Modelling and Analysis of Real World Airport Gate Allocation Problem

Urszula Monika Neuman, MSc

Thesis submitted to The University of Nottingham
for the degree of Doctor of Philosophy

December 2014

Abstract

With airports becoming busier and often struggling with insufficient capacity, the efficiency of the airports resource utilisation becomes more and more important all over the world. The efficiency may be improved by integration of the airport operations which historically were handled in separation. More effective resource utilisation would not only smooth the airport operation but also should have a positive environmental impact. The gate allocation problem is one of the important airport operations, which is often solved without considering the links with other airport operations. Modelling and analysis of new constraints which allow the ground movement information to be taken into consideration in the allocation planning in advance as well as design of appropriate solution methods are discussed. It is observed that when the additional information from the ground movement is incorporated in the allocation planning process the number of expected routing conflicts, both around gates and on taxiways, drops. This should results in a smoother airport operation during the day of operation. Data from Manchester Airport is used in this thesis to build the model, as well as to test and to validate the solution methods.

Acknowledgements

I would like to thank Dr. Jason Atkin, my first supervisor, for his guidance, constant support and deep interest in my research during those four years of my study at the University of Nottingham. I would also like to thank Prof. Dr. Rong Qu, my second supervisor, for being actively involved in my work especially during my writing up period.

I would like to thank professionals who shared their knowledge and experience with me and provided data which I could use in experiments: Tim Walmsley, who introduced me to the gate allocation controllers at Manchester Airport, Garry Cookson, Airfield Liaison Manager at Manchester Airport, who let me spend two days with his team learning and observing their work, and Giovanni Russo who invited us to visit and observe the gate allocation controllers at Zurich Airport.

Last but not least, I would like to thank my husband Bartosz, my sisters and my parents for believing in me and supporting me through the entire time.

Contents

List of Figures

List of Tables

To my husband Bartosz and my parents.

Chapter 1

Introduction

1.1 General Motivation and Aims of the Project

The thesis was realised as a part of the "Integrating and automating airport operations" project. The project was supported by the Engineering and Physical Sciences Research Council (EPSRC) as well as by Manchester (MAN) and Zurich (ZRH) airports and involved students and researchers from four UK Universities: Nottingham, Lincoln, Stirling and Loughborough. There are several aspects of airport operations that were addressed in the project: the gate allocation problem, the arrival and departure sequencing problems, the ground movement problem, the baggage handling problem and the user interface issues associated with any decision support system.

The aspects of the gate allocation, the arrival and departure sequencing, the ground movement, the baggage handling are currently managed in separation from each other, which influences the effectiveness of overall airport operations. There is a place for improvement which can reduce airport congestion, delays and produced pollution. This improvement can positively affect finances of airports.

The general aim of the project is to bring the airport operations closer to each other. Integration of these aspects would not only improve the smoothness and effectiveness of the operations but would also have a positive environmental impact. Nowadays the environmental aspects play a very important role due to the constant increases in air traffic and the associated pollution.

1.2 Motivation and Aims of the Thesis

The motivation and aims of my part of the project are directly related to the general project assumptions. I look at the gate allocation problem (GAP) and the links between it and other airport operations and try to use them in the GAP model to improve the gate allocation planning process. The new allocation model includes more aspects and should therefore suggest solutions which are better for the overall airport operation. The links with the ground movement problem, which seem to have the strongest impact on the overall operation, are identified and the allocation planning is modified so that the ground movement becomes smoother during the day of operation. New constraints and a framework which enables cooperation of the ground movement system and the gate allocation system are designed. Additionally a decomposition method which utilises the receding horizon (RH) approach is suggested as an alternative, faster way of solving the problem.

1.3 Contributions of the Thesis

The crucial contributions of the thesis can be briefly summarised as flows:

- Chapter 4
  - A basic model of the GAP which includes a new approach to maximisation of the time gaps between allocations is presented.
  - The model sensitivity against the preferred time gap parameter settings is tested.
  - The RH approach is applied for the first time to decompose the GAP.
- Chapter 5
  - The method of detection of conflicts which may occur around the gate which uses groups of gates is presented.
  - A new constraint which resolves the conflicts around the gate is added to the model of the GAP.
- Chapter 6
  - A new dynamic implementation of the RH approach is proposed.
  - The performance of the static and the dynamic version of the Receding Horizon applied for the GAP is compared.

  - Best parameters settings of the RH method are proposed.

- Chapter 7

  - A framework which allows to integrate a ground movement model and a GAP model is described.
  - A seeding process which speeds the up the integration process is presented.
  - A new constraint which incorporates the feedback information from the ground movement in the GAP model is discussed.

1.4 Publications and Talks

The material presented in this thesis was used to prepare the following publications, the work was additionally presented at several conferences and meetings:

- "Integration Aspects of the Gate Allocation Problem", 3rd Student Conference on Operational Research (SCOR 2012), Nottingham, UK, 21.04.2012, talk
- "Receding Horizon Approach to the Gate Allocation Problem, Preliminary Study", OR54 Annual Conference, Edinburgh, UK, 06.09.2012, talk
- "Airport Gate Assignment: a Versatile Model of the Problem", SANDPIT: Integrating and Automating Airport Operations Project Meeting, Windsor, UK, 29.07.2013, talk
- "Airport Gate Assignment Considering Ground Movement", 4th International Conference on Computational Logistics (ICCL'13), Copenhagen, Denmark, 26.09.2013, talk
- Urszula M. Neuman, Jason A. D. Atkin Airport,"Gate Assignment Considering Ground Movement", in proceedings of the 4th International Conference on Computational Logistics (ICCL 2013), Copenhagen, Denmark, 25-27.09.2013, paper
- "Receding Horizon Approach to the Gate Allocation Problem, Advanced Study", LANCS Air Transportation Workshop, Nottingham, UK, 21.01.2014, talk
- Urszula M. Neuman, Jason A. D. Atkin Airport, Rong Qu "Analysis of Receding Horizon Approach to the Gate Allocation Problem with Gate Groups", paper submitted to Computers & Operation Research Journal, submitted

1.5 Structure of the Thesis

The thesis consists of eight chapters in total. The relevant literature is first presented in Chapter 2. The problem description and the real world datasets which were used in the experiments are discussed in Chapter 3. Chapter 4 focuses on the basic formulation of the GAP model and presents the preliminary study on the Receding Horizon application. An advanced model of the GAP, which includes a new constraint which aims to resolve conflicts at the gates is presented in Chapter 5. In depth analysis of the Receding Horizon approach to the GAP is discussed in Chapter 6. The semi-integrated model of the GAP and the ground movement, which utilises the feedback loop framework is introduced in Chapter 7. The final conclusion is provided in Chapter 8.

Chapter 2

Review of the Previous Approaches and Models

2.1 Introduction

There are five different literature areas upon which the presented thesis is based: models of the gate allocation problem (GAP), solution methods applied for the GAP, the receding horizon (RH) decomposition method, airport ground movement problem and simulation optimisation.

Section 2.2 is dedicated to mathematical models for GAP and provides an overview of the existing models in the literature. Section 2.3 focuses on the alternative modelling methods used for the GAP. The integration aspects of the GAP which were so far discussed in the literature are summarised in Section 2.4. This is an important context of this research since one of the research aims is to create an accurate model which includes integration aspects of the GAP. The created model can be solved using various methods. Some of the solution methods which have been proposed in the literature are discussed in Section 2.5.

A new solution method which uses the RH approach is proposed in this thesis. The RH approach decomposes a large problem into smaller sub-problems which are easier to solve. The solution of the large problem is created based upon the solutions obtained for the sub-problems. It has been previously applied in many areas but not to decompose the GAP. Section 2.6 introduces the approach and discusses some of its previous applications.

New integration aspects of the GAP model which are discussed in this thesis are related to the airport ground movement problem. New constraints are created based upon airport layout and feedback information from the ground movement system. Section 2.7 gives therefore a short overview of the existing ground movement algorithms.

The final part of the thesis describes a framework which allows a cooperation

FIGURE 2.1: An aircraft taxiing towards gates at Zurich Airport.

of the gate allocation algorithm with a routing algorithm. The framework works similar to simulation optimisation systems and hence Section 2.8 gives a brief overview of the simulation optimisation methods.

2.2 Mathematical Programming Models of GAP

A gate has been defined by Hamzawi [43] as follows: "...a gate position refers to the area of the terminal apron designated for the parking of an aircraft in order to load and unload passengers and to perform aircraft ground services such as refuelling, baggage handling, cleaning, servicing, etc." in one of the first papers on the problem. Figure 2.1 shows an example scene from an airport. Two aircraft are visible in the figure, the aircraft in the foreground has just arrived and is now taxiing towards the assigned gate. A second aircraft, visible in the background, is already on its gate. The gate allocation problem can be understood as a process of finding appropriate places to park aircraft at an airport. This is an optimisation problem and in order to solve it a good model is needed in the first place. The mathematical models which have been proposed in the literature are briefly discussed

in the following section.

2.2.1 Mathematical Programming

The mathematical programming model [9, 10, 83] involves a set of mathematical relations that correspond to relations existing in a real problem. The relations are defined as a set of constraints and an objective function that is optimized. Several sorts of mathematical programming models are described in the literature. If all relations defined in the model are linear then we deal with the linear programming model. Alternatively the non-linear programming model can be used (e.g. the quadratic model that has a quadratic objective function). The objective function and the constraint operate on variables. Example formulation 2.1 is a mathematical programming model which aims to optimise the objective function z.

$$\begin{aligned} &\textit{maximise:} && z = x_1 + x_2 \\ &\textit{subject to:} && x_1 - x_2 = 4 \\ & && x_2 \geq 3 \\ & && x_1 \geq 0 \end{aligned} \tag{2.1}$$

The input variables x_1 and x_2 are constrained by two constraints: x_1 must exceed x_2 by 4, and x_2 must be greater than or equal to 3. The maximum sum of x_1 and x_2 subject to the two constraints is the optimal solution of the model. Accordingly to the type of variables used in the model there exist integer programming (IP) modes which use integer variables, or mixed integer programming (MIP) models which use both continuous and integer variables. The various sorts of mathematical programming models correlate with the variety of problems that can be modelled using mathematical programming. The GAP is one of them.

2.2.2 Constraints and Objectives

All search methodologies operate in a search space. In order to create the search space a model of the real problem is usually constructed. The model mirrors the features of the real problem that may influence the solution. Most of the real world problems are to some degree uncertain, there are however modelling methods which allow to incorporate the uncertainty in a model in order to achieve accurate solutions. The amount of aspects of a problem (certain or uncertain) which is modelled translates to the size of the search space, more

complex models are the source of bigger and potentially more complicated search spaces. So, creation of an appropriate model (enough detail but not too complicated) is one of the hardest tasks when solving optimization problems.

The model construction is correlated with the methodology that has been chosen to solve the problem. There are approaches like mathematical programming or constraint programming [33] where model description is general and separable from the solving process. Separation of the model description process from the solving process enables using the same model description for more than one solver.

Mathematical programming is the most common modelling approach used for the GAP. It is also used in this thesis to model the problem and hence the overview of models starts with mathematical programming models. The following constraints and objectives have been discussed in the relevant literature.

Constraints

There are two core constraints which have to be included in every model of GAP:

- No two flights should be allocated to the same gate at the same time
- A flight should be allocate to at most one gate

Other constraints can be included in the model. The decision about adding them depends mostly upon the specific airport preferences and the aspects of the problem the designers focus on. The list below includes constraints discussed in the literature, some of them are included in the models used in this thesis which are described in Chapter 4 and Chapter 5.

- The time gap constraint, which introduces a variable for each pair of flights which follow each other on the same gate. The variable is used in the objective function to maximize the time gaps between allocations. Maximisation of the time gaps between allocations is an important aspect of the problem as it allows small delays on gates to be absorbed. More robust solutions are obtained when the constraint is included in the model. This constraint has been discussed in [7, 69] and is included in the model presented in Chapter 5.
- Fixed minimum buffer time between two allocations [55, 57, 69], which is also included in the model from Chapter 5. It should be added to the model as it gives the departing

aircraft time to clear the gate. Fixed minimum buffer times are usually used in real world planning.

- Push back constraints concern allocations which may lead to push back conflicts in the area around the gates. In [57] a fixed buffer time is introduced which provides enough time for flights to push back from gates. This is similar to the idea of fixed minimum buffer times between allocations but is introduced only between the allocations which may lead to push back conflicts. The constraint is not used in the model presented in Chapter 5 as the fixed minimum buffer times are.

- The shadowing constraint concerns gates which cannot be used simultaneously [26, 57, 69, 70], for example usage of one gate may block usage of another. The shadow constraints often occur in real world problems. They are included in the model from Chapter 5 as they exist in the instances which are solved in the experiments.

- Constraints which introduce various features of gates, for example sizes [21, 22, 26, 27, 69]. This is an important constraint which usually occurs in the real world. It is included in the advanced model used in this research (see Chapter 5).

- The towing constraint which concerns flights which are staying long at an airport and are towed away from the stands. It is often modelled by splitting the flights into three (or two) parts: arrival, departure and parking activity (or just arrival and departure). It has been discussed in [21, 22, 26, 27, 57, 69]. The towing procedure is not included in the model presented in Chapter 5 as relatively few tows occurred at the airport the model was designed for.

- The passenger walking distance constraint, which introduces a variable for each pair of allocations. The variable becomes one if the two allocations appear and is weighted and minimised in the objective function. This constraint was very popular among early stage researchers [16, 23, 42, 84, 63], when the walking distance seemed to be a crucial aspect of the problem. It is a bit less important nowadays when passengers are welcome to stay in the commercial areas of the airport as long as possible and pass it more often, as it improves the sale rates of the airports. This constraint is therefore neglected in the model from Chapter 5.

- The time window constraint, which defines the time duration during which an aircraft

may stay at an airport [63]. The time window is longer than the time an aircraft actually stays at an airport and provides a margin for possible allocation delays. The sum of delays is minimised in the objective function. This idea is similar to the time gap constraint as it aims to absorb the delays on gate. It assumes however that the aircraft leave the airport after the time window is ended which may not always be the case. The time gap constraint rather than the time window constraint was therefore used in the thesis.

- The preceding constraint indicates that a flight can have at most one following flight at the same gate [63, 84]. This is a very interesting idea, which simplifies the model formulation. It is not included in the model presented in Chapter 5, but may be an interesting future work direction.

Objectives & Objective Function

Several objectives are often used in the GAP models and the objective function is usually a weighted sum of them. Weights are set by the model designers to find a solution that reflex the user preferences. There are also papers which provide a comparison of results obtained for various weights in the objective function [85] or investigate the Pareto front of a multi objective formulation of the problem [30]. A list of objectives found in the literature is presented below, several of them are used in the model presented in Chapter 5.

- Minimisation of the absolute deviation from the originally planned schedule [16, 26, 28, 30, 70]. In [28] it is suggested that there may be more than one planned schedule for a flight, for example a flight used to use one gate but lately it started to use another one. It makes the objective similar to the maximisation of airline preferences objective which is discussed in [69] where it has been observed which gate has been used in the past by which airlines and how frequently. Based upon this the allocations which match the observed preferences are prised in the objective function. The model presented in Chapter 5 includes airline preferences in the objective function implemented like in [69].

- Maximisation of the flight to gate preferences [26, 27, 28, 30, 52, 69, 70], for example the size preferences are taken into account, sizes of gates should match sizes of flights as it results in more robust schedules. This objective is included in the advanced

model presented in Chapter 5, it is an important objective which often occurs in the real world.

- Maximisation of the idle times (the time gaps between allocations) on gates [7, 21, 22, 27, 28, 69]. In [7, 25] the variance of idle times at the gates is minimised while in [21, 22] a weighted sum of the gap durations is maximised. Although the implementations are slightly different the aim and the final effect is similar. The weighted sum of the time gaps is also used in [69] where only the gaps which are smaller than a preferred value are considered in the objective value. The models presented in the thesis use the idea from [69].

- Minimisation of the time an aircraft has to wait for a gate. This constraint relates to the situation when there are not enough gates at an airport or they are inappropriately allocated and therefore some arriving aircraft may be delayed because they have to wait for gates to become free [16, 48, 63]. This objective should be included when very busy airports are considered. The problem occurs rarely for the instances used in the thesis, there are usually enough gates available, and therefore the objective is not considered in this thesis.

- Minimisation of the number of ungated flights [16, 21, 23, 24, 28, 30, 70, 69]. The flights which are not assigned to the terminal gates (ungated flights) have to use the remote stands, which are not preferred by airlines as additional service has to be provided in order to get the passengers from an aircraft to a terminal. This objective is used in the models of the thesis, allocating flights to remote stands is strongly penalised in the used objective functions.

- Minimisation of the number of conflicts between flights within areas around gates [54]. The model presented in [14, 23, 42, 84] is used, however the variables set for pairs of allocations are weighted using a conflict likelihood function not the passengers walking distance. The idea of resolving conflicts in a part of the terminal is used in Chapter 5. It is however formulated differently as the architecture of the airport which is used in the experiments is very different from the one discussed in [54].

- Minimisation of baggage handling distance [14, 48]. It is done analogously to the passenger walking distance minimisation. The baggage handling problem is not in the scope of this thesis.

- Minimisation of the total passenger walking distance. This objective is not considered in the model used in the thesis for the same reason for which the walking distances constraint is not included. Airports strongly prefer to have passengers wandering around the shopping areas for longer as it has a positive impact on their finances. [23, 30, 66] describe the total walking distance as a sum of arriving, departing and transferring passenger walking distances. In [66] the walking distances of transferring passengers is calculated using a uniform probability distribution of all walking distances, with assumption that for a transferring passenger it is equally likely to board a flight on any gate. [14, 23, 42, 84] use the walking distance constraint to set up a variable for each pair of allocations and then minimise a weighted sum of all these variables in the objective function. In [25] the same idea is used but the objective function is not linearised, the objective function contains weighted sum of multiplications of pairs of allocation variables. In [16] aircraft which carry more passengers are assumed to use gates which are closer to the check-in or claiming areas. [85] introduces a constant for each allocation which symbolises the total walking distance for passengers. The constant is added to the objective function if the allocation appears in the solution. [65] uses the objective in the re-assignment problem, the walking distance of transferring passengers is minimised for the reassigned flights. The minimisation of the walking distance in the context of re-assignment and only transferring passenger is more reasonable than in the case of assignment problem. The time for changing flights may sometimes be very tight and a bad re-assignment plan may cause serious inconveniences for passengers, including missing a transfer connection.

- Minimisation of the total passenger waiting time [55, 85]. In [55] the total time that passengers spend at an airport is minimised. There are two activities that take passengers' time at an airport. One is moving around a terminal in order to get from/to the right gate, the other is the time spend on an aircraft when it is manoeuvring at the ground. The authors introduce two types of weight coefficients which are used in the objective function: linear and quadratic. The linear coefficients are used to weight the terminal times of departing and arriving passengers in the objective function. The quadratic coefficients are used to weight the terminal times of transferring passengers and the conflicts at the gates in the objective function. The idea of resolving conflicts is discussed in the thesis as it has a strong impact on the overall airport operation. It

is observed that only the conflicts at the gates but also the conflicts which occur on taxiways can be taken into consideration in the objective function. This is discussed in more depth in Chapter 7.

- Minimisation of the number of towing operations [26, 27, 28, 52]. Every towing operation is an additional movement at an airport which can cause conflicts. Hence it is highly recommended to reduce the number of towing operations, especially during rush hours. In [26, 27, 28, 30, 70] each flight is modelled as three separate activities: arrival, parking and departure. Each of the activities can be allocated to a different gate but the objective function aims to minimise such allocations. The towing procedure is not considered in the thesis as the problem seemed to be marginal for the instances solved in the experiments. It should however be included in models used to solve more busy airports problems.

- Maximisation of gates occupation time [35]. This objective is in contradiction with the discussed by other researchers maximisation of idle times robustness objective and is not considered in the thesis. It results in putting as many flights as possible on each gate but apparently it is in favour of the airport for which the objective was designed.

Based upon the above review and according to Manchester Airport preferences models of GAP which are described in Chapter 4 and in Chapter 5, and further extended in Chapter 7 were created. Mathematical programming formulations were used to define the constraints and the objectives. The used objective functions are weighted sums of several elements similarly to most approaches presented in the literature.

2.3 Alternative Models of the GAP

The mathematical programming models are used in this thesis to describe the GAP. Not all researchers use these models to describe the problem, models for genetic algorithms, expert systems as well as hybrid models have been proposed in the literature. Although the alternative models of the GAP are not in the scope of the thesis they are briefly described below to provide an appropriate background of the research.

2.3.1 Simulation Approach to GAP

One of the first papers on the GAP [43] described an interactive simulation program which simulates arrivals and departures and assigns gates. Although the program did not include the mathematical programming formulation of the problem, it discussed many important constraints which were later formally described by other researchers. It namely included aircraft-gate size constraint, airline preferences, minimisation of walking distance and passenger delays, buffer times between flights allocated to the same gate, maximum delay which is allowed for a flight on gate, maximal gate occupancy time.

2.3.2 Fuzzy Logic Application

Fuzzy logic [81] is based upon degrees of truth rather than on the traditional true or false logic. The degrees of truth are assigned to uncertain data in order to obtain the most accurate reasoning.

Fuzzy logic was applied in [25] to model the uncertainty of the GAP. In the model the idle times between allocations were fuzzy. One of the objectives was to maximise the idle times which are weighted in the objective function using membership grades calculated with an adjustment function. Relation between the shape of the adjustment function and the obtained results was observed and it was suggested that the shape should be chosen according to the preferences of a particular airport. The genetic algorithm (GA) was applied to solve the fuzzy model of the GAP. Introducing both the fuzzy logic and using the GA was innovative and gave promising results for the simple GAP model.

2.3.3 Expert Systems for the GAP

An expert system [38] is a computer system which imitates a decision process of an expert. Expert systems have a very different principle of working than other method discussed here. The systems don't use mathematical models, neither objective function nor constraints are defined for them. Instead typical expert systems use a knowledge base in the reasoning process. The knowledge base is a set of if-then rules which are based upon a human expert experience.

Expert systems were used to solve the GAP in [39, 52]. In [16] a hybrid of a linear programming (LP) model and an expert system was presented. The LP model contained only the core constraints of the GAP. The basic solution obtained using the simple LP

model was modified using the rules from the expert system.

2.3.4 The Gate Re-assignment Problem

The research presented in the thesis focuses on the planning of allocations in advance. However the plan which is created in advance often has to be re-planned during the day of operation due to unexpected situations, e.g. delays, emergencies, bad weather. The problem which tackles the changes on-line, during the day of operation is called the re-assignment problem. Although the re-assignment problem is not in the scope of this thesis several papers are mentioned in here due to a close relevance.

The re-assignment problem is usually solved on-line and therefore requires models and solution methods which are quick to solve.

Genetic Algorithms and the Re-assignment Problem

Genetic algorithm (GA) [47] is an adaptive metaheuristic (see Section 2.5.2) search algorithm based upon natural selection. The algorithm utilises a population of individuals which is modified using specific operators in each iteration. Each individual has a fitness value which defines how good it is in terms of an objective function optimised by the algorithm. Individuals who have a better fitness value are slightly more likely to survive and become a member of the next population in the next iteration of the algorithm. The population tends to improve slowly when the number of iterations grows. The key modelling challenges in a genetic algorithm are design of an individual, definition of the basic operators (typically mutation and crossover) which are performed on the individuals, and definition of the rules which decide if an individual is feasible.

In [41] a genetic algorithm was proposed to solve the re-assignment problem, where one individual included one possible way of allocating the flights which had to be re-assigned. The objective was to re-assign the flights influenced by a delay so that an extra delay was minimised.

The work presented in [41] was extended in [48] where a genetic algorithm with uniform crossover which never results in infeasible solutions was presented.

MIP Model of the Re-assignment Problem

In [82] a MIP model of the re-assignment problem was presented. The objective of the model was to reduce absolute deviation from the originally planned schedule. The model was solved exactly, on-line whenever there was a need of re-assigning a flight (only flights which were affected by a delay were considered in the re-assignment so the problem was usually easier to solve than the full GAP). Good calculation times and improvement in comparison to re-assignment performed manually were reported.

A similar approach was presented in [65] where a MIP model with the objective to reduce the walking distance of transferring passengers after re-assignment was described. The model was solved exactly only for the flights which are influenced by occurring delays.

2.4 Integration Aspects of the Models

One of the main focuses of this thesis is the integration of the GAP with the ground movement problem. The work which has been previously done on modelling various integration aspects of the GAP is summarised here.

2.4.1 Integration with Ground Movement

In [54] elements of the ground movement problem were added to the GAP model. The aim was to minimise the number of conflicts which occur within a ramp. In order to validate the allocation results a routing simulation model was designed. It is used to test which of the identified conflicts should be penalised stronger in the allocation model. The quality of allocations obtained using the MIP model was compared against the results obtained using a tailored heuristic. To our knowledge the approach is the first attempt to incorporate elements from the ground movement problem into the gate allocation model. The presented results concern only simple cases and more extensive tests should probably be performed, however, the paper makes a very important first step towards the integration of the gate allocation problem with the taxi routing problem. The above research was extended in [55] by adding a second objective to the objective function. The new objective function aimed not only to minimise the number of conflicts within a ramp but also to minimise the total time passengers spend in a terminal. Similarly to previous models the arriving, departing and transferring passengers were considered. In the formulation the fact that passengers use an underground system in order to commute from one part of the

airport to the other was also considered in the total time which is a new concept. In this thesis (see Chapter 5) the idea of minimising the number of conflicts within ramps is applied to a different architecture of an airport, groups of gates are used instead of ramps. The way the conflicts are minimised in this thesis is different as one global constraint is introduced instead of introducing a constraint for each conflicting allocation.

A simulation system which allowed conflicts occurring in the area around gates to be analysed has been published by Cheng [17]. In the work the gate allocation plan was assumed to be fixed and the number of conflicts was reduced by postponing the operations which may cause conflicts. Kim et al. [53] continued the work from [17] and designed a simulation optimisation system which aimed to test the impact of the robust gate allocation on the number of gate conflicts. A queuing model was used to simulate the departure process and validate the current gate allocation and the new robust gate allocation. Results obtained using various solution methods (exact and heuristic) which took the expected conflict into account were compared using a simulation model. The number of conflicts dropped after the robust gate allocation was applied. The framework presented in Chapter 7 uses also a routing system to validate the allocation plans. The allocation plan is gradually improved while the allocation system works in a feedback loop with the routing system.

2.4.2 Integration with Bus Planning

In [21, 22] the GAP was integrated with the bus planning problem. Column generation [6, 64] was used to model both of the problems initially and then the two models were combined and solved as one large problem. Column generation is a method of solving linear programming models which allows an optimal solution to be found without generating all model variables a priori. It generates only the variables which can potentially improve the value of the objective function which may significantly improve the efficiency of the search process. Despite using column generation (in [21, 22]) finding the exact solution for the GAP combined with the bus planning problem was found to be very hard. The method needed to be additionally tuned in order to solve the combined problem within a reasonable time.

Although integration of the GAP with bus planning is not in the scope of this thesis the application of the column generation method to solve a large integrated problem is a very interesting idea. Column generation could potentially be used to solve an integrated

problem which includes both the GAP and the ground movement problem (instead of the bus planning problem). Application of column generation for such problem may however be very challenging.

2.5 Solution Methods

All possible solutions to a problem exist within a search space. The search space is bounded by the constraints and it can be explored by a search methodology in order to find a good solution of a problem. The search methodology can either be an exact search or a heuristic.

2.5.1 Exact Methods

Exact enumeration methods consider every possible solution and return the best one. The solution obtained using an exact enumeration is truly the best existing one. But if the solved problem is a real world problem the search space can be large and an exact enumeration would require require extremely long search times. Advanced exact algorithms work by eliminating much of the search space from consideration, allowing an exact solution to be obtained more quickly. They usually take advantage of the structure of the problem to do so, so may not always perform well. The methods successively divide the search space in order to exclude those parts of the space that cannot contain a better solution than the current one.

Branch and bound (B & B) [68] and branch and cut algorithms [72] are examples of that approach. In B & B all the optional solutions of the problem form a tree like graph. While the algorithm searches though the branches of the tree it improves the best known solution. The branches can be discarded (pruned) from the search tree if they cannot produce a better solution than the best solution which was found so far. The branch and cut algorithm uses the branch and bound tree search adding to it a cutting planes idea which improves the search process. It is used in most of the commercially available solvers [83]. For example the commercial solver IBM ILOG CPLEX, which is used in the experiments for this thesis, employs the branch and cut technique.

The technique is very powerful but it struggles with symmetries [78]. Symmetries occur in a search tree when more than one configuration of variables leads to the same objective value, which creates equivalent branches of the search tree. All of the equivalent branches must be kept in the search tree and evaluated in the search process as long as there

is no pruning rule which allows discarding them. Hence, the symmetries may have a very negative impact on the performance of the algorithm if the needed pruning rule is found late in the search process. The symmetries may very likely occur in job scheduling problems which aim to schedule a set of identical jobs on several identical machines [50, 51, 71]. They can be broken by adding additional constraints or objectives to the formulation [50, 51, 61, 62, 71]. The GAP can be seen as a special case of the job scheduling problem, where the gates are the machines and the flights are the jobs to be done, its formulation can also have symmetries. An example of symmetry would be two identical gates to which the same configuration of flights can be assigned along the day with the same cost. Symmetry breaking constraints have therefore been designed and implemented in the used in this thesis model formulation. They are discussed in more depth in Chapter 5.

Another exact search methodology that is mentioned further in the section is dynamic programming [10, 19, 29]. It is commonly used for problems that can be broken down into simpler sub-problems. An algorithm that uses a dynamic programming approach solves the problem in stages, working with all possibilities that occur at a particular stage. It is often used in solving problems that require making a sequence of decisions over time.

2.5.2 Heuristics and Metaheuristics

A heuristic is a solution method which is able to find a good feasible solution but not necessarily an optimal solution for the problem that is solved. Heuristic methods are usually rather simple algorithms designed to find a satisfactory solution often within a limited time. A greedy algorithm [11] is an example of a heuristic. The algorithm always chooses the best solution which is available at a current stage of the search. The choice may be dependent upon previous choices but no future choices are considered. Although the algorithm finds the local optimum for each stage, there is no guarantee that the final solution will be optimal.

A metaheuristic [11] is a general solution method which guides subordinate heuristics. Local (neighbourhood) search heuristics are possibly the most commonly known metaheuristics. They consider the neighbourhood of a current solution to look for a better solution. The neighbourhood is a set of alternative solutions which are chosen from the search space according to a specified algorithm.

A hill climbing search algorithm [11, 47] is an example of a simple local search

heuristic. The algorithm starts from an arbitrarily chosen solution and looks for an alternative better solution within its neighbourhood. The search continues until no better solution is found in the neighbourhood of the current solution. The hill climbing algorithm tends to get stuck in local optima. Tabu search [36, 47] and simulated annealing [1, 47] are slightly more advanced local search metaheuristics. They include mechanisms which help the search to escape from the local optimum. Although the local search heuristics are often not very complicated algorithms they can obtain good results even for relatively hard problems, hence they are widely used.

Other popular metaheuristics are genetic algorithms (see Section 2.3.4). They use a set (population) of solutions, which is modified in each step of the search process, rather than a single solution.

All of the above mentioned methods were introduced several years ago. Nowadays the methods are very often combined or mixed in order to improve the efficiency of the traditional methods. New methods are still being created which are very often highly specialised and designed specially to solve a concrete problem. One possible remedy for highly specialised and usually expensive-to-implement systems are hyper-heuristics [12]. These aim to be a general, easy-to-use tool that can produce good solutions for a range of related problems. The search space of hyper-heuristics consists of heuristics rather than solutions.

2.5.3 Exact and Heuristic Methods for GAP

Exact solution methods are often used to solve small instances of the GAP even by researchers which suggest alternative solution methods to solve larger instances of the problem. The exact methods are slower but are very useful when a reference optimal solution is needed to validate results obtained using other methods. Moreover the exact methods are often used when authors wish to demonstrate the complexity of the problem formulation and to justify application of heuristic methods. This approach is also adopted in the thesis where the simple instances are solved exactly and the harder ones are solved using a receding horizon heuristic method which is explained in Section 2.6.

Mangoubi and Mathaisel [66] suggested using B & B algorithm as well as a simple heuristic algorithm which aim to assign flights which have the biggest number of passengers on board to gates which correspond to the shortest walking distances. They suggested also

that the heuristic can be used to produce an initial solution for the exact algorithm when the optimal solution is required. Hanghani and Chen [42] proposed also a heuristic approach to solve their formulation of the problem, they provided comparison with the optimal solution obtained using CPLEX solver for small test instances. Yan et al. [86] used both the exact method and greedy heuristics (see Section 2.5.2). They investigate how adding stochastic delays to the problem influences the results of the algorithms. Although these heuristics can give a good solution to the problem they were not considered in the thesis as they may tend to get stuck in local optima.

Chang [14] formulated the GAP as a Quadratic Assignment Problem to minimise the walking distances and baggage handling distances. The problem was solved exactly, several heuristic approaches were also proposed, from which the simulated annealing performed the best. Xu and Bayley [84] used mainly a tabu search (see Section 2.5.2) heuristic to solve their GAP model formulation. However in the experiments they provided a comparison with the optimal solutions obtained for the relatively small instances of the GAP which they used. Ding et al. [23, 24] provided a continuation of the work of Xu and Bayley [84]. They used a very similar model and solved it with a more advanced tabu search algorithm, simulated annealing as well as a hybrid of the two algorithms. They compared the results with the optimal solutions. They obtained good calculation times and can solve large problems which have been generated randomly. They showed that the hybrid algorithm performs best. The work was further developed by Lim et al. [63] described a tabu search and a memetic algorithm which applies two of the neighbourhood moves proposed in [23, 24, 84] and suggested an improvement which uses an idea of time windows. Kim et al. [55] referred to all the previous successful implementations of tabu search and also used it to solve their model formulation. The results are compared with solutions obtained using an exact branch and bound algorithm which was allowed to run for a reasonable period of time. They used a simple, parallel ramp configuration and randomly generated data regarding the flight schedule and the passenger flow in order to test how the proposed tabu search performs. The tabu search and the simulated annealing are quite powerful metaheuristics and they were seriously considered as a research direction. The receding horizon (RH) method (see Section 2.6) was finally chosen and described in the thesis for two reasons. The RH method uses the mathematical model which was anyway created to obtain the exact solution, while application of metahuristiscs requires defining new elements like for example neighbourhood moves. Moreover the application of the RH method to the GAP is a new

idea which gives potentially more new opportunities than another application of one of the metaheuristics.

Bolat [7] solved smaller instances of the problem exactly using a branch and bound algorithm, for larger instances he suggests several heuristics. The first is a constructive algorithm which uses a preference function in order to decide which gate to choose for every next flight arriving at an airport. The second one utilises the branch and bound algorithm but restricts the number of nodes to be checked in the search and in that way reduces the time and memory needed to finish the search. Dorndorf at al. [26] used a truncated branch and bound algorithm which is different from the typical branch and bound (see Section 2.5.1) as it uses two types of branching and a constraint propagation technique. None of the improved branch and bound algorithms were tested in the thesis as CPLEX was used in the experiments. It is believed to be a very advanced search engine which should be able to outperform the suggested branch and bound algorithms. Validation of this assumption is a possible future work direction.

Yan and Huo [85] are the first authors to propose the use of column generation to solve the GAP. In contrast to most of the GAP models (including the models used in this thesis) in their model the allocation variables are independent, i.e. allocating a flight to a gate does not influence other allocations. This allows variables to be separated and the column generation method to be easily applied. Their column generation approach used less than 10% of all of the possible variables in order to get the optimal solution. This speeded up the calculations a lot. They provided sensitivity analysis of the model not only regarding the weights in the objective function but also other input data of their model, like number of available gates, ground service time and buffer time between sequencing allocations.

Heuristic Methods for NP-hard GAP

Some of the GAPs are proven to be NP hard [32]. This means they are very hard to solve as they are harder than (or as hard as) the hardest problem in the class of NP (non-deterministic polynomial time) problems. NP problems are those for which it is possible to validate a given solution within a polynomial time. The proof of NP-hardness is a good enough justification to apply heuristic methods. This approach could not be used in the thesis as the formulation used wasn't proved to be NP-hard.

Dorndorf at al. [27] applied a heuristic called ejection chain algorithm to solve the

GAP modelled as a clique partitioning problem (CPP). This way of modelling allowed for an easy implementation of the robustness goal (keeping a time margin between allocations). Since CPP is an NP-hard problem no comparison with the optimal solution was provided.

In [28] several periods (days) were solved sequentially using an iterative heuristic algorithm. This was the first attempt (presented in the literature) to solve several consecutive days in one go so that the solution obtained for the previous day is considered when the current day is solved. The results indicate that using the previous day allocation as an initial solution for the next day's improves the performance of the heuristic. No comparison with exact method was provided as the used model formulation is NP-hard and there is no way to reach an exact solution in a reasonable time for the used size of instances.

Heuristic Methods

Finally there is a group of papers which suggest heuristics without either referring to the exact solutions or proving the complexity of the problem.

Expert systems are quite commonly used to solve the GAP ([39, 52]) and are very different from exact methods. It is hard to create an equivalent mathematical model which would provide the optimal solutions to which the authors could compare their results. Many, very specific rules are usually included in the knowledge base of an expert system. All of them would have to be transformed into appropriate constraints and objectives. Hence the comparison is usually not provided in the publications.

An interesting combined approach utilising an expert system and an exact method is presented in [16]. An LP model with core constraints was first solved to optimality. The solution was then modified using rules from an expert system.

