Optimization of the Gate Assignment Problem at Keflavík International Airport

Hanna María Hermannsdóttir

Thesis of 30 ECTS credits
Master of Science in Engineering Management

June 2015

Optimization of the Gate Assignment Problem at Keflavík International Airport

Hanna María Hermannsdóttir

Thesis of 30 ECTS credits submitted to the
School of Science and Engineering
at Reykjavík University in partial fulfillment
of the requirements for the degree of
Master of Science in Engineering Management

June 2015

Supervisors:

Eyjólfur Ingi Ásgeirsson, Ph.D., Supervisor
Assistant Professor, School of Science and Engineering at Reykjavík University

Hlynur Stefánsson, Ph.D., Supervisor
Associate Professor, School of Science and Engineering at Reykjavík University

Examiner:

Sigurður Óli Gestsson, C.Sc., Examiner
Managing director, Rhino Aviation.

Optimization of the Gate Assignment Problem at Keflavík International Airport

Hanna María Hermannsdóttir

30 ECTS thesis submitted to the School of Science and Engineering
at Reykjavík University in partial fulfillment
of the requirements for the degree of
Master of Science in Engineering Management

June 2015

Student:

_____

Hanna María Hermannsdóttir

Supervisors:

_____

Eyjólfur Ingi Ásgeirsson, Ph.D.

_____

Hlynur Stefánsson, Ph.D.

Examiner:

_____

Sigurður Óli Gestsson, C.Sc.

The undersigned hereby grants permission to the Reykjavík University Library to reproduce single copies of this project report entitled **Optimization of the Gate Assignment Problem at Keflavík International Airport** and to lend or sell such copies for private, scholarly or scientific research purposes only.

_____

Date

_____

–

Hanna María Hermannsdóttir
Master of Science

Optimization of the Gate Assignment Problem at Keflavík International Airport

Hanna María Hermannsdóttir

June 2015

ABSTRACT

The assignment of flights to gates is a complicated and an important scheduling problem that airport management faces daily. The assignment is complex due to different features gates can have and therefore all flights cannot be assigned to all gates. Gate assignments need to be suitable for the airport's operations and convenient for passengers. For Keflavík international airport (KEF), gates are an extremely limited resource and control the scheduled passenger traffic at the airport. Reaching maximum utilization of this limited resource is therefore very important. Today the assignment at KEF is done manually and nothing is being done to verify if the optimal solution being reached.

In this study the current practice of the gate assignment at KEF is examined and outlined. Based on the current practice a binary optimization model is developed. The objectives of the model are to minimize total passenger walking distance, the use of bus and walk-in, walk-out gates and to assign as many sensitive and heavy freight flights to the best suitable gates as possible. The model is tested with two weeks of real data, one week of low season and one of high season period, and the results are compared to the original gate assignment. All solutions are validated using a simulation program, ARCport. The results are promising and show that the developed model can be used to solve the gate assignment problem at KEF, replacing most of the manual work needed to execute the gate assignment. Manual interventions is however needed in some cases when running the model. The use of the model can save the airport time and effort and can possibly result in a better utilization of the airport's resources.

Keywords: Airport – Gate assignment problem – Binary optimization – Simulation.

Bestun stæðisúthlutunar á Keflavíkur flugvelli

Hanna María Hermannsdóttir

Júní 2015

ÁGRIP

Að úthluta flugum á stæði og hlið er flókið og mikilvægt skipulagsvandamál sem stjórn flugvalla stendur frammi fyrir á hverjum degi. Úthlutunin er flókin vegna mismunandi eiginleika hliða sem gerir það að verkum að ekki er hægt að úthluta öllum flugum á öll hlið. Stæðisúthlutunin þarf að vera viðeigandi fyrir rekstur flugvallarins og hentug fyrir farþega. Fyrir Keflavíkur flugvöll (KEF), eru hlið ákaflega takmörkuð auðlind og stjórna því áætlaðri farþega umferð á flugvellinum. Að ná fram hámarks nýtingu á hliðum er því afar mikilvægt. Í dag er úthlutunin á KEF gerð handvirkt og ekkert er gert til að sannreyna hvort að ákjósanlegustu úthlutun hafi verið náð.

Í þessari rannsókn er núverandi framkvæmd stæðistúthlutunar hjá KEF könnuð og útlistuð. Byggt á núverandi framkvæmd er bestunarlíkan með tvíundarbreytum þróað. Markmið líkansins er að lágmarka heildar gönguvegalengd farþega, lágmarka notkun á rútu- og gönguhliðum og að úthluta viðkvæmum flugum og flugum með mikla frakt á viðeigandi hlið ef það er mögulegt. Líkanið er keyrt með tveim vikum af raunverulegum gögnum, ein vika af háannatíma og ein vika af lágannatíma, og eru niðurstöður bornar saman við upprunalegu stæðistúthlutunina. Allar lausnir líkansins eru sannreyndar með hermiforriti, ARCport. Niðurstöðurnar gefa til kynna að líkanið geti verið notað til að leysa stæðistúthlutunina hjá KEF og kemur þá í stað handvirku vinnunnar sem var þörf á við framkvæmd stæðisúthlutunarinnar. Í einhverjum tilfellum er þó þörf á mannlegum afskiptum svo að líkanið geti fundið lausn. Þess er vænst að notkun líkansins geti sparað flugvellinum tíma og virði og geti mögulega skilað sér í betri nýtingu á aðföngum flugvallarins.

Lykilorð: Flugvöllur – Stæðisúthlutun – Bestun með tvíundarbreytum – Hermun.

ACKNOWLEDGEMENTS

This master thesis is the final project before receiving my master's degree in Engineering Management at Reykjavík University. I would like to express my gratitude to my supervisors Eyjólfur Ingi Ásgerisson and Hlynur Stefánsson for their support, guidance and assistance during the making of this thesis. I consider myself lucky to have had such great advisors who always showed interest and believe in the project.

Special thanks go to Isavia for giving me the opportunity of working on this project and for providing me with the data and the simulation program ARCport. I express my gratitude to all Isavia's employees who contributed to the making of the thesis, especially:

Guðmundur Daði Rúnarsson – for his help on developing the project and advices.

Pétur K. Ingólfsson – for his endless help regarding data collection.

Trausti Tómasson – for all his assistance and patience during the examining and outlining of the current practice of the gate assignment at KEF.

Thanks go to my good friend Ragna Sif Þórarinsdóttir for taking the time to proofread the thesis and to my brother-in-law, Kristófer Júlíus Leifsson, for his programming advices.

Finally I would like to thank my dear Valur Hermannsson and my family for their invaluable support and believe in me through my academic years.

May 2015, Reykjavík, Iceland

Hanna María Hermannsdóttir

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

GAP – The Gate Assignment Problem

KEF – Keflavík International Airport

PGAP - Passenger distance Gate Assignment Problem

AGAP – The airport gate assignment problem

BGAP - Minimum baggage transport distance GAP

PBGAP - The combined minimum passenger distance and baggage distance GAP

QAP - Quadratic Assignment Problem

GLPK – GNU Linear Programming Kit

1 INTRODUCTION

This chapter will present the guidelines for the thesis. First the background and statement of the problem are introduced along with the research aims and objectives. Followed by the contributions, methodology and limitations of the research. Finally the structure of the thesis is presented.

1.1 Background

Keflavík International Airport (KEF) is the main international airport in Iceland. In the year 2014, the airport served 3.867.418 passengers [1]. The airport's location, as shown in Figure 1-1, makes it an ideal transfer stop for travel between Europe and America and around 30% of all the airport's passengers are transfer passengers. The airport has two heavy traffic banks where most of the flights arrive and depart. A morning bank (5 a.m. to 8 a.m.) and an afternoon bank (2 p.m. to 5 p.m.).

Figure 1-1: Keflavík international airport's location [2].

Like other airports, KEF faces many decision and scheduling problems every single day. Disruption management, passenger flow, slot assignments and ground operations scheduling are all examples of scheduling problems faced by the airport's management. Strong competition between airlines and the demand of passengers for more comfort make those scheduling problems highly complex. One of the most important and most complicated airport management scheduling problem is the flight gate scheduling [3].

When aircrafts land at KEF they need to be assigned to a stand and if the aircraft is carrying passengers it also needs to be assigned to a gate. A stand is the parking position of an aircraft while a gate is the location inside the terminal where passengers enter when transferring from and to their aircraft. Stands are either next to the terminal and connected to a gate or they are located somewhere else at the airport, so called remote stands. During the airport's traffic banks these assignments can get complicated since flights and gates have different identifications and not all flights can be assigned to all gates. The problem of assigning flights to stands and gates is generally known as the gate assignment problem (GAP) [3].

The GAP directly influences the airport's operational costs and revenues. Total passengers' walking distance affects passengers' satisfaction and if passengers spend less time walking they have more time to spend at the airport's shopping facilities. When traffic is heavy at airports it is sometimes necessary to tow aircrafts between stands in order to get better utilization of the gates. If no gate is available a remote stand is often used and then passengers are transferred to and from the terminal with buses. Towing procedures are expensive as well as the use of resources involved with remote stands. Those are examples of matters controlled by the GAP.

All airports have limits on their operations. The limits are different between airports and depend on their size and their emphasis. Many airports' limits are based on the allowable number of flights per hour on their runway. Some airports base their limitations on passengers comfort by not allowing more traffic than so that every passenger should be able to sit down at the airport's terminal. For KEF and other airports the number of gates they have limits their traffic. KEF has reached its limitations during peak hours and temporary bus gates have been made in order to bus passengers to and from the remote stands. An extension to the airport's building is currently being built which is estimated to be ready in 2016. With the extension one terminal stand will be added and 6 bus gates.

Number of gates is one of the airport's biggest limitation and building new gates involves a time consuming and costly re-design of terminal building or the ramp, which is usually not feasible in the short run. It is therefore of great economic importance to get the best possible utilization of the existing gates [4]. The assignment of flights to gates is a daily task that needs to be executed carefully since it is the key factor to maximizing the capacity of the airport. Disruption to the assignment can happen any time because, for

example, of flights tardiness or failure. The gate assignment needs to be suitable for the airport's operations and convenient for passengers. Therefore there are numerous rules and guidelines that need to be taken into consideration when the assignment occurs. Today the stand and gate assignment at KEF is done manually and nothing is being done to verify if the optimal solution is being reached.

1.2 Problem statement

Gate assignment problem (GAP) is a problem every airport needs to solve. It is the problem of assigning flights, or rather the aircraft serving a flight, to distinct aircraft stands and gates, while meeting operational requirements and minimizing inconveniences to passengers [3,5]. This is a complex problem since airports only have limited number of gates and each gate can have different features. Therefore not all flights can be assigned to every gate. A gate can only serve one flight at a time. Gates are often categorized into two main categories depending on if passengers need to go through border control or not. The categories are often domestic and international gates [3]. The assignment of flights to gates needs to ensure that flights are assigned to relevant gates. There are many resources that need to be taken into consideration like flights, terminals, crews, baggage, ground handling and others [3]. All of those resources are independent which makes the GAP a complex problem. Inefficient gate assignment increases costs, worsens airline service quality and lengthens passenger transfer delay. It's an easily understood but difficult to solve problem. Integer programming (IP), simulation, expert systems and several kinds of heuristics have been applied to solve the GAP [6]. The problem is further described in Chapter 2.

1.3 Aim and objectives

The objectives of this study are to examine and outline the practice of the gate assignment at KEF and to develop an optimization model based on those outlines. This particular problem of gate assignment has been researched a lot and various mathematical models and solutions to the problem have been introduced in the literature. The aim is to develop a model which is suitable for the scenario of KEF.

One of the key questions is whether or not a simple optimization model can assign all flights to gates at KEF while ensuring fulfilment of the current practice. The model should pursue the optimal solution of the assignment problem as well as to automate the assignment process.

To validate the model, a simulation model named ARCport will be used. ARCport is currently being used at KEF as a tool for managing growth, simulating the use of existing facilities, and the design or planning of terminal facilities. By running the flight schedule with the gate assignment solution from the model through ARCport, errors regarding the stand and gate use are identified. The optimized solutions from the model will be compared to the original gate assignments to check if the model succeeds in creating a feasible solution for the gate assignment.

1.4 Research contributions

The main contributions of this study are the following:

- Formal analysis of the rules and guidelines for the gate assignment at KEF.
- Introduction to the solutions and methods that have been used to solve the GAP.
- A proposed decision support model for KEF to use in its gate assignments.
- Better ability for KEF to solve the GAP in regard to passengers' satisfaction.
- A program that automatically adjusts flight data to the format needed when using ARCport.

1.5 Research methodology

To execute the aim and the objectives of this study the GAP is analyzed by interviewing the current gate operator at KEF and other key members to the gate assignment. A model is then developed to meet the gate assignment's practice and in regard to passengers' satisfaction. The literature review presented in Chapter 2 will be used to help develop the model. Real data will be collected from Isavia and used to test the model. The model's validation will also be tested by using the simulation program ARCport.

1.6 Assumptions and limitations

Many types of flights fly to KEF, for example military flights, cargo flights, technical flights and passenger flights. The flights are either scheduled flights or diverted flights that are forced to land at KEF because of unforeseen events. The current operator of the gate assignment at KEF is responsible for assigning all flight types to stands and gates.

With gates being the most limited resource KEF has, this study will be limited to looking only at the passenger related flights since they are the only ones that need to be assigned a gate.

1.7 Structure of the thesis

This thesis is organized into 7 chapters. Chapter 1 holds the general introduction and the background of the problem being studied in this research. Chapter 2 is a literature review. Chapter 3 gives an overview of the functionality at KEF and its layout, followed by an outline of the current gate assignment process. In Chapter 4 the optimization model is formulated, explained and verified. An introduction of the simulation program ARCport is also presented. Chapter 5 describes the data that was needed to run the model along with the data handling and the problems with the data collection. The results are presented in Chapter 6. In Chapter 7 the discussions along with future work are presented. Chapter 8 contains the conclusions.

2 LITERATURE REVIEW

This chapter will go over the gate assignment problem (GAP) in more detail by introducing its common objectives, constraints and inputs. The chapter gives a review of some of the main algorithms that have been applied to the problem along with recent developments.

2.1 Objectives and constraints

The objectives of the GAP can be different depending on the airport's emphasis. They can be categorized into passenger-oriented and airport-oriented objectives. Total passenger walking distance, passenger delay, the number of flights cancellations and distance passengers have to carry their baggage are examples of passenger-oriented objectives. On the other hand, total gate preferences and number of aircraft towing procedures are examples of airport-oriented objectives [3]. The most frequent objectives are; to assign every flight to a gate, to minimize passengers total walking distances since it directly influences passenger's satisfaction and minimize towing procedures since they are very expensive.

When the objective of the GAP is to minimize the total walking distance for passengers it is often classified as a Quadratic Assignment problem (QAP) which has been proven to be NP-hard [7]. The reason for that is because the minimizing of walking distance for transfer passengers is based on the transfer volume between pair of aircrafts and the distance between every pair of gates. Only implicit enumeration methods are known for solving QAPs since they are NP-hard, such as branch-and-bound type algorithms, cutting plan methods, and dynamic programming [6].

Most of the inputs to the GAP have high uncertainty and can change over time. The data uncertainty can be caused by flight or gate breakdown, flight earliness or tardiness, emergency flights, severe weather conditions, errors made by staff and many others. If one flight arrives late at the gate it was assigned to, a "domino effect" can be created for the other flights that were assigned to the same gate. This scenario is something that the gate assignment needs to be able to handle since this is a common problem that can happen daily. It is not desirable to have to reschedule the gate assignment completely if some changes to the inputs occur. Therefore most recent gate scheduling techniques often aim to find a schedule which is flexible to changes of input data. The schedule would then not

necessarily be the optimal solution but a good, robust solution which is less sensitive to uncertainty. With flexible gate assignments operators have the possibility to react quickly and properly to accommodate necessary changes or updates in the flight schedule [3].

Typical constraints that follow the GAP are that a gate can only serve one aircraft at a time, gates can handle only flights operated by aircrafts of certain sizes, gates can handle only flights from certain origins and destinations, because of safety regulations, minimum time between two subsequent aircrafts must be assured and space restrictions and service requirements with respect to adjacent stands must be fulfilled [3,8].

2.2 Inputs

The main input for the GAP is a flight schedule with flight arrival and departure times and additional detailed flight information, including pair-wise links between successive flights served by the same aircraft (arrival and departure), the type of aircraft, the number of passengers, the cargo volume, the origin or destination of a flight and the type of flight (domestic/international or Schengen/non-Schengen). Gate preferences, required airport services and inspection facilities are also required as inputs to the GAP. These inputs define the time frame for processing a flight and the subset of gates to which it can or should be assigned [3,4]. If the objective of the GAP is to minimize the total passenger walking distance, the walking distances to each gate for each passenger type, arrival, departure and transfer, are needed.

2.3 Main algorithms

Various mathematical models and solutions to the GAP problem have been introduced since the early seventies. Many of these models are dependent on the airport being researched and its characteristics.

Braaksma and Shortreed [9] made one of the first attempts to improve gate utilization by examining the turnaround operation of passenger aircrafts in 1971. They constructed a simulation model based on the Critical Path Method (CPM) and used real data from Toronto International Airport. Their model was capable of determining the total turnaround time for certain aircraft types and identify the critical paths.

In 1984, Babic et al [10] formulated the GAP as a linear 0-1 IP with the objective to minimize average passenger walking distance. They used the branch and bound technique to find the optimal solution. In their model, transfer passengers were not considered. The

model was formulated with the limitations that all aircrafts could be served by all gates, there would always be at least one available stand for an arrival aircraft and that an aircraft arrives and departs at the same stand (no towing procedures). Their optimum solution mainly affected the number of passengers that walked maximum distances and minimum distances.

Mangoubi and Mathaisel [11] used a similar approach as Babic et al, with a linear programming relaxation of an IP formulation and greedy heuristic approaches to solve the GAP. They however included transfer passengers but not with as much precision as for the local passengers. For modelling simplification, the distance for a transferring passenger was determined using a uniform probability distribution between one gate and all other gates. Consequently, the transferring distance was equal to the average distance from one gate to every other gate.

All the studies mentioned above developed methods that are only suitable for solving small dimensional problems or limited their research to problems with a small number of aircrafts and gates involved. Zhang et al. [12] were one of the first researchers that modelled the GAP for a large airport in 1994. Six heuristic methods were tested by using real connection and distance data from American Airlines at O'Hare International airport in Chicago. They modelled the GAP as a QAP which had at that time not been solved using optimal algorithms when the size of the problem was larger than 15 gates. In their research the airport studied had a total of 38 gates, therefore the optimal algorithms were out of consideration. The six heuristics methods tested were: The method of increasing degree of freedom, CRAFT procedure, The Vollmann, Nugent, and Zartler Two-phase method, Cutting Plane and Exchange method, Modified cutting plane and exchange method and Simulated annealing method. Their results were that the assignment of aircrafts to gates can be modelled as a QAP. For practical application, when a quick solution is needed, the Vollmann, Nugent, and Zartler two-phase method is recommended; when high solution quality is desired, the Cutting Plane and Exchange method or the Simulated Annealing method should be used.

Haghani and Chen [6] were one of the first to examine the GAP in consideration to a non-passenger related objective. Unlike passenger walking distance, the baggage transport distance does not influence the passenger directly as long as baggage is delivered to the passenger at the right airport. Baggage transport distance however affects the length of

time which the flight has to remain at a gate and is therefore important in terms of operating costs. The authors formulated the Passenger distance Gate Assignment Problem (PGAP) as a new nonlinear (quadratic) 0-1 integer problem in order to measure the impact of passenger walking distances. Instead of solving the problem exactly they proposed a new heuristic approach for solving the PGAP at airports. They extended the QAP formulations by introducing time constraints. The new heuristic approach could effectively assign aircrafts to gates to minimize total passenger walking distances based on the time table of flight schedules. A similar formulation could also be used to solve the minimum baggage transport distance GAP (BGAP) as a QAP. The combined minimum passenger distance and baggage distance GAP (PBGAP) is a QAP too and therefore the algorithms developed to solve the minimum PGAP could also be used to solve the minimum PBGAP. The proposed heuristic algorithm obtained good solutions for the conventional GAP on several small problem sizes. The algorithm could not be used to obtain an optimal solution in a reasonable time when the number of gates was greater than 15. To handle larger problems the authors proposed to decompose the problem into several smaller ones.

Xu and Bailey [13] formulated the GAP as a mixed 0-1 quadratic integer programming problem and then reformulated it as a mixed 0-1 integer problem with a linear objective function and constraints. Their objective was to minimize the overall connection times that passengers walk to catch their connect flight. A simple Tabu search (TS) meta-heuristic was designed to solve the problem. The algorithm exploited the special properties of different types of neighborhood moves and created a highly effective candidate list strategy. Their TS heuristics could effectively achieve significant savings on passengers' connection time compared with the gate assignment in current airline operation.

Yan and Huo [14] proposed a dual 0-1 integer programming model to solve the GAP in 2001. The first objective tries to minimize passenger walking time while the second objective aims at minimizing passenger waiting times. To efficiently solve these large-scale problems in practice they used advanced techniques, such as the column generation approach and the branch-and-bound technique, to develop a solution algorithm. The model was tested with a case study regarding the operation of Chiang Kai-Shek airport. The results showed that the model could be useful for actual operations.

Yan continued researching the GAP in 2002 and presented, along with Shieh and Chen [15], a simulation framework that analyses the effects of stochastic flight delays on static gate assignments and evaluates flexible buffer times and real-time gate assignment rules. They used a 0-1 integer program as the optimization model to formulate static gate assignments. The simplex method coupled with the branch and bound technique was used to develop a solution algorithm. In order to compare the optimised solution to the optimal solutions that will be applied in the real-time operations they developed two heuristics. One that aims at minimizing the total passenger distances and the other one that does almost the same except that it sorts all flights in order of increasing arrival time. In order to test this framework, tests were performed on Chiang Kai-Shek airport operations. The results showed that the framework could be useful for airport authorities to perform gate assignments.

2.4 Recent developments

The most recent developments have focused on taking in the real multiple criteria nature of the problem and the development of robust flight schedules.

Ding et al. [5,16,17] consider the over-constrained GAP where the number of flights exceed the number of gates available, and where the objectives are to minimize the number of non-assigned flights and the total walking distances. They formulated the problem as a binary quadratic programming problem. A greedy algorithm was developed and Tabu search meta-heuristic was used to solve the problem. The greedy algorithm minimizes non-assigned flights while a new neighbourhood search technique, the Interval Exchange Move is devised. The new search technique brings flexibility in seeking good solutions for the over-constrained GAP.

Dorndorf [18,3] presented a model which assigns available airport flight gates to three possible aircraft activities: arrival, optional intermediate parking activity (ground time) and departure. The author looks at the model as a modified multi-mode resource-constrained project scheduling problem with a multi-criteria objective function. Branch-and-bound technique is used to solve the model. By supposing that one aircraft can be assigned to different gates during its ground time the airport managers get more freedom. For example, if one flight has long ground time and the aircraft is assigned to a gate, which has to be used very often, then it can be temporally removed to another gate to make the gate free for other assignments. The model aims to optimize several objectives

which are oriented on both passenger comfort and convenience for airport services. The objectives include maximization of the total assignment preference score, a minimal number of unassigned flights during overload periods and minimization of the number of tows. Goal weights are used to set the importance of the objective functions. The choice of appropriate preference weights and priorities as well as the ordering of the partial goals by importance using the weight factors can have a substantial impact on the optimal gate assignment.

Dorndorf et al. [4,19] further developed Dorndorf's previous mentioned model in relation to robustness. The maximization of a robustness measure as well as a minimal deviation from a given reference schedule were added to the objective function. They showed that in case of a one period time horizon this objective can easily be integrated into the previous model based on the Clique Partioning Problem. They also present a heuristic algorithm to solve the problem for multiple periods.

