 125 and m is 20, the average (maximum) CPU times are 1330.44 (5985.94) seconds for Set 1, whereas it is 15.01 (18.23) seconds for Set 2. Similarly, under disruption type 3, when n is 125 and m is 20, the average (maximum) CPU times are 623.93 (2850.83) seconds for Set 1, whereas it is 22.25 (32.13) seconds for Set 2. For both Set 1 and Set 2, we also observe that disruption type 2 takes longer to solve than its disruption type 3 counterpart.

Furthermore, the average CPU time per nondominated objective vector is higher for Set 1. Its largest values occur with large number of aircraft and a relatively large number of gates, for example when n is 125 and m is 20. As the problem size increases, so do the solution times for both sets. However, due to its narrower solution space, there are not as many solutions for Set 2. This results in a higher average CPU time per nondominated objective vector values for Set 1.

5.3.4 Approximate Nondominated Objective Vectors

Procedure 5 in Chapter 4, where we take advantage of the closeness of the two extreme nondominated objective vectors to create a reduced problem and unify its solutions with the extreme supported nondominated objective vectors, is implemented on our test instances. Their results are given in Table 5.7 and Table 5.8 in which average and maximum CPU times, average and minimum P, average and maximum $D1$ and $D2$ statistics are reported.

Table 5.9 Heuristic Procedure for Set 1

Disruption Type	*n*	*m*	*CPU Time*		*P*		*D1*		*D2*	
			Avg	*Max*	*Avg*	*Min*	*Avg*	*Max*	*Avg*	*Max*
1	50	10	3.28	8.67	82.85	45.45	0.04	0.25	0.13	0.70
		20	0.78	1.23	100.00	100.00	0.00	0.00	0.00	0.00
	75	10	8.79	26.02	70.36	28.57	0.05	0.22	0.18	0.67
		20	1.72	3.95	100.00	100.00	0.00	0.00	0.00	0.00
	100	10	11.70	30.33	92.33	66.67	0.01	0.10	0.06	0.37
		20	9.03	16.64	88.29	66.67	0.02	0.06	0.07	0.25
		40	3.15	5.38	100.00	100.00	0.00	0.00	0.00	0.00
	125	10	13.18	25.34	85.62	36.36	0.02	0.10	0.08	0.23
		20	19.00	40.22	77.54	30.77	0.03	0.12	0.10	0.25
		40	4.39	6.66	100.00	100.00	0.00	0.00	0.00	0.00
	150	10	16.58	28.72						
		20	101.95	572.41						
		40	5.01	5.92						
2	50	10	8.25	13.86	90.86	75.00	0.01	0.05	0.06	0.25
		20	0.95	1.66	100.00	100.00	0.00	0.00	0.00	0.00
	75	10	17.13	51.70	76.63	21.21	0.02	0.05	0.09	0.25
		20	8.34	20.47	94.11	77.78	0.01	0.07	0.08	0.36
	100	10	29.38	98.44	92.17	50.00	0.00	0.01	0.02	0.07
		20	232.37	1034.02	92.04	33.33	0.00	0.01	0.01	0.07
		40	5.10	7.69	100.00	100.00	0.00	0.00	0.00	0.00
	125	10	19.09	35.22	89.63	50.00	0.01	0.03	0.04	0.17
		20	706.73	4499.83	83.50	52.00	0.00	0.02	0.05	0.11
		40	7.16	10.23	100.00	100.00	0.00	0.00	0.00	0.00
	150	10	32.60	55.31						
		20	1699.25	6493.59						
		40	18.27	46.08						
3	50	10	6.27	21.09	55.86	33.33	0.03	0.06	0.11	0.17
		20	3.50	5.56	92.44	44.44	0.02	0.10	0.05	0.27
	75	10	10.55	35.22	38.19	17.07	0.03	0.05	0.13	0.24
		20	52.58	238.25	76.49	33.33	0.01	0.09	0.05	0.25
	100	10	17.53	75.53	44.97	15.91	0.03	0.05	0.15	0.32
		20	43.99	214.48	36.94	21.67	0.02	0.02	0.07	0.10
		40	18.18	38.55	100.00	100.00	0.00	0.00	0.00	0.00
	125	10	19.22	49.86	44.54	14.29	0.04	0.06	0.14	0.26
		20	286.75	1030.66	62.78	20.41	0.01	0.03	0.05	0.13
		40	761.56	4008.38	95.63	82.14	0.00	0.01	0.01	0.07
	150	10	13.37	40.30						
		20	435.45	2137.91						
		40	7396.45	59410.89						