Drexel and Nikulin [30] used a simulated annealing approach to produce a representative approximation of a Pareto front of a multi-objective formulation of the GAP. They used three objectives and each feasible solution was defined by a three element vector containing the values of each of the three objectives. The authors described the way the vectors are compared in order to get a set of non-dominated solutions (Pareto set) and how to apply the simulated annealing algorithm on the defined problem. In [70] the research from [30] was extended by modelling the uncertain arrival and departure times using fuzzy logic. The uncertainty was modelled in previous publications but the fuzzy logic had never been used before. The papers [30, 70] have a very theoretical meaning since the generated

approximation of a Pareto front is rather hard to interpret for a controller. In order to get a single solution (single schedule) based upon the produced Pareto set, what is usually expected in practical applications, an additional post-processing of the set will have to be added. The post-processing stage was just mentioned in the conclusions but not implemented or tested.

[35] presents a resource management system in which an innovative solution method related to a Big Bang Big Crunch (BB-BC) algorithm is applied to solve the GAP model. BB-BC is a population based algorithm. The algorithm starts with creating an initial population randomly. Next a Big Crunch converge operator is applied. One output, called center of mass, is created as a result of the Big Crunch. In the next stage, called the Big Bang stage, a new population is created. Members of the new population are uniformly distributed around the center of mass. The algorithm performs the Big Crunch and the Big Bang stages until it reaches a stopping condition (for example a requested number of iterations). The algorithm presented in this paper, called Single Leap BB-BC, refers to the original algorithm but it differs from it. It works with only one solution rather than with a population of solutions. The solution is modified iteratively by the algorithm. Operators which are used to modify the solution are referred to the ideas of center of mass and the explosion strength but they vaguely remain the original ideas. An initial solution required by the Single Leap BB-BC was produced using a greedy heuristic approach based upon sorting flights and gates in a particular manner and then assigning flights one after another. The resource management system apparently used also a rule base and an optimisation engine to obtain the final solution, but no specific information regarding this part of the system was provided, which makes the research hard to repeat. The system is fast and has been used in both the off-line planning stage and the on-line operational stage to solve real world instances of the problem.

2.5.4 Datasets

The presented publications use various kinds of datasets. Some authors randomly generated the input datasets [7, 23, 24, 35, 63, 84] while others created them using real datasets [22, 27, 85]. Some of the authors considered only a part of a real dataset (e.g. deal only with delayed flights [65, 82] or with flights and gates at one terminal [69]) when they believed it is reasonable to do so. Smaller datasets were usually easier to solve.

2.6 Receding (Rolling) Horizon

The receding horizon (RH) is a well known method which is often used in control systems when the controlled process changes dynamically. In this type of problems events are known in the future but the time horizon is limited. The solution method is adapted to the limited horizon and attempts to solve the problem using only the known information. Moreover the future events may depend upon the decision made for the current moment.

2.6.1 Common On-line Applications

A survey article by Mayne et al. [67] gives an overview of RH applications in various control systems. The RH method is rather on the slow side of the on-line methods. When a good solution is required a long enough horizon should be used which may result in calculation times which are relatively slow (for an on-line system). That is why it should not be applied in the problems when decisions have to be made immediately. A very good place to apply this type of control is the oil and petrochemical industry where the system dynamics is rather slow. The stability and robustness of the method are also discussed in the paper. The importance of the used parameters is pointed out, it is shown that short horizon may have fatal consequences for the stability of the method.

2.6.2 Hybrid Systems

A more recent survey paper by Lee [58] gives an overview of the applications of this method and discusses how the method has developed since it first occurred in the literature. A part of the paper refers to the newest history of the method which is particularly interesting for the research presented in this thesis. Hybrid systems are described where mixed integer linear programming models have been solved on-line using the receding horizon method. This is the way the receding horizon is used in our research.

2.6.3 Applications in Other Research Fields

Successful applications of the RH method in the oil and petrochemical industry control systems woke up an interest of researchers from other fields. They have applied the method to different on-line problems. For example Borrelli et al. [8] applies the RH method in vehicle traction control systems. Perea-Lopez et al. [73] applied it with success in a supplied

chain optimisation, while Atkin et al. [5] used it in an airport push-back time allocation problem. An analogous method called sliding window was also applied to the nurse rostering problem [44]. Tang et al. [82] applied it in dynamic parallel machine scheduling problem.

The machine scheduling problem is to some degree similar to the GAP. One could treat the gates as machines and the flights as jobs which are processed on the machines. This implies the idea of applying the RH method to the GAP problem.

Typically the RH method has been applied to on-line applications. However we found out that it can also be beneficial in off-line applications. The method becomes then a time based decomposition and can speed up the calculations massively if appropriate parameters are chosen. Chapter 6 discusses in more depth how to choose the parameters of off-line version of the RH method depending upon the characteristics of the problem.

2.6.4 Benefits of the RH Approach

Applying the receding horizon method as a decomposition method in some ways gives a similar effect to applying a metaheuristic search. It speeds up the search process but reduces the quality of the solution. Using the RH method instead of a metaheuristic has several important benefits however. There is no need to create a tailored metaheuristic for the problem, any MIP model formulation can be used to solve the sub problems in the window. The quality of the solution is strongly dependent upon the parameters and there is a known trend in the dependency. That makes the parameter tuning a bit easier than in the case of the metaheuristic. Moreover the solution is deterministic, for each run with fixed parameters the same solution will be obtained. That makes the comparisons and parameter tuning easier.

2.7 Ground Movement Problem

For completeness and since this problem is considered in Chapter 7, this section gives an overview of the ground movement problem. The ground movement problem is a routing and scheduling problem which considers routing aircraft on taxiways to their destination points in timely manner. The travel time (or alternatively the distance) and the number of taxiway conflicts are minimised. Moreover appropriate distances between aircraft have to be maintain. Two main approaches to the airport ground movement exist in the literature. The first one utilises MIP models which are often solved using commercial solvers e.g.

CPLEX. The MIP-oriented research is described in Section 2.7.1. The second approach utilises genetic algorithms (GAs) to model and to solve the ground movement problem, it is briefly discussed in Section 2.7.2. Various other approaches were investigated, and the most relevant for the research presented in this thesis are introduced in Section 2.7.3.

2.7.1 Mixed Integer Programming Models for the Ground Movement Problem

The first one divides the whole problem duration into n periods and assigns times and paths for flights period by period. The final solution is build gradually from the partial solutions obtained for each time period. It has been presented in [3] and [77]. The second one orders all flights (decides about the flight order for each part of the taxiway network) and then schedules times for each flight to traverse each part of its path. Examples of this approach can be found in [18, 37, 75].

2.7.2 Genetic Algorithms for the Ground Movement Problem

Various approaches which utilise GAs have been presented in the literature, they can be classified into three main groups based upon the way they tackle conflicting situations during taxiing. The first allows a delay to be introduced only on gates, before starting push-back procedures. The GA determines a delay and a route for each aircraft. This approach is suggested in [46]. The second approach allows a delay to be introduced at any point of a route. The GA determines a route for each aircraft as well as decides where and when to apply a delay and what duration it should have. Examples of this approach are presented in [49, 74]. The last approach, instead of introducing delays directly, prioritises aircraft. When a conflict occurs during movement the aircraft with a higher priority proceeds first. The GA determines the relative priorities of aircraft and their routes. GAs which utilise this approach can be found in [20, 49].

2.7.3 Other Approaches

Other approaches to the ground movement problem were also discussed in the literature. They were not as popular as the two main approaches discussed above but still well worth mentioning:

- Queueing models [2]

- Dijkstras algorithm [15]
- A* algorithm [59, 60]

2.7.4 Utilised Approach

A routing algorithm is applied in a framework presented in Chapter 7. The framework is used to investigate the influence of an allocation plan on the routing and possibly to modify the allocation plan so that the routing is smoother. The algorithm is a greedy algorithm which assumes that all the possible routes for each aircraft type are known. It assigns the routes sequentially. An in depth description of the greedy algorithm application is provided in Chapter 7.

Other algorithms, mentioned in this section can as well be applied in the framework, according to user preferences. The future work plan includes application of an algorithm which was published in [76] and is based upon Dijkstras algorithm. It routes aircraft sequentially, using a directed graph model of the airport. No prior knowledge of the aircraft routes is required.

2.8 Simulation Optimisation Methods

The framework introduced in Chapter 7 allows communication between the gate allocation system and the ground movement system. Since the framework has similarities to simulation optimisation methods an overview of the methods is provided below.

2.8.1 Simulation

Simulation is a common technique used to imitate actual real world systems and their dynamic behaviour over time. It is often used in order to predict and test how the real system would behave for different parameters or input data, in so called what-if analysis. Simulation plays an important role in cases where building the actual system is too complex from the practical point of view. Typically only the key features of the real system are included in the simulator. A simulator imitates the real world but it does not optimise any problem. However when it is used in a simulation optimisation system it takes an active part in the optimization process.

In the framework proposed in Chapter 7 an optimisation algorithm is actually used to solve the routing problem depending upon the output of the allocation optimisation algorithm. The approach should rather be called an optimisation approach than a simulation optimisation approach. However some of the methods used in the simulation optimisation can be applied in the framework.

2.8.2 Stochastic Optimisation

Simulation optimisation is usually mentioned in the literature along with stochastic optimisation methods. Characteristic feature of the stochastic optimisation methods is randomness in either formulation of the model (random constraint formulation or objective function) or in the search process like for example randomly chosen next step of an iterative search process [80]. The group of methods which can be classified as stochastic optimisation is large and contains many, very different methods which have the randomness in common. The random element can be introduced to the optimisation process by simulation and then the process is called simulation optimisation.

Simulation optimisation [34, 56] methods are used in situations when some elements of an addressed problem are highly stochastic. It is easier to simulate the element instead of modelling when knowledge is limited. Simulation optimisation methods vary.

2.8.3 Various Simulation Optimisation Approaches

In some applications the highly stochastic element (or several elements) is simulated and an optimisation method is used to provide input (inputs) for a simulator. This approach is used when there are a lot of possible inputs but some of them are preferred. A set of preferred inputs is created using an optimiser and then given to the simulator. The best from the set is identified based upon the results obtained from the simulation. This kind of approach has been for example used in [58] in a simulation optimisation system for cargo flights assignment problem. A set of solutions for the multi objective problem is created using a mixed integer programming model by keeping one objective and changing the other objectives into hard constraints. The solutions are validated using simulation.

In other applications a simulator and an optimizer work in a closed feedback loop and cooperate to find a good solution. The optimiser produces an input (inputs) and passes it to the simulator. The simulator checks how good the input is and gives feedback

information to the optimiser. The procedure is repeated until set stopping criterion is met. This approach to simulation optimisation has been discussed by [56] and is called recursive approach.

In opposition to the recursive approach are applications in which simulation and optimisation are run separately (non-recursive [56]). The simulation is run first in order to establish some approximate values for stochastic elements of a problem. The obtained approximate values are then used by an optimizer. Again several of possible approximate values can be provided and the optimiser can be run several times-once for each of the values. The best of the approximate values is found based upon the solution obtained for the optimiser. In [45] several examples of the non-recursive method usage in simple manufacturing problems are discussed. Discussed problems are very simple, far from real world problems, but the key features of the simulation optimisation approach are clearly stated.

The simulation optimisation is a large group of methods and very different approaches can be classified to this group of methods. The three discussed above groups give a rough classification of the methods which are based upon the way optimisation and simulation communicate with each other.

The framework presented in Chapter 7 works analogously to the recursive simulation optimisation approach. The approach provides two-side communication between the two parts of the system, the GAP optimiser and the routing optimiser. More details about the implementation and obtained results are given in the relevant chapter.

2.9 Summary

This chapter provides an overview of the five main literature areas: modelling methods, solution methods, receding horizon solution method, ground movement problem and simulation optimisation. They are all related to the aspects discussed later in the thesis.

A general discussion about traditional and alternative modelling methods was first provided. The traditional mathematical formulations as well as alternative GAP models presented in the literature were described and referred to the models which are used in this thesis (see Chapter 4 and Chapter 5). The integration aspects of the published models, which are particularly important for the presented research, were considered in a separate section.

A general overview of the known solution methods was then presented, both the exact solution methods and the heuristics were introduced. The solution methods applied to solve the GAP were briefly described. The solution methods which were chosen to be used in the thesis were discussed in context of other methods in order to motivate the choice.

The next part of the chapter focuses on the RH method which is applied in this thesis (see Chapter 4 and Chapter 6) to solve harder instances of the GAP. Various examples of previous applications in other research areas were presented.

The literature related to the ground movement algorithms was then very briefly reviewed. The ground movement problem is not the main aspect of this thesis. The framework presented in Chapter 7 requires however a routing algorithm in order to detect conflicts occurring on the taxiways.

Finally simulation optimisation methods were discussed in this chapter because of similarities between some of the simulation optimisation methods and the framework presented in Chapter 7.

Chapter 3

Problem Description, Definitions and Datasets

3.1 Introduction

The gate allocation problem (GAP) is a process of finding appropriate places to park aircraft at an airport, this is a general definition which was introduced in Section 2.2, Chapter 2. In this chapter a real world example is used to explain the problem in more depth. The data and basic terms used in the rest of this thesis are defined. Manchester Airport is used as an example airport to demonstrate the scale of the problem, the issues which are resolved by the currently used systems and the way that the problem is tackled at the moment. Research directions, which originate from the real problem analysis, are discussed by the end of this chapter.

3.2 Basic Terms

When the GAP is discussed, several terms should be clearly defined since naming and meaning varies between authors.

- landside - is the part of the airport which serves passengers. Desks, security checks, baggage claiming areas, terminals, waiting areas belong to the landside.
- airside - is the part of the airport designed to serve the aircraft. Runways, taxiways belong to the airside.
- gate - is the passage for passengers between airside and landside. It has a land side, which is an area where passengers can wait before boarding an aircraft, and an air side, which is an area next to a terminal where an aircraft can park (it is sometimes called

an on-pier stand). In the research presented here the air side of gates is primarily considered. Whenever the word gate is used, it should be understand to mean the air side of a gate.

- flight - is an aircraft which has a scheduled arrival time and a scheduled departure time.
- remote stand - is an area on the airside which is designed for aircraft to park but is not attached to a terminal building (sometimes called an off-pier stand).
- allocation - a gate with an allocated flight is often called an allocation in this work.
- taxiway - a route along which aircraft can taxi from a runway/gate to another runway/gate
- push back - the operation of pushing an aircraft away from a gate when it is ready for departure. An aircraft will usually only start its engines once it has completely pushed back.
- towing - an operation of moving an aircraft from a remote stand/gate to another remote stand/gate (usually using a tractor or tug).

3.3 Manchester Airport as an Example Airport

Relevant elements of airport operations are discussed using Manchester Airport as an example. Firstly the gate allocation problem is discussed, then basic information about the runways and the taxiways is given.

3.3.1 Scale of the Gate Allocation Problem

Gate allocation information from the whole airport (3 terminals) for a week of airport operation has been provided by Manchester Airport. There are twenty one datasets in total, each dataset covering one day of one terminal's operations. Each dataset includes scheduled allocations and records of the actual events as well as sizes of gates and aircraft. Preferences of airlines were estimated based upon the records using statistical data analysis.

A preliminary analysis of traffic density was done and proved that each of the terminals has slightly different traffic characteristic. Terminal T1 has one hundred twenty

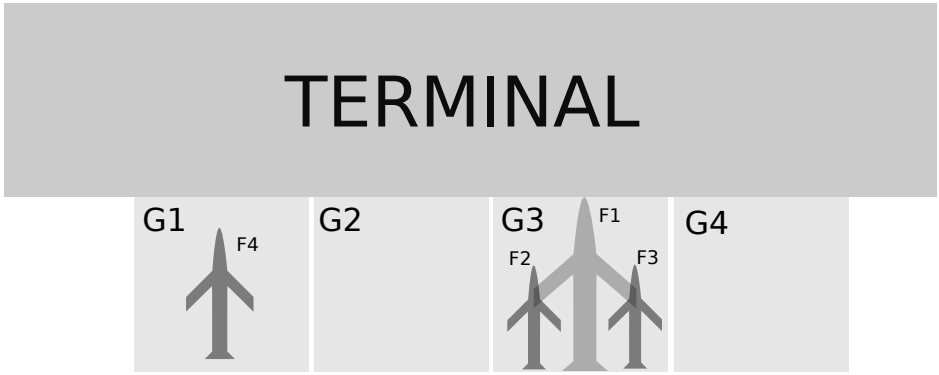


FIGURE 3.1: Gate usage

flights per day across its twenty five gates, the most flights per day from all terminals. In contrast, terminal T2 is a very quiet terminal, it has on average sixty flights per day across twenty one gates. On average there are one hundred eight flights per day on terminal T3 which is fewer than on terminal T1 however the terminal has only twenty one gates. Terminal T2 has on average less than three flights per gate. The density of flights is higher for terminal T1 than for terminal T2 but it is less than five flights per gate per day. The average number of flights per gate at terminal T3 is five which is the highest of all of the terminals.

The on-pier stands have various sizes, some can accommodate only small flights but for the majority of the gates it is possible to allocate either one bigger or two smaller aircraft. Sometimes one smaller and one bigger can fit, depending upon the size of the gate and the type of aircraft. Figure 3.1 shows an schematic example of four gates, showing that $G3$ can be occupied either by two small flights $F2, F3$ or by one large flight $F1$. There are around forty remote stands at the airport, which are jointly used by all terminals. It is preferable not to allocate any flights to the remote stands but during very busy periods it is sometimes unavoidable. When an aircraft has to take a remote stand, although it arrives according to the schedule, the airport covers the passengers' coaching costs. The remote stands are often used at night; an aircraft is towed to a remote stand for the night, then towed back in the morning. Sometimes an airline can ask for a remote stand, for example washing requires a special remote stand. Manchester Airport has two remote wash stands

which are properly equipped with washing facilities and drainage systems.

3.3.2 Gate Allocation Rules

The allocation rules define generally how the gates will be assigned, when a daily plan will be published, how the remote stands will be assigned and how the towing will be organised.

Additionally, the size of gates and aircraft are taken into account. All types of aircraft are divided into size groups depending upon the wingspan. The groups of aircraft are given in Appendix A. A list of all gates is also given in Appendix B. The various exceptions are also detailed, which occur when it is possible to allocate some types of aircraft from an upper size group to a gate that normally takes aircraft from a lower size group.

The minimum expected duration of an allocation is also specified in the allocation rules of the airport, for big aircraft it is typically 1:45 h and for smaller aircraft 1:30 h, however there are also flights that only stay on an airport for 30 minutes. The time gap which should be allowed between following allocations on the same gate used at Manchester Airport is at least 15 minutes long for outbound flights (to allow some time slips). Flights which are towed do not require the gap; it is assumed that the towed aircraft will be on gates on time.

There are additional practical rules that the controllers know from experience. Aircraft on the same combinations of gates can block each other, for example it is impossible to push back simultaneously from some gates. Allocating some gates which do not have large passenger waiting areas to big aircraft can cause passenger congestion and delays in a terminal because passengers that need to go to gates which are located further away have to crowd through the queue of passengers waiting by the gate which was allocated for the big aircraft. The airlines have also preferences and some flights have more strict security procedures that are easier to follow with fixed gates. These are only a few examples of the expert knowledge which is needed to make good allocation decisions. In order to capture this information in this research, where information about rules was not available, the historic allocations have been analysed and used as preferences.

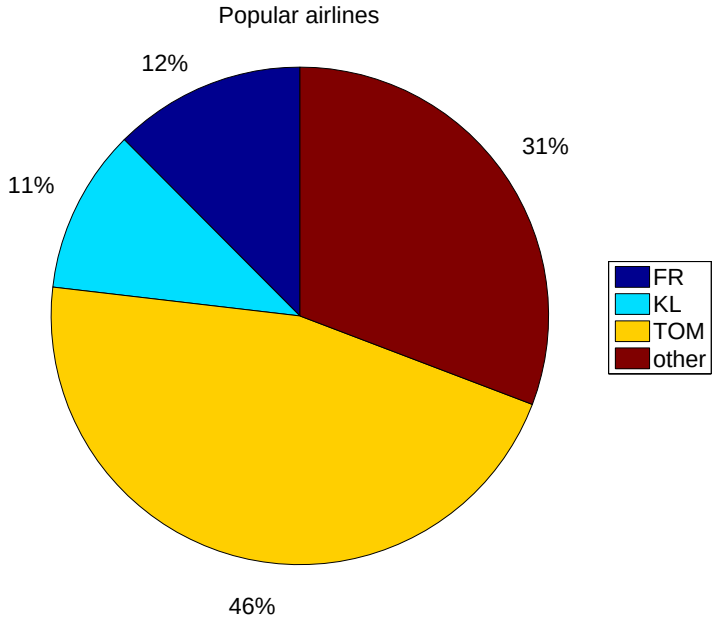
3.3.3 Airline Preferences

Unfortunately information about the airlines and airport preferences is not given explicitly. It has been estimated by analysing the data to extract the information.

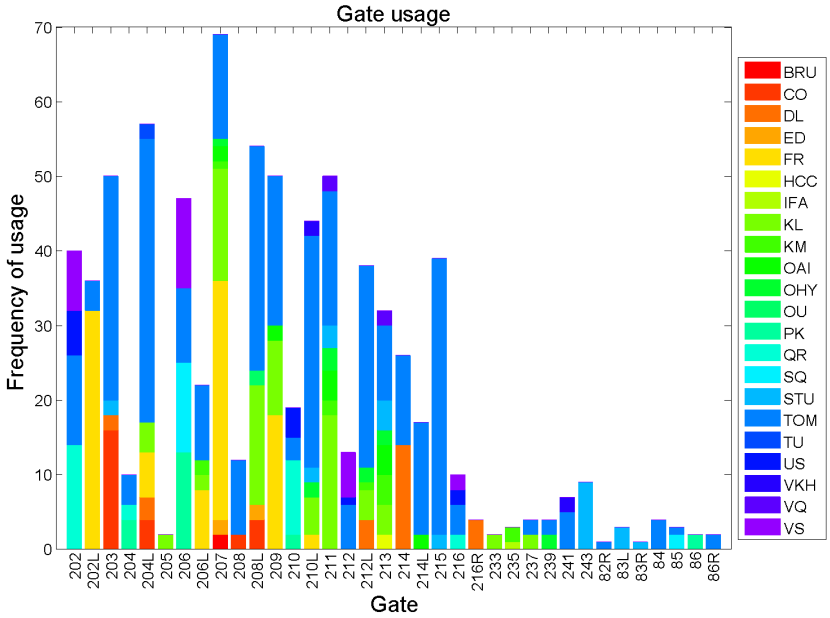
The data analysis helped to answer several questions - related to airline preferences: which airlines were most common for each terminal, which gates were occupied by which airlines, and for gate-size preferences: how many times a gate has been occupied by a particular size of aircraft. The analysis was done for all the three terminals. The results obtained are presented below using terminal T2 as an example.

Figure 3.2 presents two plots which visualise two aspects of data (terminal T2): the most common airlines and the airline gate usage. Figure 3.2a shows a pie diagram on which the most popular airlines of terminal T2 are presented. It can be observed from the figure that there is one dominating airline called *TOM*. A bar plot which is presented in Figure 3.2b shows which gates were occupied by which airline. The names of gates are indicated in the horizontal direction and the frequency of usage in the vertical direction of the figure. Each airline is marked in different colour. It can be seen that *TOM* used almost every gate. It is therefore rather hard to say which of the gates are the most preferred by this airline. It is easier to observe the preferences for the second most popular airline *FR* which used only six gates: 202L, 204L, 206L, 209 and 210L. Allocating flights which are operated by *FR* to these gates should be prioritised (according to the frequency of usage) in the model. Similar analysis has been performed for every airline in order to establish its preferences. The airline preferences are used in the objective function of the model presented in Chapter 5. The aim being to find solutions which are similar to known acceptable solutions.

Figure 3.3 presents a bar plot which is analogues to the one shown in Figure 3.2b, it shows the sizes of aircraft that have been allocated to gates instead of the airlines. Colours in the image correspond to aircraft size groups that are used by the airport, the lighter the colour the larger the size group of the aircraft. It has been checked that the aircraft sizes usually match the maximal sizes of the gates, showing that the airport tries to fill gates with maximal aircraft size, as expected. These preferences are incorporated in the MIP model (see Chapter 5) in the size preference part of the objective function. The larger the difference between the size of the aircraft and the maximal size that can be put on a gate, the bigger the penalty.



(a) The most popular airlines, Terminal 2, Manchester Airport



(b) Airline gate usage, Terminal 2, Manchester Airport

FIGURE 3.2: Airline statistics.

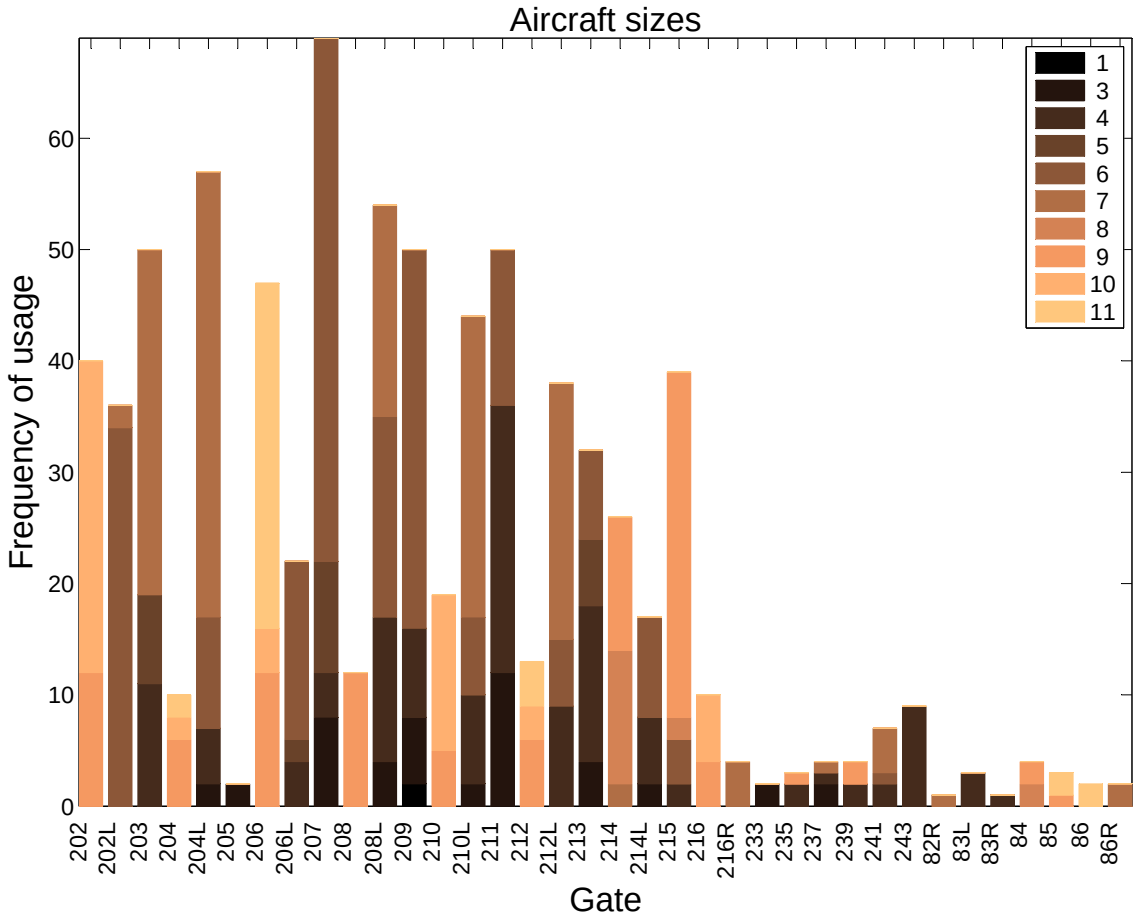


FIGURE 3.3: Sizes of aircraft on gates

3.3.4 Gate Allocation Planning

The gate allocation planning is done in several stages. Planning in advance (strategic planning) is usually done four months in advance for a six month period, twice a year - once before a summer season and once before a winter season. When planning for a summer season the busiest week from a previous summer season is used as a base for the new plan, similarly, for a winter season the hardest winter week is used. The planning can start when all of the time slots for the coming season are known. The time slots are prepared for Manchester Airport by an external company. The company creates the plan after all agreements between the airport and the airlines have been completed.

The advanced plan usually has to be updated one week before every week due to various events that cannot be predicted so long in advance (tactical planning), for example maintenance work. When the plan is updated, remote stand allocations from the previous week and the planned remote allocations are sent to all airlines. Publishing information about the remote allocations in advance as well as sending the list of remote stands assigned in a past week with explanations is a fairness policy of the airport. Most of the airlines prefer to use on-pier stands and should therefore be informed about the remote allocation well in advance.

3.3.5 Current Software Solution

The planning, both four months in advance and a week in advance, is accomplished using the Chrome Assign software. Chrome Assign is a rule based system created by Amor Group Company, and was introduced at Manchester Airport several years ago. The system is based upon rules introduced by an experienced controller. Figure 3.4 shows a rule that gates number 202 cannot be used by airline *CO*.

There are approximately two hundred rules in the system, divided into three groups:

1. The preference rules (these are the various preferences that airlines have);
2. The stand hard constraints (size of gates needed to fit each aircraft type, some gates are non-intercontinental etc.);
3. The adjacency rules (e.g. which gates block other gates);

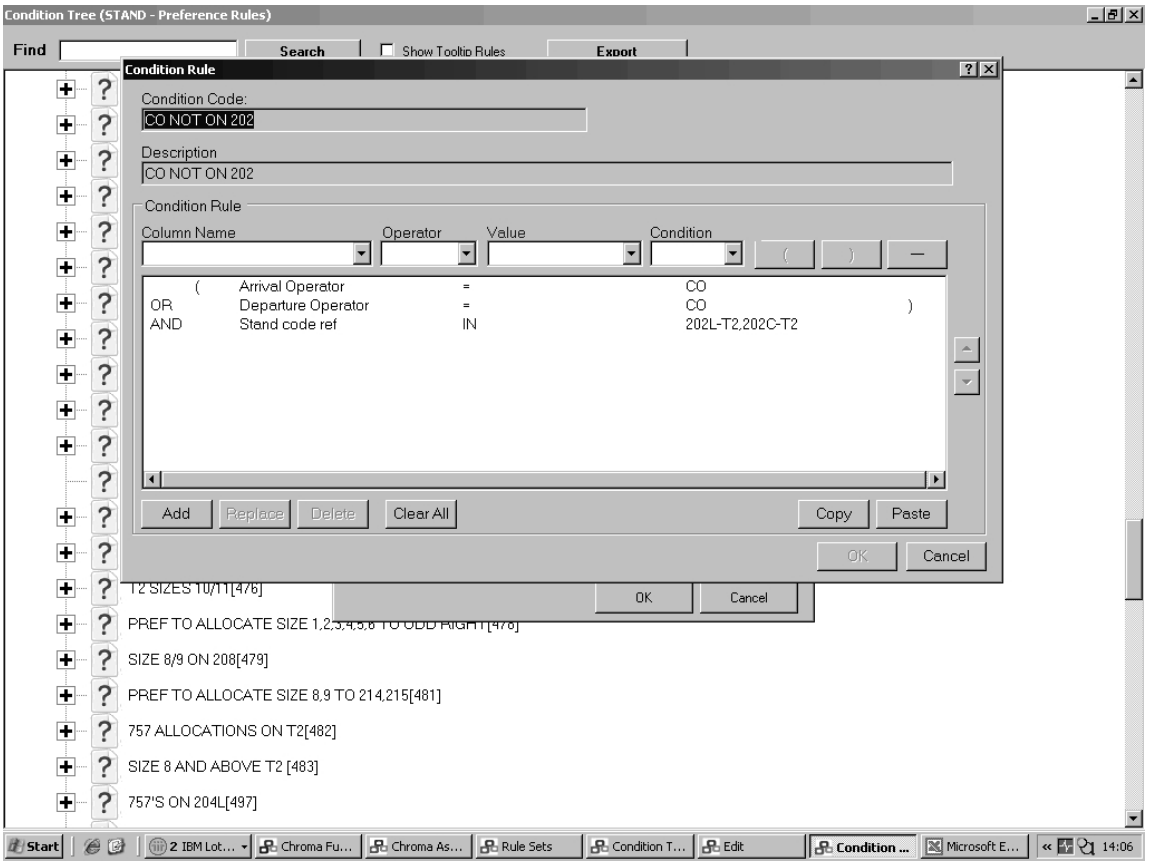


FIGURE 3.4: A hard allocation rule

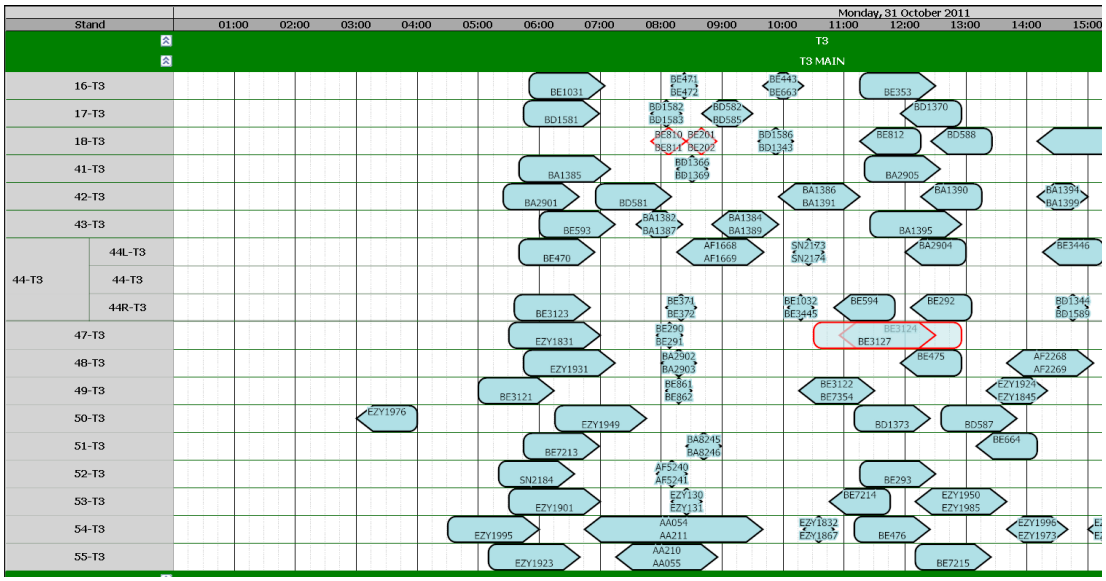


FIGURE 3.5: An allocation plan displayed by Chrome Assign

The last stage of planning (operational planning) happens during the day of operation. It is done manually by the controllers who can tune the existing plan using the user interface. Figure 3.5 shows how the user interface looks. The program displays flights as boxes. The boxes are located on a chart that has names of gates in the vertical direction and the time in the horizontal. When the allocation is correct and no rules are broken a flight box is dark blue. If something goes wrong, the colour of the box changes, becoming light blue with read frame. When a controller selects the box he/she gets a description of the problem. A light blue colour means that a soft rule has been broken (e.g. an airline prefers a different gate). A red colour means that there is an impossible allocation (e.g. too large an aircraft at the gate, or two of them at the same gate). Additionally an allocation planned for one example day for all terminals using the Chrome Assign is shown in Appendix **??**.

3.3.6 Runways and Taxiways

Manchester Airport has two runways. The runways can operate in two directions, depending upon the wind direction. As the west winds are more common, the west orientation of the runway is usually used (aircraft take off facing south-west and land facing south-west). The runways are not always simultaneously used. A mixed mode usage of one runway can often

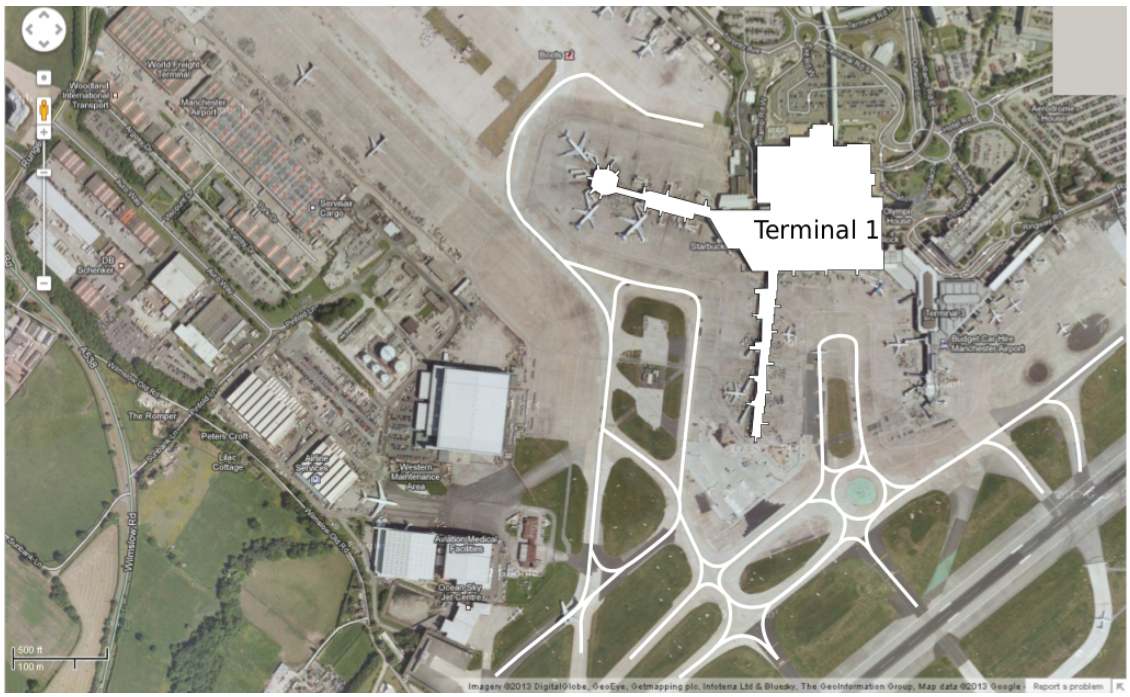


FIGURE 3.6: Terminal 1 of Manchester Airport (Google satellite map, Imagery ©2013 TerraMetrics, Map data ©2013 Google)

be observed during wintertime when the traffic is lower and weather conditions tend to be worse (doing this reduces the snow and ice removal expenses).

