

**MARMARA UNIVERSITY
INSTITUTE FOR GRADUATE STUDIES IN
PURE AND APPLIED SCIENCES**

**IMPLEMENTATION OF AIRPORT GATE
ASSIGNMENT PROBLEM**

Canan ERSAN

**THESIS
FOR THE DEGREE OF MASTER OF SCIENCE
IN
INDUSTRIAL ENGINEERING**

**SUPERVISER
Prof. Dr. M. Akif Eyler**

İSTANBUL, May 2010

**MARMARA UNIVERSITY
INSTITUTE FOR GRADUATE STUDIES IN
PURE AND APPLIED SCIENCES**

**IMPLEMENTATION OF AIRPORT GATE
ASSIGNMENT PROBLEM**

**Canan ERSAN
(141100920070286)**

**THESIS
FOR THE DEGREE OF MASTER OF SCIENCE
IN
INDUSTRIAL ENGINEERING**

**SUPERVISER
Prof. Dr. M. Akif Eyler**

İSTANBUL, May 2010

ACKNOWLEDGEMENT

I have worked, during my graduate study, with people whose contribution in assorted ways to the research. It is a pleasure to convey my gratitude to them all in my acknowledgment.

In the first place I would like to record my gratitude to Prof. Dr. M. Akif Eyller for his supervision, advice, and guidance from the very early stage of this research as well as believing in me, inspiring me and providing encouragement throughout this thesis. I am indebted to him for making the coding process much easier for me.

I must acknowledge the invaluable contributions of Assist. Prof. Dr. Serol Bulkan, his helpfulness, endless patience in answering my questions and providing constructive comments throughout my thesis. Additionally, I want to thank TÜBİTAK-BİDEB Değerlendirme ve Destekleme Kurulu for the support of my graduate study.

Lastly, I offer my regards to all of those who supported me in any respect during the completion of the project.

TABLE OF CONTENTS

ACKNOWLEDGEMENT	i
ÖZET.....	v
ABSTRACT.....	vi
SYMBOLS.....	vii
ABBREVIATIONS.....	viii
FIGURES.....	ix
TABLES.....	x
CHAPTER I INTRODUCTION AND OBJECTIVES	1
I.1. INTRODUCTION	1
I.2. OBJECTIVES.....	2
CHAPTER II OPTIMIZATION PROBLEMS IN THE	
AIRLINE INDUSTRY.....	3
II.1 OPTIMIZATION PROBLEMS AT THE AIRPORT	3
II.2 TYPES OF OPTIMIZATION PROBLEMS AT THE AIRPORT	3
II.2.1 Mobile Resource Optimization Problems	4
II.2.1.1 Crew optimization problem	4
II.2.1.2 Vehicle optimization problem.....	4
II.2.2 Immobile Resource Optimization Problems	5
II.2.2.1 Gate assignment problem.....	5
II.2.2.2 Carousel optimization problem.....	6
CHAPTER III GATE ASSIGNMENT PROBLEM:	
MATHEMATICAL PROGRAMMING	
TECHNIQUES	7
III.1 PROBLEM FORMULATION AND EXACT SOLUTION	
ALGORITHMS.....	7
III.1.1 Problem Formulation.....	7
III.1.2 Exact Solution Models	9
III.2 HEURISTIC ALGORITHMS.....	10

III.2.1 Tabu Search	10
III.2.2 Genetic Algorithms	13
III.2.3 Ant Systems.....	14
III.2.4 Simulated Annealing Methods	15
CHAPTER IV GATE ASSIGNMENT PROBLEM: SIMULATION AND RULE BASED EXPERT SYSTEMS	17
IV SIMULATION AND RULE BASED EXPERT SYSTEMS	17
IV.1 Simulation.....	17
IV.2 Rule Based Expert Systems	18
CHAPTER V GATE ASSIGNMENT PROBLEM: IMPLEMENTATION	20
V.1 PURPOSE OF IMPLEMENTATION	20
V.2 METHODS AND METHODOLOGY.....	20
V.3 PROBLEM DEFINITION: ISTANBUL ATATURK AIRPORT	20
V.3.1 Objective Function	21
V.3.2 Data	23
V.3.3 Constraints.....	27
CHAPTER VI GATE ASSIGNMENT PROBLEM: INITIAL SOLUTION.....	28
VI.1 EVOLUTION OF COST	28
VI.2 METHODS FOR INITIAL SOLUTION	31
VI.2.1 Greedy Algorithm.....	31
VI.2.1.1 The Elements of Greedy Algorithm	33
VI.2.1.2 Activity Selection Problem.....	33
VI.2.1.3 Gate Assignment Problem.....	35
VI.2.1 Sorting According to Departure Time	36
VI.2.2 First Come First Served Model.....	38
VI.2.3 Longest Processing Time Algorithm	39
VI.2.4 Results	40
CHAPTER VII TABU SEARCH META HEURISTICS	42
VII.1 NEIGHBORHOOD SEARCH METHODS	42

VII.1.1 Interval Exchange Move	43
VII.1.2 Apron Exchange Move	48
VII.1.3 Insert and Remove Move	50
VII.2 RESULTS	53
CHAPTER VIII CONCLUSION AND FUTURE RESEARCH...	57
VIII.1 CONCLUSION	57
VIII.2 FUTURE RESEARCH	58
REFERENCES	59
CURRICULUM VITAE	64

ÖZET

BİR HAVALİMANINDA KAPI ATAMA PROBLEMİ UYGULAMASI

Havaalanı işletmeciliği son derece hizmet yoğun bir faaliyettir. Havacılığın genel olarak yüksek profilde bir müşteri kitlesine sahip olması nedeniyle yüksek kalite ve performans beklentisi vardır. Bu beklenti; zamanında, hatasız, etkin maliyetli çözümler gerektirmektedir.

Bu çalışma ile bir havalimanında en önemli maliyet unsuru kabul edilen geliş ve gidişlerde uçakların kapı yanaşmalarının optimize edilmesini sağlayan Kapı Atama Problemi'nin çözümüne yönelik bir uygulamanın ortaya çıkarılması amaçlanmıştır.

Bu çalışmada, havalimanındaki mobil olmayan kaynaklardan biri olan kapıların kullanımlarının eniyilemesi ile ilgili literatürde yapılan çalışmalar ve bu çalışmalarda kullanılan çözüm metotları, araçlar ve teknikler incelenmiştir.

Kapı atama problemi tanımlanırken Atatürk havalimanındaki gerçek maliyet unsurlarından yola çıkılarak 'kapıların kullanım oranının' artırılmasının birincil amaç fonksiyonu olmasına karar verilmiştir. En uygun başlangıç çözümünün bulunması için farklı algoritmalar tasarlanmıştır. Bu algoritmalar ile elde edilen sonuçlar üzerinde tabu arama uygulanmıştır.

Son olarak, Atatürk Havalimanı Dış Hatlar Terminali'nden temin edilen veri üzerinde algoritma ve teknikler test edilmiştir. Sonuçlar üzerinde test ve analizler yapılmıştır.

ABSTRACT

IMPLEMENTATION OF AIRPORT GATE ASSIGNMENT PROBLEM

Today, airport industry is mostly a service activity. As a result of the airport industry's high value customer profile, there is an expectation of high quality and performance. Therefore, it can be said that the airport industry needs cost efficient, errorless and on time solutions.

The gate assignment problem tries to minimize the most important cost in airline industry, flight parking at gates while landing or departing. This study attempts to implement and solves the gate assignment problem at a real airport company.

In this study, the researches about the utilization of gates, one of the immobile resources in the airport, in the literature and the tools, techniques, solution methods used in the studies are analyzed.

While defining the Airport Gate Assignment Problem it is determined that the maximization of gate utilization rate should be main objective function. Some algorithms are designed to find the most appropriate method for initial solution. Moreover, tabu search algorithm used on this feasible solution.

Finally, the implementation is run on the real data taken from International Terminal of Atatürk Airport. Tests and analysis conducted on these results.

SYMBOLS

a_i	: Arrival time of flight i
d_i	: Departure time of flight i
f_{ij}	: Number of passengers transferring from flight i to flight j
n	: Total number of flights
N	: Set of flights arriving at and/or departing from the airport
m	: Total number of gates
$m + 1$: Represents remote stand area (apron)
M	: Set of gates available at the airport
$t_{i,m+1}$: Ground time period of flight i assigned to gate $m + 1$ (apron)
w_{kl}	: Walking distance for passengers from gate k to gate l
y_{ij}	: Binary variable: If the flight i is assigned to gate k , its value equals 1, else 0.

ABBREVIATIONS

ACI	: Airport Council International
ACS	: Ant Colony System
AGAP	: Airport Gate Assignment Problem
DI	: The Aircraft Arrives as Domestic, but Leaves as International Flight
FCFS	: First-Come-First-Served
GA	: Genetic Algorithm
HAS-AGAP	: Hybrid Ant System for AGAP
II	: The Aircraft Arrives and Leaves as International Flight
IP	: Integer Programming
ITS	: Interval Exchange Tabu Search
LP	: Linear Programming
LPT	: Longest Processing Time
LS	: Local Search
MOGAP	: Multi Objective Gate Assignment Problem
QAP	: Quadratic Assignment Problem
SA	: Simulated Annealing
SADT	: Sorting According to Departure Time
TAGTA	: Total Aircraft Ground Time at Apron
TPWD	: Total Passenger Walking Distance
TS	: Tabu Search
XTS	: TS Heuristics of Xu and Bailey

FIGURES

	<u>PAGE NO</u>
Figure V.1 Gate Assignment Solution Chart of TAV at 10.01.2009	25
Figure V.2 The Layout of International Terminal of Atatürk Airport	26
Figure VI.1 Assignment of Small Sample Data.....	29
Figure VI.2 Greedy Algorithm Structure.....	32
Figure VI.3 <i>CompareTo</i> Method for Sorting Flights.....	36
Figure VII.1 Illustration of Interval Exchange Move.....	44
Figure VII.2 Four Time Points of an Interval.....	44
Figure VII.3 Gantt Chart – Gate Assignment of Sample Data	45
Figure VII.4 t1, t2, t3 and t4 Time Points of Current Interval A.....	46
Figure VII.5 t1, t2, t3 and t4 Time Points of Current Interval B.....	46
Figure VII.6 Two Gates before the Interval Exchange Move	48
Figure VII.7 Pseudocode of Algorithm I.....	49
Figure VII.8 Pseudocode of Algorithm II.....	51

TABLES

	<u>PAGE NO</u>
Table V.1 Flight Classes According to Aircraft-Gate Size Compatibility.....	24
Table V.2 The Sample of Flight Schedule Time Table.....	25
Table VI.1 List of the Activities Ranked According to Increased Order Finishing Time	34
Table VI.2 The Comparison of SADT and FCFS Principle Solutions	39
Table VI.3 Gate Utilization Rates of Three Greedy Algorithms	41
Table VII.1 The Results of Real Flight Data of AHL with SADT Algorithm and Tabu Technique	54
Table VII.2 The Results of Real Flight Data of AHL with FCFS Algorithm and Tabu Technique	54
Table VII.3 The Results of Real Flight Data of AHL with LPT Algorithm and Tabu Technique.....	55

CHAPTER I

INTRODUCTION AND OBJECTIVES

I.1. INTRODUCTION

Air transportation continues its rapid growth all over the world. According to the reports taken from Airport Council International (ACI) member airports, total worldwide passenger traffic reached an all time high in 2007, increasing by 6.9 percent over 2006. The 1200 member airports of ACI welcomed 4.8 billion passengers, processed 88.5 million metric tons of cargo and 76.4 million aircraft movements (ACI, 2007).

The rapid growth of air traffic with periodic fluctuations is forcing airports to both expand their capacities and use their existing capacity more efficiently. If the physical improvement of capacity is impossible, valuable or scarce resources should be used effectively. The gates of airports are one of these scarce resources. How to efficiently allocate gates at airports to coming or outgoing flights has become one of the most important problems which managers of airlines and airports have to concern about. Basically, this problem is a type of scheduling problem. However, by considering operational constraints of airports and airlines companies, more complicated problems than most other traditional scheduling problems are faced by managers (Dorndorf, 2005).

Assigning flights to gates is an important problem because there are several issues, like arrival and ground time of flights, aircraft size and other geometrical considerations, service requirements, passenger transfer pattern and the walking distance in a terminal building, flight crew and aircraft rotations, regulations and restrictions for international flights, and additional marketing and management directions, are needed to be addressed (Bolat, 1999).

Various techniques have been applied to solve gate assignment problem. Linear binary programming (Babic et. al., 1984), 0–1 linear programming (Bihr, 1990), genetic algorithm (Gu and Chung, 1999), mixed 0-1 quadratic integer programming and tabu search (Xu and Bailey, 2001), multi-objective programming (Yan and Huo, 2001) and stochastic programming (Lim and Wang, 2001) are some examples of these techniques.

While some of the approaches in the literature are meeting with success at the stage of implementation, some others face implementation problems because of unexpected situations in practice as well as unconsidered or understated constraints.

It is important to applicability of the methods as well as improvement of them, which provide assigning flights to gates efficiently. Providing optimization of both passenger satisfactory and the cost of airline companies during the whole operational process, also providing smooth flow of passengers have considerable important, so they are part of the objectives.

I.2. OBJECTIVES

The main goal of this study is deciding the appropriate method according to the needs and requirements of the real airport company by evaluating the solving approaches to gate assignment problem in the literature and also implementing this method.

CHAPTER II

OPTIMIZATION PROBLEMS IN THE AIRLINE INDUSTRY

II.1 OPTIMIZATION PROBLEMS AT THE AIRPORT

Between the arrival time of an aircraft and its departure time lots of work should be done. Transferring of passengers from aircraft to exit is one of the most obvious works. Additionally, passengers' baggage have to circulate, the aircraft needs to be refueled, new passengers need to be checked and boarded, new supplies have to be put on board, the aircraft has to get cleaned. All of the actions take place while the aircraft is standing at a gate and even after the departure of aircraft (Diepen, et. al., 2008).

At an airport a series of assignment problems need to be solved before aircraft can arrive and depart and passengers can embark and disembark. A lot of different parties are involved within this, each of which has their own objectives and constraints (Diepen, et. al., 2008).

II.2 TYPES OF OPTIMIZATION PROBLEMS AT THE AIRPORT

The airport infrastructure has a capacity limitation which is influenced by many factors. The goal must be to utilize all resources (check-in desks, departure lounges, gates, aircraft stands, baggage carousels, staff and equipments) to their maximal extent. This guarantees low unit costs and high profit potential (Kelemen, 2005).

