

Improving the Realism of Ground Movement Models



Christofas Stergianos
MSc Computer Science and Entrepreneurship with Distinction
School of Computer Science
University of Nottingham

Thesis submitted to the University of Nottingham for the degree of
Doctor of Philosophy (PhD)

2017

Abstract

As air traffic increases, more airports are facing capacity problems. A growing number of airports are considering optimisation methods as a solution for increasing their capacity and improving their efficiency. However, modelling an airport is complicated as there are many processes that happen in parallel, each with different constraints and objectives that need to be considered. This thesis focuses on the ground movement problem, the problem of moving aircraft efficiently around an airport. This problem links the problems at the stands and at the runways, and a good model for this problem can not only help the controllers who direct the aircraft to do so more effectively, but can also feed into taxi time estimation improvements, which can aid the solution of other optimisation problems, such as take-off sequencing. Firstly, the effects of the pushback process are investigated and a model that includes this process is presented. Secondly, the effects of different levels of prioritisation between arrivals and departures is investigated. Thirdly, the effects that the airport layout has on the routing process is investigated by examining various airport morphologies and by identifying areas that can cause delays. Different airport morphologies are compared, and the use and importance of alternative paths is highlighted. Moreover, the gate allocation process that also affects the ground movement problem is considered, and a model that uses the routing process of aircraft as a tool to provide a more informed and tailored allocation of aircraft to the gates is presented. A 52% decrease in the duration of delays was observed during the routing process of aircraft when the two processes were integrated. Furthermore, a new routing algorithm that solves the routing problem faster than what is currently used in academia for routing aircraft by taking into consideration all the available paths is presented. The results show a 46% to 67% (depending on the airport) improvement on execution time. Finally, the model is applied in a flight simulator cockpit - a tool for assisting the air transportation operations - and it was integrated with other novel technologies in other research fields. This research provides a more realistic and faster way to solve the ground movement problem of aircraft.

Acknowledgements

First of all, I would like to thank Dr. Jason Atkin for his guidance and his advice. I have been extremely fortunate to have him as my supervisor and I am deeply grateful for his ongoing support and help. He has not only been a great supervisor, but a fantastic person to work with.

I would also like to thank Dr. Patrick Schittekat and Dr. Tomas Nordlander for the support and guidance regarding my research, but also for helping me integrate in the company and socialise during my placement in Sintef.

Moreover, I would like to thank Prof. Herve Morvan for setting such a high standard for the INNOVATE project and for making sure that we get the best training, experience and opportunities throughout the duration of the project.

I would also like to thank my parents for believing in me and for supporting me through the entirety of my studies.

Finally, I am truly grateful to my partner Kyriaki for her continued support and encouragement. She has been always eager to help me in any way possible and I am convinced that I wouldn't be where I am today, if it wasn't for her.

The research leading to these results has received funding from the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme (FP7/2007-2013) under REA grant agreement no 608322.

Contents

List of Figures.....	ix
List of Tables.....	xi
1 Introduction	1
1.1 Background and Motivation.....	1
1.2 Aims and Scopes	2
1.3 Contributions of this Thesis	2
1.4 Publications and Talks	4
1.5 Non-disclosure Agreement.....	6
1.6 Collaborations with Sintef.....	6
1.7 INNOVATE and the Virtual and Physical Demonstrator	7
1.8 Structure of the Thesis.....	8
2 Background and Related Work.....	9
2.1 Introduction	9
2.1.1 Basic terms.....	9
2.2 The Ground Movement Problem.....	10
2.2.1 Constraints	10
2.2.2 Objectives	14
2.3 Ground Movement Problem Solutions.....	16
2.3.1 Genetic Algorithms.....	16
2.3.2 Mixed Integer Linear Programming	18
2.3.3 Other Solution Methods.....	21
2.3.4 Comparison of the approaches.....	23
2.4 Pushback Process	23
2.5 The Gate Assignment Problem	24
2.5.1 Constraints	24
2.5.2 Objectives	26
2.5.3 Gate Assignment Problem Solutions	26
2.5.4 Integration with the Ground Movement Problem.....	27
2.6 Runway Scheduling.....	27
2.6.1 Constraints	28
2.6.2 Objectives	29
2.6.3 Previous work on runway scheduling.....	29
2.6.4 Integration with Ground Movement	30
2.7 Taxi time prediction	32
2.8 Conclusions	34

3	The QPPTW algorithm, Datasets and Airports	35
3.1	Introduction	35
3.2	The QPPTW Algorithm	36
3.2.1	Introduction of the QPPTW Algorithm and Benefits of Using this Methodology	36
3.2.2	Notation and Definitions.....	37
3.2.3	Definitions of Key Concepts.....	38
3.2.4	Explaining the QPPTW Algorithm.....	39
3.3	Airports and Graphs	42
3.3.1	Zurich Airport.....	42
3.3.2	Stockholm Arlanda Airport	44
3.3.3	Manchester Airport.....	46
3.4	Conclusions	48
4	Pushback Delays on the Airport Ground Movement Problem	49
4.1	Introduction	49
4.2	The Pushback Process	50
4.2.1	Previous Research on Pushback Process	51
4.3	Integrating the Pushback Process to the Routing Process	52
4.3.1	Notation and Definitions.....	52
4.3.2	The QPPTW Algorithm with Pushback Process	53
4.3.3	Developed QPPTW Models.....	56
4.3.4	Calculating the Delays	57
4.3.5	Finding the Minimum Taxi Time	57
4.4	MILP Routing Model with the Pushback Process.....	58
4.4.1	Notation and Definitions.....	58
4.4.2	Developed Model.....	59
4.4.3	Constraints	60
4.4.4	Objective Function.....	61
4.5	Comparison and Insights	61
4.5.1	Experimental Set-up.....	62
4.5.2	Comparison Results	63
4.6	Comparison of the two Methodologies	68
4.6.1	Comparison of two Methodologies when Arrivals are Included	69
4.7	Conclusions	69
5	Pushback Process with Stand Holding and the Effects of Prioritisation Levels for Arrivals or Departures	71
5.1	Introduction	71
5.2	Integrating the Stand Holding Process with the Pushback Process.....	73

5.2.1	Notation and Definitions.....	73
5.2.2	The QPPTW Algorithm with Pushback Process and Stand Holding.....	74
5.3	The Effects of the Pushback Process with Stand Holding	76
5.4	Priority Between Arrivals and Departures	79
5.4.1	Importance of the Consideration Order	80
5.4.2	Trade-off Results After Prioritising Departures	81
5.5	Mixed Prioritisation.....	82
5.5.1	Mixed Prioritisation Results	82
5.5.2	Reasons that Delays Happen.....	84
5.5.3	Further Investigation of Varying the Prioritisation of Departures against Arrivals	86
5.6	Conclusions	88
6	The Effects of Airport Layout and Re-routing on Taxiing.....	89
6.1	Introduction	89
6.2	Problem Description.....	90
6.3	Airport Layouts	92
6.3.1	Airports - Similarities and Differences	92
6.3.2	New Airport Layouts	94
6.3.3	Using the Same Data in Different Layouts	97
6.3.4	Investigating the Chosen Path.....	98
6.4	Results	99
6.4.1	Explaining why Delays Happen.....	100
6.4.2	Comparing Zurich and Arlanda Airports	103
6.4.3	Further Investigation of Different Airport Layouts	106
6.5	Conclusions	114
7	Considering the Gate Allocation Process	116
7.1	Introduction	116
7.2	Previous Work on Integration with the Ground Movement Problem	117
7.3	The Gate Allocation Problem.....	119
7.3.1	Definitions of the Variables for the Gate Allocation Model.....	119
7.3.2	Constraints	120
7.3.3	Objective Function.....	122
7.3.4	Effects on the Ground Movement Process.....	123
7.4	Solving the Ground Movement Problem and Finding Conflicting Aircraft	124
7.4.1	Definitions of the Variables	124
7.4.2	Finding the Conflicting Combinations of Aircraft.....	125
7.4.3	The Algorithm for Finding the Conflicting Combinations of Aircraft	128
7.5	Implementation Issues.....	130

7.5.1	Adjusting the QPPTW Algorithm for Solving Side Problems and Running Combinations of Aircraft for Potential Conflicts.....	130
7.5.2	Testing Combinations of Aircraft for Potential Conflicts.....	131
7.6	The Integration Framework.....	132
7.6.1	Definitions of the Variables	132
7.6.2	The Feedback Loop.....	133
7.6.3	Adding the Ground Movement Feedback to the Gate Allocation Model	135
7.6.4	The Stopping Condition.....	136
7.7	Executing the Integrated Model	136
7.7.1	Experimental Settings	137
7.7.2	Results.....	137
7.8	Buffer Time Between Aircraft	141
7.8.1	Implementing the Buffer Time Between Aircraft.....	141
7.8.2	The Impact of Adding a Buffer Time Between Aircraft.....	142
7.9	Conclusions	145
8	An A* Approach for the Quickest Path Problem with Time Windows	148
8.1	Introduction	148
8.2	The A* Approach and Overview of Implementations	149
8.2.1	Previous Work	149
8.2.2	A Heuristic Estimation of the Cost	150
8.2.3	Definitions of the Variables	152
8.2.4	The ASQPPTW Algorithm	153
8.2.5	The Differences Between the Departing and Arriving Process	155
8.3	Experimental Settings	159
8.4	Execution Times for Each Algorithm	159
8.4.1	Solving the Full Problem	160
8.4.2	Solving Arrivals and Departures Separately	162
8.4.3	Solving the Departures without the Heuristic for the Pushback Process	164
8.5	Investigating the Expansion of the two Algorithms	165
8.5.1	The Number of Labels that are Generated with Each Algorithm	166
8.5.2	Tests on a Different Airport Layout.....	170
8.5.3	Prioritising Arrivals	173
8.5.4	Traffic and ASQPPTW Performance.....	176
8.5.5	Label Generation and Airport Characteristics	177
8.5.6	Examination of the Heuristic	179
8.6	Conclusions	181
9	Integration of the Ground Movement Problem in a Flight Simulator Cockpit.....	183

9.1	Introduction	183
9.2	Advanced Receiver Autonomous Integrity Monitoring	184
9.3	Solving the Ground Movement Problem and Applying the Solution	184
9.3.1	Using the GPS Locations of the Aircraft to Find their Position on the Graph	185
9.3.2	Real Time Navigation of an Aircraft on the Ground	187
9.4	Physiological Monitoring of Human Performance	187
9.5	The Physical Demonstrator	189
9.6	Conclusions	191
10	Conclusions	192
10.1	General Summary	192
10.2	Key Results	193
10.3	Future Work	195
	References	197

List of Figures

3.1: Model of Zurich airport as a graph with edges and nodes	43
3.2: Traffic for different hours during the day in Zurich airport.....	44
3.3: Model of Arlanda airport as a graph with edges and nodes	45
3.4: Model of Manchester airport as a graph with edges and nodes	47
3.5: Traffic for different hours during the day in Manchester airport.....	48
4.1: Causes of pushback delays, delaying other aircraft or the aircraft pushing back	50
4.2: Blocked edges during pushback.....	56
4.3: Total delay for each algorithm/model.....	64
5.1: Delay graph that shows number of aircraft that are delayed by “x” or more for day 1	79
5.2: Delay graph for arrivals/departures for different prioritisations, showing the delay (in seconds) against the number of arrivals (A) and departures (D) aircraft which have that delay or higher, for three different configurations.	84
5.3: Example case where departing aircraft 466 has to push back early to avoid traffic	85
6.1: Graph of Arlanda airport.....	93
6.2: Graph of Zurich airport.....	94
6.3: Graph of Arlanda airport.....	95
6.4: Graph of Zurich airport.....	96
6.5: A departing aircraft (red) delaying an arriving aircraft (blue)	100
6.6: Two departing aircraft heading towards different runways	101
6.7: Departing aircraft need to start their pushback process earlier to avoid a conflict	102
6.8: Number of aircraft for each time duration of delay in each airport	107
6.9: Distribution of taxi duration for each airport.	108
6.10: Delay graph for different airport layouts, showing the delay (in minutes) against the number of aircraft which have that delay or higher	109
6.11: An example in Zurich airport where sufficient number of taxiways ensure that there is no delay	110
6.12: An example in Zurich airport where removing taxiways introduces a delay.....	111
6.13: Example of aircraft that needs to take a longer path due to traffic (instance 1).....	112
6.14: Example of aircraft that needs to take a longer path due to traffic (instance 2).....	113
6.15: Example of aircraft taking a shorter path when a new taxiway is introduced	113
7.1: Example of three aircraft interacting with each other	127
7.2: Flow diagram of the integrated model	134
7.3: The duration of delays during the ground movement process and the increase of the objective value of the gate allocation model for each iteration, for terminal 1, day 2	140
7.4: The duration of delays during the ground movement process and the increase of the objective value of the gate allocation model for each iteration, for terminal 1, day 2 when buffer time between aircraft is included	145
8.1: Distribution of labels generated by each algorithm (left) and cumulative graph that shows the number of aircraft that generate “x” labels or more (right) for Zurich airport.....	167

8.2: Distribution of labels generated by each algorithm (left) and cumulative graph that shows the number of aircraft that generate “x” labels or more (right) for Zurich airport (arrivals and departures).....	168
8.3: Distribution of labels generated by each algorithm (left) and cumulative graph that shows the number of aircraft that generate “x” labels or more (right) for Zurich airport (arrivals and departures separate).....	169
8.4: Distribution of labels generated by each algorithm (left) and cumulative graph that shows the number of aircraft that generate “x” labels or more (right) for Arlanda airport.....	171
8.5: Distribution of labels generated by each algorithm (left) and cumulative graph that shows the number of aircraft that generate “x” labels or more (right) (arrivals and departures - Arlanda airport).....	172
8.6: Distribution of labels generated by each algorithm (left) and cumulative graph that shows the number of aircraft that generate “x” labels or more (right) (arrivals and departures separate - Zurich airport).....	173
8.7: Distribution of labels generated by each algorithm (left) and cumulative graph that shows the number of aircraft that generate “x” labels or more (right) when arrivals are prioritised...	174
8.8: Distribution of labels generated by each algorithm (left) and cumulative graph that shows the number of aircraft that generate “x” labels or more (right) when arrivals are prioritised...	175
8.9: Distribution of labels generated by each algorithm (left) and cumulative graph that shows the number of aircraft that generate “x” labels or more (right) with less traffic	176
8.10: The improvement in the number of labels generated in relation to the time duration travelled.....	178
8.11: The improvement in the number of labels generated in relation to the distance travelled	179
9.1: Finding the closest node.....	186
9.2: Graph representation of Zurich airport (quickest path highlighted)	188
9.3: The flight simulator cockpit.....	189
9.4: Quickest path highlighted after taking into consideration the positions of other aircraft ..	190

List of Tables

3.1: Table of definitions for the QPPTW algorithm.....	37
3.2: Specifications of instances	46
4.1: Table of definitions	52
4.2: Table of definitions for the MILP model	58
4.3: Total delays and total taxi time for each algorithm/model for instances 1 and 2.	63
4.4: Total delays and total taxi time for each model/algorithm for running instances 3, 4 and 5.	63
4.5: Flights which are affected by ground movement delays (Instance 1).....	65
4.6: Results of including the pushback process.	66
4.7: Comparison between the two methodologies	69
5.1: Table of definitions for QPPTW with stand holding	73
5.2: Delays when the stand holding process is used	78
5.3: Different priorities in routing arrivals and departures.....	81
5.4: Delays of aircraft for different prioritisations	83
5.5: Delays for different “mixed prioritisation” settings.....	87
6.1: Groups and types of stands	98
6.2: Total taxi time if there were no delays.....	103
6.3: Duration of delays in seconds	104
6.4: Total taxi time in seconds	104
6.5: The number of delays for each airport.....	105
7.1: Table of definitions for the gate allocation model	119
7.2: Table of definitions for finding the conflicting combinations of aircraft	124
7.3: Table of definitions for the integration process	133
7.4: Results after solving the gate allocation process while considering the ground movement	138
7.5: Results after solving the gate allocation process while considering the ground movement including buffer times between aircraft	143
8.1: Table of definitions	152
8.2: Execution times in milliseconds for routing aircraft (Zurich Airport).....	160
8.3: Execution times in milliseconds for routing aircraft (Arlanda Airport).....	161
8.4: Execution times in milliseconds for routing departing aircraft (Zurich Airport).....	162
8.5: Execution times in milliseconds for routing arriving aircraft (Zurich Airport)	162
8.6: Execution times in milliseconds for routing departing aircraft (Arlanda Airport)	163
8.7: Execution times in milliseconds for routing arriving aircraft (Arlanda Airport)	163
8.8: Execution times in milliseconds for routing departing aircraft when the heuristic for the pushback process is not included (Zurich airport)	164
8.9: Execution times in milliseconds for routing departing aircraft when the heuristic for the pushback process is not included (Arlanda airport)	165
8.10: Number of labels generated with each algorithm (ASQPPTW vs QPPTW)	166

8.11: Number of labels generated in Arlanda airport (ASQPPTW vs QPPTW)	170
8.12: Execution times in milliseconds for routing aircraft when arrivals are prioritised (QPPTW vs ASQPPTW)	174
8.13: Number of labels that are generated with each set-up	180
9.1: Table of Notation and Definitions	185

Introduction

1.1 Background and Motivation

This research is part of a Marie Curie project funded by the European Union called the INtegration of NOVel Aerospace TEchnologies (INNOVATE) and is supported by the Institute of Aerospace Technology of the University of Nottingham. The project builds on previous PhD theses at the school of Computer Science and considers some of the real-world issues when integrating this research into airports.

This thesis considers the problem of more accurately modelling the movement of aircraft around an airport, and using this model in a system to direct aircraft around the airport. This is an increasingly important problem, with airports getting busier over time, and many facing capacity problems. Furthermore, reducing CO₂ emissions is becoming an increasingly important consideration at many airports (SESAR 2015). Even though a considerable amount of optimisation research exists for various airport processes, few airports actually use automated optimisation processes, and those that do often use them only for a few processes. One of the major concerns that has been expressed to us by controllers as a reason for not adopting more automation is that the automated systems often do not have a realistic enough model of the problem, or consider enough of the special cases (i.e. aircraft need to take a longer or uncommon route when there is traffic). This research aims to address this issue by improving the accuracy of the models and using these improved models to investigate the problem in more detail. The finished system has also been embedded into a demonstrator that will be presented in Chapter 9.

In practice, the routing problem is still usually solved manually by air traffic controllers. The ground movement controller is responsible for guiding aircraft around the taxiways, monitoring intersections where aircraft arrive and deciding upon the prioritisation of

movements. At quieter airports, the same person may also determine the take-off sequence, or the pushback times for aircraft, but it is more common for different controllers to handle different problems. Consequently, the different processes are usually handled independently even though they affect one another. However, due to the complexity of the problem, a high level of coordination is needed, and the optimal solution is not always achieved. Optimisation models can provide a better solution for airports as was demonstrated in Kjenstad et al. (2013b) where their model, when tested on a real airport, outperformed expert controllers.

1.2 Aims and Scopes

The main objective of this thesis is to provide a more integrated and realistic model for the ground movement of the aircraft in airports. More specifically, the research in this thesis has the following aims.

- To take into consideration more parameters for the ground movement model and to enable integration with other processes that affect or get affected by the ground movement process as well as producing new algorithms to achieve this.
- To produce a more realistic model that will be able to identify delays that can happen during the ground movement process of aircraft
- To get a better understanding of the morphology of an airport and how it affects the delays that can happen during the taxiing process of aircraft.
- To increase the processing speed of algorithms that are used for solving the routing and scheduling process of aircraft, if possible.
- To integrate this research into a prototype system that also makes use of other novel aerospace technologies.

This research focuses on the ground movement problem, but other processes and technologies outside of this area are also considered where they are relevant to this work or where understanding these is beneficial to understanding the work in this thesis.

1.3 Contributions of this Thesis

The contributions of this thesis can be summarised as follows:

Chapter 4

- The pushback process is explicitly modelled within the routing process for aircraft, making it possible to identify many real-world delays that would otherwise pass unnoticed by the model.
- The effects of considering the pushback process are examined using two different ground movement models. This provides a better understanding of where and when delays during the ground movement process of aircraft actually happen.

Chapter 5

- The prioritisation of arrivals over departures or vice versa to different degrees is examined. The performance (in terms of delays) of different prioritisation setups between arriving and departing aircraft is described. This provides important insight into how these delays happen when the pushback process is explicitly modelled and how they can be reduced.

Chapter 6

- The performance of a routing algorithm that can take into consideration multiple paths is compared between two airports. This provides a better understanding of how the morphology of an airport can affect the effectiveness of a routing methodology.
- The morphology of the airport is examined, focusing on where delays can happen and the use of non-shortest paths.

Chapter 7

- A new algorithm that can find, in a short amount of time, aircraft that can interact with each other, which can be used for providing feedback to the gate allocation process, is presented.
- An integration framework between the ground movement process and the gate allocation process is described that reduces the delays that can happen during the ground movement process of an aircraft by avoiding allocations that can cause delays.

Chapter 8

- An improved algorithm for solving the Quickest Path Problem with Time Windows, that uses heuristics to speed up the routing process of aircraft is presented. This algorithm provides a better tool for real time routing but also provides a fast routing method that is ideal for integrating with other processes that happen in the airport.
- The performance of the new algorithm is presented in various airports and set-ups.

Chapter 9

- A framework where the routing process, the aircraft position process and the mental load of the pilot observation are integrated within a virtual cockpit is described. This provides an important insight into how a routing model can be used in a real-world scenario.
- A methodology of real time routing of aircraft, using the position of the aircraft, is presented that deals with the practical issues of implementing a routing model.

1.4 Publications and Talks

During the research leading to this thesis a number of publications have been produced. The work has also been presented in various conferences and events. The publications and talks are presented below, grouped into the corresponding chapters of this thesis:

Chapter 4

- “Pushback delays on the routing and scheduling problem of aircraft”, presentation at the International Conference on Applied Operational Research (ICAOR 2015), Vienna, Austria (16/07/2015), talk
- Stergianos, C., Atkin, J.A.D., Schittekat, P., Nordlander, T.E., Gerada, C., Morvan, H. (2015). “Pushback delays on the routing and scheduling problem of aircraft”. 7th International Conference on Applied Operational Research, ICAOR. Lecture Notes in Management Science 7, pp. 34-40, paper
- Stergianos, C., Atkin, J.A.D., Schittekat, P., Nordlander, T.E., Gerada, C., Morvan, H. (2015). “The effects of pushback delays on airport ground movement”. Journal of Applied Operational Research, Vol. 7, No. 2, pp. 68-79, paper

Chapter 5

- “The importance of considering pushback time and arrivals when routing”, presentation at the International Conference on Applied Operational Research (ICAOR 2016), Rotterdam, The Netherlands (29/06/2016), talk
- Stergianos, C., Atkin, J.A.D., Schittekat, P., Nordlander, T.E., Gerada, C., Morvan, H. (2016). “The importance of considering pushback time and arrivals when routing departures on the ground at airports”. 8th International Conference on Applied Operational Research, ICAOR. Lecture Notes in Management Science 8, pp. 41-46, paper

- Stergianos, C., Atkin, J.A.D., Morvan, H. (2016). “Airport ground movement with pushback modelling: Understanding the effects of prioritisation levels for arrivals or departures”. Journal of Applied Operational Research, Vol. 8, No. 1, pp 42-53, paper

Chapter 6

- Stergianos, C., Atkin, J.A.D., Morvan, H. “The effects of airport layout and re-routing on taxiing”, paper in preparation

Chapter 7

- “Combining the ground movement and gate allocation problems”, presentation at the Student Conference on Operational Research (SCOR 2014), Nottingham, UK (03/05/2014), talk
- Stergianos, C., Atkin, J.A.D., Morvan, H. “Airport gate assignment considering the delays that occur during the ground movement process of aircraft.”, paper in preparation

Chapter 8

- “An A-Star Approach for the Quickest Path Problem with Time Windows”, presented at the International Conference on Computational Complexity and Algorithms (ICCA 2017), Sydney, Australia (27/01/2017), talk
- Stergianos, C., Atkin, J.A.D., Morvan, H. “Using a guided search for solving the quickest path problem with time windows” paper in preparation

Furthermore, I have presented my work in the following events:

- “A brief introduction to the Innovate Programme”, presentation at the LANCS workshop on Air Transportation, Nottingham, UK (21/01/2014)
- “WP3 - Green ground operations: Improving the ground movement of airplanes and implementing assisted take-off”, presentation at the INNOVATE Launch event, Nottingham, UK (10/04/2014)
- “The aircraft ground movement problem and its integration with the gate allocation problem”, presentation at the ASAP Seminar, Nottingham, UK (29/10/2014)
- “Optimised preparations to take-off”, presented at the INNOVATE Annual Showcase, Nottingham, UK (15/1/2015)
- “INNOVATE Work Packages Presentation WP3: Ground operations technologies”, presented at the INNOVATE Annual Showcase, Nottingham, UK (15/1/2015)
- “Optimised preparations to take-off”, presented at the INNOVATE Midterm Report Meeting, Nottingham, UK (23/01/2015)
- “Optimised preparations to take-off”, presented at Sintef, Oslo, Norway (03/02/2015)

- “The importance of considering pushback time and arrivals when routing departures on the ground at airports”, presentation at the ASAP Seminar, Nottingham, UK (19/10/2016)
- “Improving the Realism for Ground Movement Models”, presented at the INNOVATE End of Project Conference, Nottingham, UK (25/05/2017)

Moreover, I have presented the following poster:

- “Optimised preparations to take-off”, presented at the ESO Division PhD Poster Competition, Nottingham, UK (05/11/2014)

Finally, I have made two blog posts in order to promote the work of the INNOVATE project.

- “Image recognition on a UAV” INNOVATE Blog at the University of Nottingham blogs (08/12/2014)
- “Is air traffic optimisation really that important?” INNOVATE Blog at the University of Nottingham blogs (22/04/2016)

1.5 Non-disclosure Agreement

The data that has been used for the experiments in this thesis are covered by two non-disclosure agreements. For this reason, the datasets that were used cannot be provided. Chapter 3 provides some details of these datasets.

1.6 Collaborations with Sintef

During my PhD, I had two placements (six months in total) in Sintef – a research centre in Oslo, Norway. The “Optimisation” research group of Sintef Digital has done significant work in optimising air traffic control and has developed their own Air Traffic Control Service that can help the decision-making process of air traffic controllers.

I worked closely with Dr. Patrick Schittekat and Dr. Tomas Nordlander. They provided me with data for Arlanda airport that was used in – and contributed in – three publications. Chapters 4 and 5 include the work that was performed during this placement. The linear model that is used in Chapter 4 is a simplified version of the model that was used in Sintef for the research that was performed for Arlanda airport. Dr. Schittekat and Dr. Nordlander have both contributed in making the model and advising what research directions should be followed during my stay.

1.7 INNOVATE and the Virtual and Physical Demonstrator

As previously mentioned, this research was a part of the INNOVATE project. The aim of the INNOVATE project is to integrate novel technologies in the aerospace sector, such as Propulsion Technologies, Airframe and Control Technologies, Ground Operations Technologies (where most of this research lies) and Innovative Navigation & Communication Technologies. This project aims to link a series of technological advances in various research sectors in aerospace in a virtual and physical demonstrator that will show the applicability of the research as well as the benefits of it.

As stated in the INNOVATE program handbook, “INNOVATE aims to train the next generation of highly skilled engineers and scientists able to understand, undertake and support state-of-the-art technological activities and challenges for the aero-sector in a multi-disciplinary environment.”

13 early stage researchers from different disciplines in the wider aerospace technologies field work together in order to improve and integrate technologies and processes for the aerospace sector. One of the objectives of this project is to work together in order to build a blueprint for the air transportation system that will be used in the future.

Being a part of this project led to the development of two fully autonomous Unmanned Aerial Vehicles (UAV's or drones). I was responsible for building the image recognition process that could detect a specific letter on the ground and release a payload near the target letter. This project was designed to be a learning experience of the methodologies and procedures that would be necessary in order to build a physical demonstrator with the technologies that we would develop during our research.

The researchers of the INNOVATE project were divided into 4 teams and each team would have to build a virtual and physical demonstrator that would integrate the technologies of the research that was performed by the members of the team. The demonstrator that was developed by the team that I was part of is described in detail in Chapter 9 and demonstrates the use of this research in an integrated environment.

1.8 Structure of the Thesis

This thesis is structured as follows:

Chapter 2 presents the relevant background work in the area. It focuses on research that is related to the ground movement problem, but also presents important papers for other processes (such as the pushback, the gate allocation and the runway sequencing process) that are connected with the ground movement process, and attempts to integrate the ground movement process with these processes. Chapter 3 provides a small description of the ground movement problem and explains the primary algorithm that was used for the routing process of aircraft. Moreover, the airports and the datasets that were used are summarized in this chapter. Chapter 4 investigates the effects of the pushback process to the ground movement process by implicitly modeling the pushback process. Chapter 5 examines the effects of different prioritization levels between arrivals and departures when the stand holding process and the pushback process are implemented for departing aircraft. Chapter 6 investigates the effects of the airport layout on the delays that can happen during the ground movement process. The effects of the number of taxiways are examined in terms of the number and the duration of delays, as well as the total taxi time and the use of alternative paths. Chapter 7 presents an integrated model of the ground movement process and the gate allocation process. The conflicting combinations that can arise from an allocation are found and are avoided when the gate allocation problem is solved in order to decrease the delays that are likely to happen when the aircraft are routed. Chapter 8 presents an enhanced approach for routing aircraft using a heuristic that uses the remaining time that is necessary for an aircraft to reach its destination. Chapter 9 presents an implementation of the system into a flight simulator cockpit, along with the outputs of two other projects. The ground movement process is integrated with the positioning and navigation process of the aircraft while the mental demand of the pilot is monitored with non-invasive techniques. Finally, the key results from this research and the various conclusions are discussed in Chapter 10.

Background and Related Work

2.1 Introduction

The movement of aircraft on the surface of an airport is one of the more important and difficult to model (due to the large number and unpredictability of the parameters) optimisation problems at an airport and includes a number of sub-problems that are beneficial to optimise (Atkin et al. 2010, Kjenstad et al. 2013b). Departing aircraft will first push back from the stands (the pushback process), then taxi around the airports (the taxi process), and queue for the runway (runway sequencing process), whereas arrivals will land on a runway and need to taxi to the stands, potentially traversing taxiways in the opposite directions to the departing aircraft.

In this chapter, the related research that has been performed in the areas mentioned above is presented. Section 2.2 describes the constraints and objectives that are used for the ground movement problem and Section 2.3 presents the solutions that have been developed. Section 2.4 briefly introduces the pushback process and Sections 2.5, 2.6 and 2.7 present the related work in the gate allocation problem, the runway scheduling and taxi time prediction respectively. Finally, the chapter concludes in Section 2.8.

2.1.1 Basic terms

It is important to define several terms that are often used when discussing about the movement of the aircraft on the ground.

- Gate – is an area attached to a terminal that passengers can use to board on an aircraft that is parked there.

- Stand – is the area where aircraft can park. This includes the gates where passengers can use to board on an aircraft as well as areas that are not directly connected to a terminal.
- Pushback process - is the part of the ground movement process where the aircraft pushes back from the stand to the taxiway next to the stand and starts its engines.
- Apron – is the area where aircraft park. It includes a group of stands that are close to each other and may share the same resources. Aprons usually provide limited manageability to aircraft, and aircraft that use the aprons usually do not affect (and are not affected) by aircraft that use the main taxiways of the airport
- Towing – is a process where an aircraft is moved from a stand, so the stand (usually a gate) can be used by other aircraft.
- Conflict – (or two aircraft are too close to each other) is when two aircraft are assigned to use the same resources (i.e. segment of a taxiway) at the same.

2.2 The Ground Movement Problem

One of the problems that today's airports face is the ground movement problem. The ground movement problem considers the way aircraft move around the airport in order to reach their destination within a specific timeframe. During the movement of the aircraft it is important that two aircraft never approach too close to each other, as this would result in a conflict. Usually the solution of the routing problem consists of a path that each aircraft has to follow and/or times when the aircraft has to pass different points and intersections in order for an aircraft to reach its destination (runway, stand, de-icing, etc.). Atkin et al. (2010) highlighted the importance of the ground movement optimization tools as they improve the on-time operations within the airport. When taking multiple aircraft into consideration, especially in large and busy airports, it is important to apply a routing algorithm that would optimise their movements. This makes it possible to minimise the total travel time of aircraft moving around the airport's surface and in turn the excess fuel consumption and CO₂ emissions which is a significant concern for the airport.

2.2.1 Constraints

Regarding the ground movement problem there are multiple constraints that have been considered. These constraints include the maximum speed of the aircraft as well as defining the safety distance between the aircraft. Another important constraint of the ground movement problem is the route that each aircraft has to follow. The sequencing of arrivals and departures also plays an important role for the ground movement problem as it can cause

delays. Other constraints of the airport ground movement problem can be the earliest and latest take-off or landing time that can be an unpredictable parameter (i.e. the landing time is usually estimated as it cannot be calculated before the aircraft is near the airport).

Aircraft route

One of the constraints during the routing process of aircraft is the route that the aircraft is able to follow. There are three ways to model this constraint. One way is to limit the aircraft to a single path, another way is to have a set of paths that the aircraft can use, and finally to have no restrictions to which path the aircraft can use.

If the available routes that an aircraft can follow are predetermined, the available routes are already found, so the ground movement solver only focuses on the sequence of the aircraft that want to use a path or part of a path at the same time (Smeltink et al. 2004, Rathinam et al. 2008, Kjenstad et al. 2013a, 2013b, Weiszer et al. 2015b, 2015c, Chen et al. 2015).

If there is a set of predetermined routes that each aircraft can follow (there is more than one path for each aircraft to use) the solver is able to select the best path taking into consideration the other constraints (if there are any) and the objective function (Pesic et al. 2001, Gotteland et al. 2001, 2003a, Gotteland and Durand 2003b, Herrero et al. 2005, Garcia et al. 2005, Balakrishnan and Jung 2007, Roling and Visser 2008, Deau et al. 2008, 2009, Evertse and Visser 2017).

Finally, the paths that an aircraft can follow can be completely unrestricted and the ground movement solver has to find the shortest or the quickest path, each time an aircraft is routed (Marín 2006, Marín and Codina 2008, Keith et al. 2008, Clare et al. 2009a, Clare and Richards 2009b, 2011, Brownlee et al. 2014, Weiszer et al. 2015a, Benlic et al. 2016, Chen et al. 2016b).

For this thesis, two of these approaches (using a single path for each flight and having completely unrestricted paths) were used in Chapter 4. Having no restrictions to the paths that aircraft can use, performed better in terms of overall delays between aircraft, compared to restricting the aircraft to using a single path, so this approach was used for the rest of the research.

Separation Rules and conflict detection

Another constraint that has to be taken into account while dealing with the optimisation of the ground movement problem is avoiding conflicts between the aircraft (two aircraft being too close to each other) and avoiding accidents. The distance between the aircraft however is not standard and different authors suggest different distances, as discussed below, but this requirement has to be respected in order to avoid conflicting aircraft. There is not a strict rule

applied in airports and keeping a safe distance from the leading aircraft is at the discretion of the pilot.

There is a minimum distance of 200 meters that is suggested by Smeltink et al. (2004), Roling and Visser (2008), Rathinam et al. (2008) and Burgain et al. (2012) whereas Gotteland et al. (2001) and Pesic et al. (2001) suggest a distance of no less than 60 meters between taxiing aircraft. Weiszer et al. (2015c) use a minimum safety time distance between aircraft which is set at 12 seconds. As mentioned in the paper, this is equal to 62 meters when the aircraft is moving at the speed of 10 knots. As there are no specific separation rules applied by all airports Weiszer et al. (2014, 2015a) have restricted the aircraft from using one edge (small part of the taxiway) at a time.

A similar method to Weiszer et al. (2014, 2015a) has been used in the research of this thesis, where each edge can be used by only one aircraft at a time. The length of the edges of the graphs that have been used in this thesis has been limited to a certain size to more realistically represent the minimum distance between two aircraft as it will be seen in the next chapter.

In this thesis, the same approach that is followed by Weiszer et al. (2014, 2015a) has been implemented as it is more realistic.

Aircraft speed

Maximum and minimum speeds are also constrained as a part of the ground movement problem to ensure that aircraft move safely and efficiently in the airport. Smeltink et al. (2004) mention that speeds below 8 knots rarely occur and use a minimum speed of 5 knots in order to constrain the search space and reduce the computation time. In some more precise models, the speed is lower while aircraft traverse through a turn and higher when they move in a straight line (Ravizza et al. 2013a, Weiszer et al. 2014, Chen 2016a). Moreover, Pesic et al. (2001) consider the time aircraft need when turning. However, other factors such as the nature of the taxiway or the gate and runway accessibility (Gotteland et al. 2001) or the type and size of the aircraft (Balakrishnan and Jung 2007, Roling and Visser 2008) should be taken into account. Rathinam et al. (2008) find the speed of each aircraft in the sectors along its route using a maximum speed of 15.5 knots. Chen et al. (2016a) make a more detailed speed profile for aircraft, breaking the type of movement of each aircraft to four parts: acceleration, travelling at constant speed, braking and rapid braking. They use 30 knots as a maximum speed when the aircraft is traversing a straight segment and 10 knots when the aircraft is turning.

While maximum and minimum speeds of aircraft that are taxiing are considered as constraints for the solution of the ground movement problem, aircraft can also be held at specific points along their path while taxiing. Smeltink et al. (2004) suggest the limit of one aircraft waiting

at a holding point at a time. Gotteland et al. (2001) recommend waiting on the taxiway rather than using a longer route or wait at the gate and they search for convenient holding points. A model that suggests points where aircraft can hold is also suggested by Gotteland and Durand (2003b). Furthermore, aircraft holding is also implemented by Chen et al. (2016b) when there is a conflict and no other alternative movements can be found.

For the experiments of this thesis the speed of 16 knots has been implemented similar to Rathinam et al. (2008) since it is closer to the average speed that aircraft will move around the airport (Roling and Visser 2008). Since the research in this thesis does not focus on the fuel consumption of the aircraft and in order to reduce the complexity of the problem, the speed of the aircraft is considered to be constant when their path is not interrupted by other aircraft. This however, does not significantly affect the delays that can happen, since the aircraft are modelled to occupy segments of the taxiways for the whole duration of the time that it takes to traverse each segment, regardless the variations of the speed that an aircraft can have during the use of a segment.

Timing (departures and arrivals)

For arriving aircraft, the aim is usually to reach the stand that they are allocated to as soon as possible, as they can then turn-off their engines and unload the passengers. For this reason, most of the research uses the landing time of an aircraft as a constant and force the aircraft to start their journey to their allocated stand after this time (Ravizza et al. 2014, Weiszer et al. 2015b).

Departing aircraft need to travel from the stand that they are parked at, to the runway that they will take-off from. For departing aircraft, arriving at the runway early can result in wasting fuel since once the engines of an aircraft start up, they usually stay on and consume fuel even on idle.

There are many airports where aircraft have to wait at a holding point before they can enter the runway and depart. Furthermore, possible queuing as well as preparations that have to be completed in order for the aircraft to be able to fly should always be taken into account. As aircraft have to depart at a scheduled time (in order to respect the take-off sequence), the pushback process has to start within a fixed time slot.

Usually there are three ways to model time constraints of the departing aircraft.

- One is for the aircraft to arrive at the runway as fast as possible starting from the time that an aircraft is set to initiate its pushback process. Since however, this can result in excess fuel consumption, it is more common for aircraft to reach the runway at a specific time or within a certain time frame.

- Another aim can be for departing aircraft to reach the runway within a specified timeframe (Gotteland et al. 2003a, Deau et al. 2009, Benlic et al. 2016). According to the European Central Flow Management Unit (CFMU) each aircraft has to take-off within a specific timeframe based on the set departure time. Missing a time window can cause penalties and can be costly for the airport. During the peak time in busy airports there may be a delay or waiting time while the aircraft is taxiing or while entering the runway. In some cases, these timing uncertainties make it hard to optimize the runway sequencing problem and to respect all the time slots.
- Finally, another way to model departing aircraft is to wait at the stand (before the engines are turned on) and start taxiing as late as possible but still arrive at the runway on time. Ravizza et al. (2014) use a method called stand holding in order to have departing aircraft to start as late as possible and still reach the runway on time by taking into consideration the delays that can happen during the ground movement process, transferring all the excess waiting time at the stand.

Stand holding has also been used in this thesis (see Chapter 5) as it is the most efficient way to reduce the time that an aircraft has to wait at the runway. As discussed in Chapter 5, it is important to reduce excess waiting time at the runway since once the engines are turned on, they continue to burn fuel even on idle.

2.2.2 Objectives

The objectives of the ground movement problem and the route aircraft will have to follow differ depending on the aim of each implementation. Departing aircraft will have to be at the runway at a specific time ready for take-off, while arriving aircraft will have to reach a specific stand which they are allocated to, as early as possible. In most cases, the aim is to reduce the total taxi time or the fuel consumption which can be achieved with starting the engines as late as possible for departing aircraft and turn the engines off as early as possible for arriving aircraft. The objectives that previous work has been focusing on, are presented below.

Total taxi time

Usually the main objective of the ground movement problem is to minimise the total taxi time of the aircraft. In some cases, ground movement aims at reducing the total taxi time as well as the waiting time at the runway (Van Velthuisen 1997, Pesic et al. 2001, Smeltink et al. 2004, Marín 2006, Roling and Visser 2008, Rathinam et al. 2008, Roling 2009, 2011, Ravizza and Atkin 2011, Marín 2013, Ravizza et al. 2014).

Fuel consumption

More recently, minimising the fuel consumption is also an important objective as calculating the fuel consumption is getting more accurate (Jung et al. 2011, Nikoleris et al. 2011, Chen and Stewart 2011a, Ravizza et al. 2013b). However, lower fuel consumption and minimizing the total taxi time can be two conflicting factors as shorter times demand higher speeds. Therefore, recent research has been focusing on the simultaneous study of fuel consumption and total taxi time (Weiszer et al. 2014). In order to achieve that, Weiszer et al. (2014) suggest a set of optimal ground movement speed profiles based on a heuristic method that examines acceleration, constant speed, braking and rapid braking. Chen et al. (2015) use a multi-objective function that takes into consideration the taxi time, the fuel consumption, the HC emissions and CO emissions. Weiszer et al. (2015a and 2015b) use a multi-objective function that takes into consideration the total taxi time and the fuel consumption.

Other objectives

Ground movement research can also focus on the minimization of the delays that can happen during the routing process (Herrero et al. 2005, Garcia et al. 2005).

Other researchers consider additional factors such as respecting the CFMU time slots (Gotteland et al. 2003a, Deau et al. 2008, 2009) or scheduled time slots within which aircraft have to arrive or depart (Smeltink et al. 2004, Balakrishnan and Jung 2007) and minimising the total taxi distance (Clare et al. 2009a, Clare and Richards 2009b, 2011).

Multiple objectives can also be considered. In Marín and Codina (2008), the objective function is a combination of the total taxiing time, worst taxiing time, the delays that happen during taxiing, the number of departures and arrivals and the times controllers have to interfere.

The objective of the model that has been developed in this thesis is the minimisation of the total taxiing time (e.g. using the quickest path, and avoiding delays). Even though reducing the fuel consumption is also an important objective, it is not the primary goal of this thesis. This thesis focuses more on building and evaluating an enhanced ground movement model by implementing or integrating with other processes such as the pushback process, the stand holding process and the gate allocation process. Methods that are used in other models could be applied to this model, in order to investigate the fuel burn trade-off when considering the pushback process, the stand holding process and the gate allocation process. Since there is a high correlation between the time that an aircraft stays with its engines on and the fuel consumption, it is assumed that reducing the total taxiing time will also reduce the total fuel consumption.

2.3 Ground Movement Problem Solutions

There has been significant research regarding the ground movement problem and finding the ideal route for each aircraft. Usually, ground movement solution methods are based on a graph representation of an airport, where intersections are represented as nodes and taxiways as arcs. The result of the ground movement process is a route that an aircraft should follow, as well as the times that specific nodes on the graph should be reached or traversed.

Most of the existing research in this area involves the use of genetic algorithms or Mixed Integer Linear Programming (MILP) models to solve the ground movement problem. MILP based solutions can guarantee an optimal solution but all objectives and variables need to be linear and some problems can take a long time to solve. Genetic algorithms on the other hand might not give an optimal solution. However, they can be more realistic and may lead to a solution faster. Furthermore, other heuristic methods have been used to solve the problem as they can find a near-optimum solution in a short amount of time.

Two methodologies are considered and compared in Chapter 4, one being a MILP formulation of the ground movement problem and the other one being a heuristic algorithm that finds the quickest path for each aircraft taking into consideration the aircraft that have been previously routed. A number of alternative methods have also been used in the literature to solve the ground movement problem - such as genetic algorithms, MILP formulation and other heuristic approaches. Their use and benefits are presented in the following subsections.

2.3.1 Genetic Algorithms

Genetic algorithms are one of the methodologies that has been widely used for solving the ground movement problem as they can solve the problem in a more realistic amount of time. Genetic algorithms generate a population of possible solutions and repeatedly modify them. In each step, they use a fitness function to evaluate each solution and choose the ones that provide a better solution. Traits from the best performing solutions are used to form new solutions which will again be evaluated, and the best ones are again chosen until a near optimal solution is found.

Genetic algorithm research

Pesic et al. (2001) propose the use of a Genetic Algorithm as a method for taxi optimization at Charles De Gaulle Airport. The model selects the right taxiway in order to reduce the commuting time from the runway to the stand and vice versa. The model then tries to find a solution that does not include a conflict (e.g. two aircraft that are too close to each other). The

aim is to produce a solution with a better fitness value by using parents with fewer conflicts in order to have children with a better result. A mutation operator (a way to randomly change the existing solution to increase the diversity of solutions) was used to change the solutions with the worst fitness values focusing on making a more feasible solution.

Durand and Alliot (1998) propose a combination of genetic algorithms and a mathematical technique called partial separation. The research proposes the use of partially separable functions in order to improve the rate of convergence of the genetic algorithms in use. In order to optimize the global objective function, each variable is optimized separately and the use of the genetic algorithm with adapted crossover seems to be the most effective method.

An additional parameter is considered by Gotteland et al. (2001). They tried to minimize the time that is spent from landing to reaching the stand and from leaving the stand to taking-off through a ground congestion simulation method that tries to solve conflicts. The simulation model chooses a path and an optional holding point on the taxiway and time, using an A* algorithm which prioritises the aircraft in a way that conflicts are avoided (considering the routes of other aircraft as well).

Gotteland et al. (2003a) try to also include CFMU slot constraints in order to achieve accuracy of predicted take-off times in their ground traffic simulation model. A combination of the Dijkstra's algorithm and a Recursive Enumeration Algorithm is used to provide the appropriate paths and holding times.

Genetic algorithms are also used by Gotteland and Durand (2003b) for the simulation tool that is implemented at Charles De Gaulle airport. The aim of their research is to reduce taxi time by considering the capacity of the runways and at the same time respecting the separation rules between aircraft. In order to reduce delays a Branch & Bound algorithm which finds the optimal path for the aircraft is used. They apply two different genetic algorithm based methods; the first method finds combinations of paths and holding positions and the second method finds the combinations of paths and the priority between aircraft.

A simulation model for calculating the decrease in delays after optimising the runway sequencing of the aircraft and the conflicts that happen during taxiing is used by Deau et al. (2008). Deau et al. use the time slots that aircraft need to arrive at the stand as variables of the problem and the most important constraint they take into account is the wake turbulence separation. The aim of the research is to reduce the departure delays and be consistent with the time slots. For the solution of the problem they use a branch and bound algorithm. The solution method for conflicting aircraft used a genetic algorithm that uses the predetermined runway sequences as a target rather than a constraint. Similarly, Deau et al. (2009) contrast the runway planning delays to the total ground delays while considering all the taxiing

constraints in order to reduce the delays that happen when aircraft move around the airport. A hybrid optimisation method is introduced where a genetic algorithm searches for optimal combinations of paths.

Herrero et al. (2005) present two approaches that make use of a time-space flow algorithm and a genetic algorithm in order to reduce ground delays. In the first approach, the problem is modelled using time constrained arcs within a network and the modified flow algorithm estimates the maximum number of demanded operations that can be routed within a specific period. The second approach uses a genetic algorithm where the number of the departures is defined from the beginning. This approach provides a more flexible representation of the problem and the ability to consider the operations separately.

Garcia et al. (2005) hybridise a genetic algorithm with a time-space dynamic flow-management algorithm in order to reduce the delays. In particular, they use a modified minimum-cost maximum-flow algorithm to find the initial population. Initially the genetic algorithm with the use of an artificial intelligence method is used to solve the routing problem using the routes and time schedules as variables. Then a deterministic flow algorithm is used to form an initial flow distribution and the genetic algorithm uses a fitness function to process the initial solution and find better ones (hybrid strategy).

Weiszer et al. (2015b) use an implementation of the Fast Non-dominated Sorting Genetic Algorithm to solve the scheduling and routing of aircraft. Their model also takes into consideration other problems that are affected or affect the routing and scheduling of aircraft. The objective function tries to minimise the total taxi time and the fuel consumption. An initial population of “individuals” is randomly created and the “individuals” with the best objective values are selected. New “individuals” are created with a 2-point crossover and a mutation is applied to each of them. A similar approach is followed in Weiszer et al. (2015c) as well.

2.3.2 Mixed Integer Linear Programming

Mixed Integer Linear Programming Formulations are used in many cases of operational research and they can be used as a method to obtain optimal solutions. Mixed Integer Linear Programming Formulations are an extension of Linear Programming. In Linear Programming, all constraints and the objective function have to be linear. However, in Mixed Integer Linear Programming all or some of the variables can be integer. This restriction increases the complexity of the model and finding a solution to a large-scale problem can become difficult. Mixed Integer Linear Programming models for this problem, according to

Keith et al. (2008), can offer the advantage of continuous variables for the constraints but for the routing and runway order integer variables can be used.

Mixed Integer Linear Programming (MILP) research

The mixed integer programming formulation method is often used for an optimization model for the routing and scheduling of aircraft at the airport.

In the model that was developed by Smeltink et al. (2004), the route and the arrival/departure time for each aircraft are used as input and the aim is to reduce the waiting times while taxiing and to find the times that the aircraft should pass from particular points at the airport. In this paper, it is suggested that a minimum speed should be introduced in order to reduce computation time. The sequence each aircraft passes through a node is known unless two aircraft want to use the same node at the same time. The research uses binary variables for finding the optimal sequence in this case. In general, the results show that the delays due to taxi conflicts are reduced.

Marín (2006) uses a linear multicommodity flow network model for solving the routing and scheduling problem of aircraft on the ground. A Branch-and-Bound and a Fix and Relax (some variables get fixed values during various stages of the solving process to speed up the process) method are used in the MILP model which aims to reduce the total taxiing time of all flights. The results of the Fix and Relax method outperform those of the Branch-and-Bound method. Marín and Codina (2008) extend the taxi planning research that is proposed by Marín (2006) and suggest a binary multicommodity network flow approach that takes into consideration multiple objectives. The multi-objective approach attempts to reduce the controller interference regarding the solution of conflicts that may arise, improve the routing time and the delay for arrivals and departures, and the increase of number of aircraft that can move on the airport at the same time without introducing more delays. Marín (2013) use Lagrangian decomposition in order to solve the routing and scheduling process and to minimise the taxi time of the aircraft.

Another integer programming formulation is used by Balakrishnan and Jung (2007) for Dallas-Fort Worth airport in order to optimise the taxi route planning. Two different aspects of the ground movement are discussed: these are controlled pushback and taxi reroute. The model tries to find the surface routes and pushback times of the aircraft in order to reduce the cost considering capacity and speed limits and a penalty is applied when the aircraft does not depart on time. This method seems to be effective as it decreases both the departure and arrival taxi time and the waiting time of the aircraft on the runway is significantly reduced as well. All these factors could finally play an important role in saving fuel. However, the

particular model does not take into account the variation in taxi speeds and it may not be reliable enough in order to be used at the airports.

Rathinam et al. (2008), suggest another MILP optimisation model for the ground movement problem of the aircraft at Dallas-Fort Worth International airport. The objective function minimises the total taxi time. The problem formulation is based on the research of Smeltink et al. (2004). However, in Smeltink et al. the separation constraints are not included. Also, there is an attempt to minimize the size of the problem by reducing the number of variables as well as the number of constraints.