Figure 3.6 shows a map[1] on which Terminal 1 of Manchester Airport and the taxiways which lead to/from it are marked in white. The gates are placed around the two piers, several aircraft parked at gates can actually be identified in the figure.

3.4 Identified Research Directions

Push back conflicts in the area around the gates are partially incorporated (see Figure 3.4) in the rules. More rules which limit the number of conflicts could possibly be deduced from the architecture of the airport but some may be soft rather than hard constraints.

A closer cooperation between the various airport operations may help. For example elements of the aircraft routing problem could be included in the model of the gate allocations so that the number of conflicts occurring on the taxiways is reduced.

The on-line version of the problem could also be addressed. A support system which helps the controllers in re-allocating gates would apparently be very welcomed at the

[1] Google satellite map, Imagery ©2013 TerraMetrics, Map data ©2013 Google

airport, since the complexity and time requirements currently mean that these operations are performed manually.

3.5 Conclusion

This chapter introduced the problem which is addressed in the thesis. The example of Manchester Airport is used to visualise the elements of the problem and show the scale of the problems which are tackled. The datasets and all the information provided by Manchester Airport are used throughout this the thesis to model the problem and to experiment with the various implemented solution methods. One of the important aims of the project within which the research is realised was to bring the research closer to the real world problems, which can only be achieved when real datasets are used.

Manchester Airport has a medium size, thus is a very good example to use in experiments with new methodologies. The methods which are described in the thesis can be adapted to other airports. It would be interesting to experiment with larger airport instances, for example Heathrow Airport. It is however not in scope of the project for which the work is done and no data was available for experiments at the time. Experiments with other instances could be a part of a new project in the future.

The identified research directions include the on-line version of the GAP which is not addressed in the thesis. Creation of an on-line version of the semi-integrated model which is presented in Chapter 7 is another very interesting possible future research direction.

Chapter 4

The Receding Horison Approach to the Gates Allocation Problem Preliminary Study

4.1 Introduction

This chapter presents a preliminary study related to the application of the receding horizon (RH) approach to solve a basic formulation of the gate allocation problem (GAP). The basic formulation of the GAP includes the core constraints which were identified in Chapter 2 have to occur in any realistic model of the problem along with a time gap constraint which introduces the auxiliary variable that is used in the objective function to maximise the sum of the time gaps between allocations. This basic formulation of the problem is extended in Chapter 5 by adding more constraints and objectives.

In this chapter, the basic formulation is first solved exactly, using a commercially available solver called CPLEX. This part of the research aims to increase the confidence in the model and the way the solver works. The formulation is hard to solve to optimality, it requires long solution times and a lot of memory. Hence the RH approach, a window-based decomposition, which creates a series of sub-problems small enough to be solved exactly, is applied to decompose the GAP. Although the RH approach is widely used to decompose large optimisation problems [73, 79, 13, 82] it has, to the best of our knowledge, not previously been applied to solve the GAP. The main contribution of the research presented in this chapter is the first and successful application of the RH method to decompose the basic GAP. Several configurations of the parameters of the RH method are tested in order to observe how the shift size and the window size influence the calculation times and the objective values. This research indicates the high potential of the RH method and motivates

the further work which is presented in Chapter 6.

The chapter is organised as follows: first the basic formulation of the GAP model is described in Section 4.2, next the general idea behind the Receding Horizon is presented in Section 4.3, then the model and the RH method settings are discussed in Section 4.4. Finally results and conclusions are given in Section 4.5 and Section 4.6, respectively.

4.2 Gate Allocation Problem Basic Model Formulation

The constraints and the objective function are described in this section. These constraints have been discussed in other models of the problem given in the literature (Chapter 2), these are the core constraints which have to occur in every model describing the problem. The objective used in the basic model, which aims to increase the gaps between allocations was also discussed by other authors and is one of the most important objectives. The objective significantly influences the problem complexity as it refers to pairs of flights on gates not just allocating a flight to a gate. This influence is further investigated in the experiments.

4.2.1 Notation and Definitions

F	set of flights
n	total number of flights
i, j	flight indexes $(i, j) \in \{1, ..., n\}$
$f_i \in F$	a flight
e_{f_i}	on-gate time of f_i, constant for each f_i
l_{f_i}	off-gate time of f_i, constant for each f_i
G	set of gates
m	total number of gates available
k, l	gate indexes $(k, l) \in \{1, ..., m\}$
$g_k \in G$	a gate
X_{g_k,f_i}	decision variable, becomes 1 if f_i is allocated to g_k, 0 otherwise
U_{g_k,f_i,f_j}	indicator variable, becomes 1 if f_i and f_j are allocated to g_k, 0 otherwise
gap_{g_k,f_i,f_j}	time gap between f_i and f_j allocated to gate g_k
SG	minimum size of gap_{g_k,f_i,f_j}, a constant
LG	maximum size of gap_{g_k,f_i,f_j} , a constant
p_{f_i,f_j}	penalty function for putting f_i and f_j on the same gate
du	penalty for 'dummy gate'
$OVERNIGHT$	set of flights which stay overnight
$ghist_a$	gate that has been used by flight a in the historic data

4.2.2 Constraints

The time gap gap_{g_k,f_i,f_j} between two flights f_i and f_j which are allocated to the same gate g_k is defined by Equation 4.1. The variable U_{g_k,f_i,f_j} which indicates whether both f_i and f_j are allocated to gate but there is only a small gap between the allocations g_k, is only set if size of the time gap between them gap_{g_k,f_i,f_j} is between SG and LG (equation 4.2) i.e. U_{g_1,f_1,f_2} will only ever be set for flights f_1 (departure time lf_1) and f_2 (arrival time $ef_2 > lf_1$) allocated to gate g_1 only if $SG < ef_2 - lf_1 < LG$. Minimising these values is an objective.

$$gap_{g_k,f_i,f_j} = \begin{cases} e_{f_j} - l_{f_i} & if\, e_{f_j} \geq e_{f_i} \\ e_{f_i} - l_{f_j} & if\, e_{f_i} \geq e_{f_j} \end{cases},\ f_i, f_j \in F,\ g_k \in G,\ f_i \neq f_j \tag{4.1}$$

$$\left.\begin{array}{l} U_{g_k,f_i,f_j} \geq X_{f_i,g_k} + X_{f_j,g_k} - 1 \\ U_{g_k,f_i,f_j} \geq 0 \end{array}\right\} \begin{array}{l} \forall f_i, f_j \in F,\ \forall g_k \in G,\ SG < gap_{g_k,f_i,f_j} < LG, \\ 0 \leq SG \leq LG \end{array} \tag{4.2}$$

$$\sum_{k=1}^{m+1} X_{g_k,f_i} = 1, \quad \forall f_i \in F \tag{4.3}$$

$$^{1}\sum_{i=1}^{n}\sum_{k=1}^{m+1} X_{g_k,f_i} = n \tag{4.4}$$

$$X_{f_i,g_k} + X_{f_j,g_k} \leq 1,\ \forall f_i, f_j \in F\ \forall g_k \in G,\ gap_{g_k,f_i,f_j} \leq SG,\ SG \geq 0 \tag{4.5}$$

$$X_{a,ghist_a} = 1,\ \forall a \in\ OVERNIGHT \tag{4.6}$$

Each flight has to be allocated to a gate (Constraint 4.4). Each of the flights is allocated to only one gate (Constraint 4.3). The 'dummy gate' is included in these constraints and it has number $m + 1$. All the flights for which there was no gate available are allocated to the 'dummy gate'. It symbolises more than one stand hence several flights can be allocated to it at the same time. The 'dummy gate' usage is strongly penalised in the objective function.

[1] After the research had been conducted this constraint was found to be redundant. Constraint 4.3 is sufficient to force each flight to be allocated to a gate. It was however checked that incorporation of this constraint has a minor influence on the calculation times and does not have any other impact on the results.

Constraint 4.5 guarantees that two flights which overlap or are within minimum gap SG of each other are not allocated to the same gate.

Constraint 4.6 allocates the overnight stays. $OVERNIGHT$ is a set of flights which according to the historical record arrived the day before and stayed overnight at the same gate. These flights are automatically allocated to the gates that they used in the historic data ($ghist$), since they were already at the airport when the day starts.

4.2.3 Objective

The objective function is given by Formula 4.7 which is minimised in the model. It is a weighted sum of the time gap variables (the first element of Formula 4.7) and the total number of remote allocations (the second element of Formula 4.7). This preliminary study focuses on the objective which aims to maximise the time gaps between allocations. The formulation on this objective is new compared to previously published ideas, it therefore requires a careful consideration. The second objective which minimises the number of remote allocations has to be in the objective function otherwise all flights would immediately be allocated to the 'dummy gate'.

$$Min \quad \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{m} p_{f_i,f_j} U_{g_k,f_i,f_j} + \sum_{i=1}^{n} duX_{f_i,dummy} \tag{4.7}$$

$$p_{f_i,f_j} = \frac{LG}{(gap_{g_k,f_i,f_j})} \quad if\ SG < gap_{g_k,f_i,f_j} < LG \tag{4.8}$$

The penalty function is given by Equation 4.8. The shape of the function is chosen so that the smaller gaps are penalized much more than the larger gaps. Figure 4.1 shows the shape of the function when the size of SG equals $\frac{1}{4}$ of LG (as it is in the experiments). Flight pair variables (U_{g_k,f_i,f_j}) are forced by Constraint 4.2 only for gaps which are within the range $SG < gap_{g_k,f_i,f_j} < LG$.

4.3 Receding Horizon Approach

This section explains the general idea behind the Receding Horizon (RH) approach.

4.3.1 Notation

All previous definitions from Section 4.2 hold and the following additional definitions are used in the following sections.

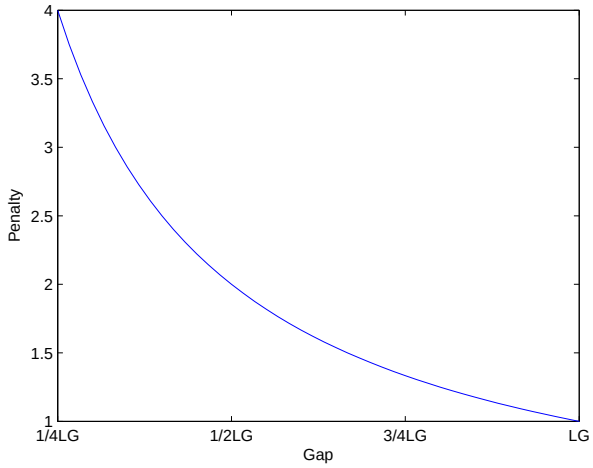


FIGURE 4.1: Gaps penalty function

T	size of window in minutes
d	size of window shift in minutes
$lastFlight$	last flight in the dataset
$pozNum$	total number of positions of window
p	position index of window $p \in \{0, ..., pozNum\}$
FT_p	subset of flights, their arrival times are in T for position p
Fd_p	subset of flights, their arrival times are in d for position p
$MIP(s)$	MIP solution for subset (s) of flights

4.3.2 Overview of the RH Method

The receding horizon (RH) approach which is used in this research to decompose the GAP into sub-problems works as follows. The sub-problems are extracted from the full problem using a window. Firstly the window is placed at the beginning of the problem and the GAP model (described in Section 4.2) is built for the flights belonging to the window (Section 4.3.3 describes how the flights in the window are determined). The GAP model is solved. The window is shifted to a next position while the part of the solution from the first position which was moved out of the window is kept and fixed for all of the following steps. The fixed solutions from the previous positions which are still inside the new window are used as an initial solution for the current window position. The procedure of shifting is repeated until the end of the dataset is reached. Figure 4.2 illustrates the general idea of window positioning and shifting. The horizontal axis denotes the time flow and the vertical denotes the gates. The rectangles (labelled $F1, F2, F3, F4, F5$) symbolise flights which occupy gates

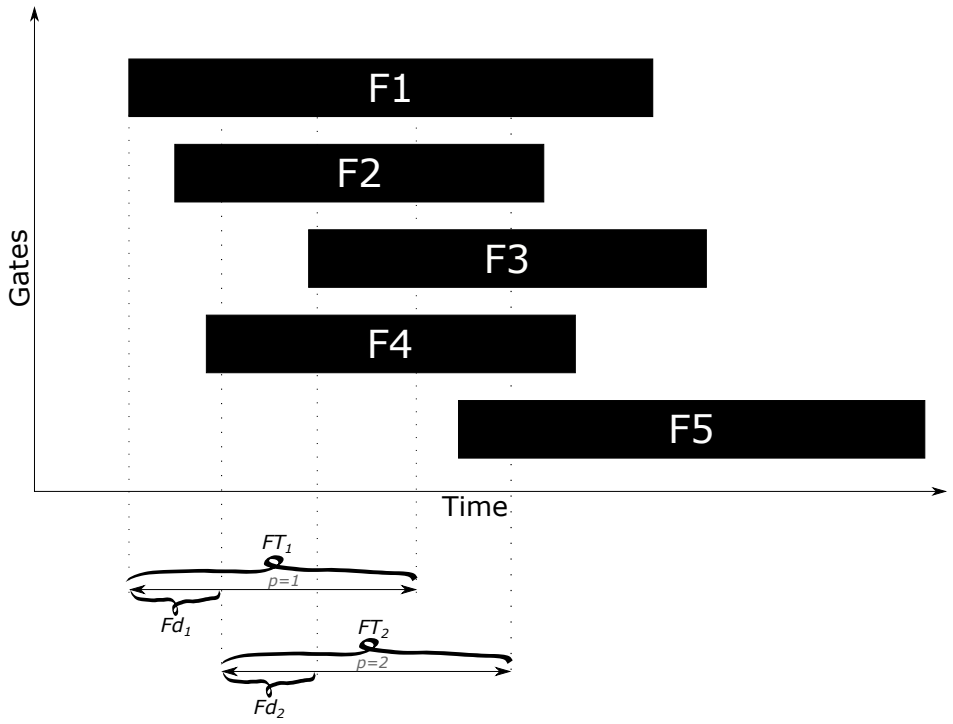


FIGURE 4.2: Window positioning and shifting.

for some time duration. Two example window positions (for $p = 1, p = 2$) are shown below the chart and the subsets of flights Fd_p and FT_p are indicated. The subsets which are marked in Figure 4.2 contain the following flights respectively:

- $Fd_1 = \{F1, F2, F4\}$
- $FT_1 = \{F1, F2, F3, F4\}$
- $Fd_2 = \{F3\}$
- $FT_2 = \{F3, F5\}$

The calculation time and the quality of the solution obtained using the RH method depends heavily on the two parameters d and T. An appropriate trade-off between the quality of the solution and the calculation time has to be found. Our research aims to investigate whether it is possible to quickly find solutions which are close to optimal using the RH approach. Pseudo-code for the RH algorithm is provided in Algorithm 1 which gives additional explanation.

Algorithm 1 Receding Horizon Method

place time window at the start of the schedule($p = 0$)
create subsets FT_p and Fd_p
create new MIP model for FT_p
solve MIP model for FT_p
store allocations for flights from subset Fd_p
advance the position ($p = p + 1$)
while $p <= pozNum$ **do**
    create subsets FT_p and Fd_p
    create new MIP model for subset FT_p
    preallocate gates from previous window positions
    solve MIP model for subset FT_p
    if $p = pozNum$ **then**
        store allocations for flights from subset FT_p
        end algorithm
    else
        store allocations for flights from subset Fd_p
    end if
    advance the position ($p = p + 1$)
end while

4.3.3 Identification of Flights in a Window

A time-oriented implementation is used in this study, where the decision about including a flight in a window is taken based upon its arrival time (alternative approaches were tested during further stages of the research and are discussed in Chapter 6). In this implementation the number of flights per window varies, since in some time periods there are more arriving aircraft than in others. The window size T and shift size d are measured in minutes. Equation 4.9 shows the number of window positions for this decomposition, Inequality 4.10 defines the flights in subset FT_p for position p, Inequality 4.11 defines the flights in subset Fd_p for position p.

$$pozNum = \left\lceil \frac{e_{lastFlight} + d - T}{d} \right\rceil \tag{4.9}$$

Note that the subset of flights within shift d may be empty for some window positions. In that case the window position is skipped and the algorithm continues for the next window position, so the number of window positions actually needing to be solved may be smaller than $pozNum$.

$$f_i \in FT_p \;\; iff \;\; (p * d) < e_{f_i} \leq ((p * d) + T) \tag{4.10}$$

$$f_i \in Fd_p \;\; iff \;\; (p * d) < e_{f_i} \leq ((p + 1)d) \tag{4.11}$$

4.4 Parameters Settings

This section explains the settings which were used in the experiments for the GAP model and the RH method. The GAP model requires deciding about the minimum and the maximum size of the time gap between allocations, based upon which the penalties in the objective function are calculated. The implementation of the RH method used here requires two parameters: the size of the shift and the size of the window. These two parameters are independently varied in the experiments so that it can be observed what their influence on the results is.

4.4.1 GAP Model

The sizes of the small and the large gap between allocations are set as follows:

Configuration Number	Window Size [%]	Shift Size [%]
1	5	20
2	5	30
3	5	40
4	5	50
5	10	20
6	10	30
7	10	40
8	10	50
9	15	20
10	15	30
11	15	40
12	15	50
13	20	20
14	20	30
15	20	40
16	20	50
17	25	20
18	25	30
19	25	40
20	25	50

TABLE 4.1: The configuration numbers and parameter settings for the RH method.

- small gap SG equal to 15 minutes
- large gap LG equal to 60 minutes

The sizes may vary from one airport to another, the above values are chosen according to the datasets used in the experiments and information about the real situation at Manchester.

4.4.2 RH Method

The sizes of the window and the shift are varied in the experiments. The following window sizes: 5, 10, 15, 20, 25 percent of the dataset duration (typically one day) and shift sizes: 20, 30, 40, 50 percent of the window size are used. There are twenty configurations in total. The configurations are numbered from 1 to 20. Table 4.1 shows all the tested configurations and the corresponding numbers assigned. For clarity and brevity, numbers are used to identify configurations rather than both of the parameters when the results are presented.

4.5 Results

This section presents results obtained using the model and the RH method which are described above. The datasets from Manchester Airport which were described in Chapter 3 are used in the experiments. There are twenty one datasets in total, which cover seven days of the airport operations (each terminal separately). The days are solved one after another. Flights from a previous day which stay overnight are automatically allocated to the gates they used in the historic data. This allows a fair comparison of the solutions obtained for different configurations of the RH method on the same day. In that way all solutions will have the same configuration at the beginning of each day. Alternatively the night stay allocations could be passed from the previous day to the following one. But then it would only be fair to consider the RH solutions obtained for an entire week, not for individual days. The same approach has been used in further experiments with the RH method in Chapter 6.

The model formulation contains only two elements in the objective function. The element which aims to maximise time gaps between allocations increases the problem complexity a lot, as many more variables have to be considered in the objective function. The size of large gap LG regulates the number of the variables U_{g_k,f_i,f_j} which are actually considered. Section 4.5.1 focuses therefore on how the model performs with and without the time gap constraint as well as on the influence of the LG parameter on the model size and performance.

Next in Section 4.5.2 the results obtained when all of the datasets were solved using CPLEX without the RH decomposition but with a time limit of 1 hour for CPLEX are presented and briefly analysed. If no optimal solution can be found in this time, the best feasible solution found is given.

Finally the RH decomposition is applied to the instances which are hard to solve using the full formulation. The same time limit of one hour is set for the total time of the RH method calculation. Various configurations of the parameters of the RH method are tested in the experiments and it is shown how they influence the calculation time and the obtained objective values. Results of the experiments and the analysis of the influence of the parameters and the best settings of the RH method for the tested instances are given in Section 4.5.3.

LG [min]	Time[sec]	Gap[%]	Variables	Constraints
-	0.89	0	2706	52006
15	1.73	0	320415	52006
50	12.2	0	320415	68134
51	4.02	0	320415	68554
52	4.42	0	320415	69226
53	16.76	0	320415	69646
54	126.73	0	320415	70234
55	579.37	0	320415	70738
56	3600	41.88	320415	71158
60	3600	69.38	320415	73216

TABLE 4.2: Model performance for an example database: terminal T3 day 4.

4.5.1 Model Performance

The model performance analysis utilises a dataset from terminal T3, day 4. There are one hundred and twenty three flights in the dataset. Terminal T3 has twenty one gates plus there is one additional gate which models the remote gates (twenty two in total). There are twelve flights which arrived at the airport the day before and stayed on the assigned gate for the whole night. There are $(m+1)*n = 22*123 = 2706$ X_{g_k,f_i} variables in the model and $m*n*n = 21*123*123 = 317709$ U_{g_k,f_i,f_j} variables as the remote gates are not included. In total there are $(m+1)*n+m*n*n = 2706+317709 = 320415$ variables in the model.

Similar calculations can be performed for the number of constraints (which are also called rows) in the model. Constraint 4.4 adds one row to the model, Constraint 4.3 adds n (123) rows to the model, Constraint 4.6 adds twelve rows (as there are twelve overnight stays). Constraint 4.5 introduces at most $m*n*(n-1)$ rows ($n-1$ as no constraints for the same flights are added). There are actually far less rows introduced by this constraint since all of the overlapping pairs of flights which are within SG of each other are excluded. Similarly Constraint 4.2 introduces much less rows than it potentially could as it is active only for the pairs of flights for which the time gap is between SG and LG. It is further investigated how the model changes when the LG parameter is varied. The size of LG is determined based upon the preferences of the airport and is not going to be changed further in the calculations. It is only varied in this part to show and better understand the model behaviour.

Table 4.2 displays the recorded calculation time, optimality gap, number of variables, number of constraints for the instance and various settings of the LG parameter. The optimality gap is the difference between the feasible solution and the lower bound found by CPLEX, divided by the lower bound value, and assesses the quality of the solution. The first row shows the results obtained for a model without the gap maximisation (without Constraint 4.2 and the first element of the objective function). The second row shows the results with all of the elements of the model but for LG equal to SG (15 minutes). The following rows displays the full model results obtained for increasing sizes of LG starting from size fifty and finishing with sixty (LG is equal to 60 minutes in further experiments).

The expected numbers of variables occur in the table, there are 2706 for the model without gap elements and 320415 for the model with. The number of variables does not change for the different value of LG as the different size of the LG only increases the number of constraints, does not change the total number of variables in the model. The number of constraints which are present in the models is the same for the first and the second row, since for LG equal to SG the gap constraint is inactive. The number of constraints grows with the size of LG since more U_{g_k,f_i,f_j} variables have to be considered in the objective function and it is harder to find the solution. When the LG reaches a size of 56 the calculation time grows rapidly and only a feasible solution can be found within the set time limit (1 hour). This rapid growth is very characteristic for the problem formulation, and can be also observed when other instances are solved. It may be related to the formulation symmetry (see Section 2.5 in Chapter 2), as any flight fits on any gate. The next section presents the results obtained for all datasets using just one preferred setting of LG. Some of the datasets were still solvable within seconds but some already reach the sizes which cause the rapid growth of the calculation time.

4.5.2 Full GAP Model Formulation Results

The times taken by the twenty one instances, as well as the solution status and the optimality gaps for the feasible solutions, the number of variables and the number of constraints are presented in Table 4.3.

It is discussed in Chapter 3 that terminal T2 is quiet and has a low number of flights per gate, while terminals T1 and T3 have a larger number of flights per gate on average. Terminal T2 is therefore expected to be easier to solve than T1 and T3. Table 4.3

Day(Terminal)	Gates	Flights	Time[sec]	Gap[%]	Variables	Constraints
1(1)	25	111	2.5	0	310911	78112
2(1)	25	118	22.3	0	351168	103133
3(1)	25	121	3600	1.95	369171	116531
4(1)	25	125	3600	1.13	393875	111688
5(1)	25	133	3600	3.65	445683	125243
6(1)	25	116	3600	1.58	339416	101079
7(1)	25	106	30.4	0	283656	80814
1(2)	23	50	0.8	0	58700	15047
2(2)	23	59	0.5	0	81479	21363
3(2)	23	60	0.6	0	84240	26562
4(2)	23	59	0.5	0	81479	2444
5(2)	23	67	0.7	0	104855	29051
6(2)	23	62	0.5	0	89900	26239
7(2)	23	48	0.4	0	54144	17347
1(3)	21	91	1.6	0	175903	41378
2(3)	21	117	4.5	0	290043	64310
3(3)	21	119	11.9	0	299999	67167
4(3)	21	123	3600	69.38	320415	73216
5(3)	21	121	7.2	0	310123	65949
6(3)	21	92	0.9	0	179768	40383
7(3)	21	80	1.0	0	136160	3949

TABLE 4.3: Full GAP model formulation results

confirms the expectations, as all of the instances coming from terminal T2 were solved to optimality within a very short time: all times were shorter than one second. Terminals T1 and T3 take longer to solve, in five cases only feasible solutions could be found. The quality of the feasible solutions varies, the best obtained feasible solution has an optimality gap around 1% (instances 4(1), 3(1) and 6(1)) while the worst has 69.38% (instance 4(3)). It can be seen in Table 4.3 that the calculation times are either a few seconds long or longer than one hour in these cases.

This bipolar characteristic of the results is related to the nature of the problem formulation, specifically with the element of the objective function that maximises the gaps between allocations. This was discussed in the previous section.

The next section discusses how to decompose the problem so that the potential of the CPLEX solver is maximised. The RH method suits this purpose as it cuts out smaller sub-problems from the whole problem and uses CPLEX which is very fast at solving these sub-problems as long as they are small enough.

The datasets which cannot be solved to optimality: 3(1), 4(1), 5(1), 6(1), 4(3) are used to investigate the RH method further in Section 4.5.3.

4.5.3 RH Method with Various Parameter Configurations

The results obtained for the twenty configurations of the RH method given in Table 4.1 are presented in relation to the results presented in Section 4.5.2. The time limit of one hour is set for the total time of the RH method calculation. If no solution is found within an hour the RH method is stopped and since the configuration does not adequately improve upon the results obtained for the full solution, no solution is recorded. Each window is solved to optimality. The optimality gap is calculated for each RH objective result using Formulation 4.12:

$$100\% * \frac{rh - lb}{lb}, \tag{4.12}$$

where lb is the lower bound found by CPLEX when the full formulation is solved and rh is the RH objective value.

Table 4.6 shows the calculation times for each configuration, one column for one dataset. As previously mentioned when the calculation time of the RH method exceeds one hour no solution is given in the tables as the aim is to reduce the calculation times using the RH decomposition. The RH calculation time results have also the bipolar characteristic

Configuration Number	Day(Term) 3(1)	Day(Term) 4(1)	Day(Term) 5(1)	Day(Term) 6(1)	Day(Term) 4(3)
1	1.951	1.128	3.646	1.579	69.383
2	2.312	1.128	3.646	1.579	69.383
3	2.312	1.128	3.646	1.579	69.383
4	1.951	1.128	3.646	1.579	69.383
5	1.951	1.128	3.646	1.579	69.383
6	1.951	1.128	3.646	1.579	69.383
7	1.951	1.128	3.646	1.579	69.383
8	1.951	1.128	3.646	1.579	69.383
9	-	1.128	-	1.578	69.383
10	-	1.128	-	1.578	69.383
11	-	1.128	3.646	1.578	69.383
12	1.951	1.128	-	1.578	69.383
13	-	1.128	-	1.578	69.383
14	-	1.128	-	1.578	69.383
15	-	1.128	-	1.578	69.383
16	-	1.128	-	1.578	69.383
17	-	1.128	-	1.578	69.383
18	-	1.128	-	1.578	69.383
19	-	1.128	-	1.578	69.383
20	-	1.128	-	1.578	69.383

TABLE 4.4: Optimality gaps given in percent of the lower bound for various configurations of the RH method.

Configuration Number	Day(Term) 3(1)	Day(Term) 4(1)	Day(Term) 5(1)	Day(Term) 6(1)	Day(Term) 4(3)
1	76	76	76	76	66
2	61	63	63	61	51
3	53	53	53	51	41
4	44	47	43	46	35
5	51	52	51	49	39
6	39	43	38	39	29
7	31	33	30	30	24
8	26	28	25	26	19
9	-	41	-	38	27
10	-	30	-	27	21
11	-	24	22	22	16
12	18	20	-	17	13
13	-	31	-	28	21
14	-	23	-	21	16
15	-	18	-	17	13
16	-	15	-	14	11
17	-	25	-	23	16
18	-	18	-	16	12
19	-	15	-	14	11
20	-	12	-	12	9

TABLE 4.5: Number of window positions solved for each RH method configuration.

observed for the results obtained for the full formulation. The calculation times are strongly influenced by the size of the window. Only window sizes 5% and 10% of day duration could be solved for all datasets, with the larger window sizes being solvable (within one hour) only for some of the datasets. Apparently the larger windows meet the same problem as the full formulation. The calculation times are also influenced by the shift size. It can be observed from Table 4.6 for the first eight configurations (2 window sizes) that the times tend to be shorter for the larger shifts when the window size is fixed. This is related to the number of windows which need to be solved: when the shift is larger there are fewer window positions to be solved (see Table 4.5).

Table 4.4 displays the optimality gaps obtained for each configuration for each dataset. The obtained optimality gaps in most cases are the same as for the full formulation of the problem. The only exclusions are configurations 2 and 3 for dataset 3(1), which are worse than the objective value for the full formulation. No other influence of the RH method parameters on the objective values can be observed for the tested configurations.

Configuration Number	Day(Term) 3(1)	Day(Term) 4(1)	Day(Term) 5(1)	Day(Term) 6(1)	Day(Term) 4(3)
1	37.66	34.75	41.48	33.36	24.94
2	27.07	29.06	30.62	27.09	17.98
3	21.14	24.56	24.21	23.32	15.53
4	16.75	17.99	23.27	17.47	11.77
5	23.50	36.82	27.26	20.57	19.37
6	22.19	19.51	20.32	15.31	12.60
7	12.91	14.06	20.23	11.57	10.31
8	11.17	15.33	11.60	10.05	10.49
9	-	21.00	-	17.41	17.16
10	-	15.63	-	12.44	19.60
11	-	22.63	106.13	9.05	8.82
12	45.62	8.47	-	7.16	8.70
13	-	45.78	-	15.19	15.38
14	-	41.51	-	11.20	15.93
15	-	15.61	-	9.26	10.55
16	-	30.09	-	5.62	10.97
17	-	280.76	-	17.80	16.39
18	-	58.82	-	14.23	14.74
19	-	24.92	-	12.03	13.04
20	-	46.10	-	11.09	11.41

TABLE 4.6: Calculation times given in seconds for various configurations of the RH method.

The objective values which were found using the RH method are never better than those which were found using the full formulation. This is related to the way CPLEX finds the solution and its lower bound. It has been observed that CPLEX gets to the same objective value as the RH method quickly and uses the rest of the time to improve the lower bound. So, although CPLEX spends over an hour on the full model calculations the only improvement it makes after reaching the good objective value is the improvement of the lower bound. This lower bound is then used to calculate the optimality gap of both the RH method and the full formulation. Hence it cannot really be said CPLEX took an hour to find the same solution that the RH method as it actually finds the same objective value much quicker but it improves the probable lower bound a lot. In order to compare the times more fairly, another test was performed and its results are presented in Table 4.7. The table shows what the objective value for the full formulation is when the time limit for CPLEX is set to the longest time that the RH method took (the top part of Table 4.7) and the shortest time that the RH method took (the bottom part of Table 4.7). The table shows the optimality gaps but these are calculated using the same lower bound that the RH method uses (the lower bound found by CPLEX after one hour). Thanks to the results in Table 4.7 it can be shown whether the RH method reaches a better objective value than the full formulation within the same time. When the longest time RH took is used the results indicate that the same objective values were reached, however when the shortest times were used the objective values reached using the full formulation tend to be worse.

The RH method does actually improve the calculation times if its parameters are chosen correctly. Otherwise no improvement is noticed, for some configurations even the decomposed problem is not solvable within the chosen time limit of one hour. When the best configuration is to be chosen one of the first eight configurations should be taken. Configuration number eight would be a good choice, it takes on average 12 seconds to calculate the RH solution using this configuration.

4.6 Conclusion

The basic model of the GAP was presented in this chapter. The model behaviour was first investigated using one example dataset. The size of LG parameter which indicates the desired time gap between allocations has a huge effect upon the model calculation times. It was observed that only a small increase of the parameter size causes a sudden increase

Day(Term)	Longest RH Time[sec]	Optimality Gap[%]
3(1)	45.62	1.951
4(1)	280.76	1.128
5(1)	106.13	3.646
6(1)	33.36	1.579
4(3)	24.94	69.383
Day(Term)	Shortest RH Time[sec]	Optimality Gap[%]
3(1)	11.17	2.207
4(1)	8.47	1.128
5(1)	11.60	4.015
6(1)	5.62	1.889
4(3)	8.70	79.176

TABLE 4.7: Objective values obtained for the full formulation with time limits coming from the RH times.

of the calculation time.

The full model formulation was then solved for each of the available datasets using one value of LG which was set according to the airport preferences. Some of the datasets are solved within seconds while other could not be solved within an hour as the size of LG which causes the sudden raise of time is different for different datasets.

The RH decomposition method is described and applied for the datasets which cannot be solved to optimality within the one hour time limit set for the CPLEX solver. Twenty configurations of the RH method were tested. The obtained results show a strong relationship between the size of the window and the calculation time. The larger the window the longer it takes to solve the problem. The size of the shift influences also the calculation times but to smaller degree, configurations with larger shifts are faster to solve.

The objective values obtained for the full formulation and using the RH decomposition were almost always the same. The relationship between the size of the window or shift and the objective value reached cannot therefore be observed. It was however discussed that, when the full formulation is solved, the main part of the calculation time is taken to improve the lower bound of a solution which is found relatively quickly. It was therefore checked what objective value can be reached by the full formulation when the shortest and the longest time of the RH calculation for that dataset are used. This indicates that the RH method reaches the same objective values faster than the full formulation when appropriate parameters are used.

The preliminary results obtained for the RH method show the importance of the

parameter settings. The RH method seems to be very useful when the parameters are set appropriately. The results motivated to the further study on the method which is presented in Chapter 6. The decomposition will become much more important as the problems get harder.

Chapter 5

The Mixed Integer Programming Model of the Gate Allocation Problem Resolving Conflicts at the Gates

5.1 Introduction

One of the research directions which was identified in Chapter 3 was resolving the conflicts which occur in the area around the gates. Gates which are reached using similar routes should not be assigned to flights which are going to move at similar times as this is more likely to lead to congestion. This chapter focuses on modelling the gate allocation problem (GAP) taking into consideration possible conflicts in the area close to the gates. A new objective which aims to minimise the number of allocations which may lead to conflicts is proposed. Considering the possible conflicts in the early stage of allocation planning should result in smoother airport operation on the day of operation. It is a first step in the process of integration of the ground movement with the GAP, which has not been addressed before.

The model is tested on real world instances provided by Manchester Airport. Each terminal is solved separately, the whole airport problem can usually be decomposed in this way since it is very uncommon for an aircraft to be allocated to a different terminal than scheduled, since this would be very inconvenient for the airlines which usually have a specific agreement with an airport regarding the terminal they want to use and all of the necessary ground services would have to be moved to a different part of the airport.

However conflicting allocations may also occur between terminals, when for example gates belonging to two different terminals are placed opposite each other. One way of dealing with these conflicts would be to solve the terminals which have conflicting gates at

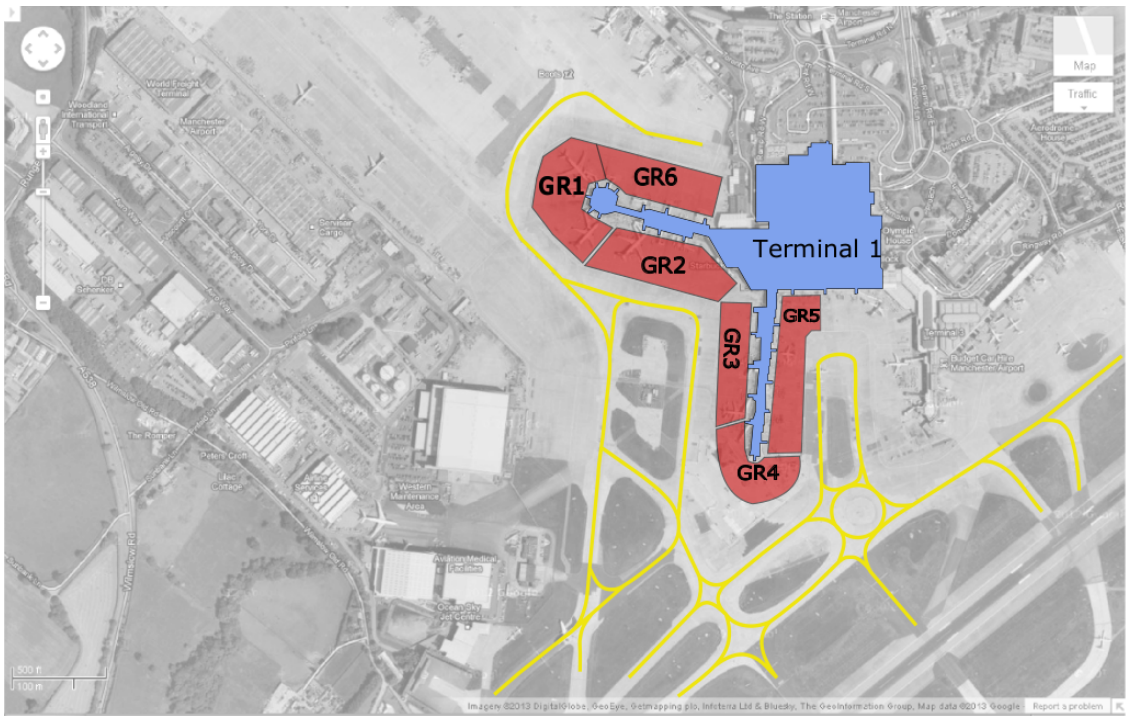


FIGURE 5.1: Map of Manchester Airport showing Terminal 1, the taxiways and groups of gates (Google satellite map, Imagery ©2013 TerraMetrics, Map data ©2013 Google)

once, however this makes the problem much larger and increases the need for a heuristic or decomposition method. The Receding Horizon method which was initially discussed in Chapter 4 is further investigated in Chapter 6 as a possible way of solving larger instances of the GAP.