There are two main types of optimization problems in the airline industry; mobile and immobile resource optimization problems.

II.2.1 Mobile Resource Optimization Problems

Managing airport resources is fundamental for the successful management of an international airport. The best solution is to develop a well-structured Resource Management System. A Resource Management System is the solution for the efficient management of handling mobile and immobile resources, what could be a first major step to the integration of a central airport operational database system (Kelemen, 2005).

II.2.1.1 Crew optimization problem

Crew assignment, constitute one of two phases of crew scheduling problem. First crew pairing problem is solved and good pairings are found by giving flight schedule (timetable) as an input. After solving crew pairing problem, crews are assigned to these pairings. The goal of crew scheduling is to allocate crew members, pilots and flight attendants to individual flights (Özdemir, 2009).

Crew scheduling problem aims to minimize the total cost and maximize robustness and suitability of pairings for roster construction, also crew satisfaction under various constraints. Main constraints of crew optimization problem are legality rules for individual pairings: e.g. maximum duty time, and distribution between crew bases. Moreover, this problem has to take into account crew preferences: e.g. pilot usually have their preferences to specific flights, vacations, training and experience constraints which make problem more complex (Qi, et. al., 2004).

For solving crew scheduling problem branch and bound, column generation, tabu search, Lagrangian relaxation, and Dantzig-Wolfe decomposition techniques and tools are used (Nowak, 2008).

II.2.1.2 Vehicle optimization problem

Some flights are not assigned to a gate with an air bridge but to a remote stand. This implies that passengers have to be transported to and from the aircraft by buses. At

this point which bus will transport which passengers to or from the aircraft; in other words bus optimization problem emerges (Diepen, et. al., 2008).

In the simplest form the assignment problem can be formulated in terms of linear programming and solved with a help of simplex method, network algorithms or assignment method (Zak, et. al., 2009).

On the other hand, in real life situations the researches on integration of different assignment problems have been performed. Some of the models combine gate assignment, some integrate fleet assignment, and some combine fleet sizing with crew assignment (Diepen, et. al., 2008).

II.2.2 Immobile Resource Optimization Problems

Two of the immobile resource optimization problems are gate assignment, and carousel optimization problems. Check in counter and baggage chute optimization problems are also that type of problems, however they are easy to solve for most of the airport industries.

II.2.2.1 Gate assignment problem

The gate assignment problem deals with assigning a given set of flights to a set of gates while meeting operational requirements and maximizing the benefits of airports, airlines and passengers (Bolat, 1999).

Assigning flights to gates is an outstanding matter, because there are several considerations, like arrival and ground time of flights, aircraft size and other geometrical issues, service requirements, passenger transfer pattern and the walking distance in a terminal building, flight crew and aircraft rotations, regulations and restrictions for international flights, and additional marketing and management directions (Bolat, 1999).

All these requirements make the Gate Assignment Problem (GAP) very complicated both from a theoretical and a practical point of view. In fact, any practical GAP

instance for a big international airport usually has to deal with a large number of daily flights (around 800) which have to be assigned to a large number (around 100) of different gates. This means that the model with standard linearization techniques has around 64×10^8 variables, and moreover, both a quadratic objective and quadratic constraints make the problem computationally intractable for any mixed-integer programming solver. This huge size means that the design of an efficient heuristic is of considerable interest. Additionally, the multiple criteria and multiple constraints nature of the problem make it very unlikely that a so-called ideal optimal solution which simultaneously optimizes all objectives can be found and verified. Therefore, one has to determine a solution that provides an appropriate compromise between all the different objectives while ensuring a set of hard constraints (Drexler and Nikulin, 2008).

Airport Gate Assignment Problem (AGAP) can be classified according to the main objectives considered. Typical objectives are minimizing the number of un-gated aircraft, minimizing total walking distance for passengers and total transport distance for baggage, maximizing preferences of airline for particular gates and maximizing the robustness of the assignment (Dorndorf, 2005).

II.2.2.2 Carousel optimization problem

A baggage carousel delivers checked luggage to the passengers at the baggage claim area at their final destination. Because baggage delivery speed affects the customer satisfaction, handling baggage carousels is an important optimization problem. The constraints are the capacity of load of baggage carousels, varying number of arriving passengers on flights, airline preferences and locations of stands and gates at terminal buildings. The goals of the problem are providing improved quality of service, minimizing waiting time and the energy to operate the assembly line (Kelemen, 2005).

CHAPTER III

GATE ASSIGNMENT PROBLEM: MATHEMATICAL PROGRAMMING TECHNIQUES

III.1 PROBLEM FORMULATION AND EXACT SOLUTION ALGORITHMS

One of the two main research streams developed in gate assignment problem is mathematical programming techniques.

Exact solution algorithms had used firstly to find solutions to gate assignment problem. However, due to the overwhelming complexity of these models, variety of heuristics was improved (Dorndorf, 2005).

III.1.1 Problem Formulation

The basic input data for gate scheduling is a flight time table with arrival and departure times and additional specifications of flights: the origin and destination of a flight, the type of aircraft, the number of passengers, the cargo volume, the type of flight (domestic or international) as well as the gate preferences, the required airport services and the inspection facilities (Drexl and Nikulin, 2008).

GAP can be modeled as a quadratic assignment problem, linear integer program or mixed integer programming. Variety of optimization function can be used as main objectives of that assignment problem. Minimizing total walking distances of passengers, total passenger delay, number of flights cancellations, and range of unutilized time periods of gates are some of objectives.

There are two traditional goals of GAP, minimizing the total passenger walking or baggage transportation distance and assigning flights to terminal gates. These goals have been addressed together by Ding et. al. (2004).

Distances from check into gates for originating passengers, from gates to baggage claim areas for destination passengers and from gate to gate for transfer passengers are considered. In addition to these they include the distance from the apron to the terminal. The numbers of passengers, who embark, disembark and transfer is also important to calculate the total passenger walking distance (Ding, et. al., 2004).

The notations and their definitions are given to understand the model in the paper as follows:

N set of flights arriving at and/or departing from the airport

M set of gates available at the airport

n total number of flights ($|N|$, where $|N|$ denotes the cardinality of N)

m total number of gates, that is $|M|$

a_i arrival time of flight i ($1 \leq i \leq n$)

d_i departure time of flight i ($1 \leq i \leq n$)

w_{kl} walking distance for passengers from gate k to gate l ($1 \leq k, l \leq m$)

f_{ij} number of passengers transferring from flight i to flight j ($1 \leq i, j \leq n$)

For their model Ding et. al. (2004) add two dummy gates; gate 0 for representing entrance or exit of the airport and gate $(m + 1)$ for representing the apron where flights arrive at when no gates are available.

y_{ik} is a binary variable, the value of y_{ik} is 1 when the flight i is assigned to gate k ($0 < k \leq m + 1$), and 0 otherwise.

There is a constraint to prevent assigning any two flights to the same gate;

$y_{ik} = y_{jk} = 1 (k \neq m + 1)$ implies $a_i > d_j$ or $a_j > d_i$ ($1 \leq i, j \leq n$)

The Airport Gate Assignment Problem is expressed by Ding et. al. (2004) as follows;

minimize

$$\sum_{i=1}^n y_{i,m+1} \quad (\text{III. 1})$$

$$\sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^{m+1} \sum_{l=1}^{m+1} f_{ij} w_{kl} y_{ik} y_{jl} + \sum_{i=1}^n f_{0i} w_{0i} + \sum_{i=1}^n f_{i0} \quad (\text{III. 2})$$

subject to:

$$\sum_{i=0}^n y_{ik} = 1 \quad (1 \leq i \leq n) \quad (\text{III. 3})$$

$$a_i \leq d_i \quad (1 \leq i \leq n) \quad (\text{III. 4})$$

$$y_{ik} y_{jk} (d_j - a_i)(d_i - a_j) \leq 0 \quad (1 \leq i, j \leq n, k \neq m + 1) \quad (\text{III. 5})$$

$$y_{ik} \in \{0,1\} \quad (1 \leq i \leq n, l \leq k \leq m + 1) \quad (\text{III. 6})$$

First constraint (III.3) ensures that every flight can be assigned to one and only one gate or assigned to the apron. Second constraint (III.4) says that each flight's arrival time is before than its departure time. Third constraint (III.5) specifies that two flights' schedule at the same gate cannot overlap (Ding, et. al., 2004).

III.1.2 Exact Solution Models

Some solution techniques which have been used in assignment problems are dynamic programming, cutting plane techniques, and branch and bound procedures.

Babic et. al. (1984) improve a method to find aircraft stand assignment to minimize average total walking distance. Their research is one of the papers which imply branch and bound algorithm.

Compared to random aircraft stand assignment there is a decrease in the number of passengers walking the longest distances and vice versa, in the situation that the number of passengers varies enough (Babic, et. al., 1984).

Bolat (1999) proposes a mathematical model to utilize the available gates, while satisfying physical and managerial considerations.

The goals of the work are reaching optimal or near optimal solutions, providing robust model to absorb minor deviations, and the adaption of procedures to the real time operations support (Bolat, 1999).

Distributing the slack times, idle period between two utilization of gate, uniformly is essential for increasing the robustness of assignments. While longer slack time increases robustness, it causes decrease in utilization. On the other hand shorter slack time increases domino effect, in other words the deviations can disrupt initial assignment. Therefore, according to Bolat (1999), minimizing the range of slack times is clearly preferable.

The model firstly propose an optimum branch and bound procedure, then lower bounding scheme for detecting unpromising solutions as early as possible is constructed. In the experimental tests, it is found that the performance of the optimum algorithm is affected by the utilization level of gates (Bolat, 1999).

III.2 HEURISTIC ALGORITHMS

The extreme difficulty of GAP is made it an ideal problem for the development of heuristic search methods. Tabu search, genetic algorithms, ant systems, simulated annealing and other specialized methods are all applied to GAP. The performance of different heuristics tends to vary with certain problem characteristics.

III.2.1 Tabu Search

Tabu search is a mathematical optimization method, belonging to the class of local search techniques. Tabu search enhances the performance of a local search method by using memory structures: once a potential solution has been determined, it is marked as "taboo" so that the algorithm does not visit that possibility repeatedly. Tabu search is attributed to Fred Glover.

The basic elements of tabu search algorithm are setting feasible initial solution, moving mechanism, candidate list strategies, memory, criteria to remove tabu, and stopping conditions (Güden, et. al., 2005).

The most widely applied feature of TS is the use of a short term memory to escape from local minima. TS typically uses an aggressive local search that in each step tries to make the best possible move from s to a neighboring solution s' even if that move worsens the objective function value. To prevent the local search to immediately return to a previously visited solution and to avoid cycling, in TS moves to recently visited solutions are forbidden. This can be implemented by explicitly memorizing previously visited solutions and forbidding moving to those (Stützle, 1998).

Tabu search technique has been applied to many problems in different applications. Some of the application areas are scheduling, design, technology, graph optimization, general combinational optimization and routing. Quadratic Assignment Problem, Travelling Salesman Problems, Graph Coloring, Graph Partitioning, Job Shop Scheduling, Telecommunications Path Assignment are form a partial list of applications (Glover and Laguna, 2005).

The performance of tabu search algorithms depends very much on the size of the tabu list and on the way this list is handled (Burkard and Çela, 1996).

Among the application of tabu search algorithm there is AGAP problem too. Xu and Bailey (2001) develop a tabu search based heuristic for the AGAP problem in 2001. After formulating the problem as a mixed 0-1 integer problem with a linear objective function and constraints, they propose tabu search algorithm with dynamic tabu tenure and aspiration criterion.

Xu and Bailey (2001) show their TS heuristic achieve savings on passengers' connection time (average saving for seven test problems is 24.7%) compared with the static gate assignment in current airline operation.

Ding et. al. (2004) add new contribution called as interval exchange moves to the study of Xu and Bailey. They consider the Airport Gate Assignment Problem in

which number of flights is higher than number of gates. Their model has two objectives, minimizing the number of flights assigned to the apron and minimizing the total walking distance or connection times.

Distances from check in to gates for originating passengers, from gates to baggage claim areas for destination passengers and from gate to gate for transfer passengers are considered. In addition to these they include the distance from the apron to the terminal. The numbers of passengers, who embark, disembark and transfer is also important to calculate the total passenger walking distance (Ding et. al., 2004).

For minimizing the number of flights assigned to the apron (first objective) a greedy algorithm is used. Basically, the greedy algorithm firstly sorts all of the flights according to their departure time and then assigns the flights to the gates one by one. Any flight can be assigned to an available gate with latest departure time. In the case of lack of available gate, the flight will be assigned to the apron. After implementing the algorithm, they prove that the greedy algorithm gives the optimal number of flights that can be scheduled in gates. The result of algorithm provides feasible initial solution for tabu search heuristics algorithm (Ding et. al., 2004).

As Xu and Bailey (2001), Ding et. al. uses tabu search heuristics algorithm, however Xu and Bailey choose the initial solution randomly and do not consider the situation which the number of flights exceed the number of gates. In the work of Xu and Bailey there are three neighborhood moves to search other feasible solutions. These moves are Insert Move, Exchange I Move and Exchange II Move.

Ding et. al. (2004) use Insert Move, assign a single flight to another gate, additionally they use The Interval Exchange Move and The Apron Exchange Move. The Interval Exchange Move exchanges two flight intervals in the current assignment. This move type can exchange many flights as well as one or two flights at different gates. Therefore The Interval Exchange Move is a generalized form of The Exchange I and II Moves of Xu and Bailey. On the other hand The Apron Exchange Move can exchange only one flight that is assigned to the apron with a flight that has been assigned to the gate.

TS memory has important role in the search process. It forbids the solution attribute changes recorded in the short-term memory to be reused. *Tabu tenure* parameter identifies the number of iterations a particular restriction remains in force. Finally they conducted experiments to compare Interval Exchange TS (ITS) of them with TS heuristics of Xu and Bailey (XTS). In their research there are 5 test data sets, which differ according to size, density and the condition that some flights may or cannot assigned to the gates. The results of experiments show ITS gives better performance when size and density increase. Moreover, ITS can find optimal solutions for small data sizes and good solutions for large data set in shorter running times (Ding et. al., 2004).