Diepen et al. [8] presented a linear programming formulation that is based on so-called gate plans in 2012. Their objective was to find a more robust solution in order to alleviate the problem gate planners at Amsterdam Airport Schiphol (AAS) have with constantly having to adjust the automatically generated planning to make it insusceptible against small deviations from the schedule. A gate plan is a subset of the flights that can be assigned to a single gate of the corresponding type. The authors managed to develop a two phase solution approach. In the first phase, flights are assigned to gate plans and in the second phase, planners at AAS assign gate plans to the actual gates. The second phase consists of a number of relatively small problems that can be solved by hand and in which additional operational constraints can be incorporated. For the first phase they presented a different way of formulating the gate assignment problem as an Integer Linear Program (ILP). Their algorithm focuses explicitly on robustness, and is expected to give the gate planners more time for solving larger conflicts that arise during the actual day.

Kim et al. [20] analysed trade-offs between three objective functions and proposed a balancing objective function in 2013. The first objective was to minimize passenger transit time. The transit time being the time from the security checkpoint to a gate, from a gate to baggage reclaim, and from one gate to another gate. The second objective was to minimize taxi time on ramps. The last objective was to minimize disturbances in gate operations or equivalently to maximize the robustness of the gate assignment. The authors manage in

their study to create a balancing objective function which satisfies all the three objectives which out performs the current gate assignment in every objective. They therefore showed that the efficiency of traffic flow in passenger terminals and on ramps, as well as on the robustness of gate operations, had a potential to be improved regarding the gate assignment of the studied airport.

3 KEFLAVIK INTERNATIONAL AIRPORT

In this chapter, an overview of the functionality at Keflavik international airport and its layout will be given, followed by an outline of the current gate assignment process. Interviews were carried out in order to outline the current gate assignment, three with the current operator of the gate allocation at KEF and one with the deputy terminal director. The actual gate allocation process at KEF today was observed one morning for that day and the following day. One meeting was also attended with the main stakeholders that operate at the airport, where their needs for the gate assignment at KEF were discussed.

3.1 Functionality

Passengers at KEF can be divided into three types; departing, arriving and transfer passengers. Departure passengers are those that arrive to the airport with ground transportation. They go through check-in facilities and security checks to go to the departure area and from there they walk to their departure gate. Arriving passengers arrive at the airport by aircraft. They then walk from the arrival gate to the baggage reclaim area. Transfer passengers go from one aircraft to another without leaving the airport. About 30% of all the airport's passengers are transfer passengers.

Since 25th of March 2001, Iceland has been a part of the Schengen cooperation along with 24 other European countries. The purpose of the cooperation is to make travel between the member's countries easier by not having to go through border control. All airports in the Schengen area, Figure 3-1, need to divide passengers that are flying within the Schengen area and those passengers that are flying to or from countries that are not in the Schengen area, non-Schengen countries. USA and United Kingdom are the two non-Schengen countries Iceland has the most air transport with. Passengers traveling to or

Figure 3-1: Schengen area [21].

from Non-Schengen countries need to go through border control. Passengers entering the country go through immigration while passengers leaving the country go through emigration. This also applies to passengers that do not leave the airport and are connecting to another country within the Schengen area [22].

All transfer passengers arriving at KEF from countries outside the European Economic Area (EEA) are required to go through security screening according to the regulation No 300/2008 of the European Parliament [23]. USA are an exception to this rule as their security standards have been recognised as equivalent to the EEA standards. Most passengers arriving from countries outside the EEA at KEF are passengers arriving from Canada. Because of this demand an additional security checkpoint for arriving passengers was built in the south building of the terminal. Since it has not been made possible to divide passengers that are arriving to Iceland from transfer passengers, all passengers arriving from countries outside the EEA need to go through the additional security screening. Because of its location, only passengers arriving at gate 5 and 6 can enter it. Frequently it happens that three flights from Canada are arriving at a similar time, therefore an entrance for passengers arriving with buses from the remote stands has been added.

3.2 Layout

A gate at an airport is the area inside the terminal where passengers enter when transferring from and to their aircraft. Stands however are the parking position of the aircraft. Stands at Keflavik international airport can be divided into three types;

- *Terminal stands,* stands that are right next to the terminal and passengers can walk from/to the aircraft to/from the terminal via bridge.
- *Walk-in, walk-out stands*, stands that are right next to the terminal but no bridge connects the aircraft to the terminal and therefore passengers have to walk from/to the aircraft to/from the terminal.
- *Remote stands,* stands that are not close to the terminal. Passenger arriving or departing at those stands need to take a bus between their gate at the terminal to the stand where their aircraft is parked.

Today KEF has eleven terminal stands, number 1-9 and 11-12, two walk-in, walk-out stands, number 10 and 14, and several remote stands. The terminal stands are the first

choice of all stakeholders since passengers can walk straight from the aircraft to the terminal via bridge. Both the terminal stands and the walk-in, walk-out stands are directly connected to gates since their location is right next to the terminal. The stands are either connected to one or two gates, depending on if both Schengen and non-Schengen passengers can enter the terminal, at that location, or only Schengen passengers. A stand which has two gates is called a dual stand. The first floor of the south building is for non-Schengen passengers. Non-Schengen passengers need to go through border control on the second floor and afterwards they go down to the first floor where all the non-Schengen gates are located. The bridge connected to the terminal stands is on the second floor of the terminal so non-Schengen passengers need to travel from their gates back up to the second floor to enter the bridge and vice versa. Therefore non-Schengen passengers walk longer distances than Schengen passengers at the same dual stand.

Remote stands number 20 to 24 are most used for passenger related flights but if needed other remote stands can be used. In the last two years, five bus gates have been added to the terminal, A, 8B, 8C, 25B and 32D in order to increase the usage of the remote stands. Unlike all the other gates, gates 25B and 32D are only arrival gates, not departure gates. Gate 25B was specially made for the transfer security. If more than two flights are coming in at a similar time that need to go through transfer security, passengers can be transferred with buses from the remote stands to gate 25B where they can enter the security. Gate 32D was created to ease the flow for non-Schengen passengers arriving with buses. All arriving non-Schengen bus passengers can go to that gate and from there they walk the shortest distance to border control of all the non-Schengen bus gates. The layout of the airport and most of the stands used for passenger related flights can be seen in Figure 3-2. All gates' identifications are listed in Table 3-1 where the last column, Tow, displays the gates that are best suitable for towing procedures.

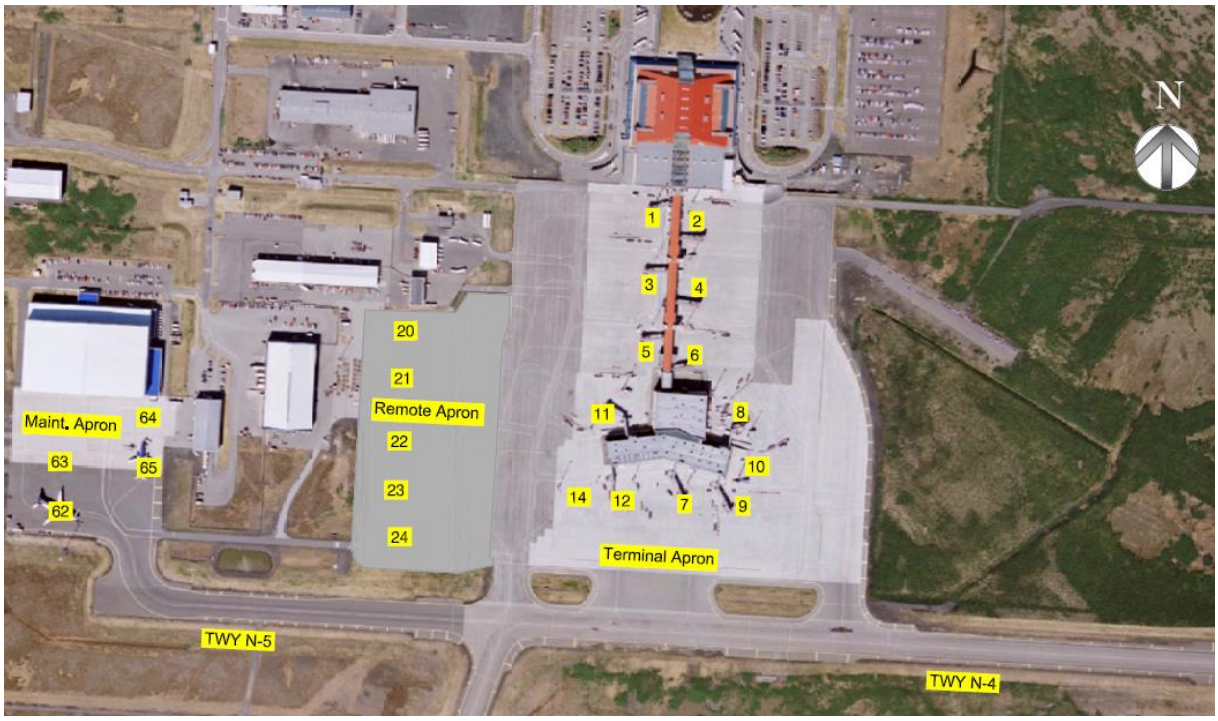

Figure 3-2: Layout of Keflavik International Airport [24].

Table 3-1: Gates identifications.

Gate	*Terminal gate*	*Bus gate*	*Walk-in gate*	*Schengen gate*	*Non-Schengen*	*Only arrival gate*	*Large gate*	*Freight gate*	*Canada arrival*	*Tow*
1	✓			✓						✓
2	✓			✓						
3	✓			✓						✓
4	✓			✓			✓			
5/25	✓			✓	✓				✓	✓
6/26	✓			✓	✓		(✓)		✓	
7/27	✓			✓	✓		✓	✓		
8/28	✓			✓	✓					
9/29	✓			✓	✓			✓		
10/30			✓	✓	✓					
11/31	✓			✓	✓					✓
12/32	✓			✓	✓		✓	✓		
14/34			✓	✓	✓					
8B/28B		✓		✓	✓					
8C/28C		✓		✓	✓					
A		✓		✓						
25B		✓			✓	✓			✓	
32D		✓			✓	✓				

3.3 Gate assignment process

To be able to fly to KEF, airlines need to apply for a slot which gives them the right to schedule an arrival or departure during a specific time period. When most of the airlines apply for slots they don't know which arrival and departure flights will be connected and connect them almost randomly before applying for slots. Except for some airlines that fly from one location to Iceland and back to the same location. The connection between arrival and departure flights needs to be known before the gate assignment because it has to be taken into account for how long the aircraft will hold the gate. The real connection of flights constantly changes and isn't known until the day before operation. Even after that the connections can change because of aircraft's breakdowns and delays. Because of this uncertainty the airport does the actual stand and gate allocation the day before and on the day of operation. Despite of only doing the gate allocation for a short amount of time at once there are still on average 3 to 4 changes made to the gate allocation each day.

Today at KEF the gate allocation is mostly done by hand. A program called SKOR is used to do the first assignment with the allocated slots. The program doesn't take into account any constraints, it only knows how many gates KEF has and if they are Schengen gates or dual gates. SKOR places all the flights as they have been listed when the airlines apply for a slot. It seeks best utilization of the resources KEF has, the gates. The actual gate assignment is then done by hand by an operator the day before and on the actual day of operation. The gate operator has gained knowledge over the years of all stakeholders' needs to the gate assignment. Based on that knowledge, the airport's functionality and layout, the gate assignment is executed.

3.3.1 Rules and guidelines

With a program called Stand planner the operator starts the gate assignment in the morning by manually linking flights together for the current day and the next. The stand planner would give error message if the linkage is not possible. The flights are then manually assigned to a gate, first after the airport's functionality and layout and then after guidelines based on stakeholders' needs. In heavy traffic it can be extremely difficult to assign each flight to gates and then there is need for flights to be split. When splitting a flight, the flight arrives at a terminal gate, if it's possible, and then it's towed to a remote stand where it waits until it departs. Before departure the flight is towed back to a terminal gate or passengers are transferred with buses to the aircraft. It's better to only have to bus

either arrival or departure passengers instead of having to bus both for the same flight. Flights that are not departing until the day after arriving often need to be split so they don't hold the gate for that amount of time. The airport can demand that an aircraft has been towed from a gate within 40 minutes from its arrival. This applies to all flights. Towing procedures should however be kept at an absolute minimum since they are expensive and can be dangerous. The operator tries to solve the GAP without splitting flights but if that is not possible he views manually which flights should be split. Flight that have long time between their arrival time and departure time are the best candidates for splitting. There are three gate assignment rules that cannot be broken, those are:

- Flights arriving from Canada and other countries outside the EEA, except for USA, need to be gates where passengers can enter the transfer security, see Table 3-1.
- All non-Schengen flights need to be assigned to dual gates where it is possible to lead non-Schengen passengers to border control.
- Large aircrafts, jumbo jets, can only be assigned to large gates. Today no scheduled passenger flights at KEF are with a large aircraft, or jumbo jets, but in the year 2016 Icelandair airline plans to add two large aircrafts to their fleet. There is however one aircraft that is longer than other passenger aircrafts at KEF which cannot be assigned to all gates. This aircraft needs to be assigned to one of the large gates but it can also be assigned to gate 6.

After these rules have been taken into account, practically all other flights can go to any gate. There are however many guidelines that the operator tries to be taken into account in the gate assignment. The most important one is placing sensitive flights to terminal gates. Sensitive flights are for example flights that have short turnaround time. They cannot endure being late because their slot on the airport they are flying to is strict and if they arrive late they can lose that slot. Sensitive flights are also flights that are flying long distances and flights with many transfer passengers. It's important that transfer passengers get from one flight to the next in the shortest amount of time since the flights have to wait for all their transfer passengers. Sensitive flights often have that in common that if they leave KEF late they will arrive late the next day. All sensitive flights need to be able to get their passengers on and off the aircraft as quickly as possible so the odds of the flight being late are kept at a minimum. The gates that are the most suitable for sensitive flights are terminal gates because passengers can walk straight from the aircraft to the terminal

via bridge. The next best gates after terminal gates are the walk-in, walk-out gates because passengers only walk short distances to get to the terminal. Bus gates are the least feasible gates for sensitive flights because it can take a long time to transfer all passengers from the aircraft to the terminal with buses. Other guidelines that are tried to take into account when assigning flights to gates but with lower priority are listed below:

- Place flights with heavy freights on stands where it is easy for the ground handler to load and unload the freight. Gates 7, 9 and 12 are the best gates for loading and unloading freight. This gives the ground handler more time to focus on other passenger related tasks like loading and unloading baggage.
- Place flights that need to be towed at gates 1, 3, 5 or 11 since it is shorter to tow them from there to the remote stands. The same applies if an aircraft needs to be towed to a gate from a remote stand. It's best to tow the aircrafts the shortest possible distance.
- Don't place two flights with the exact same departure time at two adjacent gates since it can be confusing for the passengers which queue for boarding they should go to. This is especially important if the flights have the same destination.
- Place flights that are flying shorter distances at bus gates since the remote stands don't have a gasoline tank. The aircrafts that park at those stands are instead filled with gasoline from cars.
- Place flights with the same airline at the same gate. The airline could put more emphasis on being on time if they know that the next flight arriving to the gate is their own. Also it's good for the ground handler to operate at same gate.
- Place flights that use the same ground handler at the same gate since that is more convenient for the ground handler as he can have his equipment ready. All ground handlers working at KEF have access to all gates so this is not mandatory.
- Place aircrafts from WOW airline at gates 1, 3, 5 and 11 since their offices have view over those gates and it's convenient for them to see their aircrafts' operations.

Although the main gate assignment is done during the morning, for the current day and the next day, this is an ongoing process that constantly needs to be watched. Flights can come in late or break down. Than the gate assignment needs to be converted. If one aircraft is somewhat late that could mean that another flight that was supposed to go to a remote stand can now go to a terminal stand. One of the biggest challenge of the gate assignment

at KEF are the Schengen gates number 1 to 4 since the growth in flights have mostly been to non-Schengen countries, America and Britain. It happens very frequently that there are not enough flights flying Schengen-Schengen in one traffic bank so there can be Schengen gates not being used throughout the entire traffic bank. Therefore a non-Schengen flight often has to use a bus gate while there is an unused terminal gate with a bridge.

4 THE MODEL

In this chapter the optimization model that was developed to solve the gate assignment problem at KEF is presented. First the model's objectives and constraints are introduced. Followed by an introduction of a mechanism to identify problematic flights. All indexes, data and decision variables are also introduced. Then the model is formulated and explained. The model is then verified by a small test case. Finally an introduction to the simulation program ARCport which will be used to verify the solutions from the model is presented.

To develop a model which assigns gates to three possible aircraft activities (arrival, ground time, departure) as was presented by Dorndorf [17] or that produces a robust gate assignment was thought to result in an overly complicated model for the preliminary stage of developing the first automated model to solve the GAP for KEF. With the purpose of this thesis being to develop a simple optimization model which possibly can solve the GAP at KEF it was decided to model it as a binary optimization model. Similar models were used in many of the researches introduced in Chapter 2, for example [10] and [16] where one flight consists of the arrival and departure flights.

KEF doesn't have access to the data about which transfer passengers are going to which gate, therefore a simplification needed to be made for the distance traveled by transfer passengers. Instead of traveling from gate to gate, transfer passengers travel to and from their gate to central points which are explained in detail in Chapter 5.1.2. Because of this simplification the GAP could be modeled as a linear optimization model instead of a QAP, as was needed in most of the studied researches. There are many rules and guidelines that the gate assignment at KEF tries to take into account as was introduced in Chapter 3.3.1. It was decided to implement only the most important ones to the model in order to keep the model simple and to keep the running time of the model at a minimum. Others can be added later without any technical difficulties. In order to solve the model with Gurobi 6.0.2 solver [25] the model was written in GNU math language, see Appendix A.

4.1 Objectives

The objective function of the model is divided into three main parts. The first part is to minimize the walking distance traveled by passengers. The second part is for minimizing flights that are assigned to the bus gates and the walk-in, walk-out gates. The third part is

to assign, if possible, sensitive flights to terminal gates and flights with heavy freight to gates where it is easy to load and unload freight. In order to set the preferences for each part of the objective function, weight factors are used which represent cost. Then it can be viewed, for example, as placing a flight at a bus gate costs more than placing a flight at a terminal gate. The preferences can be adjusted by changing the weight factors. Further description of the different parts of the objective function is listed below.

4.1.1 Walking distance

By keeping the total distance traveled by passengers at a minimum passenger's satisfaction will increase as distance traveled directly affects passenger experience. Total walking distances include the distance from gates to baggage reclaim areas for arriving passengers, distances from the check-in to gates for departing passengers and the distances traveled by the transfer passengers. In order to minimize the total walking distance the number of every passenger type traveling with each flight has to be known. This objective ultimately makes the model attempt to place flights with the most passengers at gates which have the lowest distances.

4.1.2 Use of bus gates and walk-in, walk-out gates

Placing passengers at bus gates and walk-in, walk-out gates where they have to walk outside can have much influence on the passenger experience and satisfaction. Especially for airports where weather conditions are not good. Instead of walking straight from the aircraft to an enclosed bridge, passengers need to walk down steep stairs from the aircraft and wait for the bus or walk straight to the terminal in weather conditions they have often not prepared for. Even though both types of gates are not desired by passengers, bus gates are considered to be a worse option. Passenger do not only have to walk down steep stairs and then a little distance to reach the terminal, they have to enter a bus, wait while the bus is filled with passengers and stay on the bus until the bus reaches the terminal, from there they have to walk outside and to the gate. Therefore it should be more desirable to assign flights to walk-in, walk-out gates rather than bus gates and that should be accounted for with the weight factors.

When the traffic is the heavy at KEF, the bus gates and the walk-in, walk-out gates must be used because the terminal gates are a limited resource. In those cases flights that are not sensitive and have the least amount of passengers should be placed at those lower service gates.

4.1.3 Sensitive flights and heavy freight flights to appropriate gates

It is preferred to assign flights that are sensitive or have heavy freight to suitable gates if it is possible, as was mentioned in Chapter 3.3.1. If many flights are sensitive and it is not possible to assign all of them to terminal gates the flights with the most passengers should be a priority. Sensitive flights have higher priority than heavy freight flights.

4.2 Constraints

The model's constraints cannot be violated under any circumstances. They ensure that practical limitations of the problem are taken into account and that the assignment rules of KEF listed in section 3.1.1 are not broken. The constraints are the following:

- **No Overlapping.** Two flights cannot be assigned to the same gate at the same time. A buffer is added between flights that can be modified. One flight's arrival time cannot overlap another flight's departure time plus the time buffer in between flights at the same gate.
- **Non-Schengen flights.** All flights arriving or departing from a non-Schengen country need to be assigned to dual stands that have special non-Schengen gates which lead the passenger to border control. All dual stands can also receive flights arriving or departing from Schengen countries.
- **Canadian arrivals.** All flights arriving from Canada and other countries outside the EEA except for USA need to be assigned to gates 5, 6 or 25B. From those gates passengers can be lead to security screening.
- **Size restrictions.** Large aircrafts cannot park at every stand at the airport. They need to be assigned to those stands that can receive large aircrafts.

4.3 A mechanism for identifying problematic flights

In order for the model to be able to give a solution even though it cannot assign all flights to gates, a dummy variable, no-gate, was created. Then the model places all flights it has difficulties assigning to a gate to the no-gate variable. To make sure that the model only assigns flights that it cannot under any circumstances assign to a gate, to the no-gate variable a penalty, set by a weight factor, is given for each flight that is not assigned to a gate. The weight factor needs to be given the highest value so for example a flight is not assigned to no-gate instead of bus gate. Assigning flights to the no-gate variable should be the model's last resort for solving.

If the model cannot solve the GAP without assigning flights to the no-gate variable some flights need to be split as was described in Chapter 3.3.1. Splitting of flights can be necessary, especially when there is a long time between the flight's arrival and departure time in order for the flight not to hold a gate for a long amount of time. The flights that the model assigns to the no-gate variable are the flights the model has the most difficulties assigning to a gate. They are often the flights that have long time between arrival and departure and are therefore often the best candidates for splitting. A program was made in MATLAB which reads in the solution file from the model and splits all flights that haven't been assigned to a gate, see Appendix B.4. Then the program writes out a new data file so the binary optimization model can be run again. It can happen that all the flights that were assigned to the no-gate variable don't have a connection and can therefore not be split. Then the flight schedule would have to be viewed manually to see which flights are good candidates for splitting. The flights that are best suited for splitting are flights that have long time between arrival and departure and are not sensitive. A flow chart for this process can be seen in Figure 4-1.

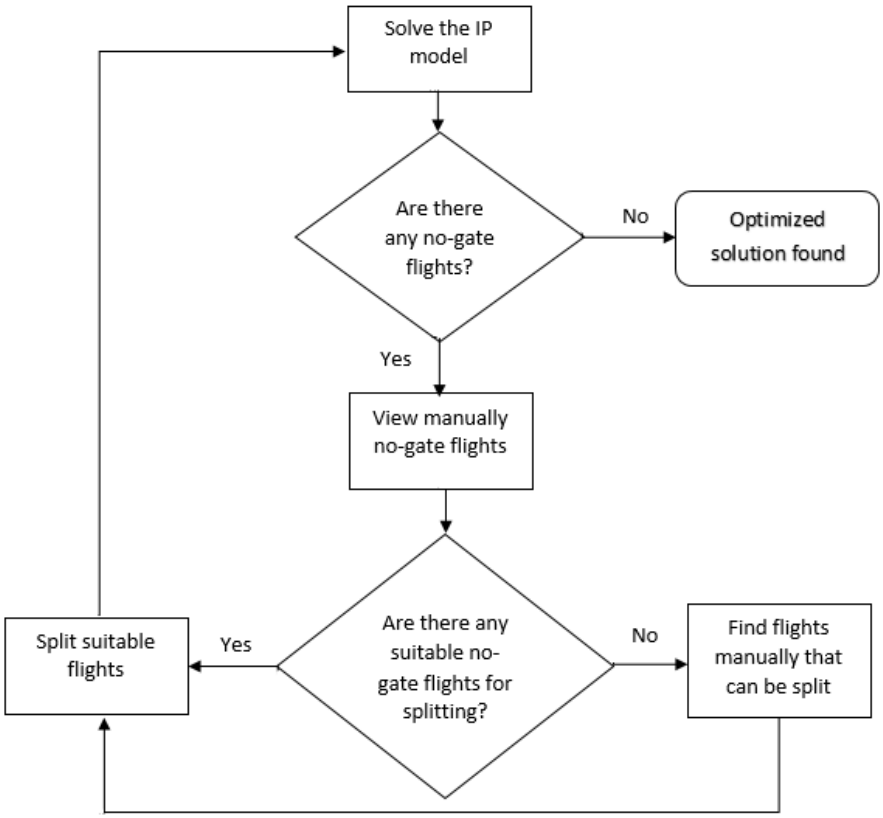

Figure 4-1: Flow chart for the usage of the no-gate dummy variable

4.4 Indices

The model uses two sets of indices to represent the flights and the gates. From those two sets, 13 subsets are formed to represent the different types of flights and gates.