This chapter is organised as follows: first the possible conflict detection mechanism is discussed. The MIP model of the problem is then presented. Next additional constraints which are applied in the model to reduce the search time are discussed. In the results in Section 5.5.2 the influence of the additional constraints on the calculation time is first shown. It is then discussed which element of the objective function slows the solver the most and finally the effects of the new elements of the objective function are validated.

5.2 Detection of Conflicting Allocations

Departing aircraft are routed from gates to runways. Similarly, arriving aircraft, once they are on the ground, must be routed from runways to gates. Several taxiways which can be used to get from one part of an airport to another are normally available, and the shortest

possible route with the minimum number of conflicts is usually chosen. Figure 5.1 shows an example satellite map [1] on which Terminal 1 of Manchester Airport is marked in blue and taxiways which lead to the gates on it are marked in yellow. All gates of the terminal are divided into groups which are marked in red in Figure 5.1. Gates which are neighbouring and are reached using the same final part of the access taxiway have been allocated to the same group. A conflict is likely to occur if two flights which are scheduled to make a move at a similar time are allocated to the same group of gates. Hence these allocations are limited in the presented model. The gate-group adherence for Terminal 1 is as follows:

- GR1: 29, 32, 31
- GR2: 21, 23, 25, 27
- GR3: 2, 4, 6, 8
- GR4: 10, 12, 13, 14
- GR5: 1, 5, 7, 9, 11, 15
- GR6: 22, 24, 26, 28

Examples of physical blockages which can be included in the early stage of planning are push-back and taxi blocking. The push-back blocking (Figure 5.2) is observed at the area around the gates and it occurs when a departing aircraft is ready to depart a gate but it is blocked by another aircraft which is currently pushing back. In Figure 5.2 aircraft $F1$ is blocked by aircraft $F2$, one of the two aircraft has to wait until the area is cleared. There are two possible ways of solving the conflict. $F1$ is prioritised, it pushes back first and delays $F2$ as shown in Figure 5.3a. Alternatively $F2$ proceeds first and stops $F1$ pushing back like it is shown in Figure 5.3b.

Figure 5.4 shows an example of taxi blocking, which occurs when two aircraft are taxiing in opposite directions along the same route or when two aircraft meet at a crossing point at the same time. This type of blocking occurs both next to the gates but also further away along the taxiways, especially close to bottlenecks (places where several routes converge). An example of a bottleneck can be observed in Figure 5.1, gates of groups $GR1$ and $GR6$ are placed behind it. Number of flights planned to pass a bottleneck at the same

[1] Google satellite map, Imagery ©2013 TerraMetrics, Map data ©2013 Google

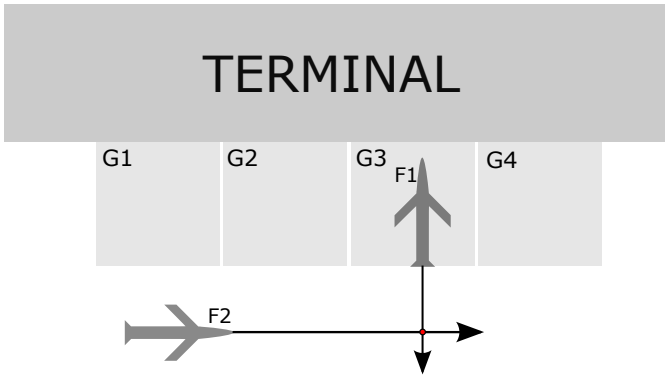
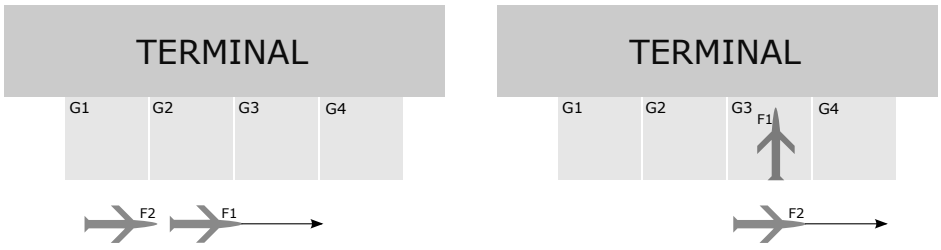


FIGURE 5.2: Push-back blocking: $F1$ in conflict with $F2$



(a) $F1$ pushes back first (b) $F2$ proceeds first

FIGURE 5.3: Push-back blocking: priority of aircraft

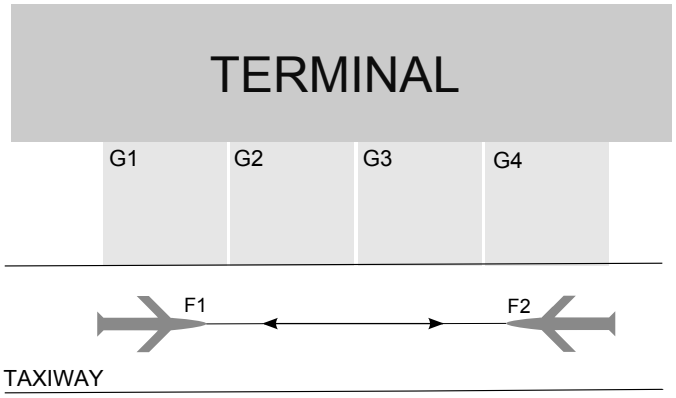


FIGURE 5.4: Taxi blocking

time could be limited in order to reduce congestion similarly to the conflicting allocations. This has not been included in the experiments of this chapter but it is considered as future work.

5.3 MIP Model Description

A MIP model which considers ground movement issues is presented in this section.

5.3.1 Notation and Definitions

Table 5.3.1 includes the notation and definitions.

5.3.2 Developed Model

The MIP model extends the basic model presented in Chapter 4, includes the same basic core constraints and the gaps between allocations are maximised using the same method. The model aims to solve an early stage allocation planning problem which takes into consideration the conflicts at gates. The idea of dividing gates into groups is used in the model and the number of conflicts within groups is minimised in the objective function.

The objective function contains also other important constraints, which were discussed before in the related literature (see Chapter 2). The GAP has a multi-objective character which is modelled in this chapter using a weighted sum of the objectives. All of the elements of the objective function and their weights are described in more detail using

F	set of flights
n	total number of flights in F
i, j	flight indices $i, j \in \{1, ..., n\}$
$f_i \in F$	a flight with index i
e_{f_i}	on-gate time of f_i, a constant for each f_i
l_{f_i}	off-gate time of f_i, a constant for each f_i
G	set of gates
m	total number of gates available in G
k, l	gate indices $k, l \in \{1, ..., m\}$
$g_k \in G$	a gate with index k
$F(g_k)$	subset of flights that can use g_k
Sh	set of pairs of gates that cannot be used simultaneously
GR	subset of gates used when minimising conflicts
z	indices of subsets of gates
NC	number of conflicting flights allocated to one GR, a variable
X_{g_k,f_i}	decision variable, it is 1 if f_i is allocated to g_k, 0 otherwise
U_{g_k,f_i,f_j}	indicator variable, it is 1 if f_i and f_j are allocated to g_k, 0 otherwise
gap_{g_k,f_i,f_j}	time gap between f_i and f_j when both are allocated to gate g_k
SG	minimum size of gap_{g_k,f_i,f_j}, a constant
LG	maximum size of gap_{g_k,f_i,f_j} , a constant
p_{f_i,f_j}	penalty for putting f_i and f_j on the same gate
r_{f_i,g_k}	penalty for putting flight f_i on gate g_k dependent upon sizes
a_{f_i,g_k}	penalty for putting flight f_i on gate g_k related to airline preferences
du	penalty for 'dummy gate'
$OVERNIGHT$	set of flights which stayed overnight at their assigned gate
$ghist_a$	gate that was used by flight a in the historic data
$C_d(f_i)$	set of flights which may be in conflict with the departure time of f_i
$C_a(f_i)$	set of flights which may be in conflict with the arrival time of f_i
$freq_{f_i,g_k}$	function indicating how often the airline for f_i has used g_k in the historic data

TABLE 5.1: Notation and definitions

mathematical formulations in Section 5.3.4. A short analysis of the impact of each of the elements of the objective function on the calculation time is provided in the results, since it is important in order to better understand the model behaviour.

The constraints included in the model are the core constraints which have been discussed before in Chapters 3 and 4 and additional constraints which result mainly from the gate attributes, i.e. aircraft and gate sizes, shadowing restrictions and security requirements. Section 5.3.3 gives a mathematical description of the constraints and explains each of them in more depth. The proposed terminal-based decomposition allowed the solution of most of

the instances used in the experiments to be found. Several additional elements have been added to the model in order to speed up the solution process and they are discussed in Section 5.3.5. The effects they have on the calculation time are discussed in Section 5.5.2.

5.3.3 Constraints

Equation 5.1 limits the gate usage so that flights are allocated only to gates that they will fit on and can use. As was described in Chapter 4, the time gap gap_{g_k,f_i,f_j} between two flights f_i and f_j which are allocated to the same gate g_k is defined by Equation 5.2. The variable U_{g_k,f_i,f_j} becomes one if and only if f_i and f_j belong to $F(g_k)$(Equation 5.4), and the size of the time gap between them gap_{g_k,f_i,f_j} is between SG and LG. Gaps larger than LG are ignored in the objective function (Equation 5.3).

$$X_{g_k,f_i} = 0,\ \forall f_i \notin F(g_k), \forall g_k \in G \tag{5.1}$$

$$gap_{g_k,f_i,f_j} = \begin{cases} e_{f_j} - l_{f_i} & if\, e_{f_j} \geq e_{f_i} \\ e_{f_i} - l_{f_j} & if\, e_{f_i} \geq e_{f_j} \end{cases},\ f_i, f_j \in F(g_k),\ g_k \in G,\ f_i \neq f_j \tag{5.2}$$

$$\left.\begin{array}{ll} U_{g_k,f_i,f_j} \geq X_{f_i,g_k} + X_{f_j,g_k} - 1 & \forall f_i, f_j \in F(g_k),\ \forall g_k \in G,\ SG < gap_{g_k,f_i,f_j} < LG, \\ U_{g_k,f_i,f_j} \geq 0 & 0 \leq SG \leq LG \end{array}\right\} \tag{5.3}$$

$$U_{g_k,f_i,f_j} = 0,\ \forall (f_i\ or\ f_j) \notin F(g_k), \forall g_k \in G \tag{5.4}$$

$$\sum_{k=1}^{m+1} X_{g_k,f_i} = 1, \quad \forall f_i \in F \tag{5.5}$$

$$^2\sum_{i=1}^{n}\sum_{k=1}^{m+1} X_{g_k,f_i} = n \tag{5.6}$$

$$X_{f_i,g_k} + X_{f_j,g_k} \leq 1,\ \forall f_i, f_j \in F(g_k),\ \forall g_k \in G,\ gap_{g_k,f_i,f_j} \leq SG,\ SG \geq 0 \tag{5.7}$$

[2]After the research had been conducted this constraint was found to be redundant. Constraint 5.5 is sufficient to force each flight to be allocated to a gate.

$$X_{a,ghist_a} = 1,\ \forall a \in\ OVERNIGHT \tag{5.8}$$

$$X_{f_i,g_k} + X_{f_j,g_l} \leq 1, \quad \forall f_i \in F(g_k),\ \forall f_j \in F(g_l), \quad \forall (g_k, g_l) \epsilon Sh \tag{5.9}$$

$$X_{f_i,g_k} + \sum_{g_k \epsilon GR} \sum_{f_i \in C_d(f_i)} X_{f_i,g_k} \leq NC, \quad \forall f_i \in F(g_k),\ \forall g_k \in GR, \quad \forall GR \in \{GR_1, ..., GR_z\} \tag{5.10}$$

Constraints 5.4 to 5.8 are analogous to the constraint in the basic model defined in Section 4.2 of Chapter 4. The model includes sizes of flights and gates and hence the constraints are more restrictive, for example variable X_{f_i,g_k} can only become 1 when f_i belongs to $F(g_k)$ which is new compared with the basic model.

Each flight has to be allocated to a gate (Constraint 5.6). Each flight is allocated to only one gate thanks to Constraint 5.5. Two flights which overlap or are within the minimum gap SG of each other are not allowed to use the same gate (Constraint 5.7). The overnight stays are automatically allocated to the gates that were used in the historic record (Constraint 5.8).

Constraint 5.9 and Constraint 5.10 are new compared with the basic model from Chapter 4. Constraint 5.9 is a shadow constraint and it refers to the gates which cannot be used simultaneously. One of the gates is blocked when the other is used by a large aircraft for example. The constraint concerns also the situations when one large gate can be used either as one or two smaller gates. The large gate is modelled as three separate gates with appropriate shadow constraints between them. The last Constraint 5.10, considers the conflicts at the gates. Flights which may be in conflict with a departure time of flight f_i (due to a similar arrival time) create set $C_d(f_i)$ and are allocated to different groups of gates if possible. An analogous constraint for flights which may be in conflict with the arrival time of f_i (due to a close departure time) also exists in the model, it utilises set $C_a(f_i)$ instead of $C_d(f_i)$. The time margin value which has been used to determine the possible conflicts in this work is given in Table 5.2 (last row). NC is the number of conflicting flights which are allowed to be allocated to the same group of gates and is minimised in the objective function. This means that no more than NC flights will use each conflicting area in each time period.

5.3.4 Objectives

The objective function is given by Formula 5.11 which is minimised in the model.

$$\begin{aligned} Min \quad & \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{m} p_{f_i,f_j} U_{g_k,f_i,f_j} + \sum_{i=1}^{n}\sum_{k=1}^{m} r_{f_i,g_k} X_{f_i,g_k} \\ & + \sum_{i=1}^{n}\sum_{k=1}^{m} a_{f_i,g_k} X_{f_i,g_k} + NC + \sum_{i=1}^{n} du X_{f_i,dummy} \end{aligned} \tag{5.11}$$

$$p_{f_i,f_j} = \frac{LG}{(gap_{f_i,f_j})} \quad if\ SG < gap_{g_k,f_i,f_j} < LG \tag{5.12}$$

$$r_{f_i,g_k} = \frac{maxSize(g_k) - size(f_i)}{biggestGateSize} \tag{5.13}$$

$$a_{f_i,g_k} = 1 - \frac{freq_{f_i,g_k}}{maxFreq}, \quad maxFreq \neq 0 \tag{5.14}$$

There are two additional elements in the objective function which will be discussed later and are weighted to have a very small impact on the optimal objective value and are described in the next section.

The first element aims to maximise the time gaps between flights which are allocated to one gate. Variable U_{g_k,f_i,f_j} (defined by Equation 5.3 and Inequality 5.4) is weighted in the objective function using the cost function p_{f_i,f_j}, given by Equation 5.12. Maximisation of the time gaps helps in achieving a more robust plan which absorbs minor delays which often occur on the gates during the day of operation.

The second element is related to an effective usage of gates. The size of a flight is matched with the size of the gate it uses, avoiding situations where large gates are used by small flights. This improves the robustness of the plan as the large gates are left free, where possible, and are therefore available when a reallocation is needed during the day of operation. This element is costed using a gate-flight size function which is described by Equation 5.13. Allocations are weighted in the objective function depending upon the size similarities, where $biggestGateSize$ is the size of the largest gate at the terminal, $maxSize(g_k)$ the maximum size of flight the gate can accommodate and $size(f_i)$ is the size of flight f_i.

The airline preferences are expressed by the third element of the objective function. It was established how often the airline of flight f_i used gate g_k ($freq_{f_i,g_k}$). Appropriate data analysis was done which is described in Section 3.3.3, Chapter 3. The frequency is used

in Equation 5.14 to calculate weights for this element. $maxFreq$ is the maximum value of all frequencies.

The fourth element, NC, refers to the number of conflicting flights in each group of gates (see Inequality 5.10). NC is a single value which is minimised across all groups of gates, which can be understand as minimising the maximum value. When it equals two each of the groups can have up to one conflict (i.e. one pair of flights in conflict) within each specified time duration.

The fifth element minimises 'dummy gate' allocations. The 'dummy gate' represents holding positions and remote stands. When an aircraft is allocated to a remote stand passengers have to be transported to and from a terminal building usually by bus. The remote stands are usually used only on a special request of an airline or when all gates are occupied.

The two additional elements of the objective function, which are not given by Formula 5.11 are described in Section 5.3.5. They have a marginal influence on the final objective value, as they have been given very small weights. One of them prioritises slightly the gates which are closer to taxiways. The other adds a small weight to the objective when an allocation of flight a to gate b occurs in the solution, where the weight is calculated based upon the indexes of flight a and gate b. They both aim to break symmetries in the search tree (see Chapter 2).

Using the weighted sum of the objectives is the most common way of modelling the objective function of the GAP, but results in the optimal solution strongly depending upon the weights which are used. In this research, the weights of the elements are fixed, and are chosen to match the preferences of the airport. The primary objective is that all flights are allocated to gates not to the remote stands if possible, the remote allocations therefore have the highest penalty in the objective function. The secondary objective is the size preference, which provides effective usage of the gate spaces. The tertiary objective is the airline preference. Next is the objective which aims to maximise the time gaps between allocations and finally the one which reduces the number of conflicts at the gates (even with the low preference it still makes a difference as will be seen in Section 5.5.4). The two additional objectives (see Section 5.3.5) have a minor influence on the final objective value.

5.3.5 Additional Elements of the Model which Speed up the Calculations

The two additional elements of the objective function, mentioned above, and a set of constraints have been added to the model in order to shorten the time needed to achieve the optimal solution.

Symmetries in a model are an important issue for many scheduling problems (see Section 2.5, Chapter 2). They may also occur in the presented GAP formulation.

An example of symmetry in our model would be two gates which have exactly the same parameters. Each of the gates is used by several flights (subset of flights) during each day. A subset of flights which can be allocated to either of the gates with the same cost would introduce symmetry in the model. Since the model formulation includes size and airline preferences it is less likely to find symmetries in the formulation than it was in the basic model formulation (see Chapter 4). However, even small numbers of symmetries can adversely affect the calculation time. Two additional symmetry breaking elements have therefore been added to the objective function. The effects on the calculation times of adding these have been tested and the results are presented in Section 5.5.2. These elements consist of:

a) A small weight for each allocation which is calculated based upon the flight index and the gate index and is given by Formula 5.15:

$$\frac{i}{n^2} + \frac{k}{m^2}, \tag{5.15}$$

where i is a flight index and k is a gate index.

b) A small weight which mirrors the distance of the gate to the exit point of a group of gates. The closer the gate is to the exit point the easier it is to access it. Hence the gates which are closer to the exit point are slightly prioritised.

5.4 Parameter Setting & Data

The settings which were used in the following experiments are given in Table 5.2. Datasets from three terminals of Manchester Airport, which are described in Chapter 3, Section 3.3.1 were used to acquire the results presented in Section 5.5.

A time limit of 10 hours was set for the calculation time of the model. If the model did not reach the optimal solution after the 10 hours, only a feasible solution was recorded.

TABLE 5.2: Model settings

Name	Symbol	Value
Size of the gap for which penalty p saturates	LG	60 min
Penalty for remote allocation	d	60
Minimum size of the gap between flights	SG	15 min
Time margin for conflicts within group of gates		5 min

5.5 Model Performance

This section presents several aspect of the model performance. Firstly a graphical representation of the model results is shown in Section 5.5.1. It shows which flights have been allocated to which gates in a similar way as illustrated by the user support systems at the airports (see Section 3.3.4, Chapter 3). Section 5.5.2 is focused on calculation times. Section 5.5.3 considers the way in which particular elements of the model influence the times. Finally Section 5.5.4 is dedicated to the new element of the model, the conflict constraint, investigating how the conflict constraint changes the number of expected conflicts in the obtained allocations. The results are also compared against the recorded allocations in the historical data.

5.5.1 Graphical Illustration

Twenty one datasets (seven days, three terminals) were solved using CPLEX. Four of them could not be solved to optimality within 10 hours, only a feasible rather than optimal solution being obtained for them.

The results can be displayed on charts. The charts can be very useful in the initial stages of the model creation (when new constraints are still being added) as they allow some of the aspects of the results to be quickly validated. Figure 5.5 shows an example allocation obtained for day one for terminal T1, which includes 111 flights. The names of the gates are indicated in the vertical direction (-1 is used for remote allocations) and the time flow is given in the horizontal direction of the chart.

Flights are shown as chains of markers. Different markers are used for different groups of gates. For the example given in Figure 5.5 the groups match the groups shown in Figure 5.1. Squares are used for GR1 (■, effectively rectangular bars), x-shaped markers for GR2 ($\times$), star-shaped markers for GR3 ($*$) , down triangular-shaped markers for GR4

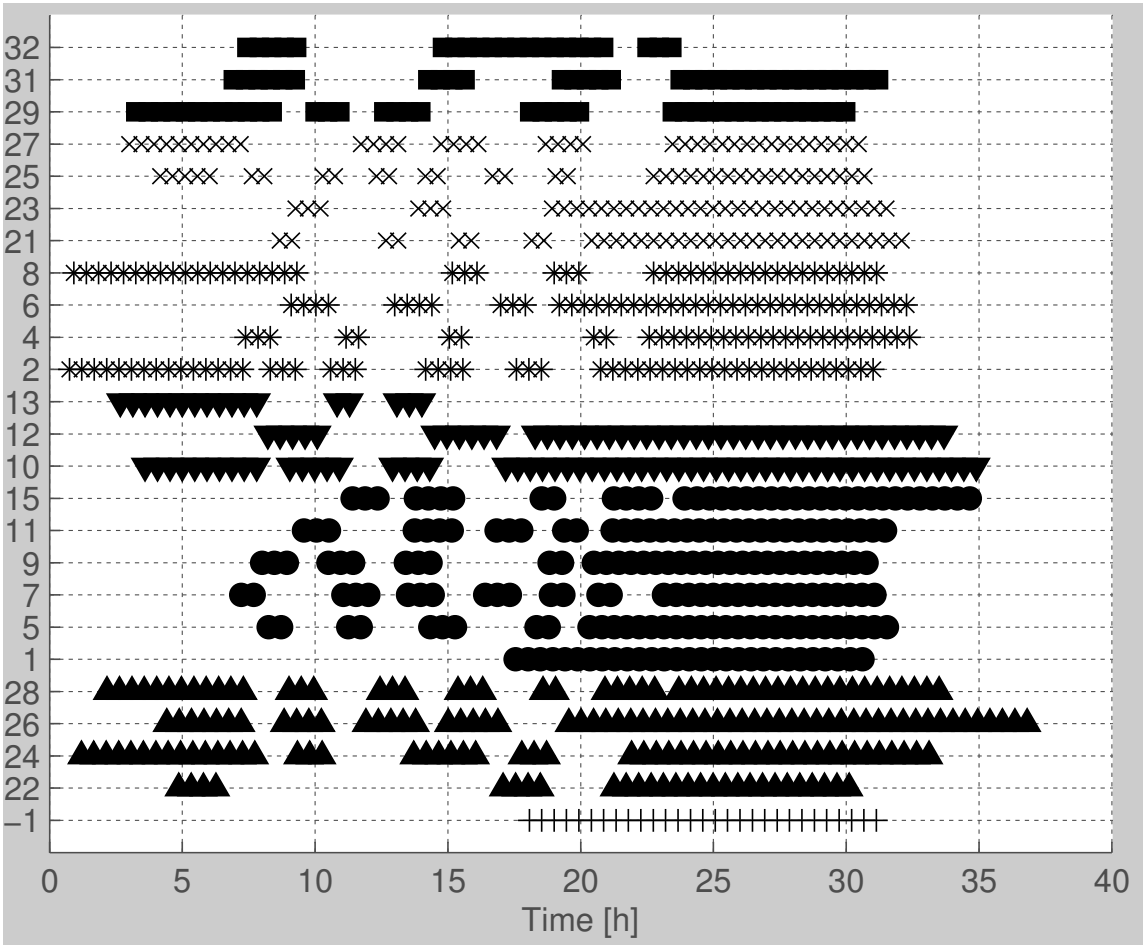


FIGURE 5.5: A graphical representation of an allocation, sorted by gate group.

(▼), dots for GR5 (•), up triangular-shaped markers for GR6 (▲) and cross-shaped markers for remote allocations (+).

The graphical representation of an allocation can for example be used to check whether the shadow constraint works correctly. A pair of gates 12 and 13 is an example for which the shadow constraint applies on terminal T1. The two gates cannot be used simultaneously as gate 13 is in fact the left part of large gate 12. It is visible in Figure 5.5 that the constraint works correctly and the two gates are never used at the same time.

5.5.2 Speeding up the Search

The additional elements which break the symmetry in the model were introduced to speed up the calculation process. This section illustrates what the influence of these elements is on the calculation times. Results obtained for the model with and without the element which uses indexes of flights and gates (point (a) in Section 5.3.5) are first compared. The ratio α defined by Equation 5.16 is used in the comparison.

$$\alpha = \frac{a}{b} \tag{5.16}$$

where a is the model calculation time without the element which uses indexes of flights and gates and b the full model calculation time. Observe that:

$$\begin{aligned} \alpha > 1 \Leftrightarrow a > b, \\ \alpha < 1 \Leftrightarrow a < b. \end{aligned} \tag{5.17}$$

Table 5.3 presents results which were obtained using the various datasets, one dataset corresponding to one day of one terminal. Calculation times for the full model are given in the second column, and the ratio α is given in the third column. The influence of the second element which was introduced to break symmetry and is based upon the distance to the exit (point (b) in Section 5.3.5) was also tested. Ratio β is used in the comparison in manner analogues to α. For β the value a in Equation 5.16 is the calculation time of the model without the second symmetry breaking element. The β values are given in the fourth column of Table 5.3. The fifth column (α & β) of Table 5.3 presents the calculation times obtained when both of the symmetry breaking constraints are applied.

It is hard to predict the calculation time for an instance before the evaluation. So, the decision about keeping or discarding elements of the model which may reduce the calculation times is often very tricky. It may happen that for some instance the additional constraint helps while for another it actually slows down the calculations. The results shown in Table 5.3 indicate that applying both of the symmetry breaking constraints gives the best results. The calculation times improve or stay the same for the model with the additional constraints, most of the values in the fifth column (α & β) of Table 5.3 are equal or larger than one. Only in two cases (1(3), 2(3)) was the calculation significantly longer after applying the symmetry breaking constraints. This is a minor loss considering the gains from other cases. It is worth to keep both of the symmetry breaking elements in the model.

Day(Terminal)	b[s]	α	β	α & β
1(1)	153	0.9	1.0	1.3
2(1)	171	1.0	1.0	1.3
3(1)	36000	1.0	1.0	1.0
4(1)	239	1.1	1.0	1.3
5(1)	36000	1.0	1.0	1.0
6(1)	198	1.0	1.0	1.4
7(1)	5902	0.5	1.0	0.7
1(2)	7	0.8	0.8	1.0
2(2)	13	0.9	1.0	1.2
3(2)	12	0.9	0.9	1.0
4(2)	13	0.9	0.9	1.0
5(2)	20	0.9	0.8	1.0
6(2)	15	0.8	0.8	0.9
7(2)	6	0.8	0.8	1.0
1(3)	29	1.2	1.1	0.4
2(3)	443	0.7	0.2	0.2
3(3)	1669	1.0	1.7	1.5
4(3)	36000	1.0	1.0	1.0
5(3)	36000	1.0	1.0	2.0
6(3)	18	0.9	0.9	1.1
7(3)	226	1.0	0.7	0.9

TABLE 5.3: Calculation times for model with and without the elements which were introduced to speed up the calculation process.

5.5.3 Objective Function Analysis

The analysis shown in this section aims to establish which of the elements of the objective function have on average the strongest influence on the calculation time. Six simplified versions of the objective function given by Formula 5.11 are prepared for that purpose:

- obj1 - objective function with only the first and the last element of Formula 5.11, this aims only to maximise the gaps between allocations,
- obj2 - objective function with only the second and the last element of Formula 5.11, this aims only to maximise the size preferences,
- obj3 - objective function with only the third and the last element of Formula 5.11, this aims only to maximise the airline preferences,
- obj4 - objective function with only the fourth and the last element of Formula 5.11, this aims only to minimise the conflicts around the gates,
- obj5 - objective function with only the symmetry breaking element which uses indexes of flights and gates (see point (a) in Section 5.3.5),
- obj6 - objective function with only the symmetry breaking element which uses distances to exit (see point (b) in Section 5.3.5).

All of the constraints were kept the same, only the objective function was modified. The last element of Formula 5.11 penalises the dummy allocations and has to always be in the objective function as without it all flights would immediately be allocated to the remote stands.

Experiments were performed with each of the simplified objective functions and the calculation times were recorded. Seven days of airport operation (similarly to the previous experiments) were used. The recorded times were compared against the time taken by the full formulation and an average relative calculation time was then calculated across all of the datasets for each of the simplified objective functions. Figure 5.6 shows a bar plot which compares the average relative calculation times taken by the six simplified objective functions, one bar for one version of the objective function.

It is clearly visible that the most time consuming element of the objective function is the element which maximises the time gaps between allocations (obj1). This increases the

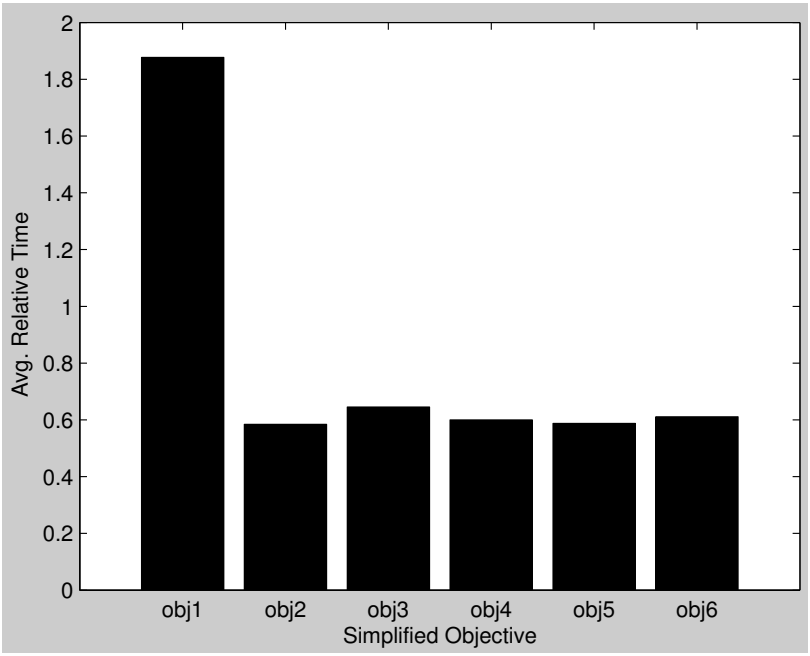


FIGURE 5.6: Times taken by the six simplified versions of the objective function.

model complexity a lot as it requires setting variables for pairs of flights on gates (U_{g_k,f_i,f_j}) rather than only for flight-gate pairs (X_{f_i,g_k}). Moreover, it is clear that its calculation time is on average almost three times as long as the calculation time of the full formulation. This indicates that adding more objectives to the model makes the search process easier for the solver. All other versions of the objective function result in similar times. This is expected, as none of these introduces the flight-flight-gates variables, only the flight-gate pair variables have to be considered.

5.5.4 Influence of the Conflict Constraint

The effects of the conflict constraint are investigated in this section. The number of possible conflicting allocations is checked for results obtained using the model with and without the conflict constraint. Additionally the number of conflicting allocations detected in the historical data records is given as a comparison.

Table 5.4 displays the total number of possible conflicts obtained for each terminal using the datasets described in Chapter 3 (seven days of Manchester Airport operation). The first column shows the number of conflicts (aircraft which may delay each other) for

Terminal	Model Solution		Recorded Solution
	With Conflict Constraint	Without Conflict Constraint	
T1	51	74	95
T2	19	51	50
T3	41	48	77
SUM	111	173	222

TABLE 5.4: The total number of conflicts calculated for the allocations obtained using the model (with and without the conflict constraint) and recorded in the historical data

Terminal	Model Solution		Recorded Solution
	With Conflict Constraint	Without Conflict Constraint	
T1 & T2	91	146	165
T1 & T3	92	122	186
T2 & T3	60	99	127
T1 & T2 & T3	132	194	256

TABLE 5.5: The total number of conflicts calculated for merged allocations obtained using the model (with and without the conflict constraint) and recorded in the historical data

the model with the conflict constraint, the second column shows the number for the model without the conflict constraint and those in the last column are for the recorded allocations. The first three rows of Table 5.4 show each terminal separately, the last row shows the sum of the conflicts. As expected, the number of conflicts drops after adding the conflict constraint, and is significantly lower than the number of conflicts in the historical record as well. The planned allocation should therefore result in smoother airport operations during the day of operation thanks to the added constraint.

When each terminal is solved separately, conflicts which may occur between gates belonging to different terminals are not included. The allocations obtained for two or three terminals separately can be merged and the number of conflicts calculated using the merged allocations will also include the conflicts occurring between terminals. In the tested instances terminal T1 and terminal T2 have conflicting gates. There are no conflicting gates between terminal T1 and terminal T3, however the two terminals have one common gate. This common gate in the model is assigned to one terminal but in the records it could be used by either of them. Terminal T3 is on the other side of the airport from terminal T2 so there is no physical relation between the two, no additional conflicts should therefore be detected in the merged allocation T2 & T3.

Table 5.5 shows the numbers of conflicts obtained for merged allocations for all

possible terminal combinations (T1 & T2, T1 & T3, T2 & T3, T1 & T2 & T3). More conflicts were detected in merged T1 & T2 & T3 than the total number of conflicts detected separately for the three terminals (compare the last row of Table 5.4 with the last row of Table 5.5) due to the conflicts between terminals. Gates at terminal T2 and terminal T3 do not have any physical relation and therefore the sum of the conflicts calculated separately is always equal to the number of conflicts detected for the merged allocation T2 & T3 (sum of rows two and three of Table 5.4 is equal to row three of Table 5.5). In the model solutions the sum of conflicts obtained for T1 and T3 separately is equal to the number of conflicts found for the merged solution T1 & T3 (sum of row one and two of Table 5.4 is equal to row two Table 5.5). This confirms that there are no conflicting gates between the two terminals. However it can only be observed for the model results not for the records because of the commonly used gate which is shared between terminal T1 and terminal T3.

The results prove that solving terminals separately is a valid idea only if the terminals don't have any physical connection, like terminals T2 and T3. If there is a relationship between terminals, they should probably be treated as one problem and solved together when the conflicts around the gates are attempted to be resolved.

5.6 Conclusion

A MIP model of the GAP which takes into consideration possible conflicts at the area around gates was presented in this chapter. A full mathematical formulation which includes the implemented model constraints and objectives is discussed in detail.

Twenty one datasets which come from Manchester Airport (seven days of operation) were solved using the CPLEX solver. A graphical implementation of the results is presented, which is similar to those which are used by the support systems at airports. It is particularly useful when the model is validated, to check if the outputs are correct.

The symmetry breaking elements of the model, which were added to shorten its calculation time, were tested. They appeared to be very useful when both of them are kept in the model.

The elements of the objective function were also considered in the experiments. The influence of each of them on the calculation time was tested. It became apparent that the objective element which aims to increase the gaps between allocations has the heaviest influence on the calculation time. This is due to the increase in the number of

variables which are considered in the search process as not only flight-gate pairs but also flight-flight-gate configurations need to be checked.

Finally the effects of the new conflict constraint on the results of the model were observed. It was proven that the conflict constraint does actually reduce the number of expected conflicts. The second part of the conflicts analysis showed that some problems between terminals cannot be tackled when the terminals are solved separately, i.e. conflicts occurring between gates belonging to different terminals and commonly used gates. Therefore it would be good to solve more than one terminal at the same time, ideally to solve all terminals simultaneously. However the exact methods used in this chapter become very slow and consume lots of memory when larger problems are solved. In fact even for some of the single terminal instances only feasible solution was found after 10 hours.

Therefore the alternative solution method based upon the Receding Horizon approach introduced in Chapter 4 is further investigated in Chapter 6 as an alternative way of solving large instances of the model formulation.

Chapter 6

The Receding Horizon Approach to the Gate Allocation Problem Parameters Analysis

6.1 Introduction

This chapter presents an in depth analysis of the receding horizon (RH) approach applied as a decomposition method of the advanced gate allocation problem (GAP).

The main contribution of the research presented in this chapter is the application of several versions of the RH method to decompose the advanced model of the GAP (described in Chapter 5) and investigation of what the best parameter settings are for the method. Among the tested versions of the RH method a new, dynamic version has been introduced and has shown good results. The presented results can be useful when the RH method is used to decompose other problems (especially other resource allocation problems), which increases the importance of the research.