III.2.2 Genetic Algorithms

Some optimization problems are so complex that they have not totally been solved until today. GAs have a good potential for solving NP-hard problems such as GAP, as large-scale parallel stochastic search optimization algorithms (Hu and Paulo, 2007).

A genetic algorithm (GA) is a search technique used in computing to find exact or approximate solutions to optimization and search problems. Genetic algorithms are a particular class of evolutionary algorithms (EA) that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover.

One of the main operators, selection, prefers fitter individuals to be chosen for the next generation and for the application of the mutation and recombination operator (Stützle, 1998).

Mutation and crossover has importance for successful applications of GAs to the GAP. Mutation is used to increase diversity of chromosomes in GAs to exploit the solution space. On the other hand crossover can identify, inherit, and protect good common genes shared by chromosomes, and recombine non-common genes (Hu and Paulo, 2007).

A GA can be expected to produce good solutions, but it might never find a perfect solution. The process can be terminated after the prespecified number of generations or / and when an individual solution reaches a prespecified level of fitness (Gu and Chung, 1999).

Hu and Paulo (2007) propose a genetic algorithm for the multi objective airport gate assignment problem (MOGAP), where passenger walking distance, baggage transport distance and aircraft waiting time on the apron are considered simultaneously. Total passenger walking distance (TPWD) in airports considered for embarking and disembarking passengers as well as transfer passengers as common. Baggage transport distance is the total distance between aircraft and baggage claim areas. Aircraft waiting time on the apron is the difference between the planned entering time to gates and the allocating time to gates. System parameter ϕ is used to make the waiting time comparable to the distance. By the help of ϕ and weights to adjust the contribution of each objective, the weighted objective function is constructed.

The feasibility problem is hard to overcome, when designing efficient evolutionary operators. Hu and Paulo (2007) make an effort to designing a novel uniform crossover operator free of the feasibility problem. Relative positions of aircraft in the queues to gates are used to construct the chromosomes.

They conduct simulation tests to simulate under-congestion, congestion and over-congestion situations on airport. The GA of Hu and Chen (2005) is extended to solve MOGAP and compared with their GA. In the evolutionary process a good balance between diversity and convergence can be kept by the GA of Hu and Paulo (2007).

III.2.3 Ant Systems

The idea of imitating the behavior of ants for finding good solutions to combinatorial optimization problems is initiated by Dorigo, Maniezzo and Colormi. The principle of these methods is based on the way ants search for food and find their way back to the nest. Initially, ants explore the area surrounding their nest in a random manner. As soon as an ant finds a source of food it evaluates quantity and quality of the food and

carries some of this food to the nest. During the return trip, the ant leaves a chemical pheromone trail on the ground. The role of this pheromone trail is to guide other ants toward the source of food, and the quantity of pheromone left by an ant depends on the amount of food found. After a while, the path to the food source will be indicated by a strong pheromone trail and the more the ants which reach the source of food, the stronger the pheromone trail left (Gambardella, et. al., 1999).

Ant colony optimization is a metaheuristic in which a colony of artificial ants cooperates in finding good solutions to difficult discrete optimization problems. Cooperation is a key design component resource to a set of relatively simple agents (artificial ants) that communicate indirectly mediated by the environment. Good solutions are an emergent property of the agents' cooperative interaction (Dorigo, et. al., 2004).

Pintea et. al. (2008) propose an ant system bounded with a local search for the over constraint AGAP problem. They are interested in selecting and allocating aircrafts to the gates with an objective of minimizing the total passenger connection time. Their algorithm uses pheromone trail information to perform modifications on AGAP solutions.

In the Hybrid Ant System (HAS-AGAP), each ant is associated with a problem solution that is first modified using pheromone trail and later is improved using a local search mechanism. The algorithm is analyzed and compared with tabu search heuristic and Ant Colony System (ACS) metaheuristic. According the test results, HAS-AGAP performs better than TS ad ACS for AGAP and has nearly the same running time as ACS, but longer running time than TS (Pintea, et. al., 2008).

III.2.4 Simulated Annealing Methods

Simulated Annealing (SA) is a generic probabilistic heuristic approach originally proposed in Kirkpatrick et al. (1983) and Kirkpatrick (1984) for global optimization. Usually, SA locates a "good" approximation of the global optimum of a given objective function z in a large search space (Drexl and Nikulin, 2008).

SA starts from some initial solution $[\pi]$, and at each iteration probabilistically chooses either to accept a new solution $[\pi]'$ or keep $[\pi]$. The probabilities are chosen so that the problem ultimately tends to move to solutions with a better objective function value. Typically this process is repeated until a solution which is "good enough" has been determined, or until a given time limit has been reached. SA uses neighborhood concept, probabilistic acceptance of a new neighborhood solution, parameter (temperature) dependent acceptance probability, cooling schedule, termination criterion (Drexl and Nikulin, 2008).

The model of Drexl and Nikulin (2008) has three objectives, minimizing the number of ungated flights and the total passenger walking distances (or connection times) as well as maximizing the total gate assignment preferences. The authors formulate the gate assignment problem as integer programming with quadratic constraints. After finding an initial solution by greedy algorithm, Pareto Simulated Annealing with the neighborhood moves of Ding et. al. (2004) is adapted. The contribution of authors is adding the priorities and preferences of airports. The results are quite typical for PSA. They have observed both the convergence properties and the results provided for this single instance for a variety of other instances as well.

CHAPTER IV

GATE ASSIGNMENT PROBLEM: SIMULATION AND RULE BASED EXPERT SYSTEMS

IV SIMULATION AND RULE BASED EXPERT SYSTEMS

The complexity of maintaining assignments and the need to improve operational efficiency have resulted in alternative methods such as simulation and expert systems (Bolat, 2001).

IV.1 Simulation

Simulation is becoming an essential tool for planning, design, and management of airport facilities. A simulation of aircraft at gates at an airport can be applied for various periodically performed applications, relating to the dynamic behavior of aircraft at gates in airport terminals for analyses, evaluations, and decision supports. Conventionally, such simulations are implemented using an event-driven method (Cheng, 1998).

Krauter and Khan (1978), and Hamzawi (1986) have developed simulation models to be utilized as planning tools to determine the appropriate number and size of gates as well as the evaluation of various assignment strategies.

Yan et. al. (2002) propose a simulation framework for airport authorities to analyze the effects of stochastic flight delays on static gate assignments, and to evaluate flexible buffer times and real-time gate assignment rules.

At the first stage of simulation process, Yan et. al. (2002) use the optimization model and two heuristics, with the necessary information, to solve for static gate

assignments. Then they use the distribution of the flight delays to generate the arrival or departure times for each flight. On the situation of airplane cannot be assigned to its planned gate, and then they apply a rule to reassign the airplane to a gate for simulating real-time gate assignments. When all flights are assigned after a one-day simulation, they evaluate the effects of stochastic flight delays on the static gate assignments.

They also evaluate flexible buffer times and real-time gate assignment rules. The simulation can be performed each day, before a new season or specific time period, to obtain good rules or guides for gate assignments in that season or time period. To test the framework, Yan et. al. (2002) perform tests on actual Chiang Kai-Shek airport operations. The test results are good, show that the framework can be useful for airport authorities to perform gate assignments.

IV.2 Rule Based Expert Systems

Brazile and Swigger (1988), Gosling (1990), Srihari and Muthukrishnan (1991), and Brazile and Swigger (1991) have developed expert systems to cope with the uncertain information and to handle additional performance criteria.

Brazile and Swigger (1988) use flight information and knowledge about current constraints to produce possible gate assignment schedules. Their constraint-satisfaction expert system is named as GATES. To make decisions, GATES uses two types of production rule: permissive rules and conflict rules. Permissive rules determine when it's appropriate to consider a particular gate for a particular flight, and permit the system to search the next level of rules to obtain an assignment. Conflict rules determine when particular flights cannot be assigned to particular gates. System operators can modify schedules by retracting rules, adjusting tolerances, and deleting information. Their system was developed for a PC, thereby providing an efficient and flexible user environment.

According to Bolat (1999), the success in practice is too limited with these approaches. Those ignoring crucial factors have not progressed past the prototype stage. Some others failed during the implementation stage because the maintenance

factor was neglected. Usually, these systems are based on knowledge bases (set of heuristic rules) that are acquired from the experts in practice. (Simulation models also apply rules while making policy decisions.) Beside the difficulty in extracting the relevant knowledge, there are two conflicting goals with respect to these rules. For a successful implementation, the expert system has to be equipped with many hundreds or thousands of rules to consider most of the relevant factors in a problem. On the other hand, maintaining such a large and complex system becomes a critical factor (Bolat, 1999).

CHAPTER V

GATE ASSIGNMENT PROBLEM: IMPLEMENTATION

V.1 PURPOSE OF IMPLEMENTATION

The main goal of this study is to decide the appropriate method for initial solution that provides good quality solutions and to adapt the selected method and Tabu Search algorithm.

V.2 METHODS AND METHODOLOGY

Initially, comprehensive literature review theory and research relevant to the gate assignment problem is done. Afterwards, the description of the Airport Gate Assignment Problem and definition of the objective functions and constraints are studied. It is defined of the operational constraints via handling a real airport's gate assignment problem. Some approaches for initial solution are tried and are adapted. Moreover, tabu search algorithm used on these feasible solutions. The particular constraints are enforced to the determined appropriate solution methods. After all, the implementation is run by using of the indiscriminate and real data and results are reported.

V.3 PROBLEM DEFINITION: ISTANBUL ATATURK AIRPORT

The most important problem among the immobile resource optimization problems in Atatürk Airport, Gate Optimization Problem is formulated in this section. This problem is more than an optimization of limited resource problem.

With successfully implemented gate assignment problem, the Airport Company and Airline Companies have a chance to achieve these goals.

- a. maximize the revenue gain from gates by optimum gate assignment solution to optimize the gate utilization rate
- b. to improve the total utilization rate of airport facilities
- c. development of overall service quality with improvement of on time departure rates and reliability
- d. obtain the passenger flow stability by decreasing the walking distance of incoming and outgoing passengers
- e. decreasing the queue length on the way of runway and prevent congestions on the remote stand areas and runways
- f. lowering the fuel consumption which is both economic and environmental factor
- g. right and certain guesses of arrival and departure times of aircrafts let doing certain calculation of wants of demand set
- h. maximizing passenger satisfaction with decreased flight lateness and provided strong tie of connected flights,

In this part of the study the objective function is tried to select among the stated aims that has considerable effect on the others. Moreover, the past data of flights is taken and with some simplification and arrangements the data formed in the format of study. By adding this data to current knowledge, the most important decisions are taken for problem formulation.

V.3.1 Objective Function

Major aim of the gate assignment problem in Atatürk Airport is maximizing the revenue gain from the flights assigned directly to the gates. The income earned from operating of gates increases by each flight assigned to the gates - not apron. The airline companies have to pay TAV Airports Holding for their flights parked to gates. The aircraft assigned to the apron/remote parking area is a loss for TAV.

The passengers' perception of service quality has strong relationship with getting on or getting off an airplane throughout the gate or bridge. They do not want to loss time by getting bus or walking long distances. For a given terminal building configuration, the average passenger walking distance may vary depending on the particular assignment of aircrafts to aircraft stands. Shortly, maximizing the flights assigned to gates brings higher passenger satisfaction.

At the assignment process it is important to consider ground time periods, in other words the difference between departure and arrival time of an aircraft. In the beginning of the study, it is expected to get lower objective function score by assigning as much as flights to the bridges. Although the number of flights assigned to apron makes the total cost higher, the essential factor affects objective function is total ground time periods of these flights. Therefore the objective function is formed as minimizing total ground time of flights assigned to apron, not as minimizing the number of flights assigned to apron. The solution with more flight number at apron but lower total ground time at apron is in preference to the solution with lower flight number but higher ground time at apron.

In addition, airport terminal building designs sometimes result in significant walking distances for passengers. This makes the distance between check-in desks and gates for departing passengers and the distance between gates and the baggage claim area for arriving passengers the objective which should be minimized. Minimizing total walking distance also has positive effect on the total passenger satisfaction. By minimizing the connection distance, the AGAP may achieve another associated benefit for the company.

Recall Problem Formulation section, it is assumed that there are n flights, and m gates, $m + 1$ represents the apron at which flights arrive when no gates are available. d_i and a_i represent departure time and arrival time of flight i . $t_{i,m+1}$ represents time interval on the ground of flight i , assigned to the apron; in other words ground time period of aircraft. Ground time periods of flights assigned to the apron can be calculated as;

$$t_{i,m+1} = d_i - a_i .$$

minimize

$$\sum_{i=1}^n t_{i,m+1} \quad (V.1)$$

$$\sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^{m+1} \sum_{l=1}^{m+1} f_{ij} w_{kl} y_{ik} y_{jl} + \sum_{i=1}^n f_{0i} w_{0i} + \sum_{i=1}^n f_{i0} w_{i0} \quad (V.2)$$

In brief, the first goal is to minimize total ground time period of flights assigned to the apron (V.1). The other aim of the problem is to minimize total passenger walking distance (V.2).

V.3.2 Data

Istanbul Atatürk Airport has a dense international passenger population with 18 million passengers in a year. About 60 percent of total visitors are international passengers. At the Atatürk Airport, which has intensive flight traffic, there are gates and remote parking areas for servicing the aircrafts. 32 of 90 parking areas are gates, 9 gates at domestic, 23 gates at international terminal and the rests are called as apron or remote parking area (TAV, 2009).

Daily flight schedule data belong to different time periods, like the days fall on national and religious holiday, end of school term, and also ordinary days of all seasons was procured by face to face interviews with competent person at TAV. This flight schedule constitutes the main input for gate assignment problem. The paired flights (two flight data served by the same aircraft) information, the arrival and departure time of flights, the class of flights according to origin and destination point (domestic or international) and the type of aircrafts come up among the daily data.

The complementary input is the information about infrastructure of gates, including gate size, the set of flight types that compatible with gate, distances from check-in desks to gate and gate to baggage claim area.

The gate assignment problem is actually a complex problem should be integrated with several management systems like energy, building, airport and with systems like

flight information display, terminals for common users, additionally with resource optimization problems as baggage chute, carousel and check-in desks resource scheduling. Therefore the problem is simplified as gate assignment of international flights in Atatürk Airport in order to overcome the problem. International flights (Domestic to International-DI) and (International to International-II), and International Gates and Stands are taken into account. The pair-wise international flights (DI and II) and international gates (G201 to G223) data is regulated for adaptation to java code and making easy to read and understand.