Table 5.10 Heuristic Procedure for Set 2

Disruption Type	*n*	*m*	*CPU Time*		*P*		*D1*		*D2*	
			Avg	*Max*	*Avg*	*Min*	*Avg*	*Max*	*Avg*	*Max*
1	50	10	1.05	1.89	94.17	66.67	0.03	0.17	0.08	0.50
		20	1.45	2.14	79.00	40.00	0.11	0.35	0.29	0.77
	75	10	1.38	2.19	97.50	75.00	0.01	0.08	0.03	0.33
		20	2.62	3.69	95.00	50.00	0.03	0.25	0.06	0.63
	100	10	2.08	3.69	96.00	60.00	0.01	0.14	0.05	0.49
		20	4.70	7.47	98.57	85.71	0.00	0.00	0.00	0.00
		40	5.49	9.16	75.67	50.00	0.12	0.23	0.37	0.69
	125	10	3.49	4.66	90.00	50.00	0.04	0.17	0.13	0.51
		20	6.95	11.41	93.33	66.67	0.03	0.17	0.10	0.50
		40	8.19	11.64	90.83	66.67	0.03	0.17	0.10	0.50
	150	10	4.98	6.58						
		20	13.19	20.53						
		40	12.90	15.62						
2	50	10	1.85	3.33	96.00	60.00	0.01	0.08	0.03	0.25
		20	6.27	10.77	90.58	68.75	0.00	0.02	0.04	0.24
	75	10	2.27	4.72	97.50	75.00	0.01	0.08	0.03	0.33
		20	8.27	13.58	94.44	66.67	0.01	0.04	0.06	0.25
	100	10	2.77	4.95	94.17	66.67	0.02	0.12	0.08	0.50
		20	9.68	11.78	98.89	88.89	0.00	0.02	0.01	0.14
		40	48.82	108.78	90.94	68.00	0.00	0.01	0.03	0.11
	125	10	5.31	10.41	94.50	70.00	0.01	0.08	0.06	0.33
		20	16.94	22.63	99.00	90.00	0.00	0.01	0.01	0.07
		40	39.35	54.95	99.55	95.45	0.00	0.00	0.00	0.04
	150	10	4.84	7.94						
		20	20.68	27.91						
		40	55.37	87.94						
3	50	10	2.73	5.22	95.71	57.14	0.01	0.09	0.03	0.29
		20	10.44	21.03	99.38	93.75	0.00	0.00	0.00	0.04
	75	10	3.87	7.11	98.00	80.00	0.01	0.06	0.03	0.32
		20	13.77	23.75	93.26	40.91	0.00	0.03	0.01	0.09
	100	10	6.95	12.72	94.53	75.00	0.01	0.03	0.05	0.19
		20	18.42	24.89	98.13	87.50	0.00	0.00	0.00	0.01
		40	66.34	108.67	98.00	80.00	0.00	0.01	0.00	0.03
	125	10	8.23	11.91	96.25	75.00	0.00	0.03	0.03	0.24
		20	25.04	36.39	97.39	88.24	0.00	0.01	0.01	0.11
		40	92.02	110.06	99.55	95.45	0.00	0.00	0.01	0.07
	150	10	8.41	13.84						
		20	29.31	41.13						
		40	101.61	137.23						