Roling and Visser (2008) also proposed a MILP model based model while trying to provide an optimal solution for the ground movement problem. In particular, it is a deterministic space and time model that tries to reduce the delays and the stops during taxiing through an optimal scheduling of the aircraft. For this reason, it provides ground movement plans for each aircraft with specified routes and times of arrival at each node. As the data used for taxi planning is not precise, uncertainty is taken into account. The objective function tries to reduce a weighted combination of the total taxi time and holding time and it is restricted within a fixed planning horizon where conflicts are avoided by allocating one flight at a time to a node. However, this idea was applied on a hypothetical airport and the suggested taxi time plan is based on future guidance and control systems that will be able to precisely predict taxiing operations.

Roling (2009) designs a taxi planning system that deals with more realistic scenarios. The objective function of the model considers the total taxi time. Moreover, the model takes into account: a constraint that blocks nodes when they are being used; a delay and route constraint that assign a specific route and delay to each flight; and a waiting time constraint that applies any delays to the next segments of an aircraft's path as well. Roling (2011) further extend this work by applying the model on different large airports to optimise the taxi times of aircraft.

Keith et al. (2008) also propose a MILP optimisation method for the combination of airport taxiway and runway scheduling problem. Their model is similar to the routing protocol that Marín (2006) used for the taxi routing time and their continuous time formulation is based on the Smeltink et al. (2004) model. In particular, the proposed optimisation model uses binary variables in order to arrange the route of each aircraft in a similar way as in Smeltink et al. (2004) and the aircraft can move from one point to another within a defined period in order to satisfy the constraints, as in Marín (2006). The objective function in this model consists of a weighted combination of the total taxi time and taxi distance, the final time on the runway and a penalty according to the remaining distance which aircraft need to traverse in order to reach their destination. The MILP model introduced by Keith et al. was modified and turned

into an iterative MILP formulation for departures by Clare et al. (2009a) in order to decrease the computational demand for the separation constraints between aircraft. At first the added constraints are relaxed, and each iteration is repeated until a solution without conflicts is found.

Clare and Richards (2009b, 20011) use Iterative Receding Horizon MILP models that solve the routing and the runway sequencing problem simultaneously. The objective function of Clare and Richards (2009b) consists of the total taxi time, the total taxi distance, a penalty which is connected to the remaining distance the aircraft need to travel and the time that the aircraft have been through the last node. The objective function of Clare and Richards (2011) takes into consideration the final time that the aircraft arrives at the runway, the total duration of taxiing, the total distance traversed, and the distance left to travel based on the terminal that is used.

Another MILP formulation is proposed by Yin et al. (2012) with data from George Bush Intercontinental airport in Houston, Texas. The model tries to minimise the total taxi times and cost through optimal taxi routes and schedules for each aircraft. Although this model can help in reducing the taxi times and the fuel consumption it is rather difficult to solve. The computational time for the optimal solution can be very long when it comes to a large-scale problem despite the use of the rolling horizon method.

Evertse and Visser (2017) present an MILP model that can plan in real time the taxi movement of aircraft. It takes into consideration unpredicted events by updating the solution every second. In order to reduce the complexity of the problem, the problem was decomposed by using a rolling horizon approach. In this formulation, the aircraft are allowed to temporarily stop at any point (node of the graph) while they are traversing a taxiway. The objective function focuses on minimising deviations from the CFMU slots and the emissions of the aircraft.

2.3.3 Other Solution Methods

Other solution methods have also been used in order to solve the ground movement problem and reach optimal or near optimal solutions. Researchers have used heuristic methods, which include algorithms that can find an almost optimal solution within a short computational time. If there is a difficult problem that cannot be solved using an exact method, heuristics could be the most effective solution method to reach to a good solution.

Anderson et al. (2000) suggest the use of simple queuing models to control the taxi-in (arrivals) and taxi-out (departures) procedures as well as a model that considers aircraft turns

in order to understand the dynamics of the airport ground operations. Simple queuing models are also used by Carr et al. (2002) in order to handle departure congestion.

In Brinton et al. (2002) an event-based A* algorithm and a Co-Evolution strategy that provide route schedules are tested. When the two approaches are compared regarding the time that aircraft need to arrive to their destination the event-based A* algorithm seems more effective than the Co-Evolution strategy.

Cheng and Foyle (2002) and Cheng (2003) suggest the use of the Ground-Operation Situation Awareness and Flow Efficiency (GO-SAFE) approach, which provides traffic information to the ground controller and supports him with taxi route planning as well as runway scheduling.

A simulation-based architecture that aims to improve the ground movement of the aircraft in the taxiways next to the gates is suggested by Confessore et al. (2005). The proposed model consists of a simulation model and an optimization module and the objective of the research is to minimise traffic and delays that happen on the ground.

A time-based simulation model for organising the movement of the aircraft on the taxiways and understanding the communication between controllers and pilots is proposed by Baik et al. A time-dependent shortest path algorithm is used to provide optimal taxi routes for aircraft to traverse between stands and runways (Baik et al. 2002, Baik and Trani 2008).

Cheng (2007) suggests a Flight-Deck Automation for Reliable Ground Operation (FARGO) model to improve taxi control and precision. The proposed system enables communication between tower controllers and pilots regarding surface traffic information.

Ravizza et al. (2011, 2014) and Weiszer et al. (2014) use heuristics in order to solve the routing and scheduling problems. They both use a routing algorithm that routes aircraft sequentially based on Dijkstra's algorithm. The algorithm finds the quickest available path for each aircraft, by taking into consideration previously routed aircraft. Ravizza and Atkin (2011) try different heuristic methods to optimise the sequence that aircraft are being considered, since the aircraft that is routed first is implicitly prioritised.

Kjenstad et al. (2013a, 2013b) use a heuristic decomposition of the ground movement and departure sequencing problem. First the shortest route is found for each aircraft and then any conflicts between aircraft are solved taking into consideration the departure sequence. Lesire (2009, 2010) solves the routing and scheduling problem with an A* approach. The Lesire (2010) approach uses a heuristic based on the Euclidian distance in order to find the optimal path for each aircraft whereas Lesire (2009) improves a Contract Reservation Algorithm (CRA) by using a heuristic and a pruning algorithm.

Mori (2010) uses cellular automata to simulate the congestion that occurs during departures. Dijkstra's algorithm is used by Gupta et al. (2010a) to find the optimal routes on a runway and taxiway network in order to solve the ground movement problem. Bhadra et al. (2011) propose an Airport Surface Detection Equipment Model X (ASDE-X) system that uses virtual queuing in order to deal with congestion that happens during departures. ASDE-X has also been used by Srivastava (2011) in order to create a taxi out prediction model.

2.3.4 Comparison of the approaches

The majority of the ground movement research has used approaches that are based on Genetic algorithms and MILP models for the solution of the problem. GAs are heuristics which are not exact solution methods and, as a result they cannot guarantee an optimal solution of the problem. However, MILP models are capable of providing optimal solutions with the condition that all of the objectives and the constraints are linear. Although there are various ways to linearise the considered constraints and objectives, this condition can result in large and complicated models that require significant computational time to provide a solution. Therefore, the accuracy of the results could be affected. A less common solution approach for solving the ground movement problem is the use of heuristics when for example the size of the problem is uncertain, or the available solution time is inadequate to find an exact solution.

As airport operation systems require real-time decision making, computation time is a very important factor for the selection of a method. Thus, GAs often outperform MILP models. For instance, in the paper of Roling and Visser (2008) the computational time increases significantly when more flights are added to their MILP formulation.

In addition, the complexity of the ground movement problem often does not allow the use of a realistic scenario. However more realistic scenarios should be considered when designing models for the optimal solution of the ground movement problem. Moreover, it would be beneficial to investigate a variety of solution methods such as metaheuristics and hybrid approaches in order to take advantage of the assets of various models.

2.4 Pushback Process

The pushback process is an important aspect to consider when optimising the ground movement process of aircraft since long delays can happen during this process. Delays can happen by aircraft that pushback and block other aircraft or by aircraft not being able to

pushback due to traffic near the stand where they are parked. For this reason, it is important to consider the pushback process when designing a ground movement optimisation model. Chapters 4 and 5 provide more information on the research that has been performed in this area. Since the pushback process is a more specific part of the ground movement process the background research on this area has been described in detail in Chapter 4. This makes it easier for the reader to better understand the context in which this chapter was developed.

2.5 The Gate Assignment Problem

Another problem that airports have to deal with is the gate allocation problem. The gate allocation problem studies how flights should be assigned to particular stands in a way that airports could avoid gate closures, blocked gates and delays in order to save money and time. Although there have been many attempts for solving this problem, there have been a few models that the gate allocation process is integrated with the aircraft ground movement problem in order to find an optimal or near optimal solution for the wider problem. The gate assignment problem can affect other airport operation processes as the stands where the aircraft need to travel can affect the taxi time, the sequence and the scheduling of aircraft resulting in delays, increased fuel consumption and consequently increased expenses.

2.5.1 Constraints

Various constraints have to be considered when dealing with the gate allocation problem (see below). The solution of the gate assignment problem includes the proper assignment of a gate to all of the aircraft as well as the times that aircraft need to be at their allocated gate.

Hard constraints

Two flights cannot be allocated to the same gate at the same time.

One of the most important constraints that need to be taken into consideration is that one stand cannot be assigned to two flights at the same time (Xu and Bailey 2001, Ding et al. 2005, Dorndorf et al. 2008, 2012, Tang et al. 2010, Xu et al. 2011, Neuman and Atkin 2013, Kim and Feron 2014, Kumar et al. 2014). If the time that the aircraft arrive or leave their stand is the input of the solution method, then two aircraft with the same stand times cannot be allocated to the same stand. In case the stand times are the output of the solution method two aircraft could be allocated to the same stand as long as there is a long enough delay to the second aircraft so that the stand times for the two aircraft no longer overlap each other

(Ding et al. 2005, Lim et al. 2005, Dorndorf et al. 2008, Neuman Atkin 2013, Kim and Feron 2014). Such a delay, however, would be costly for the airport.

Aircraft should be assigned to a suitable stand.

Each aircraft-stand assignment should be relevant to the size of the aircraft (Dorndorf et al. 2008, 2012, Neuman and Atkin 2013) and relevant stand facilities for the aircraft. Airlines might also have preferences regarding facilities that might be available at specific stands (Drex1 and Nikulin 2008, Nikulin and Drex1 2010, Genc et al. 2012, Neuman Atkin 2013). This would mean that specific flights are allocated to specific stands at airports. Furthermore, the process of stand allocation considers safety measures such as available checks if there is a domestic or international flight.

Aircraft-stand combination that are disallowed.

When an aircraft is allocated to a specific stand, aircraft might not be able to use other stands that are close by (Dorndorf et al. 2008, 2012, Nikulin and Drex1 2010). Also, there might only be specific types of aircraft that would be able to use these stands. This could be the case when the aircraft is too large and prevents the use of a nearby stand by other aircraft as in Neuman and Atkin (2013).

Soft constraints

Alongside with the above key constraints there are also other constraints that need to be considered when dealing with the gate allocation problem.

- One constraint is the effort to minimise the distance that passenger have to walk through the airport (Babić et al. 1984, Mangoubi and Mathaisel 1985). However, the walking distance problem does not seem to be taken into consideration any more as airports are more interested in environmental issues as well as ways to save fuel and consequently money.
- Other researchers introduce ways to spread out aircraft in order to avoid congestion at the airport (Kim et al. 2009, 2010).
- Airlines might have preferences regarding the gates that they use or the stands that their aircraft are parked and might also follow a particular regular schedule which could be beneficial (Nikulin and Drex1 2010, Dorndorf et al. 2012).
- Furthermore, there might be a time gap that needs to be respected between consecutive flights that are allocated to the same gate (Dorndorf et al. 2008, Kim et al. 2010, Kim and Feron 2011, Kumar et al. 20011, Neuman and Atkin, 2013).
- In addition to this, Kumar et al. (2011) suggest a pushback time gap constraint during which aircraft with the same pushback route need to perform their pushback.

- Some researches might be dealing with a towing constraint when aircraft are staying long at an airport and they have to be removed from the gate/stand (Dorndorf et al. 2007, 2008, Diepen et al. 2009, Kumar et al. 2011, Neuman and Atkin 2013).
- Lastly, the preceding constraint suggests that each flight can only be followed by a maximum one flight (Xu and Bailey 2001, Lim et al. 2005).

2.5.2 Objectives

The gate allocation problem consists of several objectives.

One of the objectives is to reduce the deviation from the schedule (Cheng 1997, Dorndorf et al. 2007, 2012, Drexl and Nikulin 2008, Nikulin and Drexl 2010,) or to increase the time gaps/reduce the deviation from the time gaps between flights that are allocated to gates (Dorndorf et al. 2008, 2012, Diepen et al. 2009, Bolat 2000, Neuman and Atkin, 2013).

Also, the objective of gate assignment research could be that flights have to be allocated to the appropriate gate (Dorndorf et al. 2007, 2008, 2012, Drexl and Nikulin 2008, Nikulin and Drexl 2010, Neuman and Atkin, 2013) or the reduction of the time an aircraft has to hold to be assigned to a gate in case there are no available gates which might cause delays (Lim et al. 2005, Hu and Di Paolo 2007).

Another objective that might be considered is to reduce the number of aircraft that are not allocated to a gate and need to use a remote stand (Ding et al. 2004a, 2005, Dorndorf et al. 2008, 2012, Drexl and Nikulin 2008, Neuman and Atkin 2013) or the conflicts that might occur around gates (Kim et al. 2009, Neuman and Atkin 2013).

Another objective is the reduction of the distance luggage has to be transferred (Hu and Di Paolo 2007) and the passenger walking distance or waiting time (Babić et al. 1984, Haghani and Chen 1998, Xu and G. Bailey 2001, Yan and Hou 2001, Ding et al. 2004a, 2004b, 2005, Hu and Di Paolo 2007, Drexl and Nikulin 2008, Kim et al. 2010).

Lastly airports might have to consider the reduction of the towing of each aircraft (Dorndorf et al. 2007, 2008, 2012, Nikulin and Drexl 2010, Kumar et al. 2014) or the maximisation of the duration that each gate is in use (Genc et al. 2012).

2.5.3 Gate Assignment Problem Solutions

Different solution methods have been used to solve the gate assignment problem.

One of the solutions for the gate assignment problem uses Linear Programming. The LP model that was developed by Bihr (1990) focuses on the minimisation of the distance passengers walk and the constraints include each aircraft being assigned to one gate and that all aircraft are assigned to a gate. However, the structure of the solution method is not that flexible when last minute changes need to be done (e.g. flight delays). Xu and Bailly (2001) propose a model that allocates gates to flights while considering daily data for passengers and their destination. The problem is developed as a mixed integer problem with linear objective function and constraints, and a meta-heuristic tabu model is used for the solution of the problem. Yan and Hou (2001) propose a multiple objective zero-one integer model that tries to reduce the walking distance and the passenger waiting time.

Mangoubi and Mathaisel (1985) compare two different methodologies for minimising passenger walking distance, an LP model and a heuristic method. Their results show that using heuristics makes it possible to obtain a nearly optimal solution to the problem much faster than when solving the problem with the LP model.

Another solution for the gate assignment problem uses a genetic algorithm which creates the chromosomes based on the relative positions between the aircraft in the queues to the gates instead of the absolute positions. (Hu and Di Paolo 2007).

A recent research of Schaijk and Visser (2017) also deal with the gate assignment problem. In their research, they change the constraints from deterministic into stochastic. The solution model assigns gates to flights in a way that the delays that happen are not more than a given value using binary integer programming.

2.5.4 Integration with the Ground Movement Problem

The gate allocation process significantly affects the ground movement process of aircraft. There have been various attempts to integrate the gate allocation process with the ground movement process. Since the integration between the gate allocation process and the ground movement process is explicitly considered in Chapter 7, and in order to provide a better context in which this chapter was developed, the research that has been performed in this area has been presented in Chapter 7.

2.6 Runway Scheduling

The runway scheduling process also affects the ground movement process of aircraft. Airports have to deal with the ground movement problem as soon as aircraft land on the

runway or when they leave their stand. As a result, ground movement operation systems require the aircraft entry and exit times. Both arrival and departure runway scheduling are an important issue that has to be dealt with when solving the ground movement problem.

In this section, the constraints and objectives of the runway sequencing problem are introduced and the previous research as well as the integration with the ground movement process at airports are presented.

2.6.1 Constraints

Various separation rules should be considered in order to increase the capacity of the runways (Atkin et al. 2008, 2009, Benlic et al. 2016).

Wake vortex/turbulence

The constraint that is mostly taken into consideration when dealing with the optimization of the runway scheduling problem is the separation between aircraft (Gotteland et al. 2001, Leese et al. 2001, Deau et al. 2008, 2009, Clare et al. 2009a, Rathinam et al. 2008, Keith et al. 2008, Atkin et al. 2008, 2009, Clare and Richards 2011, Lee and Balakrishnan 2012, Weiszer et al. 2015c, Benlic et al. 2016). Wake vortices are created by the aircraft taking-off, which are linked to the weight of the aircraft. For this reason, the aircraft that wants to use the runway next needs to wait a certain amount of time for the vortices to dissipate and use the runway safely. Deau et al. (2008) classify aircraft as low, medium and heavy according to their weight and explain that there is a minimum waiting time of 180 seconds for a “low” aircraft which is following a “heavy” aircraft departure.

En-route separations

Moreover, existing papers consider the en-route separations when aircraft follow similar routes (Keith et al. 2008, Atkin et al. 2008, 2009, Deau et al. 2009, Clare et al. 2009a, Clare and Richards 2009b, 2011, Benlic et al. 2016).

Aircraft speed

Speeds of aircraft are also considered in order to adjust separation between them (Atkin et al. 2008, 2009, Deau et al. 2009, Benlic et al. 2016).

Other parameters

Taxi time prediction is also an important factor for optimising the runway sequencing problem (Atkin et al. 2006, 2008, Ravizza et al. 2014). Moreover, another factor that should be considered is that aircraft might have to wait at a holding point close to the runway (Leese

et al. 2001, Atkin et al. 2007). Atkin et al. (2006, 2008) highlight the importance of taxi time prediction for more effective runway scheduling models.

2.6.2 Objectives

Several objectives are considered when solving the runway scheduling problem. One of the objectives is finding a sequence - that departing and arriving aircraft reach the runway - that minimises the time between two consecutive take-offs or landings. The importance of an optimised departure schedule that should be assigned to departing aircraft is highlighted in various papers (Leese et al. 2001, Anagnostakis and Clarke 2003, Atkin et al. 2006, 2007, 2008, Keith et al. 2008, Deau et al. 2008, Deau et al. 2009, Gupta et al. 2010b, Apice et al. 2014), while other papers consider sequencing of arrivals (Ernst et al. 1999, Bianco et al. 1999, Chandran and Balakrishnan 2007, Tavakkoli et al. 2012). Other papers consider optimal sequences for both arrivals and departures (Li et al. 2009, Balakrishnan and Chandran 2010, Weiszer et al. 2015c, De Maere and Atkin 2015, Benlic et al. 2016).

Papers that deal with the runway problem also focus on the maximisation of the runway throughput (Anagnostakis and Clarke 2003, Atkin et al. 2006, 2007, 2008, Chandran and Balakrishnan 2007, Li et al. 2009, Balakrishnan and Chandran 2010, Gupta et al. 2010a, Apice et al. 2014, Benlic et al. 2016).

Another objective that is considered is the reduction of the total delay of aircraft by taking into account the delays of all of the aircraft in order to minimise their total delay (Atkin et al. 2006, Gupta et al. 2010a, 2010b, Weiszer 2015c).

In some papers there might be specific time slots that need to be met (Leese et al. 2001, Karapetyan et al. 2017).

Finally, resequencing to avoid unfairness is considered by some papers that constrain the maximum position shift the aircraft can have (Psaraftis 1980, Balakrishnan and Chandran 2010) or a penalty is applied when a reference schedule is not followed (Beasley et al. 2004).

2.6.3 Previous work on runway scheduling

Bennell et al. (2011) present the different solution approaches that can be used to solve the runway optimization problem. The key methodologies that were identified are heuristics and meta-heuristics, dynamic programming and branch and bound techniques.

Atkin et al. (2007) propose a hybrid meta-heuristic model that supports air controllers by taking more departing aircraft into consideration. Heuristics are also used in optimising the runway scheduling process (Ernst et al. 1999, Bianco et al. 1999, Beasley et al. 2004, Soomer and Koole 2008).

Several papers use dynamic programming models (Psaraftis 1980, Leese et al. 2001, Chandran and Balakrishnan 2007, Li et al. 2009, Balakrishnan and Chandran 2010, De Maere and Atkin 2015). A dynamic programming algorithm which re-sequences the departures is suggested by Leese et al. (2001). The model uses holding points for aircraft that are already sequenced in order to provide an even more effective runway sequence.

Dynamic programming and backwards induction is suggested by Li et al. (2009). A recent research of De Maere and Atkin (2015) proposes a pruned dynamic programming model for the solution of the runway scheduling problem for arrivals and departures that takes into consideration different cost compositions that each aircraft can have. The results show that the model can produce optimal runway sequences for aircraft in short computation time.

Other techniques have been used to solve the runway sequencing problem for either departures or arrivals. Gupta et al. (2010a, 2010b) use MILP model to sequence departing aircraft. Tavakkoli et al. (2012) investigate arrival sequences only, using a fuzzy programming solution method and the objective function aims to reduce the deviation cost from the scheduled time. Apice et al. (2014) attempt to solve the departures sequencing problem using a model that reduces delays and increases the capacity of the runway to help air traffic controllers in airports with handling multiple runways. A two-stage algorithm is used and an optimal departure schedule is produced.

Anagnostakis and Clarke (2003) also propose a two-stage algorithm for optimal runway scheduling. Karapetyan et al. (2017) suggested the use of a user-friendly Pre-Departure Sequencer (PDS) which is created for medium sized airports. The PDS is designed to be part of a larger scale system and the researchers focused on creating a system that is clear regarding its operation and its decision making.

2.6.4 Integration with Ground Movement

Runway sequencing can also be considered as a ground traffic optimization aspect (Gotteland et al. 2001, Deau et al. 2008, 2009, Anderson and Milutinović 2013, Weiszer et al. 2015c,). While the ground movement solution methods usually focus on minimising the total taxi time of the aircraft, most integrated research focus on guaranteeing that the aircraft are at the

runway at a specific time (Rathinam et al. 2008, Keith et al. 2008, Clare et al. 2009a, Clare and Richards 2009b, 2011, Weiszer et al. 2015c).

Separation rules that are taken into consideration are mostly connected to wake vortex, however, en-route separations are also considered (Keith et al. 2008, Clare et al. 2009a, Clare and Richards 2009b, 2011, Lee and Balakrishnan 2012, Weiszer et al. 2015c, Benlic et al. 2016).

Some of the models that consider or integrate with the ground movement process are summarised below.

Atkin et al. (2006) suggest a ground controller support system that organises the sequence of the aircraft before they take off. The relationship between moving the planning horizon and the frozen take-off order as well as changes on it is investigated. The model uses a tabu search that provides an ideal take-off schedule. Then the paths close to the runway are defined and the holding point model plans the possible overtaking. Finally, the model evaluates the cost of the predicted departure times.

A meta-heuristic solution approach that provides optimized departure sequences is proposed by Atkin et al. (2008) that takes part of the ground movement process into consideration. The research considers three different separation rules in order to increase the capacity of the runways. The first separation rule is the "wake vortex" which is linked to the weight class of the aircraft, the second separation rule is applied when aircraft follow similar routes and the third rule considers speeds of aircraft in order to adjust separation between them. A holding point network before the runway is modelled in order to provide achievable take-off sequences.

Keith et al. (2008) suggest a MILP model to solve optimally a combination of the runway scheduling and routing of the aircraft while satisfying the separation constraints. Clare et al. (2009a) extend this research by using a receding horizon in order to improve the execution times. Moreover, Clare and Richards (2009b, 2011) propose models for runway sequencing using a MILP model in order to ensure that the aircraft arrive at the runway on time (see Section 2.3.2).

Runway sequencing is considered by Deau et al. (2008, 2009) as a ground traffic optimization aspect. Their research uses a ground traffic simulator to model and discuss the runway sequencing problem and the way the sequences can be implemented on the runway.

The models of Lee and Balakrishnan (2012) and Ravizza et al. (2014) hold the aircraft at the stand in order to reduce the waiting time at the runway and to make sure that the aircraft

arrive at the runway at the right time taking into consideration the ground movement of other aircraft and that they arrive at the runway at the correct sequence.

Weiszer et al. (2015c) introduce a runway scheduling model for arrivals and departures that is based on an evolutionary algorithm. The purpose of the research is to find the ideal arrival and departure times of the flights in order to reduce runway delays and fuel burn by optimising the speed profiles of the aircraft and by minimising the waiting time at the runway. However, only wake-vortex separations are considered in this paper. Benlic et al. (2016) also introduce a runway scheduling problem that sequences arrivals and departures while considering the aircraft ground movement. The aim of the model is to increase the runway throughput and reduce the total taxi times of aircraft using a Receding Horizon Control based Iterated Local Search (RHC-ILS) technique for tackling the runway sequencing problem.

2.7 Taxi time prediction

Taxi time prediction is another airport operation that should be considered when dealing with the ground movement problem. Accurate time prediction will help in better prediction of take-off times as well as more accurate prediction of arrival times (Idris et al. 2002). Also, taxi time prediction can be useful as it can be used for later engine start-up in order to achieve a decrease in fuel consumption (Atkin et al. 2011). Moreover, Simaiakis and Balakrishnan (2010) and Balakrishna et al. (2008) highlight the importance of accuracy in taxi-out prediction as a main factor in fuel burn, emissions and cost management.

Rappaport et al. (2009) examine the impact that speed reduction of turning aircraft has on taxi time since aircraft move faster when they travel straight. Moreover, Atkins et al. (2008) mention that taxi time around corners while taxiing can differ significantly.

Pina and De Pablo (2005) propose a taxi planner prototype which is based on a mathematical model that optimises the ground movement problem. Apart from estimating the ideal route for the aircraft the model also predicts its taxi time. In this model taxi times are considered as constants and they are estimated considering parameters like the weather, the runway that is being used, the type of the aircraft, the stand location or the traffic. The research shows that statistical analysis can help in reliable taxi time prediction.

Tu et al. (2008) propose a model that estimates delays for departing aircraft using seasonal and daily delay patterns to represent the distribution of delays. Various kinds of delays, such as airport congestion and weather conditions, are studied together as a large group. The main characteristic of the model is the calculation of the delay propagation effect (delays

accumulated by previous flights) which is used to estimate the congestion levels at the airport and future delays for departures.

Simaiakis and Balakrishnan (2010) carry out a research in order to analyse how congestion affects taxi time, fuel burn and emissions. They suggest that congestion at the airports occurs because of multiple departures that happen at an airport and they use metrics that: take into account the number of flights; compare the actual and the free-flow taxi-out times; and evaluate them according to the airport throughput.

Furthermore, linear regression models have been used to estimate taxi time (Kistler and Gupta 2009, Jordan et al. 2010, Atkin et al. 2011 Ravizza et al. 2013a). A statistical learning approach, which chooses the most important variables from a subset, is used to model taxi time by Jordan et al. (2010). The approach uses data from Dallas/Fort Worth International Airport and it includes both taxi-out and taxi-in times. Information from the same airport are used by Kistler and Gupta (2009) to represent the taxi time and traffic interaction. Atkin et al. (2011) examine a combined ground movement system and a statistical multiple linear regression model to analyse the predictability of taxi times at London Heathrow airport. A taxi time prediction model for arriving and departing aircraft is suggested by Ravizza et al. (2013a) who also study the key parameters that lead to changes in taxi times (taxi distance, turning angle, arriving and departing aircraft speed, traffic at the airport). However, Chen et al. (2011b) suggest the use of Fuzzy Rule-Based Systems (FRBSs) as opposed to linear regression methods in order to enhance the precision of taxi time estimation models.

Idris et al. 2002 examine the basic elements that affect the taxi-out time and consider the size of the departure queue (take-off queue) as the most crucial point for more accurate estimation. More specifically, the queue size is defined as the number of take-offs that happen between an aircraft's pushback and take-off time. The model uses the number of aircraft that are about to depart and measures the size of the take-off queue by estimating the passing of the aircraft while taxiing-out. However, the suggested model only focuses on departing aircraft taxi time. The size of the departure queue is also considered for the prediction of taxi-out time by Simaiakis and Balakrishnan (2009). They suggest a queuing model for departures that decreases taxi time in order to reduce emissions.

Taxi-out time prediction is the subject of some other studies. Balakrishna et al. (2008, 2009, 2010) suggest taxi-out estimation that is based on stochastic dynamic programming and uses the strategy of Reinforcement Learning (RL). Balakrishna et al. 2008) propose an average taxi-out time estimation method that was tested on John F. Kennedy International airport. The model estimates the taxi-out time 30-60 minutes prior to take-off and seems to be a flexible model that can adjust to the constantly changing departure operations. Balakrishna et al. (2010) is the first research that applies a nonparametric reinforcement learning method and

artificial intelligence, which is based on stochastic dynamic programming, in order to estimate taxi-out time on the departures of Tampa International Airport. An approximate dynamic programming model that is based on RL is also used by Ganesan et al. (2010).

Clelow et al. (2010) examine departures at John F. Kennedy International Airport and Boston Logan International Airport to study the elements that affect how long taxi-out procedures last. The number of arrivals and departures, the runway configuration, the weather, and the terminal where the aircraft is allocated, are considered to be the main factors that affect the taxi-out time duration but with the number of aircraft that arrive and depart being the most important parameters. This is the first approach that considers that the number of arriving aircraft has an impact on taxi-out time.

Srivastava (2011) proposes a taxi-out time prediction system that is based on historical data on traffic flow. In particular, the model digitizes the surface of the airport as logical areas (taxiways, terminals, runways, queues) and then it defines the location of the aircraft. The variables that are taken into consideration are the position of the aircraft at the queue, the runway distance, the arrival and departure rates, and weather. However, the estimations of the model are short term and it has to be readjusted whenever the traffic plan changes.

2.8 Conclusions

Finding the optimal solution to the aircraft ground movement problem - as well as to the gate allocation problem and runway sequencing problem - is a very difficult task. The majority of the papers that have been reviewed suggest solutions that solve different problems that affect each other, but do not provide an optimal solution for the overall problem.

There is no doubt that an optimal solution for the integrated problem will be an asset for the airports as they can save money by reducing fuel consumption and delays. Furthermore, it may increase the capacity of the airports and make them more passenger-friendly. All these improvements can provide a more efficient and environmentally friendly airport.

Building a model that integrates more than one airport problem is becoming more and more popular as the execution time for many methods is improved and the results get closer to optimal solution. Having a more integrated solution and considering these problems separately from the theoretical level and closer to a practical implementation is necessary in order to design effective systems.

3

The QPPTW algorithm, Datasets and Airports

3.1 Introduction

This chapter introduces the routing algorithm which was used as a starting point for various extensions that will be presented in many of the following chapters. Moreover, this chapter reviews the datasets, as well as the airports that have been used for performing the experiments in this thesis.

The majority of the experiments that were executed for this research made use of the Quickest Path Problem with Time Windows (QPPTW) algorithm. The algorithm has been developed for solving the ground movement process of aircraft and can find the quickest path for each aircraft by taking into consideration previously routed aircraft. In the next section (Section 3.2), the key concepts of the QPPTW algorithm are introduced and a pseudocode of the algorithm is presented and described.

The experiments in this thesis were executed using datasets from three different international airports. These airports are Stockholm Arlanda Airport, Zurich Airport and Manchester Airport. As the aim of this research is to increase the real-world applicability of ground movement models in airports, it is important to make use of real data when running the experiments. Using real data from airports makes it possible to run experiments in more realistic scenarios and draw more accurate conclusions. In Section 3.3 the datasets and the layouts that were used for the experiments are presented.

Finally, the conclusions are summarised in Section 3.4.

3.2 The QPPTW Algorithm

QPPTW is a routing algorithm that was based on Dijkstra's algorithm (Dijkstra 1959) and was developed by Gawrilow et al. (2008) and later modified by Ravizza et al. (2014) to make it more suitable for airports. The algorithm is fully explained in the PhD thesis of Stenzel (Stenzel 2008) and the PhD thesis of Ravizza et al. (2013c), so only a summary of its operations will be presented here, sufficient to understand the key processes and principles in the previous work, the modifications which have been made, and the contributions of the current work.

In order to maintain a thread of continuity with the original QPPTW algorithm and to make it easier for the reader to better understand additions that were made in the following chapters, the notation of variables that was used below are the same or similar to the notation that was used in the PhD thesis of Stenzel (2008) and Ravizza (2013c). For the same reason, the pseudocode that is presented later in this section (Algorithm 3.1) is also the same as the one used in the PhD thesis of Ravizza (2013c).

3.2.1 Introduction of the QPPTW Algorithm and Benefits of Using this Methodology

In summary, the QPPTW algorithm is a deterministic algorithm that finds the quickest path for each aircraft in turn, taking into consideration aircraft that have been previously routed. When there are no delays this will be the shortest path, but when some parts of the shortest path are already being used (blocked) by other aircraft at that time, then the quickest path can involve either taking an alternative route (around the blockage) or waiting for the blocking aircraft to move. The algorithm is implemented to iteratively expand out from the starting vertex to all neighbouring vertices, like Dijkstra's algorithm, but with time windows denoting when the edges are already in use. The weight of each edge is normally the time that it takes to traverse the edge, however where the vertex is occupied this edge weight is increased by the time that it will take for the other aircraft to finish using that vertex. It should be noted that since there may be multiple time windows for which an edge is occupied, there also may be multiple gaps which could be utilised by other aircraft, so the expansion has to consider not only utilising the edge after other previously routed aircraft, but also whether gaps can be utilised by new aircraft.

The QPPTW algorithm has been chosen for this research for a variety of reasons. First it is able to take into consideration multiple paths, which is an important aspect, when there are many arriving and departing aircraft that move at the airport at the same time. Chapter 4

(Section 4.6), where the QPPTW algorithm is compared with a MILP based methodology, shows the importance of being able to take multiple taxiways into consideration when solving the routing process. Moreover, the research in this thesis is not limited to the current state of traffic in airports. Optimising the ground movement of aircraft can decrease the delays and increase the number of aircraft that can move at an airport at the same time. This, however, means that aircraft will have to utilise any of the available taxiways and not only focus on using a few short paths. The importance of multiple taxiways in an airport is highlighted in Chapter 6. A compact airport with many aircraft moving at the same time is shown to result in shorter taxi times, than a sparser airport with large taxiways, due to taking into consideration multiple taxiways.

Another benefit of the QPPTW algorithm is that it can route aircraft fast, allowing for integration with other processes. Some processes that happen in an airport can affect the routing process, such as the gate allocation process (where aircraft are allocated to gates) or the runway sequencing process (where the sequence of aircraft that take-off at the runway is scheduled). Integrating these processes with the routing process can be computationally expensive, making it impossible much of the time to find a solution within a reasonable timeframe. Having a fast routing process that can be invoked multiple times by other processes is essential in any integrated model. This makes the QPPTW algorithm ideal for implementing with other processes such as the gate allocation process, which is investigated in Chapter 7.

3.2.2 Notation and Definitions

Table 3.1 shows the definitions of the notation that was used to describe the QPPTW algorithm (Algorithm 3.1).

Table 3.1: Table of definitions for the QPPTW algorithm

E	The set of all edges
V	The set of all vertices
$e \in E$	An edge
$v \in V$	A vertex
$G = (V, E)$	The directed graph representing the airport layout, with vertices $v \in V$ and edges $e \in E$
a_e^j	The start time of the j^{th} time-window on edge $e \in E$
b_e^j	The end time of the j^{th} time-window on edge $e \in E$
$F_e^j = [a_e^j, b_e^j]$	j^{th} time-window on edge $e \in E$, from time a_e^j to time b_e^j

$\mathcal{F}(e)$	The sorted set of all of the time-windows on edge $e \in E$
H	The Fibonacci heap storing the added labels
a_L	The start time of Label L
b_L	The end time of Label L
$I_L = [a_L, b_L]$	The time interval used in a label L
$pred_L$	The predecessor label of label L
$L = (v_L, I_L, pred_L)$	A label on vertex $v_L \in V$ with a time interval I_L and predecessor label $pred_L$
$\mathcal{L}(v)$	The set of all of the labels at vertex $v \in V$
R	A conflict-free route that is being generated
$s \in V$	A source vertex
$t \in V$	A target vertex
$time$	The time that an aircraft sets off
p	The pushback duration
$T = (s, t, time, p)$	A taxi request to route, from source $s \in V$ setting off at time $time$, to target $t \in V$ and with pushback duration p (for departures)
w_e	The weight (necessary taxi time) of edge $e \in E$
$H.getMin()$	A function that returns the element with the lowest value in heap H
$Maximise(a, b)$	A function that returns the element with the largest value between elements a and b
$head(e)$	A function that returns the vertex y of an edge e that is directed from vertex x to vertex y .

3.2.3 Definitions of Key Concepts

The expansion steps of the QPPTW algorithm are similar to Dijkstra's algorithm. However, the QPPTW algorithm can be expanded several times to the same vertex as it can use different time-windows. In order to explain the algorithm, it is necessary to define the following concepts.

Firstly, the time windows are used in order to provide the availability of each edge.

Definition: Set of sorted time-windows

The sorted set of time windows $\mathcal{F}(e)$ contains the time intervals $F_e^j = [a_e^j, b_e^j]$ for edge e that identify the time gaps that edge e can be used by an aircraft. The times that edge e is occupied by aircraft that have been previously routed are therefore excluded. This set is defined for each edge and all of them are used as an input every time an aircraft is routed.

The algorithm uses labels to expand to a vertex.

Definition: Label

A label $L = (v_L, I_L, pred_L)$ consists of the vertex v_L that the current aircraft considers expanding to, the time interval $I_L = [a_L, b_L]$ that it would take for the current aircraft to reach vertex v_L , and a reference to the label where label L has expanded from ($pred_L$). The vertices of each predecessor label ($pred_L$) can be used to form a route, and the time intervals I_L denotes the traversal times for each edge along the route.

A dominance check between labels takes place, in order to remove the labels that are dominated.

Definition: Dominance

A label $L' = (v_{L'}, I_{L'}, pred_{L'})$ is dominated by another label $L = (v_L, I_L, pred_L)$ on vertex $v_{L'} = v_L$ if and only if $I_{L'} \subseteq I_L$. This means that the following happens: $a_L \leq a_{L'}$ and $b_L \geq b_{L'}$.

Once the quickest path is found, the time-windows are adjusted based on the time intervals that the edges were used, and the algorithm is executed again for the next aircraft.

3.2.4 Explaining the QPPTW Algorithm

The QPPTW algorithm requires as an input the graph $G = (V, E)$ (see figures 3.1, 3.3 and 3.4), with each edge having its own weight function w_e . Each edge has its own set of time windows $\mathcal{F}(e)$ that provides the availability of the edge. The algorithm also requires the details of the flight i that will route $T_i = (s_i, t_i, time_i, p_i)$ and returns the quickest available path R that aircraft i can use to go from vertex s_i to t_i for arrivals (or from vertex t_i to s_i for departures since they are routed backwards) by respecting the previously routed aircraft (time-windows).

The pseudocode of the QPPTW algorithm is provided in Algorithm 3.1. The algorithm is sorting the labels that are generated in a Fibonacci heap H (see Table 3.2) that is initialised in line 1 of Algorithm 3.1. A Fibonacci heap is a data structure where data can be stored, and it is used for retrieving the element with the smallest value in a short amount of time. Using a Fibonacci heap provides the same benefits as it does for Dijkstra's algorithm regarding the running time of Algorithm 3.1. The labels are stored in a list \mathcal{L} that is used for referencing the previous vertex that a label has expanded from and is initialised in line 2. This list is used for accessing the labels for the dominance check that happens in lines 23-29 and for building the route starting from the latest label that has reached the target (or source) vertex all the way back to the source (or target) vertex by finding the previous label of each label when

reconstructing the route in line 9. The first label that uses the source vertex is generated in line 3 and is inserted into the Fibonacci heap and into the label list in lines 4 and 5. When a label is inserted in the Fibonacci heap it is sorted based on the key that the label was added, which is the earliest arrival time to that vertex.

If the queue is empty (line 6) at this point it means that there is not a new vertex for the algorithm to expand to, which means that there is no available path to reach the target vertex (line 32). If the Fibonacci heap is not empty, the label with the smallest key (line 7) is pulled.

If the vertex of the pulled label is the target vertex (line 8), the route is formed (lines 9-10), otherwise the algorithm expands to the available vertices. The time-windows of the outgoing edges are examined (lines 11-12). The algorithm checks if the time interval of the time-window of each edge is sufficient for the algorithm to expand to the vertex at the end of the edge (lines 14 and 16).

The earliest time that the edge under consideration can be exited is found in lines 18-19. If the time-window of the edge is wide enough for expanding towards the next vertex (line 20) a new label is generated (lines 22).

The algorithm then checks if the new label is dominated by previous labels and if so, it is not considered (lines 25-26) or if the new label dominates a previous label, then the dominated label is removed from the Fibonacci heap (lines 27-29). Finally, the new label is added in the Fibonacci heap with the key being the time that the aircraft arrives at the vertex where the algorithm has expanded (line 30).

Since the label with the smallest key is always pulled by the Fibonacci heap, this means that the time that the aircraft arrives at the vertex of the label under consideration is always going to be the earliest time that the aircraft can arrive at that vertex. If that vertex is the target vertex, this means that the algorithm has found the fastest way to reach this vertex and the route that is formed is the quickest path from source to target vertex.

The presented algorithm has been enhanced and altered for the purposes of this research and new versions as well as the original algorithm have been used for the routing process of aircraft in airports in the chapters to follow.

Algorithm 3.1: Quickest Path Problem with Time Windows (QPPTW)

Input: Graph $G = (V, E)$ with weights w_e for all $e \in E$, the set of sorted time-windows $\mathcal{F}(e)$ for all $e \in E$, a taxi request $T_i = (s_i, t_i, time_i, p_i)$ for aircraft i , with the source vertex $s_i \in V$, the target vertex $t_i \in V$ and the start time $time_i$.

Output: Conflict-free route R from s_i to t_i with minimal taxi time that starts at the earliest at time $time_i$, respects the given time-windows $\mathcal{F}(e)$ or returns the message that no such route exists.

```
1  Let  $H = \emptyset$ 
2  Let  $\mathcal{L}(v) = \emptyset \quad \forall v \in V$ 
3  Create new label  $L$  such that  $L = (s_i, [time_i, \infty), nil)$ 
4  Insert  $L$  into heap  $H$  with key  $time_i$ 
5  Insert  $L$  into set  $\mathcal{L}(s_i)$ 
6  while  $H \neq \emptyset$  do
7      Let  $L = H.getMin()$ , where  $L = (v_L, I_L, pred_L)$  and  $I_L = [a_L, b_L]$ 
8      if  $v_L = t_i$  then
9          Reconstruct the route  $R$  from  $s_i$  to  $t_i$  by working backwards from  $L$ 
10         return the route  $R$ 
11     forall the outgoing edges  $e_L$  of  $v_L$  do
12         foreach  $F_{e_L}^j \in \mathcal{F}(e_L)$ , where  $F_{e_L}^j = [a_{e_L}^j, b_{e_L}^j]$ , in increasing order of  $a_{e_L}^j$  do
13             /*Expand labels for edges where time intervals overlap*/
14             if  $a_{e_L}^j > b_L$  then
15                 goto 11 /*consider the next outgoing edge*/
16             if  $b_{e_L}^j < a_L$  then
17                 goto 12 /*consider the next time-window*/
18             Let  $time_{in} = \text{Maximise}(a_L, a_{e_L}^j) \quad /*a_{e_L}^j > a_L \Rightarrow \text{waiting}*/$ 
19             Let  $time_{out} = time_{in} + w_{e_L}$ 
20             if  $time_{out} \leq b_{e_L}^j$  then
21                 Let  $u = \text{head}(e_L)$ 
22                 Let  $L' = (u, [time_{out}, b_{e_L}^j], L)$ 
23                 /*dominance check*/
24                 foreach  $\hat{L} \in \mathcal{L}(v)$  do
25                     if  $\hat{L}$  dominates  $L'$  then
26                         goto 12 /*next time-window*/
27                     if  $L'$  dominates  $\hat{L}$  then
28                         Remove  $\hat{L}$  from  $H$ 
29                         Remove  $\hat{L}$  from  $\mathcal{L}(v)$ 
30                 Insert  $L'$  into heap  $H$  with key  $a_{L'}$ 
31                 Insert  $L'$  into set  $\mathcal{L}(v)$ 
32 return "there is no  $s_i - t_i$  route"
```

3.3 Airports and Graphs

In this section, the airports and graphs as well as the datasets that were used for the experiments in this thesis are presented. These airports were: Stockholm Arlanda Airport (located in Stockholm, Sweden), Zurich Airport (located in Zurich, Switzerland) and Manchester Airport (located in Manchester, England). It is important to mention that the datasets are not open for publication as a non-disclosure agreement for all of the datasets has been signed by the University of Nottingham and Sintef – a research centre in Oslo, Norway that is also involved with optimising the ground movement operations in airports (see Chapter 1 Section 1.6).

A graph model of each airport has been developed using nodes and edges. Since the QPPTW algorithm restricts the use of an edge to only one aircraft, the length of each edge has been limited in order to represent more reasonable distances (the maximum weight w_e of each edge e was limited to 20 seconds or less). This avoided blocking edges for an unrealistically long time, providing a more accurate model of the interaction between aircraft.

3.3.1 Zurich Airport

Zurich airport is the largest international airport in Switzerland and is the primary hub for Swiss International Air Lines. In 2016 the airport handled around 27.6 million passengers and 269,160 aircraft movements.

Figure 3.1 shows the developed graph model of the airport, as well as the location of the three terminals and the three runways. The stands and the areas where aircraft could park are marked by grey nodes.

The airport has three passenger terminals (Terminals A, B and E) and three runways (16/34, 14/32 and 10/28, where the numbers represent the magnetic azimuth of the runway's direction in decadegees). The airport uses the runways in three ways. The north concept uses runway 14 and 16 for landing and runway 28 and 16 (and runway 10 for north-easterly winds) for take-offs. The east concept uses runway 28 for landing and runways 32 and 34 for take-offs. The south concept uses runway 34 for landing and runways 32 and 34 (and partially runway 28) for take-offs. The north concept is used for the majority of the time during the day.

The data that was used for this airport consisted of the following information. The type of the flight (arrival or departure), the landing time or arrival time of the aircraft (depending on the type of the flight), the size of the aircraft, the runway where the aircraft land or take-off and the stands where the aircraft park or start their journey from. The above data was provided

for an entire week, from the 27th of June until the 3rd of July 2011. The dataset contains 5609 flight movements during this week. Due to the limited availability of real datasets that can be provided by airports, it was not possible to run the experiments with more recent data and with more datasets.

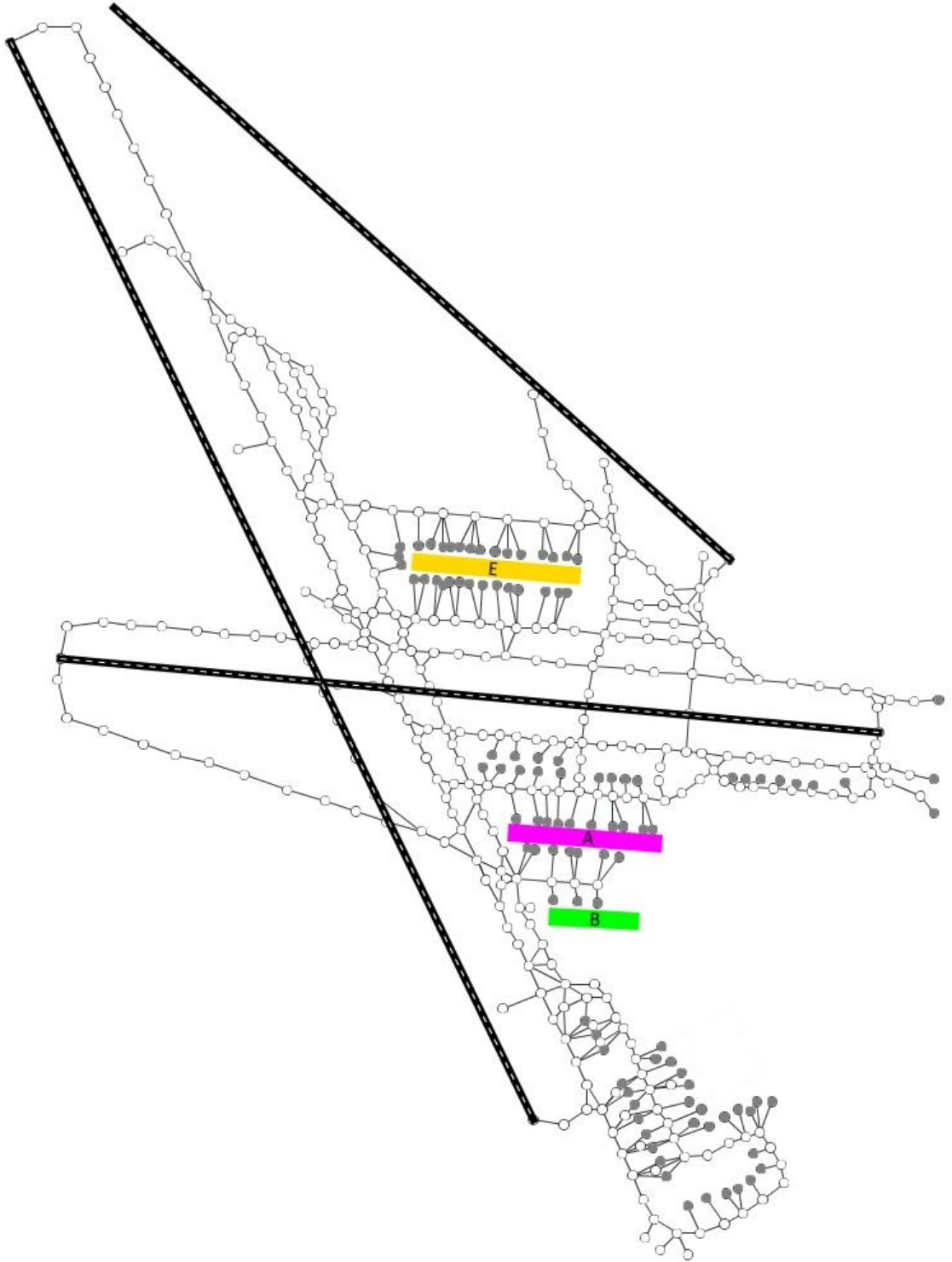


Figure 3.1: Model of Zurich airport as a graph with edges and nodes

The data about Zurich airport also provided the locations of the stands and the runways making it possible to build a graph with nodes and edges that would simulate the layout of the airport (see Figure 3.1). The developed graph model of the airport consists of 424 nodes (94 of which are stands) and 507 edges.

Figure 3.2 shows the number of aircraft movements for each hour during the day. Arrivals and departures are separated using different colours. As the graph shows, the airport is mainly busy between 7 a.m. and 9 p.m. with an exception during 2 p.m. to 4 p.m.

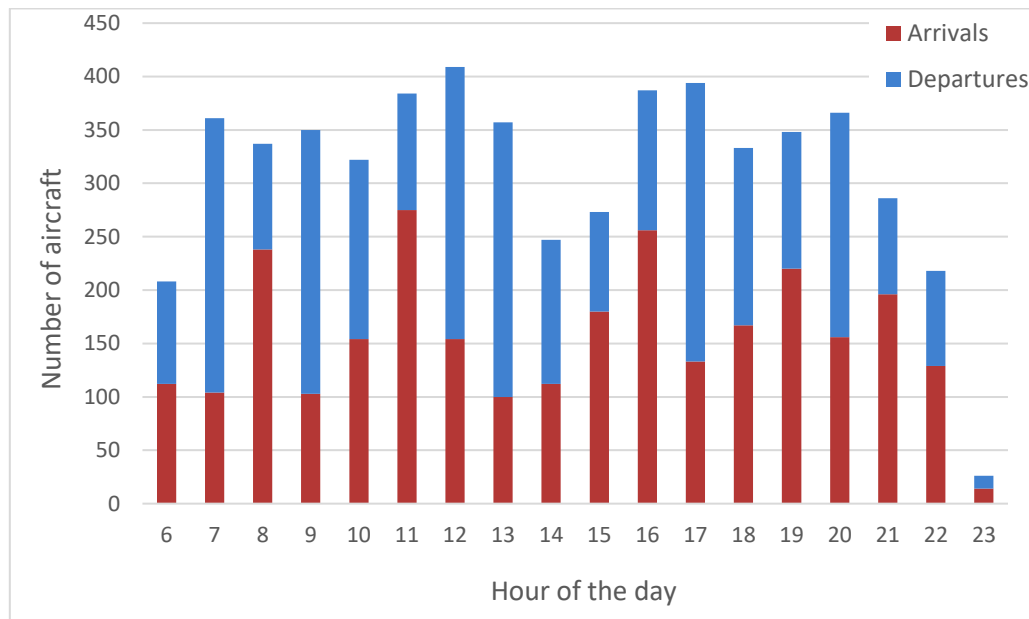


Figure 3.2: Traffic for different hours during the day in Zurich airport

3.3.2 Stockholm Arlanda Airport

Stockholm Arlanda Airport is the largest international airport in Sweden and is a major hub for Scandinavian Airlines and Norwegian Air Shuttle. In 2016 the airport handled 24.7million passengers.