The original flight schedule data and gates with assigned flights chart can be seen from Table V.1 and Figure V.1.

Table V.1 The Sample of Flight Schedule Time Table

04.01.2010										
Arrival Flight No	Date	STAD	Cat	Departure Flight No	Date	STAD	Cat_1	Airline IATA	Aircraft Type	Registration No
TK 675	01.04.2010	22:25	Dom	TK 732	01.05.2010	05:20	Dom	TK	B734	TCJDG
AZ 702	01.04.2010	23:10	Int	AZ 703	01.05.2010	04:15	Int	AZ	A321	IBIXQ
TK 253	01.04.2010	06:20	Dom	TK 1473	01.04.2010	07:10	Int	TK	B738	TCJHE
TK 327	01.04.2010	16:05	Dom	TK 640	01.04.2010	17:45	Dom	TK	A320	TCJPT
TK 8552	01.04.2010	12:55	Int	LH 3353	01.04.2010	13:55	Int	TK	A321	DAIRF
UA 9298	01.04.2010	12:55	Int	TK 8553	01.04.2010	13:55	Int	UA	A321	DAIRF
TK 1876	01.04.2010	19:30	Int	TK 476	01.04.2010	21:35	Dom	TK	B738	TCJGB
AF 2390	01.03.2010	21:15	Int	AF 2391	01.04.2010	07:00	Int	AF	A320	FGFKS
TK 151	01.04.2010	20:05	Dom	TK 696	01.04.2010	21:30	Dom	TK	B738	TCJFK
D9 411	01.04.2010	07:30	Int	D9 412	01.04.2010	08:30	Int	D9	B734	VQBAO
TK 1976	01.04.2010	15:35	Int	TK 1162	01.04.2010	18:10	Int	TK	A321	TCJRJ
TK 1956	01.04.2010	01:50	Int	TK 1937	01.04.2010	05:55	Int	TK	A321	TCJRD
TK 1426	01.04.2010	14:45	Int	TK 238	01.04.2010	16:00	Dom	TK	B738	TCJHC
AA 6465	01.03.2010	20:10	Int	BA 675	01.04.2010	07:05	Int	AA	A320	GEUUI
TK 1265	01.04.2010	04:35	Int	TK 1967	01.04.2010	06:20	Int	TK	B738	TCJFK
8Q 208	01.04.2010	02:15	Int	8Q 032	01.04.2010	04:45	Dom	8Q	A320	TCOBD
TK 341	01.03.2010	21:35	Dom	TK 546	01.04.2010	05:00	Dom	TK	A320	TCJPI
TK 1187	01.04.2010	15:50	Int	TK 1348	01.04.2010	18:15	Int	TK	B738	TCJGU
TK 1263	01.03.2010	20:30	Int	TK 1184	01.04.2010	05:40	Int	TK	A321	TCJMI
TK 237	01.04.2010	08:05	Dom	TK 1505	01.04.2010	09:00	Int	TK	B738	TCJFJ
KC 911	01.04.2010	08:10	Int	KC 912	01.04.2010	09:20	Int	KC	A320	P4PAS
TK 1722	01.04.2010	13:25	Int	Tk 1445	01.04.2010	15:05	Int	TK	A321	TCJRG
TK 655	01.04.2010	11:40	Dom	TK 468	01.04.2010	15:00	Dom	TK	A320	TCJPH
JP 648	01.04.2010	01:00	Int	JP 649	01.04.2010	03:30	Int	JP	CRJ	S5AAL
SV 217	01.04.2010	10:45	Int	SV 218	01.04.2010	12:00	Int	SV	B757	TCOGS
TK 459	01.04.2010	10:30	Dom	LH 2099	01.04.2010	12:15	Int	TK	B738	TCJHB

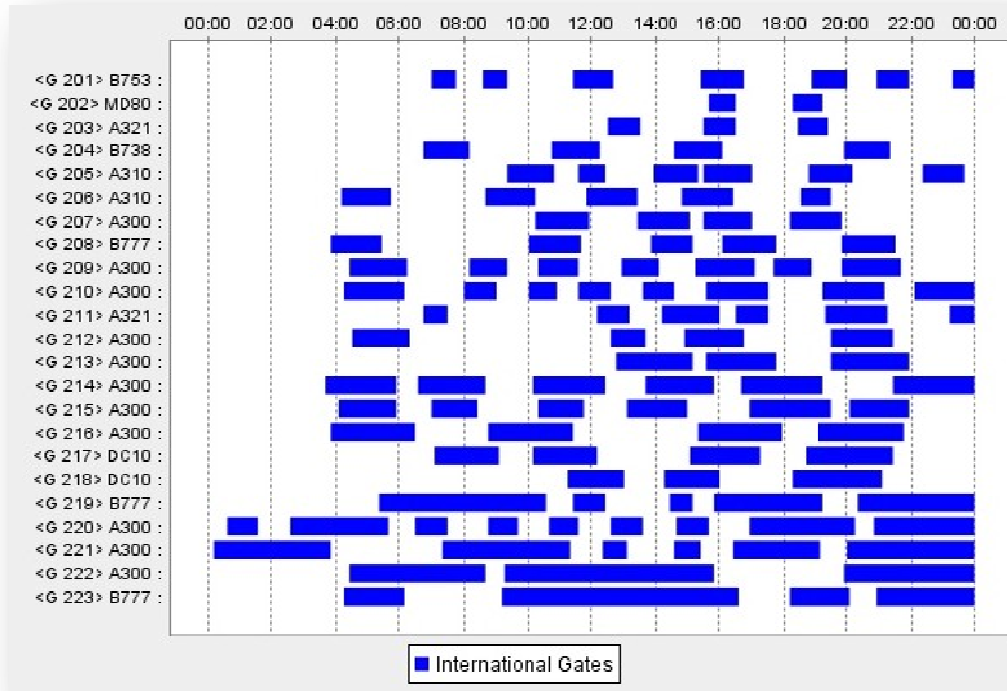


Figure V.1 Gate Assignment Solution Chart of TAV at 10.01.2009

For achieving successful assignment of flights to gates, gate-aircraft size compatibility should be considered. Therefore, during the assignment process the group of the aircraft, the aircraft types compatible to a gate classed as a one group, is considered important. Again for obtaining convenience in practice, the group of flights is named as A to H Class. From the tenth column of flight schedule (Table V.1) (Gökten, 2010), it can be seen the aircraft types are named like B738, A320 and MD83. It is preferred to group aircrafts as in the Table V.2 on the basis of real data of international gates and international aircrafts.

Table V.2 Flight Classes According to Aircraft-Gate Size Compatibility

A	B	C	D	E	F	G	H
F70	A319	B738	B733	A310	A300	DC10	B747
F100	A320		B753	A343	A330	DC8	B773
B734	A321		B757	B763			
B735							
B736							
B737							
MD80							
MD82							
MD83							
MD88							

Additionally, gate distances to the entry and exit points and total number of passengers in a flight is needed to calculate total walking distance of passengers. For this purpose by referencing the layout of Atatürk Airport, it has been seen the gates of international terminal are located linearly (Figure V.2).

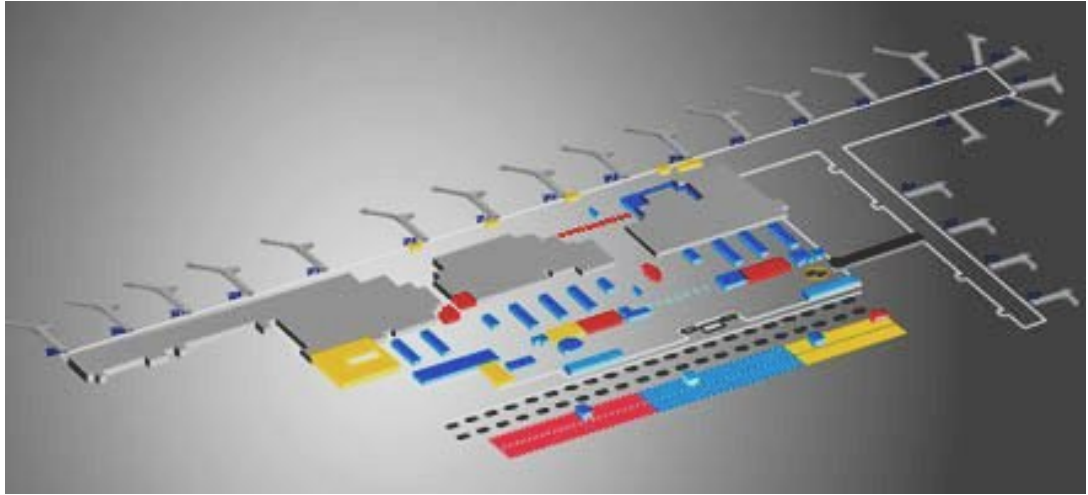


Figure V.2 The Layout of International Terminal in Atatürk Airport (DHMI, 2009)

The distance from passport control to Gate 201, Gate 201 to baggage claim area and the distance between two adjacent gates are set to 2 units (Berber, 2009). For penalizing the flight assignment to apron or remote stand area, the distance between passport control and apron is set to 100.

During the determination process of time tables of flight schedule, buffer time for unboarding passengers and crew members from the incoming flight, for cleaning the cabin and supplementing the supplies, and for boarding the crew members and passengers to the departure flight have been considered (Xu and Bailey, 2001). However, before assigning process the buffer time is needed too, with the aim of decreasing the negative effects of short delays. The buffer time between the departure time of flight and arrival time of successive flight at the same gate is decided as fifteen (15) minutes and added to the flight departure time for convenience (Berber, 2009).

V.3.3 Constraints

As mentioned in the Problem Formulation section, there are 3 major constraints of AGAP.

1. Every flight must be assigned to one and only one feasible gate or assigned to remote stand area.
2. No two aircrafts may be assigned to the same gate at the same time.
3. Gate is available if and only if the before flight has departed for a buffer time.

Additionally the AGAP is considered with other hard constraints like the compatibility of Gates and flights' gates by defining the size of gates and type of flights. A large gate has the flexibility to accommodate various sizes of aircraft whereas a small gate is more limited (Gökten, 2010). In this study gate-aircraft size compatibility constraint is taken into account during both initial and final solution seeking processes.

Moreover, there are side constraints like assigning the flights of specific airlines to the predetermined gate set. As an example, one of the international airline companies has a specific gate which is close to passport control point and the flights of this company is certainly assigned to the same gate. Similarly, some other companies' flights are assigned to the separated gates for the security reasons. Because these secondary constraints make the problem much more complex, they are not considered.

Furthermore, like for each international gate, for each remote stand area there is predetermined compatible flight set. The flight which should be assigned to remote stand areas preferentially sent to the closest apron. However, in this study it is assumed that there is a single apron which can serve to the flights even they have overlapped time periods.

CHAPTER VI

GATE ASSIGNMENT PROBLEM: INITIAL SOLUTION

VI.1 EVOLUTION OF COST

There are various costs associated with gate assignments which need to be optimized simultaneously. We consider the costs both published in OR journals as objective functions and faced by the airport managers in practice.

The main goal of TAV as mentioned before is directing as much flights as possible towards gates instead of remote stand areas. According to the aircraft type and its parking time at gate, the parking cost of flights can be determined. This cost is among the incomes of TAV. Therefore, firstly the objective function formed as maximizing the number of flights assigned to gates; in other words minimizing the number of flights assigned to aprons.

During the search, it is realized that assigning more flights to gates increases the gate usage; however there are solutions which have better gate capacity utilization rates. The reason of this situation can be demonstrated by the assignment solution of sample data. Assume that the flight i has long ground time and the consecutive flights j , k and l are interchangeable with flight i and have short ground time. By applying greedy algorithm, the flight with earlier departure time is assigned first approach, the result as on below figure (Figure VI.1) is reached. In this result, there are as much as scheduled flights, three flights, at gate. However, assigning flight i to apron charges to airport for the flight's huge size, great population amount and heavy baggage volume and most importantly for small usage rate of gate. Allocating the flights in different way higher gate utilization rate can be obtained. If flight j , k , l is assigned to apron and flight i to gate, this rate will increase from 0,67% to 0,87%.

As a result it is decided to change first objective as minimizing the apron usage rate in other words, total aircraft ground time period at Apron (TAGTA).

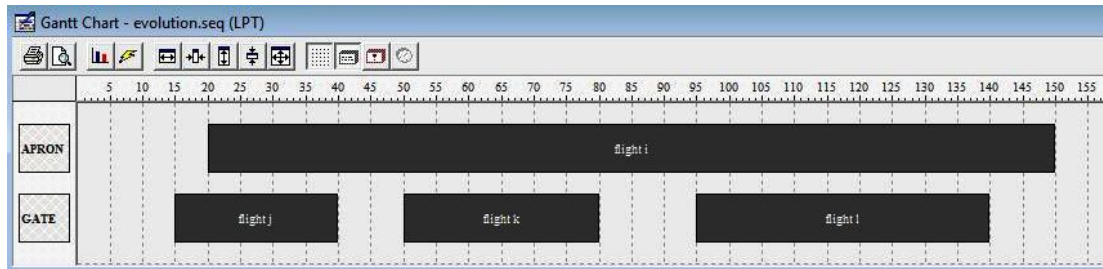


Figure VI.1 Assignment of Small Sample Data

Moreover, the most frequently used objective is minimization of the walking distance of all the passengers inside the airport. Most papers include the walking distance of passengers as a component of the objective function; e.g., the pure distance-based objective in Haghani and Chen (1998); total connection times that passengers walk in Xu and Bailey (2001); the passenger walking distance and passenger waiting time in Yan and Huo (2001) and Yan et al. (2002); the number of flights assigned to gates and passenger walking distance in Ding et. al. (2004).

Although, it is seen from the published literature about the gate assignment problem that total walking distance has a considerable proportion on objective function, in practice it is important only on airports with more terminals and for the transfer passengers. One of the few airports which take consider total walking distance is British Airways.

According to most airport managers, the more essential thing is sitting without any physical activity for considerable periods of time. To manage this, the managers focus on the encouragement of passengers to walk around when they have the time. In view of this, it is inappropriate to place a great emphasis on minimizing the total walking distance of all airline passengers inside airports (Yu, et. al., 2009).