As expected, as the problem size increases, the CPU times increase. However, it takes a much shorter time to solve the same set of problems with the heuristic procedure compared to the exact algorithm. This becomes more vivid in the following example: when n is 125 and m is 20 with disruption type 2 for Set 1, the average (maximum) CPU time to generate all nondominated objective vectors is 1330.44 (5985.94) seconds as seen in Table 5.5. The same instances are solved using the heuristic procedure with the average (maximum) CPU time of 706.73 (4499.83) seconds as shown in Table 5.7. We observe a significant time reduction when the heuristic procedure is used for this sizeable problem set. In this reduced time, the average (maximum) of the reported P value is 83.50% (52%), which shows many of the nondominated objective vectors can be generated in a much shorter time. Considering the high average (maximum) number of nondominated objective vectors 45.2 (72), we would be also generating a high number nondominated objective vectors for the decision maker to choose from.

From Table 5.7, for Set 1, we see high P values for disruption type 1 and 2, and satisfactory P values for disruption type 3 when supported with the considerrable CPU time reductions compared to the exact algorithm and considering the high number of nondominated objective vectors in Set 1.

From Table 5.8, for Set 2, we observe high P values for all disruption types, which is consistent with the lower number of nondominated objective vectors in Set 2 compared to the Set 1 as shown in Table 5.5 and Table 5.6. To illustrate, for Set 2, when n is 125 and m is 40 with disruption type 2, on average 99.55% of all nondominated objective vectors are generated using the heuristic procedure. The corresponding average CPU time is 39.35 seconds. On the other hand, for the same instances, the exact algorithm generates 13.4 nondominated objective vectors on average with an average CPU time of 57.63 seconds as shown in Table 5.6. We observe that a good percentage of all nondominated objective vectors can be found within a reduced time by our heuristic procedure.

Table 5.7 and Table 5.8 demonstrate that the heuristic procedure returns very satisfactory $D1$ and $D2$ statistics. Their, in general, very low values show the nondominated objective vectors found by the heuristic procedure are close to the ones found by the exact algorithm.

5.3.5 Optimal Decomposition Rule

For the optimal decomposition rule proposed in Procedure 6 in Section 4.6, we generated new instances where the problem can be decomposed into two ($r = 2$) or three ($r = 3$) small problems where no aircraft pair between the small problems is in the system at the same time interval.

Table 5.9, Table 5.10, Table 5.11, and Table 5.12 report the average and maximum number of nondominated objective vectors, and CPU times without the decomposition rule and with the decomposition rule.

Table 5.9 is prepared for Set 1, with disruption type 2 and the case where the problem is decomposed into two small problems.

Table 5.11 Decomposition Algorithm, Set 1, Disruption Type 2, $r = 2$

n	*m*	*Number of ndovs*		*CPU Time*			
				without decomposition		*with decomposition*	
		Avg	*Max*	*Avg*	*Max*	*Avg*	*Max*
75	10	11.8	19	8.45	13.50	6.09	14.52
	20	12.6	19	21.96	73.92	9.37	20.03
	40	1	1	2.93	3.45	2.14	2.88
100	10	15	24	14.45	23.45	10.70	25.17
	20	23.8	33	76.09	372.23	18.57	29.17
	40	1	1	4.13	5.59	3.28	3.86
125	10	16.4	31	20.07	35.00	11.65	19.91
	20	27.4	37	72.48	103.42	34.93	52.19
	40	1.5	4	7.97	21.00	6.16	14.92

Similarly, we observe a more complex problem with the increased problem size. We also observe as the number of nondominated objective vectors increase, so does the CPU times. From Table 5.9, we directly observe the improvement in CPU times, that decomposing the problem provides.

Table 5.10 is prepared for Set 1, with disruption type 2 and the case where the problem is decomposed into three small problems.