Sets:

N: set of all flights, n = $|N|$.

M: set of all available gates at the airport, m =$|M|$.

$NSch \subseteq N$: subset of flights that are either arriving or departing to a non-Schengen country.

$C \subseteq N$: subset of flights that are arriving from countries outside EEA (except USA).

$F \subseteq N$: subset of flights that have heavy freight.

$L \subseteq N$: subset of flights with a large aircraft.

$S \subseteq N$: subset of flights that are sensitive.

$G1 \subseteq M$: subset of terminal gates that are dual, both Sch. and Non-Sch. flights can use.

$G2 \subseteq M$: subset of bus gates that are dual, both Sch. and Non-Sch. flights can use.

$G3 \subseteq M$: subset of bus gates that only flights arriving and departing to Sch. countries can use.

$G4 \subseteq M$: subset of dual gates that are walk-in, walk-out gates.

$G5 \subseteq M$: subset of gates that serve passengers arriving from Canada.

$G6 \subseteq M$: subset of gates that serve large aircrafts.

$G7 \subseteq M$: subset of gates that are suitable for flights with heavy freight.

$G8 \subseteq M$: subset of gates that are suitable for sensitive flights (all terminal gates).

Indices:

i: index for flights, i ∈ N.

j: index for gates, j ∈ M.

i1, i2: secondary indices for flights, used when checking overlaps of flights, i1 ∈ N, i2 ∈ N.

4.5 Data

The model uses the following data as input:

PA_i: total number of arriving passengers on the *i*th flight.

PD_i: total number of departing passengers on the *i*th flight.

$PTarr_i$: total number of transfer passengers on the arrival of the *i*th flight.

$PTdep_i$: total number of transfer passengers on the departure of the *i*th flight.

$paxTotal_i$: total number of all passenger types on the *i*th flight.

DA_{ij}: walking distance for the *i*th flight from the *j*th gate to baggage reclaim.

DD_{ij}: walking distance for the *i*th flight from check in hall area to the *j*th gate.

$DTarr_{ij}$: walking distance for the *i*th flight from the *j*th gate to transfer central point.

$DTdep_{ij}$: walking distance for the *i*th flight from the transfer central point to the *j*th gate.

TA_i: time of arrival of the *i*th flight.

TD_i: time of departure of the *i*th flight.

$\propto_1$: weight factor 1 to determine the importance of minimizing walking distances.

$\propto_2$: weight factor 2 to determine the importance of not using bus gates.

$\propto_3$: weight factor 3 to determine the importance of not using walk-in, walk-out gates.

$\propto_4$: weight factor 4 to determine the importance of assigning flights with heavy freight to G7.

$\propto_5$: weight factor 5 to determine the importance of assigning sensitive gates to G8.

$\propto_6$: weight factor 6 to determine the importance of not assigning flights to the no-gate dummy variable.

bigM: sufficiently large number.

B: buffer time between flights.

4.6 Decision variables

The main decision variables of the model are binary variables indicating whether a flight is assigned to a gate or to the dummy variable no-gate. Other variables are substitutes for the different parts of the objective function.

$$y_{i,j} = \begin{cases} 1, & if\ flight\ i\ is\ assigned\ to\ gate\ j;\ \forall i \in N, j \in M \\ 0, & otherwise \end{cases}$$

$$nogate_i = \begin{cases} 1, & if\ flight\ i\ is\ not\ assigned\ to\ any\ gate;\ \forall i \in N \\ 0, & otherwise \end{cases}$$

$Z_1: Total\ walking\ distance$

$Z_2: Total\ number\ of\ passengers\ assigned\ to\ bus\ gates$

$Z_3: Total\ number\ of\ passengers\ assinged\ to\ walk\_in, walk\_out\ gates$

$Z_4: Total\ number\ of\ heavy\ freight\ flights\ assinged\ to\ suitable\ gates$

$Z_5: Total\ number\ of\ passengers\ on\ sensitive\ flights\ assinged\ to\ suitable\ gates$

$Z_6: Total\ number\ of\ passengers\ assinged\ to\ the\ dummy\ variable\ nogate$

4.7 Binary optimization model

In this section the binary optimization model that was developed to imitate the real gate assignment problem at KEF is presented.

Objective function

$$\text{Min } \propto_1 * Z_1 + \propto_2 * Z_2 + \propto_3 * Z_3 + \propto_4 * Z_4 + \propto_5 * Z_5 + \propto_6 * Z_6 \quad (4.1)$$

Constraints

$$Z_1 = \sum_{i=1}^{n} \sum_{j=1}^{m} y_{i,j} * (PA_i * DA_{ij} + PD_i * DD_{ij} + PTarr_i * DTarr_{ij} + PTdep_i * DTdep_{ij}) \quad (4.2)$$

$$Z_2 = \sum_{i=1}^{n} \sum_{j \in G2 \cup G3} paxTotal_i * y_{i,j} \quad (4.3)$$

$$Z_3 = \sum_{i \in N} \sum_{j \in G4} paxTotal_i * y_{i,j} \quad (4.4)$$

$$Z_4 = \sum_{i \in F} \sum_{j \in G7} y_{i,j} \quad (4.5)$$

$$Z_5 = \sum_{i \in S} \sum_{j \in G8} paxTotal_i * y_{i,j} \quad (4.6)$$

$$Z_6 = \sum_{i=1}^{n} paxTotal_i * nogate_i \quad (4.7)$$

$$\sum_{j=1}^{m} y_{i,j} + nogate_i = 1, \forall i \in N \quad (4.8)$$

$$(TD_{i1} + B - TA_{i2}) * (TD_{i2} + B - TA_{i1}) \leq 0 + (1 - y_{i1,j}) * bigM + (1 - y_{i2,j}) * bigM,$$

$$i1 \neq i2, \forall i1 \in N, \forall i2 \in N \quad (4.9)$$

$$\sum_{j \in G1 \cup G2 \cup G4} y_{i,j} + nogate_i = 1, \forall i \in Nsch \quad (4.10)$$

$$\sum_{j \in G5} y_{i,j} = 1, \forall i \in C \quad (4.11)$$

$$\sum_{j \in G6} y_{i,j} = 1, \forall i \in L \quad (4.12)$$

$$y_{i,j} \in \{0,1\}, \forall i \in N, j \in M \quad (4.13)$$

$$nogate_i \in \{0,1\}, \forall i \in N \quad (4.14)$$

$$Z_1, Z_2, Z_3, Z_4, Z_5, Z_6 \geq 0 \quad (4.15)$$

The objective function (4.1) minimizes the six different parts it composes of, total walking distance, usage of bus and walk-in, walk-out gates, sensitive and heavy freight flights that are assigned to suitable gates and the number of flights that are assigned to the dummy variable no-gate. Each part is minimized by multiplying it to relevant weight factor which represents penalty. Assigning sensitive and heavy freight flights to suitable gates is desirable and therefore the weight factors for those parts is set to minus. Constraint (4.2) represents the total distance all passenger types have to walk. Constraint (4.3) represents the total number of passengers that are assigned to bus gates. Constraint (4.4) represents the total number of passengers that are assigned to walk-in, walk-out gates. Constraint (4.5) represents the total number of heavy freight flights that are assigned to suitable gates. Constraint (4.6) represents the total number of passengers on sensitive flights that are assigned to suitable gates. Constraint (4.7) represents the total number of passengers that are not assigned to a gate. Constraint (4.8) ensures that each flight is either assigned to a gate or to the no-gate variable. Constraint (4.9) ensures that two flights' schedules cannot overlap if they are assigned to the same gate by making sure that one flight's arrival time is always greater that another flight's departure time plus the time buffer. Therefore equations $(TD_{i1} + B - TA_{i2})$ and $(TD_{i2} + B - TA_{i1})$ are used in the left side of (4.9). If

one flight's arrival time, *TA*, is greater then another flight's departure time, *TD*, plus the time buffer, *B*, then the left side is in minus. If two flights, *i1* and *i2*, are assigned to the same gate, *j*, then the right side of the equation, $0 + (1 - y_{i1,j}) * bigM + (1 - y_{i2,j}) * bigM$, equals zero. Therefore one of the flights needs to have arrival time that is larger than the other flight's departure time plus the buffer time so that either of the equations in the left side results in minus, if the two flights are to be assigned to the same gate. If two flights' schedules are overlapping, the left side would be larger than zero and therefore the right side of the equation needs to be larger than zero as well. The only way for that to happen is if the two flights are not assigned to the same gate so the right side of the equation equals *bigM*, which is a sufficiently large number. Constraint (4.10) ensures that all non-Schengen flights are either assigned to dual gates or to the no-gate variable. Constraint (4.11) ensures that if a flight is arriving from a country outside EEA (except USA) it goes to a gate that serves those types of passengers and leads them to security screening. Constraint (4.12) ensures that size restrictions are fulfilled and places flights that have a large aircraft to a stand that can serve large aircrafts. Constraints (4.13) and (4.14) ensure that the binary conditions of the variables $y_{i,j}$ and $nogate_i$ are satisfied. The last constraint (4.15) ensures that variables Z_1 to Z_6 are equal or larger than zero.

4.8 Model assumptions and limitations

Flights that are classified as being sensitive are a large group at KEF. To make the model take into account different flights based on their sensitivity it would be possible to split them into groups. Then additional weight factors would be needed that give different amount of points for each group of sensitive flights. The more sensitive the flight is the more additional points would be given if the flight is assigned to a terminal gate.

There can be many aircrafts departing with heavy freight around the same time. The flights that have the most freight should have priority to the gates where it is easy to load and unload freight. The model doesn't take that into consideration, it only gives additional points if a flight that is classified as a heavy freight flight gets assigned to a gate where it is easy to unload and load freight. The model could make the flights with the most freight have priority by taking in the amount of freight as an input and multiplying it to the weight factor. Then the model would assign the flights with the most freight to the freight gates.

There are many other adjustments like those mentioned above that could be made. The model could also account for all the guidelines mentioned in Chapter 3.3.1. Due to the

number of inputs needed those matters were not included and the model was limited to only consider the main objectives and constraints regarding the gate assignments at KEF. Those matters and more can however be implemented to the model later without any technical difficulties.

Another limitation of the model is the use of the bus gate A. If a flight is parked at a remote stand and either the departure or the arrival of the flight is non-Schengen and the other one is Schengen it is possible for the flight to use two different bus gates. Gate A for the Schengen part of the flight and gate 8B or 8C for the non-Schengen part. The model cannot do this since if either the arrival or the departure of the flight is non-Schengen the flight is identified as a non-Schengen flight which can only go to dual gates and gate A is not a dual gate. This limitation is dealt with in some way by splitting flights. With the binary optimization model as it is there are only certain flights that are split but all flights parked at a remote stand can in reality use two bus gates, gate A for the Schengen part of the flight and another bus gate for the non-Schengen part.

4.9 Model verification

A small test case was created in order to check the functionality of the model. The case consists of seven flights that are to be allocated between six gates. All the problems attributes were to be tested by this small case. Because of the size of the test gate there was no need for using the no-gate dummy variable. All information about flights A-G is given in Tables 4-1 to 4-3.

Table 4-1: Test case - Number of passengers in each flight.

Flight	Arriving	Departing	Transfer-Arr.	Transfer-Dep.
A	50	60	10	10
B	70	20	5	5
C	60	50	15	15
D	55	55	15	15
E	65	60	10	10
F	70	65	20	10
G	90	70	10	10

Table 4-2: Test case - Flights time schedule.

Flight	Time of arrival	Time of departure
A	05:00	06:00
B	07:00	08:00
C	07:00	08:00
D	07:00	08:00
E	08:00	09:00
F	09:00	10:00
G	09:00	10:00

Table 4-3: Test case - Flights identifications.

Flight	*Non-Schengen*	*Arrival from Canada*	*Heavy freight*	*Large aircraft*	*Sensitive flight*
A					✓
B	✓	✓			
C	✓		✓	✓	
D	✓				
E	✓				✓
F	✓				
G					

The six gates were given different identifications in order to check all aspects of the model and see if the model assigned the flights to the appropriate gates. All information about gates 1 to 6 is given in Table 4-4 and Table 4-5.

Table 4-4: Test case – Walking distance for each type of passenger in meters.

Gate	Arrival	Departure	Transfer-Dep.	Transfer-Arr.
1	10	15	20	20
2	15	20	10	10
3	20	25	5	5
4	25	30	7	7
5	25	30	7	7
6	8	15	20	20

Table 4-5: Test case - Identifications of gates.

Gate	*G1-Terminal,dual*	*G2-Dual bus*	*G3- Sch. bus*	*G4-Walk-in*	*G5-Canada*	*G6-Large*	*G7-Freight*	*G8-Sensitive*
1						✓		✓
2	✓				✓			✓
3		✓					✓	
4				✓		✓		
5	✓					✓	✓	✓
6			✓					

4.9.1 Inputs

The necessary time buffer, B, between flights was set to be 30 minutes. The weight factors were set as can be seen in Table 4-6. Weight factor 6 was not used in this test case since it is only used with the no-gate dummy variable. For this particular test case these values were sufficient to validate the model's behavior. The weight factors will be set with more precise in Chapter 5.3.

Table 4-6: Test case - Weight factors.

Weight factor	Value
$\propto_1$	1
$\propto_2$	5
$\propto_3$	4
$\propto_4$	-3
$\propto_5$	-3

4.9.2 Results

This test case was modelled and solved with GLPK (GNU Linear Programming Kit) [26] which gave a feasible solution. Table 4-7 shows the visual results from the model. No flights are overlapping and the 30 minute buffer is fulfilled. Since flight B was arriving from Canada it goes to the gate that can receive arrivals from Canada and gate 2 was the only one with that identification.

Flight A is a Schengen flight and can therefore go to any gate and it is a sensitive flight. Gates 1, 2 and 5 are all terminal gates which are the best choice for sensitive flights. The model assigns flight A to gate 1 since that has the least walking distance of the three terminal gates. Gate 6 however has the lowest walking distance but because it is a bus gate

and flight A is a sensitive flight the model doesn't assign it there. There are four dual gates, gates 2 to 5, in the test case and there are 5 non-Schengen flights, B to F, that need to be assigned to those gates. The model assigns all of those flights to dual gates. Even though gates 1 and 6 would give flights B to F less total walking distance the model does not assign the flights at those gates which means that the model doesn't break the constraint that non-Schengen flights can only be served by dual gates.

Flight E is the only non-Schengen flight that is sensitive. Gates 2 and 5 are the only dual terminal gates but because of the 30 minute buffer time flight E cannot be assigned to gate 2 and therefore it is assigned to gate 5. The model behavior is therefore correct regarding the 30 minute buffer time and assigning sensitive flights to terminal gates. Flights B, C and D all have the same arrival time and are all assigned to different gates, flight F however arrives one hour after flights B, D and C have departed and can therefore use gates 2, 3 and 5. The model assigns it to gate 2 because it has the shortest walking distance. Flight G is a Schengen flight which could go to any gate except gates 2 and 5 because of the no overlapping constraint, (4.9), and the 30 minute buffer. The model assigns it to gate 1 because it is a terminal gate and has the lowest walking distance of the terminal gates.

Table 4-7: Test case – Results.

Gate	05:00	06:00	07:00	08:00	09:00	10:00
1	Flight A				Flight G	
2			Flight B		Flight F	
3			Flight D			
4			Flight C			
5				Flight E		
6						

4.10 Simulation model

For credibility of the binary optimization model, every solution is run through a simulation model called ARCport. The simulator is based on concepts of symbiotic systems and is used at major transport and retail facilities around the world. It uses Monte Carlo techniques only and no classical queuing theory or complex mathematics is involved [27].

ARCport is currently being used at KEF as a tool for managing growth, simulating use of existing facilities, and the design or planning of terminal facilities. ARCport provides a clear view of all operations at KEF. A model of the airport's terminal and runways has

been built in ARCport, Figure 4-2, where all the airport's operations can be viewed visually. Further visualization of the model can be seen in Figures 4-3 and 4-4.

When building the airport's model all passengers' flows were defined. All locations need to be defined, Schengen or non-Schengen, so the model knows which flow passengers should follow. ARCport would indicate an error if a passenger cannot follow his defined flow or if a flight is assigned to a gate it cannot be assigned to. It would also give an error message if two aircrafts are parked at the same stand at the same time. By running all gate assignments from the binary optimization model through ARCport it can be validated whether or not the solution is feasible for the airport's operations.

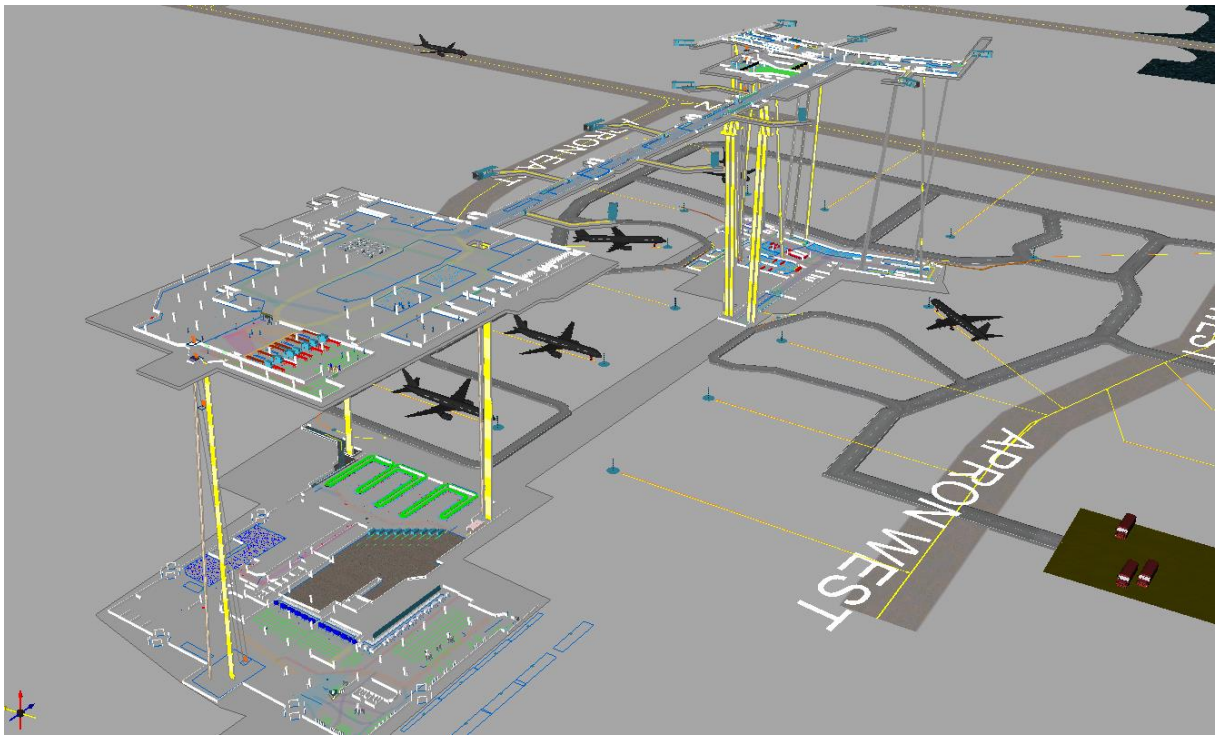

Figure 4-2: Model of KEF airport's terminal and runways in ARCport

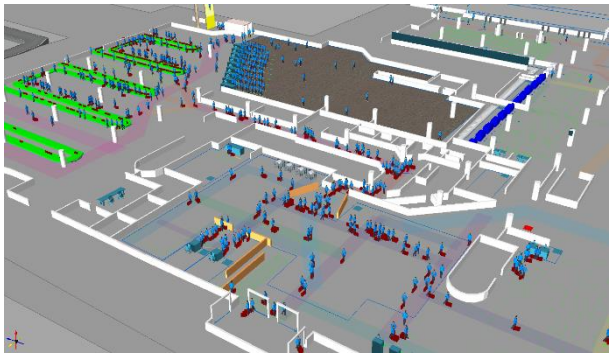

Figure 4-3: KEF airport's arrival hall in ARCport

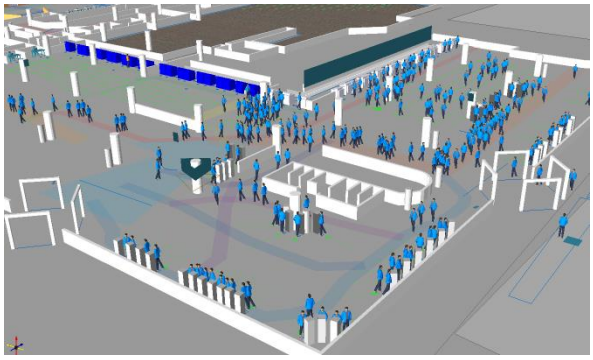

Figure 4-4: KEF airport's check-in hall in ARCport

5 DATASET

This chapter will describe the data that was needed to run the model described in Chapter 4. First all flight related data needed will be introduced and then all the gate related data. Followed by a description of the weight factors and buffer time. Data handling will also be discussed in this chapter along with problems with the data collection.

5.1 Flight data

Real flight data was received from Isavia. An Excel document which is connected to Airport Operational Database (AODB) was created in cooperation with the IT department at Isavia. The AODB is a database that holds information about all flights that go through Keflavik international airport along with all information related to the airport's operations in association with each flight. All flight data needed to run the binary optimization model can be found in the AODB but in order to access them, different documents needed to be opened. The Excel document however contains all the needed flight data one year back in time and half a year ahead in time in one document. In order to run the binary optimization model and to run the solutions through ARCport the following information about each flight was obtained from the Excel document:

- Flight number
- Aircraft's registration number associated with the flight
- The aircraft type associated with the flight
- Day of arrival or departure depending on if the flight was arriving or departing
- Name of the airline the flight belongs to
- Scheduled time of arrival/departure
- Real time of arrival/departure
- Total number of arriving/departing passengers
- Total number of transfer passengers (arriving/departing)
- Origin/destination
- Total freight
- Index indicating if the flight was a Schengen (1) or non-Schengen (0) flight
- Gate the flight was assigned to
- Stand the flight was assigned to

- Load factor of the aircraft associated with the flight
- The number of the arrival reclaim belt the flight was assigned to

In the data both the flights scheduled time of arrival/departure and the actual time of arrival/departure are given. It was decided to use the scheduled time since that is the known time if the model were to be used in real time.

Based on the flight's destination/origin it can be identified if the flight is a sensitive flight or not. Flights that are arriving or departing to the following locations are sensitive flights;

- London Heathrow (LHR)
- London Gatwick (LGW)
- Helsinki (HEL)
- Munich (MUC)
- Frankfurt (FRA)
- Paris (CDG)
- New York (JFK)
- Washington (IAD)
- Baltimore (BWI)
- Minneapolis (MSP)
- Toronto (YYZ)

Even though the list of sensitive flights used for the data of this thesis is not a complete list of all sensitive flights at KEF it is a good portion and includes all of the main sensitive flights.