Because the gate optimization problem is a multi-objective problem, it can be solved by combining the various objectives into a single function. As a result, a weighted single objective function that covers the two objectives mentioned before is formed. The weights of objectives are set by regarding to emphasize total walking distance

whether few or much. The experiments are carried out for balancing the two objectives in the way that little change in the first objective gate utilization rate should be more important than large change on the second objective, total walking distance. In other words, the most important criterion is the improvement in cost. When a flight, has been assigned to apron, is assigned to gate the improvement in cost should be higher than the improvement in cost of total walking distance in an alternate assignment.

$$J_{TAGTA} = \sum_{i=1}^n t_{i,m+1} \quad (VI.1)$$

$$J_{TPWD} = \sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^{m+1} \sum_{l=1}^{m+1} f_{ij} w_{kl} y_{ik} y_{jl} + \sum_{i=1}^n f_{0i} w_{0i} + \sum_{i=1}^n f_{i0} w_{i0} \quad (VI.2)$$

Two objectives of the problem are minimizing the total aircraft ground time period at apron (VI.1) and total passenger walking distance (VI.2). The following weighted objective function is used to cover the two objectives (VI.3).

$$J_{MOGAP} = \alpha J_{TGTA} + \beta J_{TPWD} \quad (VI.3)$$

The ground times of aircrafts are measured in minutes and the walking distances of passengers measured in meters. Because, the fundamental aim of the managers in Atatürk Airport in this assignment process is minimizing the cost, the measures can be converted to the cost according to the potencies of objectives. α is a parameter for finding the equivalent cost value of the total ground time and β is a parameter for finding the equivalent cost value of the total walking distance. α is set to 100 and β is set to 0,1.

$$J_{MOGAP} = 100 * J_{TGTA} + 0,1 * J_{TPWD} \quad (VI.4)$$

Final weighted objective function is as on the above function (VI.4).

VI.2 METHODS FOR INITIAL SOLUTION

Assigning international flights to the international gates effectively is an important and difficult task. The algorithms are designed to find initial solution and meta-heuristic is used to achieve this task successfully. For efficiency of tabu search algorithm, it is important to give a good quality initial solution as a starting point. It has proven that the greedy algorithm, adopted in AGAP finds optimal number of flights assigned to gate when the number of flights exceeds the number of gates by Ding et. al. (2004). Although in this study objective function and constraints are different than them, their greedy algorithm is implemented to the real and random data.

The sequence order of flights has an important role on the initial solution quality. A sequence order of flights with respect to time specifies in which order they are assigned to gates. If the sequence order of flights is changed and the assignment is done according to this order, a different initial solution will be reached. There may several methods for finding initial feasible solution except greedy algorithm. Here two of them which are considerably good, will be discussed.

VI.2.1 Greedy Algorithm

A greedy algorithm always makes a locally optimal choice in the hope that this choice will lead to a globally optimal solution. It does not always yield optimal solution, but for many problems they do (Corman, et. al., 2001).

By greedy algorithms the optimal solutions of the problems widening from change making for normal coin denominations, minimum spanning tree, single source shortest paths, and simple scheduling problems to Huffman codes can be find. Moreover, the fractional knapsack problem is solvable by a greedy strategy, whereas the 0-1 problem is not (Corman, et. al., 2001).

A greedy algorithm obtains an optimal solution to a problem by making a sequence of choices. For each decision point in the algorithm, the choice that seems best at the moment is chosen.

Generally, greedy algorithms are designed according to the following sequence of steps:

1. Cast the optimization problem as one in which we make a choice and are left with one sub problem to solve.
2. Prove that there is always an optimal solution to the original problem that makes the greedy choice, so that the greedy choice is always safe.
3. Demonstrate that, having made the greedy choice, what remains is a sub problem with the property that if we combine an optimal solution to the original problem (Corman, et. al., 2001).

To construct the solution in an optimal way, algorithm maintains two sets. One contains chosen items and the other contains rejected items.

Greedy Algorithm Structure

- **Initially the set of chosen items is empty i.e., solution set.**
- **At each step item will be added in a solution set by using selection function.**
- **IF the set would no longer be feasible**
 - **reject items under consideration (and never consider again).**
- **ELSE IF set is still feasible THEN**
 - **add the current item**

Figure VI.2 Greedy Algorithm Structure (Saudi, 2008)

The greedy algorithm consists of four functions.

1. A function that checks whether chosen set of items provide a solution.
2. A function that checks the feasibility of a set.
3. The selection function tells which of the candidates is the most promising.
4. An objective function, which does not appear explicitly, gives the value of a solution (Saudi, A., 2008).

VI.2.1.1 The Elements of Greedy Algorithm

The algorithm which simply seeks to add the element with highest possible weight available at the time of selection that does not violate the structure of an optimal solution in an obvious way has two elements (Mock, 2002).

1. Greedy choice property
2. Optimal substructure (ideally)

Greedy choice property: Globally optimal solution can be arrived by making a locally optimal solution. The greedy choice property is preferred since then the greedy algorithm will lead to the optimal, but this is not always the case – the greedy algorithm may lead to a suboptimal solution. Similar to dynamic programming, but does not solve sub problems (Mock, 2002).

Optimal substructure: Optimal solution to the problem contains within it optimal solutions to sub problems. This implies sub problems can be solved and solutions are built up to solve larger problems (Mock, 2002).

VI.2.1.2 Activity Selection Problem

An activity selection is the problem of scheduling an exclusive resource among several competing activity. Furthermore, by the greedy principle it is possible to get optimal schedule of the problem. Scheduling the use of a room (only one entity can use it at a time) when several groups want to use it and renting out some piece of equipment to different people is some of the versions of this problem (Mock, 2002).

Definition: Set $S = \{1, 2, \dots, n\}$ of activities. Each activity has a start time s_i and a finish time f_i , where $s_i < f_i$.

Activities i and j are compatible if the half-open interval $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap, that is, i and j are compatible if $s_i \geq f_j$ and $s_j \geq f_i$.

A simple greedy algorithm solves this problem optimally:

- Sort input activities in order by increasing finishing time
- $n \leftarrow \text{length}[S]$
- $A \leftarrow 1$
- $j \leftarrow 1$
- for $i \leftarrow 2$ to n
 - if $s_i \geq f_j$ then
 - $A \leftarrow A \cup \{ i \}$
 - $j \leftarrow i$
- return A

Example:

Table VI.1 List of the Activities Ranked According to Increased Order Finishing Time

I	Start	Finish	1	2	3	4	5	6	7	8	9	10	11	12
1	1	4	█	█	█	█								
2	3	5		█	█	█	█	█	█	█	█	█	█	█
3	0	6	█	█	█	█	█	█						
4	5	7			█	█	█	█	█	█	█	█	█	█
5	3	8			█	█	█	█	█	█	█	█		
6	5	9				█	█	█	█	█	█	█	█	█
7	6	10					█	█	█	█	█	█	█	█
8	8	11							█	█	█	█	█	█
9	8	12									█	█	█	█

Just marches through each activity in terms of the finishing time, and schedules it if possible (Mock, 2002).

Schedule job 1:

```
1111
  222
333333
```

Job two and three do not fit, don't add them. Try job 4:

```
1111
  444
```

Fits, so leave it in. Try job 5, 6, 7:

```
1111444
  555555
    66666
      77777
```

None of these fit, try job 8:

```
1111444
      8888
      99999
```

Job 8 fits, job 9 does not fit.

```
11114448888
```

This is the final, optimal schedule that maximizes the number of people that want to use of the room. The runtime is simple $O(n \lg n)$ to sort, and then $O(n)$ to run through the finishing times, making this algorithm $O(n \lg n)$ overall (Mock, 2002).

The activity selection problem is to select a maximum-size set of mutually compatible activities (Mock, 2002).

VI.2.1.3 Gate Assignment Problem

The model of Ding et. al. (2004) attempt to assign flights to gates to minimize the number of flights that are assigned to apron and to minimize total walking distances between gates. They firstly consider the minimization of ungated flights and adopt an activity selection principle, which is used to assign limited infrastructure resources to a number of prescheduled competing activities.

The basic idea of the greedy algorithm is as follows. After sorting all the flights in order by a criteria (e.g. increasing departure time), flights are assigned to the gates one by one. Any flight will be assigned to an available gate with latest departure time. If there are no gates available, the flight will be assigned to the apron.

The basic details of the algorithm are same as activity selection problem. The greedy solution not only gives us the optimal number of flights that can be scheduled in gates, but also helps us to get a feasible initial solution.

Greedy algorithm is applied to a daily schedule of a major Turkish airport corporation at one of its major international terminal.

For finding good quality feasible initial solutions three different greedy algorithms are attempted to improve, and are cited below individually.

VI.2.1 Sorting According to Departure Time

The details of the algorithm and java code are as follows.

- 1- Put the flights data in decreasing order according to their departure times

A class named **Flight** is created to hide the type (char t_i), arrival time (int a_i) and departure time (int d_i) of flights. It is needed that **Flight** class implements comparable class and it has *compareTo* method (Figure VI.2). *compareTo(Flight f)* sorts the flights according to their departure times (d_i). If the aircrafts have equal departure time, the flights with earlier arrival time and then with broad gate size will ranked firstly.

When *compareTo* method faced with flights which have same type as well as same arrival and departure time, it should distinguish them. With the help of *hashCode* method, *compareTo* method can perceive them as equal but not same flights.

```
1-     public int compareTo(Flight f) {
2-         if (this.di < f.di)
3-             return -1;
4-         if (this.di > f.di)
5-             return 1;
6-         if (this.ai < f.ai)
7-             return -1;
8-         if (this.ai > f.ai)
9-             return 1;
10-        if (this.ti > f.ti)
11-            return -1;
12-        if (this.ti < f.ti)
13-            return 1;
14-
15-        else
16-            return (hashCode() - f.hashCode()); }
```

Figure VI.3 *compareTo* Method for Sorting Flights

2- Sort the gates according to their earliest available time.

A class named **Gate**, implements Comparable class to put in order the gates according to their earliest available time is created. The earliest available time of gate k (g_k) is simply equals the departure time of last flight assigned to this gate. It equals -1 for all gates at the beginning of the assignment process. After sorting the **Gate** objects, the gate with a maximum earliest available time will be at the first order and the gate with the minimum earliest available time will be at last.

3- For each flight i

- Find gate k such that $g_k < a_i$.
- If such k exists, assign flight i to gate k , update $g_k = d_i$.
- If k does not exist, assign flight i to the apron.

4- Output the result.

The greedy algorithm gives the optimal number of flights that can be scheduled in gates. As a result the number of flight that assigned to apron is minimized.

Some tests are conducted to this algorithm with the small and density input random data as well as the real flight data set of free and ordinary days. Firstly for convenience, Flight and Gate size compatibility was not taken into consideration.

As the application results of greedy algorithm on other problems, the activity number in other words number of flights assigned to gates is maximized. However, because our first and considerable objective function, shaped during the problem configuration of the problem, is maximizing the utilization rate of gates; it is needed to make new analysis on the results.

According to the results of the manual tests with small input size data, it is confirmed that there are better quality solutions. Therefore, new models are tried to improve to provide better feasible solutions, or optimal solutions.

Firstly, *first come first served* principle is implemented with a greedy algorithm to allocate flights to gates. Then a comparison between the performance of that algorithm and the performances of others is made.

VI.2.2 First Come First Served Model

Assigning the flights according to their arrival times to gate provide an essential mechanism for achieving scheduling efficiency, scheduling fairness and controller preference. The sequence order that often meets the requirements of all three of these objectives simultaneously is the First-Come-First-Served (FCFS) order (Erzberger, 1995).

The FCFS sequence is established by time-ordering arrivals of aircrafts according to increasing planned entering time to gates. Beginning with the first aircraft in the sequence, each aircraft is assigned to the gate with the minimum earliest available time, while ensuring that the specified buffer time between two successive flights at the gate is met.

The first flight at the list is assigned to the first convenient gate in the situation that the hard and soft constraints are not considered. At the result of the tests with small input, the good gate utilization rate achievement was observed. In this algorithm which proceeds like SADT algorithm, the sequence of flights were changed, and the assignment accomplished in this manner.

One of the small data set consists of the arrival and departure information of 55 flights and 7 gates. The output shows; the number of flights which is assigned to the apron is generally more than the output of SADT. However, total time duration of these flights at apron is less than total apron usage period of SADT solution as presented in the below table. Table VI.2 represents the comparison of resulting assignment when the aircraft are scheduled according to an FCFS sequence and SADT algorithm.

The details of results are presented in the last section of this chapter.

The table shows the gate utilization rate as well as gate and apron usage both in time and in quantity of two principles. The FCFS sequence, which is input to the scheduler, orders the aircrafts according to increasing planned entering time to gates. When aircraft are sequenced and scheduled to be first-come-first-served at gates, gate utilization (85.98%) is observed higher than the rate of SADT algorithm (80.57%), although the number of flights at apron is higher (9) than the number of SADT (8).

Table VI.2 The Comparison of SADT and FCFS Principle Solutions

Principle	Flights at Gate (G)	Flights at Apron (A)	Gate Utilization (G/G+A)
	<i>Total time on Terminal</i>	<i>Total time on Apron</i>	
SADT	4685	1130	%80.57
FCFS	5000	815	%85.98
	<i># of Flights</i>	<i># of Flights</i>	
SADT	47	8	
FCFS	46	9	

However, according to Hu and Paolo (2007), because the FCFS principle does not take into account the layout of airport terminals, the result is usually not optimal or even not near-optimal.

VI.2.3 Longest Processing Time Algorithm

The longest processing time rule sorts the jobs in the order of decreasing processing times. Whenever a machine is freed, the largest job ready at the time will begin processing. It schedules the longest jobs first so that no one large job will "stick out" at the end of the schedule and dramatically lengthen the completion time of the last job (Hochbaum, 1999).

In our problem the jobs refer to the flights and processing time refers to the time difference between two consecutive flights served by the same aircraft, the departure time of first and the arrival time of latter flight. As it is mentioned before between the arrivals time of an aircraft and its departure time lots of work should be done.

Transferring of passengers and circulating the baggage from aircraft to exit, refueling and cleaning the aircraft, checking and boarding new passengers to the aircraft are the major part of these works. All of these actions determine the processing time, the time period that aircraft should stand at the gate or apron of the aircraft.

Firstly the processing time of aircrafts on the ground (terminal) is calculated. Then the priority list is created by listing the aircrafts in decreasing order of processing times (longest flight first, shortest flight last). Flights with equal waiting times on the ground were listed in any order.