It is shown in Chapter 5 that sometimes conflicts may occur between gates belonging to two different terminals, for example when two gates of two different terminals are placed opposite each other. It would therefore be beneficial to test if solving more than one terminal at the same time allows the resolution of more conflicts. This is investigated and the results are described in the final part of Section 6.4.

This chapter starts with a description of the four versions of the RH method which have been implemented and tested (Section 6.2). The description is followed by Section 6.3 where the parameter settings which have been used in the experiments are discussed. Results of the experiments are then provided in Section 6.4 and the chapter ends with conclusions in Section 6.5.

6.2 Overview of Implementations

This section explains the various implementations of the Receding Horizon (RH) approach. It was explained in Chapter 4 how the Receding Approach works in general. Only one way of extracting the subsets of flights was discussed in Chapter 4, namely the static time oriented implementation. In this chapter three more implementations are presented and compared: static flight-oriented decomposition, dynamic time-oriented decomposition and dynamic flight-oriented decomposition.

6.2.1 Notation

The notation is the same as given in Chapter 4, several definitions which are crucial for this chapter are repeated below.

T	size of window
d	size of window shift
$lastFlight$	last flight in the dataset
$pozNumb$	number of positions of window
p	position index $p \in \{1, ..., pozNumb\}$
FT_p	subset of flights, their arrival times are in T for position p
Fd_p	subset of flights, their arrival times are in d for position p
$MIP(s)$	MIP solution for subset s of the flights
i	flight index $i \in \{1, ..., n\}$

6.2.2 Description of Implementations of the Receding Horizon Method

Four different ways of extracting the subsets of flights to use in a window were implemented and tested.

Decomposition Based Upon Time

The first approach is time-oriented decomposition, it was used in the preliminary study and is already described in Chapter 4. Hence only a short description is given in here. A flight is included in a window depending upon its arrival time. Equation 6.1 gives the number of window positions. Inequality 6.2 and Inequality 6.3 define the subset of flights in window T and the subset of flights in shift d respectively.

$$pozNumb = \left\lceil \frac{e_{lastFlight} + d - T}{d} \right\rceil \tag{6.1}$$

$$f_i \in FT_p \;\; if \;\; p * d < e_{f_i} \leq ((p * d) + T) \tag{6.2}$$

$$f_i \in Fd_p \;\; if \;\; p * d < e_{f_i} \leq ((p + 1)d) \tag{6.3}$$

Decomposition Based Upon Number of Flights

The second way of creating the subset of flights is based upon the number of flights in a window. The sizes of T and d are basically measured in number of flights. When the number of flights in the data is insufficient to fill the last window position the last subset of flights has a smaller number of flights. It is assumed that the flights are ordered according to increasing arrival times of flights (the smallest index i is given to the earliest flight) for the following equations. Equation 6.4 defines the number of window positions which have to be solved. Inequality 6.5 specifies the subset of flights in the window T. Inequality 6.6 defines the subset of flights in shift d.

$$pozNumb = \left\lceil \frac{n + d - T}{d} \right\rceil \tag{6.4}$$

$$f_i \in FT_p \;\; if \;\; p * d \leq i < ((p * d) + T) \tag{6.5}$$

$$f_i \in Fd_p \;\; if \;\; p * d \leq i < ((p + 1)d) \tag{6.6}$$

Dynamic Variation of the Decompositions

Modified versions of the two decompositions described above are proposed in this section. In the modified versions the size of the window is not fixed any more, it changes dynamically. It is different for each window position and depends upon the flights included in the shift size. The size of the shift is defined exactly as before (Equation 6.6 or Equation 6.3) based on either a fixed number of flights (the flight-oriented decomposition) or fixed time duration (the time-oriented decomposition).

For the dynamic version of the flight-oriented decomposition starting from the given position the next d flights are included in the window, as are all later flights which overlap with any of these flights. Fd_p is defined by Equation 6.6. Flights included in FT_p depend upon the flights in Fd_p. It is checked which future flights overlap with the flights from set Fd_p and the flights which overlap are added to set FT_p. Set FT_p is defined by Expressions 6.7 and 6.8:

$$f_i \in FT_p \ \ if \ (p * d \leq i \wedge f_i \in OverlapFd_p), \tag{6.7}$$

$$f_i \in OverlapFd_p \Leftrightarrow (\exists f_j \in Fd_p) \ \ l_{f_j} \geq e_{f_i}, \tag{6.8}$$

where $OverlapFd_p$ is a set of flights which overlap with flights in Fd_p. The total number of positions which are solved in the dynamic version of the decomposition is not known explicitly and depends upon the specific problem instance.

A dynamic version of the time-oriented decomposition is created analogously to the dynamic version of the flight-oriented decomposition. The idea of creating the dynamic versions occurred during initial tests performed using the non-dynamic decompositions. It was considered that including all of the overlapping flights from the future could result in improvement of times and quality of the solutions. This hypothesis has been tested experimentally and the relevant results are presented in Section 6.4. One advantage of the dynamic versions is the smaller number of parameters: only the shift size has to be set explicitly, which simplifies the tuning of the RH method.

Figure 6.1 which is also shown in Chapter 4 provides a Gantt chart to visualise the general idea of window positioning and shifting. Flights are symbolised by the rectangles ($F1, F2, F3, F4, F5$). Two window positions ($p = 1, p = 2$) and two subsets of flights Fd_p and FT_p are shown in Figure 6.1. When the static implementations are applied the subsets which are shown in Figure 6.1 contain the following flights respectively:

- $Fd_1 = \{F1, F2, F4\}$
- $FT_1 = \{F1, F2, F3, F4\}$
- $Fd_2 = \{F3\}$
- $FT_2 = \{F3, F5\}$

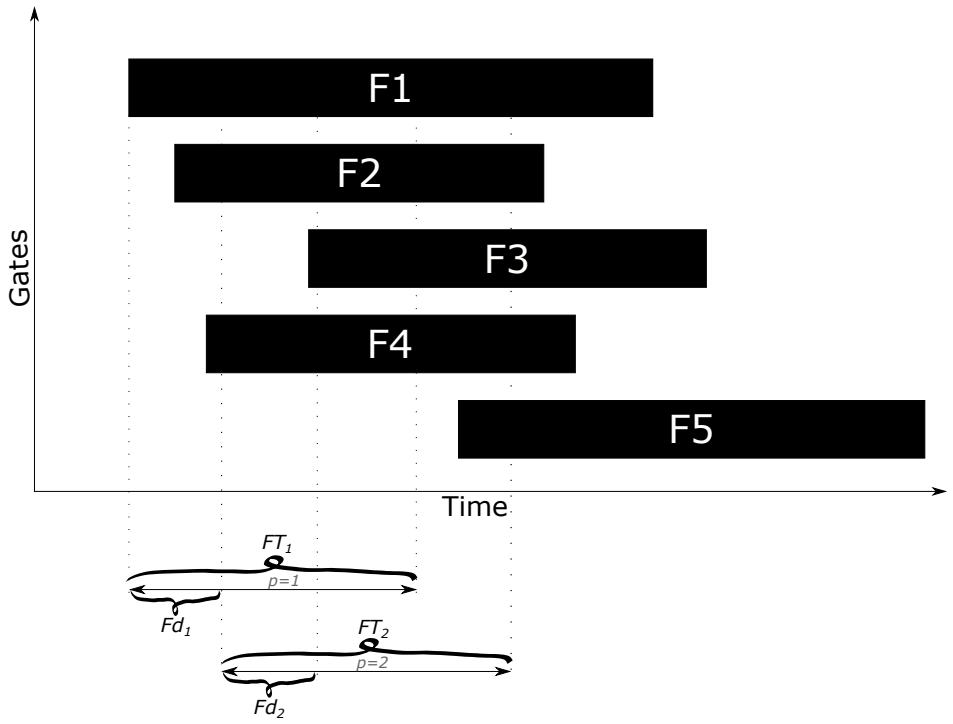


FIGURE 6.1: Window positioning and shifting.

When the dynamic implementations are used the subsets Fd_1, Fd_2 are as they were, but FT_1, FT_2, FT_3 change. They contain the following flights:

- $FT_1 = \{F1, F2, F3, F4, F5\}$
- $FT_2 = \{F3, F5\}$

6.3 Experimental Settings

The experimental settings which were used for the MIP model and for the RH method are discussed in this section.

6.3.1 Receding Horizon Method Settings

Twenty parameter configurations were tested for each of the decomposition methods which are described in Section 6.2. The configurations were chosen relative to the instance size, so that the parameters are not too large and not too small in comparison to the whole instance.

The tested sizes of the window are:

- for the static time-oriented decomposition: 10%, 20%, 30%, 40% and 50% of instance duration,
- for the static flight-oriented decomposition: 10%, 20%, 30%, 40% and 50% of the total number of flights in an instance
- there is no need to set the size of the window for the dynamic versions

The sizes of the shifts are:

- for the static time-oriented decomposition: 20%, 30%, 40% and 50% of each window duration (i.e. the biggest tested shift size is 25% of the whole instance duration)
- for the static flight-oriented decomposition: 20%, 30%, 40% and 50% of each of the window sizes (the biggest tested shift size is 25% of all flights in an instance)
- for dynamic-time oriented decomposition: starting from 6% and ending at 25% of the instance duration using 1% steps (the biggest shift size is 25% of the whole instance duration)
- for dynamic flight-oriented decomposition: starting from 6% and ending at 25% of all flights in an instance using 1% steps (the biggest shift size is 25% of all flights in an instance)

6.3.2 MIP Model Settings

The MIP model settings which were used in the experiments are the same as in Chapter 5, Section 5.4, Table 5.2, i.e. LG equals 60 min, SG equals 15 min, d equals 60, shortest time margin for conflicts is 3 min, longest time margin for conflicts is 10 min.

The results of the experiments obtained using the above methods with the given settings, and datasets described in Chapter 3 are discussed in the next section. A desktop PC (3.06 GHz Intel i7-950, 24GB RAM, running Linux and CPLEX version 12.6) was used to obtain all of the results.

6.4 Results

Several aspects of the RH method were tested in these experiments, for which results are presented in here. Section 6.4.1 focuses on identifying the instances of the problem which can benefit from application of the RH method. How much time was taken to obtain the optimal solution for each instance is analysed and, based upon these times, it is decided for each instance whether to apply the RH method. The results of the four RH method implementations obtained for various parameter settings are presented in Section 6.4.2. The influence of the window size and the shift size on the results is observed. The best implementation and the best parameter settings are proposed in Section 6.4.2. Finally in Section 6.4.3 the best RH decomposition method is applied to solve the combined problem of two terminals. The need to solve more than one terminal at once was investigated in Section 5.5.4, Chapter 5 where the number of conflicts around gates was analysed.

6.4.1 Dataset Size vs. RH Method Application

In order to validate the results of the RH method, an optimal solution is needed for each instance. The calculation times taken to get the optimal solution for each of the instances were already presented in Table 5.3, Section 5.5.2, Chapter 5. It can be observed from the table that instances coming from terminal T2 are very quick to solve. The reason why the RH method is not needed in such cases is explained further in this section. Table 6.1 repeats the calculation times for instances from terminals T1 and T3. A time limit of 10 hours was used in the calculations: 5 instances for Terminal 1 and 5 instances for Terminal 3 were solved to optimality within the time limit.

The RH method can decrease the calculation time but is expected to reduce the quality of the solution at the same time. It is therefore worth applying the method only when the calculation time of the optimal solution is relatively long. Instances that could not be solved to optimality within less than 10 hours are most probably going to benefit from using the RH method. It is however not clear which of the shorter instances will benefit from it. It is going to be investigated next.

Four implementations of the RH method are applied for each instance: two static and two dynamic. Twenty configurations, discussed in Section 6.3.1, are calculated for each of the four implementations. The calculation times taken are recorded and a relative time is then calculated for each test case. The relative time is the RH calculation time given as

Day(Term)	Number of flights	Calculation time[s]
1(1)	111	153
2(1)	118	171
3(1)	121	36000
4(1)	125	239
5(1)	133	36000
6(1)	116	198
7(1)	106	5902
1(3)	91	29
2(3)	117	443
3(3)	119	1669
4(3)	123	36000
5(3)	121	36000
6(3)	92	18
7(3)	80	226

TABLE 6.1: Results obtained when the full formulation is used.

a fraction of the time CPLEX took to solve the full formulation.

Table 6.2 shows the relative times. "MinTime" gives the minimum relative time across the various parameter settings for each implementation. "MaxTime" presents the maximum relative times reached. "BestObjTime" shows the times taken to calculate the best objective value for that configuration.

It can be observed from Table 6.2 that the times taken by the static implementations tend to be longer than those taken by the dynamic implementations. Moreover the difference between the maximum and the minimum time is usually larger for the dynamic implementations. BestObjTime is usually reached for times which are shorter than MaxTime.

It can be observed that the following datasets don't benefit from applying the RH decomposition method: 1(1), 2(1), 4(1), 6(1), 1(3), 6(3). Even the minimum relative time ("MinTime") is long for these. A significant calculation time improvement can, however, be observed for datasets: 3(1), 5(1), 7(1), 2(3), 3(3), 4(3), 5(3), 7(3). These datasets are used in Section 6.4.2 to investigate the impact of the RH parameters. Comparison with Table 6.1 indicates that in most cases (other than 7(3)), these are harder datasets to solve to optimality.

	Min	Max	BestObj	Min	Max	BestObj
Day(Term)	Static Time Implementation			Static Flight Implementation		
1(1)	1.2844	6.0854	1.59	1.1776	9.7271	1.1776
2(1)	1.5746	10.1666	1.5746	1.4292	12.9427	1.4292
3(1)	0.0127	0.0663	0.0612	0.0117	0.0756	0.0203
4(1)	1.5394	9.7864	2.6419	1.2414	8.3727	1.7295
5(1)	0.0228	0.0878	0.0648	0.0139	0.0829	0.0139
6(1)	1.7296	12.6344	1.7296	1.4349	13.0826	2.529
7(1)	0.0671	0.5542	0.1145	0.0376	0.2504	0.1047
1(3)	1.209	4.1907	1.6643	0.7811	4.5304	1.6097
2(3)	0.1382	0.7079	0.1665	0.1496	0.9694	0.1593
3(3)	0.0457	0.3473	0.0575	0.0489	0.2636	0.0577
4(3)	0.0041	0.6139	0.012	0.0033	0.098	0.098
5(3)	0.0027	0.6057	0.0176	0.003	0.6081	0.0094
6(3)	1.3444	7.5213	2.223	1.34	9.7326	1.4993
7(3)	0.1311	45.4432	0.5219	0.1326	0.7116	0.7116
	Dynamic Time Implementation			Dynamic Flight Implementation		
1(1)	0.8717	1.6154	1.3495	0.9407	1.6397	0.9407
2(1)	1.4079	2.1169	1.5118	1.307	2.1357	1.5362
3(1)	0.0205	0.0357	0.0205	0.0204	0.0528	0.021
4(1)	1.3702	2.0972	1.3918	1.1042	1.9505	1.1042
5(1)	0.0197	0.1571	0.0212	0.0204	0.0569	0.0463
6(1)	1.8605	4.8266	2.2491	2.1213	4.9865	2.1213
7(1)	0.0509	0.1291	0.0737	0.0529	0.1385	0.0989
1(3)	0.6172	0.9952	0.8797	0.6359	1.0458	0.6679
2(3)	0.1059	0.1985	0.1301	0.1258	0.243	0.1617
3(3)	0.046	0.0739	0.061	0.0446	0.0823	0.0512
4(3)	0.0392	0.0495	0.042	0.0383	0.0536	0.0383
5(3)	0.0021	0.0036	0.003	0.0019	0.0035	0.002
6(3)	1.0107	1.476	1.052	0.9881	1.5453	1.1726
7(3)	0.1485	0.4578	0.1733	0.1537	0.4584	0.2294

TABLE 6.2: Maximum, minimum and best objective relative times obtained in the experiments.

6.4.2 Parameters Impact

The influence of the parameters on calculation time and objective value is investigated in this section. The following datasets were used in the described experiments: 3(1), 5(1), 7(1), 2(3), 3(3), 4(3), 5(3), 7(3). The configurations, which were discussed in Section 6.3.1, are numbered, as in Section 4.4, Chapter 4. Table 6.3 is a look up table of configuration numbers with corresponding window and shift sizes. The configuration numbers rather

Configuration Number	Static Implementations		Dynamic Implementations
	Window Size [%]	Shift Size [%]	Shift Size [%]
1	10	20	7
2	10	30	8
3	10	40	9
4	10	50	10
5	20	20	11
6	20	30	12
7	20	40	13
8	20	50	14
9	30	20	15
10	30	30	16
11	30	40	17
12	30	50	18
13	40	20	19
14	40	30	20
15	40	40	21
16	40	50	22
17	50	20	23
18	50	30	24
19	50	40	25
20	50	50	26

TABLE 6.3: The configuration numbers and parameter settings for the RH method.

than the actual parameters are used in the text and the figures, since this simplifies the description. The results which were obtained for various configurations of the RH method are presented using grey scale maps.

Two separate maps are created for each of the four implementations, one shows the calculation times and the other shows the objective values. Each map consists of squares, which correspond to each of the tested configurations for the method. Twenty evaluated configurations using the eight datasets, results in one hundred and sixty squares for each map. Dataset names are shown in the vertical direction while configurations are in the horizontal direction.

All grey scale maps are present in Figure 6.2 (static implementations) and Figure 6.3 (dynamic implementations). Maps which are presented in Figure 6.2 a,c and Figure 6.3 a,c show the relative calculation times. The times for each dataset are given as a percentage of the maximum time taken by this dataset. The black colour indicates that the

calculation time was long for a given dataset. The white colour means that the time was short.

The maps which are visible in Figure 6.2 b,d and Figure 6.3 b,d show the percentage gaps. The percentage gaps are calculated using Equation 6.9, where b is the RH objective value and a is the optimal objective value. For the instances which were not solved to optimality the feasible objective values (obtained after ten hours) are used for a. In order to visualise the small differences between small values of the good objective values the results were saturated at 0.5% . The results which are worse than 0.5% are displayed as 0.5%.

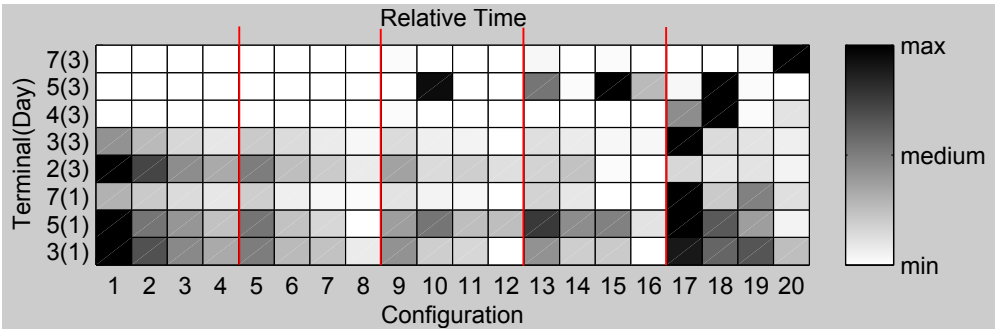
$$\%\ gap = 100\% * \frac{b-a}{a} \tag{6.9}$$

It can be observed from the maps presented in Figure 6.2 a,c that the calculation time is strongly dependent upon the shift size when the static implementations of the RH method are applied. It can be seen that the larger shift sizes take a shorter time than the smaller shift sizes if the window size is fixed. When the shift size is larger and the window size is fixed, fewer window positions have to be solved. It can also be seen from the maps in Figure 6.2 b,d that results obtained for various shifts and the same window size tend to be similar. The shift size does not appear to greatly influence the objective value. Blocks of four similar values obtained for the same window size and four different shift sizes can be observed in Figure 6.2 b,d. The red vertical lines mark the borders of the blocks.

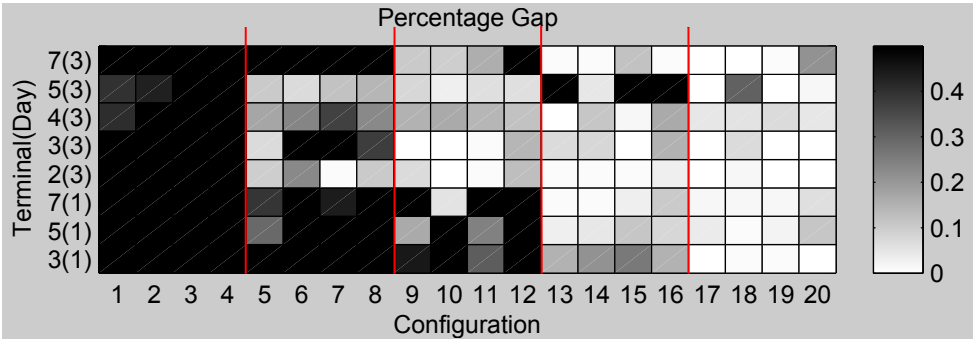
The maps obtained for the dynamic implementations are visible in Figure 6.3. The influence of the shift size on the calculation times can also be observed for the dynamic maps. The maps presented in Figure 6.3 a,c tend to be darker on the left side where the smaller shifts are situated. The tendency is however less visible than it is for the static implementations. No obvious pattern can be observed from the dynamic maps showing percentage gaps (Figure 6.3 b,d). This is to be expected if the relationship is with window size, which is dynamic.

When all of the relative time maps (Figure 6.2 a,c and Figure 6.3 a,c) are considered, it seems that the dynamic method is avoiding the exceptionally short times and the exceptionally long times. This results in times in the middle of those taken by the static implementations.

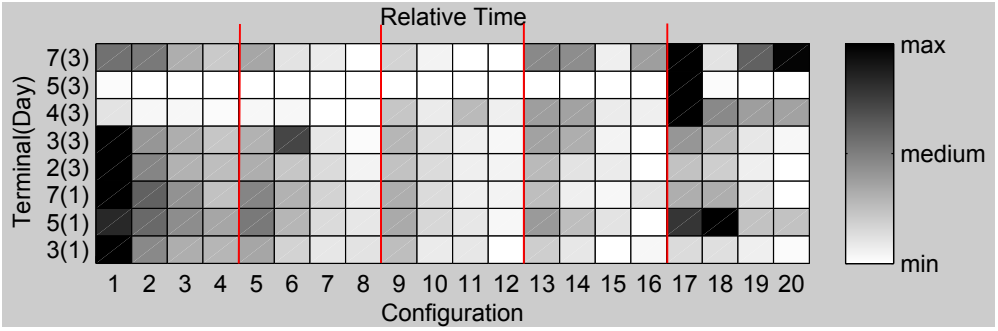
A similar observation can be made for the objective values (Figure 6.2 b,d and



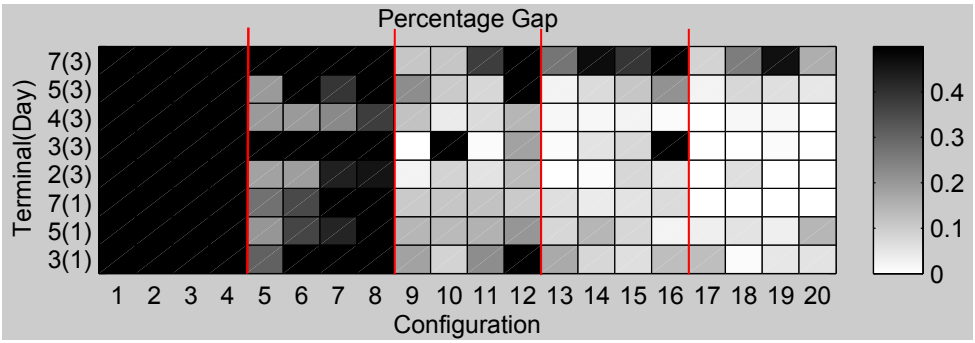
(a) Static time-oriented decomposition, relative time



(b) Static time-oriented decomposition, percentage gap

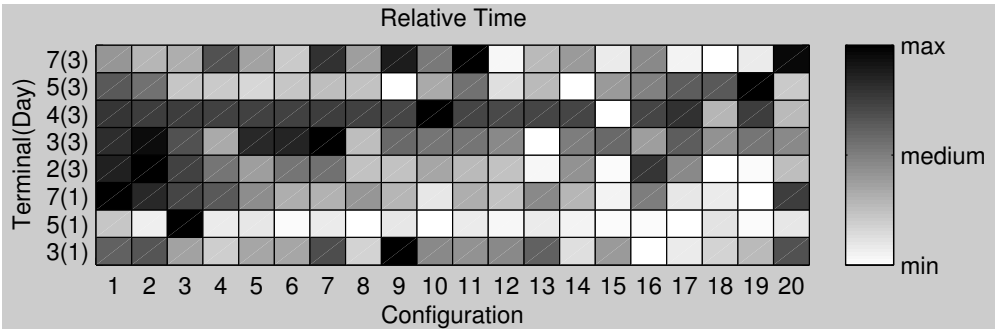


(c) Static flight-oriented decomposition, relative time

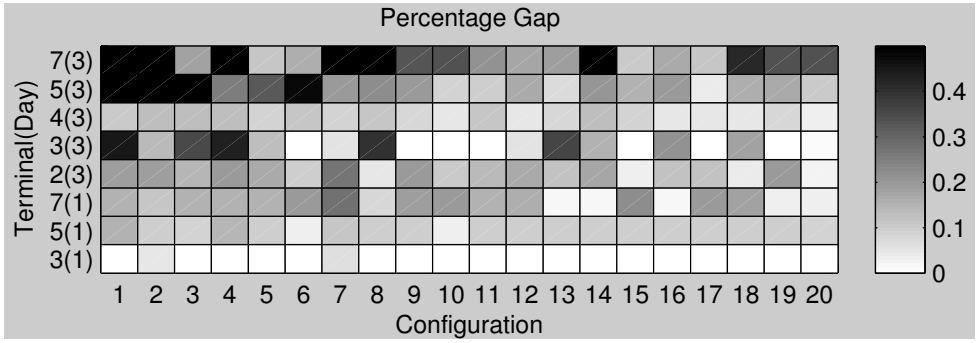


(d) Static flight-oriented decomposition, percentage gap

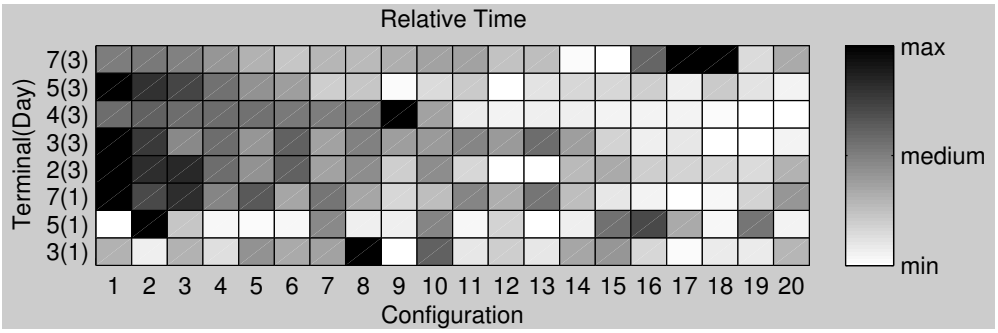
FIGURE 6.2: Grey scale maps which show relative times and percentage optimality gaps obtained using various configurations of the static RH method.



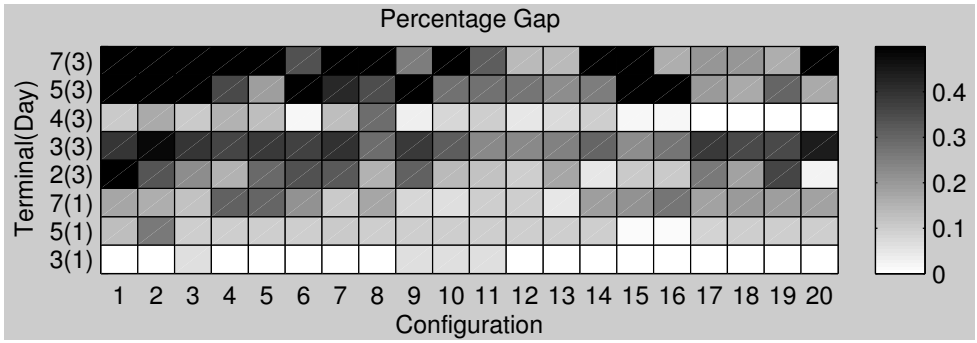
(a) Dynamic time-oriented decomposition, relative time



(b) Dynamic time-oriented decomposition, percentage gap



(c) Dynamic flight-oriented decomposition, relative time



(d) Dynamic flight-oriented decomposition, percentage gap

FIGURE 6.3: Grey scale maps which show relative times and percentage optimality gaps obtained using various configurations of the dynamic RH method.

Configuration Number	Implementation Name	Relative Time		Percentage Gap	
		Mean	Var	Mean	Var
17	static time	0.2628	0.0503	0.0141	0.0004
19	static time	0.1169	0.0871	0.0191	0.0007
17	static flight	0.2628	0.0705	0.0351	0.0024
20	static flight	0.1303	0.0555	0.0527	0.0045
20	static time	0.1303	257.3147	0.0583	0.0054
18	static flight	0.1104	0.0097	0.0618	0.0070
18	static time	0.1104	0.0617	0.0630	0.0109
14	static time	0.1130	0.0546	0.0695	0.0049
17	dynamic time	0.0660	0.0036	0.0787	0.0046
20	dynamic time	0.1068	0.0206	0.0804	0.0125

TABLE 6.4: Ten best configurations in terms of mean percentage gap to optimality.

Figure 6.3 b,d). The variation of the results obtained for the dynamic implementations is much lower than for the static ones. The dynamic results are dominated by the medium quality results, while the very good and the very bad results are reached for the static implementations.

When the static implementations are considered, no significant difference can be observed between the results obtained for the time-oriented implementation and for the flight-oriented implementation. In contrast, when the dynamic implementations are considered it is visible that the time-oriented implementation tends to be slower but achieves better objective values than the flight-oriented implementation.

Best RH Configuration

In order to find the best implementations and their best configurations mean and variance of the relative time and percentage gap were calculated across the eight datasets for each configuration. The ten best configurations in terms of mean percentage gap are shown in Table 6.4 while Table 6.5 shows the ten best configurations in terms of mean relative time.

The configuration number is given in the first column of Table 6.4 and Table 6.5. The type of implementation is given in the second column. Columns three and four present the mean and the variance of the relative time while the mean and variance of the percentage gap are given in the last two columns.

Generally the best trade-off between the objective value and the calculation time should be found, but the concept of the best does not mean the same for every user and

Configuration Number	Implementation Name	Relative Time		Percentage Gap	
		Mean	Var	Mean	Var
18	dynamic time	0.0592	0.0022	0.1433	0.0169
12	static flight	0.0608	0.0051	1.1195	4.7250
12	static time	0.0608	0.0052	2.0195	20.9924
8	static flight	0.0613	0.0049	2.3242	20.8437
8	static time	0.0613	0.0051	2.0401	20.6142
15	dynamic time	0.0615	0.0030	0.0887	0.0059
19	dynamic time	0.0620	0.0032	0.1165	0.0137
12	dynamic time	0.0647	0.0029	0.1105	0.0045
11	static flight	0.0656	0.0047	0.1373	0.0137
11	static time	0.0656	0.0230	0.2047	0.0486

TABLE 6.5: Ten fastest configurations.

can depend upon the requirements. Several cases are therefore discussed below.

If the key requirement is a good objective value then the static time-oriented implementation run with configuration number 17 (Table 6.4) should be used. However, when not only the objective value but also the calculation time is considered and a slightly worse objective value is still acceptable then the dynamic time-oriented implementation with configuration 17 is better. This reaches an objective value close to the best objective obtained (ninth place in Table 6.4) but takes considerably less time. The average relative time taken by this configuration is very close to the best average relative time (Table 6.5) and is much smaller than the other times shown in Table 6.4.

In order to understand the importance of the differences between the solutions from the RH approach and those from the full formulation, the solutions for one configuration, number 17 (dynamic time-oriented implementation), were closely examined. When the elements of the objective function were considered, the only differences were found to be in the time gaps and airline preference elements. The time gap element was on average 0.3% larger while the airline preferences were on average 0.4% larger for the receding horizon solution than for the full solution. The most significant differences between the two solutions were observed for dataset 5(1). The time gap cost element for dataset 5(1) was 2.5% larger and the airline preferences were 0.43% worse. Pairwise comparison of the gap sizes showed that the first five smallest gaps were exactly the same, they were equal to 16, 20, 24, 25 and 28 minutes. In total ten flights had a 60 minute or smaller gap in both cases, with three of these gaps differing: 42, 46 and 54 minutes in the full case, compared with 33, 42 and 60

minutes in the RH case. The RH approach avoided unnecessarily small gaps and provided a solution which was fairly comparable to that of the full approach. In the full solution 38 flights were allocated to the gates which were most preferred by their airlines. In the RH solution 36 flights used the most preferred gates. Further investigation showed that the deviations were very small and we would expect them to be easily accepted by airlines.

6.4.3 Solving Two Terminals at Once

It was suggested in Chapter 5 that solving more terminals at once may have a positive impact on the number of resolved conflicts at the gates. This is investigated in more depth in this section.

Seven datasets from terminal T1 and seven corresponding datasets from terminal T2 were merged to create the large combined instances which are used in this section. Gates which belong to different terminals but may cause conflicts (for example which are placed opposite each other) are assigned to one group of gates, i.e. some of gate groups contain flights from multiple terminals. In this way more conflicts will be considered in the group conflict constraint. Moreover it is not permitted for any flight to use a different terminal than scheduled, i.e. flights belonging to terminal T1 have to be allocated to gates at terminal T1.

The combined instances are solved using the RH method. The best trade-off configuration of the RH method which was suggested in Section 6.4.2 (dynamic time-oriented implementation with shift size equal to 22% of the instance duration) is used. Table 6.6 shows the conflicts which were detected for the seven datasets. The conflicts identified in the solutions of the large, combined instances are shown in column "Windowed". The other two columns refer to the solutions obtained for the two terminals solved separately. The separate solutions are put together and the number of conflicts is calculated (like in Section 5.5.4, Chapter 5). The column "Full combined" shows the conflicts when two solutions of full formulations are put together. The column "Windowed combined" shows the conflicts when two decomposed solutions are merged. The same RH decomposition (dynamic time-oriented implementation with shift size equal to 22% of the instance duration) is used to achieve the decomposed solutions.

It can be observed from Table 6.6 that, as expected, the number of conflicts obtained when the two terminals are solved separately is, for most cases, slightly larger than

Number of flights	Day(Terminals)	Windowed	Windowed combined	Full combined
160	1(1 & 2)	0	1	2
165	2(1 & 2)	17	9	9
166	3(1 & 2)	15	17	16
174	4(1 & 2)	18	20	19
189	5(1 & 2)	16	17	20
168	6(1 & 2)	19	28	24
145	7(1 & 2)	16	9	1

TABLE 6.6: Number of conflicts detected for various solutions of terminals T1 and T2.

when the combined instances are solved at once.

However, an unexpected growth in the number of conflicts, which is quite significant, can be observed for instances 2(1 & 2) and 7(1 & 2). Further analyses lead to the conclusion that the growth is related with the way the group conflict constraint is modelled. Namely, it minimises the maximum value of conflicts per group of gates, which means that if for one group a larger number of conflicts has to be kept in the solutions, all of the other groups can also have a bigger number of conflicts without any additional penalty in the objective function. Let say there are for example two groups: $G1$ and $G2$. $G1$ requires allowing three conflicting flights and $G2$ has two conflicting flights. The objective to minimise conflicts will then aim to reduce the number of conflicting flights to three in each group. It will not attempt to reduce the number of conflicting flights in $G2$ because the group already has less than three conflicting flights. When the terminals are solved together not only groups are bigger but also there are more groups considered than when the single terminal instances are solved. The current formulation of the conflict constraint works in general and it reduces the conflicting situations overall but it can be observed that in some situations it would be useful to also add one constraint per each group which will limit the number of conflicts for each group individually. Adding the additional constraints and testing their influence on the results would be an interesting future work.

6.5 Conclusion

This chapter considers the GAP modelled using mixed integer programming and the decomposition method called the RH approach. It has been observed that the time taken by CPLEX to solve the GAP model depends strongly upon the instance of the problem which is solved. Some instances are easy to solve to optimality and there is no need to decompose

them. However for some instances only a feasible solution can be found within even a long time limit. It is shown that RH method is a useful tool which allows very good solutions to be found for the larger instances of the GAP within times which are much shorter than CPLEX needs.

Four implementations of the RH method have been described and compared in the chapter: two static and two dynamic implementations. The static implementations are much more sensitive to the parameters than the dynamic implementations. Analysis of the static implementations indicates that the objective value is mostly related to the window size while the calculation time is influenced by the shift size. The smaller window sizes are quicker but they result in worse objective values. Larger shifts take less time to calculate than the smaller shifts for the same window size. It is harder to observe the relations for the dynamic implementations as the size of the window varies dynamically at each window position.

The most promising settings were suggested based upon the parameter study for the tested instances. When the best objective is the only criterion the static-time oriented implementation with the window size equal to 50% of the dataset duration and the shift size equal to 20% of the window size give the best results on average. However when a good trade-off between the calculation time and the objective value is sought then the dynamic time-oriented implementation with the shift size equal to 22% of the dataset duration is suggested.

Finally, the results obtained when two terminals are solved at once using the RH method are presented. Further reduction of the number of conflicts is observed for five out of seven tested incarnates. For future work it was suggested that the model could benefit from adding additional constraints which would limit the number of conflicts for each group separately. Conflicts occur not only around the gates but also on the taxiways further away from the terminals. Chapter 7 focuses on incorporating much information from the routing problem in the GAP model in order to resolve the taxiway conflicts.