If the total freight on the flight is above 3000 kg the flight is identified as a heavy freight flight. The arrival and departure parts of the flight can have different amount of freight and often it is the departure flight that has the heaviest freight since the freight is mostly fish that is being exported from Iceland.

5.2 Gate data

In order to run the model the following information was needed about each gate at Keflavik international airport:

- The walking distances to each gate, including the distance for an arriving passenger, a departing passenger and a transferring passenger.
- The gate's identification, if it's a dual gate, a bus gate, a walk-in, walk-out gate and all other possible identifications.

5.2.1 Walking distances

Walking distances for each passenger type, arriving, departing and transfer was measured manually with AutoCAD [28] drawings of the airport. Figure 5-1 shows the current layout of all gates at KEF. As the figure shows each dual stand at the airport has two gates, like stand 5 has two gates, 5 and 25. Gate 5 is for Schengen passengers and gate 25 is for non-Schengen passengers. All Schengen gates are numbered in green and non-Schengen gates are numbered in red in Figure 5-1. Schengen and non-Schengen passengers don't walk the same distance even though in both cases their aircraft is parked at the same stand. Non-Schengen passengers have to go through border control and therefore walk longer distances.

Isavia does not have access to the data on what gate each transfer passenger is going to, therefore two central points were created and measurements on walking distance for transfer passengers were based on those points. Most of the transfer traffic is traveling from a non-Schengen country to a Schengen country or the other way around. A very small portion of transfer passengers travel from a non-Schengen country to another non-Schengen country, from North-America to Britain and the other way around. It is extremely unlikely that transfer passengers travel from a Schengen country to another Schengen country. Therefore two central points were created, one for Schengen and one for non-Schengen. The Schengen transfer central point is on the second floor of KEF south building and can be viewed on Figure 5-1, the non-Schengen transfer central point is on the exact same location but on the floor below, the first floor. The walking distances for transfer passengers were measured based on the following flow for each transfer passenger type:

- Transfer departure – Schengen passenger:
    - Starts at the non-Schengen central point
    - Goes through immigration
    - Ends at his Schengen gate
- Transfer arrival –Schengen passenger:
    - Starts at his Schengen gate

- Goes through emigration
- Ends at the non-Schengen central point

- Transfer departure – Non-Schengen passenger:
  - Starts at the Schengen central point
  - Goes through emigration
  - Ends at his non-Schengen gate
- Transfer arrival – Non-Schengen passenger:
  - Starts at his non-Schengen gate
  - Goes through immigration
  - Ends at the Schengen central point

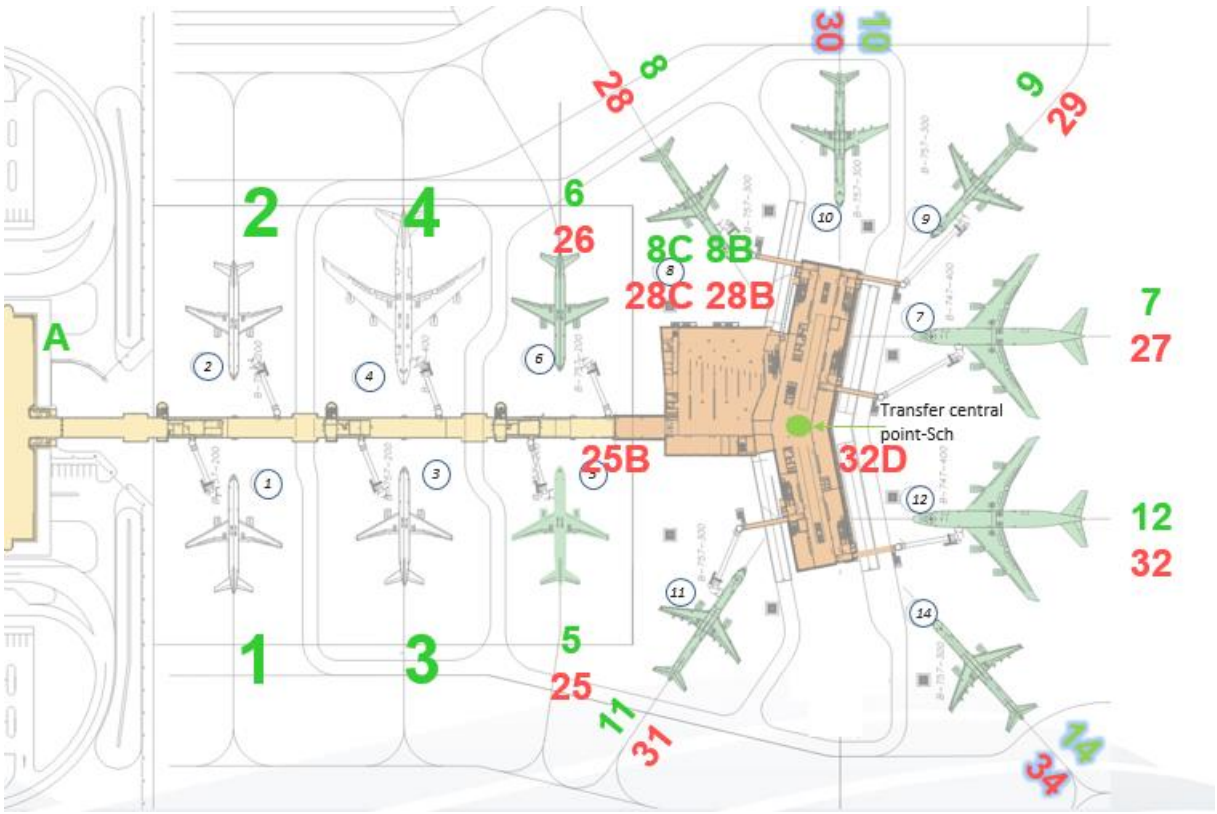

Figure 5-1: All gates at KEF and Schengen central transfer point [24].

For arriving passengers, distances were measured from each gate to the baggage reclaim area (DA). For departing passengers, distances were measured from the check-in hall area to each gate (DD). For transfer arriving passengers, distances were measured from the gate to the central point (DT_arr). For transfer departing passengers, distances were measured

from the central point to the gate (DT_dep). All measurements for each gate can be viewed in Table 5-1.

Table 5-1: Measured walking distances for each gate at KEF in meters. Colored lines mark the distances for non-Schengen passengers.

Gates	DA	DD	DT_arr	DT_dep
1	98,42	528,60	382,94	343,74
2	132,40	538,60	349,06	308,40
3	165,96	573,95	315,85	295,96
4	196,88	603,29	284,76	274,03
5	233,03	641,57	248,54	230,33
25	501,57	916,06	258,41	288,37
6	264,00	670,76	217,74	214,37
26	495,10	849,20	251,93	280,45
7	364,23	813,10	242,80	222,64
27	464,15	910,26	220,98	282,57
8	412,50	816,23	291,07	225,77
28	493,82	940,96	250,66	313,27
9	409,33	814,82	287,90	224,35
29	476,35	921,14	233,18	293,45
10	406,83	814,00	285,40	223,54
30	459,93	906,65	216,76	278,96
11	382,93	789,91	261,51	199,45
31	449,47	921,37	206,30	293,68
12	401,65	814,40	280,23	223,93
32	465,36	932,34	222,20	304,65
14	471,75	878,93	350,33	288,47
34	502,51	949,12	259,35	321,43
8B	416,45	827,86	295,02	237,39
28B	440,08	886,94	196,92	259,25
8C	438,91	850,32	317,48	259,85
28C	456,46	903,12	213,29	275,43
A	177,70	553,37	496,40	457,19

For the walk-in, walk-out gates 10 and 14 (30, 34) the distances from the gates to the aircraft were included in the measured walking distance. For the bus gates 8B, 8C (28B, 28C) and A the distances from the gates to the aircraft were not included in the measured walking distances because the passengers don't walk those distances, they are transferred with a bus. Instead those distances need to be counted for with the penalty for using bus gates which is set by a weight factor in the model.

Gates 25B and 32D are only arrival gates, not departure gates. Therefore they cannot be implemented to the model since the model looks at one flight with one arrival and one departure. The model would therefore not be able to assign a flight to those gates since they do not have departure gates. Therefore distances for those gates were not measured. If a flight is assigned to those gates in the original gate assignment it needs to be changed to gate 8B or 8C. Because 8B has the shorter walking distance, that will be used for arrival flights that were assigned to either bus gate 25B or 32D. Then when comparing the total walking distances of the optimized gate assignment to the original it will be on equal ground. In reality the arrival passengers would be transferred with buses to gates 25B if they need to go through security screening or 32D because it has the shortest walking distance of the non-Schengen bus gates, for both the optimized gate assignment and the original.

5.2.2 Gate identification

Information about each gate's identifications was received from Isavia. The gates will be listed after the subset of gates needed for the binary optimization model introduced in Chapter 4.

- Terminal dual gates (G1): Gates 5/25, 6/26, 7/27, 8/28, 9/29, 11/31 and 12/32
- Dual bus gates (G2): Gates 8B/28B and 8C/28C
- Only Schengen bus gates (G3): Gate A
- Dual walk-in, walk-out gates (G4): Gates 10/30 and 14/34
- Gates that lead passengers arriving from Canada or other country outside the EEA (except for USA) to security screening (G5): Gates 5/25, 6/26, 8B/28B and 8C/28C
- Gates that serve large aircrafts (G6): Gates 4/24, 7/27 and 12/32
- Gates that serve well flights with much freight (G7): Gates 7/27, 9/29 and 12/32
- Gates that serve well sensitive flights, all terminal gates (G8): Gates 1, 2, 3, 4, 5/25, 6/26, 7/27, 8/28, 9/29, 11/31, 12/32.

Gates 8B/28B and 8C/28C replace 25B as a gate that can lead passengers through additional security screening (G5). Even though flights would be assigned to those gates the arrival passengers would be transferred to bus gate 25B in reality, to enter security screening.

5.3 Weight factors and buffer time

The binary optimization model uses six different weight factors, $\propto_1$-$\propto_6$, to represent a penalty or reward for the different parts of the objective function. The interpretation of the weight factors can be difficult due to the different meaning of the objectives. Therefore it is important that the weight factors are adjusted to represent the same measurement that can easily be understood.

It was decided to make the weight factors represent cost but then it can be viewed as placing a flight at a certain gate costs some amount. Which it does in reality since, for example, having the most passengers travel the shortest distance will give the airport better passenger satisfaction and passengers have more time to spend in shopping facilities. Passenger's satisfaction also increases if they don't have to walk outside from their aircraft to the terminal or to a bus.

In order to measure walking distances as cost, an average cost per meter needs to be estimated. According to a survey done in 1999 on pedestrian walking speed in airport terminals, the average walking speed in airport terminals is 80.5 meters per minute [29]. According to Statistics Iceland the average salaries in Iceland in the year 2014 were 415,000 ISK a month for an average 43.1 hours per week [30]. Therefore it can be said that an hour is worth 2,407 ISK and in one hour passengers walk on average 4830 meters. Each meter can therefore be estimated at a cost of 0.50 ISK. The weight factor $\propto_1$ was therefore set to 0.50 ISK.

If a flight is assigned to a bus gate additional cost needs to be added. The weight factor $\propto_2$ was set to 100 ISK. Which means that if a flight is assigned to a bus gate an additional cost of 100 ISK is added per passenger of that flight, to the walking distance cost for that gate. Additional cost of 40 ISK is added with weight factor $\propto_3$ if a flight is assigned to a walk-in, walk-out gate. That cost is less than if a bus gate is used since it is considered a better alternative to use the walk-in, walk-out gates rather than the bus gates.

If the model succeeds at assigning sensitive flights to terminal gates a reward is given, therefore weight factor $\propto_5$ was set to minus 300 ISK. It is then multiplied to the total number of passengers on the flight so that the sensitive flight with the most passengers is in priority. Placing a sensitive flight to terminal gates needs to lower the total cost more than if a heavy freight flight is placed at the desired gates. This is in consideration to the

airport's needs to the gate assignment since sensitive flights give the airport more revenues than heavy freight flights. If there are two sensitive flights, one with one more passenger and the other one with heavy freight, the sensitive flight with the one additional passenger has priority. Therefore weight factor $\propto_4$ needed to be less than $\propto_5$ and was set to minus 299 ISK. The reward for assigning heavy freight flights at suitable gates is not given per passenger, only per flight.

Weight factor $\propto_6$ which represents the cost of assigning a flight to the no-gate variable was set to 10.000 ISK per passenger.

With these weight factors the priority of the objective functions should be the following:

1. Avoid using a no-gate variable if possible
2. Place sensitive flights at terminal gates
3. Minimize the total walking distance for all passenger types
4. Minimize the use of bus gates
5. Minimize the use of walk-in, walk-out gates
6. Place heavy freight flights at suitable gates.

The buffer time, which is used for resolving minor delays that often occur in real-time operations, was set to 30 minutes as it is today at KEF. The buffer time is added between two continuous flights that are assigned to the same gate.

5.4 Data handling and proposed inputs

To create inputs for the binary optimization model and the simulation model, ARCport, programs were created in MATLAB. The programs put the data from the Excel document in a desirable format for the models and exported them as text files. The two models read the data in a different format, therefore three programs were made, one which puts the original flight schedule on the format for ARCport to read, one that puts the data on the format needed for the binary optimization model and then one that reads in the solution file from the binary optimization model and changes the stands and gates of the original flight schedule on the format ARCport can read. All the programs can be viewed in Appendices B.1, B.2 and B.3.

5.4.1 Program for adapting data for ARCport

The following steps were made to adapt the data to the format needed for ARCport:

i. Choose only the data for the date to be examined.
ii. Choose only passenger related flights (types J, C, G and S).
iii. Divide arrival flights from departure flights. All data regarding arrival need to be in separate columns from the data involved with the departure flights.
iv. Change the names of the stands and gates to what they are named in ARCport in order for ARCport to be able to read them.

5.4.2 Program for adapting data for binary optimization model

The following steps were made to adapt the data to the format needed for the binary optimization model:

i. Choose only the data for the date to be examined.
ii. Choose only passenger related flights (types J, C, G and S).
iii. Change the time format HH:MM to number of minutes since the model cannot read a conventional time format. It is also convenient when constructing the no-overlap constraint (4.9), which makes sure that flights do not overlap in the same gate.
iv. For the model to know for how long a flight will hold a gate the arrival of a flight and the departure of the flight needs to be known. This information is not available in the data and therefore flights needed to be connected by the aircraft's registration number. If the registration number of an arrival flight and a departure flight are the same that is a connected flight. The time of the departure flight must be after the arrival time.
v. If no connection is found between flights, dummy arrival or dummy departure time needs to be created. The airport can demand that an aircraft has been towed away from a gate 40 minutes after its arrival. Because of the 30 minute buffer between flights in the binary optimization model, the dummy time was set to 10 minutes. If the flight however arrives after 22:00, the dummy time was set to 30 minutes, then there are 60 minutes between flights. This was done because it's likely that the flight is leaving the following morning and it should be able to stay at the gate if possible. Otherwise the model assigns too many flights to the same gate while other gates are free which leads to unnecessary towing procedures.
vi. Detect all flights identification and place in relevant sets.
vii. Create a distance matrix where distance for each flight for each gate is indicated depending on if the flight is Schengen or non-Schengen.

5.4.3 Program for adding solution from optimization model to format for ARCport

The program reads in the solution file from the binary optimization model and changes the gate numbers of the original flight schedule on the format for ARCport. Ultimately doing the same steps as in Chapter 5.4.1.

5.5 Problems with the data and the data collection

Some minor discrepancies were in the data. The names for some gates were missing and had the name of the stands instead. This problem only occurred for bus gates, most often arrival flights, then the name of the remote stand came instead of the name of the bus gate. For those instances the bus gate was changed to gate 8B. For the arrival flights it does not matter if the flights are overlapping because it takes a very short amount of time to offload a bus. Therefore buses for several aircrafts parked at different stands can drop off passengers at the same arrival bus gate. This does not apply for departure flights since it takes at least 30 minutes to board all passengers. Therefore the departure flights that were missing a gate needed to be looked at manually to make sure that no flights were overlapping, that however happened rarely.

Another minor problem was that even though flight type G is most often passenger related flight it sometimes occurs that a flight of that flight type has no passengers. Since this thesis is limited to passenger related flights and flights with no passengers do not need a terminal related gate, those flights were erased from the flight schedule.

When running the original gate assignments through ARCport one problem came up. By using the scheduled time as the time of arrival and departure, some flights were overlapping with the original flight schedule. This didn't occur in real time, some flights are delayed and can therefore be assigned to a different gate than if they were on their scheduled time. ARCport will only be used to check the credibility of the optimized gate assignment and to compare the walking distances. Therefore this problem was solved by changing the arrival or departure time so that no flights are overlapping for the original flight schedule. Then the actual walking distances for all passengers could be measured.

6 RESULTS

The proposed binary optimization model in Chapter 4 for optimizing the GAP at KEF was used with the data presented in Chapter 5. By running the model with real data it can be seen if the model can give a feasible solution that meets the real demand of the airport´s operations. One week of high season (summer period) and one week of low season (winter period) were run through the model and compared to the original gate assignment. As mentioned in Chapter 3 the gate assignment can only be done the day before, or on the day of operation because the connections between arrival and departure flights are not known until that time period. Therefore the data was run through one day at a time as it would be done in real time.

The model was solved using Gurobi 6.0.2 solver. Before solving the model for each day it was written to a MPS file. To test the credibility of the proposed solutions from the model all results were run through ARCport simulator.

This chapter will present all results for both the low season period and the high season period. For each period the results will be compared to the original gate assignment. The results will be based on the model's objective functions, minimizing use of bus gates and walk-in, walk-out gates, assigning sensitive and heavy freight flights to best suitable gates and minimizing total passenger walking distances. The number of splits conducted were also compared.

When counting the use of bus gates, each arrival and departure of a flight is counted separately. That was done so that flights that don't have a connection and are assigned to bus gates, could be counted as one. The same applies to the walk-in, walk-out gates and for the sensitive and heavy freight flights.

The walking distances were measured in two different ways, one with the manually measured walking distances and one where ARCport was used to measure the total walking distances. A program was created to measure the manually measured walking distance, see Appendix B.5. The program detects if the flight is Schengen or non-Schengen and calculates the total walking distance from a distance matrix. It was decided to use both methods for comparison since the manually measured distances represent the distances if passengers all walk the same line that was measured, while in ARCport passengers walk different routes to get to the same location. Passengers go to different

shopping facilities and use different booths in immigration for example. It was however not possible to measure the walking distances for transfer passengers in ARCport because the program computes randomly which transfer passengers are going where so the count for transfer passengers where never the same when the original gate assignment was compared to the optimized gate assignment. ARCport was however used to view the difference between the walking distance for Schengen passengers and non-Schengen passengers.

6.1 Low season period

The low season for KEF is during the winter from October through April. The week that was picked to represent the low season was picked at random and was the first week of January 2015, the first to the seventh. Number of flights on each day were from 32 to 51.

Four of the days were optimized in first iteration without the use of the dummy variable no-gate. For two days the model placed two flights at the no-gate variable with all the flights being flights with long time period between arrival and departure. Therefore it was easy to see why the model could not assign them to gates since they would hold the gate for a very long period of time. The no-gate flights were split as was described in Chapter 4.3 and then the model could find a feasible solution. For one day the model could not find gates for three flights. One of the flight had no connection so it was not a candidate for splitting, one of the flight had only 40 minutes between arrival and departure and the third flight had about 10 hours between arrival and departure. Instead of splitting the two flights that were suitable for splitting it was tested to split only the one with 10 hours between arrival and departure. With only that one flight split the model could find an optimized solution and placed all flights to gates. For each of the three days where flights needed to be split for the binary optimization model to find a feasible solution without the use of the no-gate dummy variable, the original gate assignment had split as many flights or more. In total the number of splits for the whole week were 38% less than the total number of splits in the original gate assignment. Which means that towing procedures were decreased by 38%.

It took the model 0.05 to 0.12 seconds to solve each of the seven days with number of binary variables ranking from 561 to 867. Each day was run through ARCport which gave no error messages, indicating that all optimized gate assignments were feasible.

6.1.1 Use of bus gates and walk-in, walk-out gates

Figure 6-1 shows the difference between the optimized gate assignment from the binary optimization model and the original gate assignment of using bus gates. The original gate assignment used bus gates in every day of the tested week. The optimized gate assignment however only used bus gates in two of the seven days. Total usage of bus gates for the week was decreased by 73% from the original gate assignment.

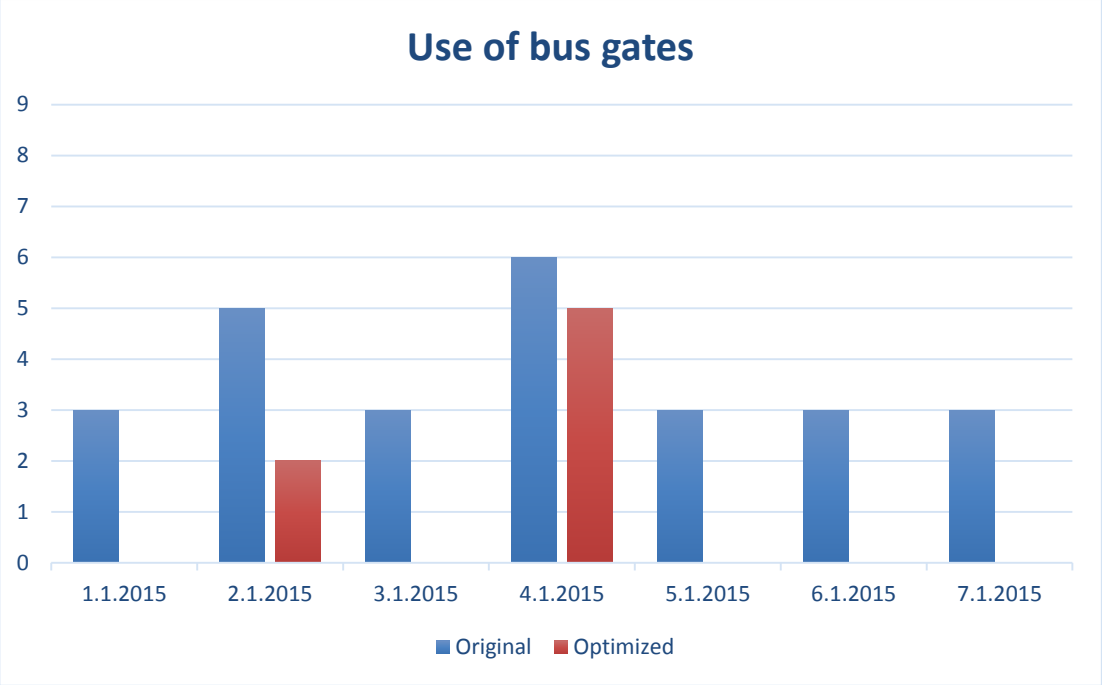

Figure 6-1: Comparison of the usage of bus gates-Low season.

Figure 6-2 shows the difference between the optimized gate assignment and the original of using walk-in, walk-out gates. Despite of the high difference in usage of bus gates the model still decreases the total usage of walk-in, walk-out gates by 3% from the original gate assignment.