One of the two reasons to assign the flights which have longer processing time on the terminal first is these flights' worth in the sight of Airport Company. The other reason is the assignment of so long flights to the remote stand area may drop gate utilization rate.

This algorithm, with which getting high usage rate of gates is expected, is tested with small sample data by again not considering the soft and hard constraints. The results of trials are relatively good. Much more effective solutions are expected with the moves of tabu search algorithms by starting from this initial solution.

VI.2.4 Results

There are the results of tests applied on the 15 different data sets at the below table.

On the random data that gate number ranges between 5 and 52, it is assumed that the flight number is exactly equals two times aircraft numbers. In other words it is assumed that the flight data is pair wise; each arrival flight has a counter departure flight. Additionally when composing the data, flight numbers are determined for normal dense data and nearly at the same rate.

The problem without consideration of size compatibility is solved by SADT, FCFS and LPT algorithms and the table constituted to compare gate utilization rates of algorithms (Table VI.3).

Table VI.3 Gate Utilization Rates of Three Greedy Algorithms

Problem Number	Number of Gates	Number of Aircrafts	Gate Utilization Rate		
			SADT	FCFS	LPT
1	5	40	81.98	85.02	86.03
2	5	30	91.93	93.48	88.97
3	10	80	83.49	85.30	86.10
4	10	70	91.63	93.32	93.32
5	10	90	80.69	82.27	83.72
6	13	120	84.00	86.50	85.17
7	15	140	82.60	85.74	83.79
8	17	150	84.19	87.35	87.04
9	20	190	88.59	91.23	89.36
10	17	130	87.36	90.07	92.00
11	23	200	87.24	88.01	85.71
12	28	250	90.35	91.20	89.24
13	37	350	89.32	91.34	87.79
14	43	450	82.39	85.85	82.11
15	52	500	90.98	92.57	89.78

As seen from the table there is no algorithm which gives the best solution for all the data sets. However, with FCFS principle best gate utilization rates can be found on most of the data sets.

By adding the hard constraints to the problem and implementing tabu search metaheuristic, better quality results are expected.

CHAPTER VII

TABU SEARCH META HEURISTICS

VII.1 NEIGHBORHOOD SEARCH METHODS

Heuristics, approximate solution techniques, have been used since the beginnings of operations research to tackle difficult combinatorial problems. While many different approaches were proposed and experimented with, the most popular one was based on Local Search (LS) improvement techniques. LS can be roughly summarized as an iterative search procedure that, starting from an initial feasible solution, progressively improves it by applying a series of local modifications (or moves). The search terminates when it encounters a local optimum with respect to the transformations that it considers, an important limitation of the method: this local optimum is often a fairly mediocre solution (Gendreau, 2003).

The basic principle of TS is to pursue LS whenever it encounters a local optimum by allowing non-improving moves; *cycling* back to previously visited solutions is prevented by the use of *memories*, called *tabu lists*, that record the recent history of the search, a key idea that can be linked to Artificial Intelligence concepts (Gendreau, 2003).

The search space of a TS heuristic is the space of all possible solutions that can be considered (visited) during the search. The search space could simply be the set of feasible solutions to the problem, where each point in the search space corresponds to a set of gate assignments satisfying all the specified constraints (Burke and Kendall, 2005).

A neighborhood of a solution S is a set of solutions that are in some sense close to S . They can be easily computed from S or they share a significant amount of structure with S (Barták and Michela, 2005).

Local Search for the Gate Assignment Problem starts at some initial, and iteratively moves to neighboring solutions, trying to reduce the total cost.

Solutions that are reachable from solution s by a single move form a neighborhood of s , denoted by $N(s)$. The attractiveness of a move from s to s' can be examined by calculating the cost (objective function value) improvement of s' compared to that of s (Pintea et. al., 2008).

Three types of neighborhood moves that prevail in the AGAP applications are developed firstly by Xu and Baily. These moves are Insert Move that moves a flight to another gate, Exchange I Move which exchanges two flights and their gates and Exchange II Move exchanges two flight pairs and their gates. Ding et. al. used neighborhood search moves too. However, when they are using the first move just as Xu and Baily's, they combined the two exchange moves and called as Interval Exchange Move. Additionally they take notice of the fact that total gate capacities in an airport generally are not adequate and some flights should be assigned to aprons. Therefore they add Apron Exchange Move as a last neighborhood search move.

Although different methods are tried for initial solution (SADT, FCFS and LPT) and different move types are applied by us; the solution with minimum cost is not gotten. Because these solutions are not optimal, the new move, Insert and Remove Move, instead of simple Insert Method, is improved by changing its function.

VII.1.1 Interval Exchange Move

The first type of neighborhood move is called as Interval Exchange Move. Instead of single flights exchange or consecutive flight pairs moves as original, exchange move preferred by Ding et al. because the original two moves can not provide feasible solutions. Additionally, there is no restriction to one-one or two-two exchange with new one. They note that the three flights in a gate can be exchanged with the two

flights in another gate. They exchange the flights whose arrival and departure time are between flight a and b in gate k with the flights whose arrival and departure time are between flight c and d in gate l. This is expressed by $(a, b, k) \leftrightarrow (c, d, l)$ and illustrated in Figure VII.1. As flights between a, b and c, d are represented by two intervals on the axis, this method is called Interval Exchange Move.

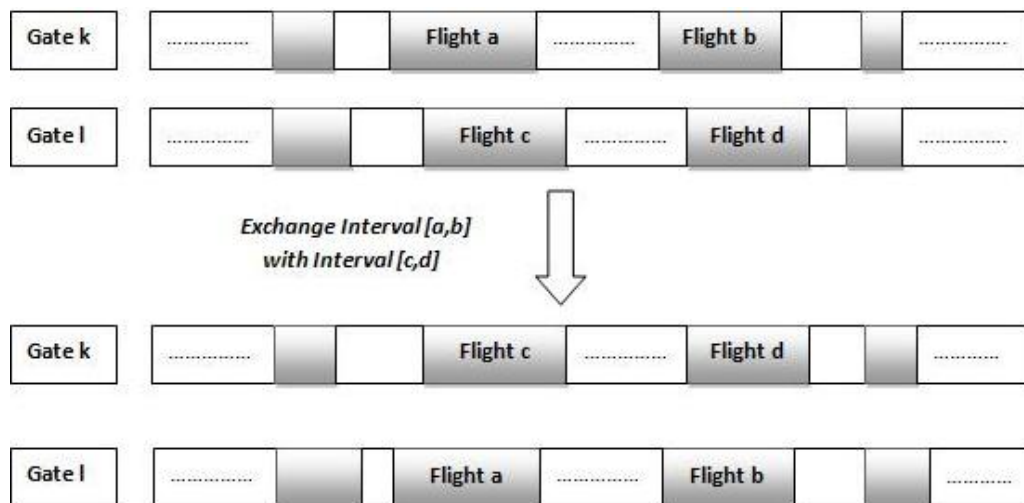


Figure VII.1 Illustration of Interval Exchange Move

To perform the Interval Exchange Move, firstly two compatible intervals should be found. In order to get feasible solutions, “Interval” data should contain four time points: the earliest available time (t_1), the start time (t_2), the end time (t_3) and the latest available time (t_4). Figure VII.2 illustrates the meaning of these four time points.



Figure VII.2 Four Time Points of an Interval

Further to this, Ding et. al., 2004 define two functions on intervals. *ExtendLeft()* extends the current interval by adding the flight which is just left to it, and *ExtendRight()* extends the current interval by adding the flight which is just right to it. The functions return Boolean values to indicate whether the operations are successful. For example, the *ExtendLeft()* operation will fail if the current interval

has included the first flight. Additionally, $Previous(i)$ returns the flight just arranged before flight i in the same gate, $Next(i)$ returns the flight just arranged after flight i . With these, it can be stated an algorithm to find compatible intervals for Interval Exchange Move with an example.

The *Interval Exchange Move Algorithm* is illustrated with the sample data. The data contains 5 gates and 24 flights. The flight information of the sample data is randomly generated. The time period that aircrafts should stand at the terminal is assumed between 20 and 50 minutes.

The SADT algorithm result, at the same time the feasible initial solution is as follows. The single aircraft is shown by arrival time and departure time in parenthesis. For example, the first aircraft in Gate 1 arrives at 4th unit time and departs at 48th unit time. Gantt Chart of this sample assignment can be seen in Figure VII.3.

```
GATE 1 [(4,48), (54,88), (94,116), (139,168), (203,240)]
GATE 2 [(25,59), (75,113), (140,175), (206,231)]
GATE 3 [(28,60), (61,86), (95,117), (146,167), (167,192), (193,227)]
GATE 4 [(44,70), (80,111), (112,148), (148,183), (183,211)]
GATE 5 [(45,73), (79,105), (132,177), (201,231)]
```

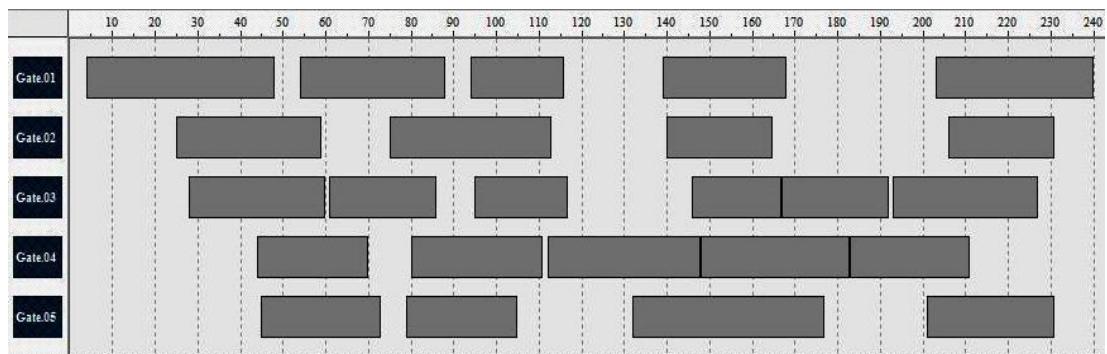


Figure VII.3 Gantt Chart – Gate Assignment of Sample Data

The details of the algorithm and the java code are as follows.

- 1- Choose a flight randomly among the flights have not assigned to apron.
The algorithm chose the fifth flight on the Gate 3 and called it as *first*.
- 2- Choose a flight which has overlap time with *first* among the flights assigned to gates other than the flights assigned Gate of *first*.
The chosen flight is the fourth flight assigned to Gate 1 and denoted as *second*.

After this step two flights with which it will be attempted to find compatible intervals.

3- Add flight *first* to Interval A and *second* to the Interval B.

Interval Class is created with instance variable, *intervalFlights*, TreeSet consists of **Flight** objects (TreeSet<Flight> intervalFlights).

Interval class has extendRight and extendLeft methods. extendRight() extends the current interval by adding the flight that is just right to it. It returns false only if the current interval includes the last flight on the Gate schedule, otherwise returns true and adds the flight. On the other hand, extendLeft() extends the current interval by adding the flight that is just left to it.

The other methods; *getT1*, *getT2*, *getT3* and *getT4* is shown by the help of the example.

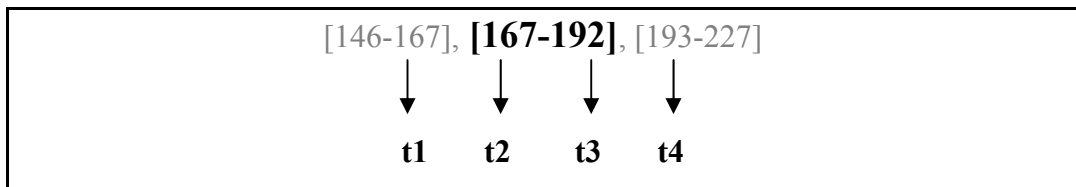


Figure VII.4 t1, t2, t3 and t4 time points of current Interval A

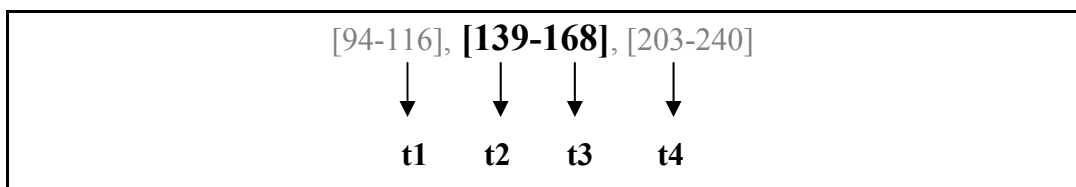


Figure VII.5 t1, t2, t3 and t4 time points of current Interval B

getT1() returns t1, the departure time of previous flight at the same Gate and *getT4()* returns t4, the arrival time of the next flight at the same Gate. Additionally, *getT2()* returns t2, the arrival time of the first flight of *intervalFlights* (Interval A in our example) and *getT3()* returns t3, the departure time of the last flight of Interval A. Because for now, there is only one flight on the **Interval** A, the first flight and the last flight of **Interval** A are the same flight.

- 4- Initialize success ← true.
- 5- Find compatible intervals, exchangeable intervals without conflict and size mismatch.

```
while (!areIntervalsCompatible(IntervalA, IntervalB) && success == true) {
  if (IntervalA.getT2() < IntervalB.getT1() // 167 <? 116 false
      && !IntervalB.extendLeft()) // do not come to this point
    success ← false // do not come to this point
```

```
  if (IntervalB.getT2() < IntervalA.getT1() // 139 <? 167 true
      && !IntervalA.extendLeft()) //false
    success ← false; // do not come to this point
```

At this step because *extendLeft()* returns true, the previous flight of Interval A is added to intervalFlights of Interval A.

Interval A ← [95-117], [146-167], [167-192], [193-227]



```
  if (IntervalA.getT3() > IntervalB.getT4() //192 >? 203 false
      && !IntervalB.extendRight()) // do not come to this point
    success = false; //do not come to this point
```

```
  if (IntervalB.getT3() > IntervalA.getT4() //168 >? 193 false
      && !IntervalA.extendRight()) // do not come to this point
    success = false; //do not come to this point
```

Interval A ← [95-117], [146-167], [167-192], [193-227]

Interval B ← [94-116], [139-168], [203-240]

The Interval A and B are exchangeable at this point. Two flights of Interval A are changeable with the flight of Interval B, and there will not be a conflict. Therefore, *areIntervalsCompatible()* returns true. Although the *success* is still true, the algorithm stops at this point.