Table 5.12 Decomposition Algorithm, Set 1, Disruption Type 2, $r = 3$

n	*m*	*Number of ndovs*		*CPU Time*			
				without decomposition		*with decomposition*	
		Avg	*Max*	*Avg*	*Max*	*Avg*	*Max*
75	10	13.4	26	9.00	22.19	4.31	8.84
	20	11.4	16	10.16	18.94	4.85	6.27
	40	1	1	1.64	1.81	1.84	2.31
100	10	14.3	23	11.36	20.14	6.64	12.47
	20	16.3	23	23.38	34.91	8.21	11.69
	40	1	1	3.61	4.14	2.49	3.00
125	10	15.4	20	15.43	21.03	5.51	8.34
	20	26.8	48	63.31	138.28	14.84	26.86
	40	4.9	9	17.25	30.17	7.73	12.33

From Table 5.9 and Table 5.10, the effect of r can be clearly observed. Under the same apron usage scenario with the same disruption type, decomposing the problem into either two or three small problems result in further decreased CPU times in general.

Table 5.11 is prepared for Set 1, with disruption type 3 and the case where the problem is decomposed into two small problems.

Table 5.13 Decomposition Algorithm, Set 1, Disruption Type 3, $r = 3$

n	*m*	*Number of ndovs*		*CPU Time*			
				without decomposition		*with decomposition*	
		Avg	*Max*	*Avg*	*Max*	*Avg*	*Max*
75	10	29.4	44	50.11	107.84	5.00	8.56
	20	20.3	25	31.73	46.86	5.52	8.31
	40	1.6	3	4.92	8.13	1.52	2.08
100	10	30.9	65	46.10	110.86	7.31	14.69
	20	43.5	77	162.39	520.70	12.86	21.98
	40	11.1	15	42.48	60.69	7.25	10.38
125	10	25.2	46	41.11	90.72	5.61	7.38
	20	71.8	105	470.60	897.23	25.56	35.83
	40	21.3	31	133.52	187.25	16.11	22.58

From Table 5.10 and Table 5.11, the effect of disruption, i.e., the number of gates closed, can be inferred. As the number of closed gates increases, the average CPU time also increases in Set 1. As a striking example, when n is 100 and m is 20 with disruption type 3 for Set 1, the average (maximum) CPU time is reported as 162.39 (520.70) seconds without using the decomposition rule. However, with the decomposition rule, the average (maximum) CPU time significantly reduces to 12.86 (21.98) seconds.

Table 5.12 is prepared for Set 2, with disruption type 3 and the case where the problem is decomposed into two small problems.

Table 5.14 Decomposition Algorithm, Set 2, Disruption Type 3, $r = 2$

n	*m*	*Number of ndovs*		*CPU Time without decomposition*		*CPU Time with decomposition*	
		Avg	*Max*	*Avg*	*Max*	*Avg*	*Max*
75	10	5.4	7	2.19	3.05	1.09	1.94
	20	9.6	13	6.94	9.41	3.28	4.31
	40	15.9	18	18.00	20.17	8.64	10.17
100	10	4.5	7	2.51	3.67	1.26	1.70
	20	8.7	12	9.53	16.31	4.59	7.59
	40	18.4	21	33.11	44.98	19.78	24.03
125	10	3.7	5	2.79	3.80	1.84	2.31
	20	8.6	10	10.86	13.13	6.74	8.78
	40	17.5	22	46.22	65.77	22.31	28.22

We again observe the improved CPU times with the decomposition rule, this time for the high apron usage scenario that is Set 2.

From Table 5.9, Table 5.10, Table 5.11, and Table 5.12, we deduce that both the average and maximum CPU times are considerably reduced by using with decomposition rule.

CHAPTER 6

CONCLUSIONS AND FURTHER RESEARCH DIRECTIONS

In this thesis, we study an airport gate reassignment problem (AGRP) where the gate disruptions make the initial plan infeasible to implement. After the disruption, we reassign the aircraft to the nonaffected gates to maximize our efficiency and stability criteria.