Figure 3.3 shows the developed graph for Arlanda airport, as well as the location of the terminals and the runways. The stands and the areas where aircraft can park are marked by grey nodes.

The airport has four passenger terminals (Terminals 2, 3, 4 and 5) and three runways (01L/19R, 01R/19L and 08/26). Runways 01L/19R and 01R/19L are parallel runways and aircraft can take-off and land simultaneously at the same time using these runways. Runway 01L is usually used for departures and 01R (right side of the picture) is used for arrivals. Runway 26 (at the top of the picture) is newer and used in off-peak conditions.

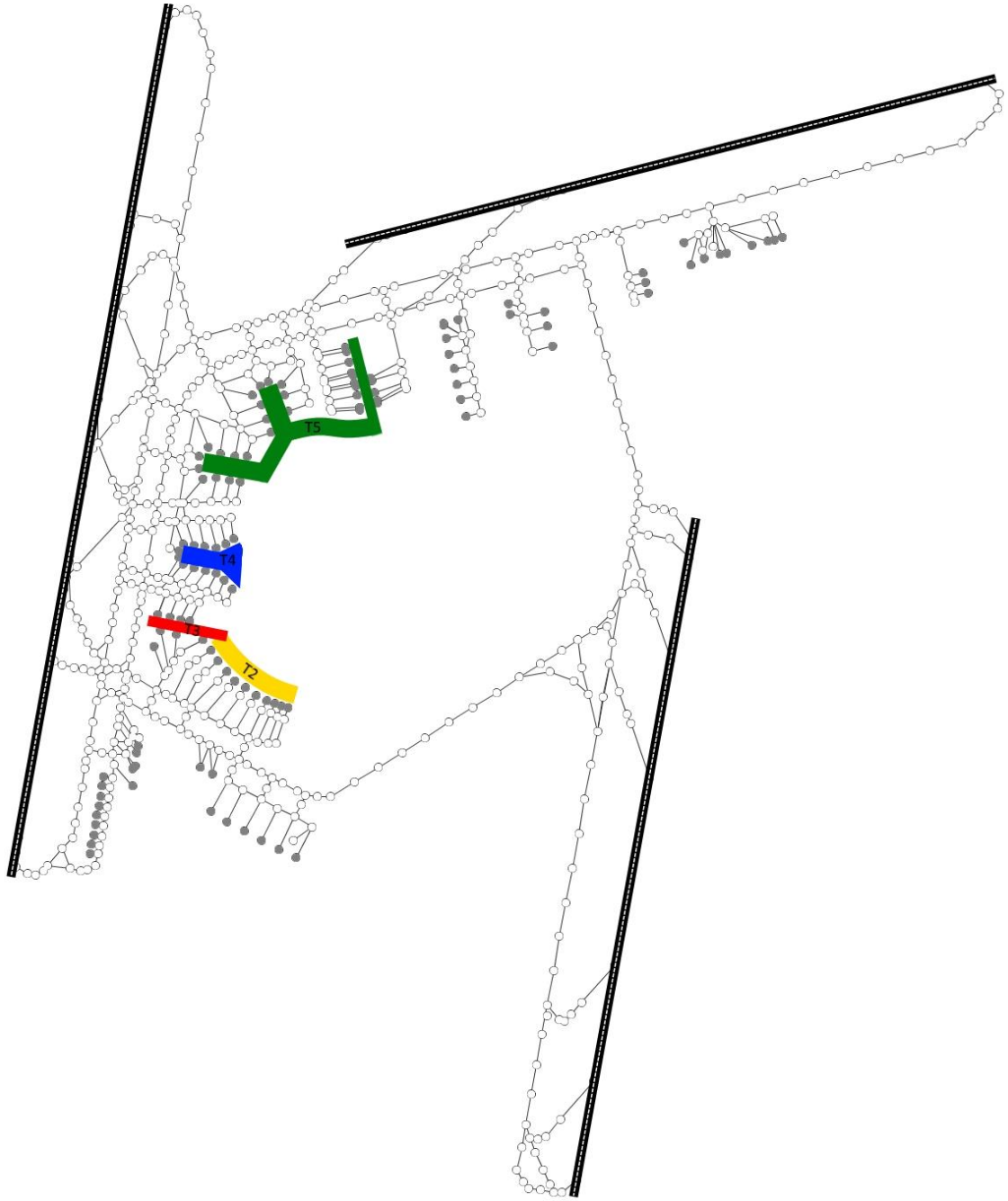


Figure 3.3: Model of Arlanda airport as a graph with edges and nodes

Two hours of historical data (from 5 a.m. to 7 a.m. on the 21st of May 2014) from the Swedish Air Navigation Service Provider (ANSP) were provided for this airport and were used as a basis for the instances that were created. The data that was used for this airport consisted of the following information: The take-off time of the aircraft, the size of the aircraft, the runway where the aircraft take-off and the stands where the aircraft start their journey from. For this airport, only data for arriving aircraft was available.

Table 3.2 shows the number of aircraft and the time span for each of the instances that were used for Arlanda airport. Instance 1 consists of the historical data and extra flights that were

added in order to increase the number of aircraft that interact with each other when the problem is solved. It includes 54 aircraft (44 from the original dataset) set to depart within 2 hours.

The remaining instances were developed to simulate the effects of heavier airport loads, by altering the original data (by adding more aircraft and assigning them to different stands), creating problems of increasing difficulty. These resulting instances therefore have different characteristics for sparsity of movements and complexity of solution.

Table 3.2: Specifications of instances

Instance No.	No. of Aircraft	Time span
1	54	2h
2	70	3h 40m
3	98	4h 50m
4	118	4h 50m
5	140	4h 50m

Further to the above instances, in Chapter 4, where the developed models are executed using departing aircraft, as well as in Chapter 6, where the two airport layouts are compared, data from Zurich airport has been utilised in order to be used in Arlanda airport. More details on how this data was applied to a different airport can be found in Chapter 6.

The locations of various taxiways, as well as the locations of the stands and the runways, were also provided by Sintef, making it possible to build a graph with nodes and edges that would simulate the layout of the airport. Figure 3.3 shows the developed graph, as well as the location of the four terminals and the three runways. The stands and the areas where aircraft could park are marked by grey nodes. The developed graph consists of 606 nodes (118 of which are stands) and 668 edges.

3.3.3 Manchester Airport

Manchester airport is the third busiest airport in the UK. In 2016 the airport handled 25.6 million passengers and 192,293 aircraft movements.

The airport has three passenger terminals (Terminals 1, 2 and 3) and 2 parallel runways next to each other (05L/23R and 05R/23L).

The historical data that was used for this airport are from the 28th of August 2011 to the 4th of September 2011 and consisted of the following information: The type of the flight (arrival or departure), the time that the aircraft will arrive to, or depart from, the stand (depending on the

type of the flight) and the size of the aircraft. In total, there are 21 datasets provided, each consisting of operations from 1 terminal for 1 day (7 days for 3 terminals). As it will be seen in Chapter 7, each terminal has been solved separately due to the more demanding execution time of the model that this dataset is used for.

The locations of nodes and the connections between them (edges) that would simulate the layout of Manchester airport was also provided. Figure 3.4 shows the developed graph, as well as the location of the three terminals and the runway. The developed graph consists of 113 nodes and 152 edges. In contrast to the graphs that were described earlier, this graph does not include nodes for the stands as for this airport the exact locations of the stands were not available. Moreover, since the two runways are next to each other and the second runway is accessible only through the first runway, to simplify the problem, this graph has only one of the runways modelled.

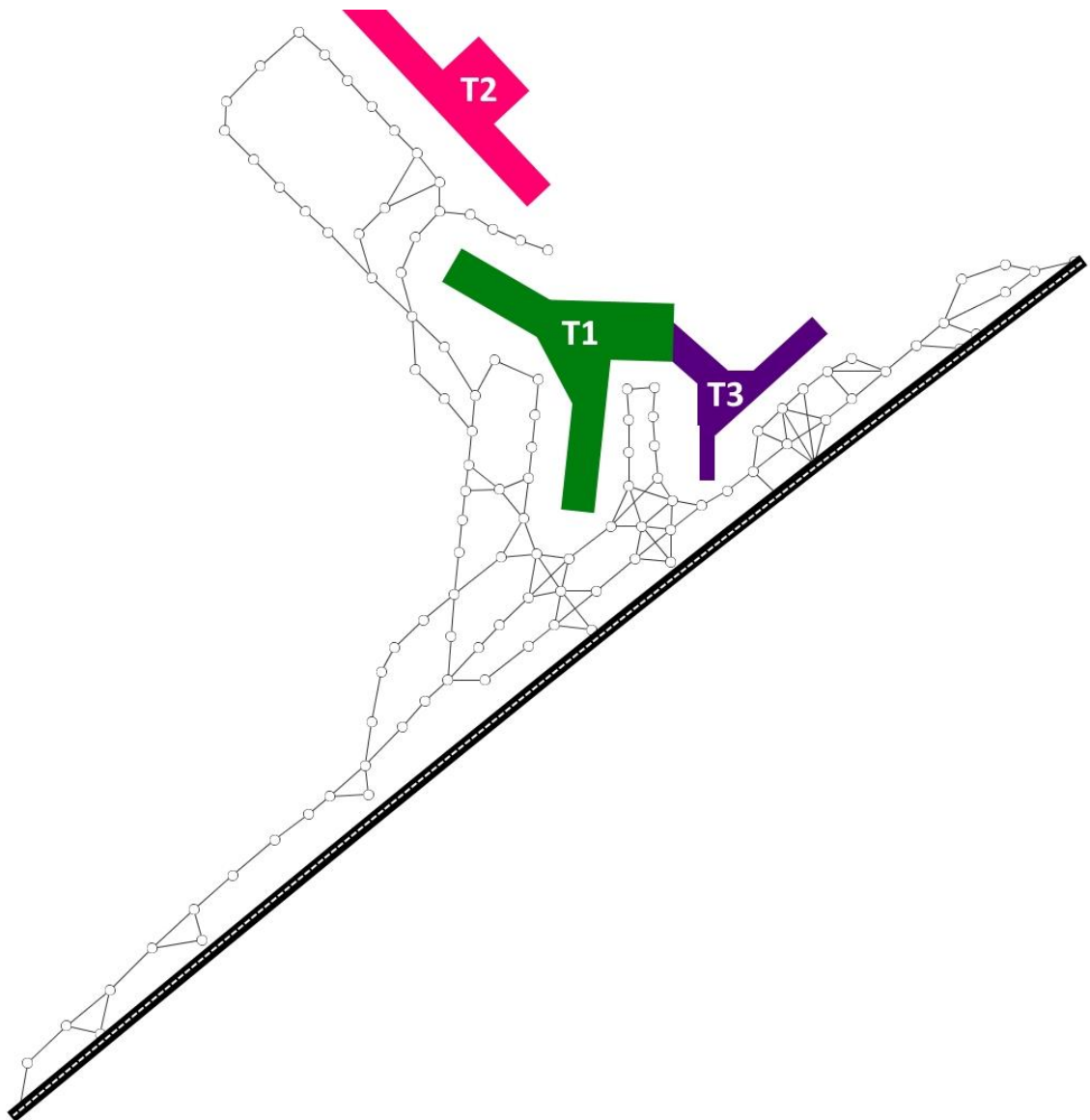


Figure 3.4: Model of Manchester airport as a graph with edges and nodes

Figure 3.5 shows the number of aircraft movements for each hour during the day. As the graph shows, the airport is mainly busy between 7 a.m. and 10 a.m. and between 5 p.m. and 8 p.m.

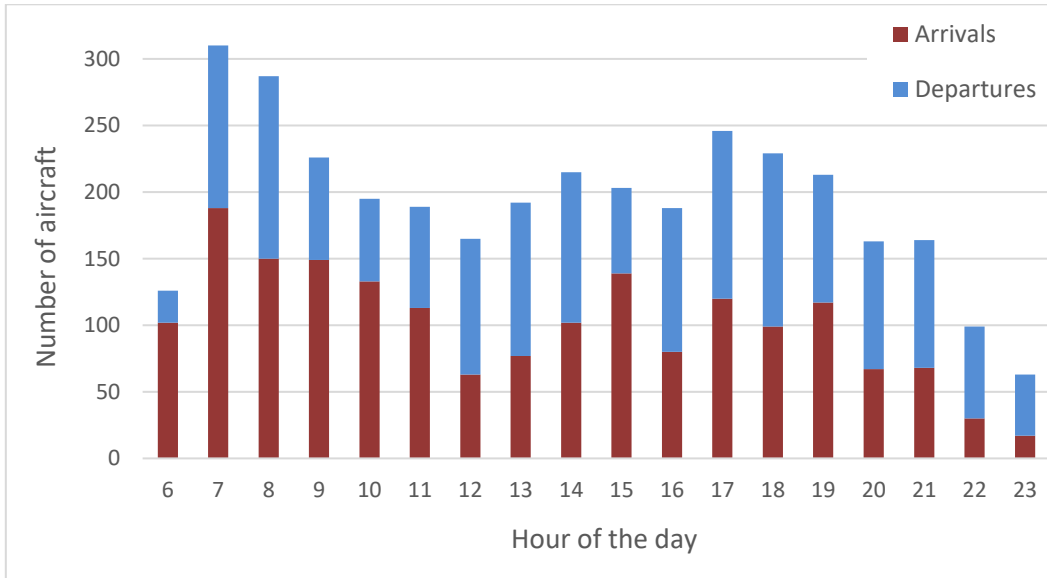


Figure 3.5: Traffic for different hours during the day in Manchester airport

3.4 Conclusions

In this chapter, the main algorithm that is used for solving this problem is presented. This algorithm has been extended and modified in various ways and used for solving the ground movement problem in each chapter.

Moreover, the details about the airports that have been utilised in this research are provided as well as the datasets that were used for them. The experiments in the following chapters use graph representations that model three airports (Zurich, Arlanda and Manchester) as well as information that is extracted from historic data (or based on historic data) from these airports.

Pushback Delays on the Airport Ground Movement Problem

4.1 Introduction

This chapter focuses on the pushback process of aircraft and the associated delays that can be caused to the routing process of aircraft. The pushback process (which is the part of the ground movement process where the aircraft pushes back from the stand to the taxiway and starts its engines) affects the routing process and it is important to have an integrated model where the pushback process is explicitly implemented. As mentioned in Chapter 2, most of the models do not explicitly implement the pushback process in the routing process and will either start routing the aircraft from the time that they have already pushed back or increase the departure time of the aircraft by a value based on estimations. However, the pushback process can affect or can be affected by other aircraft that are moving on the ground and including this process to the routing process is important for a model to be accurate.

This chapter aims to investigate and evaluate the effects upon the paths and schedules of explicitly taking into consideration the pushback process of aircraft. The effects of the pushback process are examined by implementing this process to two different routing methods. Considering two different routing methods and evaluating the differences between the delays when the pushback process delays are and are not explicitly considered, is important in order to quantify any accuracy benefits, such as improved predictions of delays.

In the next section (Section 4.2) the pushback process is introduced and in Section 4.3 an improved version of the QPPTW algorithm that includes the pushback process in the core of the algorithm will be presented. In Section 4.4 a MILP model for solving the routing process by taking into consideration the pushback process, will be presented. In Section 4.5, the

effects of the pushback delays are going to be examined, and how and in what extent they affect the routing process, by comparing different routing methodologies. Moreover, a comparison between the two methodologies is presented in Section 4.6. Finally, the overall conclusions are summarised in Section 4.7.

4.2 The Pushback Process

The pushback process is a crucial point where delays can (and do) happen. Aircraft usually need to push back from the terminals before starting their engines, since doing so near the terminals risks damaging the terminals. The pushback and engine start-up process is often a time-consuming process. While an aircraft is being pushed back and its engines are started, it can block other aircraft that are moving around the airport or it may not be able to push back if other aircraft are using stands that are close by. In cases where the taxi area around the gates is not wide enough to be simultaneously used by two aircraft, a taxiway may be blocked by the aircraft for the duration of the process. In summary, pushback operations for one aircraft can delay other aircraft. The reverse can also happen, where an aircraft may not be permitted to start the pushback process until another aircraft has passed. This is the case whenever the area that they would push back to, will not be free for the entire duration of the process. Figure 4.1 shows how delays can happen, illustrating how an aircraft pushing back would prevent another aircraft from passing, or an aircraft that is passing could prevent an aircraft from pushing back.

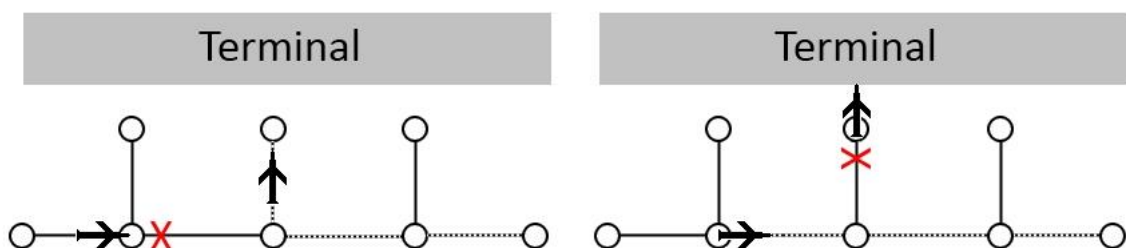


Figure 4.1: Causes of pushback delays, delaying other aircraft or the aircraft pushing back

Pushback delays can cause significant delays at airports and add uncertainty to the predicted position of an aircraft. The absence of consideration of these factors in a model can lead to further unpredicted delays later on down the path of an aircraft, since the delay in the routing process for one aircraft may cause knock-on interactions with later aircraft. For a busy airport, if not dealt with, these delays can propagate and negatively affect the overall throughput of the airport.

A take-off sequencing system would usually require knowledge of how early an aircraft can reach the runway, so any unpredicted delays, may affect the feasibility of potential take-off sequences, compromising the feasibility of these sequences. An accurate model for scheduling and routing aircraft on the ground is important for providing any automated decision support to improve runway operations.

4.2.1 Previous Research on Pushback Process

Although the ground movement problem has received significant research attention, there has been very little consideration of the pushback process.

The majority of the models that take into consideration the pushback process, usually focus on the delays that can happen close to the stands and try to avoid or minimise them (Cheng 1998, Neuman and Atkin 2013). In some cases, the pushback time and the delays that can happen during the pushback process are included in the total taxi time of an aircraft (Tu et al. 2008, Ravizza et al. 2013a). The added delays, however, are added to the total taxi time without attributing in which part of the trip exactly these delays happen.

Tu et al. (2008) attempted to identify the delays that happen during the routing process with the use of statistical analysis. They took into consideration a number of trends and patterns like weather impact, delay built up from previous flights, seasonal and daily patterns, in order to predict the difference between the scheduled time and actual time that an aircraft was going to start the pushback process. Neuman and Atkin (2013) attempted to find the conflicts that may occur because of the pushback process or the conflicts that happen close to the stands in order to better allocate aircraft to stands. Cheng (1998) developed a model that predicts and resolves conflicts on the taxiways close to the gates, in order to minimise the delay. Burgain et al. (2012) used a stochastic model of surface operations to control the pushback clearances based on the number of aircraft that are taxiing. However, these models do not explicitly examine the effects of the pushback process upon the ground movement, instead focusing on the minimisation of the total travel time and/or queuing time at the runway.

Ravizza et al. (2013a) used a model to predict the total delays for aircraft (at the stands or the runway) in order to absorb more of this time at the stand, before the pushback process of the aircraft commences. Consideration of the time which was needed to perform the pushback process and start the engines was an important element of the system. Furthermore, Ravizza et al. (2014) compared a number of approaches for predicting the total taxi time, but recognised that the real ground movement problem requires a better understanding of where delays actually occur as well as the total unimpeded delays.

4.3 Integrating the Pushback Process to the Routing Process

In this section, the implementation of the pushback process in the QPPTW algorithm is going to be presented. First the notation and definitions of the variables that were used are introduced. Then the way that the pushback process was added to the QPPTW algorithm (that was presented in Chapter 3) is described as well as the models that were developed for the experiments. Finally, other processes that are necessary for evaluating the effects of the pushback process are presented, such as the method for calculating the delays and finding the time that it takes for an aircraft to traverse its shortest path.

4.3.1 Notation and Definitions

All of the notation that is introduced in this section has been summarised in Table 4.1 below. In order to maintain a thread of continuity with the original QPPTW algorithm which the algorithm that will be later presented is based on and to make it easier for the reader to better understand the additions, the notation of common variables that was used below are the same or similar to the notation that was used in the PhD thesis of Ravizza (2013c).

Table 4.1: Table of definitions

E	The set of all edges
V	The set of all vertices
$e \in E$	An edge
$v \in V$	A vertex
$G = (V, E)$	The directed graph representing the airport layout, with vertices $v \in V$ and edges $e \in E$
a_e^j	The start time of the j^{th} time-window on edge $e \in E$
b_e^j	The end time of the j^{th} time-window on edge $e \in E$
$F_e^j = [a_e^j, b_e^j]$	j^{th} time-window on edge $e \in E$, from time a_e^j to time b_e^j
$\mathcal{F}(e)$	The sorted set of all of the time-windows on edge $e \in E$
H	The priority queue storing the added labels
a_L	The start time of Label L
b_L	The end time of Label L
$I_L = [a_L, b_L]$	The time interval used in a label L
$pred_L$	The predecessor label of label L

$L = (v_L, I_L, pred_L)$	A label on vertex $v_L \in V$ with a time interval I_L and predecessor label $pred_L$
$\mathcal{L}(v)$	The set of all of the labels at vertex $v \in V$
R	A conflict-free route that is being generated
$s \in V$	A source vertex
$t \in V$	A target vertex
$time$	The time that an aircraft sets off
p_f	The pushback duration for flight f
$T = (s, t, time, p)$	A taxi request to route, from source $s \in V$ setting off at time $time$, to target $t \in V$ and with pushback duration p (for departures)
w_e	The weight (necessary taxi time) of edge $e \in E$
FL	The set of all flights.
n	The total number of flights, $ FL $
$f \in FL := \{1, \dots, n\}$	A flight
t_f	The time at which flight f should commence its push back – i.e. the starting time for aircraft f in the datasets
$w_{f,i}$	The weight (necessary taxi time) of the i^{th} edge of flight f 's path, which connects the $(i-1)^{th}$ vertex of the path with the i^{th} vertex
m_f	The minimum time that it can take for an aircraft f to reach the runway from the stand
T_f	The time that it takes for aircraft f to reach its destination including any delays that might occur
N_f	The number of vertices on the allocated path for flight f
$v_{f,i}$	i^{th} vertex (i from 1 to N_f) on the allocated path for flight f . This determines the path that f takes through the taxiways.
$tr_{f,i}$	The time at which aircraft f arrives at the i^{th} vertex of its path (i from 1 to N_f).
$H.getMin()$	Function that returns the element with the lowest value in heap H
$Maximise(a, b)$	Function that returns the element with the largest value between elements a and b
$head(e)$	Function that returns the vertex y of an edge e that is directed from vertex x to vertex y .

4.3.2 The QPPTW Algorithm with Pushback Process

The QPPTW is a deterministic algorithm that finds the quickest path (between vertices in a graph) for each aircraft in turn, taking into consideration aircraft that have been previously

routed. When there are no delays this will be the shortest path, but when some parts of the shortest path are already being used (blocked) by other aircraft at that time, then a quickest path can involve either taking an alternative route (around the blockage) or waiting for the blocking aircraft to move. Full details of the basic algorithm can be found in Chapter 3.

In order to include the pushback process to the QPPTW algorithm some changes to the algorithm were necessary. Algorithm 4.1 shows the QPPTW algorithm (as it was presented in Chapter 3) with the addition of the pushback process. The parts that differ from the original algorithm have been underlined in red. In this implementation, the routing process of the aircraft is considered to initiate at the stand, before the aircraft is pushed back. For the first movement, the aircraft will move to the next vertex (on the taxiway) where it starts its engines. In this first movement, all of the neighbouring vertices are taken into consideration as the QPPTW algorithm would normally expand. However, for the first movement the duration is increased by the time that it takes to complete the pushback process and the engine start-up operation, and this becomes the new weight of that edge (see lines 19 and 20). All of the edges that the algorithm checks for the first move have been modified to have their weight increased by the pushback duration. This ensures that the pushback time is considered, but is also attributed to the beginning of the aircraft's journey (by the stands), where it actually happens.

Figure 4.2 shows an aircraft f_1 that is pushing back from vertex A to vertex B. The new weight of the edge AB w'_{AB} which is used by the modified algorithm can be calculated by $w'_{AB} = w_{AB} + p_f$, where w_{AB} is the normal weight of the edge. The pushback duration p_f is determined by the size of the aircraft f .

The QPPTW algorithm finds the quickest path, taking into consideration the added delay that is caused by the pushback process. All of the edges that are connected to the first edge are blocked, preventing other aircraft from coming too close to the aircraft which is pushing back. In the example in Figure 4.2, this means that all of the edges AB, BC, BD are blocked for the entire duration on the pushback process (w'_{AB}). Blocking the edges ensures that the aircraft will reach its destination in the shortest amount of time allowing for the fact that edges can be used by a maximum of one aircraft at a time.

Figure 4.2 also illustrates the situation where there is another aircraft f_2 that has to wait for aircraft f_1 to finish the pushback process. Aircraft such as f_2 that get blocked have to either wait or choose a longer path if there is one. The QPPTW algorithm which is used will ensure that the path is reallocated appropriately.

Algorithm 4.1: Quickest Path Problem with Time Windows (QPPTW) - Departures

Input: Graph $G = (V, E)$ with weights w_e for all $e \in E$, the set of sorted time-windows $\mathcal{F}(e)$ for all $e \in E$, a taxi request $T_i = (s_i, t_i, time_i, p_i)$ for aircraft i , with the source vertex $s_i \in V$, the target vertex $t_i \in V$ and the start time $time_i$.

Output: Conflict-free route R from s_i to t_i with minimal taxi time that starts at the earliest at time $time_i$, respects the given time-windows $\mathcal{F}(e)$ or returns the message that no such route exists.

```
1  Let  $H = \emptyset$ 
2  Let  $\mathcal{L}(v) = \emptyset \quad \forall v \in V$ 
3  Create new label  $L$  such that  $L = (s_i, [time_i, \infty), nil)$ 
4  Insert  $L$  into heap  $H$  with key  $time_i$ 
5  Insert  $L$  into set  $\mathcal{L}(s_i)$ 
6  while  $H \neq \emptyset$  do
7      Let  $L = H.getMin()$ , where  $L = (v_L, I_L, pred_L)$  and  $I_L = [a_L, b_L]$ 
8      if  $v_L = t_i$  then
9          Reconstruct the route  $R$  from  $s_i$  to  $t_i$  by working backwards from  $L$ 
10         return the route  $R$ 
11     forall the outgoing edges  $e_L$  of  $v_L$  do
12         foreach  $F_{e_L}^j \in \mathcal{F}(e_L)$ , where  $F_{e_L}^j = [a_{e_L}^j, b_{e_L}^j]$ , in increasing order of  $a_{e_L}^j$  do
13             /*Expand labels for edges where time intervals overlap*/
14             if  $a_{e_L}^j > b_L$  then
15                 goto 11 /*consider the next outgoing edge*/
16             if  $b_{e_L}^j < a_L$  then
17                 goto 12 /*consider the next time-window*/
18             Let  $time_{in} = \text{Maximise}(a_L, a_{e_L}^j) \quad /*a_{e_L}^j > a_L \Rightarrow \text{waiting}*/$ 
19             if  $a_L = time_i$  then
20                 Let  $time_{out} = time_{in} + w_{e_L} + p_i$ 
21             else
22                 Let  $time_{out} = time_{in} + w_{e_L}$ 
23             if  $time_{out} \leq b_{e_L}^j$  then
24                 Let  $u = \text{head}(e_L)$ 
25                 Let  $L' = (u, [time_{out}, b_{e_L}^j], L)$ 
26                 /*dominance check*/
27                 foreach  $\hat{L} \in \mathcal{L}(v)$  do
28                     if  $\hat{L}$  dominates  $L'$  then
29                         goto 12 /*next time-window*/
30                     if  $L'$  dominates  $\hat{L}$  then
31                         Remove  $\hat{L}$  from  $H$ 
32                         Remove  $\hat{L}$  from  $\mathcal{L}(v)$ 
33                 Insert  $L'$  into heap  $H$  with key  $a_L$ 
34                 Insert  $L'$  into set  $\mathcal{L}(v)$ 
35 return "there is no  $s_i - t_i$  route"
```

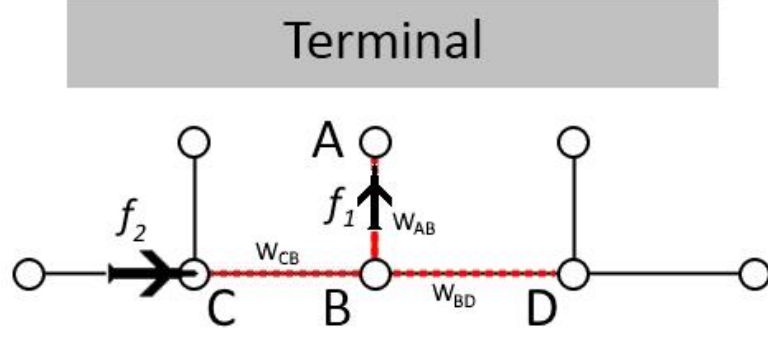


Figure 4.2: Blocked edges during pushback

4.3.3 Developed QPPTW Models

In order to investigate whether the pushback process is affecting the accuracy of the routing process when the QPPTW algorithm is used, two models were compared with each other. One includes the pushback process and the other one does not.

The first algorithm (Algorithm 1) is a typical implementation of the QPPTW algorithm, as described in Chapter 3 and routes aircraft without taking the blocking which can occur during the pushback process into consideration. In order to provide a fair comparison, rather than modelling the pushback delay by the stand, the algorithm instead delays the aircraft from setting off until the pushback duration has expired by delaying the start time of the operation. i.e. the start time for any aircraft f in Algorithm 1 is given by $t_f + p_f$. This is equivalent to assuming that the pushback and engine start-up operations will occur out of the way (at the stands) and will not delay any other aircraft while they occur. Again, to ensure a fair comparison, the calculated total taxi time is given by Equation 4.1 adding the pushback delay to the final taxi time for each aircraft.

$$Total\ routing\ time = \sum_{f=1}^n (T_f + p_f) \quad (4.1)$$

The second algorithm (Algorithm 2) is an extension of Algorithm 1, and includes the pushback duration at the start of the movement. This model uses the extension that was described in Section 4.3.2. In contrast to Algorithm 1, Algorithm 2 will start the routing process for aircraft f (which now includes the pushback process as an increased weight on the first edge) at time t_f and the final total taxi time will be determined by Equation 4.2, since the pushback delay has already been included in the taxi time.

$$Total\ routing\ time = \sum_{f=1}^n T_f \quad (4.2)$$

The total taxi time for an aircraft T_f will not only include the pushback duration for this aircraft p_f but will also include all of the delays that can be caused during the pushback process as well. These delays can be caused to an aircraft by its own pushback process (not being able to pushback immediately due to traffic) or the pushback process of other aircraft that are moving on the airport (where the pushback process of another aircraft blocks the path of this aircraft).

4.3.4 Calculating the Delays

In order to be able to compare the delays that take place in various configurations, all of the component delays need to be calculated. The total delay for a day is the sum of all of the delays that happen to all of the aircraft on that day. Since it is not possible to observe an obvious delay when an aircraft chooses a longer path (there may be no waiting time) the excess time is calculated by determining the minimum time without including the pushback time (see next subsection) that an aircraft would have needed to reach its destination and comparing this with the total taxi time of that aircraft.

After determining the total taxi time (T_f), the minimum taxi time (m_f) and the pushback duration (p_f) it is easy to find the delay (i.e. excess travel time) for each flight. Equation 4.3 shows how the sum of all of the delays that happen during a day are calculated using the notation defined in Table 4.1.

$$Total\ delay = \sum_{f=1}^n (T_f - m_f - p_f) \quad (4.3)$$

4.3.5 Finding the Minimum Taxi Time

The minimum time that an aircraft needs to reach its destination (without including the pushback time) is calculated separately for each aircraft movement, by routing the aircraft in an empty graph with the use of a tailored version of Dijkstra's algorithm. Equation 4.4 holds for all vertex times $tr_{f,i}$ on the path for f when calculating the shortest path. In normal operations, however, the time windows can cause delays for aircraft, so Inequality 4.5 instead holds for each vertex on the path for aircraft f .

$$tr_{f,i} = tr_{f,i-1} + w_{f,x,y} \forall i \in \{2, \dots, N\}, \text{ where vertex } x = v_{f,i-1} \text{ and vertex } y = v_{f,i} \quad (4.4)$$

$$tr_{f,i} \geq t_{f,i-1} + w_{f,x,y} \forall i \in \{2, \dots, N\}, \text{ where vertex } x = v_{f,i-1} \text{ and vertex } y = v_{f,i} \quad (4.5)$$

4.4 MILP Routing Model with the Pushback Process

In this section, a Mixed-Integer Linear Programming model for routing and scheduling aircraft will be presented. As mentioned in Chapter 3, the QPPTW algorithm (and as an extension the model that was presented in the previous section) routes the aircraft sequentially, based on the times that they start taxiing. However, as mentioned in Chapter 2, there are also many models that constrain the available paths and focus on optimising the sequence of the aircraft. For this reason, a MILP model, which does this, was created in order to investigate how the pushback process affects a model where the routes are predetermined but the schedule of aircraft movement on these routes (or parts of these routes) is variable.

4.4.1 Notation and Definitions

Table 4.2: Table of definitions for the MILP model

$s_{f,i}$	The time that flight f starts traveling towards the i^{th} vertex of its path, $i \in \{1..k_f\}$
C	The set of all identified conflicts where given the values of $s_{f,i}$ and the allocated paths, two flights would wish to use the same vertex at the same time
$c := \{f_1, f_2, i_1, i_2\} \in C$	A conflict between two flights f_1 and f_2 at a vertex, where the conflict vertex is the i_1^{th} vertex on the path for flight f_1 , and the i_2^{th} vertex on the path for flight f_2
$f_1(c)$	A function which will return the flight f_1 of the conflict c
$f_2(c)$	A function which will return the flight f_2 of the conflict c
$i_1(c)$	A function which will return the i_1^{th} vertex on the path for flight f_1 where conflict c happens
$i_2(c)$	A function which will return the i_2^{th} vertex on the path for flight f_2 where conflict c happens
k_f	The total number of vertices on the allocated path for flight f .
l	The number of vertices which two aircraft share after the vertex where they first conflict.

The notation that is introduced in this section is summarised in Table 4.2. The notation that was used in the previous section (Table 4.1) is still valid.

4.4.2 Developed Model

This model uses the shortest path for each aircraft and focuses on organising the sequence of the aircraft, when two aircraft need to use the same resource at the same time. It differs from the QPPTW algorithm as it focuses more on the sequence of the aircraft rather than finding alternative paths whenever there is a conflict.

The model makes use of the vertices that each aircraft traverses, instead of blocking the edges (as the QPPTW algorithm does). The time that flight f commences its journey to vertex i is denoted $s_{f,i}$. Only one aircraft can use any vertex v at any time, so a different aircraft can only use the vertex when the current aircraft uses the next vertex.

The model consists of 4 stages:

1. Find the shortest path for all of the aircraft, to determine the allocated paths (using the QPPTW algorithm).
2. Find all of the initial conflicts between aircraft (aircraft that will require the same vertex at the same time) and add constraints to resolve them.
3. Solve the LP model for all of the known conflicts and find new times (the LP model is explained below).
4. Identify any new conflicts and if there are any, add constraints to resolve them and then go to step 3.

In order to find the shortest path (as step 1 describes) the QPPTW algorithm is used to route each aircraft on an empty graph. The vertices that are used by each aircraft to form the shortest path will be used for many of the constraints that will be presented in MILP formulation.

For the 2nd step it is important to find all of the conflicts that happen when the aircraft use the shortest path that was found in the previous step. In order to identify a conflict, the movements of every aircraft are stored in each vertex every time that it is used. If any vertex is used by more than one aircraft at the same time, a conflict is added to the list of conflicts. Any conflict is between only 2 aircraft, although each aircraft can have multiple conflicts with other aircraft, and 2 aircraft can conflict with each other multiple times along their path.

A linear programming formulation is then used to solve the routing problem and determine the order in which aircraft will pass vertices where there is a conflict (step 3). The next

subsections (Section 4.4.3 and Section 4.4.4) describe the constraints and objective function of this formulation.

Where further conflicts are found, additional constraints are added to the model and it is resolved until no further conflicts exist (step 4).

4.4.3 Constraints

For the 3rd step of the developed model that was summarised above, a linear optimisation model is solved. The constraints for this model are shown below:

$$s_{f,1} \geq t_f \quad \forall f \in F \quad (4.6)$$

$$s_{f,2} \geq s_{f,1} + w_{f,1} + p_f \quad \forall f \in F \quad (4.7)$$

$$s_{f,i+1} \geq s_{f,i} + w_{f,i} \quad \forall i := \{2 \dots k_f - 1\} \quad \forall f \in F \quad (4.8)$$

Constraint 4.6 ensures that all aircraft start after their set start times. This time is allocated to each aircraft and the program is forbidden from making them start earlier.

Constraint 4.7 and Constraint 4.8 ensure that an aircraft cannot enter the next vertex on its path any earlier than the time at which it enters the current vertex, plus the time to traverse (i.e. the weight of) the edge between the two vertices. Constraint 4.7 ensures that the aircraft spends extra time on its first vertex to simulate the pushback operation (which will also delay the time at which any other aircraft can enter that vertex). Note that p_f will be 0 for the versions that do not include the pushback process.

If conflicts are found, additional constraints are added to resolve the conflicts, ensuring that one of the aircraft cannot use the vertex until the other has reached the following vertex. One of the disjunctive constraints (Constraint 4.9 or Constraint 4.10) will be added for each conflict.

$$s_{f_2(c),i_2(c)} \geq s_{f_1(c),i_1(c)+1} \quad \forall c \in C \quad (4.9)$$

$$s_{f_1(c),i_1(c)} \geq s_{f_2(c),i_2(c)+1} \quad \forall c \in C \quad (4.10)$$

For efficiency reasons and in order to avoid potentially conflicting constraints, these are actually applied to the next l vertices at the same time, where l is the number of vertices which the two aircraft share after the vertex where they first conflict. i.e. if they enter the shared path in a specific order, they must traverse all shared vertices in that order:

$$s_{f_2(c), i_2(c)+j} \geq s_{f_1(c), i_1(c)+j+1} \quad \forall j \in \{0 \dots l-1\} \quad \forall c \in C \quad (4.11)$$

$$s_{f_1(c), i_1(c)+j} \geq s_{f_2(c), i_2(c)+j+1} \quad \forall j \in \{0 \dots l-1\} \quad \forall c \in C \quad (4.12)$$

In order to have a better understanding of the types of delays that can happen during the pushback process, two variants of the model are evaluated. Instead of letting the MILP model to decide which aircraft will be prioritised during a conflict, this process is performed manually. Each variant is evaluated with and without including the pushback process. In the first variant (Model 1), the prioritisation constraints (Constraints 4.9 and 4.11) are applied in order to prioritise the aircraft that would reach the vertex first, even though the aircraft may use the vertices for different durations (e.g. aircraft which are pushing back will use the vertex for longer than aircraft which are going past). In the second variant (Model 2), this prioritisation is reversed (using constraints 4.10 and 4.12 instead of constraints 4.9 and 4.11), so priority will be given to the aircraft which starts moving towards the vertex second. Since the pushback operation is time consuming, this will usually be the one which is already taxiing rather than the one which is pushing back. This models what happens at real airports more often, since it is often better to avoid asking an aircraft which is already moving to stop. This latter approach also turns out to be more similar to the usual case for the QPPTW approach (which prioritises the aircraft which started moving first), since when an aircraft which is pushing back comes into contention with one which is already moving, the one which was already moving will almost always have commenced its pushback earlier (it has had to complete its pushback and taxi to the vertex where the problem occurs before it comes into contention with the aircraft which is pushing back).

4.4.4 Objective Function

$$\text{minimize} \sum_{f=1}^n s_{f, k_f} \quad (4.13)$$

The objective function (Equation 4.13) measures the times at which the aircraft reach the final vertices in their journeys, which is equivalent to the objective function for the first (QPPTW-based) method, allowing a comparison to be made between the two methods.

4.5 Comparison and Insights

In this section, the two methodologies of routing and scheduling aircraft that were presented in the previous sections are compared. Each methodology is implemented with and without

the pushback process in order to investigate if and how the pushback process affects the routing process. The reason two different methodologies were used is to verify that the effects of the pushback process that were observed in the results, are not unique to the methodology that is used for routing and scheduling the aircraft.

In summary, the first methodology, has a fixed sequence of aircraft and focuses on optimising the route based on the availability of the taxiways that can be partially occupied by previously routed aircraft. The second methodology is having a fixed route for each aircraft and optimising the sequence of aircraft whenever a resource (part of a taxiway) needs to be used by more than one aircraft. As seen in Chapter 2 these are two popular approaches when it comes to solving the routing and scheduling problem of aircraft.

4.5.1 Experimental Set-up

Both routing methods were executed using different instances for Stockholm's Arlanda airport, the largest airport in Sweden. The airport details and datasets for the experiments were the first datasets that were obtained for the experiments of this chapter and have been provided by Sintef during our collaboration. The data that was used for these experiments is presented in Chapter 3. The framework was programmed in Java and was executed on a personal computer (Intel Core i3-3120M, 2.5GHz, 4GB RAM). For Method 2, the framework was programmed in Java and all of the LP models were solved using CPLEX (with the use of CPLEX Java libraries for Eclipse).

The experiments were executed using 5 instances that were based on real data for Arlanda airport as it was presented in Chapter 3. Each instance has a different complexity, duration, and number of aircraft that are routed.

The execution time for both of the QPPTW algorithms was less than 1.5 seconds which is fast enough for real time routing. The average time for Algorithm 2 - 5th instance (which is the most computationally demanding instance) was 1452ms. For the linear optimisation models the execution time was usually less than 2 seconds. For instance 5, the problem was solved in 1840ms on average for Model 1 and in 2066ms on average for Model 2.

As this chapter focuses on the pushback process of aircraft and in order to simplify the problem, the experiments in this section were executed using only departing aircraft. Input information were: the starting point of the aircraft (gate), the end point (runway), and the time that aircraft initiate their journey.

4.5.2 Comparison Results

After executing the models that were developed with and without the pushback process being implemented, the results have been summarised in tables 4.3, 4.4, 4.5 and 4.6.

As Table 4.3 shows that in each case, it is apparent that significant additional delays result from the consideration of the pushback delays (Algorithm 2 and ‘push’ variants of the models). In fact, these delays are significantly longer in comparison to the delays without the pushback modelling. This shows that there is relatively little interaction between the aircraft when pushback delays are not considered, highlighting the importance for accurate models of including these delays.

Table 4.3: Total delays and total taxi time for each algorithm/model for instances 1 and 2.

		Total [s]:	Instance 1		Instance 2	
			Delay	Taxi time	Delay	Taxi time
Method 1	QPPTW no push		89	26606	1	35579
	QPPTW push		1313	28010	1778	37356
Method 2	LP Model 1 no push		45	26562	1	35579
	LP Model 1 push		1022	27719	1234	36812
	LP Model 2 no push		53	26570	19	35597
	LP Model 2 push		1332	28029	1802	37380

Table 4.4: Total delays and total taxi time for each model/algorithm for running instances 3, 4 and 5.

		Total [s]:	Instance 3		Instance 4		Instance 5	
			Delay	Taxi time	Delay	Taxi time	Delay	Taxi time
Method 1	QPPTW push		50	51669	109	62579	153	73711
	QPPTW no push		807	52426	1504	63974	2466	76024
Method 2	LP Model 1 no push		20	51639	50	62520	50	73608
	LP Model 1 push		593	52212	1075	63545	2618	76176
	LP Model 2 no push		20	51639	64	62534	64	73622
	LP Model 2 push		780	52399	1373	63843	2297	75855

For the instances in Table 4.4 the data is similar, but with gradually increasing traffic (see Chapter 3). It is apparent from the results that, as the traffic increases, the interactions between aircraft, and hence delays, increase even without the pushback delay modelling, although these delays are still relatively small. These interactions are increasing in a super-linear

manner in relation to the increase in the number of aircraft, as would be expected. With the explicit pushback delay modelling included, the consequent delays are much higher, as was observed for instances 1 and 2. It is also obvious that the rate of increase of the delays is rapid as the number of aircraft is increased. This will, therefore, be an even larger problem at busier airports than at quieter airports, with an increasing importance for explicitly considering the pushback delays.

Figure 4.3 shows a comparison of all of the algorithms/models with and without the pushback process for each instance. The graph visualises the great extent to which the delays can pass unnoticed when the pushback process is not considered, regardless of the methodology that is used for routing the aircraft.

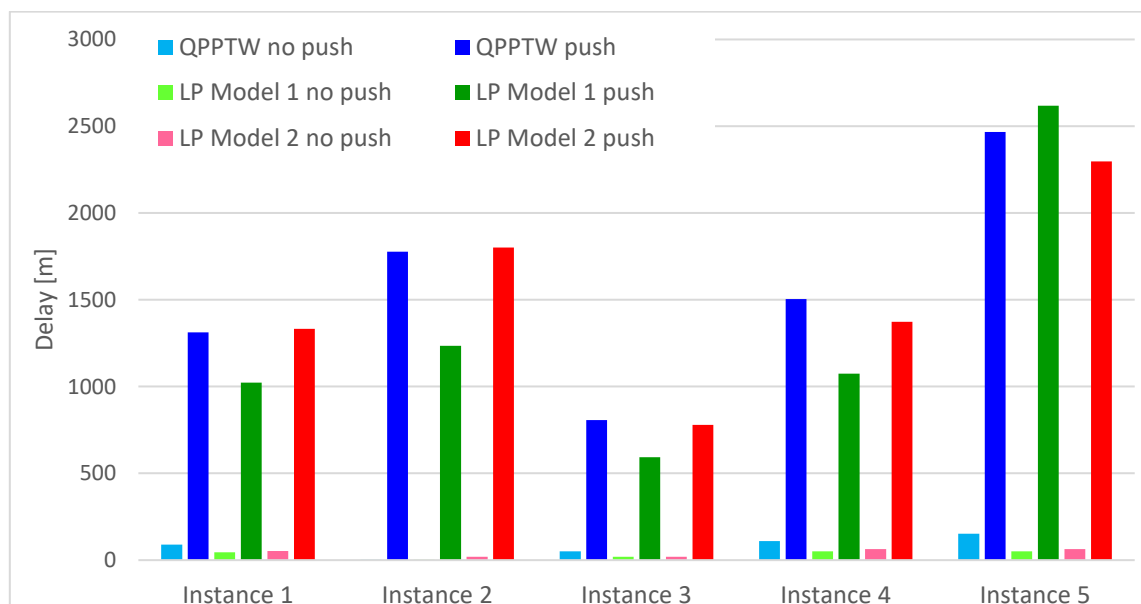


Figure 4.3: Total delay for each algorithm/model

In order to have a better understanding of the effects of the pushback process, a comparison with all of the delays that happen during Instance 1 has been performed. Table 4.5 shows the details of the flights in Instance 1 that are affected by the delays for each methodology. Flights which are unaffected have been omitted. The table presents the following information for all the models (QPPTW, LP Model 1 and LP Model 2). Columns “No push” and “Push” show the delays (in seconds) that happen when the model is executed without and with taking into consideration the pushback process respectively. Column “Difference” shows the additional delay that occurs when the pushback process is considered. Column “Type” shows the type of delay that delays the flight (when the pushback process is implemented) with “1” being a delay that happens when an aircraft is blocked due to another aircraft pushing back, “2” being a delay that happens when an aircraft cannot initiate its pushback process due to traffic by

other aircraft and “r” being a delay that happens after the pushback process has occurred where the paths of two aircraft cross or overlap one another.

Table 4.5: Flights which are affected by ground movement delays (Instance 1).

			QPPTW (Method 1)					LP 1 (Method 2)					LP 2 (Method 2)				
Flight no	Start time	Stand	No push	Push	Difference	Type	Causar	No push	Push	Difference	Type	Causar	No push	Push	Difference	Type	Causar
1	5:04:12	F37	0	0	-	-	-	0	93	93	1	2	0	0	-	-	-
2	5:04:46	F39R	0	172	172	2	1	0	0	-	-	-	0	173	173	2	1
4	5:13:53	S76	0	0	-	-	-	0	87	87	1	5	0	0	-	-	-
5	5:15:00	S78	0	166	166	2	4	0	0	-	-	-	0	167	167	2	4
7	5:24:10	53	0	0	-	-	-	0	92	92	1	8	0	0	-	-	-
8	5:24:55	57	0	173	173	2	7	0	0	-	-	-	0	173	173	2	7
9	5:29:20	G145	0	0	-	-	-	0	0	-	-	-	13	273	260	2	10
10	5:30:00	G142	0	186	186	1	9	0	186	186	1	9	0	0	-	-	-
12	5:34:45	F37	12	36	24	r	9	13	13	-	r	9	0	0	-	-	-
17	5:45:00	11	23	23	-	r	16	0	0	-	-	-	0	0	-	-	-
21	5:50:20	34	3	3	-	r	18	3	3	-	r	18	3	3	-	r	18
23	5:55:00	F33R	0	0	-	-	-	0	0	-	-	-	10	10	-	r	25
25	5:58:57	41	10	10	-	r	23	10	10	-	r	23	0	0	-	-	-
27	6:04:40	17	0	0	-	-	-	6	6	-	r	28	6	6	-	r	28
28	6:04:53	40	20	20	-	r	27	0	0	-	-	-	0	0	-	-	-
30	6:05:52	36	0	0	-	-	-	0	254	254	1	31	0	0	-	-	-
31	6:09:52	42	0	9	9	2	30	0	0	-	-	-	0	10	10	2	30
36	6:15:20	38	7	7	-	r	35	7	7	-	r	35	7	7	-	r	35
37	6:18:03	32	0	0	-	-	-	0	181	181	1	39	0	0	-	-	-
39	6:22:02	34	0	82	82	2	37	0	0	-	-	-	0	83	83	2	37
40	6:26:14	35	0	0	-	-	-	0	84	84	1	41	0	0	-	-	-
41	6:27:35	39	0	174	174	2	40	0	0	-	-	-	0	175	175	2	40
43	6:30:00	31	0	0	-	-	-	6	6	-	1	44	0	0	-	-	-
44	6:30:00	33	14	252	238	2	43	0	0	-	-	-	14	252	238	2	43
Total			89	1313	122	-	-	45	102	977	-	-	53	1332	127	-	-

It is apparent that the delays are actually affecting a small number of flights to a fairly large extent, rather than being evenly spread across many aircraft. This sort of characteristic will make it even more important to understand these delays, since they can affect the predicted taxi times considerably for these aircraft. This will make it increasingly inappropriate to use predictions which do not consider pushback operations within any integrated system. For

example, a 3 or more-minute discrepancy in predicted arrival time at the runway is likely to make a predicted runway sequence unachievable. As mentioned in Chapter 2, the runway sequence is another process that is being optimised in airports and changes to the take-off sequence of aircraft can introduce extra delays to the ground movement problem.

Similar results were observed for the remaining instances, 2 to 5. The results for all of the instances are summarised in Table 4.6.

Both of the potential causes for delays were observed to occur in the experiments; aircraft pushing back and blocking the taxi area for other aircraft (i.e. the aircraft pushing back is doing the blocking – delay type “1”) and aircraft being prevented from pushing back due to another aircraft passing at that time (delay type “2”).