- 6- Keep the current solution as *tempSol* variable.
- 7- Do the Interval Exchange Move

After finding compatible intervals, the exchange of the two intervals can be done easily. The arrangements of flights of Gate1 and Gate 2 before and after the interval exchange move are as below. Additionally the two gates before the move can be seen from the Figure VII.6.

GATE 1 [(4,48), (54,88), (94,116), (139,168), (203,240)]
 GATE 3 [(28,60), (61,86), (95,117), (146,167), (167,192), (193,227)]

GATE 1 [(4,48), (54,88), (94,116), (146,167), (167,192), (203,240)]
 GATE 3 [(28,60), (61,86), (95,117), (139,168), (193,227)]

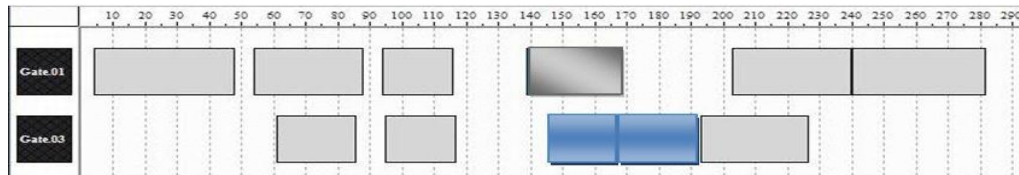


Figure VII.6 Two Gates before the Interval Exchange Move

8- Check if the algorithm has been to certain state before.

If so return *tempSol*.

If not add to tabu list.

The solution that has the flights listed in the same order should be tabooed which have seen before. To form the tabu list, a class **Schedule** is needed. This class taboos each solution as the two dimensional array. **Schedule**[*i*][*j*], *i* represents the gate number and *j* represents the flight number, has a cost and HashCode.

With one **Schedule** class object *best*, it is possible to hide the **Schedule** that has best cost value. When performing any move, the cost of the current solution is checked whether it has better cost value or not than the solutions that have visited before. If so, the revision of the *best* with the **Schedule** of new solution will be done.

VII.1.2 Apron Exchange Move

The Apron Exchange Move is used to deal with the flights that are assigned to the apron (Ding et. al., 2004). In each move, it is looked for exchanging one flight that has been assigned to the apron currently with a flight that has been assigned to a gate.

Below in Figure VII.7, there is an outline of the steps of the Apron Exchange Move. The *Apron Exchange Move Algorithm* is illustrated with the sample data again. The example is one in the previous section. Additionally the apron and assigned aircrafts to the apron are represented.

Algorithm I – Apron Exchange Move

- 1- Select randomly a flight i in apron.
- 2- Initialize Interval 1 \leftarrow flight i , success \leftarrow false
- 3- Choose a flight j which has been assigned to a gate that has overlap time with flight i and add to flights of Interval 2.
- 4- If the Interval1 and Interval2 are compatible
- 5- success \leftarrow true
- 6- Else step 3.
- 7- If (success)
- 8- Keep the current solution as *tempSol* variable
- 9- Interchange the flight assigned to apron with the chosen flight of gate.
- 10- if has been to certain state before
- 11- return *tempSol* else add to tabu list
- 12- Else output “Apron Exchange is failed”

Figure VII.7 Pseudocode of Algorithm I

```
GATE 1 [(4,48), (54,88), (94,116), (139,168), (203,240)]
GATE 2 [(25,59), (75,113), (140,175), (206,231)]
GATE 3 [(28,60), (61,86), (95,117), (146,167), (167,192), (193,227)]
GATE 4 [(44,70), (80,111), (112,148), (148,183), (183,211)]
GATE 5 [(45,73), (79,105), (132,177), (201,231)]
APRON:
[(41,74), (50,89), (79,126), (90,133), (157,189), (165,201), (203,242)]
```

When the flights that should be assigned to apron are added, the assignment solution is gotten as in above. In that schedule the gate utilization rate that wanted to be maximize is % 73.39.

Firstly a flight is selected among the flights of apron randomly. The selected flight in apron is [79,126] and it is added to the flights of **Interval 1**. Then the candidate flights that have overlap time with **Interval 1** are selected. Among them [79,105] is chosen because *compatible Intervals* are obtained when **Interval 2** is initialized with it. As a result the two **Intervals** can be exchanged.

GATE 5 [(45,73), (79,105), (132,177), (201,231)]

APRON:

[(41,74), (50,89), (79,129), (90,140), (152,189), (162,208), (203,242)]

GATE 5 [(45,73), (79,129), (132,177), (201,231)]

APRON:

[(41,74), (50,89), (79,105), (90,140), (152,189), (162,208), (203,242)]

After move the new gate utilization rate is % 76.83 (763/993). It is clear that the objective function value is increased only with a single move.

In the problem formulation of Ding et. al. (2004), one of the objective functions is maximization of flights assigned to gates. Because the minimal number of flights out of the gates has been determined by the greedy algorithm, they do not need to apply more variable moves. However getting higher objective function values with exchanging many flights at a gate by a flight at apron may be possible. Even in the situation that exchange of two consecutive flights at a gate with a single flight at apron, higher gate utilization rate can be observed.

In practice, the aircrafts' time period on the ground show more variability than the randomly generated test data of ours. In real data the ground time period ranges over a wide term, in random data ground time is assumed between 25 and 50 minutes. Both types of data are used for making analysis.

VII.1.3 Insert and Remove Move

When flight and gate size compatibility constraint which is hard constraint is added, it is seen that the greedy algorithm and other tried algorithms cannot give qualified solutions. There is no guarantee that the maximum number of flights is assigned to gates as well as maximum gate utilisation rate is achieved.

Because one less flight in apron has great effect on the objective function value, the ways to remove a flight from apron and insert this flight to the flights of a gate is searched. Remove and Insert Move is created which can make considerable improvement on the current solution.

The algorithm steps are as in the figure below.

Algorithm II- Remove and Insert Method

- 1- Select randomly a flight assigned to apron, Flight *a*.
- 2- Iterate gates; from Gate 0 to Gate *GateCount*
- 3- find the set of flights in Gate *m* that has overlap time with Flight *a*
- 4- if seekForRemove() for each flight in the set,
- 5- success ← true
- 6- stop iteration
- 7- else clear the set of flights and *m++*
- 8- if (success)
- 9- insert Flight *a* to Gate *m*

Figure VII.8 Pseudocode of Algorithm II

The *seekForRemove* method that tries to remove a single flight from its gate and insert it to another gate is defined. The purpose is providing the sufficient empty time period for the flight in apron. When the remove of all the flights in the flight set as in the example below is performed, the flight at apron can be inserted to the gate.

(255,515,B)← selected flight in apron; **Flight a**

[(270,345,A), (390,465,A), (500,575,B), (595,720,C), (775,850,A), (925,1030,C), (1030,1125,A), (1130,1210,B), (1265,1454,B)] ← the flights of **Gate m**.

[(270,345,A), (390,465,A), (500,575,B)] is the overlapped flight set; flight s, flight e and flight t consecutively.

There are sufficient free time period for inserting two of three flights to Gate *i* and one flight to Gate *j*. The Flight s;(270,345,A) and e;(390,465,A) can be inserted to the Gate *i* without conflict and size mismatch. Again, the remained flight t;(500,575,B) can be inserted to the Gate *j* and there will be no conflict and size mismatch.

[(60,225,A), (510,630,A), (660,735,A), (790,870,A), (870,965,A), (1015,1085,A), (1255,1454,A)] Gate **i**.

[(215,495,E), (585,680,B), (690,750,B), (765,885,C), (960,1035,B), (1110,1185,B), (1200,1315,E), (1325,1454,B)] Gate *j*.

So simply insert the flights of set to Gate *i* and Gate *j*.

[(60,225,A), (270,345,A), (390,465,A), (510,630,A), (660,735,A), (790,870,A), (870,965,A), (1015,1085,A), (1255,1454,A)] Gate *i*.
 [(215,495,E), (500,575,B), (585,680,B), (690,750,B), (765,885,C), (960,1035,B), (1110,1185,B), (1200,1315,E), (1325,1454,B)] Gate *j*.

Because Flights *s,e,t* is removed from Gate *m*, the new Gate *m* is ready for accepting Flight *a* currently at apron.

[(595,720,C), (775,850,A), (925,1030,C), (1030,1125,A), (1130,1210,B), (1265,1454,B)] ← Gate *m*.

Therefore, Flight *a* can be inserted to the Gate *m* easily.

[(255,515,B), (595,720,C), (775,850,A), (925,1030,C), (1030,1125,A), (1130,1210,B), (1265,1454,B)] ← Gate *m*.

The mathematical representation of Remove and Insert Move is as follows.

$$(s, e, t; m) \rightarrow (s; i) (e; i) (t; j) \text{ and } (a; apron) \rightarrow (a; m)$$

The effect on the cost function is reasonably high. However, when Insert and Remove Move is adapted, it is seen that this type of arrangement is hard to be found. At the set of intensive flight data, the possibility of performing this move is more. Moreover, in the situations that gates can meet the flight data, it is observed that the need for this move is low.

VII.2 RESULTS

Each table below shows test results by applying firstly a greedy algorithm to find initial solution and then adapting tabu search algorithm on the real daily data of International Terminal of Atatürk Airport. For the applied tests, the daily data of different terms of the year 2009 and the schedule belongs to first days of 2010 are used. Because there are 23 gates on International Terminal of Atatürk Airport, the problem size is determined by the flight number. At this point, it is better to recall that the flight data is pair wise; each arrival flight has a counter departure flight. Therefore, the actual flight number equals twice the number taken place as flight number on the table.

At the tables the cost of initial solution (column 4) and the result of tabu search applied to this initial solution (column 5) can be seen. The cost value covers two objectives which are minimization of total ground time period at apron and total passenger walking distance. Finding initial solutions takes up less than 1 CPU time (second), so the CPU times of greedy and tabu algorithms are not mentioned separately. The CPU times (second) that is seen at the tables are for the application from beginning to end.

The process of tabu search is terminated after the prespecified number of generations. The number of generations for termination is determined as 2000. While at each generation the Interval Exchange Move and Apron Exchange Move are being performed randomly, Remove and Insert Move (R. and I. Move) is iterated on the flights assigned to apron for removing a flight from apron and assigning this flight at a gate. Because, the number of performed Remove and Insert Move is an important factor that affects objective function (value) appreciably, is taken place on the tables.

Lastly, the improvement rate that is found by comparing the initial solution and the best solution found by adapting 2000 generations of tabu search process to this initial solution is added to the table. If someone compares the tables, the only difference at each table will be seen as the methods for initial solution. However, these methods

for initial solution also influence the efficiency of tabu search. At each table lots of factor shows variety and are needed to be analysis separately.

Table VII.1 The Results of Real Flight Data of AHL with SADT Algorithm and Tabu Search

Date	Gate x Flight	CPU Time	Cost of SADT Alg.	Cost of TABU Tec.	R. and I. Move	Improvement %
01.03.2010	23*210	29 sec.	760.356	532.874	2	30
01.03.2010	23*205	25 sec.	511.956	282.328	2	45
01.04.2010	23*274	109 sec.	1.794.058	1.431.866	0	20
01.04.2010	23*264	85 sec.	1.308.758	978.496	0	25
01.05.2010	23*149	14 sec.	685.468	487.008	3	29
02.01.2009	23*199	38 sec.	711.584	496.376	0	30
01.12.2008	23*223	30 sec.	532.610	417.364	0	22
03.01.2009	23*185	4 sec.	273.434	168.656	4	38
04.01.2009	23*212	22 sec.	476.468	327.588	0	31
05.01.2009	23*203	24 sec.	403.944	314.840	0	22

At the above table, the method called as SADT by us and called as greedy algorithm by Ding et. al. (2004) is used for finding beginning solution. After that, tabu algorithm adapted again same as their study. The logic behind this approach is minimizing the flight number by applying the greedy algorithm and by adapting tabu search getting optimal total walking distances. However the problem defined by us has different objective function and constraints than them, so the optimum initial solution cannot be found by implementation of their methods. Nevermore, when comparison is done with other implementation results it can be seen that tabu search with SADT algorithm gives moderate results.

Table VII.2 The Results of Real Flight Data of AHL with FCFS Algorithm and Tabu Search

Date	Gate x Flight	CPU Time	Cost of FCFS Alg.	Cost of TABU Tec.	R. and I. Move	Improvement %
01.03.2010	23*210	30 sec.	608.290	460.140	3	24
01.03.2010	23*205	16 sec.	431.906	288.080	2	33
01.04.2010	23*274	111 sec.	1.481.654	1.315.216	0	11
01.04.2010	23*264	81 sec.	1.144.652	948.516	0	17
01.05.2010	23*149	4 sec.	555.424	412.624	4	26
02.01.2009	23*199	30 sec.	602.948	493.794	0	18
01.12.2008	23*223	27 sec.	398.906	324.752	0	19
03.01.2009	23*185	3 sec.	204.282	120.378	2	41
04.01.2009	23*212	23 sec.	391.334	312.932	1	20
05.01.2009	23*203	17 sec.	351.094	252.392	1	28

At the above table, the method called as FCFS is preferred for finding initial solution. After that, tabu algorithm adapted to these results. When the initial results are compared with SADT algorithm, on the average 18 % better results can be seen from the fourth column of Table VII.2. According to the comparison results of best solutions with tabu search application, again better cost values can be seen. The improvement rate of best solution is higher for tabu search with SADT than tabu search with FCFS algorithm. However, for nearly all data sets tabu search algorithm with FCFS method finds better quality results than tabu search with SADT algorithm.

Table VII.3 The Results of Real Flight Data of AHL with LPT Algorithm and Tabu Search

Date	Gate x Flight	CPU Time	Cost of LPT Alg.	Cost of TABU Tec.	R. and I. Move	Improvement %
01.03.2010	23*210	63 sec.	542.700	456.740	15	16
01.03.2010	23*205	84 sec.	375.920	295.528	12	21
01.04.2010	23*274	206 sc.	1.428.262	1.332.208	19	7
01.04.2010	23*264	191 sc.	1.070.396	991.794	8	7
01.05.2010	23*149	8 sec.	396.342	382.184	2	4
02.01.2009	23*199	81 sec.	714.614	528.148	23	26
01.12.2008	23*223	61 sec.	431.434	328.516	13	24
03.01.2009	23*185	4 sec.	182.602	115.026	8	37
04.01.2009	23*212	55 sec.	422.330	359.110	5	15
05.01.2009	23*203	40 sec.	346.316	259.902	10	25

The LPT algorithm, assigns the flights first which have longer processing time on the terminal, is used to find starting point on the above table. Then again the tabu search is run during 2000 generations. The first thing that is realized from the table is the CPU times in seconds are longer than the other implementation processes of same input. The other important distinction is the number of Remove and Insert Move performed during tabu search is higher than the implementation results of the SADT and LPT algorithms. Actually this means that, there is substantial difference between the ranges of solution sets that tabu search moves visited. The number of visited solutions with LPT algorithm and tabu search is much more than the other methods although tabu search algorithms are terminated after same number of generations.