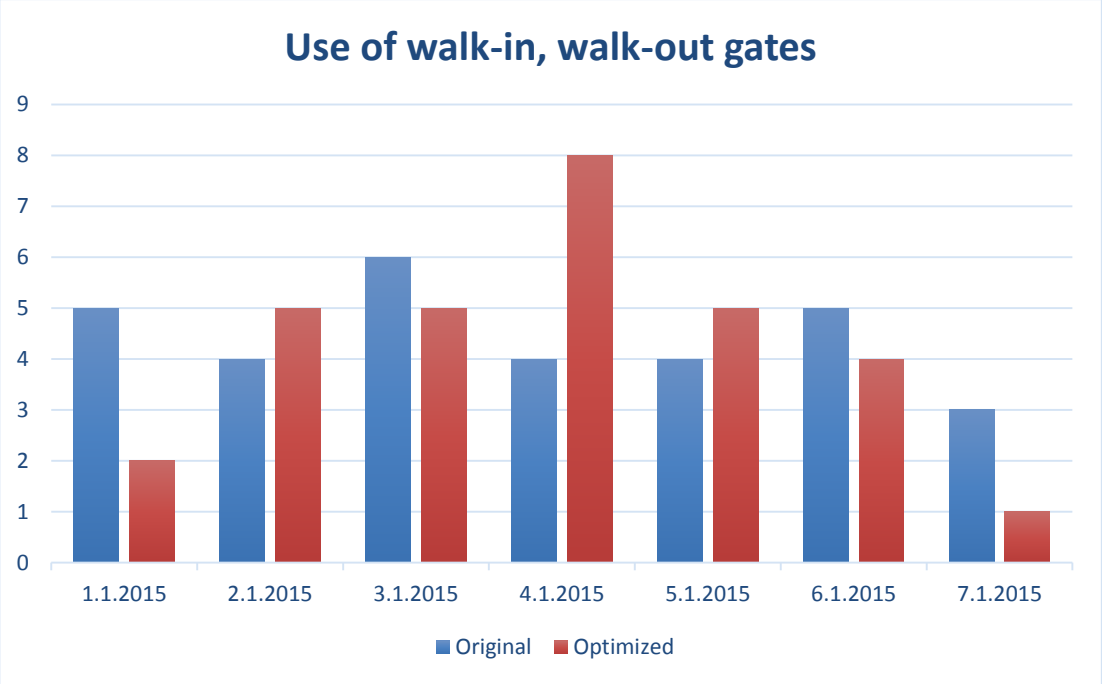

Figure 6-2: Comparison of the usage of walk-in, walk-out gates-Low season.

6.1.2 Sensitive and heavy freight flights

Figure 6-3 shows the difference between the optimized gate assignment and the original gate assignment of assigning sensitive flights to the best suitable gates. For each tested day the optimized gate assignment placed all sensitive flights at terminal gates while the original gate assignment placed 67% to 92% of sensitive flights at terminal gates per day. In total the optimized gate assignment placed 20% more sensitive flights at terminal gates.

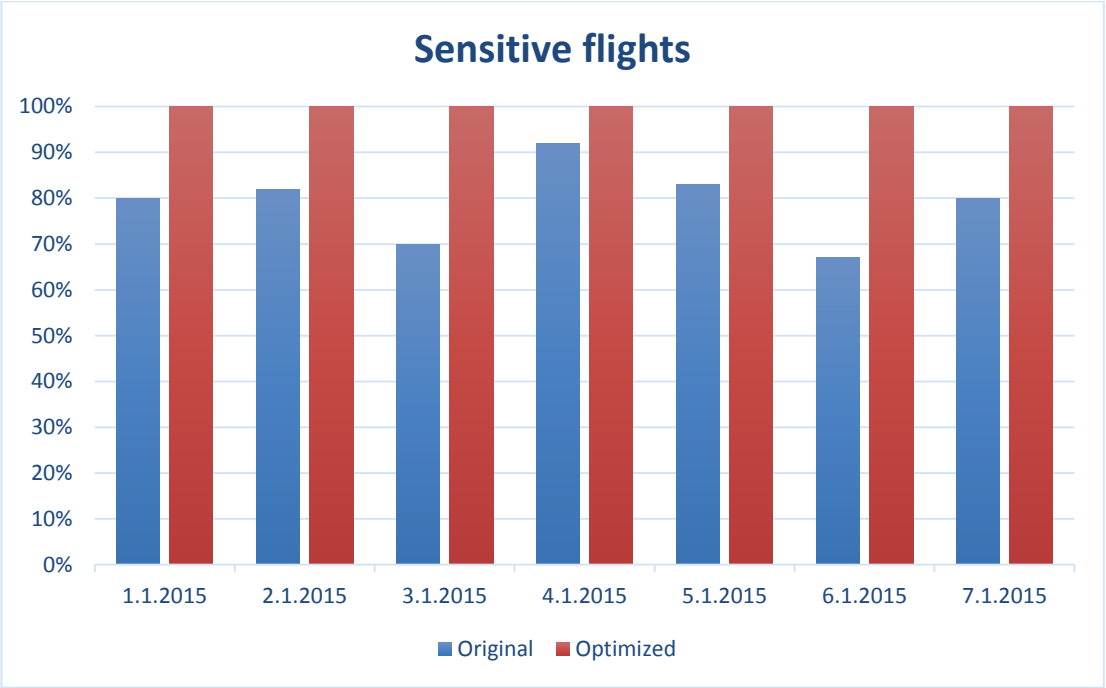

Figure 6-3: Comparison of placing sensitive at best suitable gates-Low season.

Figure 6-4 shows the difference between the optimized gate assignment and the original gate assignment of assigning flights with heavy freight to the best suitable gates. The last priority of the model was to assign heavy freight flights to appropriate gates but still it assigned in total 5% more heavy freight flights than the original gate assignment.

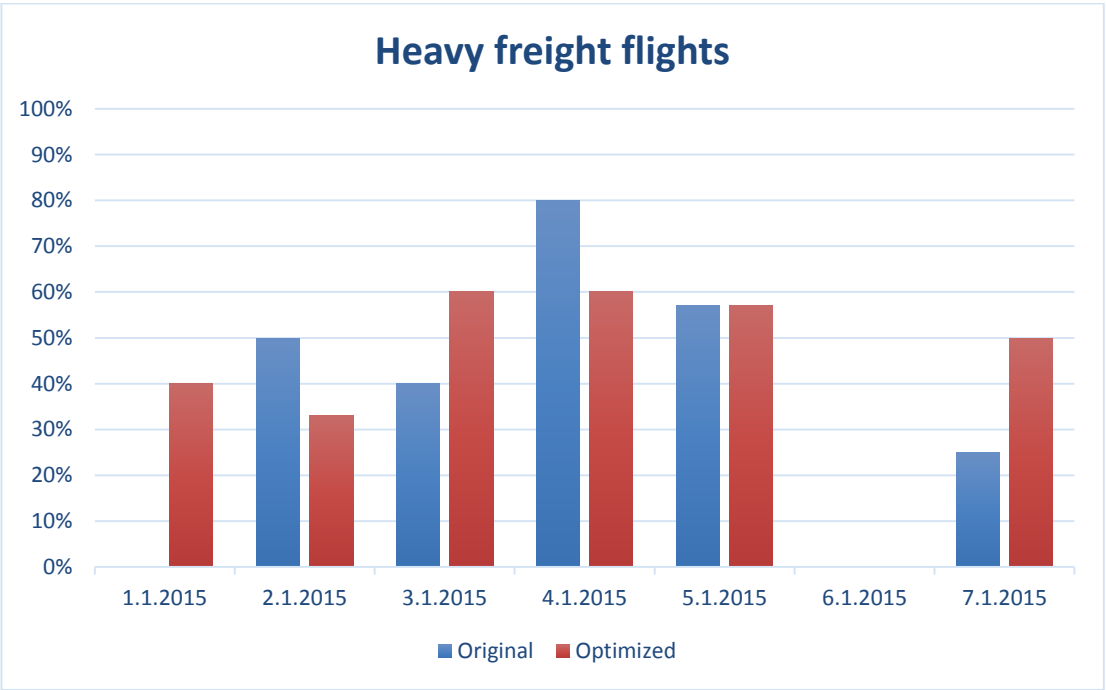

Figure 6-4: Comparison of placing heavy freight flights at best suitable gates-Low season.

6.1.3 Walking distances

For each day of the examined week the optimized gate assignment decreased the total walking distance by 0.9% to 8.3% from the original assignment, based on the manually measured walking distances. The difference for each type of passengers can be seen in Table 6-1. In total passengers walked 4% less, or in total 1380 km. With the total number of passengers for the whole week being 65,433, each passenger walked on average 21.09 meters less with the optimized gate assignment than the original. More detailed listing of the results can be viewed in Appendix C.

Table 6-1: Comparison of total walking distances (km) based on manually measured distances – Low season

	Original	*Optimized*	*Difference*
Arrival	6683	6058	-9%
Departure	21097	20424	-3%
Transfer-arr	2446	2383	-3%
Transfer-dep	2607	2588	-1%
Total	32833	31453	-4%

The walking distances for arrival and departure passengers were also measured with ARCport. Walking distances for Schengen passengers and non-Schengen passengers were measured separately and the difference can be seen in Table 6-2. As mentioned earlier it was not possible to measure the walking distances for the transfer passengers with ARCport. Similar results were received from ARCport as with the manually measured distances. For each day the optimized gate assignment decreased the total walking distances by -2.6% to 8.2% from the original assignment. In total passengers walked 3% less, or in total 799 km. On average each passenger walked 12.2 meters less with the optimized gate assignment than the original, according to ARCport. As can be seen in Table 6-2 the optimized gate assignments succeeds better at lowering passengers' walking distances for Schengen passengers than non-Schengen. More detailed listing of the results can be viewed in Appendix C.

Table 6-2: Comparison of total walking distances (km) measured in ARCport – Low season

	Original	*Optimized*	*Difference*
Arr. - Non-Sch.	6548	6503	-1%
Arr. - Sch.	3195	2680	-16%
Dep. - Non-Sch.	13978	13957	-0,1%
Dep. - Sch.	7901	7682	-3%
Total	31622	30823	-3%

6.2 High season period

The high season for KEF is from May through September. The busiest months in the year 2014 were July and August. Therefore the first week of July in the year 2014 was picked to represent a high season period for KEF with number of flights per day being 64 to 76. On average 56% more flights per day than in the low season period.

For each of the seven days the model assigned 2 to 8 flights to the no-gate variable. For July 3rd, it was enough to split only the no-gate flights for the model to find an optimal solution. For July 5th, no-gate flights needed to be split three times before model could find an optimal solution. For the rest of the days the flight schedule needed to be looked at manually to find flights suitable for splitting. One time it happened that after looking at the flight schedule manually and picking flights to split the model could still not find an optimal solution so the flight schedule was looked at again and one other flight needed to be split as well, then the model found a solution. The total number of splits were still decreased by 21% from the original gate assignment.

It took the model 0.18 to 0.33 seconds to solve each of the seven days with number of binary variables ranking from 958 to 1171. Optimized gate assignments for each day were run through ARCport which gave no error messages, indicating that they were all feasible gate assignments.

6.2.1 Use of bus gates and walk-in, walk-out gates

Figure 6-5 shows the difference between the optimized gate assignment from the binary optimization model and the original gate assignment of using bus gates during the high season period. In the high season period the binary optimization model uses bus gates for every day of the tested week. Terminal gates are a highly limited resource at KEF and when the traffic is heavy around the airport the use of bus gates cannot be avoided. The model however decreases the usage from the original gate assignment and uses in total 35% less of bus gates.

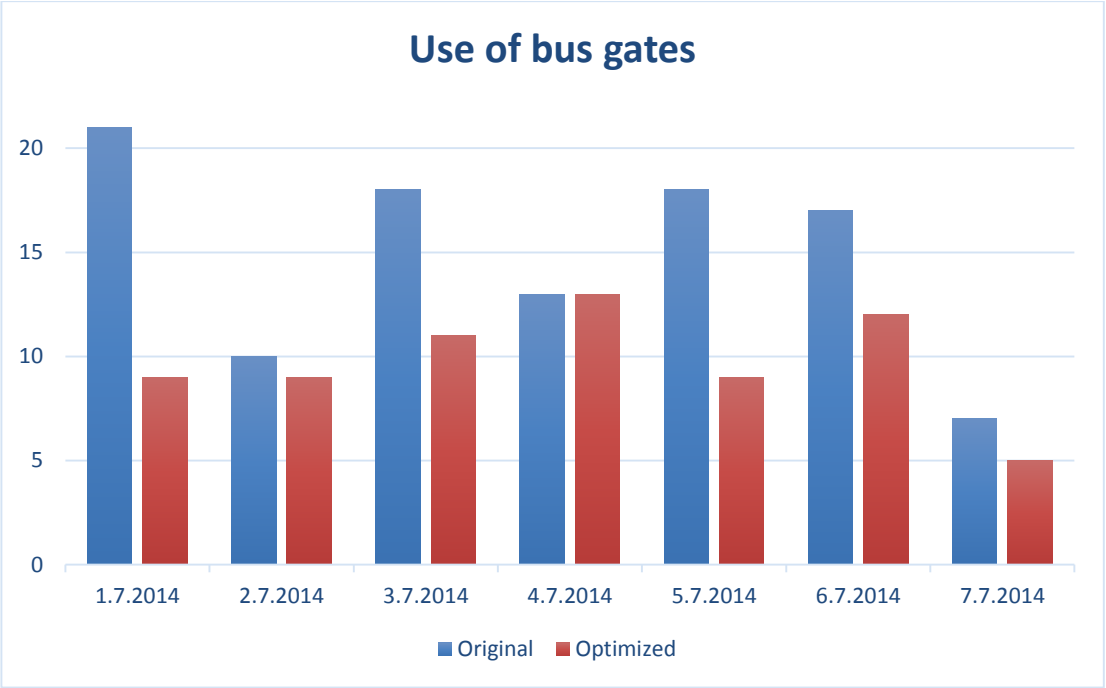

Figure 6-5: Comparison of the usage of bus gates-High season.

Figure 6-6 shows the difference between the optimized gate assignment and the original gate assignment of using walk-in, walk-out gates during the high season. Both used walk-in, walk-out gates every tested day. The binary optimization model however used in total 27% less of walk-in, walk-out gates.

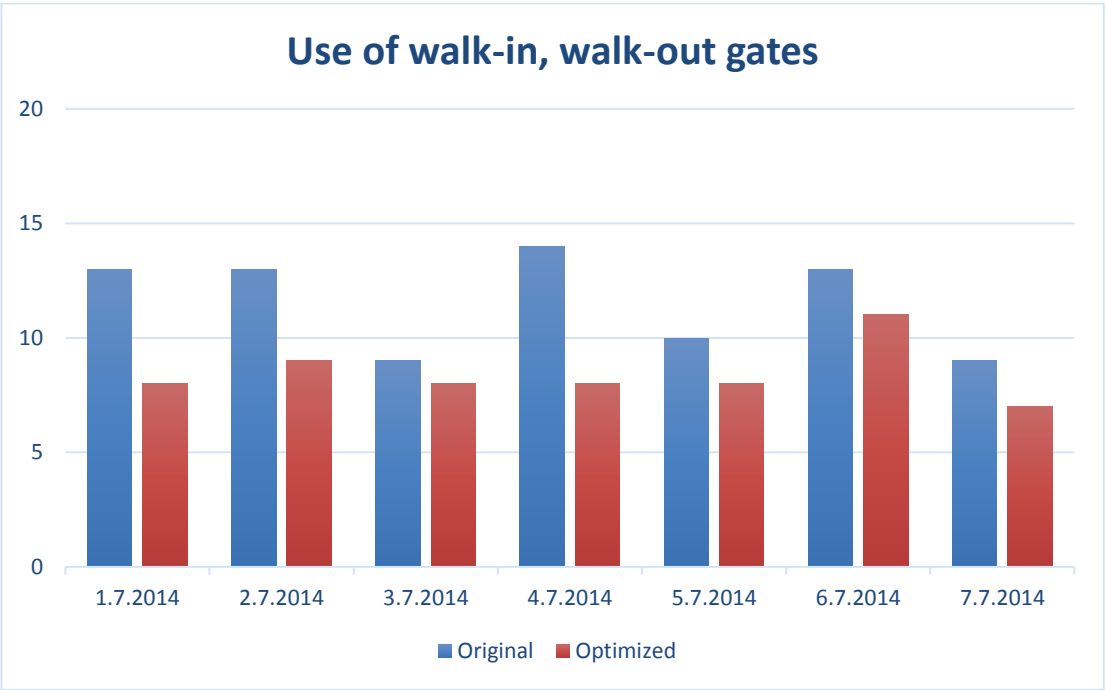

Figure 6-6: Comparison of the usage of walk-in, walk-out gates-High season.

6.2.2 Sensitive and heavy freight flights

Figure 6-7 shows the difference between the optimized gate assignment and the original gate assignment of assigning sensitive flights to the best suitable gates, the terminal gates. For every day of the tested week except one the model assigned all sensitive flights at terminal gates. For the one day that the model didn't assign all sensitive flights at terminal gates it assigned all but one. The original gate assignment assigned 67% to 82% of all sensitive flights to terminal gates. The optimized gate assignment in total assigned 24% more sensitive flights to terminal gates than the original gate assignment.

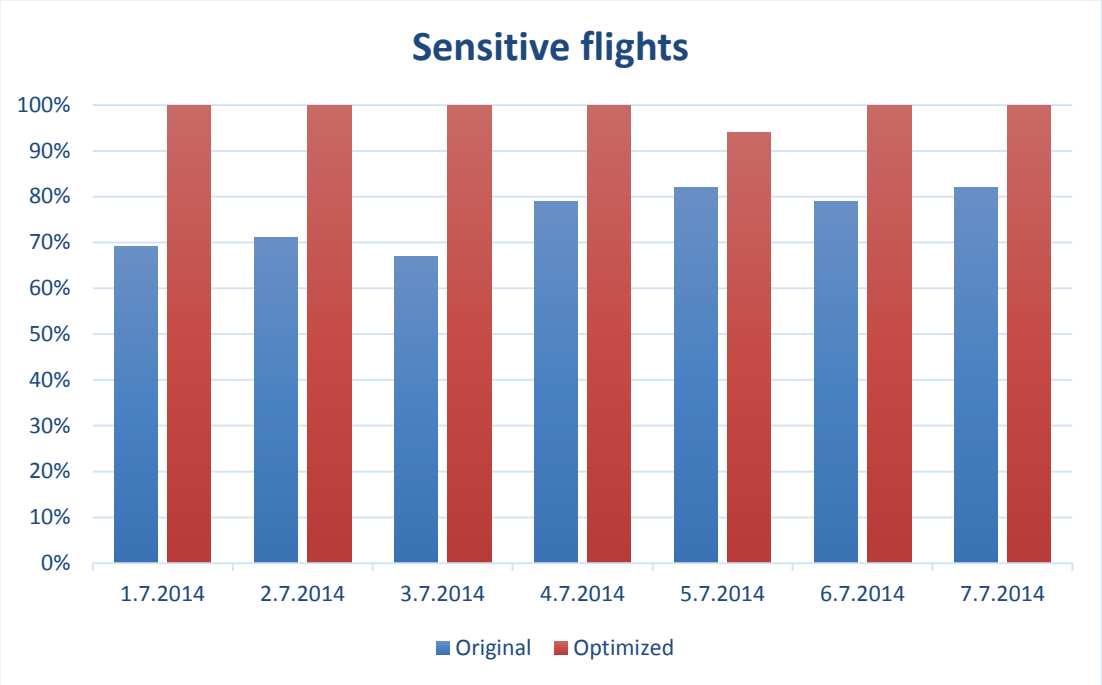

Figure 6-7: Comparison of placing sensitive flights at terminal gates-High season.

Figure 6-8 shows the difference between the optimized gate assignment and the original gate assignment of assigning flights with heavy freight to the best suitable gates. Even though assigning heavy freight flights to best suitable gates is the model's last priority it assigned in total 21% more heavy freight flights than the original gate assignment.

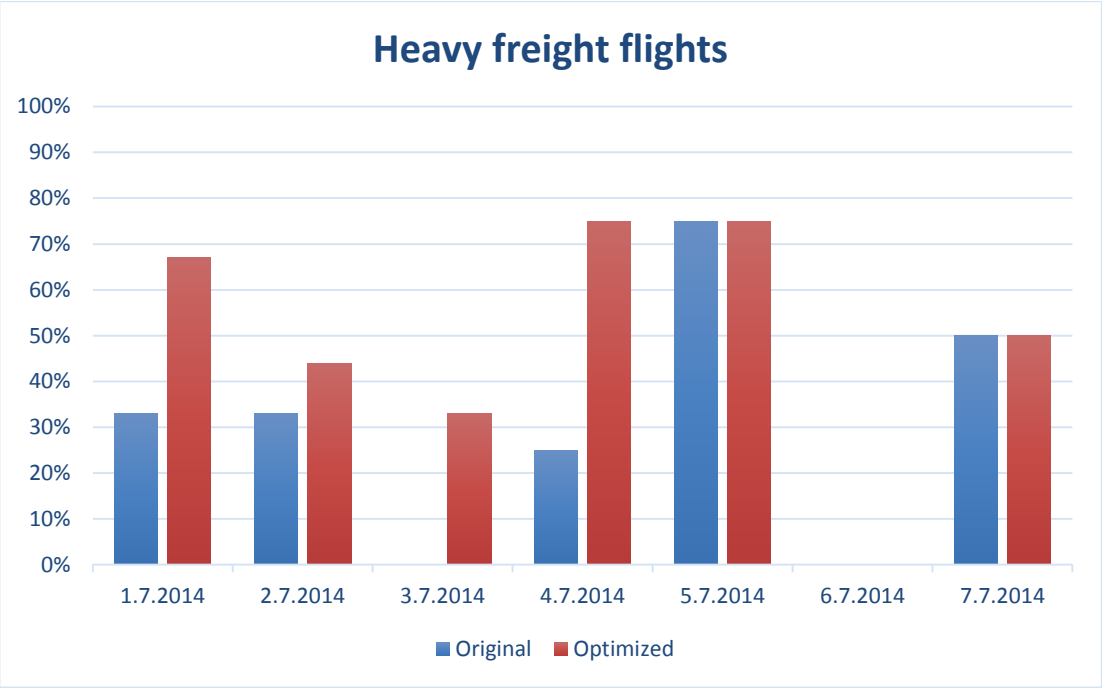

Figure 6-8: Comparison of placing heavy freight flights at best suitable gates-High season.

6.2.3 Walking distances

For each day of the examined week the optimized gate assignment decreased the total walking distance by 6.3% to 11.9% from the original assignment, based on the manually measured walking distances. The difference for each type of passengers can be seen in Table 6-3. In total passengers walked 8% less, or 4231 km. With the total number of passengers for the whole week being 119,403, each passenger walked on average 35.43 meters less with the optimized gate assignment than the original. More detailed listing of the results can be viewed in Appendix D.

Table 6-3: Comparison of total walking distances (km) based on manually measured distances – High season

	Original	*Optimized*	*Difference*
Arrival	14144	11754	-16.9%
Departure	28136	26306	-6.5%
Transfer-arr	4838	4808	-0.6%
Transfer-dep	5243	5262	0.3%
Total	52361	48130	-8.1%

The walking distances for arrival and departure passengers were also measured with ARCport. Walking distances for Schengen passengers and non-Schengen passengers were measured separately and the difference can be seen in Table 6-4. For each day the optimized gate assignment decreased the total walking distances by 1.9% to 5.6% from the original assignment. In total passengers walked 3% less, or in total 1641 km. On average each

passenger walks 13.74 meters less with the optimized gate assignment than the original, according to ARCport. More detailed listing of the results can be viewed in Appendix D.

Table 6-4: Comparison of total walking distances (km) measured in ARCport – High season

	Original	*Optimized*	*Difference*
Arr. - Non-Sch.	11340	11200	-1.2%
Arr. - Sch.	9230	8694	-5.8%
Dep. - Non-Sch.	18686	18736	0.3%
Dep. - Sch.	15765	14749	-6.4%
Total	55021	53380	-3.0%

6.3 Summary

The difference between the optimized gate assignment and the original gate assignment for both periods is summarized in Table 6-5. The table shows the total usage of bus and walk-in, walk-out gates for the whole week. The percentage of total sensitive and heavy freight flights that were assigned to the best suitable gates is also indicated along with the total number of splits. Fewer splits means fewer towing procedures which results in less operational cost. As shown in the table the optimized gate assignment manages in all cases to assign more sensitive and heavy freight flights at suitable gates. The difference in total passenger walking distance is also shown in the table, which was both measured with the manually measured distances and with ARCport.