Table 4.6: Results of including the pushback process.

	QPPTW (Method 1)		LP Model 1 (Method 2)		LP Model 2 (Method 2)	
	Additional no. of delays	Additional delay [s]	Additional no. of delays	Additional delay [s]	Additional no. of delays	Additional delay [s]
Instance 1	9	1224	7	977	8	1279
Instance 2	12	1777	11	1233	12	1783
Instance 3	5	757	6	573	5	760
Instance 4	13	1395	8	1025	8	1309
Instance 5	22	2313	16	2568	16	2233

In most cases where aircraft delay each other with the QPPTW algorithms and the LP Model 2, the delay was experienced by the aircraft that was set to pushback later (see Table 4.5 – delay type “2”), as expected. These aircraft will often not be able to start the pushback process at all since the edges in front of the stand would need to be clear for the whole duration of the pushback process. However, with the LP Model 1 these kinds of delays were avoided as the aircraft that was set to push back had the priority most of the time (its operation takes longer so it was more likely to start earlier when the two were in contention) and the second type of delay was observed more often.

An example of this case can be seen with two flights that conflict with each other in Instance 1. Flight 1 is set to start its pushback process 34 seconds before flight 2 and it passes through the areas that flight 2 pushed back. When the routing of the aircraft was performed using the QPPTW algorithm or the LP Model 2, flight 2 could not complete its pushback process before flight 1 had to traverse in front of the stand where flight 2 was parked. This resulted in flight 2 being forced to initiate its pushback process almost 3 minutes later. When the LP Model 1

was used for the routing process of the two aircraft, flight 2 would initiate its pushback process on time, forcing flight 1 to wait for around 1.5 minutes for this process to be completed.

Comparing the approaches, it can be observed from the results that, even though the pushback process can increase the amount of delay, the LP Model 1 seems to perform better than the LP Model 2 and the QPPTW algorithm for the first four instances. The LP Model 1 routes all of the aircraft in advance and still takes the pushback process into consideration, however the main reason that there are fewer delays is that it will allow an aircraft to push back immediately even if it has to delay a taxiing aircraft to do so (as seen in the example above). This may not be practical at real airports, however. In the LP Model 1 if aircraft have the same departure time they can also push aircraft back in parallel, resulting in aircraft not interacting with each other. However, in instance number 5 where there are aircraft departing every 2-3 minutes and it is harder for an aircraft not to interact with another (as it is when pushing back at the same time in the same taxi area for Model 1) the QPPTW algorithm and the LP Model 2 perform better than the LP Model 1.

The QPPTW algorithm has the advantage of being able to re-route aircraft when necessary, whereas the LP models always apply the shortest paths. In this case, however, this advantage seems to be no help. This implies that the shortest path approach works well for Arlanda. Investigating the extent to which this is, or is not the case for other airport layouts, where there are more options for paths with similar lengths, is investigated in Chapter 5.

The majority of the additional delay was experienced within the area around the stands and was caused directly by the pushback process. However, in some cases the delays close to the stands also caused later delays, with aircraft being delayed enough to interact with other aircraft later on. For example, in instance 1 with the QPPTW algorithm, aircraft 9 delays aircraft 10 and then aircraft 12 and 10 interact. This can also affect the order in which aircraft arrive at the runway in some cases. For example, with Algorithm 1 aircraft arrive at the runway in the order 10 – 11 – 9 – 12, whereas with Algorithm 2 it is 11 – 9 – 10 – 12. When considering the integration of systems, this can affect the feasibility of potential take-off sequences with appropriate re-sequencing no longer being possible, and hence is also important to understand.

To sum up, the results show that the pushback process significantly affects the routing process regardless of the routing methodology that is used. Firstly, it affects the number and duration of delays and failure to consider the pushback process meant that the taxi times for some aircraft could be greatly underestimated, substantially reducing the potential benefits from using a ground movement system. Secondly, in some cases the interactions between aircraft

even changed the order in which aircraft reached the runways, which could affect the potential benefits from the interaction with a take-off sequencing system.

The take-off sequence is an important parameter in optimising the ground movement process in airports that can have significant implications in the capacity of the airport as mentioned in Atkin et al. (2007). In the next chapter (Chapter 5), the effects of the pushback process are examined when the runway sequence is used as an input to the routing process in order to guarantee the correct sequence.

4.6 Comparison of the two Methodologies

As it was observed in the previous section, the LP model 1 seems to outperform the QPPTW algorithm. Even though this chapter focused on examining how the implementation of the pushback process can affect the routing process, it is important to understand what the benefits of each methodology are, and which methodology is more suitable for routing aircraft.

The LP models focus on the sequence that the aircraft will use the resources while using only the shortest path, whereas the QPPTW algorithm has a fixed sequence and focuses on the quickest path for each case individually. The experiments in the previous section, were solely designed to highlight the effects of the pushback process between departing aircraft and focused mainly on delays that occur in the areas where aircraft push back. As mentioned in the previous section, most of the delays were observed close to the stands, between aircraft pushing pack close to each other. However, when solving the full ground movement problem, delays can happen in many areas around the airport (especially in bottleneck areas) and by arriving aircraft as well.

Arriving aircraft, usually head towards the opposite direction from departing aircraft (from the runway to the stand). This makes it harder for a methodology that uses the shortest path for routing each aircraft - such as LP model 1 – to route all of the aircraft without adding long delays to the solution every time there is a conflict between an arriving and a departing aircraft. When two aircraft travel towards opposite directions and they have to use the same resources (taxiway) one will always have to wait for the other one to complete using these resources before it is able to continue its journey. QPPTW on the other hand, has an advantage in these cases, since it can route aircraft around a longer path (taking advantage of multiple or parallel taxiways), avoiding long delays.

For this reason, further experiments were executed, this time, including arriving aircraft.

4.6.1 Comparison of two Methodologies when Arrivals are Included

In order to be able to see the effect of arriving aircraft, new instances were used that included arriving aircraft as well (see Chapter 3).

After executing the experiments, it becomes apparent that any benefit that the LP Model 1 has over the QPPTW algorithm is lost when arriving aircraft are also included. As Table 4.7 shows, the QPPTW algorithm outperforms the LP Model 1 in each instance that was tested. QPPTW would perform 28% to 84% better than the LP Model 1 and in total it would route all the aircraft producing a third of the delays that the LP Model does.

Table 4.7: Comparison between the two methodologies

	Aircraft	LP Model 1 Delay [s]	QPPTW Delay [s]	Improvement
Instance 1	79	1537	1102	28%
Instance 2	75	1416	199	86%
Instance 3	207	2070	1153	44%
Instance 4	127	860	387	55%
Instance 5	95	847	478	44%
Instance 6	77	163	105	36%
Instance 7	95	6237	978	84%
Total	755	13130	4402	66%

The results in this and the previous section show that even though a methodology that focuses on the sequence between aircraft instead of the paths, can perform well when there are few aircraft considered, when the full problem is solved, it cannot compete against a methodology that can take into consideration multiple paths.

4.7 Conclusions

This chapter has investigated the importance of the pushback process in the routing and scheduling problem of the ground movement of aircraft.

Two different routing methods were considered, with various configurations to examine the effect of the pushback process. All of the methods (QPPTW, LP Model 1 and 2) had versions which did and did not take the pushback process into consideration.

In all cases, the pushback process had a considerable effect upon the resulting delays. Failure to consider the pushback process meant that the taxi times for some aircraft could be greatly underestimated, substantially reducing the potential benefits from using a ground movement system. It was observed that, although most of the delays occurred around the stands, where the pushback process happens, in some cases the delays had further effects later on, causing other aircraft to interact. In some cases, the interactions even changed the order in which aircraft reached the runways, which could affect the potential benefits from the interaction with a take-off sequencing system.

The order of considering the aircraft is important, as the methodology that would change the order of the aircraft (LP model 1) performed better than the QPPTW algorithm. The ordering between aircraft will be investigated in Chapter 5. However, it is also important to consider arrivals. The methodology that uses the shortest path ignores other paths. The QPPTW algorithm benefits from using multiple paths when arrivals are included. It will be seen in Chapter 6 that the QPPTW algorithm frequently uses alternative paths and that increasing the number of alternative paths can result in less delays. For this reason, it is important to use a methodology that can take the full potential of the airport into consideration and therefore the QPPTW algorithm was used from now on.

Pushback Process with Stand Holding and the Effects of Prioritisation Levels for Arrivals or Departures

5.1 Introduction

In this chapter, the priority between arriving and departing aircraft is examined. This research considers finding an appropriate taxi path and timings for each aircraft so as to reduce the taxi times for both arriving and departing aircraft when the pushback process is included (contrary to the previous chapter that only considered departures). The trade-off between the arrivals and departure delays, the reasons for the trade-off, and the potential for controlling this trade-off, are all questions that are considered. It is important to quantify any benefits or drawbacks, such as increased delays, and have better insight into how different prioritisation of arriving aircraft can lead to larger or smaller delays.

Moreover, stand holding is added to the routing process for departing aircraft, while taking into consideration the pushback process. Stand holding consists of finding the latest time that an aircraft can start its pushback process and still arrive at the runway on time. This means that, where aircraft would not take off as soon as they would arrive at the runway, delay should be experienced at the stand (setting off later) rather than the runway queue (waiting). i.e. the slack is moved from the end of the journey to the start. We note that airports would still want to allow some small slack at the runway to provide some flexibility in case of uncertainties such as unpredicted delays that can result in deviations from the original time schedule.

Some models take into consideration different priorities between aircraft but the available paths for aircraft are constrained. For example, Gotteland and Durand (2003b) use different

prioritisation levels between aircraft in order to optimise the use of the available paths. However, only specific paths are available for each aircraft to use. In the paper of Hayashi et al. (2015) the departing aircraft are prioritised every time there is traffic that prevents a departing aircraft from pushing back. Their model finds a time that aircraft need to be at the runway and estimates the necessary time that is needed for each aircraft to reach the runway and determines a time that aircraft should push back. The aircraft are then held at the stand until that time.

In the paper of Evertse and Visser (2017) where the ground movement of aircraft is optimised, the arrivals are implicitly prioritised over departing aircraft in order to minimise specific emissions (carbon monoxide and unburned hydrocarbons) that arriving aircraft produce in higher quantities compared to departing aircraft. Ravizza et al. (2014) use stand holding in order to minimise the time that an aircraft waits at the runway. In their paper, the latest time that an aircraft can leave the stand by taking into consideration the delays that can happen later on the path, is found. This transfers the excess waiting time to the stand, before the engines are turned on, resulting in less fuel consumption overall.

The algorithm which is utilised in this work is based on the QPPTW algorithm as it was presented in Chapter 3 and Chapter 4. In this chapter, the algorithm is extended in order to include the stand holding process, while the pushback process is still in use. The sequence of consideration of aircraft can affect both the routing process and, consequently, the pushback process, and thus is an important aspect, the effects of which need to be investigated. The effects that the choice to prioritise departures, arrivals or to have a mixed prioritisation method can have on the delays, are therefore also considered in this chapter. Providing the users with more control over choices, that can be adjusted to different prioritisation needs, can provide a more useful decision support system (a computer based system that can help with the decision-making process regarding the ground movement of aircraft) for air traffic control practitioners. Having a better understanding of what is actually happening and the effects of prioritising departures over arrivals, or vice versa, may potentially be even more useful for the airports, beyond work on ground movement decision support.

In the next section (Section 5.2), the implementation of stand holding with the pushback process into the QPPTW algorithm is presented and in Section 5.3, the effects of this implementation are examined. Section 5.4 compares the effects of prioritising departures over arrivals and Section 5.5 investigates the effects of having a mixed prioritisation. Finally, the conclusions are discussed in Section 5.6.

5.2 Integrating the Stand Holding Process with the Pushback Process

As was seen in the previous chapter, the ground movement process directly affects the runway sequencing problem. As mentioned in Chapter 2, it is important to take into consideration the order in which aircraft take off when solving the routing problem. For this it is necessary for the aircraft to arrive at the runway in the same order in which they are going to take-off. In a system with any kind of pre-departure sequencing and an aim to reduce engine running time (see for example Atkin et al. 2012 and Ravizza et al. 2014), departures tend to have a fixed arrival time at the runway rather than a fixed departure time from the stands. The actual time at which the taxi operation commences is, therefore, an output of the algorithm, rather than an input. The aim is to find the latest time that each aircraft can start the pushback process that guarantees that the aircraft will reach the runway on time. This reduces the waiting time at the runway by moving the excess waiting time to the start of the journey, before the engines are started. Furthermore, this guarantees that other aircraft cannot arrive earlier at the runway and affect the take-off sequence that is scheduled as it was seen in the previous chapter.

In this section, an extended version of the QPPTW algorithm is presented, which uses stand holding as well as including the pushback process.

5.2.1 Notation and Definitions

Table 5.1 shows the definitions of the notation that was used in the description of this work, including Algorithm 5.1 (the QPPTW algorithm with pushback process and stand holding), which is shown later. In order to maintain a thread of continuity with the original QPPTW algorithm which the Algorithm 5.1 is based on and to make it easier for the reader to better understand the additions, the notation has been kept the same for common variables that were used in the PhD thesis of Ravizza (2013c).

Table 5.1: Table of definitions for QPPTW with stand holding

E	The set of all edges
V	The set of all vertices
$e \in E$	An edge
$v \in V$	A vertex
$G = (V, E)$	The directed graph representing the airport layout, with vertices $v \in V$ and edges $e \in E$

a_e^j	The start time of the j^{th} time-window on edge $e \in E$
b_e^j	The end time of the j^{th} time-window on edge $e \in E$
$F_e^j = [a_e^j, b_e^j]$	j^{th} time-window on edge $e \in E$, from time a_e^j to time b_e^j
$\mathcal{F}(e)$	The sorted set of all of the time-windows on edge $e \in E$
H	The Fibonacci heap storing the added labels
a_L	The start time of Label L
b_L	The end time of Label L
$I_L = [a_L, b_L]$	The time interval used in a label L
$pred_L$	The predecessor label of label L
$L = (v_L, I_L, pred_L)$	A label on vertex $v_L \in V$ with a time interval I_L and predecessor label $pred_L$
$\mathcal{L}(v)$	The set of all of the labels at vertex $v \in V$
R	A conflict-free route that is being generated
$s \in V$	A source vertex
$t \in V$	A target vertex
$time$	The time that an aircraft sets off
p	The pushback duration
$T = (s, t, time, p)$	A taxi request to route, from source $s \in V$ setting off at time $time$, to target $t \in V$ and with pushback duration p (for departures)
w_e	The weight (necessary taxi time) of edge $e \in E$
$H.getMin()$	Function that returns the element with the lowest value in heap H
$Maximise(a, b)$	Function that returns the element with the largest value between elements a and b
$head(e)$	Function that returns the vertex y of an edge e that is directed from vertex x to vertex y .

5.2.2 The QPPTW Algorithm with Pushback Process and Stand Holding

In order to apply the stand holding process for departing aircraft, the algorithm needs to route the aircraft “backwards”. The “backwards” version of the QPPTW algorithm uses as a start time, the time that the aircraft needs to be at the runway (instead of the time that it has to push back), and works backwards along the route towards the stands, going backwards in time, calculating the time that it needs to be at each earlier vertex on the route in turn.

Algorithm 5.1: Quickest Path Problem with Time Windows (QPPTW) - Departures

Input: Graph $G = (V, E)$ with weights w_e for all $e \in E$, the set of sorted time-windows $\mathcal{F}(e)$ for all $e \in E$, a taxi request $T_i = (s_i, t_i, time_i, p_i)$ for aircraft i , with the source vertex $s_i \in V$, the target vertex $t_i \in V$ and the start time $time_i$.

Output: Conflict-free route R from s_i to t_i with minimal taxi time that starts at the earliest at time $time_i$, respects the given time-windows $\mathcal{F}(e)$ or returns the message that no such route exists.

```
1  Let  $H = \emptyset$ 
2  Let  $\mathcal{L}(v) = \emptyset \quad \forall v \in V$ 
3  Create new label  $L$  such that  $L = (t_i, [0, time_i], nil)$ 
4  Insert  $L$  into heap  $H$  with key  $-time_i$ 
5  Insert  $L$  into set  $\mathcal{L}(t_i)$ 
6  while  $H \neq \emptyset$  do
7      Let  $L = H.getMin()$ , where  $L = (v_L, I_L, pred_L)$  and  $I_L = [a_L, b_L]$ 
8      if  $v_L = s_i$  then
9          Reconstruct the route  $R$  from  $s_i$  to  $t_i$  by working backwards from  $L$ 
10         return the route  $R$ 
11     forall the outgoing edges  $e_L$  of  $v_L$  do
12         foreach  $F_{e_L}^j \in \mathcal{F}(e_L)$ , where  $F_{e_L}^j = [a_{e_L}^j, b_{e_L}^j]$ , in increasing order of  $a_{e_L}^j$  do
13             /*Expand labels for edges where time intervals overlap*/
14             if  $a_{e_L}^j > b_L$  then
15                 goto 11 /*consider the next outgoing edge*/
16             if  $b_{e_L}^j < a_L$  then
17                 goto 12 /*consider the next time-window*/
18             Let  $time_{out} = \text{Minimise}(b_L, b_{e_L}^j)$  /*  $b_{e_L}^j < b_L \Rightarrow \text{waiting}$  */
19             Let  $u = \text{head}(e_L)$ 
20             if  $u = s_i$  then
21                 Let  $time_{in} = time_{out} - (w_{e_L} + p_i)$ 
22             else
23                 Let  $time_{in} = time_{out} - w_{e_L}$ 
24             if  $time_{in} \geq a_{e_L}^j$  then
25                 Let  $L' = (u, [a_{e_L}^j, time_{in}], L)$ 
26                 /*dominance check*/
27                 foreach  $\hat{L} \in \mathcal{L}(v)$  do
28                     if  $\hat{L}$  dominates  $L'$  then
29                         goto 12 /*next time-window*/
30                     if  $L'$  dominates  $\hat{L}$  then
31                         Remove  $\hat{L}$  from  $H$ 
32                         Remove  $\hat{L}$  from  $\mathcal{L}(v)$ 
33                 Insert  $L'$  into heap  $H$  with key  $-b_{L'}$ 
34                 Insert  $L'$  into set  $\mathcal{L}(v)$ 
35 return "there is no  $s_i - t_i$  route"
```

In order to integrate the pushback process into the backwards QPPTW algorithm, some changes to the main algorithm had to be made. Algorithm 5.1 shows the QPPTW algorithm that implements the stand holding process and the pushback process. The parts that are different from the original algorithm have been underlined in red

Using the stand holding process means that every time the algorithm expands to another vertex, it searches for the latest time that it needs to start moving from the previous vertex in the aircraft's path, to reach the current vertex. Similar to the QPPTW algorithm that was used in the previous chapter every time the algorithm tries to expand in a new vertex, it has to check if that vertex is a stand (line 20) which will be the starting point of the aircraft and the target point of the algorithm. If yes, then the time interval that needs to be available on the edge that leads to the stand vertex increases by p_f (see line 21). As mentioned earlier in Chapters 3 and 4, the QPPTW algorithm is implemented to iteratively expand out from the starting vertex to all neighbouring vertices, like Dijkstra's algorithm, but with time windows denoting when the edges are already in use. In this implementation, the algorithm will look for an available time window on that edge that can fit the extended time interval that is necessary for an aircraft to use the edge as it does for all other edges (see line 24). However, using the stand holding process means this time it searches for available time windows backwards in time. The weight of the edge leading to the stand (where the engine start-up will take place) will also increase by an amount equal to the pushback duration for the aircraft which is pushing back (see line 21). Finally, the available time windows of this edge are going to be updated based on the extra time that is needed for the pushback process in order to block the nearby edges from being used by other aircraft during this time.

5.3 The Effects of the Pushback Process with Stand Holding

In this section, the delays that can happen during the routing process when the pushback process is implemented are investigated using the stand holding process to route the departing aircraft and also considering the arriving aircraft which are routed forwards in time as in Chapter 4.

In order to understand the effects of the stand holding process in practice, the developed model (see previous section) was executed (for departing aircraft) using real data of this from seven different days from Zurich airport (see Chapter 3); as the previous datasets did not include arriving aircraft, new datasets have been used. Since arriving aircraft need to arrive

at the stand and turn their engines off as soon as possible, the arrivals were routed first using the typical implementation of the QPPTW algorithm (see Chapter 3). The departing aircraft were routed second, using the algorithm that was presented in the previous section.

The airport where the experiments were executed has been changed in order to be able to include data that contain arriving aircraft and multiple runways as well (see Chapter 3). Input information were: whether a flight was arriving or departing, the weight class (light, medium, or heavy), the starting point of the aircraft (the runway for arrivals, the stand for departures), the end point (the stand for arrivals, the runway for departures), the landing time for arrivals and the take-off time for departures.

The framework was programmed in Java and executed on a personal computer (Intel Xeon E5-1620, 3.7GHz, 32GB RAM). The execution times varied from 9 to 15 seconds for routing all moving aircraft for one day (from 780 to 840 aircraft movements), which is fast enough for real time routing, especially since far fewer aircraft would be simultaneously routed in practice. Since stand holding and the pushback process are concepts that are only relative for departing aircraft, departures have been routed using the algorithm that was described in the previous section (Algorithm 5.1) and arriving aircraft have been routed using the original QPPTW algorithm (see Chapter 3).

Table 5.2 shows the total delays that occur when the program routes different sets of aircraft. As defined in the previous chapter (Section 4.3.4) a delay is the excess travel time of an aircraft and is calculated by subtracting the minimum taxi time and the pushback duration (if there is any) from the total taxi time of that aircraft. The second column shows the delays in seconds that occur when all of the aircraft are routed, both arriving and departing. The information in parentheses shows the split of the delay between arriving (first number) and departing (second number) aircraft. The third column shows the delays that occur when only the arriving aircraft are routed and column four when only the departing aircraft are routed. Column five shows the additional delay that is introduced to departures when the arriving aircraft are taken into consideration. Finally, the last column shows the number of additional individual delays that are introduced when arriving aircraft are also taken into consideration.

Firstly, a significant difference in the delay between arriving and departing aircraft is apparent. This is expected as the pushback process can cause both more delays and longer delays than those that happen when the aircraft are routed without modelling this process (see Chapter 4). Arriving aircraft, on the other hand, do not have to wait for anything as they are prioritised (by routing them first), so they can park as soon as possible after they land.

By comparing the results of the two set ups, it is clear that most of the delays occur because of arriving aircraft. When departing aircraft are solved separately (without taking into

consideration arriving aircraft), the delay for each aircraft would be on average around 8 seconds. When the arriving aircraft were also routed, the delay for each aircraft rose by 17 seconds (211%) to 25 seconds. The departing aircraft need a clear apron to push back onto, and arriving aircraft disturb this process.

Table 5.2: Delays when the stand holding process is used

	Arrivals + Departures total delays [s]	Arrivals only delays [s]	Departure only delays [s]	Difference [s]	No. of additional delays
Day 1	23904 (80/23824)	80	7814	16010	65
Day 2	20452 (63/20389)	63	4624	15765	72
Day 3	15646 (9/15637)	9	4888	10749	56
Day 4	25501 (74/25427)	74	9349	16078	77
Day 5	18427 (93/18334)	93	5436	12898	50
Day 6	22017 (29/21988)	29	8116	13872	58
Day 7	14906 (91/14815)	91	4905	9910	62
Sum	140853 (439/140414)	439	45132	95282	440

Figure 5.1 shows the number of aircraft that are delayed by “x” or more during day 1. As shown in Figure 5.1, the majority of delays are short in duration, with 50% of the observed delays during day 1 being less than 34 seconds (see blue dotted line). The longest delays that happen are because departing aircraft cannot find a large enough window to push back in between other aircraft, whether arrivals or other departures. Even if one aircraft passes in front of a parked aircraft once every 4 minutes the parked aircraft simply cannot commit to initiating the pushback process. This results in departing aircraft pushing back much earlier than needed in order to ensure that they are going to arrive at the runway on time. In one extreme case, an aircraft (flight 264 on day 1) had to start pushing back approximately 25 minutes before it would have done in isolation. With a 4-minute pushback duration and only 3 minutes to travel the distance, this is a considerable increase in time, and equates to a significant additional engine running time, wasting fuel. This is obviously unrealistic, and controllers would not allow this in practice. However, this is a rare case. The majority of the delays for day 1 have a much shorter duration. As Figure 5.1 shows, a significant number of aircraft (15%) that are delayed by more than 4 minutes (see green line) but few delays (3.3%) last longer than 11 minutes (see red line).

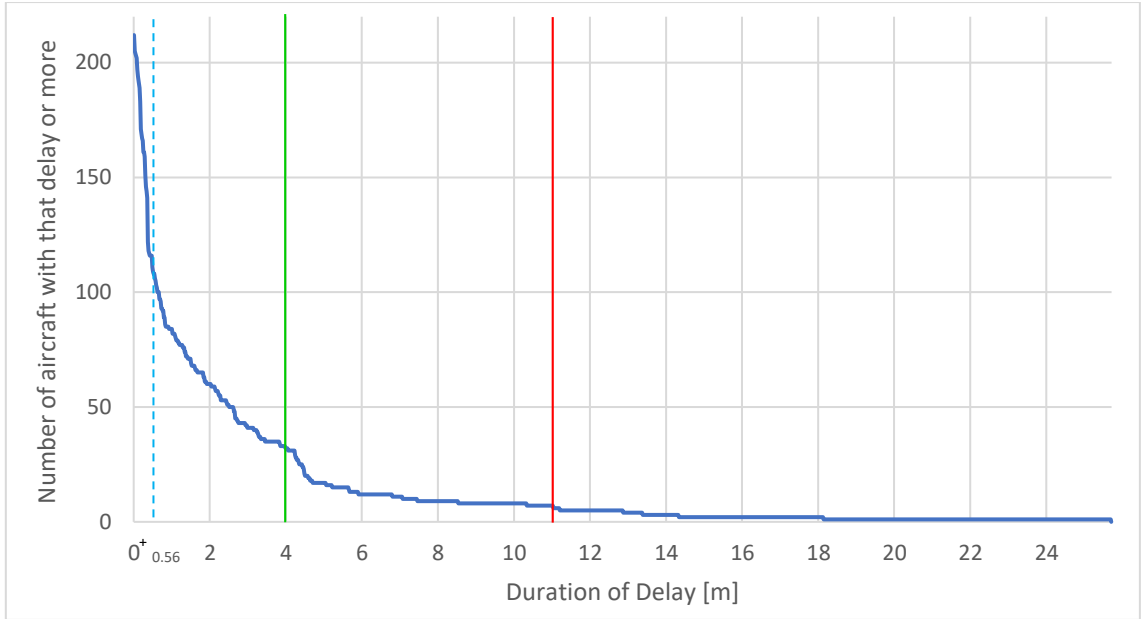


Figure 5.1: Delay graph that shows number of aircraft that are delayed by “x” or more for day 1

The above results led to the belief that aircraft during the pushback process should have a higher priority. The hypothesis is that if departing aircraft are prioritised over arriving aircraft, or at least have a higher priority than modelled here, there would not be such a large difference in the duration of delays between arriving and departing aircraft. For this reason, the next section focuses on the prioritisation between arriving and departing aircraft and how this affects the duration of delays.

5.4 Priority Between Arrivals and Departures

Arriving aircraft need to taxi from the runway (where they land) to their destination stand (where they unload the passengers and cargo). Departing aircraft need to taxi from the stand to the runway, and usually need to be at the runway at a specific time. The objective in both cases is usually to reduce the taxi time, the time that the engines are running and burning fuel. Unfortunately, these taxi operations usually take place on the same taxiways, potentially in opposite directions, so that there is often a decision about which aircraft should use a taxiway first. There are often situations where one aircraft has to wait for another to move and some kind of prioritisation is needed.

In the previous section arriving aircraft were prioritised, since parking at the designated stand is a straightforward process and shutting down the engines as soon as possible will save fuel. At busy airports, the runway throughput is often the primary capacity constraint. A take-off sequence may even be pre-planned at some airports in order to ensure a high runway

throughput, and pushback times may have been determined in order to ensure a good runway throughput (Atkin et al. 2012). Delaying aircraft from reaching the runway may risk a loss of runway capacity, or missing a take-off slot, so should be avoided.

In this section, the effects of prioritisation between arrivals and departures is examined. As was seen in the previous section, when the pushback process and the stand holding process are implemented for the departing aircraft, the number and the duration of delays can be significantly different for arrivals and departures. For this reason, prioritising the departures is examined and compared to prioritising arrivals.

The same model and framework that were used in Section 5.3, were also used here for all of the experiments, as well as the same flight data from Zurich airport.

5.4.1 Importance of the Consideration Order

Since QPPTW considers aircraft one at a time, the order in which they are considered is important. A first come first served order, whereby the first aircraft which will set off is routed first, is often sensible, for two reasons: 1) when time windows are allocated for edges it is much easier to allocate a time window after the existing usage of the edge, than to try to fit a time window in between other time windows (the gaps will often not be large enough, as discussed later); 2) first-come-first-served is often considered to be fairer when considering which aircraft should get priority at a vertex/edge. In order to do this, however, it is necessary to know when each aircraft will start moving. For the arrivals, this is not a problem, since aircraft are routed based on the time that they arrive at the airport; since there is no advantage to waiting at the runway before starting their taxi back to the stands and the objective is to reach the designated stand as quickly as possible, aircraft will start to taxi as soon as they land. This is not the case for departures, however.

As discussed earlier in Section 5.3, in a system with any kind of pre-departure sequencing and an aim to reduce engine running time (see for example Atkin et al. 2012 and Ravizza et al. 2014), departures tend to have a fixed arrival time at the runway rather than a fixed departure time from the stands. The actual time at which the taxi operation commences is, therefore, an output of the algorithm, rather than an input. Finding an approximate start time (pushback time) for the departing aircraft can be achieved by subtracting the *minimum* time (see ‘Calculating delay’ Chapter 4, Section 4.3) that an aircraft needs to reach the runway from a specific stand from the time at which an aircraft must reach the stand. This time does not take into account any delays, however, so it is possible for the actual start time to need to

be earlier than this. Once taxi start times (real for arrivals, or estimated for departures) are known, then these are utilised for the consideration/ prioritisation order of aircraft.

5.4.2 Trade-off Results After Prioritising Departures

In the previous section (Section 5.3) the experiments were executed by giving a priority to arriving aircraft. In order to investigate the effects of different prioritisation set-ups between aircraft, new experiments were executed in this section having the departures prioritised.

Table 5.3 shows the total delays that occur when the arrivals are prioritised and when the departures are prioritised. The first two columns show the sum of the delays in seconds that occur when all of the aircraft are routed for each day, and a breakdown of the delays that are caused to arriving and departing aircraft, respectively, in the parentheses. The third column shows the difference in the delays between prioritising arrivals and departures.

After comparing the delays when arrivals are prioritised to the delays when departures are prioritised (in Table 5.3) it is evident that in all of the cases, prioritising the departures results in a huge increase in delays for arrivals. However, the total delays are considerably lower, ranging from a 35% to a 51% decrease in each case. This is a significant difference, reducing the total delay to be closer to 3 hours rather than the 6 hours from just having different priorities. Part of the problem is that departures are more limited as they must use the apron near to their stand to push back onto, but this apron also forms a part of the route for other aircraft. Routing departing aircraft first effectively reserves this apron for the pushback operation, potentially finding alternative routes around it for the other aircraft. The delays that can happen due to the morphology of the area where aircraft push back, as well as the use of alternative paths will be examined in more detail in Chapter 6.

Table 5.3: Different priorities in routing arrivals and departures

	Prioritise Arrivals (Arr/Dep) [s]	Prioritise Departures (Arr/Dep) [s]	% Difference
Day 1	23904 (80/23824)	11735 (7814/3921)	-51% (9668%/-84%)
Day 2	20452 (63/20389)	13222 (4624/8598)	-35% (7240%/-58%)
Day 3	15646 (9/15637)	8402 (4888/3514)	-46% (54211%/-78%)
Day 4	25501 (74/25427)	15260 (9563/5697)	-40% (12823%/-78%)
Day 5	18427 (93/18334)	10470 (5313/5157)	-43% (5613%/-72%)
Day 6	22017 (29/21988)	12132 (8116/4016)	-45% (27886%/-82%)
Day 7	14906 (91/14815)	8373 (4905/3468)	-44% (5290%/-77%)

The results in Table 5.3 show that arriving aircraft play a significant role in an airport. It is clear that the interaction of arrivals and departures can greatly increase the delay for whichever are routed second, so it is obviously important to consider both in the routing process. This shows that solving only half of the ground movement problem (either arrivals or departures) is unlikely to provide realistic estimates of delay at airports such as Zurich, where common taxiways are often used during the pushback process. These results also indicate that it is better to prioritise the departures, due to the time that they spend in the pushback process, the need for that to be a continuous span of time in one location, and their inflexibility in where to spend this time.

5.5 Mixed Prioritisation

Further to solving the problem by prioritising departing aircraft, the experiments were also executed with having a mixed prioritisation between arrivals and departures. In this section, the effects of a mixed prioritisation are examined, and the reasons why delays happen are investigated. Furthermore, the experiments were also executed with different levels of mixed prioritisation, where the priority between arrivals and departures is balanced in order to have more control over the trade-off.

5.5.1 Mixed Prioritisation Results

The results for each of the days are summarised in Table 5.4. The arrivals column shows the sum of the delays in seconds for each day when arrivals are prioritised, with the breakdown of the delays between arrivals and departures being shown in parentheses. The departures column shows the same information for when departures are prioritised and the mixed column when the prioritisation is mixed. When departures and arrivals are ordered separately, arrivals are ordered by landing time and departures by take-off time. When the two are prioritised together, the approximate taxi starting times are determined for departures (as described earlier in Section 5.4.1) and are used along with the landing times for arrivals to prioritise the aircraft.

It can be seen from Table 5.4 that, on average, having a mixed priority between arrivals and departures results in a much better overall efficiency than prioritising arrivals alone, but is slightly worse than prioritising the departures. On average, having a mixed priority can save more than 2 hours and 20 minutes of delays per day compared with prioritising arrivals. However, prioritising departures can save around another 3 minutes more per day compared with mixed priority.

Table 5.4: Delays of aircraft for different prioritisations

	Priority					
	Arrivals [s] (Arr/Dep)		Departures [s] (Arr/Dep)		Mixed [s] (Arr/Dep)	
Day 1	23904	(80/23824)	11797	(3983/7814)	12044	(3928/8116)
Day 2	20454	(63/20391)	13330	(8706/4624)	13762	(8282/5480)
Day 3	15646	(9/15637)	8469	(3581/4888)	8782	(3592/5190)
Day 4	25502	(74/25428)	15261	(5912/9349)	14544	(6203/8341)
Day 5	18427	(93/18334)	10482	(5046/5436)	11091	(4879/6212)
Day 6	22017	(29/21988)	12141	(4025/8116)	12230	(3893/8337)
Day 7	14906	(91/14815)	8388	(3483/4905)	8799	(3178/5621)

Importantly, however, the prioritisation also changes the composition of the delays. When prioritising arrivals, even though the total delay is very high, the delays that affect the arriving aircraft are much smaller (almost insignificant) in comparison to the delays that are caused to the departing aircraft. In contrast, when having a mixed prioritisation or when prioritising the departures, the delays that affect the arrivals are much higher, and closer to the delays that affect the departures. On average, the arrival delays are still less than the departure delays, but the overall difference is much smaller, especially for when the arrivals are prioritised.

To better understand the delay allocation, the results for day 1 have been considered in more detail. Figure 5.2 shows the number of aircraft of different types that are delayed by “x” or more seconds, for different prioritisation methods. The “Arrival” prioritisation values demonstrate the delays that happen when arrivals are prioritised, with “Arrivals (A)” showing the arriving aircraft that are affected by this prioritisation and “Arrivals (D)” the number of departing aircraft. It is clear that the delay is affecting a small number of departure aircraft by a large amount, rather than being spread evenly across the departures. It is obviously important to understand why these delays differ so much, so further investigation was performed and is discussed in the next subsection.

The “Mixed” (A) and (D), and “Departure” (A) and (D) results, similarly, show the delays for arriving and departing aircraft when there is a mixed prioritisation or when departures are prioritised. Even when departures are prioritised for day 1, more departing aircraft are still delayed (147) than arriving aircraft (103), and it is clear that this holds for any level of delay considered (the departures delay line is consistently above the arrivals delay line). With a mixed prioritisation, there is a smaller total difference between the number of delays for arriving and departing aircraft, having slightly more arriving aircraft delayed (134) than departing (115). However, when arrivals are prioritised, very few arriving aircraft are affected (7) compared to the number of delays that happen for departing aircraft (212). This is not

surprising for arrivals, since the runway will often be the bottleneck on the throughput, so arrivals will be automatically nicely spaced to follow each other around the taxiways – the only delays being due to interactions with departures.

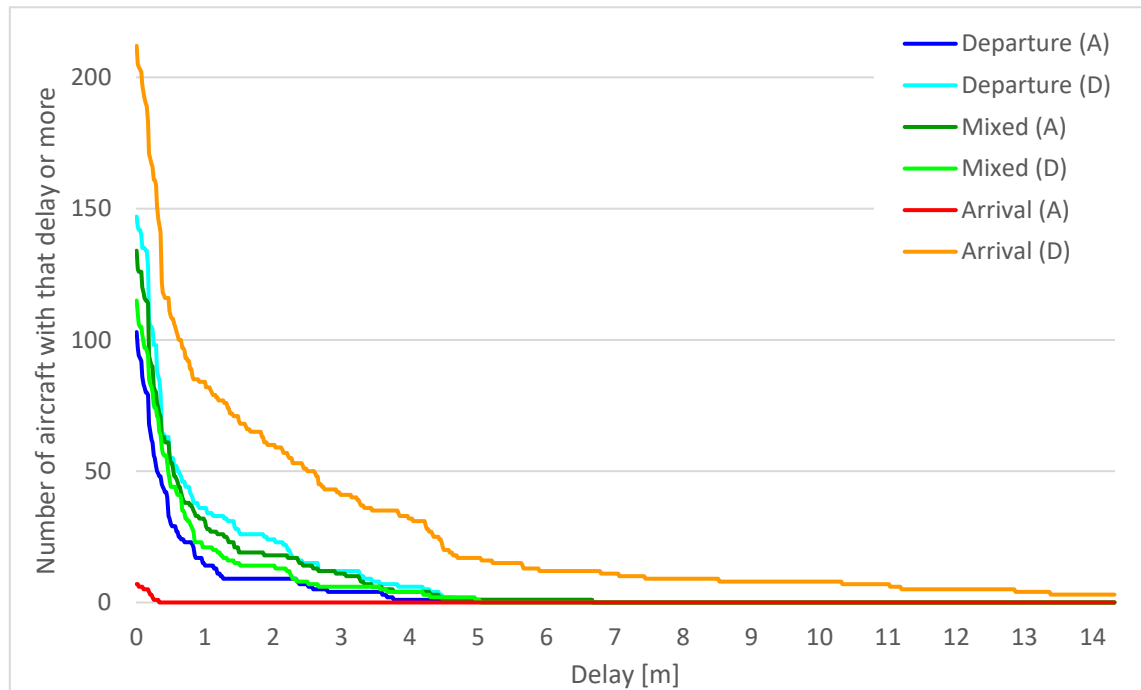


Figure 5.2: Delay graph for arrivals/departures for different prioritisations, showing the delay (in minutes) against the number of arrivals (A) and departures (D) aircraft which have that delay or higher, for three different configurations.

5.5.2 Reasons that Delays Happen

As has been mentioned earlier, many delays can happen during the pushback process. Aircraft that push back need a clear area to push back to and start their engines, which can be a time-consuming process. This area must be clear during the entire duration of the process. Prioritising the arrivals disrupted this process as the area that an aircraft needs to pushback to was often not clear for the entirety of the necessary time; which could apply a long delay waiting for a large enough gap. Depending on the size of the aircraft, the pushback and engine start-up time varies between 200 seconds (for small aircraft) to 280 seconds (for large aircraft). The taxiway needs to be clear for at least this amount of time for an aircraft to be able to initiate the pushback process.

An example will now be considered to illustrate the problem. Figure 5.3 shows the aircraft that are considered in this example and their routes. Departing aircraft number 466 (coloured

in yellow) on day 1 needs to push back. This aircraft is parked on the northwest side of terminal A.

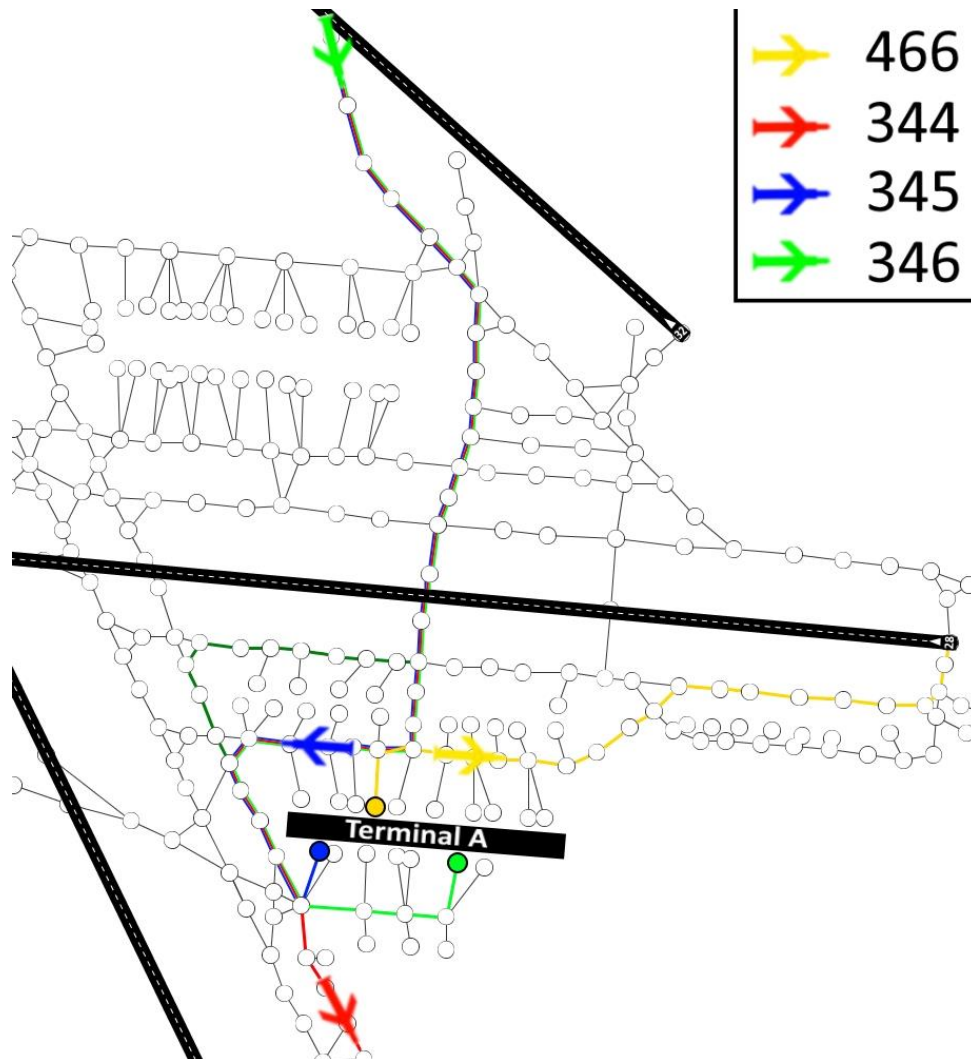


Figure 5.3: Example case where departing aircraft 466 has to push back early to avoid traffic

Three arriving aircraft (344, 345 and 346 – marked in red, blue and light green respectively) landed on runway 14 within a time window of 4.5 minutes. Two of these head towards the south side of terminal A and one of them towards the south part of the airport. Since the quickest path for these arriving aircraft passes in front of the northwest side of terminal A, when the arriving aircraft are prioritised they will block the taxiway for the duration that they are on it, preventing the departing aircraft from starting its pushback process. The 3 arriving aircraft do not leave a long enough gap between them for the departing aircraft to perform its pushback, so the algorithm handles this by performing the pushback process before any of the 3 aircraft arrive. This results in the departing aircraft pushing back 6 minutes and 48 seconds earlier than it needed to have done if there was no traffic at the airport. On the other

hand, when departures are prioritised, the departing aircraft is able to push back (being scheduled to do so before the arrivals are considered), and blocks the taxiway for a time. Because of this, the quickest path of the last of the 3 arriving aircraft (346) is temporarily blocked and the algorithm re-routes the aircraft onto a (now) quicker path, forcing it to choose a slightly longer path (marked in dark green) but only increasing its total taxi time by 11 seconds. The taxiway layout is fairly flexible at Zurich and it is possible to find an alternative path relatively easily, hence the arrival delays are relatively low. This situation will probably be different when fewer alternative taxiways are available and will be examined in the next chapter.

Having a mixed priority makes it possible for departing aircraft to start the pushback process before at least some of a stream of consecutive arriving aircraft have had a chance to block the taxiway to which the departure will push back. However, prioritising departing aircraft ensured that this was more often the case and provided a slightly better performance overall. Neither approach gave ideal results, however, so further investigation of prioritisation was performed.

5.5.3 Further Investigation of Varying the Prioritisation of Departures against Arrivals

The ability to prioritise arrivals or departures provides a useful facility for airport practitioners, and deserves further investigation. In the mixed prioritisation, the balancing of the times provided a facility to have more control over the trade-off. In order to see the trade-off between mixed prioritisation and prioritising arrivals, the flights were sorted again but this time various small “time priorities” were given to departures. To do this, extra time was deducted from the departure times when sorting the flights, resulting in more of the departures being prioritised before arrivals even when the start time of the departure was a bit later (depending on the value of the extra time). Different extra times were considered on an increasing scale (namely 10, 30, 60 and 90 seconds) in order to better understand the effects of prioritising departures over arrivals. The results are summarised in Table 5.5.

It can be observed from Table 5.5 that, as the extra time increases, (on average) the total time decreases. Even though there is some variation between the days, the total delay decreases when the departures have a higher priority. With 10 seconds extra time there is an 84 seconds improvement in terms of delays for the whole week and for 30 seconds, a bit more than 3 minutes less delay. For 1 minute extra time, the overall benefit is a bit more than 4 minutes and for 1.5 minutes extra time 6 minutes less delays occur. Furthermore, as the extra time increases, the departure delay significantly decreases, and the arrival delays increase. For 1

minute extra time, the departure delays decreased by 19.5 minutes while the arrival delays increased by 16 minutes. Eventually, the best total delay is achieved by completely prioritising the departures.

Table 5.5: Delays for different “mixed prioritisation” settings

	Delay (Arrival/Departure) [s]					
	Extra time [s]: 0		10		30	
Day 1	12044	(3928/8116)	12036	(3959/8077)	11985	(4034/7951)
Day 2	13762	(8282/5480)	13691	(8293/5398)	13773	(8466/5307)
Day 3	8782	(3592/5190)	8707	(3596/5111)	8684	(3621/5063)
Day 4	14544	(6203/8341)	14531	(6259/8272)	14527	(6303/8224)
Day 5	11091	(4879/6212)	11089	(4879/6210)	11070	(4911/6159)
Day 6	12230	(3893/8337)	12280	(3954/8326)	12219	(4019/8200)
Day 7	8799	(3178/5621)	8834	(3222/5612)	8807	(3256/5551)
Total	81252	(33955/47297)	81168	(34162/47006)	81065	(34610/46455)
Difference			84	(-207/291)	187	(-655/842)

	Delay (Arrival/Departure) [s]					
	Extra time [s]: 0		60		90	
Day 1	12044	(3928/8116)	11900	(4008/7892)	11879	(4014/7865)
Day 2	13762	(8282/5480)	13754	(8519/5235)	13756	(8545/5211)
Day 3	8782	(3592/5190)	8662	(3679/4983)	8658	(3679/4979)
Day 4	14544	(6203/8341)	14536	(6320/8216)	14536	(6320/8216)
Day 5	11091	(4879/6212)	11140	(4990/6150)	11083	(4988/6095)
Day 6	12230	(3893/8337)	12218	(4043/8175)	12209	(4058/8151)
Day 7	8799	(3178/5621)	8792	(3322/5470)	8773	(3324/5449)
Total	81252	(33955/47297)	81002	(34881/46121)	80894	(34928/45966)
Difference			250	(-926/1176)	358	(-973/1331)

It is clear from these results that the deduction of this extra time from the planned times for departures successfully allows the planner to trade off the consequent arrival delay against the departure (and total) delay, which is a vital ability for any live implementation to have.

5.6 Conclusions

In this chapter, a more accurate model that uses the runway sequence as an input to the QPPTW algorithm was developed. This model implements the pushback process with the stand holding process, which finds the latest time that an aircraft can push back and still arrive at the runway on time. Using the stand holding process when arriving aircraft were considered as well, led to large delays for the departing aircraft.

In order to minimise the pushback delays that occur in an airport, a higher priority was provided to departing aircraft. However, it should be acknowledged that this will slightly increase the delay for arrivals, which may not always be advisable. It is expected that the delays for arrivals would be even worse if there were no other, almost as good, taxi routes available for these arrivals to use instead.

A mixed prioritisation is possible, but has a problem in that the start time for the taxi operations is only known in advance for arrivals. A method for estimating the start times for departures was presented here, based upon the shortest path, and was found to have some success in the mixed prioritisation. A parameterisation method was then investigated, whereby departures could be further prioritised by varying amounts using an additional time factor, to allow for more fine tuning of the balancing between arrivals and departures. It has been shown that it is possible to control this balance by altering the value of the additional time factor, providing a simple but effective way for solving this problem.

Having a better understanding of the effects and the types of delays that occur from various prioritisations of aircraft can lead to a more realistic and accurate model that can assist the decision-making process of the ground movement operations. Furthermore, having a flexible model that can modify the priority between arrivals and departures may also provide a more tailored routing tool that will be able to adjust to differing demands across or within airports. For example, if the delay for arrivals is increasing, it may be sensible to prioritise them more, whereas it may be sensible to prioritise departures otherwise. There is value in giving an airport an additional parameter that can be used to tune operations to their preferences, however we believe that this approach can be refined further, perhaps by tailoring the prioritisation for different aircraft according to the situation at the time, for example to encourage re-routing of some arrivals when it will not greatly affect them, but to allow other departures to be pushed back earlier instead when the arrival delays would be unacceptable.

The Effects of Airport Layout and Re-routing on Taxiing

6.1 Introduction

This chapter considers the effects that different airport layouts have on the taxi time of aircraft and the delays that happen to aircraft while moving on an airport. This is an important problem, as there are many airports that have different layout morphologies, but research is usually limited to a particular airport case in each study. This is often the case due to limited datasets that are available as well as the difficulty in comparing results from different datasets. In general, using optimisation methods in airports can be very useful as it can reduce the levels of CO₂ emissions. However, it is important to know the effects of an optimisation method in different airport layouts and more importantly how the airport layout affects the solution when the problem is optimised.

The optimal solution for routing aircraft in order to go from the stand to the runway and vice versa can be significantly different in two airports. Furthermore, having an optimal ground movement solution in an airport where resources are badly allocated can still result in large delays. An example of this can be seen in Chapter 4 (Section 4.5.2 – Instance 5) where aircraft were allocated to close stands in Arlanda airport, resulting in large delays. Spending many resources in making costly improvements such as adding long parallel taxiways around an airport, is expensive and moreover it can provide only marginal improvements if the type of the longer delays that happen on that airport are not related to the number of taxiways. However, areas that delays can still happen (even though an optimised routing process is in use), can be identified and refined by making small changes.

It is important to make use of all of the available paths of the airport when routing the aircraft. For this reason, the QPPTW algorithm is used in this research as it takes into consideration all of the available parts of the airport in order to find the quickest route instead of using pre-selected paths.

In this chapter, the types of delays that can happen in airports with multiple terminals and runways are investigated. The effects that the morphology of the airport - such as physical size, number of taxiways and type of areas where aircraft can park - can have to the routing process of aircraft is considered. How the routing process is affected can be examined by observing the differences in the total taxi time of each aircraft, the number and the duration of delays as well as the use of alternative paths.

The structure of this chapter is as follows: Section 6.2 describes the airport layout problem in more detail and Section 6.3 provides more details of the airport layouts that were chosen and later modified in order to be able to compare the performance of QPPTW algorithm in different layouts. Moreover, details on how the new layouts were built are provided as well as details on how to identify the types of delays. Section 6.4 presents the results that derive after solving the problem in different layouts and discusses the findings. Finally, the chapter summarises the conclusions in Section 6.5.