For this sample data, there is no information of optimum solutions, so it is only possible to compare with each other. Recall from the results section in Chapter VI,

there is no algorithm which gives the best initial solution for all the data sets. Again it can be deduced from the new results that there is no method which gives better solution for all the data sets.

CHAPTER VIII

CONCLUSION AND FUTURE RESEARCH

VIII.1 CONCLUSION

A typical Gate Management System optimizes the use of gates, stands and parking positions to decrease the airport's costs and to provide best service to the airlines and their passengers.

During the airline daily operations, assigning the arriving aircrafts to the available gates based on the published schedule is a very important issue. All of the models in the literature have their own objectives and constraints. Therefore it is not possible to compare these methods according to their success. However, an opinion about all of the gate assignment solving methods is obtained.

In this thesis, the over constrained AGAP is considered with a weighted single objective function. The objective function combines two objectives, minimizing the number of flights assigned to the apron and minimizing the total walking distances. Some greedy approaches for initial solution are tried and adapted. Moreover, Tabu Search algorithm is proposed with a new neighborhood search technique, the Remove and Insert Move, which helps us to get better quality solutions than previously employed insert move used for this problem. The particular constraints same as size compatibility are enforced to the determined appropriate solution methods. Because aircraft-gate size compatibility makes the problem complex and very hard to find optimum solutions, this constraint is added to very few study in the literature as a hard constraint.

After all, the implementation is run by using the indiscriminate and real data and results are reported. The three models for initial solutions are compared with each

other. The LPT algorithm, assigns the flights first which have longer processing time on the terminal, is preferred to find starting point at one of the three methods. The LPT method is not among the methods seen in the literature to find initial solution for AGAP. According to the results, the number of Remove and Insert Move performed during *tabu search adapted after LPT* is higher than the implementation results of *tabu search with SADT* and *tabu search with FCFS*. This means there is a substantial difference between the ranges of solution sets that tabu search algorithm visits. The number of visited solutions with *LPT algorithm and tabu search* is much more than the other methods although tabu search algorithms are terminated after same number of generations.

Additionally, most of the best results among the three approaches can be obtained by *tabu search with LPT*, and some can be obtained by *tabu search with FCFS*. In brief, results show the superiority of *tabu search with LPT* in the AGAP.

VIII.2 FUTURE RESEARCH

At each step of this thesis study some new approaches were improved. These new approaches can be a basic resource for new researches. Additionally this project will be an essential framework for other researches and the author's PhD thesis in academic life. Some other metaheuristic or specifically designed heuristics can be applied for solving the problem.

REFERENCES

- Airports Council International, 2007 World Airport Traffic Report (2007), http://www.airports.org/cda/aci_common/display/main/aci_content07_c.jsp?zn=aci&cp=1-5-54_666_2__, (13.04.2009).
- Babic, O.; Teodorovic, D.; Tosic, V.: "Aircraft Stand Assignment to Minimize Walking", *Journal of Transportation Engineering*, 110:1 (1984) 55-66.
- Barták, R.; Michela, M.: *Integration of AL and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, Springer, Berlin, Heidelberg, Germany (2005).
- Berber, M. F.: *Personal Meeting*, (TAV Airports Holding), (2009).
- Bihir, R. A.: "A Conceptual Solution to the Aircraft Gate Assignment Problem Using, 0,1 Linear Programming", *Computers and Industrial Engineering*, 19 (1990) 280-284.
- Bolat, A.: "Assigning Arriving Flights at an Airport to the Available Gates", *Journal of Operational Research Society*, 50 (1999) 23-24.
- Bolat, A.: "Models and a Genetic Algorithm of Static Aircraft-Gate Assignment Problem", *Journal of the Operational Research Society*, 52 (2001) 1107-1120.
- Brazile, R. P.; Swigger K. M.: "GATES: An Airline Gate Assignment and Tracking Expert System", *IEEE Expert*, Summer (1988) 33-39.
- Brazile, R. P.; Swigger K. M.: "Generalized Heuristics for the Gate Assignment Problem", *Control and Computers*, 19 (1991) 27-32.
- Burkard, R. E.; Çela, E.: "Quadratic and Three-dimensional Assignments: An Annotated Bibliography", *Technical Report 63*, Discrete Optimisation Group, Technische Universität Graz, Austria, (1996).

Burke, Edmund K.; Kendall, G.: "Tabu Search", *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, 1st Edition.; Springer, New York, USA, **(2005)** 168-170.

Cheng, Y.: "Network-Based Simulation of Aircraft at Gates in Airport Terminals", *Journal of Transportation Engineering / American Society of Civil Engineers*, 124:2 **(1998)** 188-196.

Corman, H. T.; Leiserson, C. E.; Rivest, R. L.; Stein, C.: *Introduction to Algorithms*, 2nd Edition, MIT Press **(2001)** 370-382.

Diepen, G.; v. d. Akker, J. M.; Hoogeveen J. A.: "Integrated Gate and Bus Assignment at Amsterdam Airport Schiphol", *ATMOS 2008 - 8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems* **(2008)**.

Ding, H.; Lim, A.; Rodrigues, B.; Zhu, Y.: "New heuristics for Over-constrained Flight to Gate Assignments", *Journal of Operational Research Society*, 55 **(2004)** 760-768.

Dorigo, M.; Stützle, T.: *Ant Colony Optimization*, Massachusetts Institutes of Technology Press, United States, America **(2004)** 33-35.

Dorndorf, U.; Drexler, A.; Nikulin, Y.; Pesch, E.: "Flight Gate Scheduling: State-of-the-Art and Recent Developments", *Omega*, 35 **(2005)** 26-334.

Drexler, A.; Nikulin, Y.: "Multicriteria Airport Gate Assignment and Pareto Simulated Annealing", *IIE Transactions*, 40:4 **(2008)** 385-397.

Erzberger, H.: "Design Principles and Algorithms for Automated Air Traffic Management", *NATO / AGARD Lecture Series 200 on Knowledge-based Functions in Aerospace Systems*, 7 **(1995)** 1-31.

Gambardella, L.M.; Taillard, E.; Dorigo, M.: "Ant Colonies for the Quadratic Assignment Problem", *Journal of the Operational Research Society*, 50 **(1999)** 167-176.

Gendreau, M.: "An Introduction to Tabu Search", *Handbook of Metaheuristics*, 1st Edition.; Glover, F. W.; Kochenberger, G. A. Editors.; Springer, New York, USA, **(2003)** 37-54.

Glover, F.; Laguna, M.: "Tabu Search" **(2005)**
http://www.dei.unipd.it/~fisch/ricop/tabu_search_glover_laguna.pdf (01.07.2009).

Gosling, G. D.: "Design of an Expert System for Aircraft Gate Assignment", *Transportation Research*, 24A **(1990)** 59-69.

Gökten, M.: *Personal Meeting*, (TAV Airports Holding), **(2010)**.

Gu, Y.; Chung, C.: "Genetic Algorithm Approach to Aircraft Gate Reassignment Problem", *Journal of Transportation Engineering*, 125 **(1999)** 384-389.

Güden, H.; Vakvak, B.; Özkan, B. E; Altıparmak, F.; Dengiz, B.: "Genel Amaçlı Arama Algoritmalarıyla Benzetim Eniyilemesi: En İyi Kanban Sayısının Bulunması", *Endüstri Mühendisliği Dergisi*, 16-1 **(2005)** 2-15.

Haghani, A.; Chen, M.: "Optimizing Gate Assignments at Airport Terminals" *Transportation Research*, A 32 **(1998)** 437-454.

Hamzawi, S. G.: "Management and Planning of Airport Gate Capacity: A Microcomputer-Based Gate Assignment Simulation Model" *Transportation Planning and Technology*, 11 **(1986)** 189-202.

Hochbaum, D.; "Algorithms", *The Scheduling Problem* **(1999)**
<http://riot.ieor.berkeley.edu/riot/Applications/Scheduling/algorithms.html>
(23.03.2010).

Hu, X. B.; Paolo, E. D.: "An Efficient Genetic Algorithm with Uniform Crossover for the Multi-Objective Airport Gate Assignment Problem", *IEEE Congress on Evolutionary Computation, CEC 2007*, Singapore, **(2007)** 55-62.

Kelemen, Z.: "Resource Management System - The First Step to the Airport Information System Integration", *Periodica Polytechnica Ser: Transportation Engineering*, 33 (1-2) **(2005)** 15-24.

- Kirkpatrick, S.: “Optimization by Simulated Annealing-Quantitative Studies”, *Journal of Statistical Physics*, 34 (1984) 975–986.
- Kirkpatrick, S.; Gelatt, C.; Vecchi, M.: “Optimization by Simulated Annealing” *Science*, 220 (1983) 671–680.
- Krauter, K. R.; Khan, A. M.: “Planning and Management of Airport Gates: A Simulation Methodology” *ITE Journal*, September (1978) 31-37.
- Mock, K.: “Greedy Algorithms”, *Course Handouts* (2002) 1-7,
<http://www.math.uaa.alaska.edu/~afkjm/cs411/handouts/greedy.pdf> (15.03.2010).
- Nowak, I.: “Recent Advances in Airline Crew Scheduling”, *Lufthansa Mathematik Symposium*, Frankfurt, Germany (2008).
- Özdemir, U.: “Methodology for Crew-Pairing Problem In Airline Crew Scheduling”, *Ms thesis*, System and Control Engineering, Bogaziçi University, Istanbul, Turkey (2009) 1-3.
- Pintea, C.; Pop, P.; Chira, C.; Dumitrescu, D.: “A Hybrid Ant-Based System for Gate Assignment Problem”, *Hybrid Artificial Intelligence Systems*, Springer Berlin, Heidelberg (2008) 273 – 280.
- Qi, X.; Yang, J.; Yu, G.: “Scheduling problems in the airline industry” In: Leung J.Y.-T, editor, *Handbook of Scheduling-Algorithms, Models and Performance Analysis* (2004) 1095-1098.
- Saudi, A.: “Greedy Algorithm”, *Lecture Notes* (2008) 6-15,
<http://www.azalisaudi.com/aa/AA-Week2-Greedy.pdf> (10.02.2010).
- Srihari, K.; Muthukrishnan, R.: “An Expert System Methodology for Aircraft-Gate Assignments”, *Computers and Industrial Engineering*, 21 (1991) 101-105.
- Stützle, T.: “Local Search Algorithms for Combinatorial Problems--Analysis, Improvements, and New Applications”, *PhD thesis*, FB Informatik, Technische Universität Darmstadt, Darmstadt, Germany (1998) 28-108.

TAV, Airports Holding: “Airports” *Istanbul Atatürk Airport* (2009) 1-3,
http://www.tavhavalimanlari.com.tr/index_en.asp(13.09.2009).

Xu, J.; Bailey, G.: “The Airport Gate Assignment Problem: Mathematical Model and a Tabu Search Algorithm”, *In Proceedings of the 34th Hawaii International Conference on System Sciences*, 3 (2001) 10-19.

Yan S.; Huo C.: “Optimization of Multiple Objective Gate Assignments”, *Transportation Research, A* 35 (2001) 413-432.

Yan, S.; Shieh, C.; Chen, M.: “A Simulation Framework for Evaluating Airport Gate Assignments”, *Transportation Research, A* 36 (2002) 885-898.

Yu, V. F.; Murty, K. G.; Wan, Y.; Dann, J.; Lee, R.: “Developing a DSS for Allocating Gates to Flights at an International Airport” *International Journal of Decision Support System Technology*, 1 (2009) 46-68.

Zak, J.; Jaskiewicz, A.; Redmer, A.: “Multicriteria Optimization Method for the Vehicle Assignment Problem in a Bus Transportation Company”, *Journal of Advanced Transportation*, 43-2 (2009) 658-664.

CURRICULUM VITAE

Canan ERSAN was born in Kadıköy in 1984. She graduated from Istanbul Atatürk High School in 2002. In the same year she admitted to Business Administration Department of Işık University. Addition to Management training, she applied to Minor Program of Industrial Engineering and got the Minor Program Certificate. In 2007, she admitted to Industrial Engineering Department of Marmara University for starting her graduate studies. Her research interests are graph theory, assignment problems and heuristics.

She worked in consulting industry between 2007 and 2009. She made analysis of innovation process of SMEs, and accomplished documentation process of applying international financial support funds.

MARMARA UNIVERSITY
THE INSTITUTE FOR
GRADUATE STUDIES IN PURE AND APPLIED SCIENCES

ACCEPTANCE AND APPROVAL DOCUMENT

The jury established by the Executive Board of the *INSTITUTE FOR GRADUATE STUDIES IN PURE AND APPLIED SCIENCES* on 24.05.2010 (Resolution no:2010/10-05) has accepted Ms Canan ERSAN's thesis titled **"Implementation of Airport Gate Assignment Problem"** as Master of Science thesis in Industrial Engineering (Industrial Engineering Program).

Advisor : Prof. Dr. M. Akif EYLER (Marmara University)

1. Member of the jury: Prof. Dr. M. Akif EYLER (Marmara University)

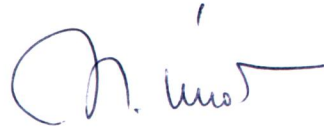
2. Member of the jury: Assist. Prof. Serol BULKAN (Marmara University)

3. Member of the jury: Assist. Prof. Ali Fuat ALKAYA (Marmara University)

Date: 09.06.2010

APPROVAL

Ms. Canan ERSAN has satisfactorily completed the requirements for the degree of Master of Science in Industrial Engineering at Marmara University. The Executive Committee approves that she be granted the degree of Master of Science on 20.08.2010 (Resolution no: 2010/16-02).



DIRECTOR OF THE INSTITUTE

Prof. Dr. Meral Ünal