In our efficiency criterion, we aim to maximize the number of aircraft assigned to gates and the number of passengers in these aircraft. In our stability criterion, we aim to maximize the number of same gate assignments from the initial plan and their number of passengers as well as the number of aircraft assigned to gates. Both the efficiency criterion and the stability criterion are made up of multi-objectives: two objective functions are defined for the efficiency criterion and three for the stability criterion. We formulate the problem with an Assignment Based Model and a Network Based Model.

We first consider the hierarchical optimization, i.e., maximizing the efficiency (stability) measure while keeping the stability (efficiency) value at its maximum level. In doing so, we use the Assignment Based Model whose superiority is shown over the Network Based Model.

We use the Assignment Based Model to generate all extreme supported nondominated objective vectors and all nondominated objective vectors with respect to our efficiency and stability criteria. We make real-life applications for three airports located in the three largest cities in Turkey: İstanbul Airport in İstanbul, Esenboğa Airport in Ankara, and İzmir Adnan Menderes Airport in İzmir, namely.

To generate all nondominated objective vectors, we follow two model-based approaches: optimization and approximation. Our optimization algorithm could solve instances up to 150 aircraft and 40 gates, in less than two hours. With the

approximation algorithm, we handle instances with up to 200 aircraft and 40 gates and report excellent performance results, in terms of solution times and the power of representing the exact nondominated objective vectors.

We develop an optimal decomposition rule that decomposes the problem into subproblems from the time intervals that reside in no aircraft. We find that with the use of the decomposition rule, the problems could be solved in considerably small times. We also discuss the potential heuristic application of the decomposition rule when there are no time intervals with no aircraft.

We anticipate that, in real-life instances, there may be only few cases where our optimal decomposition rule can be used directly. As further research directions, some heuristic approaches that may take our decomposition rule as basis can be developed. We propose first creating a problem that is decomposable by taking out some set of aircraft, applying our decomposition rule to get a new plan, and then considering the set of aircraft that were taken out of the problem through some insertion or exchange heuristics.

As further research directions, we propose some aircraft-gate eligibility constraints, where some gates are reserved for certain airlines. We foresee that such a restriction can be made through defining the assignment decision variables only for the eligible aircraft-gate pairs, hence reduce the complexity of the problem.

Another proposition would be to consider some side-by-side compatibility constraints, where sizes of the aircraft factor into the decision-making process, i.e., two large aircraft cannot be assigned to juxtaposed gates. We foresee that such a restriction can be made through altering existing constraint sets and may increase the problem complexity.

We believe implicit enumeration techniques, such as a branch and bound algorithm, can be designed to generate all nondominated objective vectors, simultaneously as opposed to our sequential generation methods. Moreover, optimization algorithms for a known, however complex utility function can be developed and different

efficiency and stability measures can be tried out. Lastly, we make an emphasis on creating robust gate assignment plans that would reduce the effort spent for gate reassignments.

REFERENCES

Ali, H., Guleria, Y., Alam, S., & Schultz, M. (2019). A passenger-centric model for reducing missed connections at low cost airports with gates reassignment. *IEEE Access, 7*, 179429-179444.

Cai, X., Sun, W., Misir, M., Tan, K. C., Li, X., Xu, T., & Fan, Z. (2019). A bi-objective constrained robust gate assignment problem: Formulation, instances and algorithm. *IEEE transactions on cybernetics, 51*(9), 4488-4500.

Czyzżak, P., & Jaszkiewicz, A. (1998). Pareto simulated annealing—A metaheuristic technique for multiple-objective combinatorial optimization. *J. Multi-Criteria Decis. Anal, 7*(1), 34-47.

Daş, G. S., Gzara, F., & Stützle, T. (2020). A review on airport gate assignment problems: Single versus multi objective approaches. *Omega, 92*, 102146.