6.2 Problem Description

During the routing process of an aircraft there are areas of the airport where congestion can occur and where delays are more likely to happen. In some airports aircraft are physically constrained to use only one path in order to reach their destination. Most airports, however, have multiple taxiways in order to maximise their capacity. It is important for the routing process of the aircraft to have an algorithm that can take all of the available taxiways into consideration before a route is assigned. Always following the shortest path becomes less efficient as the number of aircraft moving around the airport increases. Considering multiple paths can result in some aircraft taking a longer path but overall the total taxi time of all of the aircraft is minimised.

Airport morphology is an important aspect of the routing process. The limited number of taxiways and especially the areas where only one taxiway can be used, can create a bottleneck effect as multiple aircraft are competing for the same resource. This can result in significant delays, especially when two aircraft are heading in opposite directions. For example, when there is one arriving and one departing aircraft that need to use the same taxiway or when two aircraft start from different stands and are heading towards different runways and vice versa.

Furthermore, some aircraft can perform operations that result in blocking parts of a taxiway for a significant period of time such as when aircraft are pushing back directly to a taxiway.

Larger and more sparse airports with many taxiways may help increase the capacity of the airport and avoid or reduce these delays but the travelled distance of each aircraft increases. Usually many airports have the runways close to the terminals. However, in some cases, due to lack of available space or expansions that happen after an airport is built, the distance between the terminals and the runway can be significant. This can be observed, for example, in Arlanda airport where the 3rd runway (01R/19L) was constructed as an extension to the main airport, and is situated far from all of the passenger and cargo terminals. As optimisation methods are getting more and more efficient though, it is possible to utilise them for many processes that happen in airports and reduce the delays that can happen. This allows airports to have more aircraft that can share the same resources without causing many delays. This can make compact airports to better handle congestion and result in shorter taxi times. An example of using optimisation techniques to decrease the delays in busy airports can be seen in Heathrow airport with the Target Start-up Approval Time (TSAT) system that was utilised and resulted in shorter taxi times (Atkin et al. 2009).

The layout of the terminals and their position relative to the taxiway can also affect the movement of aircraft. Airports that have aircraft pushing back directly on a taxiway can result in more aircraft that are moving around being blocked or forced to choose another path. On the other hand, airports that have aprons (area where aircraft park without affecting the taxiways) can lead to long delays since there is only one entry/exit to the apron.

Using a different path to move around an area that is blocked - if one is available - can be advantageous when an aircraft has to wait for a long time for the preferred path. Once the aircraft pushes back and turns on its engines, the fuel consumption does not vary greatly during the taxi operation. Some aircraft need to keep pushing the brakes while the aircraft is stopped as the thrust of the engines is enough for the aircraft to move even on idle. This means that even if the aircraft is stopped at a crossroad, or waits for traffic to clear, a significant amount of fuel is consumed, similar to the amount that is consumed when moving (Wood et al. 2008). For this reason, the fuel consumption can be decreased even though a longer path is allocated to the aircraft as long as the aircraft reaches its destination earlier in time.

Understanding the effects of airport layout on the routing process of aircraft can lead to more specialised algorithms for each airport and provide a better guide in improving the infrastructure of airports. Algorithms that take into consideration multiple paths could potentially reduce the total delay in one airport, or increase it in another. So far, each algorithm is usually tested in one airport and the effectiveness of the algorithm can be significantly different on another.

6.3 Airport Layouts

In order to investigate the effects of the airport morphology, aircraft were routed around four different airport layouts. Graph representations of Arlanda airport (Sweden) and Zurich airport (Switzerland) were used for this work, and then two further airport layouts were created (based on Arlanda and Zurich airport) to identify and then illustrate some of the important characteristics of the layouts. A weighted graph for each of the airport layouts was created using as a weight for each edge, the time that is needed to traverse that edge. During the pushback process the edge weight is also affected by the weight of the aircraft as was described in Chapter 4.

6.3.1 Airports - Similarities and Differences

Zurich and Arlanda airports were chosen due to their similarities in terms of number of runways and capacity but also their differences in airport morphology. Figure 6.1 and Figure 6.2 show the graph layouts for Zurich and Arlanda respectively. The runways are shown as bold black lines and the areas where the passenger terminals, the cargo terminals and the hanger and maintenance areas are enclosed in boxes that are marked as terminals, cargo and hangar respectively. The groups are explained later in Section 6.3.3.

Both airports have three runways and handle a similar number of flights, as indicated by the similarity in the total number of passengers in 2016: 27.7 million for Zurich and 24.7 million for Arlanda. Arlanda has two parallel runways (east and west) and one runway to the north. All of the terminals/gates are enclosed between the runways. Moreover, it has long single taxiways to connect the terminals to the east runway and most of the stands are located within enclosed aprons (areas that are used for parking aircraft and are connected to the taxiway on one side). In contrast, all of the runways in Zurich are close to each other with two runways crossing one another. The terminals and stands are located to both sides of the south runway and there are many parallel taxiways and alternative routes for an aircraft to reach the runways starting from a stand or vice versa. Furthermore, the majority of stands are connected directly to the taxiways having only one small apron that is limited to one entry/exit point.



Figure 6.1: Graph of Arlanda airport - Groups

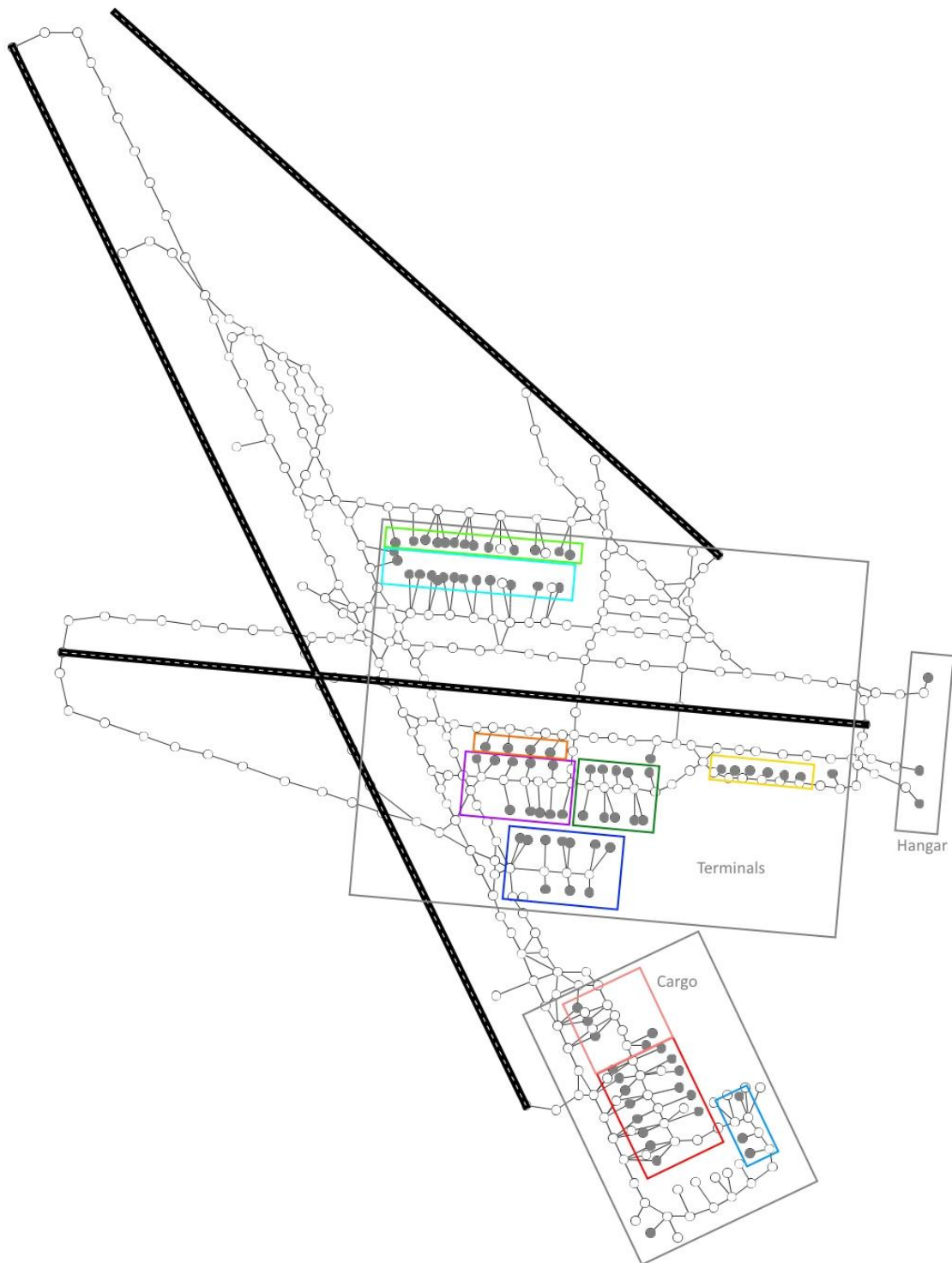


Figure 6.2: Graph of Zurich airport - Groups

6.3.2 New Airport Layouts

Based on these airports, two new airports were designed to test the effects of having additional or fewer taxiways and of aircraft pushing back onto taxiways instead of aprons. All of the airports were converted to graphs that consist of nodes and edges. Only one aircraft can use

a node at a time and aircraft can travel to all of the nodes that are connected through an edge and are not used at the same time by another aircraft that has been previously routed. If an edge is being used, the aircraft could wait for it to be free, or they could take an alternative route. The weight of each edge is the time that it takes for an aircraft to traverse this edge and the edges that are connected to stands have a dynamic weight that is subject to the size of the aircraft using them.

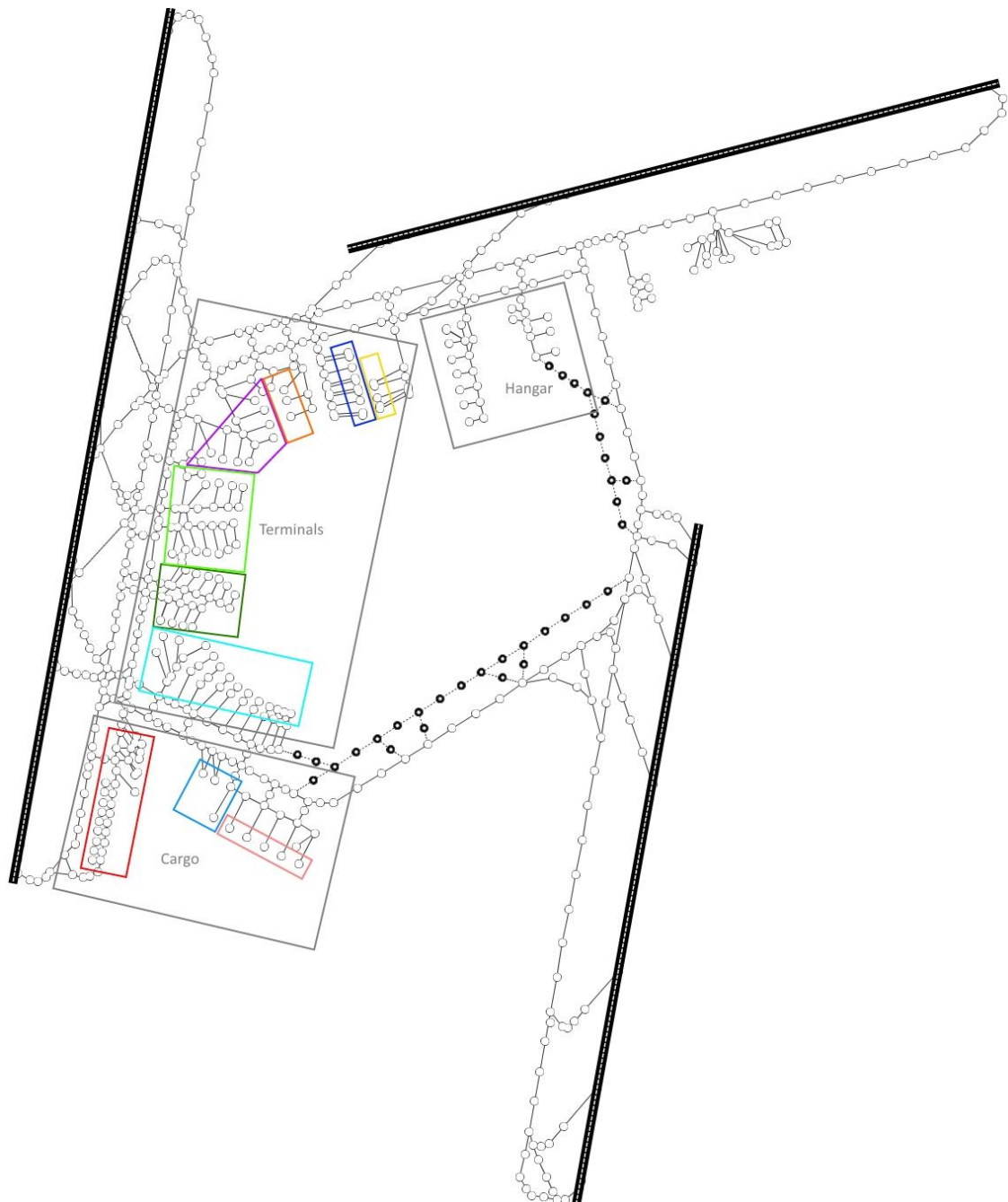


Figure 6.3: Graph of Arlanda airport – Added nodes

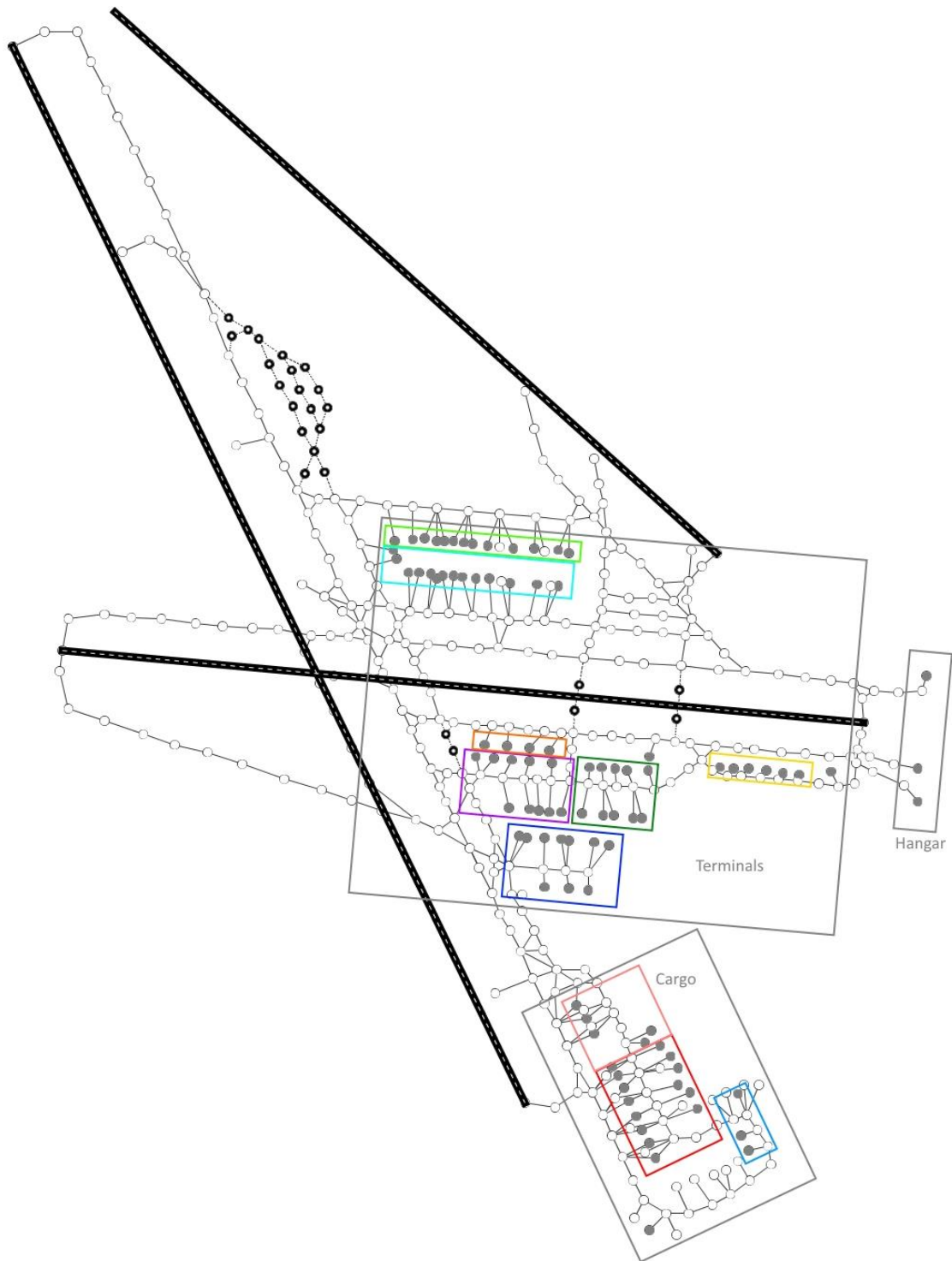


Figure 6.4: Graph of Zurich airport – Nodes removed

Alternative airport 1 is based on the Arlanda airport but has extra taxiways. The taxiways were added at strategic locations where bottlenecks were observed to occur. Previously, the eastern runway was connected to the rest of the airport (the other two runways and all of the terminals) by just two taxiways, north and south of the airport. In this layout two more taxiways were added in parallel to each aforementioned taxiway providing more options to access the eastern runway. The bold nodes and the dotted edges in Figure 6.3 represent the

extra nodes and edges that were added in order to form the new taxiways (see Figure 6.1 for comparison). The reason for adding these extra runways was to examine whether reducing the delays that happen at a bottleneck area will reduce the total delay and increase the use of longer routes.

Alternative airport 2 is based on Zurich airport but has fewer taxiways. Short parts of the taxiways were removed in order to reduce the available paths for aircraft. More specifically, taxiways that crossed the (28) runway (the runway that separates the airport to two parts, south and north) were reduced and the taxiway that connects the terminals A and B to the stand at the south part of the airport and the (34) was removed. The aforementioned changes in addition to other small alterations to a few taxiways were performed in order to reduce the alternative paths of aircraft and force them to use the preferred/shortest path, as can be seen in Figure 6.4. Contrary to Figure 6.3 the bold nodes and dotted edges in Figure 6.4 represent the edges that were removed in order to restrict the use of these taxiways (see Figure 6.2 for comparison).













6.3.3 Using the Same Data in Different Layouts

In order to have comparable results from each airport layout, the same data was used for running the experiments for all of the airport layouts. The time that aircraft depart from the stands or land at the airport as well as the size of the aircraft have remained the same (so it is possible to compare only the effects of the airport layout). The only parameter from the datasets that has been changed is the stand where aircraft start or end their journey. The stands of Zurich airport in the datasets have been converted to similar stands in Arlanda airport. In order to avoid having an inconsistent or biased use of the data, which was originally obtained from Zurich airport, and to make sure that the results are coherent, the stands from Zurich airport have been matched with the stands from Arlanda airport in the following way.

Initially each airport was divided in groups/parking areas, where aircraft are more likely to interact with each other because the resources (taxiways) were shared. These groups are marked in Figure 6.3 and Figure 6.4 with different colours. Moreover, each group was assigned one of the following types: passenger terminal, cargo terminal or hangar/maintenance area. All of the groups are enclosed by grey boxes in Figure 6.3 and Figure 6.4 that represent their type. Each group from Zurich airport was assigned (in a sensible manner) to a corresponding group in Arlanda airport firstly of the same type and then of the same or similar size. Each group in Figure 6.3 corresponds to the group with the same colour in Figure 6.4. Table 6.1 shows the number of stands and the type of each group as well as the colour that each group is marked in Figure 6.3 and Figure 6.4. Finally, each stand that

was used in Zurich airport was assigned to a random stand in Arlanda airport from the same group using a random number generator. This process was repeated 30 times and the results that are provided below refer to the average value of the 30 individual experiments that were executed for the Arlanda and the Arlanda based layout.

Table 6.1: Groups and types of stands

	Group	Stand type	Number of stands
	1	Passenger	14
	2	Passenger	10
	3	Passenger	10
	4	Passenger	13
	5	Passenger	10
	6	Passenger	5/6
	7	Passenger	4
	8	Passenger	2
	9	Cargo	14
	10	Cargo	5
	11	Cargo	3
	12	Hangar/Maintenance	3

For the examples that are provided in the results section, where the specific cases for Arlanda or Arlanda airport with extra taxiways are examined - such as distribution of taxi time or the number of aircraft that are affected by different delay duration – the most representative instance is presented. This happened to be instance 6 (out of the 30 instances in total) where the sum of all of the delays for the whole week was the closest to the average value of the 30 instances.

6.3.4 Investigating the Chosen Path

In order to be able to investigate the effects of adding or removing taxiways, it is important to be able to identify whether an aircraft is using the shortest path or a longer path. As mentioned in Chapter 3, the QPPTW algorithm is finding the quickest path, which can be either the shortest path (with or without delays during the journey) or a longer path, that was preferred as the shortest path was blocked by a previously routed aircraft and it was faster to de-tour rather than wait for the necessary resources (parts of a taxiway) to be available. However, when the aircraft is routed it is impossible to know whether the path that is chosen is the shortest path. The algorithm can only guarantee that it is the quickest at the time. For this reason, before the problem is solved, each aircraft is routed on an empty graph using a

variation of Dijkstra's algorithm that implements many characteristics of QPPTW algorithm as described in Chapter 3. The nodes of the shortest path that are visited by the aircraft are stored as a parameter of the flight that is being considered when the full problem is solved. When the aircraft is routed, the two paths (sequences of nodes) are compared, and if they are the same, any delays that happen are considered "delays on the shortest path" and if the paths are different, the delay is considered a "delay due to choosing an alternative path". The alternative path will always have a different distance, even if the difference in some cases can be very small. The sequence of nodes of the shortest path that is found for each flight will be used to evaluate the use of alternative paths in the next section.

6.4 Results

The experiments were executed using real data from Zurich airport (the largest airport in Switzerland). In order to have consistency across days, seven different days of data were used for the experiments. Using real data made it possible to have more reliable results, as there is no bias in the allocation of aircraft to stands and take-off times or how the aircraft are spread across the day.

The parameters that were used as input were: the type of movement (arriving or departing aircraft), the weight class of the aircraft (light, medium or heavy), the starting point of the aircraft (the runway for arrivals, the stand for departures), the end point (the stand for arrivals, the runway for departures), the landing time for arrivals and the take-off time for departures.

The experiments were executed on a personal computer (Intel Core i3-3120M, 2.5GHz, 8GB RAM) and the model was programmed in Java. The execution times varied between 5 to 11 seconds for routing all of the aircraft of a single day (from 780 to 840 aircraft movements per day) and between 38 to 64 seconds for solving the whole week (5609 aircraft movements in total).

The aircraft were routed sequentially, by routing the departing aircraft first (based on the take-off time), starting from the latest departing aircraft and moving earlier in time, then sequentially routing the arriving aircraft (based on their landing time). Priority was given to departing aircraft in order to minimise the delays as described in Chapter 5. The departing aircraft were sequenced backwards in time in order to guarantee that the routed aircraft would arrive at the runway in the correct order, respecting the take-off sequence that aircraft were allocated on the runway.

6.4.1 Explaining why Delays Happen

After examining the results, it is apparent that many delays happen at both of the airports (Arlanda and Zurich) even though aircraft arrive to - and depart from - each runway at different times. This can be explained by the following reasons where cases from the results are examined.

Firstly, arriving aircraft and departing aircraft move around the airport at the same time and usually towards different directions. This can cause delays especially in areas where there are not enough taxiways (bottlenecks). An example of this kind of delay can be seen in Figure 6.5 where a departing aircraft delays an arriving aircraft during day 1 of the experiments in Zurich airport. Aircraft 386 (which is coloured in red) has started pushing back from stand A and is heading towards runway 16 (see red path) where it needs to arrive before 07:04:00 in order to take-off. Aircraft 423 (which is coloured in blue) has landed at 06:55:22 on runway 16 (see blue path) and heads towards stand B. The two aircraft meet at 06:57:51 where they both want to use node C. The departing aircraft (386) has started its journey first and has priority over the arriving aircraft (423) resulting in a 24 second delay to the latter.

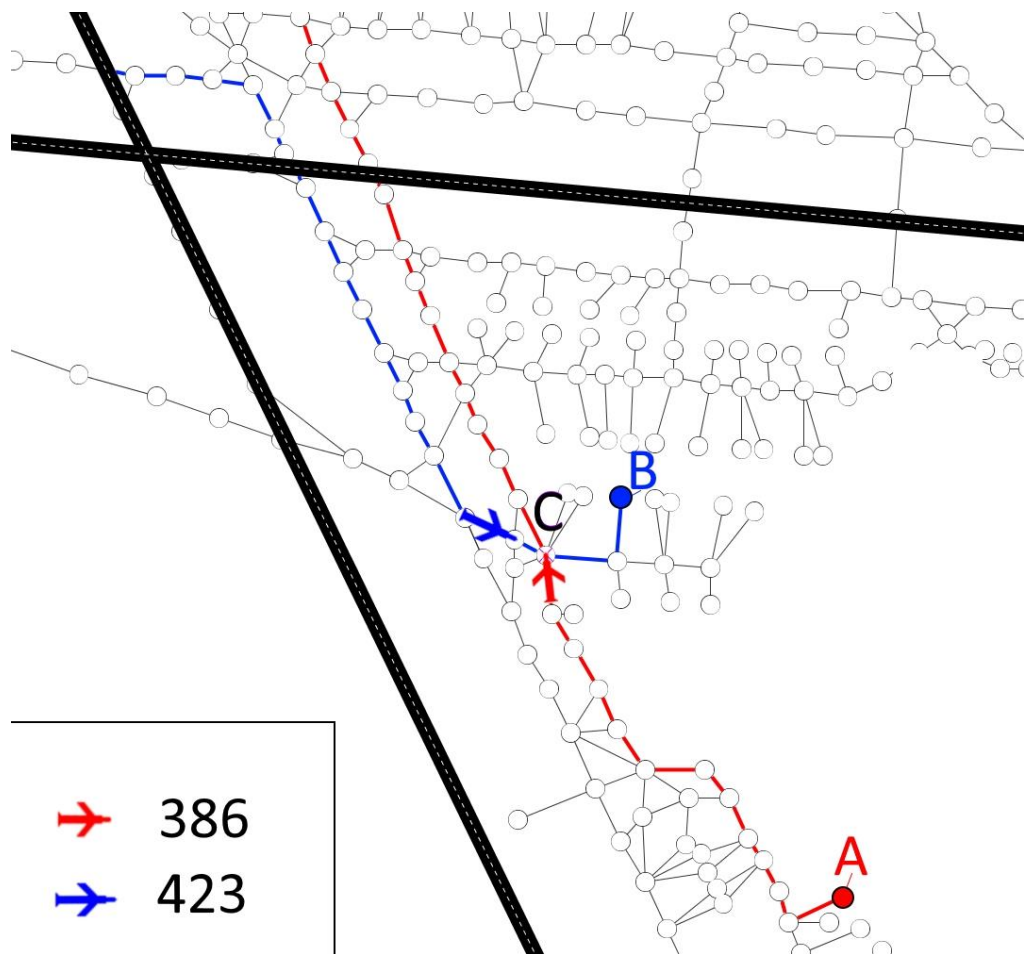


Figure 6.5: A departing aircraft (red) delaying an arriving aircraft (blue)

Furthermore, arriving aircraft are affected by departing aircraft that push back as they lock resources for a substantial amount of time. In Zurich airport pushback delays are more common as the terminals are surrounded by taxiways and aircraft that pushback can block parts of the taxiways that are more frequently used. Out of the 95 stands in Zurich airport, 82 are surrounded by taxiways (86%). In Arlanda there are more terminals that extend outwards from the main airport structure making the pushback delays mainly a problem of aircraft that park at the same area/apron. Only 36 out of the 94 stands (38%) are surrounded by taxiways.

The second reason delays happen is caused by the fact that the airports have 3 runways and stands in different areas around the airport. This causes even flights of the same type (arrivals or departures) to be on crossing paths if they start and head towards different destinations. Figure 6.6 shows a case where two departing aircraft that are heading towards different directions cause a delay. Aircraft 386 (which is coloured in red) starts at 06:52:18 from stand A and is heading towards runway 16 (see red path) where it needs to arrive before 07:04:00. Aircraft 387 (which is coloured in blue) starts from stand B and is heading towards runway 28 (see blue path) where it needs to be before 07:03:00. In order for aircraft 387 to avoid a conflict in nodes C and D, it starts the pushback process 6 seconds in advance and it is forced

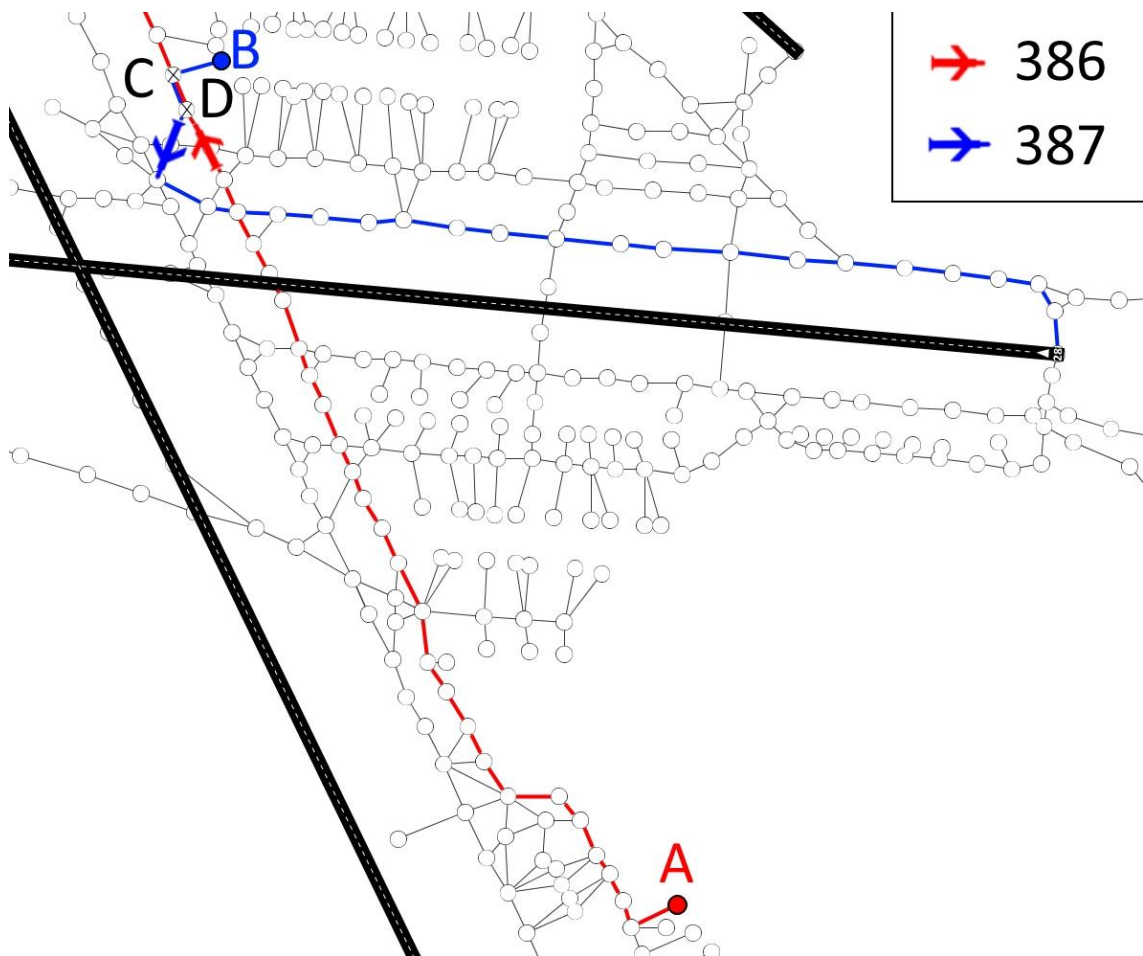


Figure 6.6: Two departing aircraft heading towards different runways

to follow a path that is slightly longer than the shortest path which further increases the travel time by 15 seconds.

The third reason that can cause delays is that some departing aircraft can interact with other aircraft that pushback in their path. Since they need to arrive at the runway at a certain sequence, some departing aircraft may need to start earlier in order to avoid delays from aircraft that are allocated later but will start their pushback process before the first aircraft has reached its destination. An example of this case can be seen in Figure 6.7, where three departing aircraft push back at the same area during day 1 in Zurich airport. Aircraft 402 (which is coloured in green) needs to be at the runway 32 at 06:23 starting from stand C. Aircraft 404 (which is coloured in blue) needs to be at the same runway before aircraft 402 at 06:21 starting from stand B. Both of these aircraft need to use the same node, (node D) so aircraft 404 is forced to start pushing back 1 minute and 4 seconds earlier in order to avoid a conflict with aircraft 402. This has a knock-on effect, as aircraft 407 (coloured in red) which needs to be at the runway at 06:16, has to start its pushback process earlier as well. Aircraft 407 starts from stand A and also departs from runway 32. Since it also needs to use node D, the aircraft is forced to start 11 seconds earlier in order to avoid blocking aircraft 404.

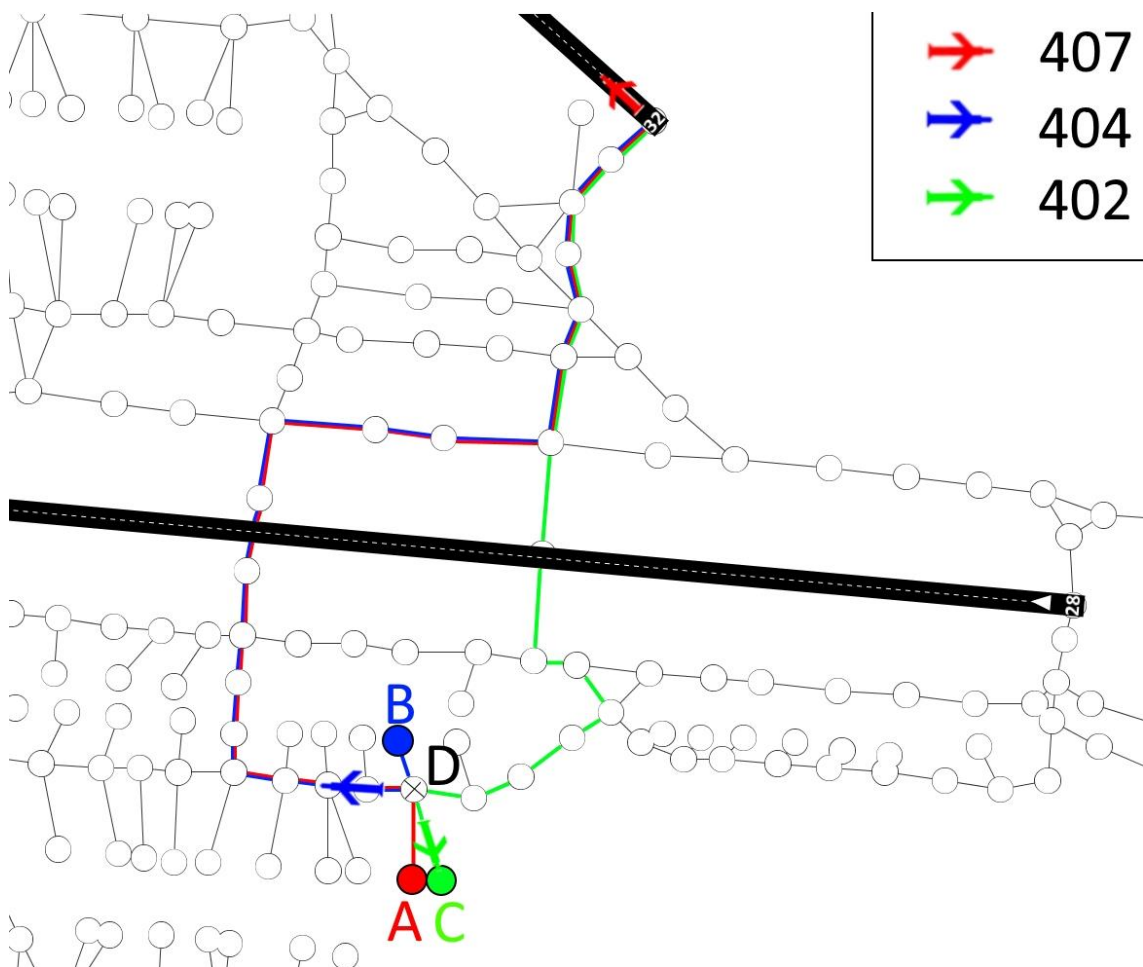


Figure 6.7: Departing aircraft need to start their pushback process earlier to avoid a conflict

6.4.2 Comparing Zurich and Arlanda Airports

After running the experiments for Zurich and Arlanda airport the total time that is needed for each aircraft to reach its destination was considerably longer in Arlanda airport. The full results are summarised in Table 6.3, Table 6.4 and Table 6.5, in the columns “Zurich” and “Arlanda”. The rest of the columns contain the results of the next subsection but were added to one table to make it easier for the reader to compare the results from all the airport layouts.

Table 6.2 shows the total taxi time which would be needed if there were no delays. It can be seen that aircraft in Arlanda airport need to travel for a significantly longer duration in order to reach their destination resulting in longer taxi times for aircraft in Arlanda. The stands in Arlanda airport are on average further from the runways compared to Zurich airport, forcing aircraft in Arlanda to keep their engines on for an extra time of 2 minutes and 26 seconds compared to Zurich airport.

Table 6.2: Total taxi time (in seconds) if there were no delays

	Zurich	Zurich less taxiways	Arlanda	Arlanda extra taxiways
Day 1	155902	174663	268753	261115
Day 2	148978	166867	265190	257960
Day 3	134085	152723	259629	251186
Day 4	157091	174753	275335	268014
Day 5	138459	157629	261185	253224
Day 6	138189	152873	247912	240908
Day 7	144963	163024	258986	251457
Total	1017667	1142532	1836992	1783864

When the problem is solved by taking into consideration all of the aircraft that are moving around the airport (and all the delays that can arise) the excess time that aircraft in Arlanda airport need to keep their engines on is slightly lower (2 minutes and 23 seconds) which means that for each aircraft routed around Zurich, 3 extra seconds of delay occur (see Table 6.3).

After examining the duration of delays that occur in each airport, the total duration of delays was slightly higher for Zurich, having a sum of 39 extra minutes of delays per day (on average) or a bit less than 3 seconds extra delay per aircraft (see Table 6.4). The delays contribute about 6.7% to the total taxi time of aircraft in Zurich compared to 3% in Arlanda.

However, even though there is more congestion in Zurich airport since the airport is more compact, the total taxi time is considerably lower.

Table 6.3: Duration of delays in seconds

	Zurich	Zurich less taxiways	Arlanda	Arlanda extra taxiways
Day 1	10214	10118	8317	6701
Day 2	12213	12307	7884	7045
Day 3	7437	8221	7562	5302
Day 4	14586	15228	8927	7962
Day 5	9680	10777	9459	8195
Day 6	11917	11879	8682	7250
Day 7	7347	7810	6184	5001
Total	73394	76340	57014	47455

Table 6.4: Total taxi time in seconds

	Zurich	Zurich less taxiways	Arlanda	Arlanda extra taxiways
Day 1	166116	184781	277071	267816
Day 2	161192	179175	273074	265005
Day 3	141522	160944	267191	256488
Day 4	171677	189982	284262	275975
Day 5	148139	168407	270644	261419
Day 6	150106	164752	256594	248158
Day 7	152310	170834	265170	256458
Total	1091062	1218875	1894006	1831319

The total number of delays is significantly higher in Zurich airport - 33% more in total or 53 more delays per day on average – compared to Arlanda airport. More importantly however, on average, 78% of the delays in Zurich were caused by aircraft choosing a longer path, whereas in Arlanda airport this percentage was 71%. This shows that in Zurich, aircraft are more likely to choose an alternative path.

Furthermore, the delays that happen due to rerouting of aircraft (taking a longer path) also contribute more to the total delay that occurs in Zurich airport. 78% of the delays are caused

by aircraft using a different path from the shortest one compared to 71% of the aircraft that do the same in Arlanda.

Table 6.5 shows for each layout a) the number of delays, b) the number of delays that happen due to rerouting of aircraft and c) the percentage of delays are caused by aircraft using a different than the shortest path.

The results indicate the following: An outspread airport with long taxiways and few connections between them such as Arlanda airport is more likely to have flights with longer routing times, smaller duration of delays, smaller number of delays and more delays that are caused by waiting for another aircraft to free the necessary resources instead of choosing a longer path.

Table 6.5: The number of delays for each airport

	Zurich			Zurich less taxiways			Arlanda			Arlanda extra taxiways		
	Total delays	Rerouting delays	Percentage of rerouting delays	Total delays	Rerouting delays	Percentage of rerouting delays	Total delays	Rerouting delays	Percentage of rerouting delays	Total delays	Rerouting delays	Percentage of rerouting delays
Day 1	225	181	80%	207	153	74%	169	124	73%	168	127	76%
Day 2	230	176	77%	200	145	73%	172	127	74%	172	130	76%
Day 3	204	167	82%	178	136	76%	117	78	67%	123	79	64%
Day 4	231	184	80%	229	165	72%	186	135	73%	191	145	76%
Day 5	197	143	73%	194	132	68%	143	90	63%	145	95	65%
Day 6	212	164	77%	187	130	70%	169	126	74%	157	117	75%
Day 7	196	153	78%	175	125	71%	167	123	73%	162	126	78%
Total	1495	1168	78%	1370	986	72%	1122	802	71%	1116	818	73%

The results show that the morphology of the airport can significantly affect the routing process of aircraft. This lead to the hypothesis that if the morphology of each airport (Arlanda and Zurich) was altered, by removing or adding taxiways, it is expected to see the same effects. For this reason, further experiments were executed and are presented in the following part of this section.

6.4.3 Further Investigation of Different Airport Layouts

The number of taxiways and alternative paths that the two airports have affect the movement of aircraft. In order to validate and quantify this effect, the experiments were run in the two altered airport layouts that were mentioned earlier in this section. These layouts use the same airport characteristics as the previous airports with the exception of extra taxiways for Arlanda and fewer taxiways for Zurich. The aim is to make Zurich airport less interconnected and Arlanda airport more interconnected. The results are summarised in Table 6.3, Table 6.4 and Table 6.5, columns “Zurich less Taxiways” and “Arlanda extra Taxiways”.

When the experiments were executed in Zurich airport with fewer taxiways, the total taxi time was increased by 12% adding approximately 20 seconds extra time to the duration of each aircraft’s travel time compared to the results using the original Zurich graph (see Table 6.3). On the other hand, in Arlanda airport with more taxiways the total taxi time was decreased by 3%, saving about 10 seconds from each aircraft’s journey time compared to the results that were observed in the original Arlanda graph. This shows that the total taxi time is indeed affected by the number of taxiways. Figure 6.8 shows the number of aircraft that are delayed by different time durations, for each airport. As the graph shows, when the number of taxiways is increased (Arlanda to Arlanda with extra taxiways), the less time it takes (on average) for an aircraft to complete its journey as the number of aircraft that are affected by long delays decreases. Arlanda airport has more aircraft that have long delays (over 90 seconds) whereas Arlanda airport with extra taxiways has more aircraft that are affected by short delays (mainly between 10 and 50 seconds). Similarly, when the taxiways are decreased (Zurich to Zurich with less taxiways), the routing time of an aircraft, on average, will increase as the number of aircraft that are affected by long delays increases. Zurich airport has more aircraft that are affected by short delays (mainly between 10 and 30 seconds) compared to Zurich airport with less taxiways that has more aircraft that are being affected by longer delays (over 40 seconds).

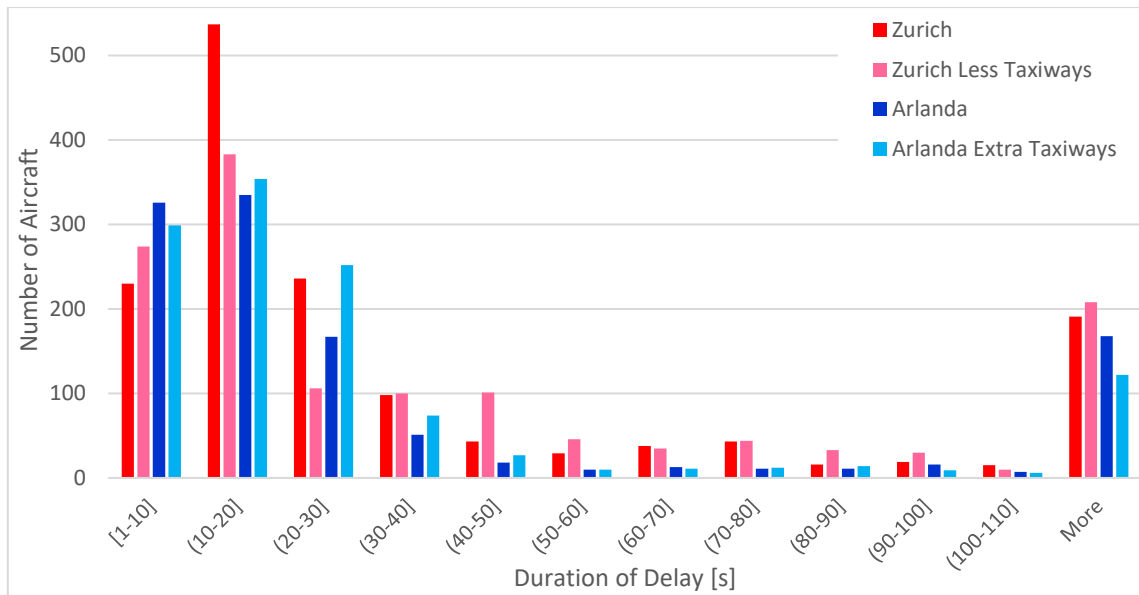


Figure 6.8: Number of aircraft for each time duration of delay in each airport

Figure 6.9 shows the distribution of taxi duration (under 16 minutes) that aircraft have during the week for all the airports. As the graph shows, even though the total taxi time was increased when the problem was solved for the Zurich layout with extra taxiways (compared to the results of the original Zurich layout) and was decreased when solved in the Arlanda layout with extra taxiways (compared to the results of the original Arlanda layout); the total taxi duration was still significantly higher when solved in the altered Arlanda layout compared to the altered Zurich layout. This shows that even though extra taxiways can improve the total taxi time in an airport, a congested airport that is smaller in size can still provide faster taxiing. This indicates that a more compact airport should be preferred compared to a more outspread one, once the ground movement model that solves the problem is implemented.

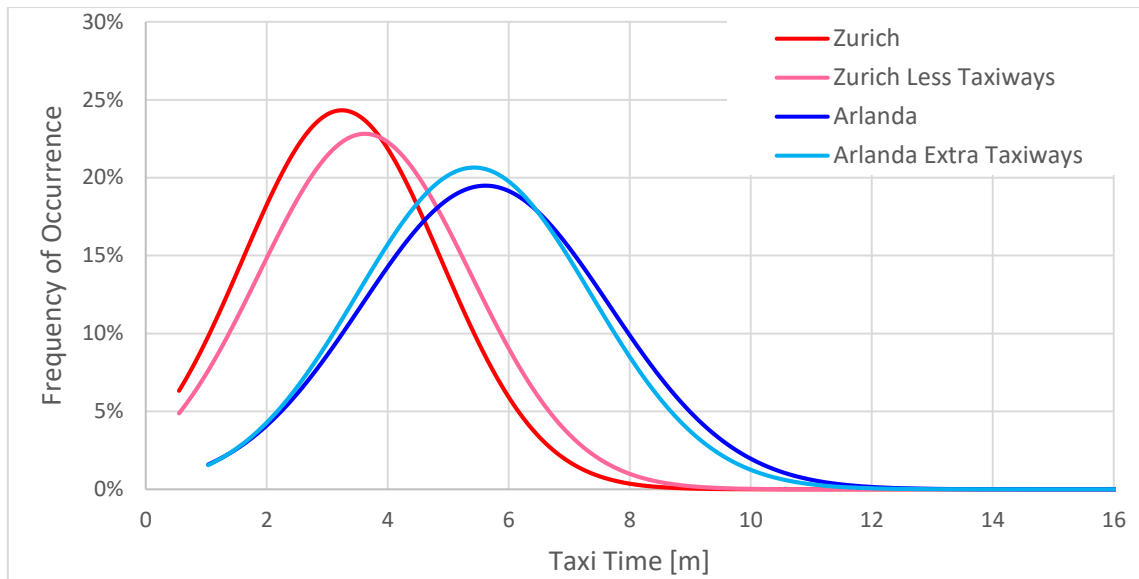


Figure 6.9: Distribution of taxi duration for each airport

Furthermore, the total duration of delays also increased in Zurich airport with less taxiways by 4% contributing to a 7-minute increase per day in delays compared to the results that were reported when the original Zurich airport was used (see Table 6.3). Again, when the taxiways were increased in Arlanda airport, the total duration of delays was decreased by 17% which is approximately 23 minutes less delays per day compared to the total duration time that was observed when using the original Arlanda graph. Figure 6.10 shows the cumulative graph of delays under 6 minutes for each airport layout. As the graph shows, delays longer than 22 seconds (see green line) happen more often when there are less taxiways. Zurich airport with less taxiways has the biggest number of aircraft that are affected by long delays, and Arlanda airport with the extra taxiways has the fewest.

Moreover, in Arlanda airport a smaller number of aircraft is affected by short delays than in Arlanda airport with extra taxiways (see left of the green line) and the reversed is observed for Zurich airport, where more aircraft are affected by short delays than in Zurich airport with less taxiways. This shows that it is more common to have shorter delays happen when extra taxiways are added. These results show that the number of taxiways affects the duration of the delays. Overall, when the number of taxiways was decreased the duration of delays increased and when the number of taxiways was increased the duration of delays decreased.

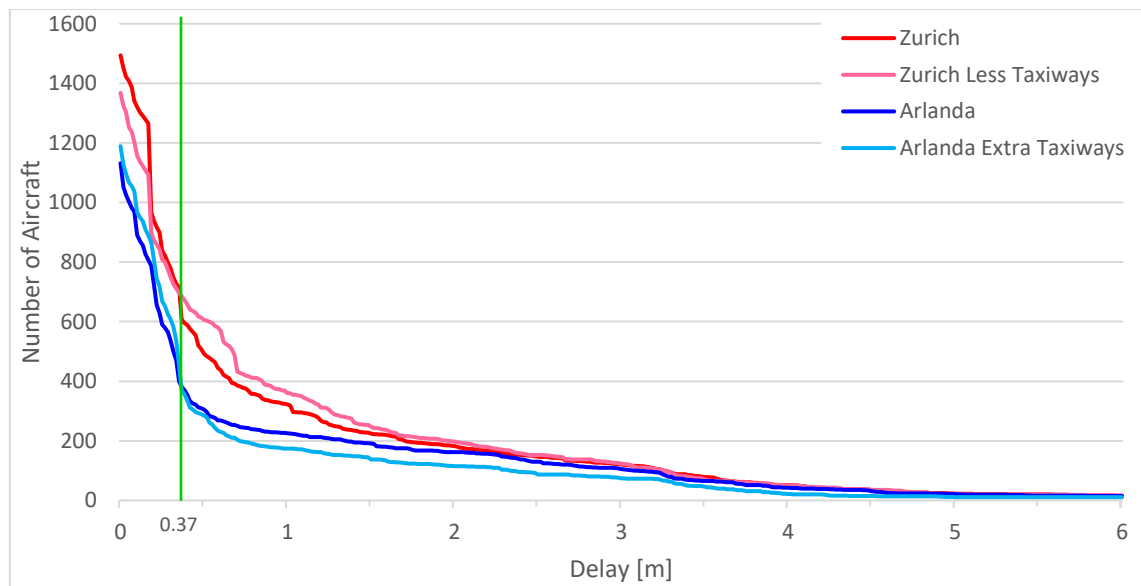


Figure 6.10: Delay graph for different airport layouts, showing the delay (in minutes) against the number of aircraft which have that delay or higher

Even though there is a significant improvement (17%) in the total duration of delay that happens when a few taxiways are added in Arlanda airport, it is not enough in order to perform as good as a smaller -in physical size- airport. Making a congested airport (such as Zurich) even more congested will increase (by 4%) the duration of the total delay but it will still overwhelmingly outperform a larger -in physical size- airport when it comes to the total time that an aircraft will keep their engines on (total taxi time) as Figure 6.9 and Figure 6.10 show.