Table 6-5: Summarized results

	Original	*Optimized*	*Difference*
Low period			
Bus gates	26	7	-73%
Walk-in gates	31	30	-3%
Sensitive	80%	100%	20%
Heavy freight	39%	44%	5%
Splits	8	5	-38%
WD-manual (km)	32833	31453	-4%
WD-ARCport (km)	31622	30823	-3%
High period			
Bus gates	104	68	-35%
Walk-in gates	81	59	-27%
Sensitive	76%	99%	24%
Heavy freight	34%	55%	21%
Splits	53	42	-21%
WD-manual (km)	52361	48130	-8%
WD-ARCport (km)	55021	53380	-3%

7 DISCUSSIONS AND FUTURE WORKS

The binary optimization model solved the GAP with little manual intervention for every tested day and according to the simulation program ARCport they were all feasible solutions. Compared to the original gate assignment the binary optimization model concluded a better outcome for each observed scenario. Both for the low season period and the high season period. Some difference is between the two periods. For the high season, the model obtains better outcomes than for the low season, for every scenario except usage of bus gates and number of splits. The reason for this difference could be because it's harder to consider all the guidelines of the gate assignment, when assigning flights to gates manually, when the traffic is the heaviest around the airport. For example keeping track of the number of each passenger type for every flight can be hard when there are numerous flights arriving and departing at the same time. The main focus goes to making sure that all flights get assigned to a gate without breaking any rules. The model however takes in every data of the flight schedule and optimizes the gate assignment in regard to all defined rules and guidelines.

The results are promising and indicate that the proposed model could be used when solving the gate assignment problem at KEF. It is important however to keep in mind that the model is based on assumptions so the presented results do not represent 100% reality. There could be some reasons for why the original gate assignment used more splits, for example, than the optimized gate assignment. An aircraft might have broken down and needed to be towed to another location for repair. Information like those were not known when the results were executed. The results were executed on the available data and when assumptions were made it was tried to implement them for both the original and the optimized gate assignment. For example, if gate 25B or 32D was used in the original gate assignment, it was changed to gate 8B or 28B, so when measuring the walking distance the comparison on the two gate assignments would be reasonable. The approach to measure the walking distance for transfer passenger with central points can be argued but since the transfer data is not available for the gate assignment at KEF it is better to include the transfer passengers in some way. If the date were to be available the model could optimize the walking distance for transfer passenger with more precision.

The binary optimization model was tested with data back in time. Given that the running time of the model was under one second for every day of the two tested weeks it should

not be a problem to run the model in real time when all necessary data are available. Saving the airport time and effort.

Automating the gate assignment at KEF gets more important as the number of total annual passengers increases. Today the airport is considered a small airport compared to other international airports, with only 16 gates but the total number of passengers is increasing. According to the latest passenger forecast the total number of passenger in the year 2020 will be just under 7 million and close to 14 million in the year 2040 [31]. Number of gates will need to increase parallel to the passenger growth and according to the airport's master plan there will be 24 gates in the year 2020 and 40 gates in the year 2040. With this growth in passenger number and enlargement of the airport's terminal, it will be extremely complicated to do the GAP manually. Optimizing the gate assignment in regards to the airport's operations and passengers' satisfaction will be necessary in the near future. This study provides the airport with the first research on how the airport can execute the gate assignment automatically.

The binary optimization model developed in this study can be used for further studies and improvements as will be explained in the next section.

7.1 Future works

The weight factors and the buffer time defined in Chapter 5.3 have not been fine tuned. It is therefore presented as future work to do a sensitivity analysis of the weight factors and the buffer time in order to see how sensitive the results are in regard to those aspects. The procedure of the sensitivity analysis would be to make several changes to each aspect, both decreasing and increasing it. Then all data would be run again through the binary optimization model and ARCport, with one change at a time to document how each change affects the results.

The results presented in Chapter 5 revealed that the need for splitting flights is great, especially for high season period where flights need to be split every day so all flights can be assigned to gates. Splitting flights that the model assigned to the no-gate dummy variable or flights that are chosen manually does not ensure optimal gate assignment. There could be flights that are better suited and by splitting them possibly fewer flights need to be split in total. As a future work it would be interesting to develop the model in the direction presented in [18] and [3]. Then all flights would be split into three different

aircraft activities, arrival, parking and departure. The three activities are modeled separately and can therefore be assigned to different positions by the model. Then there would be no need for splitting the flights afterwards. One of the objectives would be to minimize towing procedures so the model would try to assign all three activities to the same stand if possible. The manual intervention could possibly be decreased even more by this type of modeling.

The binary optimization model presented in this study was not developed in regard to its ability to respond to uncertainty. If flights would be delayed it would be needed to run the model again with changed assumptions or change the gate assignment manually, as it is done today at KEF. It's not desirable to have to reschedule the gate assignment completely if some changes to the input occur. Therefore it is presented as future work to improve the model in regard to robustness. As mentioned in Chapter 2 several researches have been published that consider the GAP in regard to robustness, for example, [4], [8], [19] and [20].

It is proposed as a future work to implement the developed binary optimization model to the planning process of the gate assignment at KEF. So the usage of the model would be easy and a part of the normal planning process at the airport. The Excel document created to access all the data needed for the model is directly connected to the airport's AODB and could be used when running the model in real time. The model could also possibly be implemented into the stand planner program.

8 CONCLUSIONS

Today the assignment of flights to gates at Keflavík international airport is done manually and nothing is being done to verify if the optimal gate assignment is being reached. The objectives with this study were to examine and outline the rules and guidelines for the gate assignment at KEF and to check if an optimization model could be used to solve the gate assignment problem. After examining and outlining the needs of the GAP a binary optimization model was developed with the main objectives to minimize the total passenger walking distance, minimize the use of bus and walk-in, walk-out gates and to assign sensitive and heavy freight flights to the best suitable gates. Weight factors were used to set the importance of each objective function.

Two weeks of real data were used to test the model, one week of low season period and one of high season period. Running one day at a time through the model and solving it with Gurobi 6.0.2 solver. The binary optimization model succeeded in finding solutions to the gate assignments for all examined days, for some days manual intervention was needed to determine which flights should be split. The running time of the model ranged from 0.05 to 0.33 seconds. Each optimized solution was run through the simulation program ARCport which confirmed the validation of the solutions. A direct comparison was made between the optimized solutions and the original gate assignment based on the model's objectives and the number of splits conducted on flights.

The results were promising and indicated that the proposed model could be used when solving the GAP at KEF. The binary optimization model would replace most of the manual work needed to execute the gate assignment, saving the airport time and effort. Minimum manual intervention is however needed in some cases, especially when the traffic is the heaviest, in order to decide which flights need to be split. Further studies and improvements of the model were suggested as future work.

According to the latest passenger forecast the number of passenger will increase by about 3 millions in the next five years at KEF. The number for gates is estimated to be 24 in the year 2020 and 40 in the year 2040. With this increase of passengers and number of gates the gate assignment becomes more complicated. The need for automating the gate assignment will therefore only increase in the current future. This study provides the airport with the first research on how to execute the gate assignment automatically while attempting to reach the optimal in regard to passenger satisfaction and the airport's operations.

REFERENCES

[1] 'Statistics - Kefairport.com', *Keflavik International Airport*. [Online]. Available: http://www.kefairport.is/English/Shortcuts/Statistics/. [Accessed: 24-Mar-2015].

[2] 'Isavia – Keflavik Airport | Iceland Naturally'. [Online]. Available: http://icelandnaturally.com/partner/isavia-keflavik-airport. [Accessed: 19-May-2015].

[3] U. Dorndorf, A. Drexl, Y. Nikulin, and E. Pesch, 'Flight gate scheduling: State-of-the-art and recent developments', *Omega*, vol. 35, no. 3, pp. 326–334, Jun. 2007.

[4] U. Dorndorf, F. Jaehn, and E. Pesch, 'Flight gate scheduling with respect to a reference schedule', *Ann. Oper. Res.*, vol. 194, no. 1, pp. 177–187, Nov. 2010.

[5] H. Ding, A. Lim, B. Rodrigues, and Y. Zhu, 'The over-constrained airport gate assignment problem', *Comput. Oper. Res.*, vol. 32, no. 7, pp. 1867–1880, Jul. 2005.

[6] A. Haghani and M.-C. Chen, 'Optimizing gate assignments at airport terminals', *Transp. Res. Part Policy Pract.*, vol. 32, no. 6, pp. 437–454, Aug. 1998.

[7] T. Obata, 'Quadratic Assignment Problem: Evaluation of Exact and Heuristic Algorithms', Rensselaer Polytechnic Institute, Troy, NY, USA, 1979.

[8] G. Diepen, J. M. van den Akker, J. A. Hoogeveen, and J. W. Smeltink, 'Finding a robust assignment of flights to gates at Amsterdam Airport Schiphol', *J. Sched.*, vol. 15, no. 6, pp. 703–715, Dec. 2012.

[9] J. P. Braaksma and J. H. Shortreed, 'Improving Airport Gate Usage with Critical Path', *Transp. Eng. J.*, vol. 97, no. 2, pp. 187–203, May 1971.

[10] O. Babić, D. Teodorović, and V. Tošić, 'Aircraft Stand Assignment to Minimize Walking', *J. Transp. Eng.*, vol. 110, no. 1, pp. 55–66, 1984.

[11] R. S. Mangoubi and D. F. X. Mathaisel, 'Optimizing Gate Assignments at Airport Terminals', *Transp. Sci.*, vol. 19, no. 2, pp. 173–188, May 1985.

[12] S. X. Zhang, J. Cesarone, and F. G. Miller, 'A comparative study of an aircraft assignment problem at a large airport', *Int. J. Ind. Eng.*, vol. 1, no. 3, pp. 203–212, Aug. 1994.

[13] J. Xu and G. Bailey, 'The airport gate assignment problem: mathematical model and a tabu search algorithm', in *Proceedings of the 34th Annual Hawaii International Conference on System Sciences, 2001*, 2001, p. 10 pp.–.

[14] S. Yan and C.-M. Huo, 'Optimization of multiple objective gate assignments', *Transp. Res. Part Policy Pract.*, vol. 35, no. 5, pp. 413–432, Jun. 2001.

[15] S. Yan, C.-Y. Shieh, and M. Chen, 'A simulation framework for evaluating airport gate assignments', *Transp. Res. Part Policy Pract.*, vol. 36, no. 10, pp. 885–898, Dec. 2002.

[16] H. Ding, A. Lim, B. Rodrigues, and Y. Zhu, 'New heuristics for over-constrained flight to gate assignments', *J. Oper. Res. Soc.*, vol. 55, no. 7, pp. 760–768, 2004.

[17] H. Ding, A. Lim, B. Rodrigues, and Y. Zhu, ‘Aircraft and gate scheduling optimization at airports’, in *Proceedings of the 37th Annual Hawaii International Conference on System Sciences, 2004*, 2004, p. 8 pp.–.

[18] U. Dorndorf, *Project Scheduling with Time Windows: From Theory to Applications; with 17 Tables*. Springer Science & Business Media, 2002.

[19] U. Dorndorf, F. Jaehn, and E. Pesch, ‘Modelling Robust Flight-Gate Scheduling as a Clique Partitioning Problem’, *Transp. Sci.*, vol. 42, no. 3, pp. 292–301, Mar. 2008.

[20] S. H. Kim, E. Feron, J.-P. Clarke, A. Marzuoli, and D. Delahaye, ‘Airport Gate Scheduling for Passengers, Aircraft, and Operation’, *ArXiv13013535 Cs*, Jan. 2013.

[21] ‘The countries of the Schengen area | AXA Schengen’. [Online]. Available: https://www.axa-schengen.com/en/countries-schengen-area. [Accessed: 30-May-2015].

[22] ‘Schengen’, *Utanríkisráðuneyti*. [Online]. Available: http://www.utanrikisraduneyti.is/samningar/schengen/nr/9. [Accessed: 12-Apr-2015].

[23] ‘EUR-Lex - 32008R0300 - EN - EUR-Lex’. [Online]. Available: http://eur-lex.europa.eu/legal-content/EN/TXT/?qid=1430998213197&uri=CELEX:32008R0300. [Accessed: 18-May-2015].

[24] Isavia, privat communication, May, 2015.

[25] ‘Gurobi Optimization, Inc.’ [Online]. Available: http://www.gurobi.com/. [Accessed: 29-May-2015].

[26] A. Makhorin, ‘GNU Linear Programming Kit Reference Manual for GLPK Version 4.55’, Moscow Aviation Institute, Moscow, Russia, Aug. 2014.

[27] ‘Welcome to Arc’. [Online]. Available: http://www.arc-us-ca.com/. [Accessed: 14-May-2015].

[28] ‘AutoCAD Products | Find The Right CAD Program For You’. [Online]. Available: http://www.autodesk.com/products/all-autocad. [Accessed: 29-May-2015].

[29] S. Young, ‘Evaluation of Pedestrian Walking Speeds in Airport Terminals’, *Transp. Res. Rec. J. Transp. Res. Board*, vol. 1674, pp. 20–26, Jan. 1999.

[30] ‘Statistics Iceland’. [Online]. Available: http://www.statice.is/. [Accessed: 29-Apr-2015].

[31] Grétar Már Garðarsson, Project manager at Isavia, privat communication, May, 2015.

APPENDICES

A. Binary optimization model written in GNU math language

```
set N;          #Set of flights

set M;          #Set of gates


/*Subsets of flights*/

set Nsch;       #set of flights that are either arriving or departing to a non-Schengen country

set C;          #set of flights that are arriving from Canada

set F;          #set of flights that have heavy freight (>3000kg)

set L;          #set of flights that have a large aircraft

set S;          #set of flights that are sensitive


/*Subsets of gates*/

set G1;         #set of terminal gates that are dual, both Sch. and Non-Sch.

set G2;         #set of bus gates that are dual, both Sch. and Non-Sch.

set G3;         #set of bus gates that only flights arr and dep to Sch. can use

set G4;         #set of gates that are walk-in walk-out gates and dual

set G5;         #set of gates that serve passengers arriving from Canada

set G6;         #set of gates that serve large aircrafts

set G7;         #set of gates that serve well flights with much freight

set G8;         #set of gates that serve well sensitive flights


param PA{i in N}, >= 0;             #Number of arrival pax in flight i

param PD{i in N}, >= 0;             #Number of departure pax in flight i

param PT_arr{i in N}, >= 0;         #Number of arriving transfer pax in flight i

param PT_dep{i in N}, >= 0;         #Number of departure transfer pax in flight i

param paxTotal{i in N} := PA[i]+PD[i]+PT_arr[i]+PT_dep[i];
```

```
param DA{i in N, j in M}, >= 0;          #Distance for Arrival pax from gate j

param DD{i in N, j in M}, >= 0;          #Distance for Departure pax from gate j

param DT_arr{i in N, j in M}, >= 0;      #Distance for arriving transfer pax from gate j

param DT_dep{i in N, j in M}, >= 0;      #Distance for departure transfer pax from gate
j


param TA{i in N}, >= 0;                  #Time of arrival of the ith flight

param TD{i in N}, >= 0;                  #Time of departure of the ith flight


param B >= 0;                            #Constant buffer time

param bigM >= 0;                         #Very large number

param alpha1;        #weight factor 1 to determine the importance of minimizing walking
distance

param alpha2;        #weight factor 2 to determine the importance of not using bus gates

param alpha3;        #weight factor 3 to determine the importance of not using walk-in,
walk-out gates

param alpha4;        #weight factor 4 to determine the importance of heavy freight flights

param alpha5;        #weight factor 5 to determine the importance of assigning sensitive
flights to G8

param alpha6;        #penalty for not assigning flight to a gate


var y{i in N, j in M} binary;

var nogate{i in N} binary;


Minimize GateAssignment: sum{i in N, j in M}(PA[i]*DA[i,j]+PD[i] * DD[i,j] +
PT_arr[i] * DT_arr[i,j] + PT_dep[i] * DT_dep[i,j]) *alpha1* y[i,j] +

                sum{i in N, j in G2 union G3} alpha2 * paxTotal[i] * y[i,j] +

                sum{i in N, j in G4} alpha3 * paxTotal[i] * y[i,j] +

                sum{i in F, j in G7} alpha4 * y[i,j] +

                sum{i in S, j in G8} alpha5 * paxTotal[i] * y[i,j]+

                sum{i in N} alpha6 * nogate[i] *paxTotal[i];
```

```
/* Constraints */

s.t. FlighttoGate {i in N}: sum{j in M} y[i,j] + nogate[i] = 1;

s.t. NoOverlap {i1 in N, i2 in N, j in M: i1 <> i2}: (TD[i1]+B-TA[i2])*(TD[i2]+B-TA[i1]) <= 0+(1-y[i1,j])*bigM +(1-y[i2,j])*bigM;

s.t. NonSchengen {i in Nsch}: sum{j in G1 union G2 union G4} y[i,j] + nogate[i]= 1;

s.t. CanadaArr {i in C}: sum{j in G5} y[i,j] = 1;

s.t. LargeAircrafts {i in L}: sum{j in G6} y[i,j] = 1;

end;
```

B. Programs made

1. Adapting data for insert into ARCport

```
import cell2csv.*                    %
http://www.mathworks.com/matlabcentral/fileexchange/4400-cell-array-to-
csv-file--cell2csv-m-
%import dlmcell.*                    %
http://www.mathworks.com/matlabcentral/fileexchange/25387-write-cell-
array-to-text-file/content/dlmcell.m

filename = 'BIKF_26.04.2015.xlsx'; %a file needs to exists with all
flight schedules

%date = filename(13:22);

[num,txt,raw] = xlsread(filename);        %Reading the file
[rows,columns] = size(raw);               %checking the size of the file

date = input('Please enter the date that is desired to view on the
format dd.mm.yyyy: ','s');
% date = '21.01.2015';

k=1;                    %initializing the new array with the input date

%creating a new array with the input date and types J,C,G,S
for i = (2:rows)
    if strncmp(raw{i,41},date,10) && (isequal(raw{i,3},'J') ||
isequal(raw{i,3},'C') || isequal(raw{i,3},'S')|| isequal(raw{i,3},'G'));
        Newarr(k,:)=raw(i,:);
        k=k+1;
    end
end

%deleting all unnecessary columns
Newarr(:,[2,3,8,9,13,14,15,18,19,20,21,23,24,25,26,27,28,29,30,31,32,33,
34,35,36,39,40,42,43,44,45,46,47,48,49,50,52])=[];

[r,c]=size(Newarr);

%Erasing flights that have no passengers on it
for i=1:size(Newarr,1);
    if Newarr{i,12}==0 && Newarr{i,13}==0;
        Newarr(i,2)=num2cell(20000);
    end
end
Newarr(any(cellfun(@(x)x(1)==20000, Newarr),2), :) = [];

%creating a new array in order to split up columns from the Newarr
ArrDep = cell(size(Newarr,1),29);
ArrDep(:,[6,8]) = Newarr(:,[6,8]); %%column 6 & 8 stay the same
for j = (1:size(Newarr,1))
    if isequal(Newarr{j,9},'KEF'); %%if the flight is a Departure flight
        ArrDep(j,[18:29]) = Newarr(j,[1:3,5,7,11:17]);
        ArrDep(j,10) = Newarr(j,10); %% Destination stays the same
    else
        ArrDep(j,[1:5,7,11:17]) = Newarr(j,[1:5,7,11:17]); %%if the
flight was a arrival flight, the columns remain the same (Arr)
        ArrDep(j,9) = Newarr(j,9); %%the origin stays the same
    end
```

```
end

%%Change the name of the stands and gates to the names ARCport knows
for j = (1:size(ArrDep,1))
    if isempty(ArrDep{j,2}) == 0;
        switch ArrDep{j,2}
            case '8B'
                ArrDep{j,2} = 'ARRGATE-BUS-SHENGEN-8B';
            case '8C'
                ArrDep{j,2} = 'ARRGATE-BUS-SHENGEN-8C';
            case '28B'
                ArrDep{j,2} = 'ARRGATE-BUS-SHENGEN-8B';
            case '28C'
                ArrDep{j,2} = 'ARRGATE-BUS-SHENGEN-8C';
            case 'A'
                ArrDep{j,2} = 'ARRGATE-BUS-SHENGEN-A';
            case '25B'
                ArrDep{j,2} = 'ARRGATE-BUS-SHENGEN-25B';
            case '32D'
                ArrDep{j,2} = 'ARRGATE-BUS-SHENGEN-8B';
            case {'21','22','23','24','62','63','64','65','111'}
                ArrDep{j,2} = 'ARRGATE-BUS-SHENGEN-8B';
            case {'1','2','3','4','5','6','7','8','9','11','12'}
                ArrDep{j,2} = ['ARRGATE-CONTACT-SHENGEN-',ArrDep{j,2}];
            case {'25','26','27','28','29','31','32'}
                gildi = str2num(ArrDep{j,2})-20;
                gildi = num2str(gildi);
                ArrDep{j,2} = ['ARRGATE-CONTACT-SHENGEN-',gildi];
            case {'10','14'}
                ArrDep{j,2} = ['ARRGATE-WALKOUT-SHENGEN-',ArrDep{j,2}];
            case {'30','34'}
                gildi2 = str2num(ArrDep{j,2})-20;
                gildi2 = num2str(gildi2);
                ArrDep{j,2} = ['ARRGATE-WALKOUT-SHENGEN-',gildi2];
        end
    end
     if isempty(ArrDep{j,3}) == 0;
        switch ArrDep{j,3}
            case
{'1','2','3','4','5','6','7','8','9','10','11','12','14','15'}
                ArrDep{j,3} = ['STAND-',ArrDep{j,3}];
            case {'20','21','22','23','24','62','63','64','65','111'}
                ArrDep{j,3} = ['STAND-REMOTE-',ArrDep{j,3}];
        end
    end
    if isempty(ArrDep{j,19}) == 0;
        switch ArrDep{j,19}
            case {'8B','21','22','23','24','62','63','64','65','111'}
                ArrDep{j,19} = 'DEPGATE-BUS-SHENGEN-8B';
            case '8C'
                ArrDep{j,19} = 'DEPGATE-BUS-SHENGEN-8C';
            case '28B'
                ArrDep{j,19} = 'DEPGATE-BUS-SHENGEN-8B';
            case '28C'
                ArrDep{j,19} = 'DEPGATE-BUS-SHENGEN-8C';
            case 'A'
                ArrDep{j,19} = 'DEPGATE-BUS-SCHENGEN-A';
            case {'1','2','3','4','5','6','7','8','9','11','12'}
                ArrDep{j,19} = ['DEPGATE-CONTACT-SHENGEN-
',ArrDep{j,19}];
            case {'25','26','27','28','29','31','32'}
```

```
                gildi = str2num(ArrDep{j,19})-20;
                gildi = num2str(gildi);
                ArrDep{j,19} = ['DEPGATE-CONTACT-SHENGEN-',gildi];
            case {'10','14'}
                ArrDep{j,19} = ['DEPGATE-WALKOUT-SHENGEN-
',ArrDep{j,19}];
            case {'30','34'}
                gildi = str2num(ArrDep{j,19})-20;
                gildi = num2str(gildi);
                ArrDep{j,19} = ['DEPGATE-WALKOUT-SHENGEN-',gildi];
        end
    end
    if isempty(ArrDep{j,20}) == 0;
        switch ArrDep{j,20}

case{'1','2','3','4','5','6','7','8','9','10','11','12','14','15'}
                ArrDep{j,20} = ['STAND-',ArrDep{j,20}];
         case {'20','21','22','23','24','62','63','64','65','111'}
                ArrDep{j,20} = ['STAND-REMOTE-',ArrDep{j,20}];
        end
    end
    if isempty(ArrDep{j,4}) == 0;
        switch ArrDep{j,4}
            case {'1','2','3'}
                ArrDep{j,4} = ['RECLAIM-',ArrDep{j,4}];
        end
    end
end

%Change the load factor from 0.96 to 96.00
for i = 1:length(ArrDep);
    ArrDep{i,15} = ArrDep{i,15}*100;
    ArrDep{i,27} = ArrDep{i,27}*100;
end

header = {'Sched-arr time', 'Arr gate', 'Arr stand', 'Arr belt', 'Arr
Id', 'Airline', 'Arr-reg', 'AC type', 'Origin', 'Destination', 'Real-arr
time', 'Transfer-arr pax','Arr-pax', 'Arr day','Arr load factor', 'Arr
freight','Arr sch/nonsch','Sched-dep time', 'Dep gate', 'Dep stand',
'Dep Id', 'Dep-reg','Real-dep time', 'Transfer-dep pax','Dep pax','Dep
day', 'Dep load factor','Dep freight','Dep sch/nonsch'};
FinArr = [header; ArrDep]; %%creating new array with header row

name = strcat('Flight_sch-',num2str(date),'.csv'); %%making the input
date being the name of the new csv file
cell2csv(name,FinArr,';',2007, '.'); %%writing out the csv file with the
adapted flight schedule
%dlmcell(name,FinArr);
```