Chapter 7

An Integration Framework for the GAP and the Ground Movement Problem

7.1 Introduction

This chapter focuses on a closer integration of the gate allocation problem (GAP) with the ground movement problem. Chapter 5 discussed how conflicts at the gates can be resolved. This chapter focuses on conflicts occurring on taxiways.

At many airports the gate allocation problem and the ground movement problem are solved using two separate systems. One of the systems solves the allocation problem, often before the day starts, without any ground movement information. The other solves the ground movement problem based upon the allocation provided by the gate allocation system; it tries to find the shortest possible non-conflicting route for each aircraft. The ground movement system considers and deals with the congestion and delays which could occur during the day of operation. The taxiways which aircraft take are directly dependent upon the gates they are assigned to, hence there is a close link between the assigned gates and congestion on the taxiways.

The key aim of this research is to establish if it is possible to intelligently change the gate allocation plan by considering the feedback information from the routing by simulating and solving in advance, so that fewer conflicts occur on the routes during the day of operation.

The design and evaluation of a framework within which the two optimisation systems: the ground movement system and the gate allocation system can cooperate is the main contribution of this chapter. The two problems are still solved separately, the

information flow is however allowed by the framework. The chapter provides preliminary results which indicate a positive impact of incorporating the additional ground movement information in the GAP model at the early stage of planning.

The full formulation of the GAP is used in the framework, and the ways in which the decomposition from earlier chapters could be integrated is also discussed so that larger instances of the GAP can be solved.

The ground movement problem is described using Manchester Airport as an example in Section 7.2. Next the ground movement algorithm which is used in the experiments is briefly described in Section 7.3. Section 7.4 explains how the two optimisation models: the GAP model and the routing algorithm communicate with each other using the feedback loop framework. The way in which the feedback loop is organised is explained along with the stopping criterion of the loop. The preliminary experimental results are then described in Section 7.5. An alternative version of the feedback loop framework which uses the RH decomposition is described in Section 7.6. The chapter summary is provided in Section 7.7.

7.2 Ground Movement Problem

The ground movement problem is a routing problem combined with a scheduling problem. It considers routing aircraft on taxiways of an airport so that they get to their destinations on time. The aim is to minimise the travel time (or alternatively the distance) and the taxiway conflicts. Two aircraft should never be within a given distance of each other on an airport surface at the same time.

The presented algorithm requires a graph representation of the layout of airport taxiways. The graph nodes are added in places where a path splits or crosses with other paths. Additionally nodes are added for holding positions as aircraft can be held there. Nodes are also added between the existing nodes which are placed far away from each other. The number of nodes which are added between the exiting nodes can be found by checking how many aircraft can be queued between the nodes or safely occupy the path [4]. In our experiment it has been assumed that every distance between existing nodes which is bigger than 150 meters requires additional nodes. Very long distances may require several of them. However a minimum distance between nodes should also be maintained as it is assumed that any two neighbouring nodes can be occupied by aircraft at the same time, if necessary. It has been reported in [49, 40, 74] that this distance should be at least 60

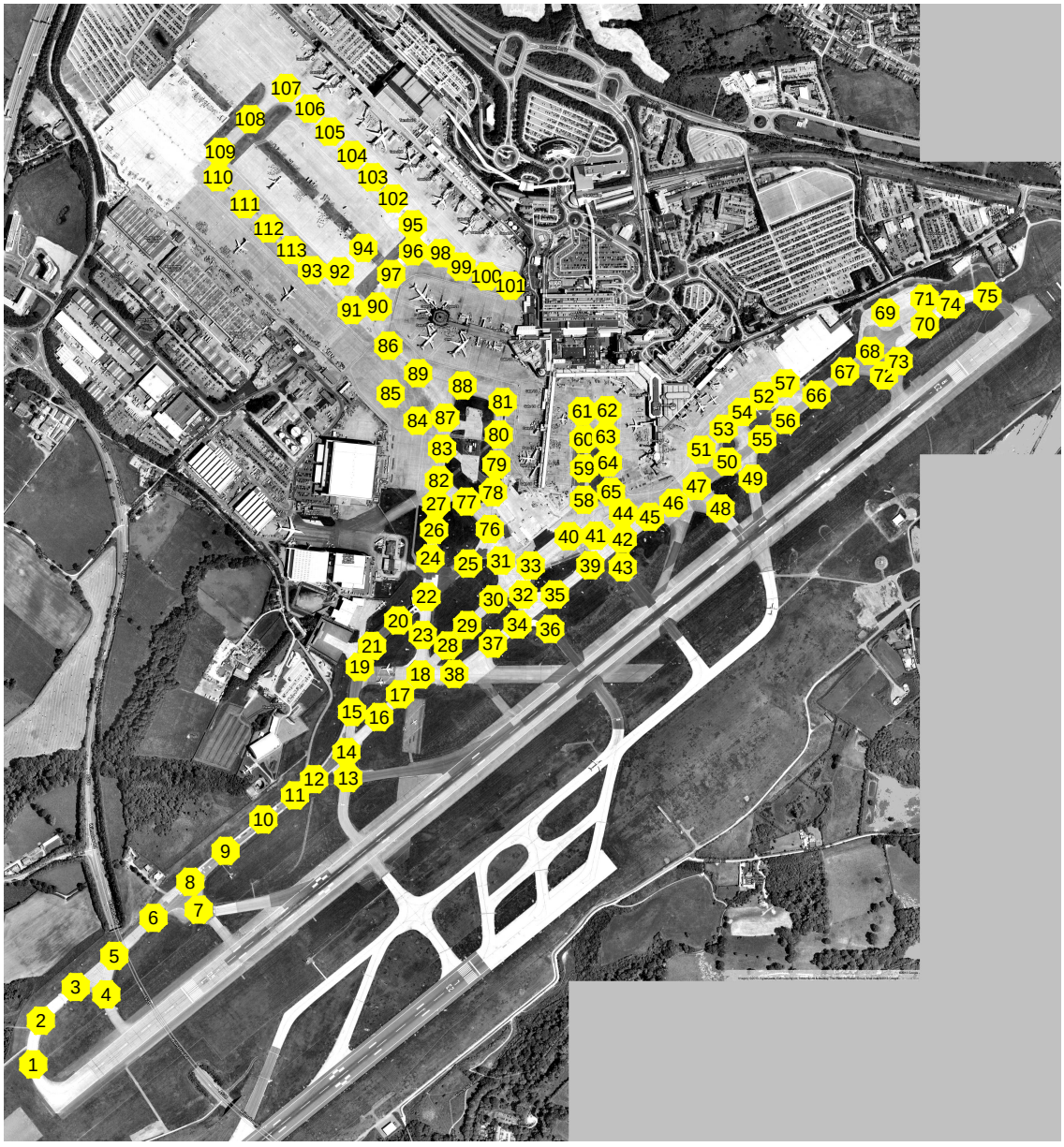


FIGURE 7.1: Manchester Airport map with the nodes used by the routing algorithms (Google satellite map, Imagery ©2013 TerraMetrics, Map data ©2013 Google.)

meters. The graph edges connect the nodes and symbolise the paths which aircraft can follow when transferring between the nodes.

Manchester Airport is used as a real world example in the experiments provided in this chapter. Its layout and nodes of the graph are presented in Figure 7.1.

An algorithm has been developed for these experiments to simulate the ground movement. It is a greedy algorithm which assigns full paths to the flights. It requires all possible paths leading from runways to each gate and from each gate to runways to be known before the algorithm starts to run. It is effective only if it is easy to define the optional paths from each gate to the runway manually. This was time consuming for Manchester Airport and would be more so for larger airports.

The departure and arrival paths depend upon the runway orientation which is in use and the orientation is chosen by the airport according to the weather, i.e. the wind direction. The west winds are more common for Manchester Airport, the west orientation of the runway is therefore usually used (aircraft take off facing south-west and land facing south-west). In these experiments the west orientation is therefore assumed. Two runways are assumed to be in use. The one which is closer to the terminals is used by arriving aircraft and the one further from the terminals by departing aircraft.

A web-page ([31]) which provides air traffic information, both live and historical, was used to identify the optional paths. Records from several days (for west orientation of two runways) were analysed.

There are usually several paths which lead from/to each gate. In total 560 paths were identified. The length of each path is calculated between the gate and a departure node (if it is a departure path) or an arrival node (if it is an arrival path). The arrival nodes are the nodes though which arriving flights can enter the taxiways when coming from the runway after landing. The departure nodes are those through which the departing aircraft can access the area where they queue before departing. The following departures nodes: 36, 43, 48, 49 and arrival nodes: 1, 7, 13 (see Figure 7.1) were used in the paths. It can be seen in Figure 7.2 that the lengths of paths vary a lot. Some of the gates are placed much closer to the runways than others. The data shows that closer gates tend to be more popular among the airlines as they are easier to access, with lower taxi times and hence fuel burns on the ground. Figure 7.2 shows the lengths of all of the optional paths, which were created based upon the analysis ordered by increasing length. The vertical direction indicates the length in kilometres, the horizontal the path number.

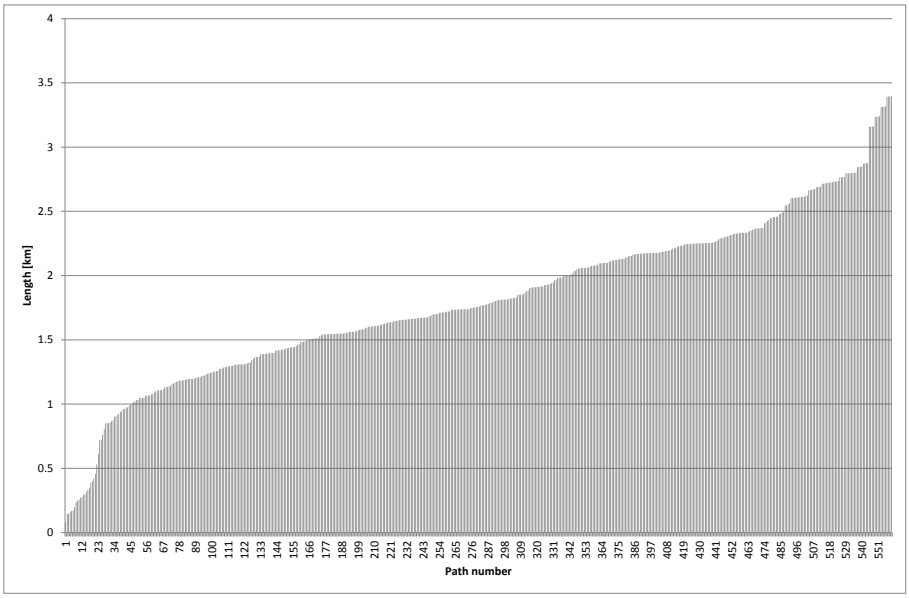


FIGURE 7.2: Lengths of all paths identified at Manchester Airport, which are used by the greedy routing algorithm.

7.3 The Greedy Ground Movement Routing Algorithm

This section explains how the greedy routing algorithm works. Firstly the notation is given and then the steps of the algorithm are explained.

7.3.1 Notation and Definitions

r	a route: a path from a gate to a runway or vice versa
a	an allocation of a flight to a gate
SA	an allocation plan, containing details of all allocations
OR_a	a set of optional arrival/departure routes for allocation a
TO	the total number of routing conflicts
mar	the time margin used to detect conflicts, i.e. flights which move within this time period of each other will be assumed to be in conflict

The greedy routing algorithm assigns the whole routes to aircraft departing and arriving at an airport. For each gate there exists a set of possible routes (arrival and

departure paths). The routes are created based upon an airport map and the observations of live traffic. Each route r consists of a set of nodes, the distances between the nodes are known. The route depends not only upon the arrival/departure gate but also upon the size of the aircraft to which it is assigned. Large aircraft, for example, need to use a longer spread of the runway and need a bit different paths than small aircraft.

7.3.2 Algorithm

The steps of the greedy routing algorithm are shown in Algorithm 2.

Algorithm 2 Steps of the routing algorithm

Require: Allocation plan SA

Require: Set of optional arrival/departure routes OR_a for each allocation a

```
Sort SA according to flight size (small to large)
for each allocation a in SA do
    Assign shortest arrival and departure route
end for
Calculate TO
for each allocation a from the SA do
    Save current arrival/departure route as cr_a
    for each optional arrival/departure route r_a in OR_a do
        Assign an optional arrival/departure route r_a
        Calculate TO
        if TO decreases then
            Replace cr_a with r_a
        else
            Keep cr_a as it was
        end if
    end for
    Assign cr_a
end for
```

The input of the routing algorithm is the allocation plan SA (flight-gate pairs) and a set of optional arrival/departure routes OR_a for each flight-gate pair a (for each allocation).

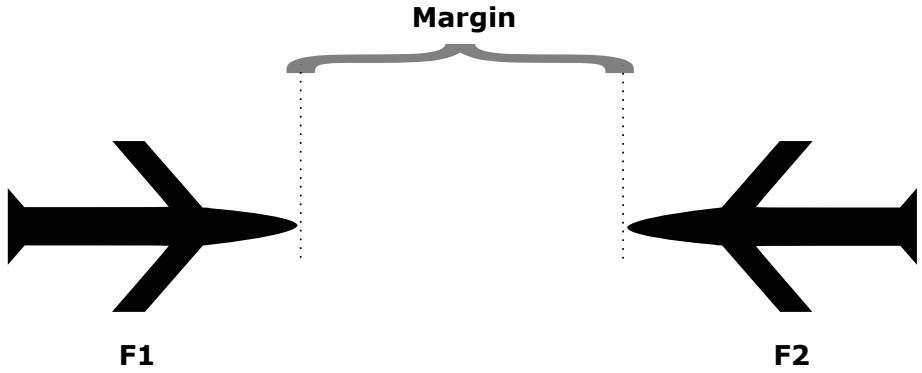


FIGURE 7.3: Margin (distance) between aircraft ($F1$,$F2$) going into opposite directions

First (step 1 in Algorithm 2) the allocations are sorted according to the size of aircraft (small to large). Then the algorithm allocates the shortest arrival and departure path for each allocation in the supplied plan SA (steps 2-4 in Algorithm 2). The total number of conflicts (TO) is calculated (step 5 in Algorithm 2). The exact definition of a conflict is given further in this section.

Next the algorithm tries to find an optional, longer arrival/departure path for each of the sorted allocations (steps 6-16 in Algorithm 2). If there is a longer arrival/departure path which reduces the total number of conflicts, it is used instead of the shortest one (step 12 in Algorithm 2). The algorithm terminates when all of the possible allocations have been checked or all conflicts have been resolved. All of the remaining conflicts will have to be resolved during the day of operation by delaying some of the aircraft on holding positions or preferably on stands. The reason for checking the allocations of the smallest aircraft first is an assumption that it is better from the environmental point of view when the smaller rather than the larger aircraft make a detour.

A conflict is defined for each pair of aircraft which has common nodes of assigned paths in which they are most likely going to meet. If the two aircraft will pass (going in opposite directions) at least one of the common nodes within a particular time margin mar then they are in conflict. All speeds of the aircraft are assumed to be constant. Only pairs of aircraft which pass a node going in opposite directions are considered in the conflict

checks. The conflicts which occur between flights which head in the same direction are not in scope of the investigation as they are easier to resolve, with one flight following the other. Figure 7.3 shows the time margin converted into distance using the constant speed and a pair of aircraft which will be in conflict if the time margin is smaller than the assumed time margin.

7.4 Combining the Ground Movement Problem and the GAP

This section introduces the integration framework. First the notation is introduced, then the overview of the framework is given. Next the feedback information from routing and the stopping criterion are discussed in more depth. Finally an additional seeding procedure is presented which was introduced after initial tests with the framework. The seeding procedure feeds an initial solution in the Gate Allocation Optimiser which reduces the number of feedback loop iterations.

7.4.1 Notation and Definitions

The notation introduced in Section 7.3.1 is still valid, it is here extended by the following elements which are used in the description of the framework.

CA	a subset of allocations which take part in conflict
$affected_a$	a subset of allocations affected by a
$feedList$	a feedback list, each of its elements includes configuration c_i and number of conflicts $conf_i$
l	the total number of elements in $feedList$
i	an index of $feedList$ elements $(i) \in \{1, ..., l\}$
c_i	a combination of allocations
$conf_i$	the number of conflict detected for c_i, an element of $feedList$
nc_i	the total number of allocations in c_i
j	an index of allocation $(j) \in \{1, ..., nc\}$
g_j	the gate of allocation with index j
f_j	the flight of allocation with index j
w_i	the penalty for having configuration c_i in the allocation
X_{g_k,f_i}	a decision variable, it becomes 1 if f_i is allocated to g_k, 0 otherwise
$rvar_i$	an indicator variable, becomes 1 if configuration c_i of $feedList$ occurs in the allocation plan

7.4.2 Feedback Loop Framework

The framework contains three main elements: the Routing Optimiser which uses the routing algorithm described in Section 7.3, the Gate Allocation Optimiser which uses the model described in Chapter 5 and the Manager which co-ordinates the information flow between the two. Figure 7.4 shows a block diagram with relations between the elements of the framework. Firstly the Gate Allocation Optimiser is run in order to find the initial allocation,

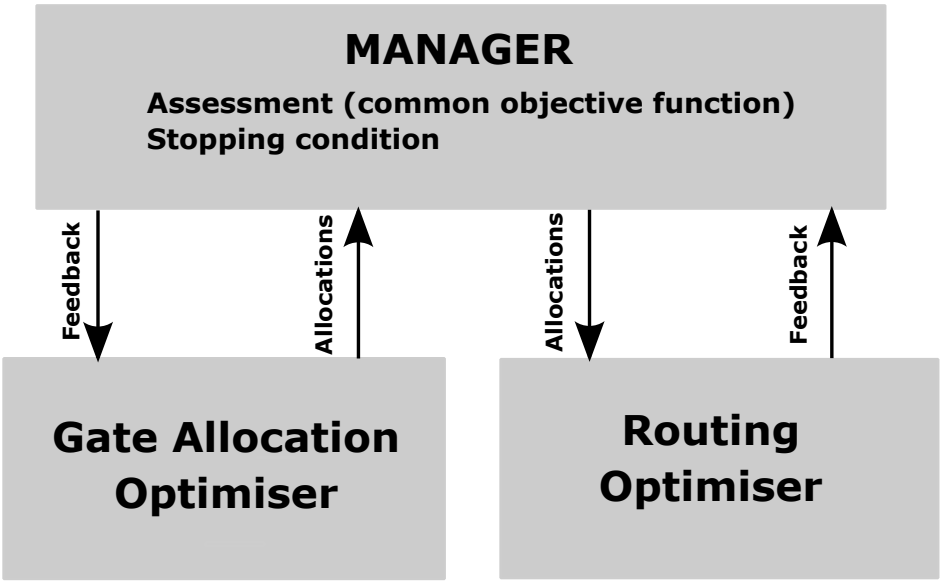


FIGURE 7.4: The framework elements.

of aircraft to gates. The initial allocation is then passed by the Manager to the Routing Optimiser which assigns routes to each allocated flight. The Routing Optimiser creates feedback information which is passed to the Manager; Section 7.4.3 explains exactly how the feedback information is created. The Manager passes the feedback information to the Gate Allocation Optimiser. The feedback information is added to the optimiser as additional constraints, which is explained in more detail in Section 7.4.4. The Gate Allocation Optimiser finds a new allocation, considering the routing feedback information. The new allocation is then passed to the Manager which then makes another routing request. If

the stopping condition, which is described in Section 7.4.5, is met the Manager stops the integration process. Otherwise new elements are added to the routing feedback information and the Gate Allocation Optimiser is executed again with more additional constraints created based upon the new elements of feedback. Pseudo code describing these steps for the integrated process is provided in Algorithm 3.

Algorithm 3 The integrated gate and route allocation algorithm

```
run the Gate Allocation Optimiser to obtain an initial allocation
pass the initial allocation to the Manager
run the Routing Optimiser for the initial allocation
obtain feedback information from the Routing Optimiser to the Manager
while stopping condition not met do
    pass the routing feedback information to the Gate Allocation Optimiser
    run the Gate Allocation Optimiser to obtain a new allocation
    pass the new allocation to the Manager
    run the Routing Optimiser for the new allocation to obtain new feedback information
    pass the new feedback information to the Manager
    add the new feedback information to the previous feedbacks
end while
```

Sometimes the algorithm does not find any allocation plan which reduces the number of conflicts. It may happen that all the possible allocations lead to the same number of conflicts. When no improvement in the number of conflicts is observed, the initial allocation plan found before the feedback constraints started to be added to the Gate Allocation Optimiser should be used. There is no point in modifying the allocation plan if the Routing Optimiser does not improve anyway.

7.4.3 Generation of the Routing Feedback

For every allocation plan, the Routing Optimiser generates new elements which are added to a feedback list here named the $feedList$. There is only one $feedList$ in the whole integration process, which slowly grows in each iteration. Each element of $feedList$ includes a combination of allocations c_i and a number of conflicts that it causes, named $conf_i$. A conflict is defined as a relation of a pair of allocations. When there are, for example, three allocations in c_i and they all meet at a crossing at the same time (one of the three flights

heading in the opposite direction of the other two) $conf_i$ is equal to two as two pairs of allocations are in conflict: the two flights heading in the same direction will not be in conflict.

Algorithm 4 describes the steps in which the new elements of $feedList$ are generated. In the first step a subset of all allocations CA, which take part in any conflict is

Algorithm 4 Feedback generation

Require: $feedList$

1: Create CA
2: **for** each a in CA **do**
3: Create $affected_a$
4: Create a set of all possible combinations ($comb$) from $affected_a$ (including a)
5: **for** each element c of $comb$ **do**
6: Find number of conflicts $conf$
7: **if** $conf > nc - 1$ **then**
8: Add a new element to $feedList$ which includes c_i and $conf_i$
9: **end if**
10: **end for**
11: **end for**

created. Then for each allocation a of CA all allocations which can possibly interact with a are detected and collected in a subset called $affected_a$ (step 4 in Algorithm 4). This is done by checking what is the time between arrival/departure of the flight for allocation a and departures/arrivals of flights for other allocations. Allocations which happen during different times of the day will not affect each other and they can be excluded from further consideration. Subset $comb$ of all possible combinations of allocations included in $affected_a$ and allocation a is then created (step 5 in Algorithm 4). For each combination c which includes nc allocations the number of conflicts ($conf$) is checked (steps 6-7 in Algorithm 4). If the number of conflicts is at least $nc-1$ a new element i which includes the combination and the number of conflicts is added to $feedList$ (c, $conf$, nc receive index i). If c generates less than $nc-1$ it is assumed redundant as a smaller combination (which includes less allocations) can generate the same conflicts.

For example if $affected_a$ contains two allocations: a_1 and a_2 the following combinations are included in $comb$: (a, a_1), (a, a_2) and (a, a_1, a_2). If combination (a, a_1) $(nc = 2)$

does not cause conflicts, (a, a_2) $(nc = 2)$ introduces one conflict and (a, a_1, a_2) $(nc = 3)$ introduces one conflict then only combination (a, a_2) will be added to $feedList$.

7.4.4 Routing Feedback in the Gate Allocation Optimiser

In each iteration of the integration process the $feedList$, which is larger than in the previous iteration, is passed to the Gate Allocation Optimiser. The Gate Allocation Optimiser uses the GAP model which is introduced in Chapter 5. Each element i of $feedList$ becomes an additional constraint in the model. The constraint forces an additional variable $rvar_i$ to become one if the combination of allocations included in the element of $feedList$ appears in the solution. Inequality 7.1 gives the formulation of the constraint.

$$\sum_{j=1}^{nc_i} X_{g_j,f_j} - rvar_i \leq nc_i - 1, \quad \forall i \in \{1, ..., l\} \tag{7.1}$$

where nc_i is the number of allocations in combination c_i, g_j is the gate for the allocation with index j and f_j is the flight for the allocation with index j. The variable $rvar_i$ is weighted in the objective function using the weight w_i. The values of w_i used should be in balance with other components of the objective function. The model should try to avoid the conflicting allocations without breaking other important elements of the objective function (which are presented in Section 5.3.4, Chapter 5).

One of the important components of the objective function is the component related to the time gaps between allocations. It is acceptable to shorten the time gaps which are larger than the preferred time gap (*goodGap* which was defined in Section 5.3.1, Chapter 5), but forcing gaps to be shorter than *goodGap* in order to reduce the possible routing conflicts is not preferred. When the time gap between two allocations is equal to *goodGap* a penalty equal to 1 is added to the objective function. The penalty w_i should be significantly smaller than that. Equation 7.2 was used to calculate w_i in these experiments. The number of conflicts $conf_i$ detected for the combination c_i (for which $rvar_i$ is equal to one) is divided by *goodGap*. w_i should be small enough in comparison to other objective function components given that $conf_i$ is rarely bigger than 2. Pairs of allocations are actually the most common combinations in the $feedList$, for which $conf_i$ is equal to one. When the *goodGap* is set to 60 minutes (as it was in Chapter 5), w_i is equal to 0.02.

$$w_i = \frac{conf_i}{goodGap} \tag{7.2}$$

$$\sum_{i=1}^{l}(w_i * rvar_i) \tag{7.3}$$

The sum of all weighted variables $rvar_i$ given by Formula 7.3 is added to the objective function of the GAP. The changes of the objective values due to the weights are analysed in Section 7.5.

7.4.5 Stopping Condition

The integration process is stopped when the Routing Optimiser does not return any new feedback information, i.e. the $feedList$ is the same before and after running the Routing Optimiser for a given allocation plan.

The size of $feedList$ is equal to zero in the initial run. It grows gradually during the integration process. In each iteration new conflicting configurations are detected by the Routing Optimiser for the allocation plan and are added to $feedList$. The Gate Allocation Optimiser modifies the allocation plan according to the elements of $feedList$, it tries to find solutions which avoid the conflicting allocations detected by Routing Optimiser so far. When the Routing Optimiser does not return any new elements (new conflicting configurations of allocations) to be added to $feedList$ the process stops. Lack of new elements to be added to $feedList$ means that the Gate Allocation Optimiser already knew (from previous loop iterations) about the conflicting configurations detected by the Routing Optimiser in the current allocation plan and that it all necessary conflict variables in the model are forced.

7.4.6 Seeding

Instead of starting from no knowledge about the routing problem we can seed part of the routing information in the initial run of the Gate Allocation Optimiser in order to reduce the number of loop iterations. Pairs of allocations are used in the seeding procedure, since these are the most common unavoidable conflicts.

A set of pairs (explained below) is created and the Routing Optimiser is run for each of the pairs. It checks if a pair causes a conflict. If a conflict is detected it is added to $feedList$. The procedure terminates when all of the pairs from the set are checked. Then the Gate Allocation Optimiser is started with $feedList$ filled with the information about pairs.

Only selected pairs are added to the set of pairs which are initially checked by the Routing Optimiser. There is no point in looking for conflicts between a flight which arrives and departs early in the morning and a flight which arrives and departs in the afternoon as they are not going to meet on the taxiway. The set of pairs includes pairs of allocations which will most likely meet at the airport. For this purpose, all allocations for which the time difference between the arrival of one and departure of another is smaller than 15 minutes are used in the seeding procedure.

The conflicts for pairs are relatively quick to check using the Routing Optimiser, so the seeding procedure is quite short. When the information about the pairs is already seeded in, the feedback loop process only introduces information about more complex combinations of allocations, for example three or more flights which together cause conflicts. It is observed in Section 7.5.3 that the seeding procedure did actually reduce the number of loop iterations.

7.5 Results

The preliminary results obtained using the framework are presented in this section. The aim is to establish if the integration system reduces the number of conflicts, what is the influence of the integration process on the allocation quality and how long is the integration process. A set of instances for which the Gate Allocation Optimiser is relatively quick to solve have been chosen to experiment and to observe the basic aspects of the results. The following instances were used in the experiments: 1(1), 2(1), 4(1), 6(1), 1(2), 2(2), 2(3), 2(4), 2(5), 2(6), 2(7), 1(3), 6(3) (see Section 5.5.2, Chapter 5 for the calculation times).

The aircraft belonging to different terminals interact with each other when they taxi to/from the gates. This is potentially a source of more conflicts which are not considered in the experiments presented here. Section 7.6 describes an alternative version of the system which uses the Receding Horizon (RH) decomposition (see Chapter 6 for more details). All three terminals could be solved simultaneously, to resolve more routing conflicts, using the alternative version.

Section 7.5.1 gives the parameter values which were used in the experiments. Section 7.5.2 focuses on using the system without the seeding procedure. It is shown for one, chosen instance how the number of routing conflict variables set to one by the Gate Allocation Optimiser and the number of conflicts detected by the Routing Optimiser change when the integration process proceeds. Next, the results obtained using more instances are

discussed. For each test instance the relative change of the objective value is shown. The number of routing conflicts resolved as well as the number of loop cycles which was solved till the stopping criterion was met are also presented. Section 7.5.3 focuses on the results with the seeding procedure. Similarly to the version without seeding, it is shown, for each dataset, what the relative change of the objective function is, how many routing conflicts were resolved and how many iterations of the loop had to be solved in order to meet the stopping condition.

7.5.1 Parameters

The parameters of the Gate Allocation Optimiser (the GAP model) are set as in Chapter 5. There are two additional parameters which have to be set for the Routing Optimiser. One is the speed of aircraft on taxiways and the other is the size of the time margin m (see Figure 7.3). m is set to 5 minutes, the average speed of aircraft used in the experiments is 10 km/h.

7.5.2 Results without Seeding

This system, which is based upon an information flow between the Routing Optimiser and the GAP the Gate Allocation Optimiser, causes a gradual change in the allocation plan. As explained in Section 7.4.5 in each loop iteration the Gate Allocation Optimiser receives a larger feedback list (including new conflicting configurations), takes it into consideration (sets appropriate routing variables if needed) and suggests a new modified allocation plan which is validated by the Routing Optimiser. At some point the Routing Optimiser does not find any new conflicts in the modified allocation plan to be added to the feedback list, which means the Gate Allocation Optimiser already knew about all the conflicting configurations from the previous loop iterations. This is when the iterative integration process stops.

Figure 7.5 uses instance 4(1) to illustrate the process. The vertical direction of the figure indicates the number of conflicts, the horizontal indicates the steps in the process (iterations of the feedback loop). The blue colour shows the number of conflicts detected by the Routing Optimiser. The red line shows the number of routing variables set to one by the Gate Allocation Optimiser.

It can be observed in Figure 7.5 that the number of conflicts detected by the Routing Optimiser is equal to eight at the beginning of the process. It drops gradually, as

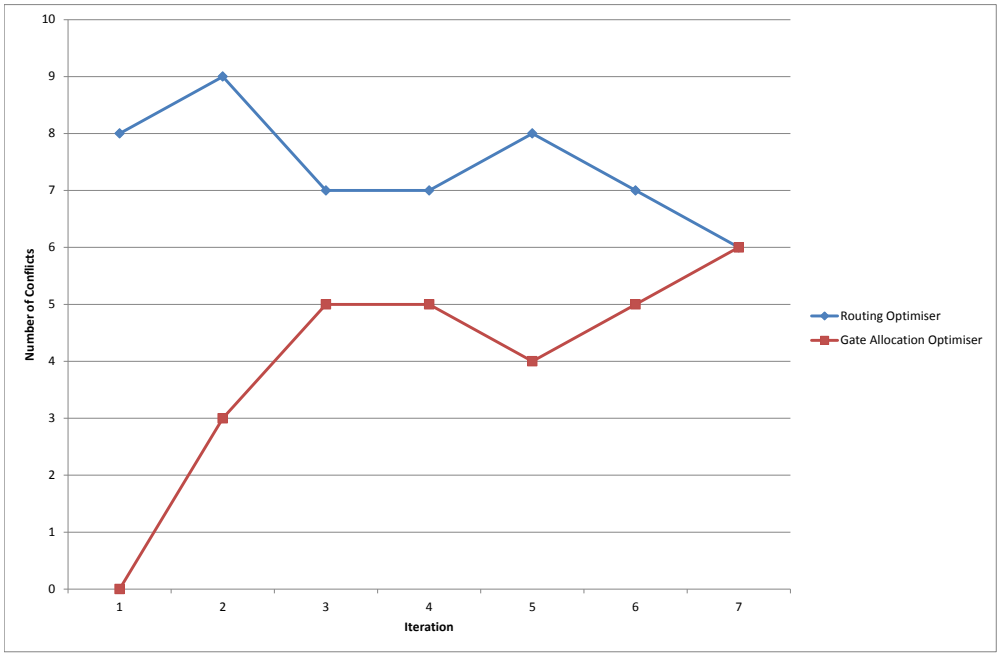


FIGURE 7.5: The number of conflicts detected by the Routing Optimiser and the number of conflict variables set to one by the Gate Allocation Optimiser recorded for each iteration, results obtained for instance 4(1).

the process progresses. No routing variables are forced at the beginning of the process, so the red line starts from zero. The number of variables set to one increases gradually until it is equal to the number of conflicts detected by the Routing Optimiser. The two lines meet, the process is stopped in the seventh iteration. Six routing conflict variables were forced in Gate Allocation Optimiser (the GAP model). The number of conflicts detected on the taxiways was reduced from eight to six.

The process was recorded for each of the test instances. Table 7.1 presents the recorded values. The column labelled "Iterations" displays the number of iterations that had to be solved before the stopping criterion was reached. The column labelled "Initial Conflicts" shows the number of conflicts detected by the Routing Optimiser initially, while the column labelled "Final Conflicts" presents the number of conflicts detected by the Routing Optimiser when the integration process is finished. The column labelled "Objective"

Day(Terminal)	Iterations	Initial Conflicts	Final Conflicts	Objective [%]	Time [min]
1(1)	5	6	4	0.03	13.291
2(1)	6	7	6	0.01	17.605
4(1)	7	8	6	0.01	28.525
6(1)	5	9	8	0.02	16.771
1(2)	8	3	3	0.06	0.755
2(2)	9	4	4	0.02	0.228
3(2)	2	1	1	0.00	0.384
4(2)	7	2	2	0.01	1.389
5(2)	5	7	7	0.05	1.691
6(2)	13	6	6	0.07	3.057
7(2)	7	3	3	0.02	0.643
1(3)	20	8	1	0.04	28.677
6(3)	5	2	2	0.01	1.591

TABLE 7.1: Results of the integration process without the seeding procedure.

shows the difference between the initial objective value and the final objective value of the Gate Allocation Optimiser, as a percentage of the initial value. It indicates what the impact of the integration process on the objective function values was. The column labelled "Time" displays the total calculation time of all of the iterations, in minutes.

The integration process has a positive impact on the number of routing conflicts. The positive impact is observed for all instances for terminal T1 and one for terminal T3. In total thirteen conflicts were resolved out of sixty six by only slight modifications of the allocation plan. It can be observed from Table 7.1 that the objective value changes are very small even when several routing conflict variables were set to one. This is related to the penalty which is set for the routing conflict element of the objective function. A stronger penalty would result in more resolved conflicts but would also have a stronger impact on the objective value.

It is however not always possible to reduce the number of conflicts by changing the allocation plan. This was clearly observed for the instances coming from terminal T2, and may be caused by a bottleneck which is on the path between terminal T2 and the runways. The bottleneck is a major access problem for terminal T2, as all of its gates are separated from the runways by the bottleneck. There is a chance that allocating the conflicting aircraft to any gate of the terminal would result in the bottleneck conflict. Some, but not all, gates belonging to terminal T1 are also placed behind the bottleneck. It is easier to resolve the

bottleneck conflicts when terminal T1 is solved since one of the flights which is involved in a bottleneck conflict can usually be moved to one of the gates at T1 which are not behind the bottleneck.

The number of iterations varies between the datasets, the smallest is 2 and the largest 20. This is related to the structure of the problem and it is impossible to predict in advance how many iterations will be needed. The large number of iterations is acceptable only if the allocation can be solved quickly (as in the presented experiments). But for harder datasets the number of iterations should be reduced as much as possible, otherwise the calculation times will become extremely long. A seeding procedure was implemented in order to reduce the number of iterations. The results obtained for the same datasets with the seeding procedure are presented in Section 7.5.3.

7.5.3 Seeding Procedure Impact

The seeding procedure is an initial stage which is solved before the Gate Allocation Optimiser is run for the first time. The results of the seeding procedure are passed to the Gate Allocation Optimiser, so that it has some initial knowledge about the routing problem. The seeding procedure influences the number of iterations and the total calculation times, the other results should remain the same.

Table 7.2 shows the results obtained when the seeding procedure is added to the system. Similarly to the results from the previous section the number of iterations, the number of routing conflicts before and after the integration process, the values of $rvar$, the objective value relative change and the total calculation time in minutes is presented in the table. The column "Objective" shows the difference between the objective value without the seeding procedure (the same as for the results shown in Table 7.1) and the final objective value change as a percentage of the objective value without the seeding procedure. Similarly the initial number of conflicts given in the column labelled "Initial conflicts" is the same as given in Table 7.1 which is the number of conflicts detected by the Routing Optimiser for allocation plan with no information from the Routing Optimiser.

It can be observed from Table 7.2 that the only elements which are different from Table 7.1 are the number of iterations and the calculation times. This shows that the system is implemented correctly and behaves as expected. The number of iterations and the calculation times obtained for the two versions of the system, without and with seeding,

Day(Terminal)	Iteration	Initial Conflicts	Final Conflicts	Objective [%]	Time [min]
1(1)	1	6	4	0.03	11.859
2(1)	1	7	6	0.01	14.352
4(1)	1	8	6	0.01	12.928
6(1)	3	9	8	0.02	19.638
1(2)	1	3	3	0.06	1.611
2(2)	1	4	4	0.02	2.565
3(2)	1	1	1	0.00	0.806
4(2)	1	2	2	0.01	1.441
5(2)	3	7	7	0.05	4.001
6(2)	13	6	6	0.07	5.400
7(2)	1	3	3	0.02	1.537
1(3)	1	8	1	0.04	6.619
6(3)	1	2	2	0.01	2.853

TABLE 7.2: Results of the integration process with the seeding procedure.

are displayed in Figure 7.6 and Figure 7.7 so that it is easier to compare the results.