The number of delays that happen during the week was also affected (see Table 6.5). In Zurich airport with less taxiways there were 125 less delays than the delays that happened when the problem was executed in the original airport graph of Zurich. On the contrary, the delays that happen when the problem is solved in the graph of Arlanda airport with extra taxiways did not change significantly, having a decrease of 6 delays during the whole week.

More importantly however, is the composition of these delays. When the problem was solved using the Zurich layout with less taxiways, the delays that are attributed to aircraft taking a longer path decreased compared to the delays that happened when the original Zurich layout was used (see Table 6.5). As a percentage, 72% of the delays in the Zurich layout with less taxiways were caused by aircraft choosing an alternative path compared to 78% that was the percentage for Zurich airport. Similarly, when the problem was solved using the Arlanda layout with extra taxiways, 73% of the delays were caused by aircraft taking a longer path compared to 71% that was the percentage for when the original Arlanda layout was used. This shows that when there are fewer available taxiways there will be fewer aircraft that will use alternative paths, since bottlenecks are more likely to occur. Adding extra taxiways where

bottlenecks can happen will increase the number of alternative paths that are used, given an algorithm that takes into consideration multiple paths.

These results show that adding or removing taxiways from an airport will produce a cascade of effects. Firstly, the total duration of delays is decreased (or increased) as the number of taxiways are increased (or decreased) as it was also demonstrated in Figure 6.8. This affects the total taxi time of each aircraft making the total taxi time to follow the same trend as total duration of delays as it was observed in Figure 6.9.

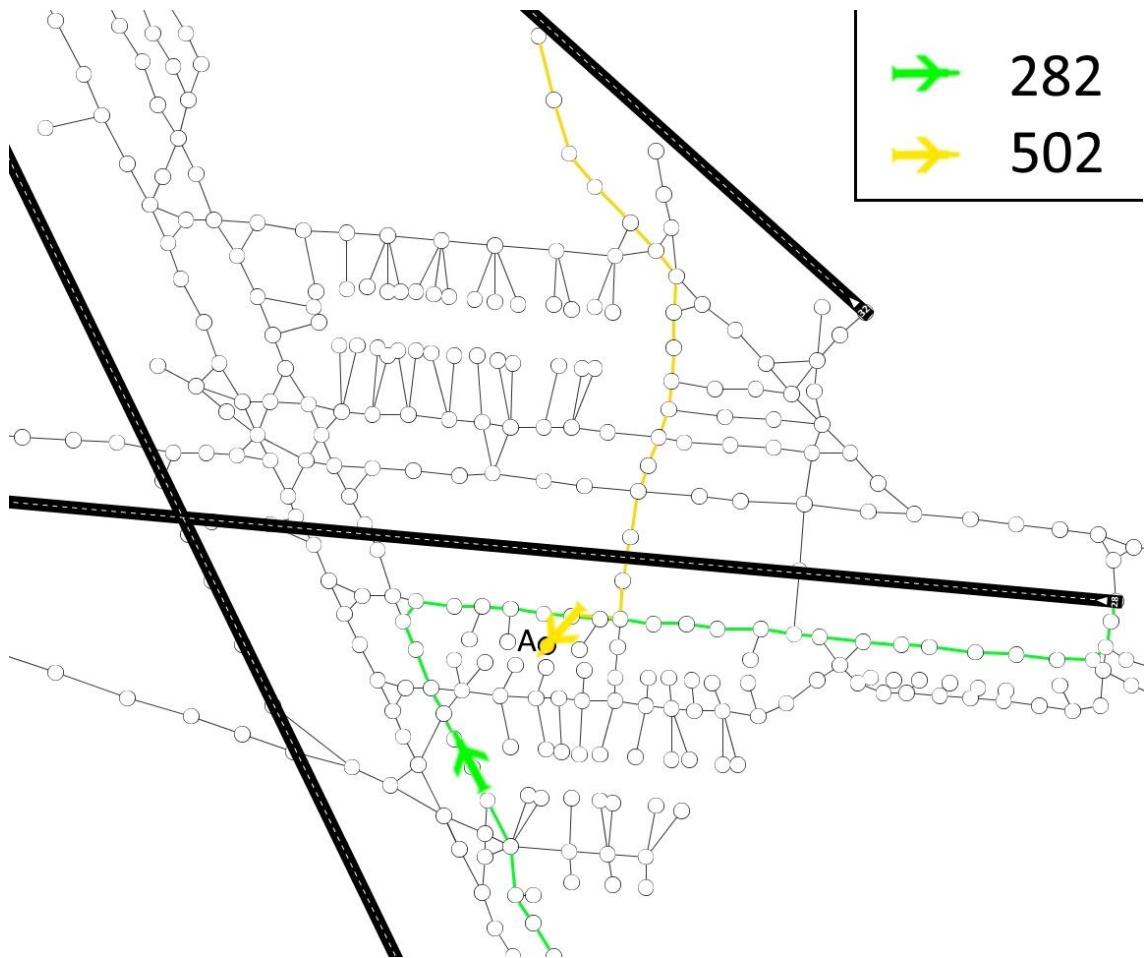


Figure 6.11: An example in Zurich airport where sufficient number of taxiways ensure that there is no delay

Furthermore, the number of taxiways can have a strong and consistent effect on the types of delays that are observed in the airports that were tested in this chapter. More taxiways result in delays with a smaller duration. More aircraft use alternative paths and that leads to a smaller sum of the duration of all of the delays. In contrast having fewer taxiways decreases the number of delays, but delays last for longer. Aircraft are limited to fewer paths and the total duration of delays is increased.

Figure 6.11 and Figure 6.12 show a case from day 1 in Zurich airport where two aircraft that are not delayed can interact with each other when taxiways are removed. When the experiments were executed using the original Zurich graph, departing aircraft 282 (marked in green) pushes back and starts heading towards runway 28. Arriving aircraft 502 (marked in yellow) lands on runway 14 and heads towards stand A.



Figure 6.12: An example in Zurich airport where removing taxiways introduces a delay

Similarly figures 6.13, 6.14 and 6.15 show a case from day 1 in Arlanda airport, where introducing taxiways, significantly reduces the delay of a flight. Overlapping paths of departing aircraft, have been painted only with the colour of the leading departing aircraft, when the remaining path of an aircraft is the same with the one of the leading aircraft.

When the aircraft are routed using the original Arlanda graph, flight 416 lands on the runway 01R at 06:22:07 and has to park at stand A (see blue path at Figure 6.13). The shortest path for this trip is to use the south taxiway (bottom of the figure) that connects runway 19L/01R with the rest of the airport/terminals. However, this path is constantly occupied by other departing aircraft. As Figure 6.13 shows, the south taxiway (part of the shortest path for aircraft 416) is allocated to aircraft 400 (see red path), which needs to be at runway 19L at

06:30. The same taxiway is then used by aircraft 399 (green aircraft) that also needs to be at runway 19L at 06:33. Furthermore, the same taxiway is allocated back to back to aircraft 398 (marked in cyan), 397 (marked in yellow) and 396 (marked in magenta), which all follow a similar path, starting from stands D, E, F respectively, and all need to reach the same runway 19L at 6:36, 6:38 and 6:40 respectively (see Figure 6.14). This makes it impossible for arriving aircraft 416 to use the south taxiway from runway 19L/01R and is forced to take a much longer path from the north taxiway (see blue path). The resulted delay due to taking a longer path is almost 5 minutes.

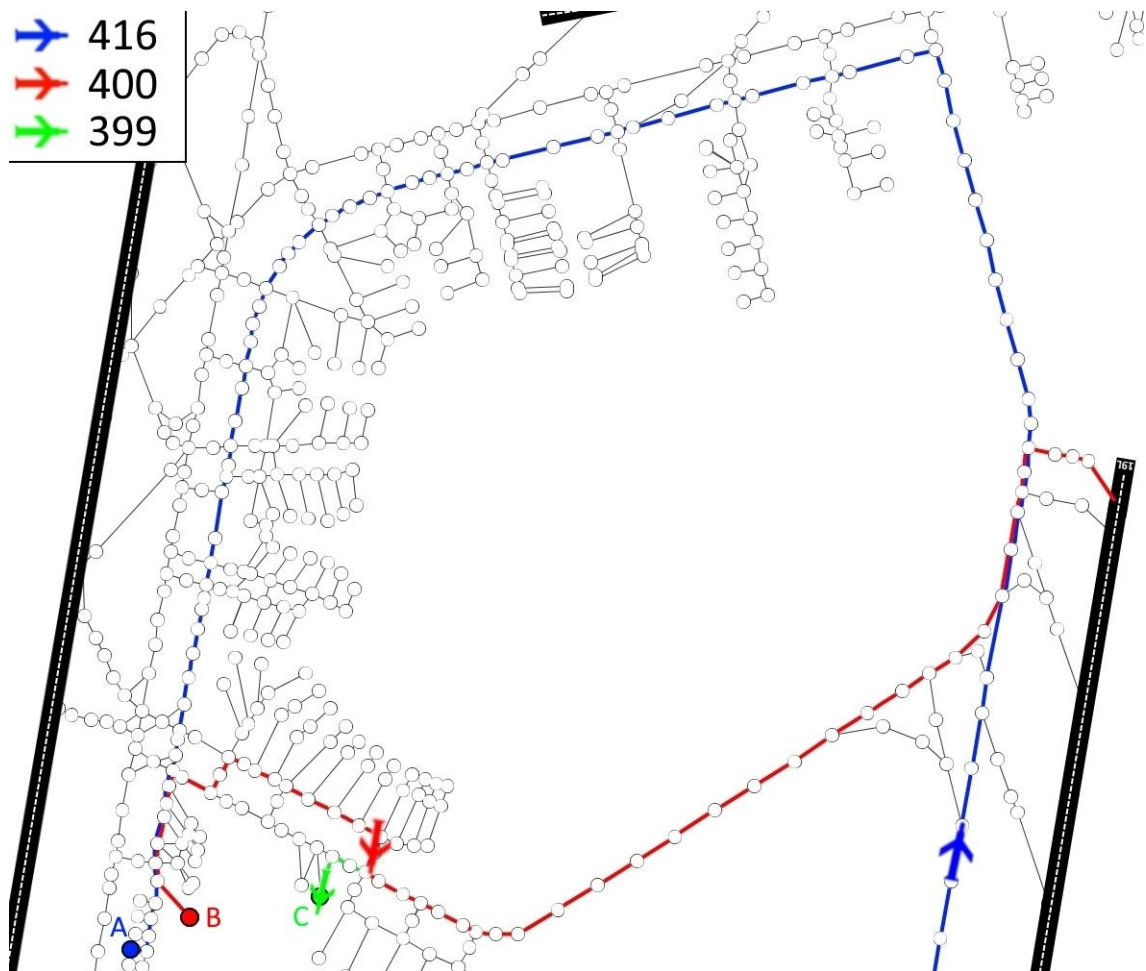


Figure 6.13: Example of aircraft that needs to take a longer path due to traffic (instance 1)

On the contrary, when the aircraft are routed around the Arlanda graph with extra taxiways, aircraft 416 can use the extra taxiway that is parallel to the taxiway that was previously occupied by multiple aircraft (connecting the runway 19L/01R with the rest of the airport on the south).

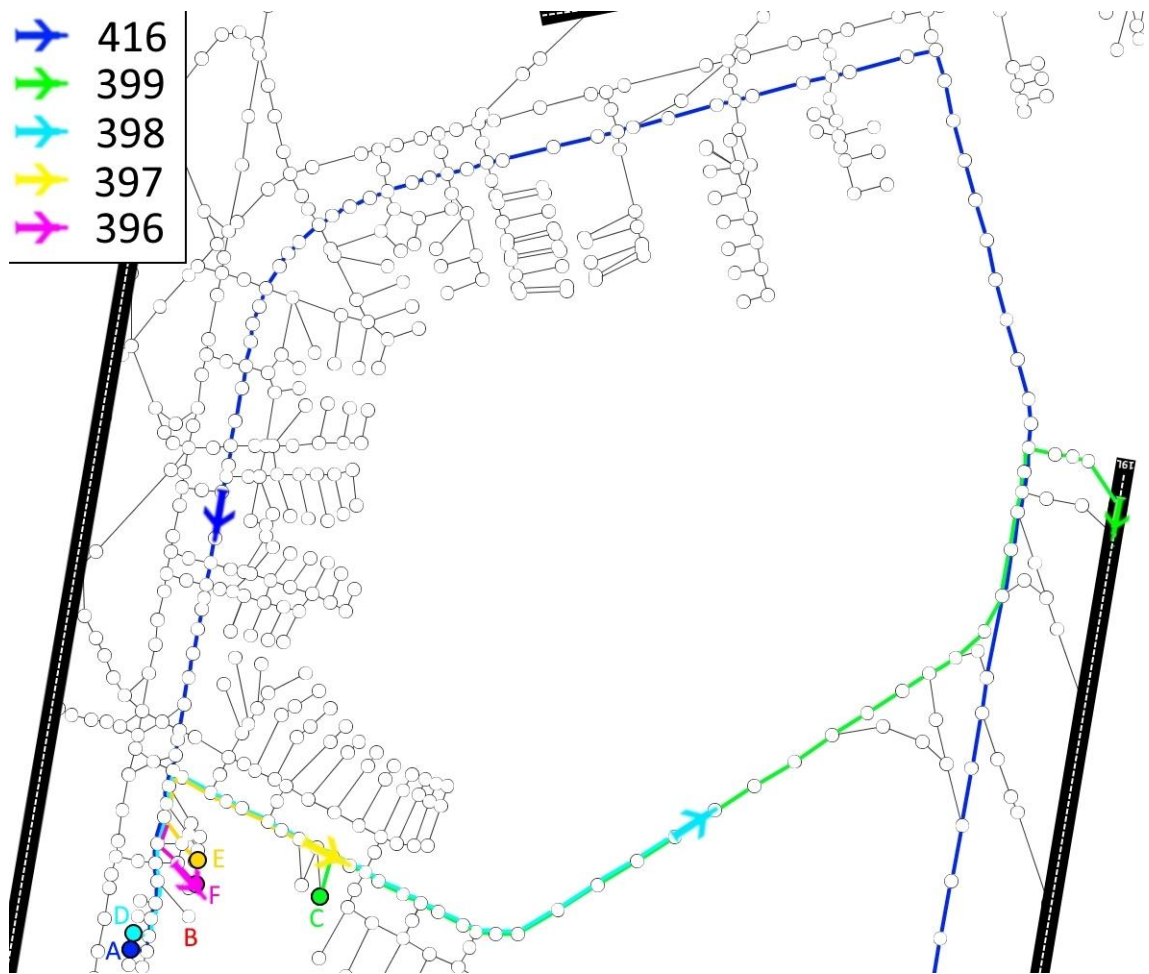


Figure 6.14: Example of aircraft that needs to take a longer path due to traffic (instance 2)

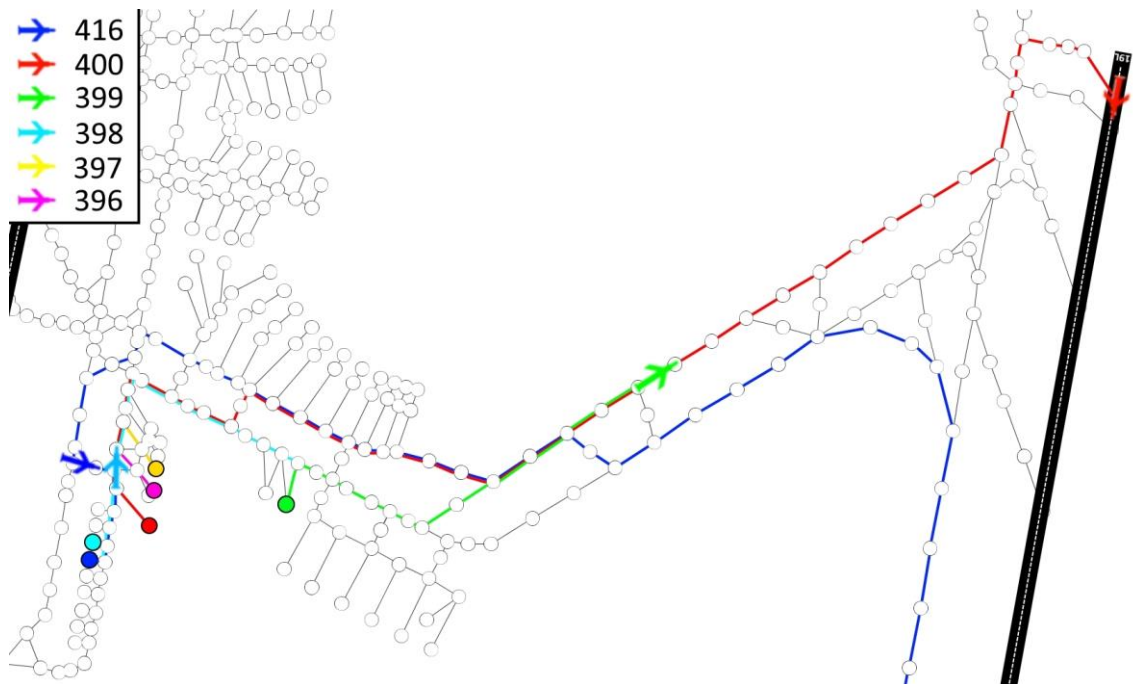


Figure 6.15: Example of aircraft taking a shorter path when a new taxiway is introduced

Now aircraft 416 can use the parallel taxiway (see blue path in Figure 6.15) avoiding departing aircraft 400 and 399. Eventually aircraft 416 only needs to wait for a few seconds for aircraft 398 to use a small part of its path before it can park at stand A. Aircraft 416 is not affected in this case by aircraft 397 and 396 since it has parked before any of them completes its pushback process. The introduction of the taxiway resulted in a reduction in the delay of aircraft 416 by 4 minutes and 22 seconds.

6.5 Conclusions

In this chapter, the effects of airport layout in the delays that can occur at an airport were investigated. QPPTW algorithm was used for routing the aircraft, which is an algorithm that can consider, and make use of, multiple paths. This algorithm routes aircraft sequentially and respects the movements of previously routed aircraft. The proposed model that was used (see Chapter 5) takes into consideration a more realistic movement of aircraft by implementing parameters that are known to cause or affect delays such as the pushback process (that was explicitly modelled) and the take-off sequence of aircraft (that was used as an input).

The experiments were executed for four different airports that were converted to graphs. Two - similar in number of passengers - airports that had different layout characteristics in terms of number of taxiways and connectivity between them, were initially chosen. These differences were then decreased by making two intermediate airport graphs - an airport graph similar to Arlanda but with more taxiways, and an airport similar to Zurich but with fewer taxiways.

The results showed that airports with more taxiways and more connections between them can provide a broader number of alternative paths. Even though a compact airport can have more congestion, using an alternative path for an aircraft is more commonly preferred than waiting, as the extra delay is shorter than waiting for a busy taxiway to clear. It was found that, by adding taxiways to strategic areas where bottlenecks are more likely to occur, the total duration of delays could be significantly decreased. Furthermore, alternative paths were then more likely to be used even though they might need an aircraft to travel longer distances on the ground. This can reduce the total travel time of the aircraft and can also reduce the total fuel consumption. Moreover, removing taxiways from an interconnected airport such as Zurich can introduce bottlenecks and increase the total delays.

Having a compact airport layout that has many short and interconnected taxiways increases the number of delays as well as the duration of the delays, but overall, it will reduce the total taxi time. Even if a physically larger airport such as Arlanda airport improves its infrastructure

by adding taxiways in areas where bottlenecks can occur, it still cannot compete in terms of total taxi time to a more compact airport even if this airport has fewer taxiways. A small taxi duration time provides less fuel consumption and CO₂ emissions. A routing and scheduling algorithm that takes into consideration all of the available paths when routing aircraft, can easily handle the increased complexity and route all the aircraft by minimising the total taxi time.

Considering the Gate Allocation Process

7.1 Introduction

Even though the ground movement process and the gate allocation process affect one another, in most cases they are solved separately. This chapter focuses on integrating the ground movement process with the gate allocation process. Allocating aircraft to their stands is a process that usually happens before the day starts without taking into consideration any information from the ground movement process. Regarding the ground movement problem, aircraft are routed using as a constant the stands where the aircraft start their journey from or park. As was mentioned earlier, in Chapter 4, many delays can and usually do happen close to the stands due to the pushback process. The area where departing aircraft pushback to, as well as the taxiways that arriving and departing aircraft use, depend on the stand that they are assigned to. This makes it necessary for both the routing process and the gate allocation process to be considered when the problem is solved in order to have a better and more accurate solution. Solving each problem optimally without considering one another, will most likely lead to a solution that is not optimal for the overall problem.

The main purpose of this research is to build a model that will provide a better and more intelligent gate allocation process that will take into consideration the delays that can happen for every combination of allocated aircraft that is considered. The objective function of the model (apart from considering the typical parameters of the gate allocation process) will also include minimising the delays that happen when the aircraft are moving on the ground.

The proposed framework includes two optimisation systems developed in collaboration. The first system uses a process which was designed by Dr. Neuman - and is described in Section 7.2 - to solve the gate allocation problem of aircraft and the second system solves the routing problem of the aircraft on the ground using the methodology developed in this thesis. This

chapter extends the collaborative framework which was described in the 4th and 5th chapter of Neuman's PhD thesis (2015). Importantly, this chapter will present how these two distinct systems can work together and communicate with each other in order to provide a better overall solution in a realistic timeframe. Furthermore, it will be shown that including the ground movement process of aircraft can result in a gate allocation solution with less duration of delays when the aircraft are routed.

The basic QPPTW algorithm will be used for the routing process of aircraft, as has been described in Chapter 4 where the pushback process is explicitly implemented in the routing process. A fast algorithm that can find all of the combinations of aircraft that can interact with each other, has also been developed and is presented.

In the next section (Section 7.2) the previous work on the integration between the gate allocation and the ground movement problem are presented. In Section 7.3 the gate allocation problem is described as well as the gate allocation model that was used for this integration. Section 7.4 explains the way that the conflicting aircraft are identified and how the algorithm for this process works. Furthermore, the changes that were made to the routing algorithm for this integration are also described. Section 7.5 describes the issues that arise when the routing process is implemented in the integrated model. Section 7.6 presents the integration framework and how the two models communicate. Moreover, the way that the ground movement process feedback is included in the gate allocation model is described. Section 7.7 presents the results after executing the experiments in a model of Manchester airport using real data for 7 days. Section 7.8 improves the robustness of the model and the effects of the increased robustness are tested after executing further experiments. Finally, the chapter summarises up the work in Section 7.9.

7.2 Previous Work on Integration with the Ground Movement Problem

Despite the variety of aspects and different proposed algorithms for solving the gate allocation problem, only a few models combine the gate allocation problem with the routing of the aircraft on the ground. The gate allocation problem could also be combined with the bus planning problem (Diepen et al. 2009) however in this thesis we will only consider integration models with the ground movement problem.

Kim et al. (2009) provide one of the first models that tries to integrate the gate allocation problem with the ground movement problem. In particular they try to reduce the conflicts that can happen where the aircraft park near each stand by creating a simulation tool. First a MILP

model is used to allocate aircraft to stands and then the results are compared with the allocations of a heuristic method. Kim et al. (2010) extended the above research by introducing the reduction of the total time the passengers have to spend at the terminal as an extra objective.

Cheng (1998) introduces the conflicts of the aircraft that can happen during the ground movement process of aircraft to the gate allocation problem as a factor that should also be considered. Cheng studies the ground movement of the aircraft and the conflicts that occur near stands. He presents a network based simulation method in order to approach the problem and to reduce conflicts and consequently minimise delays.

Kim and Feron (2014) introduce a queuing departure simulator. A robust gate assignment is presented that reduces the conflicts near stands using departure metering (controlling the number of aircraft that are moving around the airport at any time).

A hybrid gate allocation solution approach that can take into consideration the delays that can happen while aircraft move around the airport as a solution to the gate allocation problem is suggested by Guclu and Cetek (2013). It proposes an algorithm that assigns the available stands to the arriving aircraft and offers better delay management while minimising the delays that can happen during the ground movement of aircraft.

Neuman and Atkin (2013) suggest a gate allocation method that considers the taxiway conflicts that might occur near stands. Real data was used for the approach and taxiway information is used early in the allocation process for a more effective gate assignment. A new constraint is added to this model. This constraint considers the reduction of the number of aircraft with similar routes that are moving near the stands area and could result in a conflict. The model was based on a Mixed Integer Programming formulation and the objective function consists of the time gaps, airline preferences, reduction of the number of stands that are assigned to remote stands and the appropriate use of the space at the stand. The number of allocations that can produce a conflict are reduced but the disadvantages of this method are the extensive calculation time and the increase of the number of flights that need to be towed.

More recent research presents a hybrid dynamic method that integrates gate allocation with the routing process of the aircraft (Guclu and Cetek, 2017). The solution model allocates aircraft to parking points while considering the minimisation of taxi time and delays. It is also designed to provide real time new routes and parking points. For the application of the model a fast-time simulation tool was used, and the applied algorithm used real data. Moreover, all of the important capacity and traffic constraints were considered. The results of the hybrid

dynamic method show a decrease in delays that happen during the ground movement process as well as a decrease in taxi time.

7.3 The Gate Allocation Problem

The gate allocation problem consists of allocating aircraft to airport gates or stands by taking into consideration the preferences or constraints of the airline and the airport, as well as the availability and the size of the stand (in order to be able to serve the appropriate size of aircraft). There has been extensive research in this area and there are many models that suggest how to solve this problem as were presented in Chapter 2.

In summary, the gate allocation process is solved using a mixed integer programming model with a multi-objective function. The model was developed by Neuman and is described in the 4th and 5th chapter of her PhD thesis (2015). In this section, this model is going to be briefly explained in order to examine the way that this model takes into consideration the ground movement process and how this may affect the integrated model. First the notation that was described in Neuman's PhD thesis will be presented and then the constraints and objective function are described and examined.

7.3.1 Definitions of the Variables for the Gate Allocation Model

Table 7.1 show the notation that was used to describe the gate allocation model. In order to make it easier for the reader, the notation and definitions for the variables that were used below are the same to what was used in the PhD thesis of Neuman (2015). Full details of this part of the model and the aims and objectives of gate allocation can be found in that thesis.

Table 7.1: Table of definitions for the gate allocation model

F	Set of flights
n	Total number of flights in F
i, j	Flight indices $i, j \in \{1, \dots, n\}$
$f_i \in F$	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 $k, l \in \{1, \dots, m\}$

$g_k \in G$	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
$maxFreq$	Function that returns the maximum value of all frequencies ($freq_{f_i, g_k}$)
$size(f_i)$	Function that returns the size of flight f_i
$biggestGateSize$	Function that returns the size of the largest gate of the terminal
$maxSize(g_k)$	Function that returns the maximum size of the aircraft that can park at gate g_k

7.3.2 Constraints

The model implements the following constraints:

$$X_{g_k, f_i} = 0, \forall f_i \notin F(g_k), \forall g_k \in G \quad (7.1)$$

$$gap_{g_k, f_i, f_j} = \begin{cases} e_{f_j} - l_{f_i} & \text{if } e_{f_j} \geq e_{f_i} \\ e_{f_i} - l_{f_j} & \text{if } e_{f_i} \geq e_{f_j} \end{cases}, f_i, f_j \in F(g_k), g_k \in G, f_i \neq f_j \quad (7.2)$$

$$gap_{g_k, f_i, f_j} = \begin{cases} e_{f_j} - l_{f_i} & \text{if } e_{f_j} \geq e_{f_i} \\ e_{f_i} - l_{f_j} & \text{if } e_{f_i} \geq e_{f_j} \end{cases}, f_i, f_j \in F(g_k), g_k \in G, f_i \neq f_j \quad (7.3)$$

$$\left. \begin{array}{l} U_{g_k, f_i, f_j} \geq X_{f_i, g_k} + X_{f_j, g_k} - 1 \\ U_{g_k, f_i, f_j} \geq 0 \end{array} \right\} \forall f_i, f_j \in F(g_k), \forall g_k \in G, SG < gap_{g_k, f_i, f_j} < LG \\ 0 \leq SG \leq LG \quad (7.4)$$

$$U_{g_k, f_i, f_j} = 0, \forall (f_i \text{ or } f_j) \notin F(g_k), \forall g_k \in G \quad (7.5)$$

$$\sum_{k=1}^{m+1} X_{g_k, f_i} = 1, \forall f_i \in F \quad (7.6)$$

$$X_{f_i, g_k} + X_{f_j, g_k} \leq 1, \forall f_i, f_j \in F(g_k), \forall g_k \in G, gap_{g_k, f_i, f_j} \leq SG, \\ SG \geq 0 \quad (7.7)$$

$$X_{a, g_{hist_a}} = 1, \quad \forall a \in OVERNIGHT \quad (7.8)$$

$$X_{f_i, g_k} + X_{f_j, g_l} \leq 1, \forall f_i \in F(g_k), \forall f_j \in F(g_l) \forall (g_k, g_l) \in Sh \quad (7.9)$$

$$X_{f_i, g_k} + \sum_{g_k \in GR} \sum_{f_i \in C_d(f_i)} X_{f_i, g_k} \leq NC, \forall f_i \in F(g_k), \forall g_k \in GR, \forall GR \in \{GR_1, \dots, GR_z\} \quad (7.10)$$

The first constraint (Equation 7.1) constrains the usage of a gate to aircraft that fit (aircraft are not bigger in size than what the gate can accommodate). The second constraint (Equation 7.2) specifies the required (minimum) time gap between two aircraft that use the same gate (the one after the other).

Equation 7.3 and Equation 7.4 define the time gap that two aircraft (f_i and f_j) that are allocated to the same gate (g_k) should have and Equation 7.5 indicates if the two aircraft are allocated in the same gate. Equation 7.6 ensures that each flight can be allocated to only one gate.

Moreover, two flights cannot use the same gate at the same time (Equation 7.7), and the aircraft that stay at the airport overnight need to be allocated to gates that are used for overnight stay (Equation 7.8).

Another constraint was added for prohibiting gates being used simultaneously in cases where a large aircraft does not allow the use of a neighbouring gate (Equation 7.9).

Finally, a constraint was added for avoiding conflicting arriving with departing aircraft based on the times that they were most likely to park at the gate or initiate the pushback process respectively (Equation 7.10).

7.3.3 Objective Function

$$\begin{aligned} Min \quad & \sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^m p_{f_i, f_j} U_{g_k, f_i, f_j} + \sum_{i=1}^n \sum_{k=1}^m r_{f_i, g_k} X_{f_i, g_k} \\ & + \sum_{i=1}^n \sum_{k=1}^m a_{f_i, g_k} X_{g_k, f_i} + NC + \sum_{i=1}^n duX_{f_i, dummy} \end{aligned} \quad (7.11)$$

$$p_{f_i, f_j} = \frac{LG}{(gap_{f_i, f_j})} \text{ if } SG < gap_{g_k, f_i, f_j} < LG \quad \forall i, j \quad (7.12)$$

$$r_{f_i, g_k} = \frac{maxSize(g_k) - size(f_i)}{biggestGateSize} \quad \forall i, k \quad (7.13)$$

$$a_{f_i, g_k} = 1 - \frac{freq_{f_i, g_k}}{maxFreq}, \quad maxFreq \neq 0 \quad (7.14)$$

The objective function of the model takes into consideration several parameters and is given by Formula 7.11.

The first element is the time duration a gate is unoccupied between flights that are allocated in the same gate. Maximising this duration (Equation 7.12) makes the solution more robust as there is less chance for a conflict between two aircraft to happen, in case the aircraft allocated to the gate first is delayed or the aircraft that is allocated to that gate second arrives early.

The second element is the size of the gate (Equation 7.13). This size should match the size of the aircraft and using a gate that is larger than the size of the aircraft is avoided. Unoccupied large gates can be used by more aircraft and this affects the robustness of the solution as well.

The third element is related to the preferences of the airline, which tries to match the gates with flights of a specific airline in order to minimise the movement of personnel and equipment between gates (Equation 7.14).

The fourth element (NC) is related to the number of aircraft that are using gates that belong

to the same group. This variable considers flights that are allocated in gates that are in the same area (in the same group) and are more likely to produce a conflict during the pushback process and/or when moving on the area near the gate.

The fifth element is minimising the use of remote stands, where passengers need to arrive at a gate and then have to be transferred by a bus to the remote stand. These are areas where aircraft can park, usually separated from the main terminals.

There are more elements in the objective function with a very small weight that are not included in the Formula 7.11. These elements have a miniscule effect on the objective value. One of them, however, gives a slight priority to the gates that are closer to the runway.

7.3.4 Effects on the Ground Movement Process

As some of the parameters of the objective function and some of the constraints show, the routing process is partially taken into consideration when the gate allocation problem is solved in this model. This model has been designed to take into considerations delays that can happen during the ground movement process especially in the areas near the stands (see Chapter 5 of Neuman's thesis, 2013)

First, in the objective function, the use of gates that belong to the same “gate group” (*NC*), by aircraft that are operating in the same time gaps is discouraged. This element has been added in order to reduce the probability of two aircraft interacting with each other which can result in a delay. If two aircraft are not in the same area at the same time they cannot produce a delay. This is particularly important close to the gates, as there is a limited manoeuvrability and aircraft that are pushing back and turn on their engines can be the cause of long delays. Adding this element (*NC*) implicitly reduces the number of possible delays that can happen during the pushback process of the aircraft.

For the same reasons (limited manoeuvrability and delays that can happen close to the stands), the gap between aircraft that are allocated in the same gate is maximised. This element in the objective function (Equation) aside from improving the robustness (since any delay can result in the two aircraft being at the same stand at the same time) can potentially improve the routing process as well. If the aircraft that are assigned at the same stand (one arriving and the other one departing) have a large gap between them, the probability of them interacting along their path is decreased. On the other hand, if the two aircraft meet close to the stand, where aircraft may have limited manoeuvrability, it is more likely for large delays to occur.

Another element that can affect the ground movement process is that the stands that are closer to the runway are preferred (last element). Even though this element has a small weight, it

can potentially reduce the total travel time of the aircraft assuming that the duration of delays is not considerably increased due to favouring these stands.

Finally, potential conflicts between aircraft are also avoided by constraining the use of nearby stands so an arriving aircraft does not park at a stand close to where an aircraft is performing the pushback process at that time (see Equation).

These parameters in the objective function and the constraints of this model, show that the ground movement process is implicitly considered when the gate allocation process is solved. However, conflicts and delays during the ground movement process of aircraft can still happen. This chapter integrates the gate allocation with the ground movement process explicitly considering the delays that can happen during the routing process of aircraft when the gate allocation process is solved. It is expected that even though there will be a low number of delays when the problem is solved by the gate allocation solver alone, considering the delays that can happen when the ground movement problem is solved, will decrease the number and the duration of delays.

7.4 Solving the Ground Movement Problem and Finding Conflicting Aircraft

In this section, the way that the combinations of aircraft that can cause delays to one another is presented. This process is vital for integrating the ground movement problem with the gate allocation problem as it provides the necessary feedback to the gate allocation solver after the routing process is considered. First the notation that was used are defined, then the way conflicting combinations are found is explained and finally the algorithm that finds the combinations of aircraft that conflict with each other is presented.

7.4.1 Definitions of the Variables

Table 7.2 presents the new variables that were used to describe the framework.

Table 7.2: Table of definitions for finding the conflicting combinations of aircraft

P	The set of all of the flights routed before flight f_i sorted in a decreasing order based on the start time of each flight
$p \in P := \{1, ..., f_i - 1\}$	A flight that is routed before aircraft f_i
C	The set of combinations of conflicting flights
$c_i \in C := \{1, ..., C \}$	A combination of conflicting flights

$f_I(c_i)$	Function that returns the (chronologically) first flight of the conflict c_i
s_f	The starting time for aircraft f in the datasets
e_f	The time aircraft f reaches its destination
l_f	The latest time that aircraft f would reach its destination in the scenario where it is delayed the most
T_f	The total taxi time of flight f
m_f	The minimum time that it takes for an aircraft f to reach its destination on the quickest path, assuming all edges are unused by any other aircraft.
Insert	Function that inserts aircraft (or sets of aircraft) that conflict with each other to a set C
runPair (p, f)	Function that returns the delay that is produced when aircraft p and f are routed together on an empty graph

7.4.2 Finding the Conflicting Combinations of Aircraft

In order for the gate allocation model to allocate aircraft by taking into consideration the delays that arise for each allocation that is considered, it is important to know which combinations of aircraft can interact with each other. All of the aircraft combinations that cause delays to one another during the routing process are identified using the Algorithm 7.1 (see Section 7.4.3) and are stored in a “conflicting combinations” list. Each combination of aircraft has a cost that is associated with that combination, which is the sum of the delays that is caused to each aircraft f (see Equation 7.15).

$$\text{Combination cost for } c_i = \sum_{f \in c_i} (T_f - m_f) \quad \forall c_i \in C \quad (7.15)$$

Every time an aircraft is routed, all of the aircraft that are moving on the airport are considered as potential conflicts. In order to find out if two aircraft are delaying one another it is necessary to route just the two of them on an empty airport, so no other aircraft can affect the duration of the delay. Each aircraft f has a minimum amount of time that it needs to reach its destination (m_f). If it takes longer than the minimum time for an aircraft to reach its destination, this denotes that there is a conflict since something delayed it. However, not only two aircraft can cause delays to one another. Sometimes interactions between three or more aircraft can cause delays to each other. For this reason, combinations of any number of aircraft can be considered for potential delays (combinations of three aircraft, four, etc.). In order for a number of aircraft to be considered as a “conflicting combination” every aircraft (except

from the first aircraft) need to be delayed by another aircraft of this combination without having a subset of aircraft that do not affect or are affected by any aircraft from the rest of the set. This means Inequality 7.16 should hold every time all of the aircraft $f \in c_i$ are routed on an empty graph.

$$T_f - m_f > 0 \quad \forall f \in (c_i \setminus f_1(c_i)) \quad (7.16)$$

Figure 7.1 shows an example of three aircraft that interact with each other. Aircraft f_2 is delayed by aircraft's f_1 pushback process (in node A). This results in aircraft f_3 being delayed by aircraft f_2 (in node B - see part 1 of Figure 7.1). If aircraft f_2 was not delayed, as it is demonstrated in part 2 of the same figure, aircraft f_2 and aircraft f_3 would not interact. Even though aircraft f_2 and f_3 share partially the same path, aircraft f_2 will arrive at node B first, early enough to avoid an interaction with aircraft f_3 . Similarly, if aircraft f_2 was not routed, aircraft f_1 and aircraft f_3 would not interact (see part 3). In this case, the combinations (sets) of aircraft that interact with each other are $[f_1, f_2]$ and $[f_1, f_2, f_3]$. Combinations $[f_2, f_3]$ and $[f_1, f_3]$ do not cause a delay to one another and for this reason they are not considered as a conflicting combination even though they can be part of a conflicting combination.

The minimum time that is needed for an aircraft to reach its destination was found by routing each aircraft from each stand to the runway and vice versa, on an empty airport. This provides a set of time durations that denote the necessary time that it takes for any aircraft to reach its destination without any delay that is caused by other aircraft. The QPPTW algorithm was used for routing any pair or combination of aircraft. Each combination is executed on an empty airport so that if a delay occurs, the only aircraft that can cause the delay are the ones that are tested.

The algorithm that finds all of the possible combinations of conflicting aircraft (see next subsection and Algorithm 7.1) is executed after each aircraft is routed. For each routed aircraft (say aircraft f), the algorithm looks for combinations of aircraft that affect this aircraft (f). The algorithm searches for delays that are caused only by aircraft that have already been routed (before aircraft f). If aircraft f is part of a combination of aircraft that causes a delay, with aircraft that are scheduled chronologically after aircraft f , the combination of these aircraft will be found later on, when the latest (chronologically) aircraft is being considered. It is important to mention that since the QPPTW algorithm routes aircraft sequentially, only aircraft routed before aircraft f can cause a delay to aircraft f . This way, all of the combinations of aircraft that can cause a delay to each aircraft are found. When the combinations of aircraft that can cause a delay to the last aircraft are found, all of the conflicting combinations of aircraft for this allocation are found.

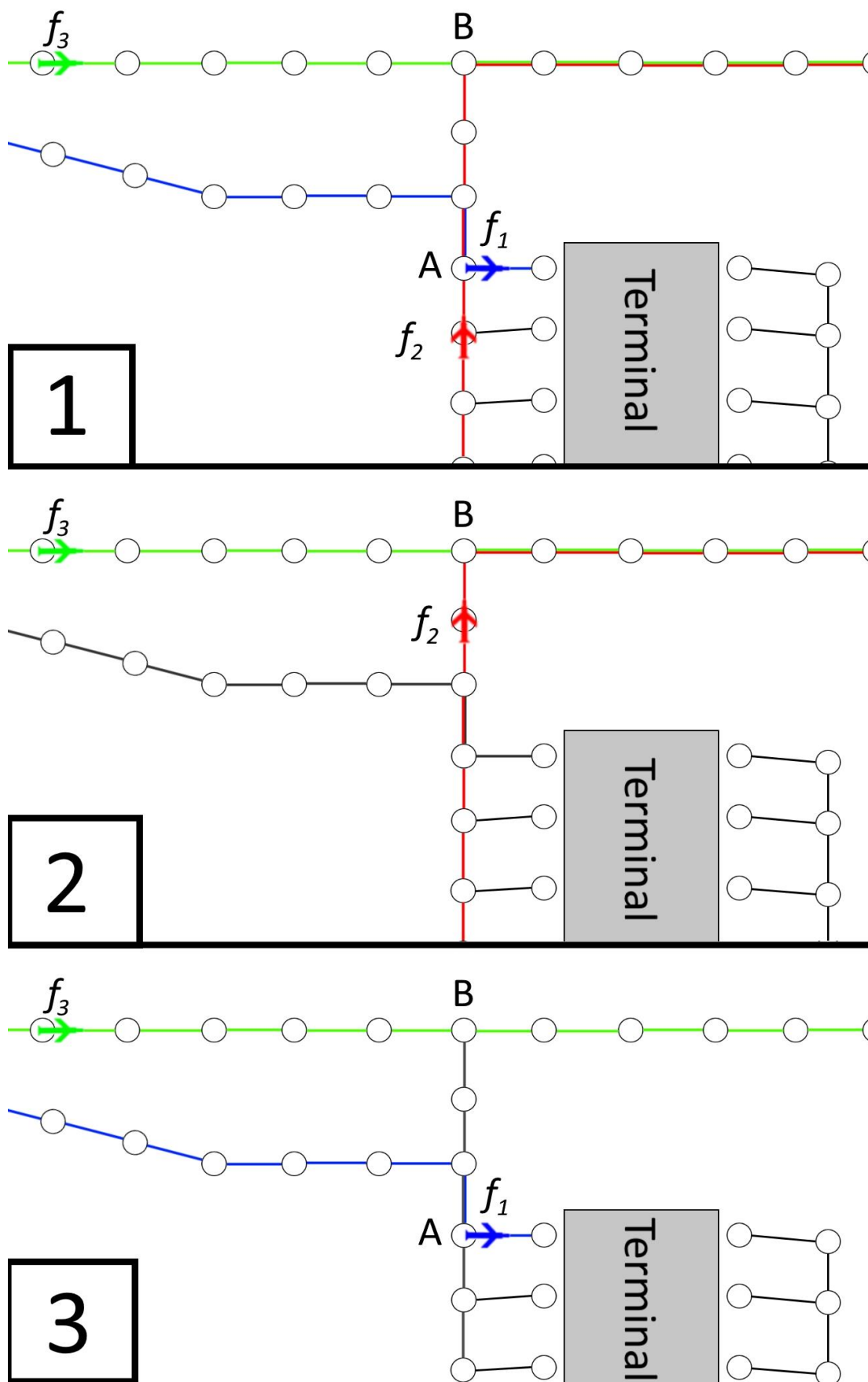


Figure 7.1: Example of three aircraft interacting with each other

7.4.3 The Algorithm for Finding the Conflicting Combinations of Aircraft

Finding all of the aircraft that can interact with each other, can quickly become a process that requires significant computational time if all of the combinations of aircraft during a day have to be tested. On an average day, more than 500 aircraft need to be routed. Considering each combination that can arise from all of the aircraft on a day is computationally challenging.

In order to reduce the search space, combinations that cannot possibly cause a delay to one another are not considered. The first way to reduce the search space for conflicting aircraft, is to consider aircraft that are moving on the airport at the same time as the aircraft under consideration is moving. If a previously routed aircraft (p) arrives at its destination (including any delays) before the aircraft under consideration (f) starts its journey, then it is impossible for these two aircraft to interact. This drastically reduces the search area as the number of aircraft that move at the same time in an airport is much lower than the number of aircraft that move around during the whole day.

In order to guarantee that a previously routed aircraft p will not be moving in the airport at the same time as aircraft f , the latest time that aircraft p will arrive at its destination is considered (see line 2 in Algorithm 7.1). The latest time that aircraft p can arrive at its destination is found after finding all of the conflicts that cause a delay to aircraft p . Since only aircraft that are routed before aircraft p can cause a delay to aircraft p , the latest time that aircraft p will arrive at its destination is known before aircraft f is considered. If aircraft f starts moving after this time, then aircraft p cannot directly cause a delay to aircraft f . It is important to mention that aircraft p can still cause a delay to another aircraft that can potentially cause a delay to aircraft f , even though aircraft f and p do not move at the same time. This case however, is found after examining the aircraft that can cause a delay to aircraft that directly cause a delay to the aircraft under consideration (f). Aircraft that can directly cause a delay to aircraft f are only the ones that move around at the same time as aircraft f .

An algorithm (Algorithm 7.1) for finding the conflicting combinations of aircraft has been developed and is described below. Algorithm 7.1 can find combinations of conflicting aircraft - that contain any of the previously routed aircraft $p \in P$ - in four ways.

The first process (lines 3 to 5) searches all of the previously routed aircraft to find which ones are (or can be if they are delayed) moving at the same time as the one that is currently being routed (aircraft f). If there is an aircraft (p) that moves at the same time as aircraft f then it is examined to determine if aircraft p causes a delay to aircraft f . If a delay exists, the combination is added to the list of aircraft that conflict aircraft f (C_f).

Algorithm 7.1: Finding the conflicting combinations of aircraft

Input: A flight $f \in F$, a set of aircraft P routed before aircraft f

Output: A set with combinations of conflicting aircraft C_f that can cause flight f a delay

```
1  foreach  $p \in P$  do
2      if  $l_p > s_f$  then
3           $T_f = \text{runPair}(p, f)$ 
4          if  $T_f > m_f$  then
5               $\text{Insert } \{p, f\} \text{ into } C_f$ 
6          if  $\{p, f\} \notin C_f$  then
7              foreach  $c_f \in C_f$  do
8                   $T_f = \text{runAllocation}(c_f, p)$            /*Algorithm 7.2*/
9                  if  $T_f > m_f$  then
10                      $\text{Insert } \{c_f, p\} \text{ into } C_f$ 
11          if  $C_p \neq \emptyset$  then
12              foreach  $c_p \in C_p$  do
13                   $T_f = \text{runAllocation}(c_p, f)$            /*Algorithm 7.2*/
14                  if  $T_f > m_f$  then
15                       $\text{Insert } \{c_p, f\} \text{ into } C_f$ 
16                      if  $C_f \neq \emptyset$  then
17                          foreach  $c_f \in C_f$  do
18                               $T_f = \text{runAllocation}(c_f, c_p)$ 
19                              if  $T_f > m_f$  then
20                                   $\text{Insert } \{c_f, c_p\} \text{ into } C_f$ 
21          else
22               $\text{break}$ 
```

After the first process is completed, the second process (lines 6 to 10) initiates, where the algorithm searches for combinations of aircraft that cause delays (more than one aircraft causing a delay in aircraft f). The algorithm checks if aircraft that are already in the list of conflicting aircraft C_f interact with any other aircraft p routed before aircraft f regardless of whether aircraft p delays aircraft f . In this way combinations of more than two aircraft can occur.

The third process (lines 13 to 15), checks if the list of conflicting combinations C_p of an aircraft g that is routed before aircraft f , can cause a delay to aircraft f . This list (C_p) contains conflicting combinations (if there are any) of aircraft that were found in a previous execution of Algorithm 7.1 when aircraft p was the aircraft under consideration. This process takes into consideration aircraft that do not cause directly a delay to aircraft f and may not even be active (have reached their destination) when aircraft f starts its routing process, but affect a chain of aircraft that can eventually affect aircraft f .

Finally, the conflicting combinations C_p of aircraft routed before aircraft f are checked for potential conflicts with the conflicting combinations C_f of aircraft f (lines 17 to 20). This again finds aircraft that do not cause a direct delay to aircraft f but affect a chain of aircraft that can eventually affect aircraft f .

It is important to mention though that any combination added in the list of conflicting combinations, has to cause an added delay on aircraft f . Due to the structure of the QPPTW algorithm, it is only necessary to check for the previously routed aircraft, as the algorithm routes each aircraft sequentially, according to time (first come – first served). For this reason, an aircraft f' that routes after an aircraft f cannot cause any delay to aircraft f .

7.5 Implementation Issues

As was mentioned in the previous section, in order to ascertain if combinations of aircraft conflict with each other, it is necessary to solve the problem on an empty graph, for these aircraft alone. Solving the problem separately running the full ground movement program for each combination that needs to be tested, requires significant computational time, as many objects (instances of various classes) need to be built. The QPPTW algorithm is a fast algorithm that can route multiple aircraft swiftly. However, there are many processes that surround the QPPTW algorithm, which do not significantly affect the computational time if executed once when the program is initialised, but can accumulate a significant delay if executed multiple times. The processes include loading the data that is needed in order to make the initial graph (nodes, edges, distances, minimum traversal times between nodes), initialising the graph, finding the conflicting edges etc.

7.5.1 Adjusting the QPPTW Algorithm for Solving Side Problems and Running Combinations of Aircraft for Potential Conflicts

In order to keep the execution time to the minimum, the initialisation of the processes that are necessary for the QPPTW algorithm to run are executed only once. Every time combinations of aircraft are examined for delays; the problem is solved in the same graph as the one that the ground movement problem is solved. These two processes are executed in parallel, as the algorithm for finding the conflicting combinations of aircraft is run after each single aircraft is routed.

The QPPTW algorithm uses time windows in order to take into consideration other aircraft that have been previously routed. Time windows allow an aircraft to use an edge only when there is a wide enough timespan for the aircraft to traverse this edge. In order to use the same graph – and as an extension the same nodes and edges – each time new combinations of aircraft are tested for conflicts, another set of time windows was introduced in the algorithm. The new time windows - which will be called temporary time windows for convenience - are a separate set of time windows that mark the availability of the edge, only when Algorithm 7.1, the algorithm for “finding the conflicting combinations”, is used.

7.5.2 Testing Combinations of Aircraft for Potential Conflicts

Every time the “run allocation” command is executed in Algorithm 7.1, the following processes are executed (Algorithm 7.2):

Algorithm 7.2: runAllocation

Input: Two sets of conflicting aircraft C_1, C_2

Output: The sum of all delays that occur in this combination

```

1:  for each edge  $e$ 
2:      clear temporary windows
3:  end for
4:  Total delay = 0
5:  for each  $f \in C_1 \cup C_2$ 
6:      run QPPTW( $f$ )      /*Algorithm 3.1*/
7:      reconstruct path
8:      adjust temporary time windows
9:      if  $T_f - m_f > 0$  and  $f \neq f_1$ 
10:         Total delay +=  $T_f - m_f$ 
11:     else
12:         return 0
13:     end if
14: end for
15: if  $l_{f_n} < e_{f_n}$ 
16:      $l_{f_n} = e_{f_n}$ 
17: end if
18: return Total delay

```

As Algorithm 7.2 shows, the first step of the algorithm is to clear all of the temporary time windows. This ensures that the combination of conflicting aircraft is going to be executed in an empty graph. For each aircraft in the conflicting combinations list the QPPTW algorithm is going to be executed using the temporary time windows instead of the actual ones. The

path is then going to be reconstructed by QPPTW once the target node is reached. Then the temporary time windows are going to be adjusted, allowing for the QPPTW algorithm to take into consideration the aircraft routed in previous iterations. Moreover, if any of the aircraft under consideration - aside from the first one - is not delayed, the process stops and Algorithm 7.2 returns the value of zero, as in order for a combination to qualify as a conflicting combination, all aircraft involved (with the exception of the last one) need to cause a delay. Otherwise, the delay is added to the total delay variable which is the variable that Algorithm 7.2 returns when all of the aircraft under consideration are routed. Finally, before the process returns the total delay, the latest time that the last routed aircraft (if it is the aircraft under consideration in Algorithm 7.1) arrives at its destination is updated. This variable is updated so the latest possible time that an aircraft can move around the airport is known, before the next aircraft is routed (when the ground movement problem is solved). This information makes it possible to skip tests for aircraft that do not move at the same time around the airport (see Section 7.4.3).