2. Adapting data for binary optimization model

```
clear all; close all; clc;
date = '01.07.2014';
filename = strcat('Flight_data-',num2str(date),'.csv'); %%a file needs
to exists with all flight schedules

[num,txt,raw] = xlsread(filename);        %Reading the file
[rows,columns] = size(raw);               %checking the size of the file

%Earsing flights that have no passengers on it
```

```
for i=2:size(raw,1);
    if raw{i,12}==0 && raw{i,13}==0;
        raw(i,2)=num2cell(20000);
    end
end
raw(any(cellfun(@(x)x(1)==20000, raw),2), :) = [];

%%Data needed for optimization
B = 30;
bigM = 100000;
alpha1 = 0.5;          %%walking distance
alpha2 = 200;          %%min bus gates
alpha3 = 40;          %%min walkin-walkout gates
alpha4 = -299;         %%min freight
alpha5 = -300;         %%min sensitive
alpha6 = 10000;        %punsihment for assigning flights to nogate

%Adjusting the data
for i = (2:rows)
    t = (raw{i,1});
    [Y,M,D,H,MN,S] = datevec(t);
    t = 60*H+MN+S/60;
    raw(i,1) = num2cell(t);
    y = (raw{i,11});
    [Y,M,D,H,MN,S] = datevec(y);
    y = 60*H+MN+S/60;
    raw(i,11) = num2cell(y);
    raw{i,13} = raw{i,13}-raw{i,12};
end

%Erasing flights that have no passengers on
for i=1:size(raw,1);
    if isequal(raw{i,12},0) && isequal(raw{i,13},0);
        raw(i,2)=num2cell(20000);
    end
end
raw(any(cellfun(@(x)x(1)==20000, raw),2), :) = [];

%Picking all arrival flights and adding them to new array Newarr
k=2;
N=0;
Newarr = {'N','TA','Flight_num','Origin','PA','PT_arr','Arr_REG','Arr-
freight','Arr-sch', 'TD','Flight_num', 'Destination','PD',
'PT_dep','Dep_REG','Dep-freight','Dep-sch','Arrgate','Depgate'};
for i = (2:size(raw,1))
    if isequal(raw{i,10},'KEF');
        N=N+1;
        Newarr{k,1} = N;
        Newarr{k,2}=raw{i,1};
        Newarr{k,3}=raw{i,5};
        Newarr{k,4}=raw{i,9};
        Newarr{k,5}=raw{i,13};
        Newarr{k,6}=raw{i,12};
        Newarr{k,7}=raw{i,7};
        Newarr{k,8}=raw{i,16};
        Newarr{k,9}=raw{i,17};
        Newarr{k,18}=raw{i,2};
        k=k+1;
    end
end
```

```
[r,c] = size(Newarr);

%Find if a connecting departing flight exists, if not add a dummy
departure
%flight that leaves 40 min later. Because of 30 min buffer it is 10min
%if the flight is arriving after 22:00 make the dummy departure be one
hour later
for l = (2:r);
    sannindi = 0;
    for i = (2:size(raw,1));
        if (isequal(raw{i,7},Newarr{l,7})) && (raw{i,1} > Newarr{l,2})
&& (isequal(raw{i,9},'KEF'))&& sannindi==0;
            Newarr{l,10} = raw{i,1};
            Newarr{l,11} = raw{i,5};
            Newarr{l,12} = raw{i,10};
            Newarr{l,13} = raw{i,13};
            Newarr{l,14} = raw{i,12};
            Newarr{l,15} = raw{i,7};
            Newarr{l,16} = raw{i,16};
            Newarr{l,17} = raw{i,17};
            Newarr{l,19} = raw{i,2};
            sannindi = 1;
        end
    end
    if sannindi == 0;
        if Newarr{l,2} > 1320
            Newarr{l,10} = Newarr{l,2}+30;
        else
            Newarr{l,10} = Newarr{l,2}+10;
        end
        Newarr{l,11} = 0;
        Newarr{l,12} = 0;
        Newarr{l,13} = 0;
        Newarr{l,14} = 0;
        Newarr{l,15} = 0;
        Newarr{l,16} = [];
        Newarr{l,17} = [];
        Newarr{l,19} = [];
    end
end

%Add the departure flights that had no arrival flight connection
N=N+1;
k=1;
for i = (2:size(raw,1));
    if (isequal(raw{i,9},'KEF'));
        foundMatch =0;
        sann=0;
        for j = (2:size(Newarr,1));
            if (isequal(raw{i,5},Newarr{j,11}));
                foundMatch=1;
                break;
            end
        end
        if(foundMatch==0);
            r=r+1;
            Newarr{r,1} = N;
            Newarr{r,2}=raw{i,1}-10;
            Newarr{r,3}=0;
            Newarr{r,4}=0;
```

```
            Newarr{r,5}=0;
            Newarr{r,6}=0;
            Newarr{r,7}=0;
            Newarr{r,8}=0;
            Newarr{r,9}=[];
            Newarr{r,10} = raw{i,1};
            Newarr{r,11} = raw{i,5};
            Newarr{r,12} = raw{i,10};
            Newarr{r,13} = raw{i,13};
            Newarr{r,14} = raw{i,12};
            Newarr{r,15} = raw{i,7};
            Newarr{r,16} = raw{i,16};
            Newarr{r,17} = raw{i,17};
            Newarr{r,19} = raw{i,2};
            N=N+1;
            sann=1;
        end
    end
end

%if dummy time is less than 0 (minus), change to 0.
for i=1:size(Newarr,1);
    if Newarr{i,2} < 0;
        Newarr{i,2} = 0;
    end
end

%make subsets of flights
%Nsch flights
k=1;
for i = (2:size(Newarr,1))
    if (isequal(Newarr{i,9},0)) || (isequal(Newarr{i,17},0));
        Nsch{k,1} = Newarr{i,1};
        k=k+1;
    end
end

%Canadian arrivals
k=1;
C = [];
for i = (2:size(Newarr,1))
    if isequal(Newarr{i,4},'YHZ')  || isequal(Newarr{i,4},'YYZ') ||
isequal(Newarr{i,4},'YEG') || isequal(Newarr{i,4},'YVR');
        C{k,1} = Newarr{i,1};
        k=k+1;
    end
end

%Heavy freight flights
k=1;
F = [];
for i = (2:size(Newarr,1))
    if ((Newarr{i,16} > 3000) | (Newarr{i,8} > 3000));
        F{k,1} = Newarr{i,1};
        k=k+1;
    end
end

%Large flights
k=1;
```

```
L=[];
for i = (2:size(Newarr,1))
    if isequal(Newarr{i,7},'TFFIX') || isequal(Newarr{i,15},'TFFIX');
        L{k,1} = Newarr{i,1};
        k=k+1;
    end
end

%Sensitive flights
k=1;
Sensitive=[];
for i = (2:size(Newarr,1))
     if isequal(Newarr{i,12},'LHR') || isequal(Newarr{i,12},'LGW')||
isequal(Newarr{i,12},'BWI')|| isequal(Newarr{i,12},'HEL') ||
isequal(Newarr{i,12},'JFK') || isequal(Newarr{i,12},'MUC') ||
isequal(Newarr{i,12},'FRA') || isequal(Newarr{i,12},'CDG') ||
isequal(Newarr{i,12},'YYZ') || isequal(Newarr{i,12},'IAD') ||
isequal(Newarr{i,12},'MSP');
         Sensitive{k,1} = Newarr{i,1};
         k=k+1;
     end
end

%Creating two arrays with measured sch distances and non-sch distances
sch_Distances = {1,98.42,528.60,382.94,343.74;
                 2,132.40,538.60,349.06,308.40;
                 3,165.96,573.95,315.85,295.96;
                 4,196.88,603.29,284.76,274.03;
                 5,233.03,641.57,248.54,230.33;
                 6,264.00,670.76,217.74,214.37;
                 7,364.3,813.10,242.80,222.64;
                 8,412.50,816.23,291.07,225.77;
                 9,409.33,814.82,287.90,224.35;
                 10,406.83,814.00,285.40,223.54;
                 11,382.93,789.91,261.51,199.45;
                 12,401.65,814.40,280.23,223.93;
                 14,471.75,878.93,350.33,288.47;
                 15,416.45,827.86,295.02,237.39;
                 16,438.91,850.32,317.48,259.85;
                 17,177.70,553.37,496.40,457.19};
non_sch_Distances = {
                     1,98.42,528.60,382.94,343.74;
                     2,132.40,538.60,349.06,308.40;
                     3,165.96,573.95,315.85,295.96;
                     4,196.88,603.29,284.76,274.03;
                     25,501.57,916.06,258.41,288.37;
                     26,495.10,849.20,251.93,280.45;
                     27,464.15,910.26,220.98,282.57;
                     28,493.82,940.96,250.66,313.27;
                     29,476.35,921.14,233.18,293.45;
                     30,459.93,906.65,216.76,278.96;
                     31,449.47,921.37,206.30,293.68;
                     32,465.36,932.34,222.20,304.65;
                     34,502.51,949.12,259.35,321.43;
                     15,440.08,886.94,196.92,259.25;
                     16,456.46,903.12,213.29,275.43;
                     17,177.70,553.37,496.40,457.19};
%creating new arry with walking distances for each flight depending on
if
%the flight is sch or non-sch.
k=1;
```

```
for i = 2:size(Newarr,1);
    for j = 1:size(sch_Distances,1);
        dist_per_flight{k,1} = Newarr{i,1};
        dist_per_flight{k,2} = sch_Distances{j,1};
        if Newarr{i,9}==1;      %arrival
            dist_per_flight{k,3} = sch_Distances{j,2};
            dist_per_flight{k,5} = sch_Distances{j,4};
        else
            dist_per_flight{k,3} = non_sch_Distances{j,2};
            dist_per_flight{k,5} = non_sch_Distances{j,4};
        end
        if Newarr{i,17} ==1; %departure
            dist_per_flight{k,4} = sch_Distances{j,3};
            dist_per_flight{k,6} = sch_Distances{j,5};
        else
            dist_per_flight{k,4} = non_sch_Distances{j,3};
            dist_per_flight{k,6} = non_sch_Distances{j,5};
        end
        k=k+1;
    end
end

name = strcat('numberOfFlights-',num2str(date),'.csv'); %%making the
inptut date being the name of the new csv file
xlswrite(name,Newarr);

name2 = strcat('Distances_per_flight-',num2str(date),'.csv'); %printing
out array with walking distances
xlswrite(name2,dist_per_flight);

M = sch_Distances(:,1);

DA = dist_per_flight(:,[1,2,3]);
DD = dist_per_flight(:,[1,2,4]);
DT_arr = dist_per_flight(:,[1,2,5]);
DT_dep = dist_per_flight(:,[1,2,6]);


G1 = [5;6;7;8;9;11;12];             %Dual contact
G2 = [15;16];                       %Dual bus
G3 = 17;                            %Sch bus
G4 = [10;14];                       %walkin
G5 = [5;6;15;16];                   %Candian arrival
G6 = [4;7;12];                      %stands for Large aircrafts
G7 = [7;9;12];                      %Freight gates
G8 = [1;2;3;4;5;6;7;8;9;11;12];     %Sensitive

%Printing data out in text file for optimisation model
string1 = 'param B := ';
string2 = ';';
string3 = 'param bigM := ';
string4 = 'param alpha1 := ';
string5 = 'param alpha2 := ';
string6 = 'param alpha3 := ';
string7 = 'param alpha4 := ';
string8 = 'param alpha5 := ';
string9 = 'set N := ';
string10 = 'param TA := ';
string11 = 'param PA := ';
string12 = 'param PT_arr := ';
```

```
string13 = 'param TD := ';
string14 = 'param PD := ';
string15 = 'param PT_dep := ';
string16 = 'set M := ';
string17 = 'set Nsch := ';
string18 = 'set C := ';
string19 = 'set F := ';
string20 = 'set L := ';
string21 = 'set S := ';
string22 = 'param DA := ';
string23 = 'param DD := ';
string24 = 'param DT_arr := ';
string25 = 'param DT_dep := ';
string26 = 'set G1 := ';
string27 = 'set G2 := ';
string28 = 'set G3 := ';
string29 = 'set G4 := ';
string30 = 'set G5 := ';
string31 = 'set G6 := ';
string32 = 'set G7 := ';
string33 = 'set G8 := ';
string34 = 'end;';
string35 = 'param alpha6 := ';
string37 = 'set Connected := ';


N = Newarr(:,1);
TA = Newarr(:,[1,2]);
PA = Newarr(:,[1,5]);
PT_arr = Newarr(:,[1,6]);
TD = Newarr(:,[1,10]);
PD = Newarr(:,[1,13]);
PT_dep = Newarr(:,[1,14]);

fName = strcat('Data-',num2str(date),'.dat');
fileID = fopen(fName,'w');
fprintf(fileID, '%s%d%s\n',string1,B,string2);
fprintf(fileID, '%s%d%s\n',string3,bigM,string2);
fprintf(fileID, '%s%d%s\n',string4,alpha1,string2);
fprintf(fileID, '%s%d%s\n',string5,alpha2,string2);
fprintf(fileID, '%s%d%s\n',string6,alpha3,string2);
fprintf(fileID, '%s%d%s\n',string7,alpha4,string2);
fprintf(fileID, '%s%d%s\n\n',string8,alpha5,string2);
fprintf(fileID, '%s%d%s\n\n',string35,alpha6,string2);
fprintf(fileID, '%s\n',string9);
for i = 2:size(N,1);
    fprintf(fileID, '%d\n',N{i,:});
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string10);
for i = 2:size(N,1);
    fprintf(fileID, '%d %d\n',TA{i,:});
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string11);
for i = 2:size(N,1);
    fprintf(fileID, '%d %d\n',PA{i,:});
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string12);
for i = 2:size(N,1);
```

```
    fprintf(fileID, '%d %d\n',PT_arr{i,:});
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string13);
for i = 2:size(N,1);
    fprintf(fileID, '%d %d\n',TD{i,:});
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string14);
for i = 2:size(N,1);
    fprintf(fileID, '%d %d\n',PD{i,:});
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string15);
for i = 2:size(N,1);
    fprintf(fileID, '%d %d\n',PT_dep{i,:});
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string16);
for i = 1:size(M,1);
    fprintf(fileID, '%d\n',M{i,:});
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string17);
for i = 1:size(Nsch,1);
    fprintf(fileID, '%d\n',Nsch{i,:});
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string18);
for i = 1:size(C,1);
    fprintf(fileID, '%d\n',C{i,:});
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string19);
for i = 1:size(F,1);
    fprintf(fileID, '%d\n',F{i,:});
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string20);
for i = 1:size(L,1);
    fprintf(fileID, '%d\n',L{i,:});
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string21);
for i = 1:size(Sensitive,1);
    fprintf(fileID, '%d\n',Sensitive{i,:});
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string22);
for i = 1:size(DA,1);
    fprintf(fileID, '%d %d %f\n',DA{i,:});
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string23);
for i = 1:size(DD,1);
    fprintf(fileID, '%d %d %f\n',DD{i,:});
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string24);
for i = 1:size(DT_arr,1);
    fprintf(fileID, '%d %d %f\n',DT_arr{i,:});
```

```
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string25);
for i = 1:size(DT_dep,1);
    fprintf(fileID, '%d %d %f\n',DT_dep{i,:});
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string26);
for i = 1:size(G1,1);
    fprintf(fileID, '%d\n',G1(i,:));
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string27);
for i = 1:size(G2,1);
    fprintf(fileID, '%d\n',G2(i,:));
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string28);
for i = 1:size(G3,1);
    fprintf(fileID, '%d\n',G3(i,:));
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string29);
for i = 1:size(G4,1);
    fprintf(fileID, '%d\n',G4(i,:));
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string30);
for i = 1:size(G5,1);
    fprintf(fileID, '%d\n',G5(i,:));
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string31);
for i = 1:size(G6,1);
    fprintf(fileID, '%d\n',G6(i,:));
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string32);
for i = 1:size(G7,1);
    fprintf(fileID, '%d\n',G7(i,:));
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string33);
for i = 1:size(G8,1);
    fprintf(fileID, '%d\n',G8(i,:));
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n\n',string34);
fclose(fileID);
```

3. Adding solution from optimization model to format for ARCport

```
close all; clear all; clc;
%import cell2csv.*      %
http://www.mathworks.com/matlabcentral/fileexchange/4400-cell-array-to-
csv-file--cell2csv-m-
import dlmcell.*              %
http://www.mathworks.com/matlabcentral/fileexchange/25387-write-cell-
array-to-text-file/content/dlmcell.m
```

```
date = '02.07.2014';        %Desired date

%Reading in the flight schedule on the format needed for Arcport
filename = strcat('Flight_sch-',num2str(date),'.csv');
[num,txt,sim] = xlsread(filename);

%Reading in the solution file from Gurobi
resultFile = strcat(num2str(date),'.sol');
fid = fopen(resultFile);
formatSpec = 'y[%d,%d] %d';
C = textscan(fid, formatSpec,'delimiter','\n','CollectOutput',true,
'HeaderLines',2);
fclose(fid);

C=[C{:}];
C(any(C==0,2),:)=[]; %array with the results

%Reading in Flights document
filename = strcat('splitFlights2-',num2str(date),'.csv');
%splitFlights2 or numberof
[num,txt,flights] = xlsread(filename);

%Adding allocated gates to the flights
flights{1,20} = 'Gates';

for i = 2:size(flights,1);
    for j = 1:size(C,1);
        if isequal(C(j,1),flights{i,1});
            flights{i,20} = C(j,2);
        end
    end
end

name = strcat('Flights_WD-',num2str(date),'.csv'); %%making the inptut
date being the name of the new csv file
xlswrite(name,flights);

%Adding the allocated gates to the format for simulation
for i = 2:size(sim,1);
    for j = 2:size(flights,1);
        if isequal(sim{i,7},flights{j,7}) &&
isequal(sim{i,5},flights{j,3}) && isequal(sim{i,9},flights{j,4});
            sim{i,2}=flights{j,20};
            sim{i,3}=flights{j,20};
        end
        if isequal(sim{i,22},flights{j,15}) &&
isequal(sim{i,21},flights{j,11}) && isequal(sim{i,10},flights{j,12});
            sim{i,19}=flights{j,20};
            sim{i,20}=flights{j,20};
        end
    end
end

%change NaN to empty
 for i = 2:size(sim,1);
     for j = 1:size(sim,2);
         NaNIndex = find(isnan(sim{i,j}));
         if NaNIndex ==1;
```

```
            sim{i,j} = [];
        end
    end
end

%change format on time back to HH:MM:SS
for i = 2:size(sim,1)
    if isempty(sim{i,1})== 0;
     t = (sim{i,1});
     [Y,M,D,HH,MM,SS] = datevec(t);
     Hrs = num2str(HH);
     Min = num2str(MM);
     Sec = num2str(SS);
     zero= num2str(0);
     t= [Hrs,':',Min,':',Sec,zero];
     sim{i,1} = t;
    end
    if isempty(sim{i,11})== 0;
     y = (sim{i,11});
     [Y,M,D,HH,MM,SS] = datevec(y);
     Hrs = num2str(HH);
     Min = num2str(MM);
     Sec = num2str(SS);
     zero= num2str(0);
     y= [Hrs,':',Min,':',Sec,zero];
     sim{i,11} = y;
    end
    if isempty(sim{i,18})== 0;
     u = (sim{i,18});
     [Y,M,D,HH,MM,SS] = datevec(u);
     Hrs = num2str(HH);
     Min = num2str(MM);
     Sec = num2str(SS);
     zero= num2str(0);
     u= [Hrs,':',Min,':',Sec,zero];
     sim{i,18} = u;
    end
    if isempty(sim{i,23})== 0;
     o = (sim{i,23});
     [Y,M,D,HH,MM,SS] = datevec(o);
     Hrs = num2str(HH);
     Min = num2str(MM);
     Sec = num2str(SS);
     zero= num2str(0);
     o= [Hrs,':',Min,':',Sec,zero];
     sim{i,23} = o;
    end
end

%Changing the gates names to what Arcport knows
for j = (1:size(sim,1))
    if isempty(sim{j,2}) == 0
        switch sim{j,2}
            case 15
                sim{j,2} = 'ARRGATE-BUS-SHENGEN-8B';
            case 16
                sim{j,2} = 'ARRGATE-BUS-SHENGEN-8C';
            case 17
                sim{j,2} = 'ARRGATE-BUS-SHENGEN-A';
            case {1,2,3,4,5,6,7,8,9,11,12}
```

```
                sim{j,2} = ['ARRGATE-CONTACT-SHENGEN-
',num2str(sim{j,2})];
            case {10,14}
                sim{j,2} = ['ARRGATE-WALKOUT-SHENGEN-
',num2str(sim{j,2})];
        end
    end
    if isempty(sim{j,3}) == 0;
        switch sim{j,3}
            case {1,2,3,4,5,6,7,8,9,10,11,12,14}
                sim{j,3} = ['STAND-',num2str(sim{j,3})];
            case {15}
                sim{j,3} = 'STAND-REMOTE-20';
            case {16}
                sim{j,3} = 'STAND-REMOTE-21';
            case {17}
                sim{j,3} = 'STAND-REMOTE-22';
        end
    end
    if isempty(sim{j,4}) == 0;
        switch sim{j,4}
            case {1,2,3}
                sim{j,4} = ['RECLAIM-',num2str(sim{j,4})];
        end
     end
     if isempty(sim{j,14}) == 0;
        switch sim{j,14}
            case {'2.7.2014'}
                sim{j,14} = '1.1.2014';
        end
     end
    if isempty(sim{j,26}) == 0;
        switch sim{j,26}
            case {'2.7.2014'}
                sim{j,26} = '1.1.2014';
        end
    end
    if isempty(sim{j,19}) == 0;
        switch sim{j,19}
            case 15
                sim{j,19} = 'DEPGATE-BUS-SHENGEN-8B';
            case 16
                sim{j,19} = 'DEPGATE-BUS-SHENGEN-8C';
            case 17
                sim{j,19} = 'DEPGATE-BUS-SCHENGEN-A';
            case {1,2,3,4,5,6,7,8,9,11,12}
                sim{j,19} = ['DEPGATE-CONTACT-
SHENGEN',num2str(sim{j,19})];
            case {10,14}
                sim{j,19} = ['DEPGATE-WALKOUT-
SHENGEN',num2str(sim{j,19})];
        end
    end
    if isempty(sim{j,20}) == 0;
        switch sim{j,20}
           case {1,2,3,4,5,6,7,8,9,10,11,12,14}
                sim{j,20} = ['STAND-',num2str(sim{j,20})];
            case {15}
                sim{j,20} = 'STAND-REMOTE-20';
            case {16}
                sim{j,20} = 'STAND-REMOTE-21';
```

```
            case {17}
                sim{j,20} = 'STAND-REMOTE-22';
        end
    end
end

name = strcat('Optimised_gap_Arcport-',num2str(date),'.txt'); %%making
the inptut date being the name of the new csv file
%cell2csv(name,sim,';',2007, '.'); %%writing out the csv file with the
adapted flight schedule
dlmcell(name,sim);
```