Figure 7.6 shows the number of iterations which were solved for each dataset using the system with (red bars) and without (blue bars) the seeding procedure. In almost all cases the number of iterations is significantly smaller when the seeding procedure is in use, only for instance 6(2) was the same number of iterations needed with and without seeding. The total number of iterations solved without the seeding procedure is 99 while the total number of iterations with seeding is 29. Thanks to the seeding, the system had to solve far less iterations. The number of iterations influences the calculation time, however the seeding itself also takes time. The actually question is therefore if the system is faster with or without the procedure.

Figure 7.7 shows the calculation times taken by the system with (red bars) and without (blue bars) the seeding procedure. The total calculation time with seeding is 85.7 minutes while the total calculation time without is 114.7 minutes, so seeding has a positive impact on the calculation times in general. It is usually useful when the calculation time (without seeding) is relatively long. There are five cases in the experiments which take longer than 5 minutes to solve without seeding. The calculation time improves visibly in four out of the five cases. In the cases which can be solved quickly without the seeding procedure there was no benefit from adding it.

The results presented above indicate the positive impact of the framework on

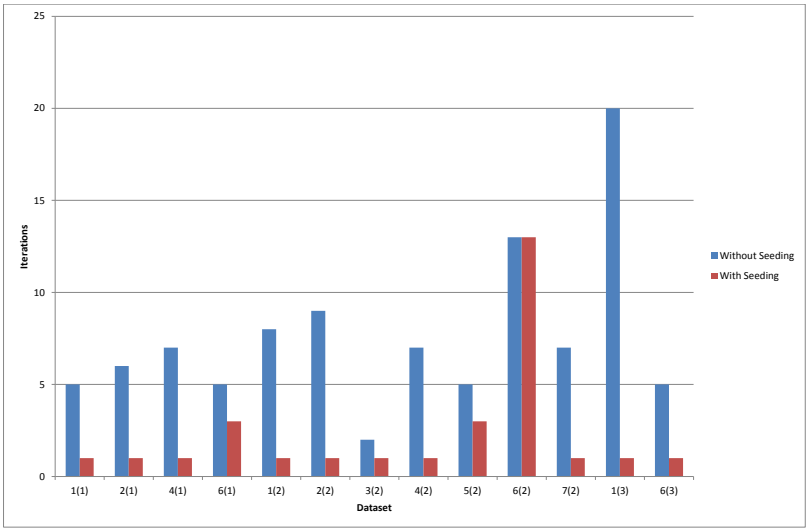


FIGURE 7.6: The number of iterations needed to be solved to meet the stopping criterion with, and without the seeding procedure.

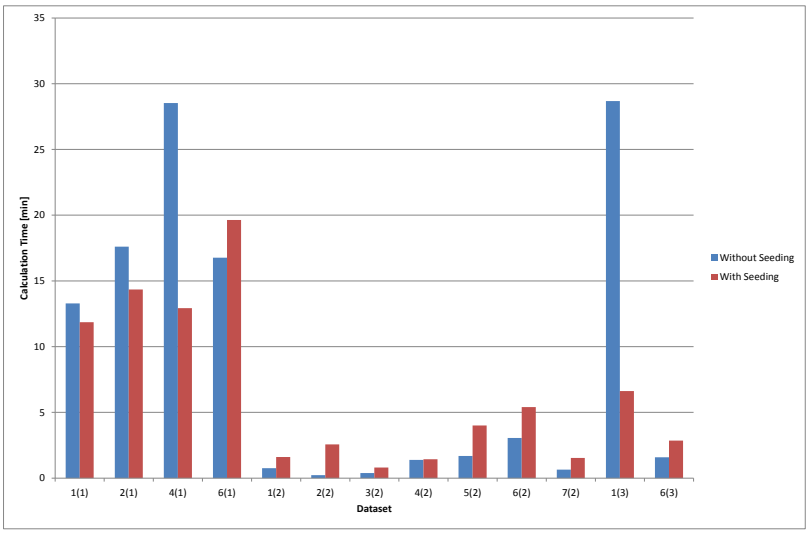


FIGURE 7.7: Calculation times taken by the system with and without the seeding procedure.

the number of routing conflicts. They do not consider the conflicts between flights from different terminals. The version of the system which is described in Section 7.6 should allow the solution of more terminals at once and the resolution of even more conflicts.

7.6 Framework with RH Decomposition

Applying the RH decomposition should allow the solution of harder instances of the problem and possibly to resolve more conflicts occurring at the taxiways.

The RH method would be run on top of the integration system. Instead of using the full GAP problem the framework would receive only a part of the problem at a time, one window. The integration system would be run for the window, exactly the same way as it is run for the full problem and stopped when the stopping criterion is met. The procedure would be repeated for every next window position unless all flights from the problem are allocated. One feedback list would be used in the whole system so the feedbacks from the previous window positions would be used in the following window position. The alternative implementation of the system would be an interesting future work.

7.7 Conclusion

Solving the allocation problem in separation from the ground movement problem which is often observed in the real world planning may result in many conflicting situations on the taxiways. The conflicting situations will be resolved during the day of operation so that there are as few delays as possible. The research presented in this chapter shows that the conflicting situations which are expected to occur on the taxiways can be taken into consideration in the early stage of allocation planning. The allocation plan can be slightly modified according to the routing information so that the ground movement problem becomes easier to solve during the day of operation.

The framework which allows information flow between the ground problem and the gate allocation problem appears to be a useful tool in the preliminary study. The routing conflicts are taken into consideration in the allocation planning and some of the conflicts can be resolved. An additional seeding procedure has been added to the original implementation of the framework which helps to reduce the number of iterations solved and to improve the longer calculation times.

The RH decomposition could also be used in the system to decompose large instances of the GAP and the implementation methods were briefly discussed in this chapter.

Ground movement and gate allocation are very closely related problems. The integration of the two systems is a very important issue which is hardly ever mentioned in the literature or in the real world. The chapter shows preliminary results which not only prove the very close link but also show that a slight modification of one problem solution can sometimes have a very positive impact on the other problem solution.

It is also important to realise that the feedback framework has more general applications. Different allocation models or routing algorithms could be used within the framework, which remains the same and still provides the important link between the two problems. Future work includes using a ground movement routing algorithm which is described in [76] in the Routing Optimiser instead of the greedy ground movement routing algorithm. A prototype has been already created by Christofas Stergianos who is currently working on the problem.

Chapter 8

Conclusion

8.1 General Summary

The recurring topic of the different chapters of this thesis is the integration of the gate allocation problem (GAP) with the ground movement problem. Both the modelling aspects and the solution methods which aim to bring the two problems closer are considered in the research.

Two new modelling aspects related to this integration are discussed. The first new modelling aspect which narrows the gap between the two problems is the new constraint which resolves conflicts at the gates within gate groups. It is included in the MIP model of the GAP described in Chapter 5. The model is further extended in Chapter 7 where the second new modelling aspect, a new feedback constraint, is added to it. The constraint allows the addition of the information from the taxiways to the model so that the routing conflicts can be resolved when the allocation is planned. Another modelling aspect which also has an impact upon the routing is maximization of the time gaps between the allocations. If the gap is large enough, an aircraft can be kept on gate longer before departing when there is taxiway congestion. Although the idea of maximisation of time gaps was already present in the literature the proposed formulation of the time gap constraint is slightly different from [7] as it takes into consideration the preferred time gap between allocations.

The RH approach, which is initially discussed in Chapter 4 and then investigated in more depth in Chapter 6, can be used to solve larger instances of the proposed GAP formulation more efficiently and to resolve more conflicts at the gates. The integration framework, which is based upon a feedback loop algorithm presented in Chapter 7, is used to pass the information between the two problems. Thanks to this the GAP model knows

about the routing conflicts and can modify the allocation plan accordingly.

8.2 Key Results

The key results of the research presented in the thesis are summed up below.

- **The group conflict constraint:** It was shown that the new conflict constraint included in the advanced model formulation (Chapter 5) reduces the number of expected conflicts within groups of gates. A stronger formulation of the constraint could however be applied, especially when more than one terminal is solved.

- **The GAP model parameters:** It was shown that the size of the preferred time gap between allocations has a crucial meaning for the model calculation times (shown in Chapter 4). Moreover the objective element which aims to increase the gaps between allocations has the heaviest influence on the model calculation time (shown in Chapter 5).

- **Application of the RH approach:** The RH method has been applied to decompose the GAP for the first time and the application was successful. It was shown that the RH method can be very useful when the parameters are set appropriately and the parameter effects on the results were investigated.

- **Parameters of the static RH approach:** The obtained results show a strong relation between the size of the window and the calculation time for the static implementations of the RH approach. The larger the window the longer it takes to solve the problem and the better objective function is reached. The size of the shift also influences the times, the larger the shift is the shorter it takes to solve the problem for the same window size.

- **Parameters of the dynamic RH approach:** It is harder to observe the relations between the parameters and the calculation times or the objective values for the dynamic implementations as the size of the window varies dynamically at each window position. It was observed that the dynamic approaches were avoiding the exceptionally short times and the exceptionally long times, and often resulted in times in the middle of those taken by the static implementations.

- **The best implementation of the RH approach:** The new dynamic time-oriented implementation of the RH approach (shift size equal to 22% of the dataset duration) seemed to be the best of all the tested implementations when a good trade-off between the calculation time and the objective value was sought. The dynamic approaches have an additional advantage over the static approaches of using only one parameter (the shift size) which has to be set explicitly (the window size is dynamic). This simplifies the tuning of the RH method.

- **Universal framework:** The system described in Chapter 7 is a universal framework within which various allocation models and routing algorithms can cooperate. The framework allows routing conflicts to be taken into consideration in the allocation planning and to resolve the routing conflicts if possible. The preliminary results provided show that the two problems can be sufficiently linked.

- **The seeding procedure:** The seeding procedure, which is added to the original implementation of the framework, helps to reduce the number of iterations solved and improves the longer calculation times.

8.3 Future Work

Potential directions for future research are listed below.

- **RH best configuration:** The best configuration of the RH method was suggested for the instances which were used in the experiments. It would be interesting to experiment with a larger set of instances and possibly with other problems which are similar to the GAP. A more general conclusion about the best configuration could be drawn based upon such experiments.

- **Deciding about the solution method in practice:** Two solution methods of the GAP were discussed in the thesis: the exact method and the RH approach. It was observed that some of the instances are quick to solve exactly while other require the RH approach to be solved in reasonable time. Practical applications would require some measure which would allow to choose between the methods. The measure is expected to be related to the average number of flights per gate calculated based upon the flight schedule and the number of gates, further investigation is required.

- **The group conflict constraint of the GAP model:** Additional constraints which could limit the number of conflicts for each group of gates individually could be added to the current model formulation. This would increase the number of resolved conflicts at the gates but also the number of variables in the model. It may however be useful to have both the overall constraint and one per group to avoid the trade-offs between groups. Further investigation is needed.

- **Replace the ground movement algorithm which is used in the framework:** Ongoing research which is being performed by Christofas Stergianos aims to replace the greedy ground movement algorithm which is used in here with a variant of the algorithm which is described in [76]. The algorithm does not require all of the paths to be known in advance and can therefore be applied to automate solution of more complex ground movement problems from larger airports.

- **The RH method and the framework:** The idea of using the RH method in the integrated system is described theoretically in this thesis. It would be interesting to test and to compare the results of various implementations as it should be able to solve multi-terminal instances and resolve the routing conflicts which occur between flights from various terminals.

- **The on-line version of the framework:** The research directions identified after the discussions with airports (Section 3.4, Chapter 3) include the on-line version of the GAP. Creation of an on-line version of the framework which is presented in Chapter 7 could have a very strong practical impact. The on-line version is required to be very time efficient. Faster heuristic solution methods would therefore have to be introduced at some point since solvers are not fast enough to solve complex problems in real time.

References

[1] E. Aarts, J. Korst, and W. Michiels. *Search Methodologies - Simulated Annealing*, chapter Simulated Annealing, pages 187–210. Springer, 2005.

[2] K. Andersson, F. Carr, E. Feron, and W. Hall. Analysis and modeling of ground operations at hub airports. In *The 3rd USA/Europe Air Trac Management R&D seminar*, 2000.

[3] M. Angel and C. Esteve. Network design: taxi planning. *Annals of Operations Research*, 157(1):135–151, 2008.

[4] J. A. D. Atkin. *Online decision support for take-off runway scheduling at London Heathrow airport*. PhD thesis, School of Computer Science, 2008.

[5] J. A. D. Atkin, G. D. Maere, E. K. Burke, and J. S. Greenwood. Addressing the push-back time allocation problem at heathrow airport. *Transportation Science*, 47(4):584–602, 2012.

[6] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46:316–329, 1996.

[7] A. Bolat. Procedures for providing robust gate assignments for arriving aircraft. *European Journal of Operational Research*, 120(1):63 – 80, 2000.

[8] F. Borrelli, A. Bemporad, M. Fodor, and D. Hrovat. An mpc/hybrid system approach to traction control. *Control Systems Technology, IEEE Transactions on*, 14(3):541–552, May 2006.

[9] R. Bosch and M. Trick. *Search Methodologies - Integer Programing*, chapter Integer Programing, pages 69 – 95. Springer, 2005.

[10] R. Bronson and G. Naadimuthu. *Schaum's Outline of Operations Research*. Schaum's Outline Series. McGraw-Hill Education, 1997.

[11] E. K. Burke and G. Kendall. *Search Methodologies - Introduction*, chapter Introduction, pages 5 –18. Springer, 2005.

[12] E. K. Burke, G. Kendall, J. Newall, E. Hart, P. Ross, and S. Schulenburg. Hyper-heuristics: An emerging direction in modern search technology. In *Handbook of Meta-heuristics*, International Series in Operations Research & Management Science, chapter 16, pages 457–474. ., 2003.

[13] S. Chand, R. Traub, and R. Uzsoy. Rolling horizon procedures for the single machine deterministic total completion time scheduling problem with release dates. *Annals of Operations Research*, 70(0):115–125, 1997.

[14] C. Chang. *Flight Sequencing and Gate Assignments at Airport Hubs*. University of Maryland at College Park, 1994.

[15] V. Cheng and D. Foyle. Automation tools for enhancing ground-operation situation awareness and flow eciency. In *The AIAA Guidance, Navigation, and Con- trol Conference and Exhibit*, 2002.

[16] Y. Cheng. A knowledge-based airport gate assignment system integrated with mathematical programming. *Computers & Industrial Engineering*, 32(4):837–852, 1997.

[17] Y. Cheng. Solving push-out conflicts in apron taxiways of airports by a network-based simulation. *Computers & Industrial Engineering*, 34(2):351–369, April 1998.

[18] G. Clare and A. Richards. Airport ground operations optimizer. In *The EU- ROCONTROL 8th Innovative Research Workshop & Exhibition (INO)*, 2009.

[19] S. Dasgupta, C. H. Papadimitriou, and U. V. Vazirani. *Algorithms*. McGraw-Hill, 2006.

[20] R. Deau, J. Gotteland, and N. Durand. Airport surface management and runways scheduling. In *The 8th USA/Europe Air Trac Management Research and Development Seminar*, 2009.

[21] G. Diepen. *Column Generation Algorithms for Machine Scheduling and Integrated Airport Planning*. Guido Diepen, 2008.

[22] G. Diepen, J. M. Akker, and J. Hoogeveen. Integrated gate and bus assignment at amsterdam airport schiphol. In R. Ahuja, R. Möhring, and C. Zaroliagis, editors, *Robust and Online Large-Scale Optimization*, volume 5868 of *Lecture Notes in Computer Science*, pages 338–353. Springer Berlin Heidelberg, 2009.

[23] H. Ding, A. Lim, B. Rodrigues, and Y. Zhu. Aircraft and gate scheduling optimization at airports. In *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on*, 2004.

[24] H. Ding, A. Lim, B. Rodrigues, and Y. Zhu. The over-constrained airport gate assignment problem. *Computers and Operations Research*, 32:1867–1880, July 2005.

[25] W. Dong-Xuan and L. Chang-You. Fuzzy model and optimization for airport gate assignment problem. In *Intelligent Computing and Intelligent Systems, 2009. ICIS 2009. IEEE International Conference on*, volume 2, pages 828–832, Nov 2009.

[26] U. Dorndorf, A. Drexl, Y. Nikulin, and E. Pesch. Flight gate scheduling: State-of-the-art and recent developments. *Omega*, 35:326 – 334, 2007.

[27] U. Dorndorf, F. Jaehn, and E. Pesch. Modelling robust flight-gate scheduling as a clique partitioning problem. *Transportation Science*, 42(3):292–301, Aug. 2008.

[28] U. Dorndorf, F. Jaehn, and E. Pesch. Flight gate scheduling with respect to a reference schedule. *Annals of Operations Research*, 194(1):177–187, 2012.

[29] K. A. Dowsland. *Search Methodologies - Classical Techniques*, chapter Classical Techniques, pages 19–68. Springer, 2005.

[30] A. Drexl and Y. Nikulin. Multicriteria airport gate assignment and pareto simulated annealing. *IIE Transactions*, 40(4):385–397, 2008.

[31] Flightradar24. http://www.flightradar24.com/.

[32] L. Fortnow. The status of the p versus np problem. *Commun. ACM*, 52(9):78–86, Sept. 2009.

[33] E. C. Freuder and M. Wallace. *Search Methodologies - Constraint Programming*, chapter Constraint Programming, pages 239–272. Springer, 2005.

[34] M. C. Fu. Optimization via simulation: A review. *Annals of Operations Research*, 53(1):199–247, 1994.

[35] H. M. Genç, E. O. Kaan, E. İbrahim, B. M. Fatih, and G. B. Onaran. A stochastic neighborhood search approach for airport gate assignment problem. *Expert Syst. Appl.*, 39(1):316–327, Jan. 2012.

[36] M. Gendreau and J.-Y. Potvin. *Search Methodologies - Tabu Search*, chapter Tabu Search, pages 165 – 186. Springer, 2005.

[37] G.Keith and A. Richards. Optimization of taxiway routing and runway scheduling. In *AIAA Guidance, Navigation and Control Conference, Honolulu, HI, USA*, 2008.

[38] A. Goodall. *The guide to expert systems*. Learned Information, 1985.

[39] G. D. Gosling. Design of an expert system for aircraft gate assignment. *Transportation Research Part A: General*, 24(1):59 – 69, 1990.

[40] J.-B. Gotteland, N. Durand, J.-M. Alliot, and E. Page. Aircraft ground traffic optimization. In *ATM2001*, 2001.

[41] Y. Gu and C. A. Chung. Genetic algorithm approach to aircraft gate reassignment problem. *Journal of Transportation Engineering*, 125(5):384–389, September/October 1999.

[42] A. Haghani and M.-C. Chen. Optimizing gate assignments at airport terminals. *Transportation Research Part A: Policy and Practice*, 32(6):437 – 454, 1998.

[43] S. G. Hamzawi. Management and planning of airport gate capacity: a microcomputer-based gate assignment simulation model. *Transportation Planning and Technology*, 11(3):189–202, 1986.

[44] F. He and R. Qu. Constraint-directed local search to nurse rostering problems. In *6th International Workshop on Local Search Techniques in Constraint Satisfaction (LSCS 2009)*, Lisbon, Portugal, September 2009.

[45] K. Healy and L. W. Schruben. Retrospective simulation response optimization. In *Proceedings of the 23rd Conference on Winter Simulation*, WSC '91, pages 901–906, Washington, DC, USA, 1991. IEEE Computer Society.

[46] J. G. Herrero, A. Berlanga, J. M. Molina, and J. R. Casar. Methods for operations planning in airport decision support systems. *Applied Intelligence*, 22(3):183–206, 2005.

[47] F. S. Hillier and G. J. Lieberman. *Introduction to Operations Research, 4th Ed.* Holden-Day, Inc., San Francisco, CA, USA, 1986.

[48] X.-B. Hu and E. D. Paolo. An efficient genetic algorithm with uniform crossover for the multi-objective airport gate assignment problem. In C.-K. Goh, Y.-S. Ong, and K. Tan, editors, *Multi-Objective Memetic Algorithms*, volume 171 of *Studies in Computational Intelligence*, pages 71–89. Springer Berlin Heidelberg, 2009.

[49] J.-B.Gotteland and N. Durand. Genetic algorithms applied to airport ground traffic optimization. In *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on*, volume 1, pages 544–551 Vol.1, 2003.

[50] R. Jans. Solving lot-sizing problems on parallel identical machines using symmetry-breaking constraints. *INFORMS Journal on Computing*, 21(1):123–136, Jan. 2009.

[51] R. Jans and J. Desrosiers. Efficient symmetry breaking formulations for the job grouping problem. *Computers and Operations Research*, 40(4):1132–1142, Apr. 2013.

[52] G.-S. Jo, J.-J. Jung, and C.-Y. Yang. Expert system for scheduling in an airline gate allocation. *Expert Systems with Applications*, 13(4):275 – 282, 1997. Selected Papers from the PACES/SPICIS'97 Conference.

[53] S. H. Kim and E. Feron. Impact of gate assignment on departure metering. *Intelligent Transportation Systems, IEEE Transactions on*, 15(2):699–709, April 2014.

[54] S. H. Kim, E. Feron, and J. P. Clarke. Assigning gates by resolving physical conflicts. In *AIAA Guidance, Navigation, and Control Conference*, 2009.

[55] S. H. Kim, E. Feron, and J. P. Clarke. Airport gate assignment that minimizes passenger flow in terminals and aircraft congestion on ramps. In *Guidance, Navigation, and Control and Co-located Conferences*, pages –. American Institute of Aeronautics and Astronautics, Aug. 2010.

[56] P. Kochel. Simulation optimisation: Approaches, examples and experiences. Technical report, TU Chemnitz, 2009.

[57] P. Kumar and M. Bierlaire. Multi-objective airport gate assignment problem. In *Proceedings of the Swiss Transport Research Conference*, Monte Verita, CH, 2011.

[58] L. H. Lee, H. C. Huang, and P. Huang. Flight assignment plan for an air cargo inbound terminal. In *Winter Simulation Conference*, pages 1872–1881. WSC, 2010.

[59] C. Lesire. Automatic planning of ground traffic. In *The AIAA Aerospace Sciences Meeting (ASM)*, 2009.

[60] C. Lesire. Iterative planning of airport ground movements. In *The 4th International Conference on Research in Air Transportation (ICRAT),*, 2010.

[61] L. Liberti. Automatic generation of symmetry-breaking constraints. In *Proceedings of the 2nd International Conference on Combinatorial Optimization and Applications*, COCOA 2008, pages 328–338, Berlin, Heidelberg, 2008. Springer-Verlag.

[62] L. Liberti and J. Ostrowski. Stabilizer-based symmetry breaking constraints for mathematical programs. *Journal of Global Optimization*, 60(2):183–194, Oct. 2014.

[63] A. Lim, B. Rodrigues, and Y. Zhu. Airport gate scheduling with time windows. *Artificial Intelligence Review*, 24(1):5–31, 2005.

[64] M. Lubbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53:1007–1023, 2004.

[65] B. Maharjan and T. I. Matis. An optimization model for gate reassignment in response to flight delays. *Journal of Air Transport Management*, 17(4):256 – 261, 2011.

[66] R. S. Mangoubi and D. F. X. Mathaisel. Optimizing gate assignments at airport terminals. *Transportation Science*, 19:173–188, 1985.

[67] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789 – 814, 2000.

[68] Z. Michalewicz and D. Fogel. *How to solve it - chaper 4*, chapter Chapter 4 - Traditional Methods Part 2, pages 84 – 109. Springer, 2004.

[69] U. M. Neuman and J. A. D. Atkin. Airport gate assignment considering ground movement. In D. Pacino, S. Voß, and R. Jensen, editors, *Computational Logistics*, volume 8197 of *Lecture Notes in Computer Science*, pages 184–198. Springer Berlin Heidelberg, 2013.

[70] Y. Nikulin and A. Drexl. Theoretical aspects of multicriteria flight gate scheduling: deterministic and fuzzy models. *Journal of Scheduling*, 13(3):261–280, 2010.

[71] J. Ostrowski, A. Vannelli, and F. M. Anjos. *Symmetry in scheduling problems*. Groupe d'études et de recherche en analyse des décisions, 2010.

[72] M. Padberg and G. Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Rev.*, 33:60–100, February 1991.

[73] E. Perea-López, B. E. Ydstie, and I. E. Grossmann. A model predictive control strategy for supply chain optimization. *Computers and Chemical Engineering*, 27(8-9):1201–1218, 2003.

[74] B. Pesic, N. Durand, and J.-M. Alliot. Aircraft ground traffic optimisation using a genetic algorithm. In L. Spector, E. D. Goodman, A. Wu, W. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 1397–1404, San Francisco, California, USA, 7-11 July 2001. Morgan Kaufmann.

[75] S. Rathinam, J. Montoy, and Y. Jung. An optimization model for reducing aircraft taxi times at the dallas fort worth international airport. In *26th International Congress of the Aeronautical Sciences (ICAS)*, 2008.

[76] S. Ravizza, J. Atkin, and E. Burke. A more realistic approach for airport ground movement optimisation with stand holding. *Journal of Scheduling*, 17(5):507–520, 2014.

[77] P. C. Roling and H. Visser. Optimal airport surface traffic planning using mixed-integer linear programming. *International Journal of Aerospace Engineering*, 2008:11, 2008.

[78] H. D. Sherali and J. C. Smith. Improving discrete model representations via symmetry considerations. *Management Science*, 47(10):1396–1407, Oct. 2001.

[79] N. Simpson. Multiple level production planning in rolling horizon assembly environments. *European Journal of Operational Research*, 114(1):15 – 28, 1999.

[80] J. C. Spall. Stochastic optimization. In J. E. Gentle, W. K. Härdle, and Y. Mori, editors, *Handbook of Computational Statistics*, Springer Handbooks of Computational Statistics, pages 173–201. Springer Berlin Heidelberg, 2012.

[81] K. Tanaka. *An introduction to fuzzy logic for practical applications.* Springer, 1997.

[82] L. Tang, S. Jiang, and J. Liu. Rolling horizon approach for dynamic parallel machine scheduling problem with release times. *Industrial and Engineering Chemistry Research*, 49:381–389, 2010.

[83] H. P. Williams. *Model Building in Mathematical Programming, 4th Edition.* Wiley, 4 edition, Oct. 1999.

[84] J. Xu and G. Bailey. The airport gate assignment problem: Mathematical model and a tabu search algorithm. *Hawaii International Conference on System Sciences*, 3:3032, 2001.

[85] S. Yan and C.-M. Huo. Optimization of multiple objective gate assignments. *Transportation Research Part A: Policy and Practice*, 35(5):413 – 432, 2001.

[86] S. Yan, C.-Y. Shieh, and M. Chen. A simulation framework for evaluating airport gate assignments. *Transportation Research Part A: Policy and Practice*, 36(10):885 – 898, 2002.

Appendix A

Size Groups of Aircraft

Manufacturer	Type	IATA Code	Wingspan [m]	Length [m]
SIZE 1				
British Aerospace	Jetstream 31	J31	15.85	14.364
Dornier	Do228	D28	16.97	16.56
British Aerospace	Jetstream 41	J41	18.42	19.33
Dassault	Falcon 50	DF3	18.86	18.5
Dassault	Falcon 900	DF3	19.33	20.21
Canadair	Challenger 601	CCJ	19.61	20.85
Bombardier-de Havilland	DHC6 Twin Otter	DHT	19.81	15.77
Embraer	ERJ 135	ER3	20.04	26.33
Embraer	ERJ 140	ERD	20.04	28.45
Embraer	ERJ 145	ER4	20.04	29.87
Dornier	Do328	D38	20.98	21.22
Dornier	Do328JET	FRJ	20.98	21.22
Canadair	Regional Jet	CRJ	21.21	26.77
Saab	SB340	SF3	21.44	19.73
SIZE 2				
Shorts	S330	SH3	22.76	17.69
Shorts	S360	SH6	22.81	21.59
Canadair	Regional Jet 900	CR9	23.2	36.4
Canadair	Regional Jet 700	CR7	23.24	32.51
Grumman	Gulfstream 3	GRJ	23.72	25.32
Grumman	Gulfstream 4	GRJ	23.72	26.9
Aerospatiale/Alenia	ATR-42	AT4	24.572	22.67
Saab	SB2000	S20	24.76	27.28
Fokker	F28-3000	F23	25.07	27.4
Fokker	F28-4000	F24	25.07	29.61
Bombardier-de Havilland	Dash-8-100	DH1	25.69	22.25
CASA	CN-235	CS5	25.81	21.4
Embraer	ERJ170	E70	26	29.9
Embraer	ERJ175	E75	26	31.68
Avro	RJ100	AR1	26.21	30.99
Avro	RJ115	ARJ	26.21	30.99
Avro	RJ70	AR7	26.21	26.2
Avro	RJ85	AR8	26.21	28.6
British Aerospace	BAe146-300	143	26.339	30.995
British Aerospace	BAe146-100	141	26.34	26.19
British Aerospace	BAe146-200	142	26.34	28.58
Aerospatiale/Alenia	ATR-72	AT72	27.05	27.166
Dornier	Do728	D78	27.12	27.04
Dornier	Do728JET	FA7	27.12	27.04
Douglas	DC-9-15	D91	27.25	31.82
Bombardier-de Havilland	Dash-8-300	DH3	27.4	25.7
SIZE 3				
Fokker	F100	100	28.08	35.53
Fokker	F70	F70	28.08	30.91
Bombardier-de Havilland	Dash-8-400	DH4	28.12	31.6
Boeing	B737-100	731	28.35	28.65
Boeing	B737-200	732	28.35	30.53
Bombardier-de Havilland	Dash-7	DH7	28.35	24.58

Douglas	DC-9-21	D92	28.44	31.82
Douglas	DC-9-32	D93	28.44	36.36
Douglas	DC-9-41	D94	28.44	38.28
Boeing	B717-200	717	28.45	37.81
Douglas	DC-9-51	D95	28.45	40.72
Grumman	Gulfstream 5	GRJ	28.5	29.39
Canadair	Global Express	CCX	28.65	30.3
Embraer	ERJ190	E90	28.72	36.24
Embraer	ERJ195	E95	28.72	38.65
Dornier	Do928	n/a	28.81	30.96
Boeing	B737-300	733	28.88	33.4
Boeing	B737-400	734	28.89	36.4
Boeing	B737-500	735	28.89	31.01
Fokker	F27	F27	29	25.059
Fokker	F50	F50	29	25.25
SIZE 4				
Antonov	An-26	A26	29.2	23.8
British Aerospace	ATP	ATP	30.632	26.009
Boeing	B737-300C	73C	31.2	33.4
McDonnel Douglas	MD-81	M81	32.85	45.02
McDonnel Douglas	MD-82	M82	32.85	45.02
McDonnel Douglas	MD-83	M83	32.85	45.02
McDonnel Douglas	MD-87	M87	32.85	39.75
McDonnel Douglas	MD-88	M88	32.85	45.02
McDonnel Douglas	MD-90	M90	32.87	46.51
Boeing	B727-100	721	32.92	40.59
Boeing	B727-200	722	32.92	46.68
Airbus Industrie	A320-100	320	33.91	37.57
Airbus Industrie	A320-200	320	33.91	37.57
Airbus Industrie	A321-100	321	33.91	44.5
Airbus Industrie	A318-100	318	34.1	31.45
Airbus Industrie	A319-100	319	34.1	33.84
SIZE 5				
Boeing	B737-600	736	34.32	31.24
Boeing	B737-700	737	34.32	33.63
Boeing	B737-700	73G	34.32	33.63
Boeing	B737-800	738	34.32	39.5
Boeing	B737-900	739	34.32	42.1
SIZE 6				
Boeing	B737-700W	73W	35.8	33.63
Boeing	B737-800H	73H	35.8	39.5
Boeing	B737-900J	73J	35.8	42.1
Boeing	BBJ	73W	35.8	33.63
SIZE 7				
Tupolev	Tu-154	TU5	37.55	47.9
Antonov	An-12	ANF	38	33.1
Boeing	B757-200	752	38.06	47.33
Boeing	B757-300	753	38.06	54.4
Lockheed	C130H	n/a	40.41	29.79
Lockheed	C130H-30	n/a	40.41	34.37

Lockheed	C130J	n/a	40.41	29.79
Lockheed	C130J-30	n/a	40.41	34.37
Lockheed	L100-20	LOH	40.41	32.33
Lockheed	L100-30	LOH	40.41	34.37
Boeing	B757-200W	75W	41	47.33
SIZE 8				
Tupolev	Tu-204	T20	42	46.22
Ilyushin	Il-62	IL6	43.2	53.12
Douglas	DC-8-61	D8M	43.4	57.12
Douglas	DC-8-71	D8M	43.4	57.12
Airbus Industrie	A310-200	312	43.9	45.89
Airbus Industrie	A310-300	313	43.9	45.89
Boeing	707-320C	703	44.42	46.61
Vickers	VC10	n/a	44.55	48.36
Airbus Industrie	A300-B2	AB4	44.83	53.61
Airbus Industrie	A300-B4	AB5	44.83	53.61
Airbus Industrie	A300-C4	AB6	44.83	53.61
Airbus Industrie	A300-600	AB6	44.84	53.85
Douglas	DC-8-63	D8M	45.23	57.12
Douglas	DC-8-73	D8M	45.23	57.12
Lockheed	L1011-1	L11	47.34	54.17
Lockheed	L1011-100	L11	47.34	54.17
Lockheed	L1011-200	L11	47.34	54.17
Douglas	DC-10-10	D11	47.35	55.55
Boeing	B767-200	762	47.57	48.51
Boeing	B767-300	763	47.57	54.94
Ilyushin	Il-86	ILW	48.06	59.54
SIZE 9				
Lockheed	L1011-500	L15	50.09	50.05
Douglas	DC-10-30	D1C	50.39	55.35
Douglas	DC-10-40	D1C	50.39	55.54
Ilyushin	Il-76	IL7	50.5	46.6
Boeing	B767-300W	76W	50.9	54.94
Boeing	787-300	n/a	51.7	56.7
Boeing	C17	n/a	51.76	53.04
Boeing	B767-400	764	51.9	61.4
McDonnel Douglas	MD-11	M11	52	61.4
SIZE 10				
Boeing	B747-100	741	59.64	70.4
Boeing	B747-200	742	59.64	70.4
Boeing	B747-300	743	59.64	70.4
Boeing	B747SP	74L	59.64	56.31
Boeing	787-800	n/a	60.1	56.7
Ilyushin	Il-96-300	IL9	60.105	55.35
Airbus Industrie	A330-200	332	60.3	59.4
Airbus Industrie	A330-300	333	60.3	63.66
Airbus Industrie	A340-200	342	60.3	59.39
Airbus Industrie	A340-300	343	60.3	63.66
Boeing	B777-200	772	60.93	63.73
Boeing	B777-300	773	60.93	73.86

SIZE 11				
Boeing	787-900	n/a	62.1	62.8
Airbus Industrie	A340-500	345	63.45	67.95
Airbus Industrie	A340-600	346	63.45	75.39
Airbus Industrie	A350XWB-800	n/a	64	60.5
Airbus Industrie	A350XWB-900	n/a	64	66.8
Airbus Industrie	A350XWB-1000	n/a	64	73.8
Antonov	An-22	A22	64.4	57.8
Boeing	B777-200LR	777	64.8	63.73
Boeing	B777-300ER	77W	64.8	73.86
Boeing	B747-400	744	64.94	70.67
Lockheed	C5A	n/a	67.9	75.5
Boeing	B747-8	n/a	*68.5*	76.4
Boeing	B747-8F	n/a	*68.5*	76.4
Antonov	An-124	A4F	73.3	69.1
Antonov	An-225	A5F	88.4	84
SIZE 12				
Airbus Industrie	A380-700	n/a	79.75	67.9
Airbus Industrie	A380-800	380	79.75	72.57
Airbus Industrie	A380-900	n/a	79.75	80