7.6 The Integration Framework

In this section, the integration framework is described. First the new notation that has been used is defined, then the way that the two models - the ground movement and the gate allocation - were integrated is presented. Moreover, the way that the ground movement feedback is added to the gate allocation model is described and the stopping condition of the iterative process of the integration is defined. Finally, an extension for making the solution more robust is presented.

7.6.1 Definitions of the Variables

For convenience, the new variables and notation that will be used to explain various concepts in this section has been collected into a single table (Table 7.3). Since the integration model was developed in conjunction with another PhD student and in order to make it easier for the reader to better understand the integration framework, the notation (and definitions) for the variables that were used below are the same or similar to the notation that was used in the PhD thesis of Neuman (2015). The variables and the notation that were presented in Table 7.2 are still valid for this section.

Table 7.3: Table of definitions for the integration process

X_{g_j, f_j}	A decision variable, it becomes 1 if aircraft f_j is allocated to gate g_j , 0 otherwise
$rvar_i$	An indicator variable, becomes 1 if configuration $c_i \in C$ occurs in the allocation plan
$sc_i = \{c_k \in C : c_i \subset c_k\}$	A set that contains all the supersets c_k that contain the conflicting combination c_i
w_i	The weight of variable $rvar_i$ when fitted into the objective function of the gate allocation model

7.6.2 The Feedback Loop

The integration framework consists of three parts. The first part is the gate allocation process that was developed by Neuman (2014) and was briefly explained in Section 7.3. The second part is the routing process of aircraft and searching the combinations of aircraft that can interact with each other that was described in Section 7.4. The third part is the feedback loop which coordinates the information flow between the two processes that were previously mentioned.

Figure 7.2 shows a flow diagram of the integrated model. Initially the gate allocation solver is executed and the optimal allocation of aircraft to stands is found. This initial solution does not take into consideration the ground movement process. The allocation of aircraft to stands that results from the initial solution is then passed by the feedback loop manager to the ground movement process solver. The ground movement solver then routes all of the aircraft using as input the stands and the times that are provided by the gate allocation solver (using Algorithm 3.1 for arriving aircraft and Algorithm 5.1 for departing aircraft). Each time an aircraft is routed, the algorithm that finds the conflicting combinations of aircraft is executed (Algorithm 7.1). When all of the aircraft have been routed, all of the conflicting combinations of aircraft that were discovered by the ground movement solver, are passed by the feedback loop manager back to the gate allocation solver. These conflicting combinations are then added to the gate allocation model as extra constraints (see Section 7.6.3). If the stopping condition (see Section 7.6.4) is satisfied, the feedback loop manager will terminate the integration process. Otherwise, the updated list of conflicting aircraft will be passed to the gate allocation process all over again until the stopping condition is satisfied.

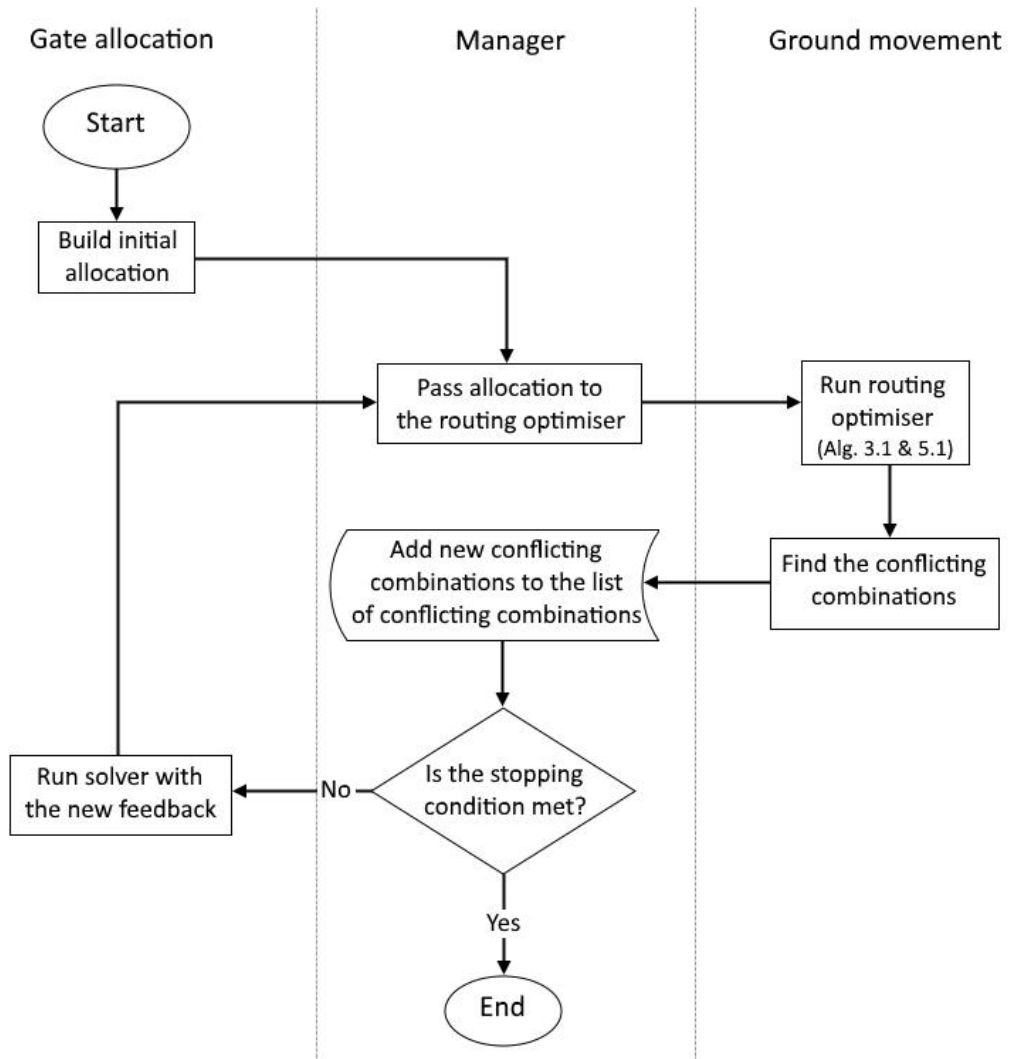


Figure 7.2: Flow diagram of the integrated model

The pseudocode of this process as described by Neuman in her PhD, is provided in Algorithm 7.3.

Algorithm 7.3: The integrated gate and route allocation algorithm

- 1: run the Gate Allocation Optimiser to obtain an initial allocation
 - 2: pass the initial allocation to the Manager
 - 3: run the Routing Optimiser for the initial allocation (Using Algorithms 3.1 and 5.1)
 - 4: obtain feedback information from the Routing Optimiser to the Manager
 - 5: **while** stopping condition not met **do**
 - 6: pass the routing feedback information to the Gate Allocation Optimiser
 - 7: run the Gate Allocation Optimiser to obtain a new allocation
 - 8: pass the new allocation to the Manager
 - 9: run the Routing Optimiser for the new allocation to obtain new feedback information
 - 10: pass the new feedback information to the Manager
 - 11: add the new feedback information to the previous feedbacks
 - 12: **end while**
-

Once the stopping condition is met, the allocation with the smallest total duration of delays is found. It is important to mention that an allocation may not always lead to a better solution for the ground movement process. The gate allocation solver does not know beforehand the total delay that will be produced once all of the aircraft are routed. It only takes into consideration combinations that are known to cause a delay and suggests the best allocation based on this information.

7.6.3 Adding the Ground Movement Feedback to the Gate Allocation Model

Each time the ground movement solver finds the conflicting combinations of aircraft, the list of “conflicting combinations” is passed to the gate allocation solver. The gate allocation solver uses the model that was developed by Neuman for her PhD thesis (2015) and was briefly described in Section 7.3. Each combination c_i of conflicting aircraft from the “conflicting combinations” C list is added as a constraint to the gate allocation model. Equation 7.17 describes the constraint (as was described in Neuman’s thesis)

$$\sum_{j=1}^{|c_i|} X_{g_j, f_j} - rvar_{c_i} - \sum_{c_k \in sc_i} rvar_{c_k} \leq |c_i| - 1, \quad (7.17)$$

$$\forall i \in \{1, \dots, |C|\}, \quad sc_i = \{c_k \in C : c_k \supset c_i\}$$

where $|c_i|$ denotes the number of aircraft that conflict with each other in combination $c_i \in C$, g_j denotes the gate of the allocation with index j and f_j denotes the flight of the allocation with index j . The constraint, will force the additional variable $rvar_i$ to have the value of 1 if the allocation of the aircraft in combination c_i is part of the solution of the gate allocation model and c_i is not already part of a larger combination of conflicting aircraft c_k which is being considered ($rvar_{c_k} = 1$). A weight w_i is used for variable $rvar_i$ in order to fit the variable to the objective function of the gate allocation model. The aim of this integration is to find an allocation that takes into consideration the routing process of aircraft, without compromising the other elements of the objective function that are used for solving the gate allocation process.

The weight w_i is equal to the cost of the conflicting combination c_i , which is the total duration of delay that the aircraft in this combination produce to each other (in a specific gate configuration). As Equation 7.18 indicates, the sum of all of the weighted (w_i) variables $rvar_i$ (as was described in Neuman’s thesis) is added to the objective function of the gate allocation model.

$$\sum_{i=1}^l (w_i * rvar_i) \quad (7.18)$$

7.6.4 The Stopping Condition

The integration process will be terminated when the ground movement solver stops providing new conflicting combinations of aircraft. This happens when the list of conflicting combinations C_f does not contain a new conflicting combination c' for any flight $f \in F$.

Each time the ground movement problem is executed, it tries to find new combinations of conflicting aircraft. The gate allocation model is then extended by adding these combinations as new constraints. The gate allocation solver then tries to find a solution where allocations of conflicting combinations of aircraft are minimised. If the ground movement problem does not provide any new combination of conflicting aircraft in the feedback list, then the integration model is stopped. If there are no new combinations to consider, the gate allocation solver cannot provide a new allocation that will lead to a better ground movement solution than the ones that are already provided.

7.7 Executing the Integrated Model

In this section, the results that were obtained after executing the integrated model are presented. The objective of the integration framework is to minimise the number and the total duration of delays that happen in the airport (during the ground movement process) after solving the gate allocation problem. For these experiments, each terminal was solved separately due to the complexity of the problem. This is a common practice when the gate allocation problem is solved, as it is rare that an aircraft is able to be assigned to gates from two (or more) different terminals. This is because airlines are usually associated with one terminal, often, to reduce resources that are needed (such as check in desks, personnel etc.). It is possible though, that aircraft from different terminals interact with each other during the ground movement process and solving the full gate allocation problem could provide a better solution. However, it has not yet been possible to find an optimal solution for the entire gate allocation problem in a reasonable timescale. Neuman in her PhD (2015) suggested the use of an alternative version of the system that implements a Receding Horizon decomposition in order to solve this problem. For this chapter however, only the basic gate allocation solver was used for the integration of the two systems and restricting the consideration to a single terminal eliminates the problem.

7.7.1 Experimental Settings

The experiments for this chapter were executed for Manchester airport as the developed model that solved the gate allocation problem was already implemented for this airport. Moreover, the datasets of Manchester airport were the only datasets available for solving the gate allocation process (previous datasets were adequate for solving the ground movement process of aircraft only). This airport has 3 terminals and 1 runway. Manchester airport was chosen for the variety of traffic characteristics that it has for each terminal. This provides a variety in the cases that are examined, especially for the gate allocation process as Neuman has highlighted in Chapter 3 of her PhD thesis (2015). 7 days of historic flight data was used for each terminal. The problem was solved independently for each terminal and each day (21 instances were solved). After executing the experiments, it was observed that the complexity of the instances differs significantly. More specifically, days 3 and 5 for terminal 1 and days 2 to 5 for terminal 3 were much harder for the gate allocation process to solve, in many cases exceeding 1 hour until a solution was found. On the other hand, for terminal 2 the integrated model was solved within a few minutes in most cases. More details about the execution times of the gate allocation solver can be found in Chapter 6 of Neuman's PhD thesis (2015).

7.7.2 Results

The results summarising the total delay and the number of conflicts are presented in Table 7.4. The first column shows the terminal and the day of the data that was used for the experiment. The second column shows the number of iterations that integrated model has performed (see line 5-12, Algorithm 7.3) for each instance and the number of the iterations that provided an improved solution in terms of total delays (in the parenthesis). The third column shows the total duration of the delays that happen in that instance when the aircraft are routed using the initial gate allocation solution (without taking into consideration the ground movement process) and the number of conflicts (inside the parenthesis). The fourth column shows the total duration of delays for the best performing allocation that was found by the integrated model as well as the number of conflicts (inside the parenthesis) for that allocation. The fifth column shows the improvement in the total duration of delays that happens when the ground movement process is considered. Finally, the sixth column shows by how much (as a percentage) the objective value - for the gate allocation problem only - is affected.

Table 7.4: Results after solving the gate allocation process while considering the ground movement

Terminal /Day	Iterations	Initial delay [s] (conflicts)	Final delay [s] (conflicts)	Improvement	Objective [%]
1 / 1	7 (5)	223 (10)	0 (0)	100%	0.62
1 / 2	15 (15)	203 (12)	0 (0)	100%	0.87
1 / 3	30 (28)	224 (13)	0 (0)	100%	1.20
1 / 4	18 (17)	274 (14)	6 (1)	98%	0.56
1 / 5	3 (3)	60 (2)	0 (0)	100%	0.02
1 / 6	18 (18)	276 (9)	0 (0)	100%	0.15
1 / 7	3 (2)	276 (9)	0 (0)	100%	0.30
2 / 1	10 (7)	53 (4)	0 (0)	100%	0.39
2 / 2	40 (35)	142 (7)	5 (1)	96%	1.16
2 / 3	49 (46)	130 (7)	5 (1)	96%	0.57
2 / 4	15 (6)	38 (4)	0 (0)	100%	0.51
2 / 5	49 (13)	43 (5)	0 (0)	100%	0.79
2 / 6	60 (58)	445 (11)	0 (0)	100%	3.75
2 / 7	12 (3)	39 (6)	0 (0)	100%	0.20
3 / 1	74 (55)	125 (13)	0 (0)	100%	1.14
3 / 2	120 (120)	398 (21)	2 (1)	99%	0.26
3 / 3	83 (40)	73 (16)	2 (1)	97%	0.08
3 / 4	194 (193)	468 (30)	0 (0)	100%	0.56
3 / 5	177 (157)	276 (19)	0 (0)	100%	0.82
3 / 6	62 (61)	185 (5)	0 (0)	100%	0.44
3 / 7	96 (89)	117 (11)	0 (0)	100%	0.39
Average	54 (46.2)	194 (10.9)	0.95 (0.24)	-	0.7
Total	-	4068 (228)	20 (5)	99.5%	-

As the results indicate, even without taking the ground movement process into consideration, the number of conflicts and delays that happen when the aircraft are routed - using the initial solution for the gate allocation problem - are moderately low. When the ground movement problem was solved, based on the initial gate allocation solution, the total number of delays was 228 and the total duration of delays was 1 hour and 8 minutes. This is lower than the delays that were observed when the ground movement problem was solved for Zurich and Arlanda airport, using real data (see Chapter 6). This is expected as this gate allocation model implicitly takes into consideration the ground movement process as was mentioned in Section

7.3. However, the low number and duration of delays also suggests that the results are affected by the fact that the problem is solved for each terminal separately.

Even though the number of initial delays is not high for each terminal, when the delays that can happen during the ground movement process are taken into consideration, there is a significant improvement in the total duration of delays when the integrated problem is solved.

For all of the 21 instances that were tested, the integrated process eliminated the delays or at least significantly improved the total duration of delays that happen when aircraft are routed, when comparing the delays that happen during the initial allocation to the final allocation that the integrated model has produced. On average, the total delay for all of the terminals over the week decreased by 99.5% when the ground movement process was taken into consideration. Overall, the total duration of delays decreased from 68 minutes to 20 seconds and the total number of delays decreased by 223, from 228 conflicts to just 5. For the majority of the instances (16 out of 21) the final allocation had no conflicts between aircraft and the worst performing allocation would produce only 6 seconds of delay for a full day on one terminal. This shows that taking into consideration the ground movement process can prevent all or at least most of the delays from happening when the aircraft move around the airport.

Furthermore, the number of iterations (see Algorithm 7.3) was on average around 54, which shows that there are many alternative allocations that do not drastically change the fitness of the objective value of the gate allocation problem, but provide a range of alternative input for the ground movement process. It is important to mention however, that not all of the iterations would provide an improvement in the total duration of delay. Out of the 54 iterations that would happen on average for each day and terminal, approximately 46 would provide an improvement in the total duration of delay. The gate allocation solver does not know beforehand if a certain allocation will provide less or shorter delays. The results however, indicate that the initial solution of the gate allocation problem is a competitive solution, providing more evidence that the gate allocation model will provide a relative good solution in terms of delays that can happen during the ground movement process. As mentioned in Section 7.3, the gate allocation model implicitly takes into consideration the delays that can happen close to the gates.

Finally, the objective function of the gate allocation problem was (in most of the cases) not significantly affected by the new solution. Out of the 21 cases where the gate allocation was improved, only 4 would increase the value of the objective function of the gate allocation model by more than 1%. The instance that was affected the most by introducing the ground movement process had its objective value increased (only for the gate allocation model) by 3.75%.

Eventually, how much the objective function of the gate allocation process should be affected depends on the importance of decreasing the delays that would happen during the routing process. Since the parameters of the gate allocation and the delays that can happen during the routing process cannot be compared in an absolute way, an airport may need to decide what is more practical or profitable for their case and adjust the weight (or limit the maximum value) of each parameter accordingly.

Figure 7.3 shows the delays that happen (on the left side axis) for terminal 1, day 2, when the ground movement problem is solved for each gate allocation provided by the integrated model, in contrast with the increase (as a percentage) of the objective value of the gate allocation solution compared to the initial gate allocation solution (on the right-side axis), for each iteration. As the graph shows, the allocations that are found do not always improve the delays that can happen in the ground movement process. Also, the solution of the gate allocation model, is not always worsening. This shows that changing the allocation of a flight may not always have the expected results when it comes to improving the delays during the ground movement process but overall it can produce multiple solutions that outperform the initial allocation when it comes to the delays that happen during the ground movement process. In this example, 16 allocations were found that would decrease the delays when the ground movement process is subsequently solved. The best performing one would decrease the delays from 4 and a half minutes to no delay at all.

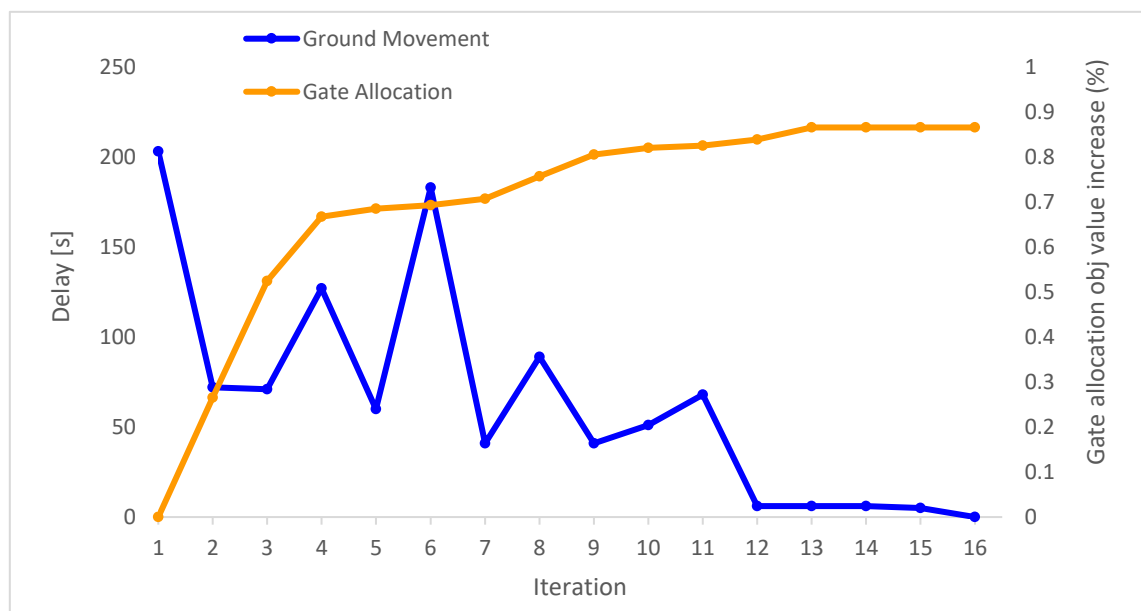


Figure 7.3: The duration of delays during the ground movement process and the increase of the objective value of the gate allocation model for each iteration, for terminal 1, day 2

7.8 Buffer Time Between Aircraft

The solution of the integrated model provides a solution that can take into consideration delays that can happen in the future when the aircraft are routed. However, in most large airports, the ground movement process is a complicated procedure where many unpredictable parameters can influence the timing of each movement. In some cases, the ground movement solvers take this unpredictability into consideration by adding extra buffer times (that forces a time gap between two aircraft) in each section of an aircraft's route. However, when the routing process is done in real time it is not necessary as the solution can be recalculated fast, and adding robustness can affect the throughput of aircraft that are moving around the airport.

On the other hand, a solution for the gate allocation process is something that cannot easily change if a problem occurs while the day is unfolding. Changing a gate allocation solution while aircraft are already using the gate according to the initial solution can require moving personnel and passengers to different gates which in many cases can be impossible and moreover moving aircraft to different stands which is impractical.

A robust solution in this case can prove to be more useful. Moreover, improving the robustness for the ground movement process when considering the gate allocation problem does not negatively affect the capacity of the airport to the extent it does for the ground movement problem. Once the gate allocation solver provides a robust solution, aircraft can be re-routed moments before they start their journey, without considering large buffer times between them.

In this section, a buffer time between aircraft is introduced in order to investigate how a more robust ground movement solution can affect the integrated process.

7.8.1 Implementing the Buffer Time Between Aircraft

In order to increase the robustness of the model, the ground movement process was expanded by implementing buffer times between aircraft. To add the buffer times in the QPPTW algorithm there have been some modifications to the algorithm. The first change is adjusting the necessary time for blocking each edge. The QPPTW algorithm uses time windows for each edge in order to identify when each edge can be available. If there is a large enough gap (time window) for an aircraft to use that edge, then the algorithm expands to that edge. For implementing the buffer time between aircraft, this time window needs to be longer in duration to include the buffer time.

The second change is when adjusting the time windows, after the route has been found. After QPPTW finds the series of nodes that form the route of the aircraft and the times that the aircraft is going to reach each node, the time windows for each edge are adjusted so they are ready to be used by the next aircraft. In order to include a buffer time between aircraft, the length of the time windows is reduced by the time duration that is defined for the buffer time. This extension was added in the temporary time windows as well, since it is necessary to find the delays that can happen when two (or more) aircraft cannot be more than 30 seconds “near” each other.

7.8.2 The Impact of Adding a Buffer Time Between Aircraft

The same experiments (see previous section) were executed again, this time including the buffer times, to evaluate the effects of the buffer time between aircraft. Buffer times between aircraft were not used for the previous section (the buffer time was set to 0), as is the case when aircraft are routed live. For this set up, the buffer time was set to 30 seconds. This buffer time will force all aircraft to have a minimum 30 seconds gap between them during the routing process. It is expected that the number and the duration of delays is going to be significantly higher when the buffer time between aircraft is included, since more aircraft will be affected if they have to keep a 30 seconds gap for shared resources (such as taxiway sections). Moreover, the improvement in the number and the duration of delays - when the ground movement process is included compared to not taking the process into consideration - is expected to be higher when the buffer time is not included. This is expected to happen because the ground movement process is more constrained and there will be fewer available sections of paths for the aircraft to use.

Moreover, as mentioned earlier, the gate allocation process already takes implicitly into consideration potential delays that can happen near the gates by avoiding to allocate departing aircraft to the same gate groups at the same time (avoiding pushback delays) and avoids allocating aircraft that are arriving and departing at the same time to gates that are near to each other. This makes the allocation more robust, as it is more likely for aircraft in these cases to interact. However, the aircraft in these cases are also more likely to interact when the buffer time between aircraft is included. For this reason, including the buffer time will potentially not greatly benefit the gate allocation process as robustness near gates is already considered by the gate allocation process.

The results of the new experiments have been summarised in Table 7.5. The format of the table has remained the same as in Table 7.4 (see previous section).

Table 7.5: Results after solving the gate allocation process while considering the ground movement including buffer times between aircraft

Terminal /Day	Iterations	Initial delay [s] (conflicts)	Final delay [s] (conflicts)	Improvement [%]	Objective
1 / 1	28 (27)	426 (13)	0 (0)	100%	0.53
1 / 2	25 (25)	647 (20)	0 (0)	100%	1.73
1 / 3	44 (32)	257 (11)	0 (0)	100%	1.56
1 / 4	21 (18)	458 (17)	0 (0)	100%	0.56
1 / 5	6 (6)	213 (10)	0 (0)	100%	0.25
1 / 6	33 (33)	598 (15)	0 (0)	100%	0.54
1 / 7	29 (29)	326 (13)	0 (0)	100%	1.16
2 / 1	332 (248)	137 (7)	25 (2)	82%	9.61
2 / 2	442 (430)	381 (12)	81 (4)	79%	2.82
2 / 3	337 (329)	380 (9)	75 (4)	80%	2.32
2 / 4	348 (301)	162 (7)	20 (1)	88%	0.59
2 / 5	394 (246)	208 (8)	18 (3)	91%	3.42
2 / 6	139 (137)	816 (14)	71 (3)	91%	5.60
2 / 7	331 (280)	235 (8)	51 (3)	78%	2.30
3 / 1	226 (206)	299 (18)	0 (0)	100%	1.01
3 / 2	178 (178)	850 (23)	0 (0)	100%	0.36
3 / 3	82 (64)	276 (20)	0 (0)	100%	0.16
3 / 4	227 (218)	677 (32)	0 (0)	100%	0.92
3 / 5	249 (245)	784 (29)	31 (4)	96%	1.34
3 / 6	14 (13)	258 (6)	0 (0)	100%	0.22
3 / 7	134 (99)	163 (14)	0 (0)	100%	0.65
Average	172 (151)	416 (15.2)	17.7 (1.14)	-	1.79
Total	-	8740 (319)	372 (24)	95.7%	-

As the results show, all of the instances that were solved, resulted in an improvement in the total duration of delays that happen when the aircraft are routed. Overall, the total duration of delays would decrease by 95.7% from 2 hours and 26 minutes to 6 minutes and 12 seconds and the total number of delays would decrease by 295, from 319 conflicts to 24. As expected, the number and the duration of delays is higher when the buffer time between aircraft is included, since it is more likely for aircraft to interact with each other when they have to keep a 30 seconds gap between them.

Taking into consideration the results from the previous section, the integrated model seems to still have a drastic improvement in the duration and number of delays when there is a buffer time between aircraft. As mentioned earlier in this section, it is harder to reduce the total duration of delays when there is a 30 seconds gap enforced to aircraft that want to use the same resources (such as parts of taxiways). The availability of parts of the paths is significantly reduced, making it harder for an aircraft to avoid delays overall.

Moreover, the gate allocation process attempts to implicitly take into consideration the ground movement process. Avoiding allocations where potential delays (during the routing process) can happen makes it more likely to avoid delays when the buffer time is also included. However, after examining the results, there are still many instances where the delays have been significantly reduced or eliminated. In 13 out of the 21 instances all the delays have been eliminated and the instance that had the smallest improvement in terms of total delays, still improved by 78%.

Having a robust solution seems to not significantly affect the improvement that can be achieved when an integrated model is used. An 95.7% reduction in the duration of delays is a significant improvement especially when considering that the overall robustness of the solution is also increased. However, it is important to mention that the number of iterations that are needed for the integration process has been increased, which in turn negatively affects the execution time. Usually, a solution was found within a few hours or half a day. However, some of the hardest instances would need over a week to solve, with the longest one taking approximately 14 days to solve.

The average number of iterations in the integrated model was around 165, 142 of which would provide a solution where the total duration of delays would be decreased. As in the previous section, this shows that there is a variety of alternative allocations that can provide a wide range of alternative input for the ground movement process, without drastically affecting the fitness of the objective value of the gate allocation problem. Again, not all of the proposed allocations improve the total delay that happens when the initial gate allocation solution is used for solving the ground movement process. This provides more evidence that the gate allocation model on its own can produce a competitive solution, in term of delays that can later happen during the ground movement process of aircraft.

Finally, the objective function of the gate allocation problem was again not greatly affected when the buffer time between aircraft was included. On average, the value of the final gate allocation solution would be around 1.86% larger than the initial gate allocation solution, which shows that an integrated model can provide an overall good solution, without greatly affecting the solutions of the individual processes.

Figure 7.4 shows on the left side axis the delays that happen when the ground movement problem is solved every time the gate allocation process produces a new allocation, and on the right-side axis, the increase (as a percentage) of the objective value of the gate allocation process for each iteration, for terminal 1, day 2. Comparing this graph, with the graph in Figure 7.3, a number of differences can be observed. In this graph, the total delay when the initial gate allocation is used for solving the ground movement problem, is larger (647 seconds versus 203 when no buffer time between aircraft is included). The additional delay is caused by the fact that more aircraft interact with each other as it has been previously highlighted. This results in a larger number of iterations as well. In Figure 7.4 the number of iterations is 26, whereas in Figure 7.3 it is 16. Furthermore, the increase of the objective function for the gate allocation problem alone is also larger (0.87% versus 1.73%), as it is more difficult to reduce all the conflicts that happen during the ground movement process when the buffer times are included.

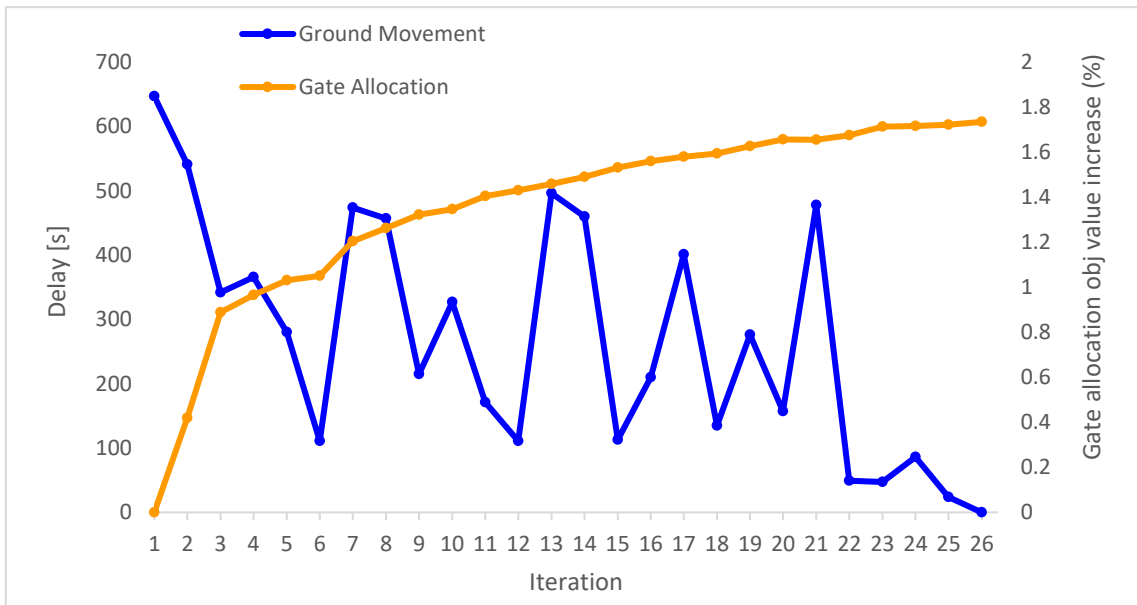


Figure 7.4: The duration of delays during the ground movement process and the increase of the objective value of the gate allocation model for each iteration, for terminal 1, day 2 when buffer time between aircraft is included

7.9 Conclusions

The gate allocation process and the ground movement process affect one another and are part of a wider problem that needs to be optimised in airports. Solving the two processes separately does not take into consideration the full problem. An integrated model is important as it can provide a gate allocation solution that will take into consideration any delay that can happen when the aircraft are later moving on the ground. Minimising these delays can provide large

benefits (such as less fuel consumption, reduced CO₂ emissions and less traffic), without greatly affecting the objectives of the gate allocation process. So far, these two processes are usually solved separately and there has been little research towards integrating them.

In this chapter, an integration framework was presented that solved the gate allocation problem by taking into consideration the ground movement process of aircraft. One of the latest and most advanced gate allocation models, that was developed by Neuman in her PhD thesis, was used for integrating with the ground movement problem. An algorithm was developed that would search for any potential conflict between combinations of aircraft (pairs, triplets, etc.) executing a light version of the ground movement solver for each combination of aircraft. This algorithm would run the ground movement process multiple times, and the problem was solved on a pre-existing graph using temporary time windows to mark the availability of each edge, making it possible to find conflicts between combinations of aircraft fast. Moreover, the routing algorithm was extended in order to include buffer times between aircraft.

Furthermore, an integration framework was developed in conjunction with Dr. Neuman that allows the information flow between the two solvers (gate allocation and ground movement) and adds the feedback from the ground movement process (conflicting aircraft) to the gate allocation solver.

After solving the problem using the integrated framework without using any buffer time between aircraft, the total duration of delays would decrease on average about 99.5%. Moreover, the value of the objective function was not affected significantly in most cases.

Moreover, since there are still many unpredicted delays that can happen that affect the ground movement process of aircraft, the framework was also executed using a 30 seconds buffer time between all aircraft in order to obtain a more robust solution. The experiments show that there was on average a significant improvement in the reduction of the total duration of delays (about 95.3%). Even though there were fewer instances where delays were completely eliminated when the buffer time was included, the robustness of the solution has been significantly increased, so in practice an integrated model that includes buffer times may eventually result in fewer delays.

Overall, the results showed that in both set-ups (with and without buffer time) the integrated model would provide a significant reduction in the total duration of delays that happen during the routing process of aircraft. Integrating the ground movement process to the gate allocation problem provides a more complete approach to the full problem and it is important to be considered as it can significantly affect the overall solution. Moreover, solving the problem

for the full airport (all terminals at the same time) may provide even better improvements in the overall solution.

To conclude, integrating with even more processes such as the runway sequencing process and improving the execution times of individual solvers can provide even better results, making airports much more effective and environmentally friendly.

An A* Approach for the Quickest Path Problem with Time Windows

8.1 Introduction

An airport is a place where many resources are shared and in order to maximise its capacity all the processes that take place need to be well organised. Many processes happen in parallel and many affect or get affected by other processes. Finding the optimal solution when taking all the processes into consideration can be computationally expensive, making it hard to use in a system that could provide real time advice to the air traffic controllers. This was particularly evidenced in Chapter 7, when solving the gate allocation problem by considering the routing process for some days with high traffic, would require a considerable amount of time for the problem to be solved.

Having a fast routing algorithm, will allow researchers to add additional features and complexity to the model and more importantly to integrate with other processes that happen in parallel in airports in a real-time system for aircraft ground movement.

As mentioned in Chapter 3, the QPPTW algorithm is a fast and useful algorithm when it comes to routing aircraft, as it can take into consideration all the available paths. However, QPPTW involves searching for the quickest path by expanding the search in all directions, similarly to Dijkstra's algorithm. Finding a way to direct the expansion can potentially assist the search and achieve a better performance.

This chapter presents an improved version of the QPPTW algorithm for solving the routing and scheduling problem that is based on the A* algorithm. The A* QPPTW (ASQPPTW) algorithm still solves the problem sequentially as the QPPTW algorithm does, but does not expand towards all directions. Instead it has a guided search for the destination node.

The results will show that the ASQPPTW algorithm can solve the problem faster than QPPTW algorithm, providing an excellent tool for solving the routing process fast. ASQPPTW algorithm can allow processes that solve other problems in airports, to have multiple routing solutions with different configurations resulting in a fast and more integrated system that can take the whole ground movement problem into consideration.

Section 8.2 presents an overview of the challenges and the new implementations of the ASQPPTW and describes the mechanics of the algorithm. Section 8.3 provides the description of the setups that were used for the experiments. Section 8.4 provides a head to head comparison between ASQPPTW and QPPTW of the execution time that is necessary for solving the routing problem. Section 8.5 investigates the reasons that ASQPPTW is able to solve the routing problem faster focusing on the way each algorithm expands and Section 8.6 discusses the conclusions that are drawn from this chapter.

8.2 The A* Approach and Overview of Implementations

In this section, the ASQPPTW algorithm is described and explained. First, the relevant previous work is presented and discussed. Then the heuristics that are used for an A* search approach are discussed and compared. Moreover, the heuristic that was chosen for improving the QPPTW algorithm is presented and described. Finally, various concepts about the ASQPPTW algorithm as well as the algorithm itself are defined and a pseudocode of the algorithm is presented.

8.2.1 Previous Work

QPPTW is a popular algorithm for solving the routing process and integrating with other processes. Many models use this algorithm as a base for the routing process and expand its functionality by adding many useful extensions and integrating with other processes.

Gawrilow et al. (2008) develop the algorithm and have used it for controlling automated guided vehicles in container terminals. Ravizza et al. (2014) implement the stand holding process in the QPPTW algorithm in order to take into consideration the runway sequencing of aircraft and to shift the runway delays to the stand where the aircraft is parked. Ravizza et al. (2013b) analyse the trade-off between taxi time and fuel consumption, by using the k-QPPTW algorithm that finds the k fastest paths. Weiszer et al. (2015a) as well as Chen et al. (2016b) both use a database for speed profiles of aircraft to reduce the calculation time for the routing and scheduling problem that is solved by the QPPTW algorithm. Benlic et al.

(2016) use the QPPTW algorithm for solving the runway sequencing problem considering the routing and scheduling problem. Furthermore, in this research the QPPTW algorithm is the main algorithm used for the routing process (see Chapters 4 to 7), integrating various functionalities such as the pushback process.

However, even though the QPPTW algorithm is a fast and reliable routing algorithm, it is a Dijkstra based algorithm. Dijkstra's algorithm is used for finding the shortest path between a start node to a goal node in a graph. It explores nodes towards all the directions until it reaches the target node. There are other graph search algorithms (such as A* algorithm) that can find the shortest path faster, by making a more informed search.

There has been an attempt for solving the routing and scheduling problem with an A* approach by Lesire (2010), but the QPPTW algorithm provides a better coverage of the solution space, potentially allowing it to find better solutions, since it can guarantee that the quickest path is found. The Lesire approach uses a heuristic based on the Euclidian distance, which can be misleading when the shortest path is blocked, since this heuristic usually favours a shortest path instead of the quickest. Finally, the execution time is mentioned in the paper to be "less than a second per flight" which is fast, but not fast enough for solving a full day multiple times (which is something that can be useful when integrating with other processes). The QPPTW version of Ravizza in his PhD thesis (2013c) that is based on Dijkstra's algorithm solved the problem in similar times.

8.2.2 A Heuristic Estimation of the Cost

A* algorithm is an algorithm that is widely used for pathfinding that is based on Dijkstra's algorithm. It makes use of heuristics in order to have a guided search, resulting in faster execution times. Instead of expanding to the node that will minimise the shortest path distance A* algorithm will select to expand to the node that minimises the shortest path distance up to that point plus the remaining cost to the goal node. More specifically, minimising the function

$$f(n) = g(n) + h(n) \quad (8.1)$$

where n is the node that is being considered, $g(n)$ is the cost of the path from the starting node to node n and $h(n)$ is the remaining cost that the heuristic has estimated.

In this case, the optimal solution is required, since any additional travel time of aircraft can result in an increase in fuel consumption and CO₂ emissions. For this reason, the ideal heuristic should be exactly equal to the cost that is required to move from node n to the goal

node. However, this is very hard as it is not always possible to know the exact remaining cost to the goal node.

In the case of the QPPTW algorithm it is the quickest path that needs to be found and not the shortest. The use of time windows by QPPTW increases the available ways that an aircraft can reach its destination, as it introduces the dimension of time. The QPPTW algorithm routes aircraft in a dynamic graph, as the node where it expands is affected not only by the original weight of the edges but also from the delays that can happen from previously routed aircraft that make use of these edges. For this reason, it is not possible to know which is the quickest path or what the cost of it will be, before actually routing the aircraft in a graph that takes into consideration all of the previously routed aircraft.

The best heuristic that can be achieved in this case is the cost of the shortest path, which is the same for all aircraft, and it is the exact cost for the remaining path when delays do not occur. In cases where there are no blocked edges due to previously routed aircraft, the heuristic will only expand to the shortest path, providing a very fast solution. In cases where there are delays that are caused by blocked edges, the cost of the shortest path will provide the largest value that can be used by a heuristic that is not larger than the actual cost that is required to reach its destination - thus guaranteeing that the optimal solution is found.

Calculating the cost of the shortest path every time the algorithm needs to expand can provide a very accurate heuristic, but it is more computationally expensive compared to calculating a less accurate cost such as the remaining Euclidean distance. However, this issue can be solved by using a pre-calculated table that can provide the remaining duration from each node to each stand or runway.

The algorithm that was developed for this chapter (ASQPPTW), uses a heuristic that calculates the remaining cost of the shortest path. ASQPPTW follows Equation 8.1 as well. In this case $g(n)$ is the duration that it takes for an aircraft to travel from the starting node to node n , including any delays that happen across this path. $g(n)$ is already known as it is the time that the aircraft will enter the next node (including any delays that can occur on the last edge due to previously routed aircraft).

The $h(n)$ is calculated by finding the duration that it takes for an aircraft to reach the destination node from node n when the shortest path is traversed. This duration is calculated beforehand and is used as input for the algorithm. In order to have this data before the algorithm is executed, aircraft were routed on an empty graph from each node to each runway for routing the departing aircraft and from each node to each stand for the arriving aircraft. A tailored implementation of Dijkstra's algorithm was used for this process. In each node of the

graph the time that it takes to reach each stand and runway is stored. This makes it possible to retrieve $h(n)$ fast whenever it is needed.

In order to quantify the benefits of a heuristic that uses the cost of the shortest path, another heuristic was also created and tested (see Section 8.5.6). This new heuristic is based on the Euclidian distance from node n to the goal node. The duration that is needed to traverse this distance was then found (using the speed that aircraft is moving) in order to be able to add any delays that can happen from previously routed aircraft. In this case, the calculation of the heuristic is almost always smaller than the actual duration that is needed by an aircraft to traverse the shortest path. The only case when the cost of the heuristic is the same as the cost from node n to the goal node, is when node n and the goal node are connected by a single edge.

8.2.3 Definitions of the Variables

Various variables and constants will be used in the explanations of the problem and the definition of concepts, as well as in the algorithm explanation. For convenience, these have been collected into a single table (Table 8.1) here for reference. In order to maintain some thread of continuity with the QPPTW algorithm which ASQPPTW is based on and to make it easier for the reader to better understand the additions of ASQPPTW, the notation of common variables that was used below are the same or similar to the notation that was used in the PhD thesis of Stenzel (2008) and Ravizza (2013c).

Table 8.1: Table of definitions

E	The set of all edges
V	The set of all vertices
$e \in E$	An edge
$v \in V$	A vertex
$G = (V, E)$	The directed graph representing the airport layout, with vertices $v \in V$ and edges $e \in E$
a_e^j	The start time of the j^{th} time-window on edge $e \in E$
b_e^j	The end time of the j^{th} time-window on edge $e \in E$
$F_e^j = [a_e^j, b_e^j]$	j^{th} time-window on edge $e \in E$, from time a_e^j to time b_e^j
$\mathcal{F}(e)$	The sorted set of all of the time-windows on edge $e \in E$
H	The priority queue storing the added labels
a_L	The start time of Label L
b_L	The end time of Label L

$I_L = [a_L, b_L]$	The time interval used in a label L
$pred_L$	The predecessor label of label L
$L = (v_L, I_L, pred_L)$	A label on vertex $v_L \in V$ with a time interval I_L and predecessor label $pred_L$
$\mathcal{L}(v)$	The set of all of the labels at vertex $v \in V$
R	A conflict-free route that is being generated
$s \in V$	A source vertex
$t \in V$	A target vertex
$time$	The time that an aircraft sets off
p	The pushback duration
$T = (s, t, time, p)$	A taxi request to route, from source $s \in V$ setting off at time $time$, to target $t \in V$ and with pushback duration p (for departures)
w_e	The weight (necessary taxi time) of edge $e \in E$
$C(v, n)$	The estimated remaining duration that it would take to reach the destination vertex $n \in V$ from the current vertex $v \in V$
$H.getMin()$	Function that returns the element with the lowest value in heap H
$Maximise(a, b)$	Function that returns the element with the largest value between elements a and b
$head(e)$	Function that returns the vertex y of an edge e that is directed from vertex x to vertex y .

8.2.4 The ASQPPTW Algorithm

The ASQPPTW is an improved version of the QPPTW algorithm. The approach that is described below is based on the PhD thesis of Ravizza (2013c). Both the “forwards” version (summarised in Chapter 3) and the “backwards” version (presented in Chapter 5) have been modified in this approach. The pushback process was explicitly implemented for the departing aircraft and both versions have been enhanced by considering the remaining time when searching for the next vertex to expand.

The ASQPPTW algorithm uses as an input the graph $G = (V, E)$ as described in Chapter 3, where each edge has its own weight function w_e . The availability of each edge is denoted by a set of time windows $\mathcal{F}(e)$ for that edge. A taxi request $T_i = (s_i, t_i, time_i, p_i)$ that contains the details of the flight i are also used as an input to the algorithm in order to route a flight and the algorithm will return the quickest available path R that can be used by aircraft i in order to reach vertex t_i , starting from s_i for arrivals (or to reach vertex s_i , starting from t_i for

departures since they are routed backwards) by respecting the aircraft that have been previously routed (time-windows).

Since departures are routed backwards, starting from the destination of the aircraft and find each previous step until the algorithm reaches the stand, the algorithm starts from the target vertex and searches for the source vertex which is opposite to what the arrival process does.

Algorithm 8.1 shows the pseudocode of the ASQPPTW algorithm for arriving aircraft and Algorithm 8.2 for departing aircraft. It is based on the QPPTW algorithm as was described in Chapter 3. Again, the parts that are different from the original QPPTW algorithm have been underlined in red.

The difference with the two algorithms is that Algorithm 8.2 includes the stand holding process for departing aircraft with the pushback process explicitly implemented and the heuristic that is used for the pushback process differs from the heuristic used in the rest of the routing process. Moreover, since the departing aircraft use the stand holding process and are routed backwards there are some differences in the way that the heuristic was implemented in the arriving and the departing part of the algorithm.

The new algorithm iteratively finds the quickest routes from source to target vertices (or from target to source) in a weighted graph by taking into consideration the time-window constraints. The expansion of the algorithm is similar to A* algorithm with the addition of taking into consideration the availability of the edges of the graph.

The priority queue H (see Table 8.1) where the labels that are generated by the algorithm are stored, is initialised in line 1 (see Algorithm 8.1 and Algorithm 8.2). The labels that are generated are also stored in a list \mathcal{L} (which is initialised in line 2). This list is used for finding the label (and the vertex of that label) that a label has expanded from. It is also used for accessing the previously created labels for the dominance check that takes place in lines 23-29 for Algorithm 8.1 (26-32 for Algorithm 8.2) and for constructing the final route. The route is formed by finding the previous label of each label, starting from the latest label that has reached the target (or source) vertex all the way back to the source (or target) vertex (line 9). An initial label that expands from the source vertex is created in line 3 and is inserted to the priority queue (line 4) and to the label list (line 5). Every time a label is inserted in the priority queue, the key which the label was inserted with, is used for the sorting process of the queue. The value of this key is calculated by adding the earliest arrival time to the vertex of that label plus the estimate of the remaining time to the target vertex for arrivals. For departures, the key for the label is the latest exit time from that vertex minus the estimate of the remaining time to the source vertex.

In each iteration of the while loop (line 6), the algorithm will check whether there are still labels consideration in the priority queue. If the queue has no more labels, the algorithm cannot expand to a new vertex, which means that there is no available path that leads to the target vertex (line 35 for arrivals or 39 for departures). If there are still labels in the priority queue, the label with the smallest key (line 7) is pulled. For departing aircraft, the algorithm needs to find the latest time that the aircraft can leave each vertex (which would be the maximum key) so the keys are stored in the priority queues with the opposite value in order to maintain the ascending order of the queue and be able to pull the label with the “smallest” key.

If the vertex of the label that has been pulled is the target vertex (line 8) for arrivals or the source vertex for departures, a function for forming the route is called (lines 9-10), otherwise the algorithm will expand to the next available vertex. Every time that the algorithm considers a new vertex, the time-windows of the edge that lead to the vertex are examined (lines 11-12). The algorithm will expand to a new vertex only if the time interval of the time-window of the edge that leads to the vertex, is available when the aircraft will start traversing this edge (lines 14 and 16). At this stage, the two algorithms start to differ. The two branches are similar, however the “arrivals” one works forward in time to find the next vertex whereas the “departures” one works backwards in time to find the previous vertex of the route.

For each edge that is considered, the earliest time that it can be exited is found for arrivals and the latest time it can be entered is found for departures, including the pushback time if the vertex that is considered (for departures) matches the source vertex (lines 18-19). If the time-window of the edge that is leading to the next vertex is wide enough for the aircraft to traverse this edge (line 20 for arrivals, 24 for departures) a new label is created (lines 22 or arrivals, 25 for departures). The new label is then checked to determine if it is dominated by labels that have been added to the list of labels earlier, and if so, it is not considered (lines 25-26 for arrivals, 28-29 for departures) or if the new label dominates a label that has been previously added to the list of labels, then the dominated label is removed from the priority queue (lines 27-29 for arrivals, 30-32 for departures). Finally, the new label is added in the priority queue with the appropriate key.

8.2.5 The Differences Between the Departing and Arriving Process

As indicated earlier, the departing and arriving processes have a few differences. The use of the heuristic for calculating the remaining duration (such as during the pushback process) can in some cases provide a more tailored cost. These differences can affect the execution time

of each process and the way the algorithm expands. For this reason, they have been highlighted below, and investigated in Sections 8.4 and 8.5.

The departing process in the ASQPPTW algorithm has an if statement in line 34. This “if” statement checks if the vertex that is considered, is the source vertex. In the case that the vertex is not the “source” vertex (the vertex where the aircraft starts the pushback process) then the cost is equal to the time that the aircraft starts leaving the current vertex ($b_{L'}$) minus the calculated remaining time to the source vertex. As mentioned earlier the sign of the cost is inverted because since the latest departure time is considered, the maximum entry time to the vertices are pulled from the priority queue which is the opposite of the minimum of the negative cost. However, when the vertex that is considered is the source vertex, this means that there is no remaining time to the destination. Moreover, if the current vertex is the source vertex, the time that the aircraft starts leaving the current vertex ($b_{L'}$) is also different. $b_{L'}$ is equal to $time_{in}$ since the end of the time interval of the current label L' (see lines 21 and 25). $time_{in}$ is smaller by the pushback duration of the aircraft (p_i) when the considered vertex is the source vertex so $b_{L'}$ is also smaller by the pushback duration of the aircraft (p_i). For this reason, the pushback duration is then added to the time that the aircraft starts leaving the current vertex ($b_{L'}$). This results in the latest label having a great advantage when it is considered by the priority queue as there is no other label that was considered earlier with higher $b_{L'}$ but with an added heuristic cost of $h(n) \geq 0$.

Algorithm 8.1: A* Quickest Path Problem with Time Windows (ASQPPTW) - Arrivals

Input: Graph $G = (V, E)$ with weights w_e for all $e \in E$, the set of sorted time-windows $\mathcal{F}(e)$ for all $e \in E$, a taxi request $T_i = (s_i, t_i, time_i, p_i)$ for aircraft i , with the source vertex $s_i \in V$, the target vertex $t_i \in V$ and the start time $time_i$.