4. Splitting no-gate flights and writing out new data file for the optimization model

```
date = '06.07.2014';        %Examined date

%Reading in the solution file from Gurobi
resultFile = strcat(num2str(date),'.sol');
fid = fopen(resultFile);
formatSpec = 'y[%d,%d] %d';
Sol = textscan(fid, formatSpec,'delimiter','\n','CollectOutput',true,
'HeaderLines',2);
fclose(fid);

Sol=[Sol{:}];
Sol(any(Sol==0,2),:)=[]; %array with the results

%Reading in Flights document from Adapt_data_optimization program
filename = strcat('splitFlights-',num2str(date),'.csv');
%splitFlights- if split has occured, numberOfFlights- if first time
split
[num,txt,flights] = xlsread(filename);

%Adding allocated gates to the flights
flights{1,20} = 'Gates';

for i = 2:size(flights,1);
    for j = 1:size(Sol,1);
        if isequal(Sol(j,1),flights{i,1});
            flights{i,20} = Sol(j,2);
        end
    end
end

%if nogate split the flight and create dummy arr and dep
r=size(flights,1);
k=1;
for i=2:r;
    if  isempty(flights{i,20}) && isequal(flights{i,7},0)==0 &&
isequal(flights{i,15},0)==0;
        r=r+1;
        flights{r,1} = r-1;
        Connected{k,1}=flights{i,1};
        Connected{k,2} = flights{r,1};
        k=k+1;
        flights{r,2}=flights{i,10}-10;
        flights{r,3}=0;
        flights{r,4}=0;
```

```
        flights{r,5}=0;
        flights{r,6}=0;
        flights{r,7}=0;
        flights{r,8}=0;
        flights{r,9}=[];
        flights{r,10} = flights{i,10};
        flights{r,11} = flights{i,11};
        flights{r,12} = flights{i,12};
        flights{r,13} = flights{i,13};
        flights{r,14} = flights{i,14};
        flights{r,15} = flights{i,15};
        flights{r,16} = flights{i,16};
        flights{r,17} = flights{i,17};
        flights{r,19} = flights{i,19};
        flights{i,10} = flights{i,2}+10;
        flights{i,11} = 0;
        flights{i,12} = 0;
        flights{i,13} = 0;
        flights{i,14} = 0;
        flights{i,15} = 0;
        flights{i,16} = [];
        flights{i,17} = [];
        flights{i,19} = [];
    end
end

%make subsets of flights
%Nsch flights
k=1;
for i = (2:size(flights,1))
    if (isequal(flights{i,9},0)) || (isequal(flights{i,17},0));
        Nsch{k,1} = flights{i,1};
        k=k+1;
    end
end

%Canadian arrivals
k=1;
C = [];
for i = (2:size(flights,1))
    if isequal(flights{i,4},'YHZ')  || isequal(flights{i,4},'YYZ') ||
isequal(flights{i,4},'YEG') || isequal(flights{i,4},'YVR');
        C{k,1} = flights{i,1};
        k=k+1;
    end
end

%Heavy freight flights
k=1;
F = [];
for i = (2:size(flights,1))
    if ((flights{i,16} > 3000) | (flights{i,8} > 3000));
        F{k,1} = flights{i,1};
        k=k+1;
    end
end

%Large flights
k=1;
L=[];
for i = (2:size(flights,1))
```

```
    if isequal(flights{i,7},'TFFIX') || isequal(flights{i,15},'TFFIX');
        L{k,1} = flights{i,1};
        k=k+1;
    end
end

%Sensitive flights
k=1;
Sensitive=[];
for i = (2:size(flights,1))
     if isequal(flights{i,12},'LHR') || isequal(flights{i,12},'LGW')||
isequal(flights{i,12},'BWI')|| isequal(flights{i,12},'HEL') ||
isequal(flights{i,12},'JFK') || isequal(flights{i,12},'MUC') ||
isequal(flights{i,12},'FRA') || isequal(flights{i,12},'CDG') ||
isequal(flights{i,12},'YYZ') || isequal(flights{i,12},'IAD') ||
isequal(flights{i,12},'MSP');
         Sensitive{k,1} = flights{i,1};
         k=k+1;
     end
end

%%Data needed for optimization
B = 30;
bigM = 100000;
alpha1 = 0.5;          %walking distance
alpha2 = 100;          %min bus gates
alpha3 = 40;           %min walkin-walkout gates
alpha4 = -3000;        %min freight
alpha5 = -300;         %min sensitive
alpha6 = 10000;

%Creating two arrays with measured sch distances and non-sch distances
sch_Distances = {1,98.42,528.60,382.94,343.74;
                 2,132.40,538.60,349.06,308.40;
                 3,165.96,573.95,315.85,295.96;
                 4,196.88,603.29,284.76,274.03;
                 5,233.03,641.57,248.54,230.33;
                 6,264.00,670.76,217.74,214.37;
                 7,364.3,813.10,242.80,222.64;
                 8,412.50,816.23,291.07,225.77;
                 9,409.33,814.82,287.90,224.35;
                 10,406.83,814.00,285.40,223.54;
                 11,382.93,789.91,261.51,199.45;
                 12,401.65,814.40,280.23,223.93;
                 14,471.75,878.93,350.33,288.47;
                 15,416.45,827.86,295.02,237.39;
                 16,438.91,850.32,317.48,259.85;
                 17,177.70,553.37,496.40,457.19};
non_sch_Distances = {1,98.42,528.60,382.94,343.74;
                     2,132.40,538.60,349.06,308.40;
                     3,165.96,573.95,315.85,295.96;
                     4,196.88,603.29,284.76,274.03;
                     25,501.57,916.06,258.41,288.37;
                     26,495.10,849.20,251.93,280.45;
                     27,464.15,910.26,220.98,282.57;
                     28,493.82,940.96,250.66,313.27;
                     29,476.35,921.14,233.18,293.45;
                     30,459.93,906.65,216.76,278.96;
                     31,449.47,921.37,206.30,293.68;
                     32,465.36,932.34,222.20,304.65;
                     34,502.51,949.12,259.35,321.43;
```

```
                    15,440.08,886.94,196.92,259.25;
                    16,456.46,903.12,213.29,275.43;
                    17,177.70,553.37,496.40,457.19};

%creating new arry with walking distances for each flight depending on
if
%the flight is sch or non-sch.
k=1;
for i = 2:size(flights,1);
    for j = 1:size(sch_Distances,1);
        dist_per_flight{k,1} = flights{i,1};
        dist_per_flight{k,2} = sch_Distances{j,1};
        if flights{i,9}==1;      %arrival
            dist_per_flight{k,3} = sch_Distances{j,2};
            dist_per_flight{k,5} = sch_Distances{j,4};
        else
            dist_per_flight{k,3} = non_sch_Distances{j,2};
            dist_per_flight{k,5} = non_sch_Distances{j,4};
        end
        if flights{i,17} ==1; %departure
            dist_per_flight{k,4} = sch_Distances{j,3};
            dist_per_flight{k,6} = sch_Distances{j,5};
        else
            dist_per_flight{k,4} = non_sch_Distances{j,3};
            dist_per_flight{k,6} = non_sch_Distances{j,5};
        end
        k=k+1;
    end
end

name2 = strcat('Distances_per_flight2-',num2str(date),'.csv'); %printing
out array with walking distances
xlswrite(name2,dist_per_flight);


M = sch_Distances(:,1);

DA = dist_per_flight(:,[1,2,3]);
DD = dist_per_flight(:,[1,2,4]);
DT_arr = dist_per_flight(:,[1,2,5]);
DT_dep = dist_per_flight(:,[1,2,6]);


G1 = [5;6;7;8;9;11;12];             %Dual contact
G2 = [15;16];                       %Dual bus
G3 = 17;                            %Sch bus
G4 = [10;14];                       %walkin
G5 = [5;6;15;16];                   %Candian arrival
G6 = [4;7;12];                      %stands for Large aircrafts
G7 = [7;9;12];                      %Freight gates
G8 = [1;2;3;4;5;6;7;8;9;11;12];     %Sensitive

%Printing data out in text file for optimisation model
string1 = 'param B := ';
string2 = ';';
string3 = 'param bigM := ';
string4 = 'param alpha1 := ';
string5 = 'param alpha2 := ';
string6 = 'param alpha3 := ';
string7 = 'param alpha4 := ';
```

```
string8 = 'param alpha5 := ';
string9 = 'set N := ';
string10 = 'param TA := ';
string11 = 'param PA := ';
string12 = 'param PT_arr := ';
string13 = 'param TD := ';
string14 = 'param PD := ';
string15 = 'param PT_dep := ';
string16 = 'set M := ';
string17 = 'set Nsch := ';
string18 = 'set C := ';
string19 = 'set F := ';
string20 = 'set L := ';
string21 = 'set S := ';
string22 = 'param DA := ';
string23 = 'param DD := ';
string24 = 'param DT_arr := ';
string25 = 'param DT_dep := ';
string26 = 'set G1 := ';
string27 = 'set G2 := ';
string28 = 'set G3 := ';
string29 = 'set G4 := ';
string30 = 'set G5 := ';
string31 = 'set G6 := ';
string32 = 'set G7 := ';
string33 = 'set G8 := ';
string34 = 'end;';
string35 = 'param alpha6 := ';


N = flights(:,1);
TA = flights(:,[1,2]);
PA = flights(:,[1,5]);
PT_arr = flights(:,[1,6]);
TD = flights(:,[1,10]);
PD = flights(:,[1,13]);
PT_dep = flights(:,[1,14]);

fName = strcat('splitData-',num2str(date),'.dat');
fileID = fopen(fName,'w');
fprintf(fileID, '%s%d%s\n',string1,B,string2);
fprintf(fileID, '%s%d%s\n',string3,bigM,string2);
fprintf(fileID, '%s%d%s\n',string4,alpha1,string2);
fprintf(fileID, '%s%d%s\n',string5,alpha2,string2);
fprintf(fileID, '%s%d%s\n',string6,alpha3,string2);
fprintf(fileID, '%s%d%s\n',string7,alpha4,string2);
fprintf(fileID, '%s%d%s\n\n',string8,alpha5,string2);
fprintf(fileID, '%s%d%s\n\n',string35,alpha6,string2);
fprintf(fileID, '%s\n',string9);
for i = 2:size(N,1);
    fprintf(fileID, '%d\n',N{i,:});
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string10);
for i = 2:size(N,1);
    fprintf(fileID, '%d %d\n',TA{i,:});
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string11);
for i = 2:size(N,1);
    fprintf(fileID, '%d %d\n',PA{i,:});
```

```
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string12);
for i = 2:size(N,1);
    fprintf(fileID, '%d %d\n',PT_arr{i,:});
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string13);
for i = 2:size(N,1);
    fprintf(fileID, '%d %d\n',TD{i,:});
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string14);
for i = 2:size(N,1);
    fprintf(fileID, '%d %d\n',PD{i,:});
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string15);
for i = 2:size(N,1);
    fprintf(fileID, '%d %d\n',PT_dep{i,:});
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string16);
for i = 1:size(M,1);
    fprintf(fileID, '%d\n',M{i,:});
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string17);
for i = 1:size(Nsch,1);
    fprintf(fileID, '%d\n',Nsch{i,:});
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string18);
for i = 1:size(C,1);
    fprintf(fileID, '%d\n',C{i,:});
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string19);
for i = 1:size(F,1);
    fprintf(fileID, '%d\n',F{i,:});
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string20);
for i = 1:size(L,1);
    fprintf(fileID, '%d\n',L{i,:});
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string21);
for i = 1:size(Sensitive,1);
    fprintf(fileID, '%d\n',Sensitive{i,:});
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string22);
for i = 1:size(DA,1);
    fprintf(fileID, '%d %d %f\n',DA{i,:});
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string23);
for i = 1:size(DD,1);
    fprintf(fileID, '%d %d %f\n',DD{i,:});
end
```

```
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string24);
for i = 1:size(DT_arr,1);
    fprintf(fileID, '%d %d %f\n',DT_arr{i,:});
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string25);
for i = 1:size(DT_dep,1);
    fprintf(fileID, '%d %d %f\n',DT_dep{i,:});
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string26);
for i = 1:size(G1,1);
    fprintf(fileID, '%d\n',G1(i,:));
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string27);
for i = 1:size(G2,1);
    fprintf(fileID, '%d\n',G2(i,:));
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string28);
for i = 1:size(G3,1);
    fprintf(fileID, '%d\n',G3(i,:));
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string29);
for i = 1:size(G4,1);
    fprintf(fileID, '%d\n',G4(i,:));
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string30);
for i = 1:size(G5,1);
    fprintf(fileID, '%d\n',G5(i,:));
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string31);
for i = 1:size(G6,1);
    fprintf(fileID, '%d\n',G6(i,:));
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string32);
for i = 1:size(G7,1);
    fprintf(fileID, '%d\n',G7(i,:));
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n',string33);
for i = 1:size(G8,1);
    fprintf(fileID, '%d\n',G8(i,:));
end
fprintf(fileID, '%s\n\n',string2);
fprintf(fileID, '%s\n\n',string34);
fclose(fileID);

name = strcat('splitFlights2-',num2str(date),'.csv');
xlswrite(name,flights);
```

5. Measuring total walking distances

```
date = '06.07.2014';        %Desired date
```

```
%Reading in the flights with allocated gates
filename = strcat('Flights_WD-',num2str(date),'.csv');
[num,txt,flights] = xlsread(filename);

for i=2:size(flights,1);
    if isempty(flights{i,18}) == 0
        switch flights{i,18}
            case {25,26,27,28,29,30,31,32,34}
                flights{i,18} = (flights{i,18})-20;
            case {'8B', '25B','28B',
'32D',20,21,22,23,24,62,63,64,65,111}
                flights{i,18}= 15;
            case {'8C', '28C'}
                flights{i,18} = 16;
            case {'A'}
                flights{i,18} = 17;
        end
          switch flights{i,19}
            case {25,26,27,28,29,30,31,32,34}
                flights{i,19} = (flights{i,19})-20;
            case {'8B', '25B', '28B',
'32D',20,21,22,23,24,62,63,64,65,111}
                flights{i,19}= 15;
            case {'8C', '28C'}
                flights{i,19} = 16;
            case {'A'}
                flights{i,19} = 17;
          end
        end
end

%Reading in distances for each flight on each gate
filename = strcat('Distances_per_flight2-',num2str(date),'.csv');
[num,txt,gatesWD] = xlsread(filename);

OriginalWD={'N','Arrival','Departure','Transf-arr','Transf-dep'};
for i = 2:size(flights,1);
    OriginalWD{i,1}=flights{i,1};
    for j= 1:size(gatesWD);
        if isequal(flights{i,1},gatesWD{j,1}) &&
isequal(flights{i,18},gatesWD{j,2});
            OriginalWD{i,2}=flights{i,5}*gatesWD{j,3}; %Arrival
            OriginalWD{i,4}=flights{i,6}*gatesWD{j,5}; %Transfer arrival
        end
        if isequal(flights{i,1},gatesWD{j,1}) &&
isequal(flights{i,19},gatesWD{j,2});   %Departure
            OriginalWD{i,3}=flights{i,13}*gatesWD{j,4}; %Departure
            OriginalWD{i,5}=flights{i,14}*gatesWD{j,6}; %Transfer
departure
        end
    end
end

OptimisedWD={'N','Arrival','Departure','Transf-arr','Transf-dep'};
for i = 2:size(flights,1);
    OptimisedWD{i,1}=flights{i,1};
    for j= 1:size(gatesWD);
        if isequal(flights{i,1},gatesWD{j,1}) &&
isequal(flights{i,20},gatesWD{j,2});
```

```
            OptimisedWD{i,2}=flights{i,5}*gatesWD{j,3}; %Arrival
            OptimisedWD{i,4}=flights{i,6}*gatesWD{j,5}; %Transf. arrival
            OptimisedWD{i,3}=flights{i,13}*gatesWD{j,4}; %Departure
            OptimisedWD{i,5}=flights{i,14}*gatesWD{j,6}; %Transf.
departure
        end
    end
end
```

C. Compared results for walking distances – Low season

WD-measured distances (km)

	1.1.2015			2.1.2015			3.1.2015		
	Ori.	Opt.	Diff.	Ori.	Opt.	Diff.	Ori.	Opt.	Diff.
Arrival	784,14	691,25	92,89	919,16	936,26	-17,10	947,50	896,26	51,25
Departure	3128,84	2898,18	230,66	3579,83	3430,63	149,20	2773,78	2790,06	-16,28
Transfer-arr	182,65	169,55	13,10	323,03	320,54	2,49	369,14	352,08	17,07
Transfer-dep	172,26	176,12	-3,86	352,31	356,34	-4,03	384,03	381,77	2,26
Total	4267,899	3935,1	332,80	5174,33	5043,77	130,55	4474,45	4420,16	54,29
Diff. In %			7,8%			2,5%			1,2%

	4.1.2015			5.1.2015			6.1.2015		
	Ori.	Opt.	Diff.	Ori.	Opt.	Diff.	Ori.	Opt.	Diff.
Arrival	1021,50	956,71	64,79	1358,87	1062,52	296,35	955,29	861,69	93,61
Departure	3261,74	3142,16	119,58	3304,27	3264,15	40,12	3005,62	2952,68	52,95
Transfer-arr	440,24	437,77	2,47	411,94	407,90	4,04	406,39	389,99	16,40
Transfer-dep	486,24	480,88	5,36	438,87	434,49	4,38	423,28	417,70	5,58
Total	5209,72	5017,52	192,20	5513,95	5169,06	344,89	4790,59	4622,05	168,54
Diff. In %			3,7%			6,3%			3,5%

	7.1.2015		
Arrival	Ori.	Opt.	Diff.
Departure	696,38	653,18	43,20
Transfer-arr	2042,72	1946,13	96,59
Transfer-dep	312,26	305,14	7,12
Total	350,43	340,43	10,00
Diff. In %	3401,79	3244,88	156,91
			4,6%

WD-Measured in ARCport (km)

	1.1.2015			2.1.2015			3.1.2015		
	Ori.	Opt.	Diff.	Ori.	Opt.	Diff.	Ori.	Opt.	Diff.
Arr - Non-Sch.	484,69	466,44	18,25	942,50	963,26	-20,76	833,98	830,35	3,63
Arr - Sch.	362,01	287,622	74,39	460,99	479,51	-18,52	460,68	510,94	-50,26
Dep - Non-Sch.	1846,43	1841,389	5,04	2347,68	2331,60	16,08	1802,92	1810,31	-7,39
Dep - Sch.	967,62	764,568	203,06	1293,52	1175,12	118,40	1149,67	1205,30	-55,63
Total	3660,75	3360,02	300,73	5044,70	4949,50	95,20	4247,24	4356,89	-109,65
Diff. In %			8,2%			1,9%			-2,6%

	4.1.2015			5.1.2015			6.1.2015		
	Ori.	Opt.	Diff.	Ori.	Opt.	Diff.	Ori.	Opt.	Diff.
Arr - Non-Sch.	1223,076	1241,934	-18,858	1218,76	1187,10	31,66	1042,31	1037,70	4,61
Arr - Sch.	467,195	419,336	47,859	696,16	391,76	304,41	400,29	312,89	87,40
Dep - Non-Sch.	2140,868	2158,61	-17,742	2353,30	2346,83	6,47	2039,01	2030,62	8,39
Dep - Sch.	1404,783	1464,561	-59,778	1116,66	1102,84	13,82	1105,51	1125,70	-20,19
Total	5235,92	5284,44	-48,52	5384,87	5028,52	356,36	4587,13	4506,92	80,21
Diff. In %			-0,9%			6,6%			1,7%

	7.1.2015		
	Ori.	Opt.	Diff.
Arr - Non-Sch.	802,49	776,44	26,06
Arr - Sch.	347,59	278,38	69,21
Dep - Non-Sch.	1447,79	1437,86	9,93
Dep - Sch.	863,51	843,75	19,76
Total	3461,39	3336,43	124,96
Diff. In %			3,6%

D. Compared results for walking distances – High season

WD-measured distances (km)

	1.7.2014			2.7.2014			3.7.2014		
	Ori.	Opt.	Diff.	Ori.	Opt.	Diff.	Ori.	Opt.	Diff.
Arrival	2097,45	1566,43	531,02	1827,84	1603,46	224,38	2328,27	1891,44	436,83
Departure	3664,33	3352,20	312,14	4099,98	3878,36	221,63	4274,84	3989,63	285,21
Transfer-arr	709,51	690,17	19,35	743,87	721,25	22,62	717,66	709,51	8,15
Transfer-dep	797,48	797,78	-0,30	743,08	741,58	1,50	801,41	796,14	5,27
Total	7268,78	6406,57	862,20	7414,77	6944,64	470,13	8122,19	7386,73	735,46
Diff. In %			11,9%			6,3%			9,1%

	4.7.2014			5.7.2014			6.7.2014		
	Ori.	Opt.	Diff.	Ori.	Opt.	Diff.	Ori.	Opt.	Diff.
Arrival	1817,56	1653,47	164,09	1846,73	1416,94	429,79	2516,27	2177,04	339,23
Departure	4158,60	3853,29	305,31	3609,26	3372,10	237,16	4228,26	3961,97	266,28
Transfer-arr	683,02	668,52	14,50	695,75	721,60	-25,85	659,39	667,69	-8,30
Transfer-dep	771,69	783,81	-12,12	760,60	766,79	-6,19	710,01	719,94	-9,93
Total	7430,87	6959,09	471,78	6912,33	6277,42	634,91	8113,93	7526,64	587,28
Diff. In %			6,3%			9,2%			7,2%

	7.7.2014		
	Ori.	Opt.	Diff.
Arrival	1709,84	1445,09	264,76
Departure	4100,55	3898,88	201,67
Transfer-arr	628,69	629,20	-0,51
Transfer-dep	659,15	655,52	3,64
Total	7098,24	6628,69	469,55
Diff. In %			6,6%

WD-Measured in ARCport (km)

	1.7.2014			2.7.2014			3.7.2014		
	Ori.	Opt.	Diff.	Ori.	Opt.	Diff.	Ori.	Opt.	Diff.
Arr - Non-Sch.	1821,91	1774,685	47,22	1722,83	1678,28	44,56	1696,70	1692,97	3,73
Arr - Sch.	1322,97	1073,556	249,41	1213,26	1174,48	38,78	1466,31	1347,42	118,89
Dep - Non-Sch.	2785,48	2812,482	-27,01	2591,28	2601,77	-10,49	2868,74	2884,53	-15,78
Dep - Sch.	2003,05	1826,568	176,48	2412,93	2321,57	91,36	2115,66	2025,55	90,11
Total	7933,40	7487,29	446,11	7940,30	7776,09	164,20	8147,41	7950,46	196,95
Diff. In %			5,6%			2,1%			2,4%

	4.7.2014			5.7.2014			6.7.2014		
	Ori.	Opt.	Diff.	Ori.	Opt.	Diff.	Ori.	Opt.	Diff.
Arr - Non-Sch.	1645,309	1627,464	17,845	1251,68	1240,65	11,03	1871,42	1859,13	12,28
Arr - Sch.	1143,499	1244,893	-101,39	1233,79	1294,27	-60,48	1547,84	1437,28	110,56
Dep - Non-Sch.	2578,572	2561,102	17,47	2503,84	2503,84	0,00	2750,95	2754,70	-3,75
Dep - Sch.	2514,795	2299,395	215,4	2205,99	2060,22	145,77	2273,48	2150,31	123,17
Total	7882,18	7732,85	149,32	7195,30	7098,98	96,32	8443,68	8201,42	242,26
Diff. In %			1,9%			1,3%			2,9%

	7.7.2014		
	Ori.	Opt.	Diff.
Arr - Non-Sch.	1330,24	1327,29	2,94
Arr - Sch.	1302,57	1121,97	180,61
Dep - Non-Sch.	2606,91	2617,54	-10,63
Dep - Sch.	2239,21	2065,82	173,39
Total	7478,93	7132,61	346,32
Diff. In %			4,6%

School of Science and Engineering
Reykjavík University
Menntavegi 1
101 Reykjavík, Iceland
Tel. + 354 599 6200
Fax + 354 599 6201
www.reykjavikuniversity.is
ISSN 1670-8539