Deng, W., Li, B., & Zhao, H. (2017). Study on an airport gate reassignment method and its application. *Symmetry, 9*(11), 258.

Dorndorf, U., Jaehn, F., Lin, C., Ma, H., & Pesch, E. (2007). Disruption management in flight gate scheduling. *Statistica Neerlandica, 61*(1), 92-114.

Dorndorf, U., Jaehn, F., & Pesch, E. (2012). Flight gate scheduling with respect to a reference schedule. *Annals of Operations Research, 194*(1), 177-187.

Gu, Y., & Chung, C. A. (1999). Genetic algorithm approach to aircraft gate reassignment problem. *Journal of Transportation Engineering, 125*(5), 384-389.

Haimes, Y. Y., Lasdon, L. S., & Wismer, D. A. (1971). On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems, Man and Cybernetics, 1*(3), 296–297.

Jaehn, F. (2010). Solving the flight gate assignment problem using dynamic programming. *Zeitschrift für Betriebswirtschaft, 80*(10), 1027-1039.

Karsu, Ö., Azizoğlu, M., & Alanlı, K. (2021). Exact and heuristic solution approaches for the airport gate assignment problem. *Omega, 103*, 102422.

Liu, J., Guo, Z., & Yu, B. (2022). Optimising Gate assignment and taxiway path in a discrete time–space network: integrated model and state analysis. *Transportmetrica B: Transport Dynamics*, 1-23.

Maharjan, B., & Matis, T. I. (2011). An optimization model for gate reassignment in response to flight delays. *Journal of Air Transport Management, 17*(4), 256-261.

Özlen, M., & Azizoğlu, M. (2009). Generating all efficient solutions of a rescheduling problem on unrelated parallel machines. *International Journal of Production Research, 47*(19), 5245-5270.

Pternea, M., & Haghani, A. (2018). Mathematical models for flight-to-gate reassignment with passenger flows: State-of-the-art comparative analysis, formulation improvement, and a new multidimensional assignment model. *Computers & Industrial Engineering, 123*, 103-118.

Pternea, M., & Haghani, A. (2019). An aircraft-to-gate reassignment framework for dealing with schedule disruptions. *Journal of Air Transport Management, 78*, 116-132.

Sahni, S., & Gonzalez, T. (1976). P-complete approximation problems. *Journal of the ACM (JACM), 23*(3), 555-565.

Tang, C. H., Yan, S., & Hou, Y. Z. (2010). A gate reassignment framework for real time flight delays. *4OR, 8*(3), 299-318.

Wang, R., Allignol, C., Barnier, N., Gondran, A., Gotteland, J. B., & Mancel, C. (2022). A new multi-commodity flow model to optimize the robustness of

the Gate Allocation Problem. *Transportation Research Part C: Emerging Technologies, 136,* 103491.

Wang, H., Luo, Y., & Shi, Z. (2013). Real-time gate reassignment based on flight delay feature in hub airport. *Mathematical Problems in Engineering, 2013*, 10.

Yan, S., & Chang, C. M. (1998). A network model for gate assignment. *Journal of advanced Transportation*, *32*(2), 176-189.

Yan, S., Chen, C. Y., & Tang, C. H. (2009). Airport gate reassignment following temporary airport closures. *Transportmetrica*, *5*(1), 25-41.

Yan, S., Tang, C. H., & Hou, Y. Z. (2011). Airport gate reassignments considering deterministic and stochastic flight departure/arrival times. *Journal of advanced transportation*, *45*(4), 304-320.

Yu, C., Zhang, D., & Lau, H. Y. (2017). An adaptive large neighborhood search heuristic for solving a robust gate assignment problem. *Expert Systems with Applications, 84,* 143-154.

Zhang, D., & Klabjan, D. (2017). Optimization for gate re-assignment. *Transportation Research Part B: Methodological, 95,* 260-284.