Appendix B

Sizes of the Gates

Stand	Max Aircraft Type (Span)	Max Aircraft Size Group	EXCEPTIONS:	
1	BOEING 737-800(W)	6	73J,739,321,721,722,MD88,MD90,MD81,MD82,MD83	
2	AIRBUS 320	4	M81, M82, M83, M87, M88, M90, 321, 721, 722,321,322	
4	BOEING 757-200(W)	7	753	
5	BOEING 737-400	3	D92, D93, 717, E90, E95	
6	BOEING 767-300	8	L10, L11, IL6, ILW CAN ACCEPT SIZE 9 AIRCRAFT 76W	
7	BOEING 737-800(W)	6	CAN ACCEPT 736,737,738,73G,73W,73H,318,319,320	
8	BOEING 757-200(W)	7	ANF, LOH	
9	BOEING 737-800(W)	6	321, 322	
10	BOEING 767-300	8	L10, L11, IL6, ILW CAN ACCEPT SIZE 9 AIRCRAFT 76W	
11	BOEING 737-900J	6	NONE	
12L	BOEING737-900J	6	IF ANY ON 12 THEN NONE ON 12L	
12	AIRBUS 388	12	IF ANY ON 12L OR 12R THEN NONE ON 12	
12R	MCDONNELL DOUGLAS MD11	9	IF ANY ON 12 THEN NONE ON 12R	
15	BOEING 757-200(W)	7	753	
16	FOKKER 50	2	CAN ACCEPT SIZE 3 AIRCRAFT: F50, F27, DH4, DH7	
17	SAAB2000	2	NONE	
18	A321	4	NONE	
21	FOKKER 50	3	D91, D92, D93, D94, E90, E95, 734, 717	
22	BOEING 767-300	8	M81, M82, M83, M84, M87, M88, M90, 154, L10, L11, ILW, IL6, ATP, TU5. CAN ACCEPT SIZE 3 AIRCRAFT: 732, 733, 734, 735,E70,E90,E95 AND SIZE 9 AIRCRAFT 76W,764	
23	BOEING 767-300(W)	8	M81, M82, M83, M87, M88, M90, ATP, TU5, IL6, ILW, 154, IL9, D8M. CAN ACCEPT SIZE 9 A/C L15,D1C,76W. CAN ACCEPT SIZE 3 A/C 732,733,734,735,E70,E90,E95	
24	BOEING 737-900	5	100, F27, F50, F70, DH4, DH7, D95, E90, E95, ATP. CAN ACCEPT SIZE 2 AIRCRAFT: ARJ, AR1, AR7, AR8, 141,142,143,146, RJ70, RJ85, RJ1	
25	BOEING 737-900(W)	6	100, F27, F50, F70, DH4, DH7, D95, E90, E95, ATP. CAN ACCEPT SIZE 2 AIRCRAFT: ARJ, AR1, AR7, AR8, 141,142,143,146, RJ70, RJ85, RJ1	
26	BOEING 777-200	10	M81, M82, M83, M87, M88, M90, ATP, TU5, 154, IL6, ILW,772 773,777, CAN ACCEPT SIZE 3 AIRCRAFT: 732, 733, 734, 735,E70,E90,E95	
27	BOEING 777-300ER	10	M81, M82, M83, M87, M88, M90, ATP, TU5, 154, IL6, ILW, 773. CAN ACCEPT SIZE 3 AIRCRAFT: 732, 733, 734, 735 ,E70,E90,E95 AND SIZE 11 AIRCRAFT 346, 77W, 744.	
28	BOEING 737-900	5	M81, M82, M83, M87, M88, M90, ATP, 154. CAN ACCEPT SIZE 3 AIRCRAFT: 732, 733, 734, 735	
29	AIRBUS A330-300	9	M81, M82, M83, M87, M88, M90, ATP, IL6, ILW, D8M, IL9, 737, 73W, 738, 73H, 739. CAN ACCEPT SIZE 10 332,333,342,343,788. NO SIZE 1,2,3,5,6 AIRCRAFT.	
31	BOEING 747-400	10	M81, M82, M83, M87, M88, M90, ATP, TU5, 154, IL6, ILW, IL9, D8M. NO SIZE 1,2,3,5,6 AIRCRAFT. CAN ACCEPT SIZE 11 AIRCRAFT: 346,77W,744	
32	BOEING 747-400	10	M81, M82, M83, M87, M88, M90, ATP, TU5, 154, IL6, ILW, IL9, D8M. NO SIZE 1,2,3,5,6 AIRCRAFT. CAN ACCEPT SIZE 11 AIRCRAFT: 346,77W,744	
41	BOEING 737-900(W)	6	721,722,MD88,MD90,MD81,MD82,MD83	
42	BOEING 767-300 (W)	8	CAN ACCEPT SIZE 9 AIRCRAFT B76W	
43	BOEING 757-200(W)	7	NONE	
44R	AIRBUS 320	4	321, 322 IF ANY ON 44 THEN NONE ON 44R	
44	BOEING 747-400	10	CAN ACCEPT SIZE 11 AIRCRAFT: 777,744 IF ANY ON 44R OR 44L THEN NONE ON 44	
44L	BOEING 737-900J	6	NONE IF ANY ON 44 THEN NONE ON 44R	
47	FOKKER 50	3	F100, D93, D94, 717, D05, E90, E95, 734 CAN ACCEPT SIZE 4 AIRCRAFT: 73C, 318, 319	
48	BOEING 757-200(W)	7	NONE	
49	BOEING 757-200(W)	7	NONE	
50	BOEING737-800(W)	3	CAN ACCEPT SIZE 4 AIRCRAFT: A318,A310,A320, SIZE 5 AIRCRAFT: B736,B73G,B738, SIZE 6 AIRCRAFT B73W,B73H	
51	FOKKER 50	3	D95	
52	FOKKER 50	3	D95	
53	BOEING 737-900(W)	6	M81,M82,M83,M87,M88,M90	
54	BOEING 757-200(W)	7	M81,M82,M83,M87,M88,M90	
55	BOEING 767-300	8	CAN ACCEPT SIZE 9 AIRCRAFT:76W, IL7, D1C, L15	
56	BOEING 73J	6	NONE	
57	BOEING 73J	6	NONE	
58	BOEING 73J	6	NONE	
60	AIRBUS A330-200	9	CAN ACCEPT SIZE 10 AIRCRAFT: 332, IL9, 787, 74L	
60L	FOKKER 50	3	IF ANY ON 60 THEN NONE ON 60L	
60R	FOKKER 50	3	IF ANY ON 60 THEN NONE ON 60R	
61	BOEING 767-300	8	NONE	
62	BOEING 747-400	10	773. CAN ACCEPT SIZE 11 AIRCRAFT: 744, 777, 345. CAN ACCEPT SIZE 12 AIRCRAFT : A388	
62L	FOKKER 50	3	IF ANY ON 62 THEN NONE ON 62L	
62R	BOEING 737-900	5	IF ANY ON 62 THEN NONE ON 62R	
63	BOEING 747-300	9	CAN ACCEPT SIZE 10 AIRCRAFT 741,742,743,74L. IF ANY ON 63L, 63R THEN NONE ON 63	
63L	FOKKER 50	3	IF ANY ON 63 THEN NONE ON 63L	
63R	FOKKER 50	3	IF ANY ON 63 THEN NONE ON 63R	
64	BOEING 777-200(LR)	10	773. CAN ACCEPT SIZE 11 AIRCRAFT: 777.IF ANY ON 64L,64R THEN NONE ON 64	
64L	FOKKER 50	3	IF ANY ON 64 THEN NONE ON 64L	
64R	BOEING 737-900	5	IF ANY ON 64 THEN NONE ON 64R	
65	MCDONNELL DOUGLAS MD11	9	IF ANY ON 65L, 65R THEN NONE ON 65	
65L	FOKKER 50	3	IF ANY ON 65 THEN NONE ON 65L	
65R	FOKKER 50	3	IF ANY ON 65 THEN NONE ON 65R	
66	BOEING 777-200(LR)	9	CAN ACCEPT SIZE 10 AIRCRAFT: 332,333,772,342,343,IL9,788,74L. CAN ACCEPT SIZE 11 AIRCRAFT 777 IF ANY ON 66L,66R THEN NONE ON 66	
66L	BOEING 737-900	5	IF ANY ON 66 THEN NONE ON 66L	
66R	BOEING 757-300	7	C130,LOH 75W IF ANY ON 66 THEN NONE ON 66R	
67	BOEING 777-200(LR)	9	CAN ACCEPT SIZE 10 AIRCRAFT: 332,333,772,342,343,IL9,788,74L. CAN ACCEPT SIZE 11 AIRCRAFT 777 IF ANY ON 67L,67R THEN NONE ON 67	
67L	BOEING 737-900	5	IF ANY ON 67 THEN NONE ON 67L	

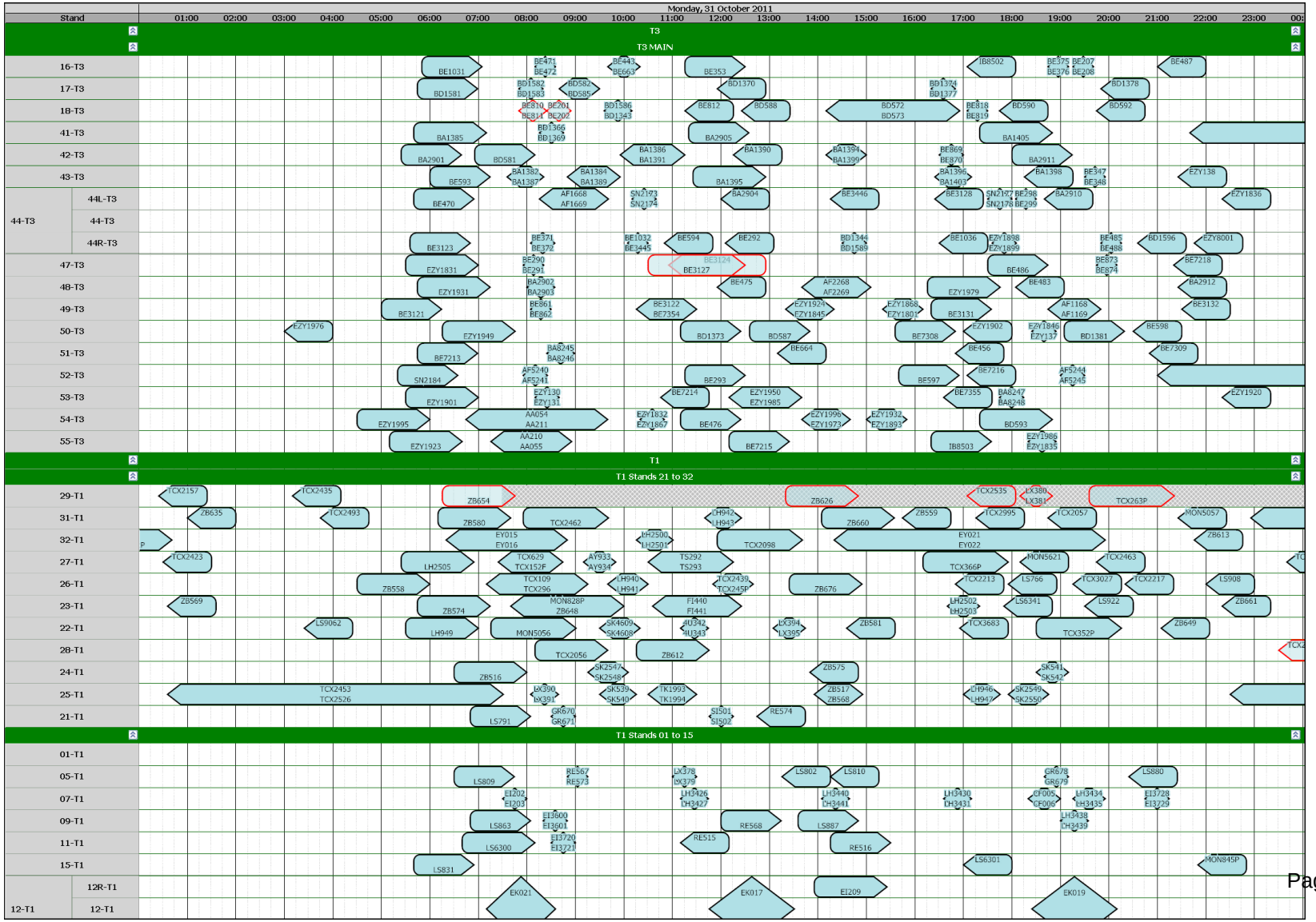
67R	BOEING 757-300	**7**	C130,LOH 75W IF ANY ON 67 THEN NONE ON 67R	
68	BOEING 777-200(LR)	**9**	CAN ACCEPT SIZE 10 AIRCRAFT: 332,333,772,342,343,IL9,788,74L. CAN ACCEPT SIZE 11 AIRCRAFT 777 IF ANY ON 68L,68R THEN NONE ON 68	
68L	BOEING 737-900	**5**	IF ANY ON 68 THEN NONE ON 68L	
68R	BOEING 757-300	**7**	C130,LOH 75W IF ANY ON 68 THEN NONE ON 68R	
69	BOEING 777-200(LR)	**9**	CAN ACCEPT SIZE 10 AIRCRAFT: 332,333,772,342,343,IL9,788,74L CAN ACCEPT SIZE 11 AIRCRAFT 777 IF ANY ON 69L,69R THEN NONE ON 69	
69L	BOEING 737-900	**5**	IF ANY ON 69 THEN NONE ON 69L	
69R	BOEING 757-300	**7**	C130,LOH 75W IF ANY ON 69 THEN NONE ON 69R	
70	BOEING 777-200(LR)	**9**	CAN ACCEPT SIZE 10 AIRCRAFT: 332,333,772,342,343,IL9,788,74L. CAN ACCEPT SIZE 11 AIRCRAFT 777 IF ANY ON 70L,70R THEN NONE ON 70	
70L	BOEING 737-900	**5**	IF ANY ON 70 THEN NONE ON 70L	
70R	BOEING 757-300	**7**	C130,LOH 75W IF ANY ON 70 THEN NONE ON 70R	
71	AIRBUS A330-200	**9**	CAN ACCEPT SIZE 10 AIRCRAFT: 332,342,IL9,788.74L IF ANY ON 71L,71R THEN NONE ON 71	
71L	BOEING 737-900	**5**	IF ANY ON 71 THEN NONE ON 71L	
71R	BOEING 757-200	**6**	CAN ACCEPT SIZE 7 AIRCRAFT:752 IF ANY ON 71 THEN NONE ON 71R	
72	BOEING 747-400	**10**	773. CAN ACCEPT SIZE 11 AIRCRAFT: 744, 777, 345 IF ANY ON 72L,72R THEN NONE ON 72	
72L	BOEING 737-900	**5**	IF ANY ON 72 THEN NONE ON 72L	
72R	BOEING 737-900	**5**	IF ANY ON 72 THEN NONE ON 72R	
73	BOEING 747-400	**11**	C5A, 124, 387, 388, 389, A5F IF ANY ON 73L,73R THEN NONE ON 73	
73L	BOEING 737-900	**5**	IF ANY ON 73 THEN NONE ON 73L	
73R	BOEING 737-900	**5**	IF ANY ON 73 THEN NONE ON 73R	
74	BOEING 747-400	**10**	773. CAN ACCEPT SIZE 11 AIRCRAFT: 74Y, 777, 345 IF ANY ON 74L,74R THEN NONE ON 74	
74L	BOEING 737-900	**5**	IF ANY ON 74 THEN NONE ON 74L	
74R	BOEING 737-900	**5**	IF ANY ON 74 THEN NONE ON 74R	
80	ANTONOV AN-225	**11**	NONE	
81	AIRBUS A330-200	**9**	M11,764 CAN ACCEPT SIZE 10 AIRCRAFT: 332,342,IL9,788.74L	
82	AIRBUS A330-200	**9**	M11,764 CAN ACCEPT SIZE 10 AIRCRAFT: 332,342,IL9,788.74L IF ANY ON 82L,82R THEN NONE ON 82	
82L	BOEING 737-900	**5**	IF ANY ON 82 THEN NONE ON 82L	
82R	BOEING 757-300	**7**	C130,LOH 75W IF ANY ON 82 THEN NONE ON 82R	
83	AIRBUS A330-200	**9**	M11,764 CAN ACCEPT SIZE 10 AIRCRAFT: 332,342,IL9,788.74L IF ANY ON 83L,83R THEN NONE ON 83	
83L	BOEING 737-900	**5**	IF ANY ON 83 THEN NONE ON 83L	
83R	BOEING 757-300	**7**	C130,LOH 75W IF ANY ON 83 THEN NONE ON 83R	
84	AIRBUS A330-200	**9**	M11,764 CAN ACCEPT SIZE 10 AIRCRAFT: 332,342,IL9,788.74L IF ANY ON 84L,84R THEN NONE ON 84	
84L	FOKKER 50	**3**	IF ANY ON 84 THEN NONE ON 84L	
84R	BOEING 757-300	**7**	C130,LOH, 75W IF ANY ON 84 THEN NONE ON 84R	
85	BOEING 747-400	**11**	C5A, 124, 387, 388, 389, A5F IF ANY ON 85L,85R THEN NONE ON 85	
85L	BOEING 737-900	**5**	IF ANY ON 85 THEN NONE ON 85L	
85R	BOEING 737-900	**5**	IF ANY ON 85 THEN NONE ON 85R	
86	BOEING 747-400	**11**	C5A, 124, 387, 388, 389, A5F IF ANY ON 86L,86R THEN NONE ON 86	
86L	BOEING 737-900	**5**	IF ANY ON 86 THEN NONE ON 86L	
86R	BOEING 757-200(W)	**7**	IF ANY ON 86 THEN NONE ON 86R	
100	FOKKER 50	**2**	CR9. CAN ACCEPT SIZE 3 AIRCRAFT: 732, 733, 735, F50, DH4	
101	FOKKER 50	**2**	CR9. CAN ACCEPT SIZE 3 AIRCRAFT: 732, 733, 735, F50, DH4	
216	BOEING 767-300	**8**	CAN ACCEPT SIZE 9 A/C 76W .IF ANY ON 216R THEN NONE ON 216	
216R	BOEING 757-200(W)	**7**	IF ANY ON 216 THEN NONE ON 216R	
217	BOEING 767-300	**8**	CAN ACCEPT SIAZE 9 A/C 76W. IF ANY ON 217L THEN NONE ON 217	
217L	BOEING 757-200(W)	**7**	IF ANY ON 217 THEN NONE ON 217L	
218	MCDONNELL DOUGLAS MD11	**9**	764,76W IF ANY ON 218R THEN NONE ON 218	
218R	BOEING 757-200(W)	**7**	IF ANY ON 218 THEN NONE ON 218R	
219	BOEING 747-400	**10**	773, 772. CAN ACCEPT SIZE 11 AIRCRAFT: 744 IF ANY ON 219L THEN NONE ON 219	
219L	BOEING 777-200	**9**	764. CAN ACCEPT SIZE 10 AIRCRAFT: 772,777 IF ANY ON 219 THEN NONE ON 219L	
231	BOEING 747-400	**10**	773. CAN ACCEPT SIZE 11 AIRCRAFT: 744	
233	BOEING 767-300	**8**	NONE	
235	BOEING 757-200(W)	**7**	NONE	
237	BOEING 767-300	**8**	NONE	
239	BOEING 767-300	**8**	CAN ACCEPT 76W	
241	BOEING 757-200(W)	**7**	NONE	
243	BOEING 737-900	**5**	NONE	
245	BOEING 767-300	**8**	CAN ACCEPT 76W	
247	BOEING 767-300	**8**	NONE	
249	BOEING 767-300	**8**	CAN ACCEPT 76W	
249L	BOEING 757-200(W)	**7**	IF ANY ON 249 THEN NONE ON 249L	
201	BOEING 737-900J	**6**	IF ANY ON 202 THEN NONE ON 201	
202L	BOEING 757-200(W)	**7**	IF ANY ON 202 THEN NONE ON 202L	
202	BOEING 777-200LR	**9**	CAN ACCEPT SIZE 10 AIRCRAFT B788,A332,A333.A342.A343.B772, SIZE 11 AIRCRAFT: 77L, IF ANY ON 202L, OR 201 THEN NONE ON 202	
203	BOEING 757-200(W)	**7**	IF ANY ON 204 THEN NONE ON 203	
204L	BOEING 757-200(W)	**7**	IF ANY ON 204 THEN NONE ON 204L	
204	BOEING 747-400	**10**	IF ANY ON 204L, OR 203 THEN NONE ON 204	
205	FOKKER 50	**3**	IF ANY ON 206 THEN NONE ON 205	
206L	BOEING 757-200(W)	**7**	CAN ACCEPT SIZE 8 AIRCRAFT: 310, 312, 313, AB3, AB4, AB5, AB6 IF ANY ON 206 THEN NONE ON 206L	
206	BOEING 777-300	**10**	CAN ACCEPT SIZE 11 AIRCRAFT: 777, 77W, 744, 345 IF ANY ON 206L 0R 205 THEN NONE ON 206	

207	BOEING 737-900J	**6**	IF ANY ON 208 THEN NONE ON 207	
208L	BOEING 757-200(W)	**7**	IF ANY ON 208 THEN NONE ON 208L	
208	BOEING 747-400	**10**	773 CAN ACCEPT SIZE 11 AIRCRAFT: 777, 744, 345 IF ANY ON 208L 0R 207 THEN NONE ON 208	
209	BOEING 737-900J	**6**	IF ANY ON 210 THEN NONE ON 209	
210L	BOEING 757-200(W)	**7**	IF ANY ON 210 THEN NONE ON 210L	
210	BOEING 747-400	**10**	773 CAN ACCEPT SIZE 11 AIRCRAFT: 777, 744, 345 IF ANY ON 210L 0R 209 THEN NONE ON 210	
211	BOEING 737-900J	**6**	IF ANY ON 212 THEN NONE ON 211	
212L	BOEING 757-200(W)	**7**	CAN ACCEPT SIZE 8 AIRCRAFT: 310, 312, 313, AB3, AB4, AB5, AB6 IF ANY ON 212 THEN NONE ON 212L	
212	BOEING 747-400	**10**	773 CAN ACCEPT SIZE 11 AIRCRAFT: 777, 744, 345 IF ANY ON 212L 0R 211 THEN NONE ON 212	
213	BOEING 737-900J	**6**	IF ANY ON 214 THEN NONE ON 213	
214	BOEING 767-300	**8**	CAN ACCEPT SIZE 9 AIRCRAFT: 76W, 764 IF ANY ON 214L OR 213 THEN NONE ON 214	
214L	BOEING 737-900J	**6**	IF ANY ON 214 THEN NONE ON 214L	
215	BOEING 767-300	**8**	CAN ACCEPT SIZE 9 AIRCRAFT: 76W, 764	

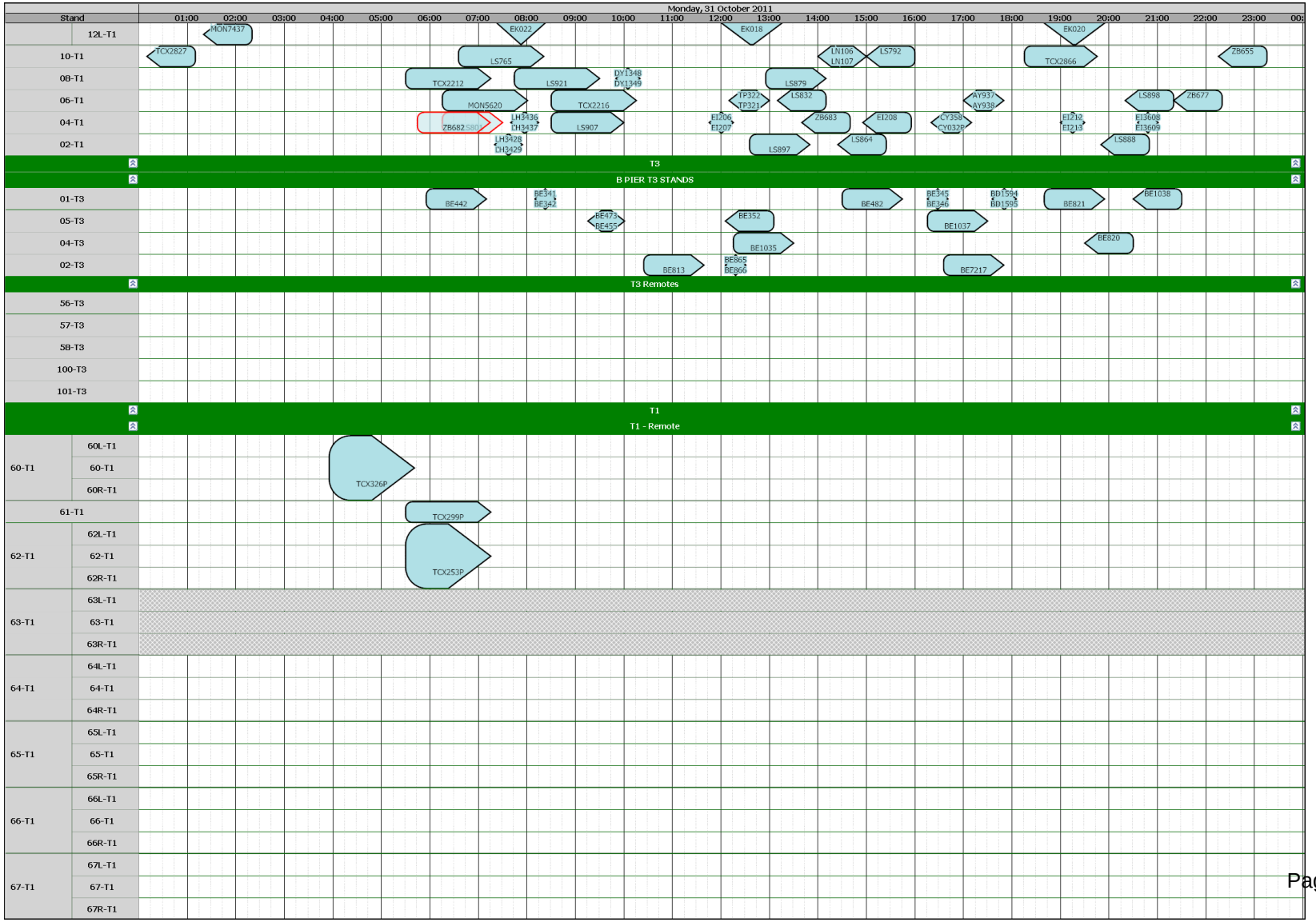
Appendix C

Gate Assignment Planned a Week in Advance

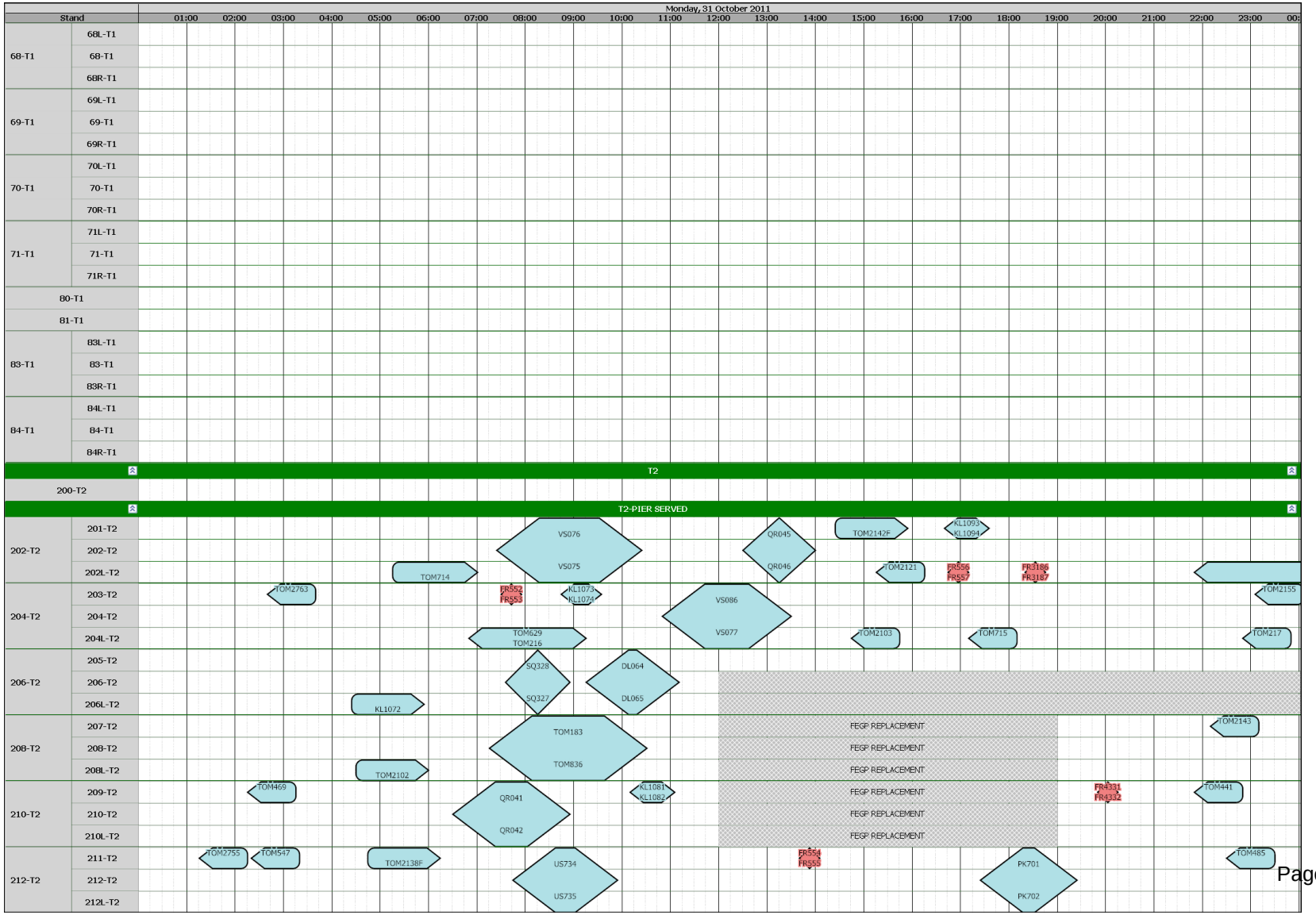
Chroma Assign 31 OCTOBER 31/10/2011 00:00 - 01/11/2011 00:00



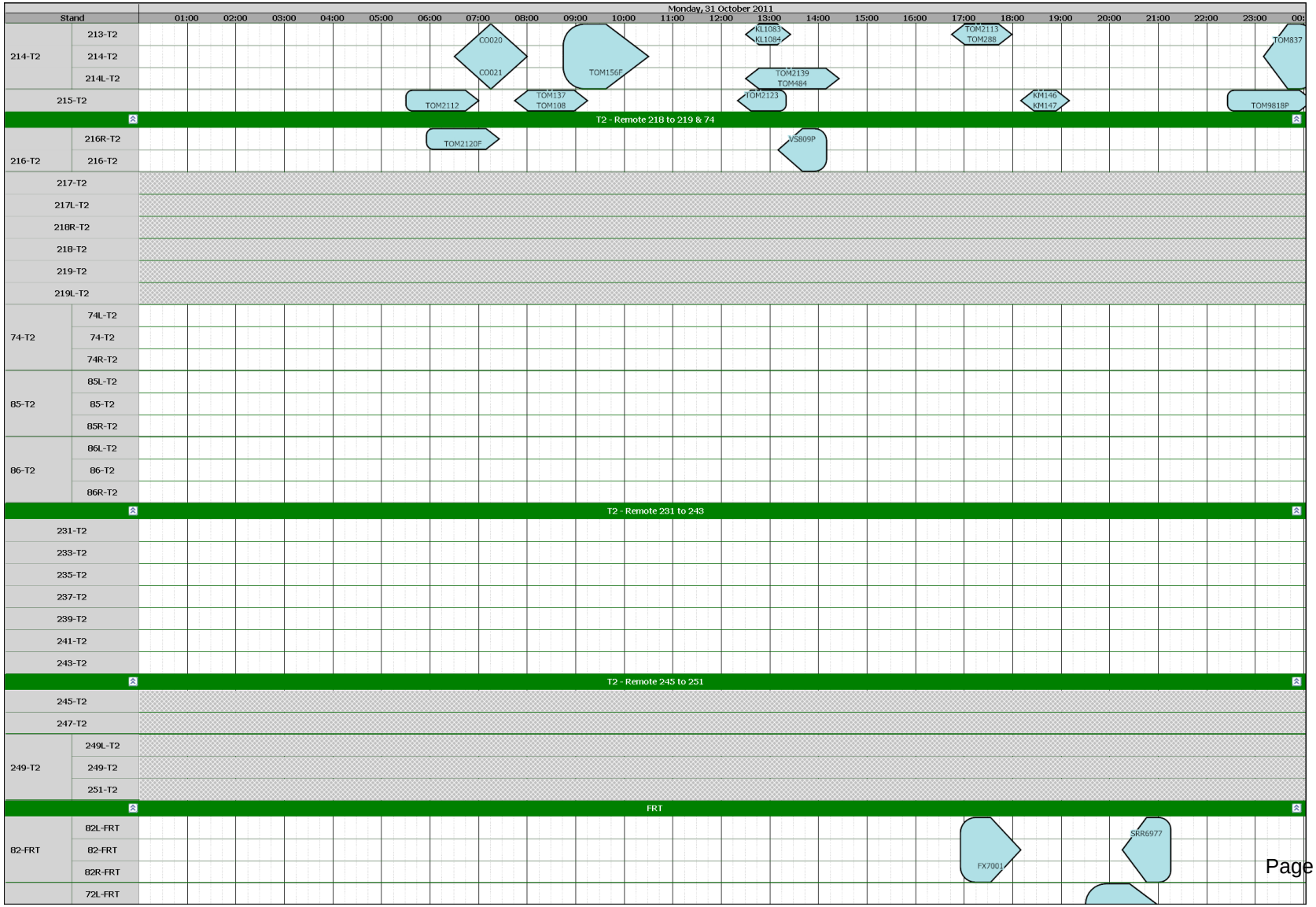
Chroma Assign 31 OCTOBER 31/10/2011 00:00 - 01/11/2011 00:00

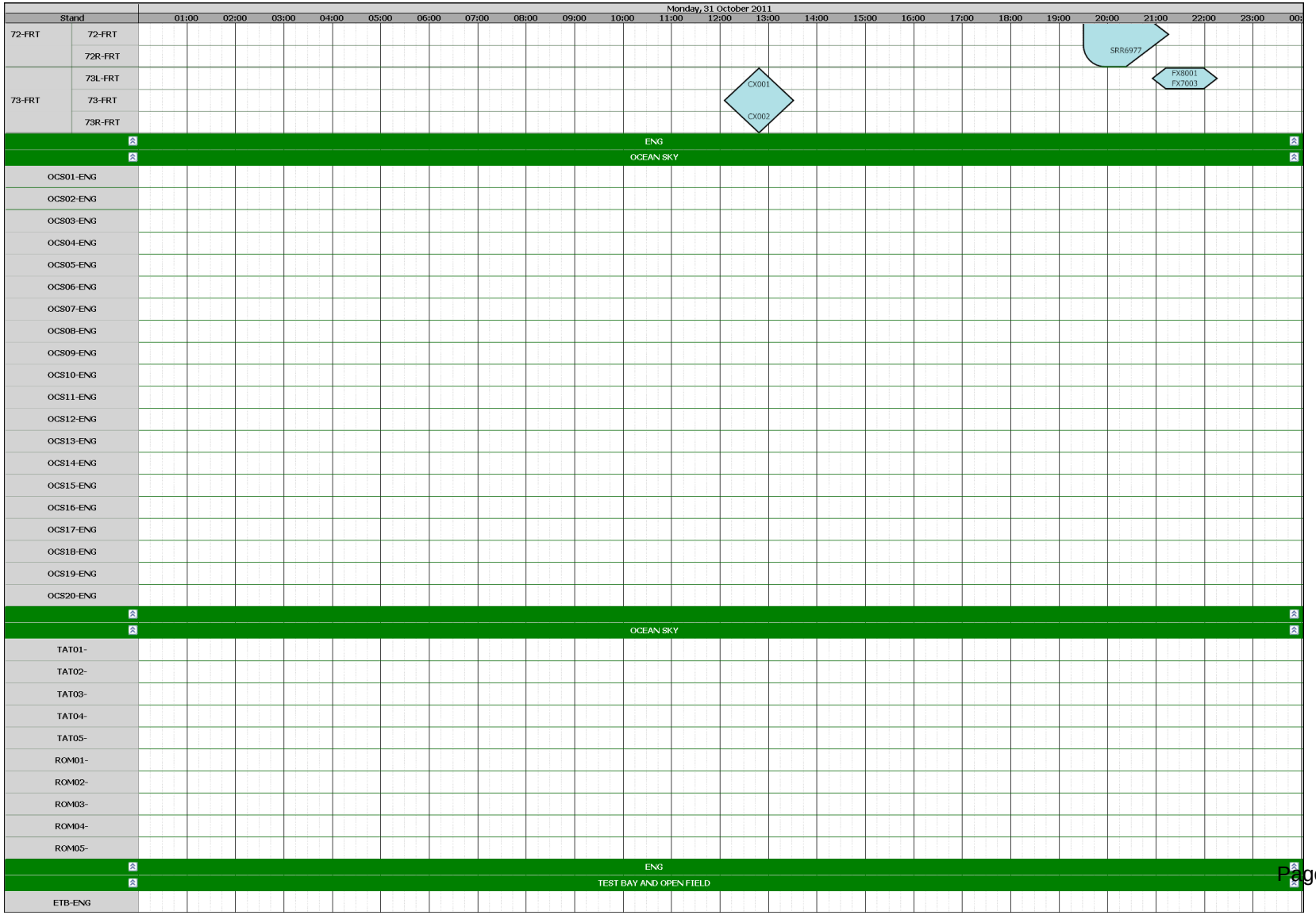


Chroma Assign 31 OCTOBER 31/10/2011 00:00 - 01/11/2011 00:00



Chroma Assign 31 OCTOBER 31/10/2011 00:00 - 01/11/2011 00:00




Monday, 31 October 2011
Stand
01:00 02:00 03:00 04:00 05:00 06:00 07:00 08:00 09:00 10:00 11:00 12:00 13:00 14:00 15:00 16:00 17:00 18:00 19:00 20:00 21:00 22:00 23:00
72-FRT
72-FRT
72R-FRT
73-FRT
73L-FRT
73-FRT
73R-FRT
SRR6977
FX8001
FX7003
CX001
CX002
ENG
OCEAN SKY
OCS01-ENG
OCS02-ENG
OCS03-ENG
OCS04-ENG
OCS05-ENG
OCS06-ENG
OCS07-ENG
OCS08-ENG
OCS09-ENG
OCS10-ENG
OCS11-ENG
OCS12-ENG
OCS13-ENG
OCS14-ENG
OCS15-ENG
OCS16-ENG
OCS17-ENG
OCS18-ENG
OCS19-ENG
OCS20-ENG
OCEAN SKY
TAT01-
TAT02-
TAT03-
TAT04-
TAT05-
ROM01-
ROM02-
ROM03-
ROM04-
ROM05-
ENG
TEST BAY AND OPEN FIELD
ETB-ENG