Output: Conflict-free route R from s_i to t_i with minimal taxi time that starts at the earliest at time $time_i$, respects the given time-windows $\mathcal{F}(e)$ or returns the message that no such route exists.

```
1  Let  $H = \emptyset$ 
2  Let  $\mathcal{L}(v) = \emptyset \quad \forall v \in V$ 
3  Create new label  $L$  such that  $L = (s_i, [time_i, \infty), nil)$ 
4  Insert  $L$  into heap  $H$  with key  $time_i + C(s_i, t_i)$ 
5  Insert  $L$  into set  $\mathcal{L}(s_i)$ 
6  while  $H \neq \emptyset$  do
7      Let  $L = H.getMin()$ , where  $L = (v_L, I_L, pred_L)$  and  $I_L = [a_L, b_L]$ 
8      if  $v_L = t_i$  then
9          Reconstruct the route  $R$  from  $s_i$  to  $t_i$  by working backwards from  $L$ 
10         return the route  $R$ 
11     forall the outgoing edges  $e_L$  of  $v_L$  do
12         foreach  $F_{e_L}^j \in \mathcal{F}(e_L)$ , where  $F_{e_L}^j = [a_{e_L}^j, b_{e_L}^j]$ , in increasing order of  $a_{e_L}^j$  do
13             /*Expand labels for edges where time intervals overlap*/
14             if  $a_{e_L}^j > b_L$  then
15                 goto 11 /*consider the next outgoing edge*/
16             if  $b_{e_L}^j < a_L$  then
17                 goto 12 /*consider the next time-window*/
18             Let  $time_{in} = \text{Maximise}(a_L, a_{e_L}^j) \quad /* a_{e_L}^j > a_L \Rightarrow \text{waiting} */$ 
19             Let  $time_{out} = time_{in} + w_{e_L}$ 
20             if  $time_{out} \leq b_{e_L}^j$  then
21                 Let  $u = \text{head}(e_L)$ 
22                 Let  $L' = (u, [time_{out}, b_{e_L}^j], L)$ 
23                 /*dominance check*/
24                 foreach  $\hat{L} \in \mathcal{L}(v)$  do
25                     if  $\hat{L}$  dominates  $L'$  then
26                         goto 12 /*next time-window*/
27                     if  $L'$  dominates  $\hat{L}$  then
28                         Remove  $\hat{L}$  from  $H$ 
29                         Remove  $\hat{L}$  from  $\mathcal{L}(v)$ 
30                 if  $v_L = t_i$  then
31                     Insert  $L'$  into heap  $H$  with key  $a_{L'}$ 
32                 else
33                     Insert  $L'$  into heap  $H$  with key  $a_{L'} + C(v, t_i)$ 
34                     Insert  $L'$  into set  $\mathcal{L}(v)$ 
35 return "there is no  $s_i - t_i$  route"
```

Algorithm 8.2: A* Quickest Path Problem with Time Windows (ASQPPTW) - Departures

Input: Same as Algorithm 8.1**Output:** Same as Algorithm 8.1

```
1  Let  $H = \emptyset$ 
2  Let  $\mathcal{L}(v) = \emptyset \quad \forall v \in V$ 
3  Create new label  $L$  such that  $L = (t_i, [0, time_i], nil)$ 
4  Insert  $L$  into heap  $H$  with key  $-(time_i - C(s_i, t_i))$ 
5  Insert  $L$  into set  $\mathcal{L}(t_i)$ 
6  while  $H \neq \emptyset$  do
7      Let  $L = H.getMin()$ , where  $L = (v_L, I_L, pred_L)$  and  $I_L = [a_L, b_L]$ 
8      if  $v_L = s_i$  then
9          Reconstruct the route  $R$  from  $s_i$  to  $t_i$  by working backwards from  $L$ 
10         return the route  $R$ 
11     forall the outgoing edges  $e_L$  of  $v_L$  do
12         foreach  $F_{e_L}^j \in \mathcal{F}(e_L)$ , where  $F_{e_L}^j = [a_{e_L}^j, b_{e_L}^j]$ , in increasing order of  $a_{e_L}^j$  do
13             /*Expand labels for edges where time intervals overlap*/
14             if  $a_{e_L}^j > b_L$  then
15                 goto 11 /*consider the next outgoing edge*/
16             if  $b_{e_L}^j < a_L$  then
17                 goto 12 /*consider the next time-window*/
18             Let  $time_{out} = \text{Minimise}(b_L, b_{e_L}^j)$  /*  $b_{e_L}^j < b_L \Rightarrow \text{waiting}$  */
19             Let  $u = \text{head}(e_L)$ 
20             if  $u = s_i$  then
21                 Let  $time_{in} = time_{out} - w_{e_L} - p_i$ 
22             else
23                 Let  $time_{in} = time_{out} - w_{e_L}$ 
24             if  $time_{in} \geq a_{e_L}^j$  then
25                 Let  $L' = (u, [a_{e_L}^j, time_{in}], L)$ 
26                 /*dominance check*/
27                 foreach  $\hat{L} \in \mathcal{L}(v)$  do
28                     if  $\hat{L}$  dominates  $L'$  then
29                         goto 12 /*next time-window*/
30                     if  $L'$  dominates  $\hat{L}$  then
31                         Remove  $\hat{L}$  from  $H$ 
32                         Remove  $\hat{L}$  from  $\mathcal{L}(v)$ 
33                 if  $v_L = s_i$  then
34                     Insert  $L'$  into heap  $H$  with key  $-(b_{L'} + p_i)$ 
35                 else
36                     Insert  $L'$  into heap  $H$  with key  $-(b_{L'} - C(v, t_i))$ 
37                 Insert  $L'$  into set  $\mathcal{L}(v)$ 
38 return "there is no  $s_i - t_i$  route"
```

8.3 Experimental Settings

In order to investigate the performance of the ASQPPTW algorithm and how the A* approach of routing aircraft with time windows performed compared to the Dijkstra's approach, a set of experiments were executed. Firstly, the execution times of the ASQPPTW and the QPPTW algorithms were examined and compared. The experiments were also run with routing only the arriving and only the departing aircraft in order to investigate if and how the stand holding process (discussed in Chapter 5) affects the performance of the new algorithm. Secondly the number of labels that are generated with each algorithm was measured by counting the labels that were inserted in the priority queue. This gives some measure of how much of the search space the addition of the heuristic to the algorithm is allowing it to ignore.

Both ASQPPTW and QPPTW were implemented in the same Java code using the same functions and algorithms for all of the processes that they share (such as dominance check, reconstruction the route, adjusting the time windows etc.). The same data was used for both of the experiments. The experiments were executed for 7 different instances (data from 7 different days) from Zurich airport and for 7 different instances from Arlanda airport in order to have a wider variety of results.

Finally, the experiments that were recording the execution times were run 5 times for each day and for each setup and the results that are mentioned in the following sections refer to the average execution times. This process was performed in order to rule out any random fluctuations in the performance of the computer. Eventually all of the execution times were homogenous and the differences in execution times was most of the times very close to the average.

8.4 Execution Times for Each Algorithm

This section presents the results after comparing the execution times of the QPPTW algorithm with those of the ASQPPTW algorithm. First the two algorithms are compared when solving the whole problem and later when solving the arriving and departing aircraft individually. Moreover, the two algorithms are compared with each other in another airport in order to verify that the improved performance of the ASQPPTW algorithm is not limited to a specific airport.

Finally, the heuristic is slightly changed in order to understand how it affects the execution time of the ASQPPTW algorithm. Indeed, the ASQPPTW algorithm outperforms QPPTW in every set-up that was tried. An average improvement of 46% in execution times was observed

for Zurich airport and 67% for Arlanda airport. Moreover, the ASQPPTW algorithm performed the best when routing the departing aircraft and this was at least partially attributed to the heuristic that is implemented for the pushback process, which the arriving aircraft do not have. In conclusion, the ASQPPTW algorithm routes each aircraft faster than QPPTW in every configuration that was tried and this is attributed to the fact that the heuristic that is used significantly reduces the search tree. In the next section (Section 8.5) the way that the ASQPPTW algorithm expands is going to be examined.

8.4.1 Solving the Full Problem

Initially the execution times for solving the whole problem were investigated. This includes other algorithms on top of the algorithm presented in section 8.2 that are necessary for the problem to be solved, such as the reconstruction of the route after the ASQPPTW algorithm has found a solution, the readjustment of the time-windows and the sorting of the reservations.

When the algorithms solved the full problem in Zurich airport (for both arriving and departing aircraft), the ASQPPTW algorithm performed significantly better than the QPPTW algorithm. The relevant results are summarised below in Table 8.2. For each individual day, ASQPPTW would solve the problem faster than QPPTW, having an improvement in execution time that ranged from 37% to 58%. In total, the ASQPPTW algorithm would route the full week (7 days) in 5.6 seconds – 46 % faster, compared to 14.6 seconds that were needed for the QPPTW algorithm to solve the same problem.

Table 8.2: Execution times in milliseconds for routing aircraft (Zurich Airport)

	No. of aircraft	QPPTW [ms]	ASQPPTW [ms]	Improvement
Day 1	818	2457	1545	37%
Day 2	806	2310	1422	38%
Day 3	781	1837	919	50%
Day 4	837	2403	1394	42%
Day 5	825	2002	851	57%
Day 6	755	1675	710	58%
Day 7	787	1944	997	49%
Total	5609	14627	7838	46%

Furthermore, the same experiments were executed in Arlanda airport in order to validate the above results and to confirm that the ASQPPTW algorithm is indeed a faster routing algorithm regardless of the type of the airport.

As mentioned in Chapter 6, Arlanda airport is similar in size and capacity to Zurich airport but the morphology of the taxiways, and the layout of the runways and terminals is significantly different. Zurich airport is more compact with “box style” terminals, located on both sides of one of the runways. Moreover, aircraft push back directly on busy taxiways and there are many parallel and interconnected taxiways across the airport. On the contrary, Arlanda airport has long taxiways that are less interconnected with each other and the terminals have aprons where aircraft can push back without affecting the main traffic.

Table 8.3 shows the execution times of the ASQPPTW and the QPPTW algorithms for Arlanda airport. Again, the ASQPPTW algorithm performs significantly better solving the problem from 60% to 73% faster than QPPTW. In total, the ASQPPTW algorithm would route the aircraft of a full week in 8.4 seconds compared to 25.2 (67% better) that was needed for the QPPTW algorithm.

Table 8.3: Execution times in milliseconds for routing aircraft (Arlanda Airport)

	No. of aircraft	QPPTW [ms]	ASQPPTW [ms]	Improvement
Day 1	818	3757	1514	60%
Day 2	806	3772	1259	67%
Day 3	781	3645	1124	69%
Day 4	837	3948	1451	63%
Day 5	825	3731	1001	73%
Day 6	755	3047	959	69%
Day 7	787	3284	1063	68%
Total	5609	25183	8371	67%

Even though the execution times for Arlanda airport are similar to those for Zurich airport when using the ASQPPTW algorithm; the improvement (as a percentage) between the execution times of the ASQPPTW algorithm and the QPPTW algorithm, gives the impression that ASQPPTW performs much better in Arlanda airport. Although this is true, it is important to mention that Arlanda airport is more outspread resulting in the graph that was used for Arlanda to have more nodes than the one of Zurich. This makes it more time consuming for QPPTW to expand towards all the directions until it reaches the target vertex. It is also portrayed in the results, as solving the full week with the QPPTW algorithm requires 15 seconds for Zurich airport compared to 25 seconds that is needed for Arlanda airport. On the other hand, ASQPPTW solves the problem in quite similar time for both of the airports, requiring just 0.5 seconds more time to route a week’s aircraft in Arlanda airport, compared to doing the same for Zurich.

8.4.2 Solving Arrivals and Departures Separately

The algorithms were also tested for solving parts of the problem. In the first run, only the departing aircraft were routed and in the second run, only the arrivals. In both cases the ASQPPTW would again perform better than the QPPTW algorithm. The results are summarised in Table 8.4 and Table 8.5.

For the departing aircraft, the ASQPPTW algorithm performed significantly better, routing all of the departures in 0.4 seconds, (90% faster) compared to the 4.3 seconds that the QPPTW algorithm did. The execution times for each day when the departing aircraft were routed with ASQPPTW were from 87% to 92% faster.

For arriving aircraft, the ASQPPTW algorithm would solve the problem 18% to 41% faster (depending on the day). In total ASQPPTW routed the full week in 3.7 seconds, 28% (1.5 seconds) faster than the QPPTW algorithm did.

Table 8.4: Execution times in milliseconds for routing departing aircraft (Zurich Airport)

	No. of aircraft	QPPTW [ms]	ASQPPTW [ms]	Improvement
Day 1	407	616	78	87%
Day 2	405	641	53	92%
Day 3	392	591	56	91%
Day 4	414	656	75	89%
Day 5	421	628	53	92%
Day 6	377	587	60	90%
Day 7	387	610	47	92%
Total	2803	4329	422	90%

Table 8.5: Execution times in milliseconds for routing arriving aircraft (Zurich Airport)

	No. of aircraft	QPPTW [ms]	ASQPPTW [ms]	Improvement
Day 1	411	1035	790	24%
Day 2	401	981	794	19%
Day 3	389	535	328	39%
Day 4	423	951	778	18%
Day 5	404	494	294	41%
Day 6	378	494	307	38%
Day 7	400	688	416	40%
Total	2806	5177	3706	28%

The same experiments were also executed in Arlanda airport. As Table 8.6 and Table 8.7 show, the results in Arlanda airport follow the same pattern. The ASQPPTW algorithm routed the departing aircraft from 86% to 92% faster than the QPPTW algorithm. Overall, the ASQPPTW algorithm routed the departing aircraft 6.4 seconds faster (88% improvement).

Moreover, ASQPPTW algorithm routed the arriving aircraft of a full week, 3.2 seconds faster which indicates a 44% improvement compared to QPPTW algorithm. The improvement in execution times varied from 35% to 57% for each individual day.

Table 8.6: Execution times in milliseconds for routing departing aircraft (Arlanda Airport)

	No. of aircraft	QPPTW [ms]	ASQPPTW [ms]	Improvement
Day 1	407	1048	144	86%
Day 2	405	1023	142	86%
Day 3	392	1008	81	92%
Day 4	414	1036	148	86%
Day 5	421	1131	115	90%
Day 6	377	968	114	88%
Day 7	387	1042	112	89%
Total	2803	7254	855	88%

Table 8.7: Execution times in milliseconds for routing arriving aircraft (Arlanda Airport)

	No. of aircraft	QPPTW [ms]	ASQPPTW [ms]	Improvement
Day 1	411	1318	758	42%
Day 2	401	1181	717	39%
Day 3	389	897	471	48%
Day 4	423	1255	812	35%
Day 5	404	807	348	57%
Day 6	378	852	406	52%
Day 7	400	966	528	45%
Total	2806	7275	4039	44%

After comparing the results, it is apparent that the ASQPPTW algorithm performs much better for departing aircraft compared to arriving aircraft. There was a 90% improvement for departing aircraft compared to 28% for arriving aircraft in Zurich airport and an 88% improvement for departing aircraft compared to 44% for arriving aircraft in Arlanda airport. In order to understand why this happens, the structure of the two processes (departures versus arrivals) were examined. The main difference in the two processes is that the departing process is using the stand holding process and includes the pushback process. In order to test

whether this feature of the ASQPPTW algorithm is responsible for the difference more experiments were executed.

8.4.3 Solving the Departures without the Heuristic for the Pushback Process

As mentioned earlier the benefits of the heuristic of the ASQPPTW algorithm are much greater when solving the departing process than when solving the arrivals. Even though these two processes are solved differently (the departures are solved backwards in order to achieve stand holding and the pushback process is included) it is useful to test what characteristics result in this improvement.

As mentioned in Section 8.2.5 the heuristic that is used during the pushback process is different than the heuristic that is used for the arrivals. In order to test whether the heuristic for the pushback process affects the performance of the ASQPPTW algorithm the algorithm was modified. The pushback heuristic has been removed and the heuristic for calculating the remaining duration to travel to the target vertex was used for all of the cases (pushback or simple movement). This was achieved by removing the “if” statement (lines 34-36 in Algorithm 8.2) that checks if the aircraft is at a stand and therefore should be performing the pushback process. In this new version, the cost of the label is always equal to the time that the aircraft starts leaving the current vertex (b_L) minus the remaining time to the source vertex $h(n)$.

Table 8.8: Execution times in milliseconds for routing departing aircraft when the heuristic for the pushback process is not included (Zurich airport)

	No. of aircraft	QPPTW [ms]	ASQPPTW [ms]	Modified ASQPPTW [ms]	Improvement
Day 1	407	616	78	348	44%
Day 2	405	641	53	325	49%
Day 3	392	591	56	351	41%
Day 4	414	656	75	324	51%
Day 5	421	628	53	324	48%
Day 6	377	587	60	295	50%
Day 7	387	610	47	320	47%
Total	2803	4329	422	2287	47%

The experiments were executed again, comparing the modified ASQPPTW with the original algorithm. The results are summarised in Table 8.8 and Table 8.9. The modified version performs significantly worse than the original ASQPPTW algorithm. As Table 8.8 and Table 8.9 show, removing the parameter of the cost calculation for the pushback process results in an improvement of 47% in execution time compared to the QPPTW algorithm instead of 90% improvement that happens when the parameter is included in Zurich airport. Similarly, in Arlanda airport the improvement without the parameter of the cost calculation for the pushback process is 52% compared to 88% that was the case when the parameter was included.

Table 8.9: Execution times in milliseconds for routing departing aircraft when the heuristic for the pushback process is not included (Arlanda airport)

	No. of aircraft	QPPTW [ms]	ASQPPTW [ms]	Modified ASQPPTW [ms]	Improvement
Day 1	407	1048	144	520	50%
Day 2	405	1023	142	515	50%
Day 3	392	1008	81	461	54%
Day 4	414	1036	148	493	52%
Day 5	421	1131	115	520	54%
Day 6	377	968	114	457	53%
Day 7	387	1042	112	488	53%
Total	2803	7254	855	3453	52%

This shows that the pushback parameter that is implemented in the ASQPPTW algorithm is at least partially responsible for the better performing execution times of the departing aircraft.

8.5 Investigating the Expansion of the two Algorithms

In this section, the way that each algorithm expands is investigated. More specifically, the research focuses on the number of labels that are generated and how they are distributed for different setups. First the number of labels that are generated for solving the full problem with each algorithm is found and then for solving the arrivals and departures separately. Initially Zurich and Arlanda airports are investigated. Then the priority between arrivals and departures is changed in order to examine how the number of labels that are generated is affected. Moreover, the way that the “number of labels that are generated” is affected by the distance and time that it takes for an aircraft to reach its destination is examined. The

experiments are also executed with less arriving aircraft in order to investigate how traffic affects the number of labels that are generated. Finally, another heuristic that is based on the remaining distance is used in order to compare the efficiency with the heuristic method that was chosen for the ASQPPTW algorithm.

The results show that ASQPPTW generates far less labels for almost every aircraft that it routes compared to the QPPTW algorithm, which explains the improved execution times of ASQPPTW. These results are consistent with every set-up that was described earlier. Moreover, it is argued and demonstrated that the heuristic that was chosen for ASQPPTW is the most efficient way to solve the quickest path problem when time windows are used.

8.5.1 The Number of Labels that are Generated with Each Algorithm

The ASQPPTW algorithm performs significantly better than the QPPTW algorithm. This is attributed to the fact that ASQPPTW has a guided search when looking for the target vertex on the graph, whereas the QPPTW expands in every direction. Both algorithms use labels as a way to consider the next vertex and to store the information about the previous labels that eventually will form the quickest path. It is expected that the ASQPPTW algorithm will generate significantly less labels (for the above reason) than the QPPTW.

Table 8.10: Number of labels generated with each algorithm (ASQPPTW vs QPPTW)

	QPPTW		ASQPPTW	
	Total labels	Average labels/flight	Total labels	Average labels/flight
Day 1	363322	444	77806	95
Day 2	352302	437	74294	92
Day 3	334396	428	63640	81
Day 4	369063	441	81662	98
Day 5	386553	469	69472	84
Day 6	328415	435	65500	87
Day 7	340136	432	62979	80
Total	2474187	441	495353	88

After executing both algorithms, the number of labels that are generated are summarised in Table 8.10. After comparing the labels that are generated between the QPPTW and ASQPPTW algorithms it is apparent that the ASQPPTW generates far less labels. Out of the 5,609 aircraft that were routed (7 days) only 3 of them were routed by generating more labels

when routed with the ASQPPTW algorithm. For solving the full week (5,609 flights movements) the QPPTW algorithm would generate 2,474,187 labels, compared to 495,353 labels that are generated when ASQPPTW solved the same problem. This is an 80% reduction in the total labels that are generated. On average, the QPPTW algorithm needs to generate 441 labels to route one aircraft, compared to 88 labels that it takes for the ASQPPTW algorithm to do the same process. Figure 8.1 shows the distribution of labels that are generated with each algorithm (left) and a cumulative graph that shows the number of aircraft that generate “x” labels or more (right). The distribution graph shows that the number of labels that are generated by ASQPPTW have a smaller mean value compared to the QPPTW algorithm, but also a smaller standard deviation (96.5 versus 175.5).

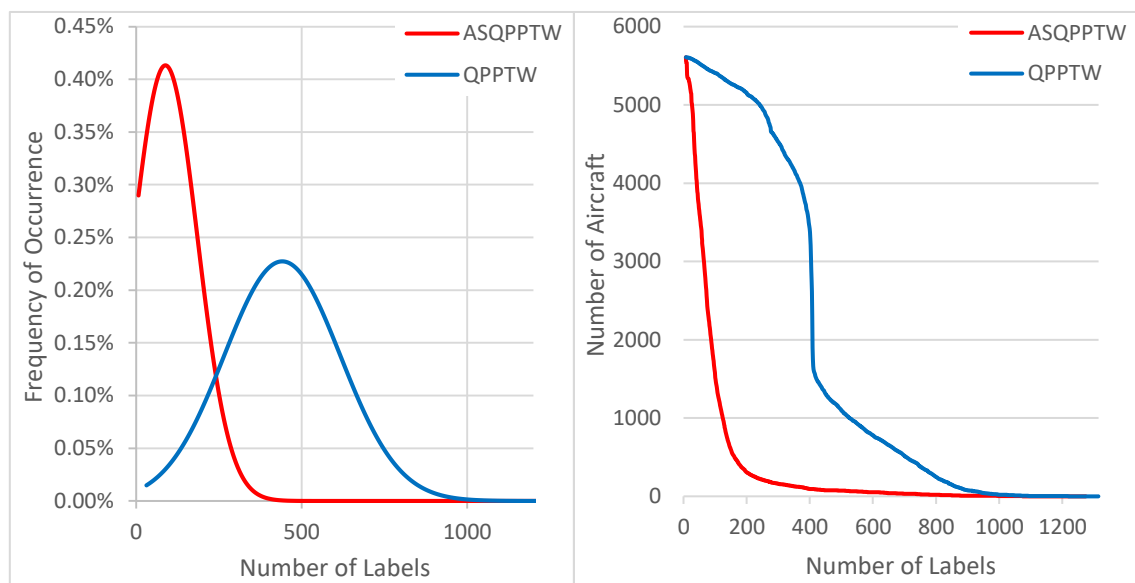


Figure 8.1: Distribution of labels generated by each algorithm (left) and cumulative graph that shows the number of aircraft that generate “x” labels or more (right) for Zurich airport

Furthermore, in order to have a better understanding of the labels that are generated with each algorithm, the same graphs were made separating the arrivals from departures. Figure 8.2 shows the distribution of labels that are generated in more detail, for the two algorithms (to the left) as well as the cumulative graph that shows the number of aircraft that generate “x” labels or more (right). Note that the ASQPPTW algorithm is marked as “A*” and the QPPTW algorithm is marked as “Q”.

The graphs show that the number of labels that are generated when the ASQPPTW algorithm routes the arriving aircraft is more likely to be bigger than it is for departures. On average, ASQPPTW would generate 65 labels for each departing aircraft, compared to 112 labels that are generated for each arriving aircraft. Similarly, for the QPPTW algorithm, on average 415 labels were generated for solving each departing aircraft, compared to 467 labels that were needed for arriving aircraft. The standard deviation in the number of labels that are generated

for departing and arriving aircraft is quite different for both algorithms. The standard deviation for departing aircraft routed using the ASQPPTW is around 60 compared to 118 for arriving aircraft. For QPPTW, for departing aircraft is around 42 compared to 242 for arriving aircraft. The standard deviation is significantly bigger for arrivals because they are routed after the departures. As mentioned earlier in Chapter 5, it is more effective to route departures first. For this reason, many arriving aircraft have to reroute as there are departing aircraft already using taxiways. This makes the number of labels that are generated more variable for arriving aircraft, as there are more “ways” for an aircraft to reach its destination due to the increased time windows. For this reason, the number of labels that are generated is affected by the number of previously routed aircraft that are moving on the airport at the same time.

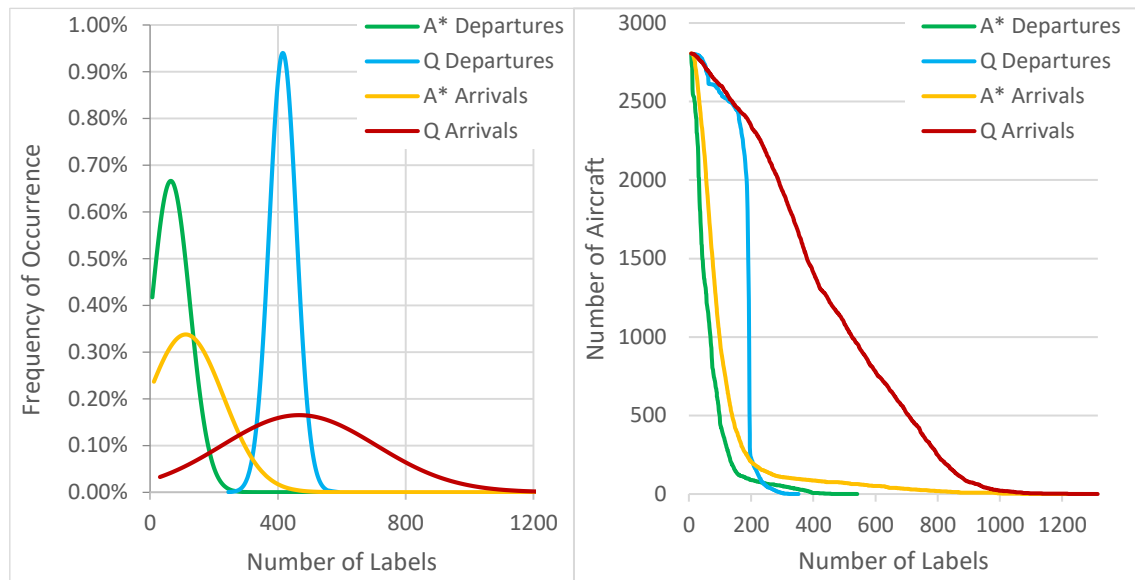


Figure 8.2: Distribution of labels generated by each algorithm (left) and cumulative graph that shows the number of aircraft that generate “x” labels or more (right) for Zurich airport (arrivals and departures)

In order to verify that the standard deviation is high for arrivals due to the previously routed departing aircraft, the experiments were executed for arriving and departing aircraft separately. Figure 8.3 shows the distribution for the labels that are generated for each algorithm on the left as well as the cumulative graph that shows the number of aircraft that generate “x” labels or more on the right, solving the arriving and departing aircraft separately, without affecting one another.

The experiments and graphs show that the standard deviation is smaller than when arriving aircraft are solved after the departing aircraft. The average number of labels that were generated with the ASQPPTW algorithm for departing aircraft was 70 when solved separately compared to 112 when solved after the departures were routed. Similarly, for QPPTW the

average is around 195 when solved separately compared to 467 when solved after the departures.

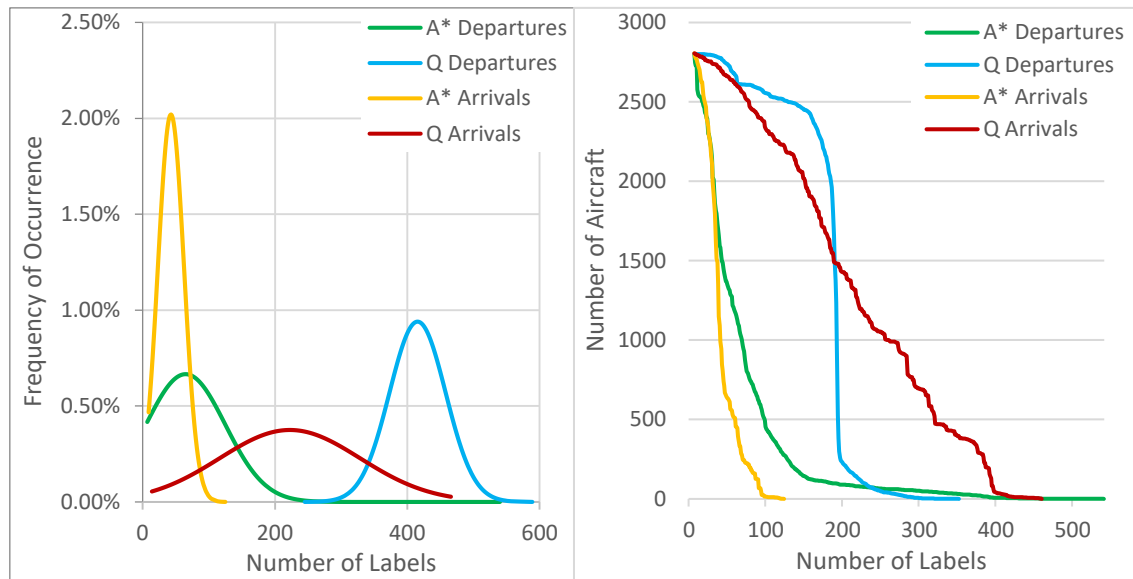


Figure 8.3: Distribution of labels generated by each algorithm (left) and cumulative graph that shows the number of aircraft that generate “x” labels or more (right) for Zurich airport (arrivals and departures separate)

Moreover, not only more labels are generated when arrivals are solved after departures, but there is a wider distribution of values. The standard deviation arrivals when solved separately with the ASQPPTW algorithm is 80 compared to 118 that is the case for arrivals when solved after departures. The same happens with QPPTW as the standard deviation for arrivals solved separately is 117 compared to 242 for when arrivals are solved after departures. This can be seen from the two graphs as well, when the arrivals (yellow and red line) are much more widely distributed across the graph, reaching up to 1200 labels when arrivals are solved after departures, compared to a much narrower distribution (less than 500 labels max) when solved separately.

This indicates that more labels are generated when there are more aircraft (previously routed) moving around the airport. This is expected as anything different than the shortest path will produce extra labels in order to find the next best available path, which is the main aspect of using time windows (that both QPPTW and ASQPPTW have).

Furthermore, Figure 8.3 shows that the average number of labels that are generated when ASQPPTW is used, is not very different for arriving and departing aircraft (see yellow and green line). The average number for departures is 65, compared to 43 that is for arrivals. However, for QPPTW there is a significant difference in the number of labels that are

generated between arrivals and departures. The average number of labels generated for departures is 416 compared to 223 that is the value for arrivals.

This also provides another indication to why the ASQPPTW algorithm when compared to the QPPTW algorithm is providing better results in execution times for departing aircraft against the arrivals. The QPPTW algorithm generates significantly less labels when it solves the problem for arrivals than for departures. Even though ASQPPTW generates similar labels for arrivals and departures, as a percentage, departures seem to route much faster than arrivals when compared to the QPPTW algorithm.

8.5.2 Tests on a Different Airport Layout

In order to verify that the number of labels that are generated for each type of aircraft (arrival or departure) is not due to the type of the airport, the experiments were also run for Arlanda airport.

Table 8.11: Number of labels generated in Arlanda airport (ASQPPTW vs QPPTW)

	QPPTW		ASQPPTW	
	Total labels	Average labels/flight	Total labels	Average labels/flight
Day 1	515978	634	98397	121
Day 2	512853	640	86463	108
Day 3	532216	686	85844	111
Day 4	543317	651	104239	125
Day 5	590646	719	117089	142
Day 6	470897	628	87807	117
Day 7	503183	643	89349	114
Total	3669090	657	669188	120

Table 8.11 shows, the average number of labels that is generated in Arlanda airport when the aircraft are routed with the QPPTW algorithm is 657 compared to 120 that is the number for when the ASQPPTW algorithm is used. This is an 82% reduction of labels that are generated when the ASQPPTW algorithm which is very close to the 80% reduction that was noticed when the same experiment was executed for Zurich airport.

Figure 8.4 shows the distribution of the number of labels that are generated for each flight for both of the algorithms on the left as well as the cumulative graph, that shows the number of aircraft that generate “x” labels or more, on the right. Similarly to the results observed for Zurich airport, the number of labels that are generated by ASQPPTW have a smaller mean

value compared to the QPPTW algorithm, but also a smaller standard deviation (140 versus 306). As expected, the number of labels that are generated for Arlanda airport are higher due to the size of the airport and -as a result- the number of vertices and arcs. It is the same reason that was mentioned in the previous section (Section 8.4) where the execution times were higher for both algorithms (ASQPPTW and QPPTW) in Arlanda airport. However, even though the number of labels that are generated when the aircraft are routed around Arlanda airport are slightly higher, the distribution pattern is quite similar to the one of Zurich airport (Figure 8.1).

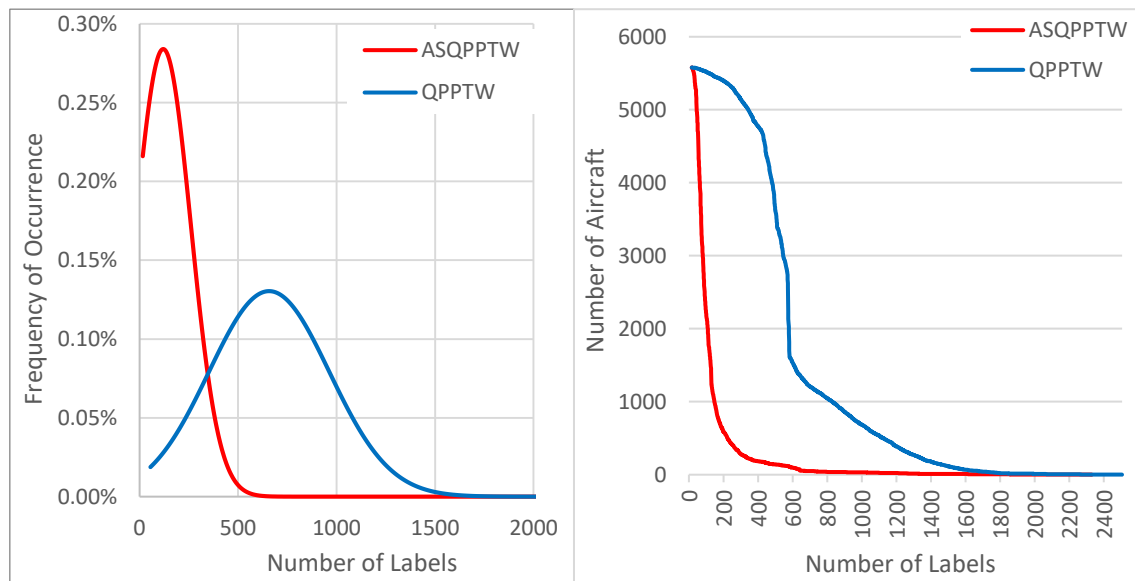


Figure 8.4: Distribution of labels generated by each algorithm (left) and cumulative graph that shows the number of aircraft that generate “x” labels or more (right) for Arlanda airport

Moreover, the same graph was made separating the arrivals from departures. Figure 8.5 shows the distribution of labels that are generated on the left, as well as the cumulative graph on the right, in more detail, for the two algorithms.

Similar to the results in Zurich airport, the graph shows that more labels are generated for both algorithms when the arriving aircraft are routed compared to when the departing are routed. Furthermore, the standard deviation in the number of labels that are generated for departing and arriving aircraft follows the same pattern that the graph for Zurich has (Figure 8.2). For departing aircraft, the standard deviation using the ASQPPTW algorithm in the number of labels that are generated per routed aircraft is 96 compared to 169 that is the value for arrivals. For the QPPTW algorithm, the value was 57 for departing aircraft and 412 for arriving aircraft.

Even though the average value and the standard deviation for arriving and departing aircraft is a bit higher compared to Zurich airport, the values show the same result. Both algorithms

produce less labels for departing aircraft and the ASQPPTW algorithm needs less labels for both arriving and departing aircraft.

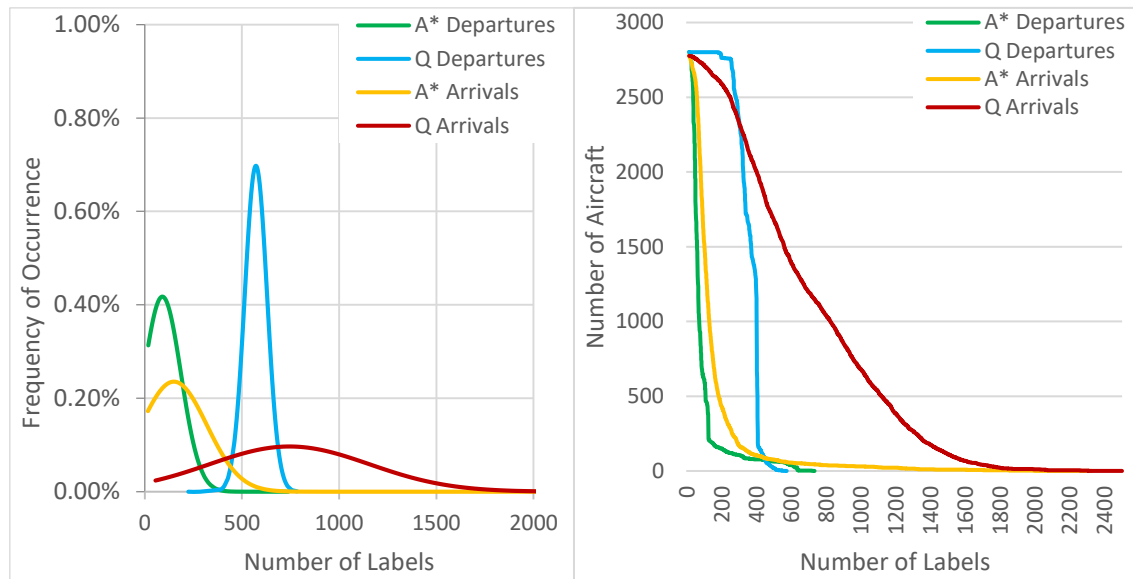


Figure 8.5: Distribution of labels generated by each algorithm (left) and cumulative graph that shows the number of aircraft that generate “x” labels or more (right) (arrivals and departures - Arlanda airport)

Finally, the experiments were executed without solving the arrivals after the departures but rather independently. This was done in order to verify that the reason that there are significantly more labels generated for routing the arriving aircraft compared to the labels that are generated from departing aircraft is because the arrivals are routed first and not because of the airport layout.

As Figure 8.6 shows, the number of labels that are generated when arrivals and departures are solved independently is different than when solved as a whole. On average, the ASQPPTW algorithm would generate 65 labels per arriving aircraft with a standard deviation of 21 compared to the QPPTW algorithm that requires 365 labels per arriving aircraft on average with a standard deviation of 157. It is important to mention that the results are again very similar to Zurich airport and that the distribution of the labels that are generated with each airport as shown in Figure 8.6 and Figure 8.3 follow the same pattern.

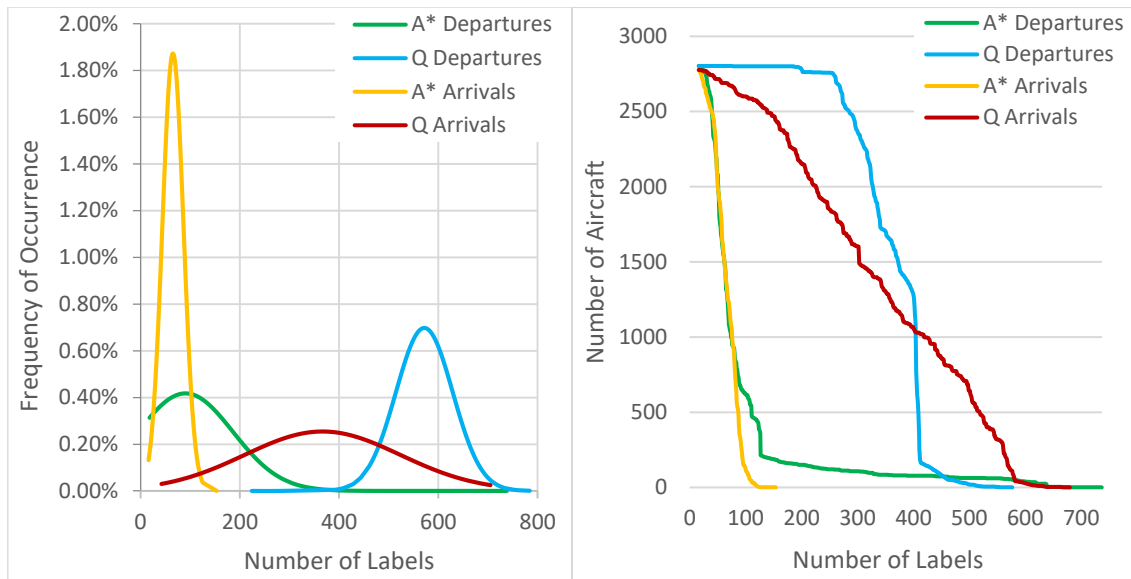


Figure 8.6: Distribution of labels generated by each algorithm (left) and cumulative graph that shows the number of aircraft that generate “x” labels or more (right) (arrivals and departures separate - Zurich airport)

8.5.3 Prioritising Arrivals

So far, all of the experiments were executed with the departures being prioritised. In order to investigate whether the results are affected by the priority of arriving or departing aircraft the same experiments were executed again with the arrivals routed first. As mentioned in Chapter 5, the set of aircraft that is routed first is implicitly prioritised over the other. Moreover, it was observed that prioritising the arrivals results in large delays to departing aircraft as many of them need to initiate the pushback process significantly earlier in order to avoid being disrupted by arriving aircraft. These delays result in examining a larger number of alternative ways for an aircraft to reach its destination that leads to more labels being generated.

After running the experiments, a significantly larger number of labels were generated for routing each flight. Figure 8.7 shows the distribution of labels that are generated on the left, as well as the cumulative graph on the right, with the ASQPPTW and QPPTW algorithms when the arrivals were prioritised. The ASQPPTW algorithm again produces significantly less labels for each aircraft compared to the QPPTW algorithm. More specifically, ASQPPTW generates on average 170 labels for routing an aircraft, whereas QPPTW on average generates 769 labels. This provides more proof that the ASQPPTW algorithm is performing significantly better regardless of which set of aircraft is prioritised (arrivals or departures).

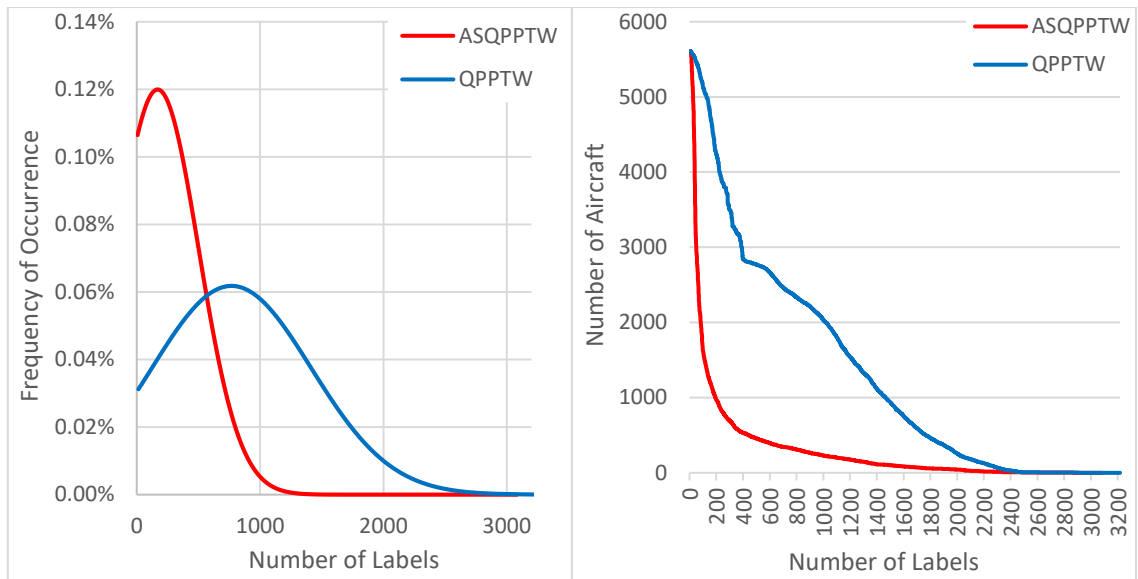


Figure 8.7: Distribution of labels generated by each algorithm (left) and cumulative graph that shows the number of aircraft that generate “x” labels or more (right) when arrivals are prioritised

Furthermore, the execution times are also significantly better with ASQPPTW compared to the QPPTW algorithm. When the arrivals are prioritised the ASQPPTW algorithm performs even better compared to QPPTW.

Table 8.12: Execution times in milliseconds for routing aircraft when arrivals are prioritised (QPPTW vs ASQPPTW)

	No. of aircraft	QPPTW [ms]	ASQPPTW [ms]	Improvement
Day 1	818	6496	1820	72%
Day 2	806	5946	1604	73%
Day 3	781	4017	828	79%
Day 4	837	6588	1726	74%
Day 5	825	4142	900	78%
Day 6	755	3622	827	77%
Day 7	787	4431	1051	76%
Total	5609	35243	8756	75%

As it is shown in Table 8.12 it took 35 seconds for the QPPTW algorithm to solve the full week, whereas the ASQPPTW algorithm solved the problem in just 8.8 seconds solving the problem 75% faster. This follows the suggestion that the ASQPPTW algorithm performs even better when the airport is more congested, and more aircraft are delayed.

Furthermore, the distribution of labels that are generated with each algorithm when arrivals are prioritised in order to have a better understanding of the labels that are generated with each algorithm, the same graph was made separating the arrivals from departures. Figure 8.8 shows the distribution of labels that are generated in more detail on the left as well as the cumulative graph on the right, separating the arrivals from the departures.

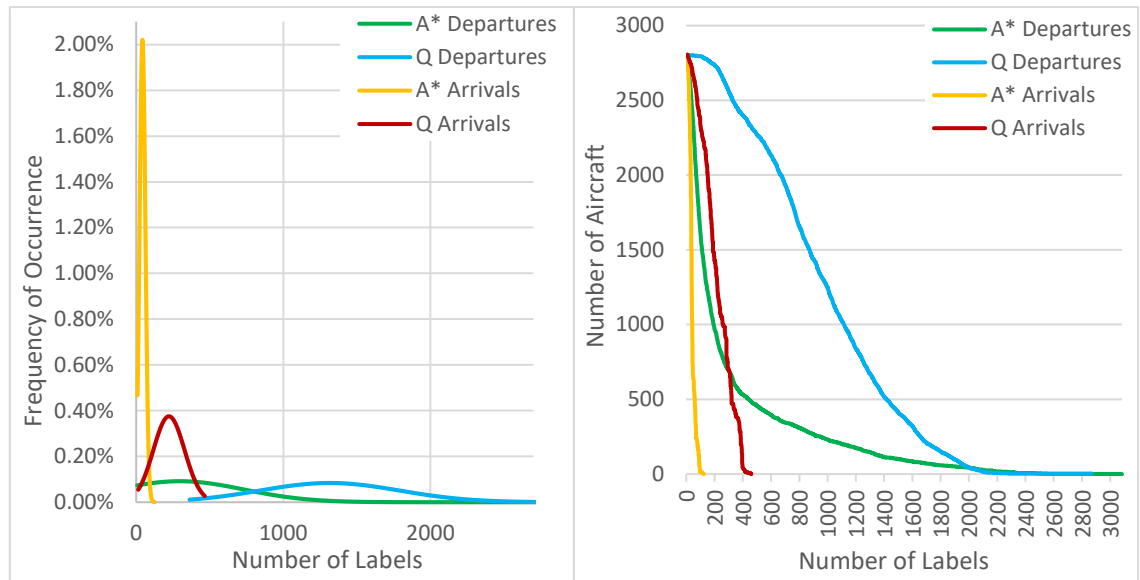


Figure 8.8: Distribution of labels generated by each algorithm (left) and cumulative graph that shows the number of aircraft that generate “x” labels or more (right) when arrivals are prioritised

The distribution of labels that are generated per departing aircraft is considerably wider than the arriving aircraft. Moreover, the distribution of labels that are generated when departing aircraft are routed is also much wider than the distribution of arriving aircraft when the departures were prioritised. On average ASQPPTW generates 296 labels for each departing aircraft when arrivals are prioritised compared to 112 that are generated for each arriving aircraft when departures are prioritised. Similarly, QPPTW generates on average 1315 labels for each departing aircraft when arrivals are prioritised compared to 476 that are generated for each arriving aircraft when departures are prioritised.

In the previous two subsections (8.5.2 and 8.5.3) the departures would produce far less labels than the arrivals when departures were prioritised. The results in this section further prove that the number of labels that are generated is significantly affected by delays that are produced by previously routed aircraft.

8.5.4 Traffic and ASQPPTW Performance

The experiments were also executed in a less congested airport scenario, where there were less departing aircraft. This configuration was tested in order to verify that the number of labels that are generated is affected by the delays that happen in the airport. To simulate the decreased traffic and less delays, every second departing aircraft was removed from the list of aircraft that were routed. This affected the departing aircraft as it was easier for them to pushback and move around having less chances to be delayed, as well as the arriving aircraft that could move around easier without many paths being blocked by departing aircraft that are pushing back and block important taxiways.

Figure 8.9 shows how the labels that were generated from this experiment are distributed on the left, as well as the cumulative graph on the right, using ASQPPTW and QPPTW and by separating the distribution of arriving aircraft and departing aircraft.

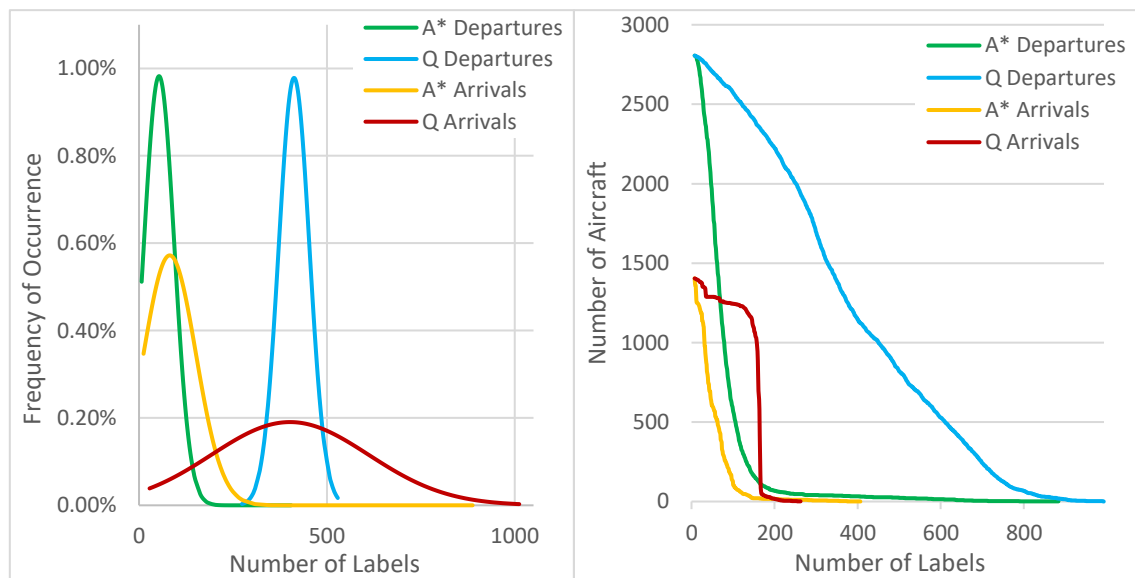


Figure 8.9: Distribution of labels generated by each algorithm (left) and cumulative graph that shows the number of aircraft that generate “x” labels or more (right) with less traffic

The average number of labels that were generated for each flight using the ASQPPTW algorithm was 53 for departing aircraft and 82 for arriving aircraft. The average number of labels that were generated for each flight when the QPPTW algorithm was used was 413 for departing aircraft and 402 for arriving aircraft.

Comparing these results with the results that were observed when all arrivals and departures were routed in Section 8.5.2, the ASQPPTW algorithm performed better in the airport with the reduced traffic. When routing all of the flights, the ASQPPTW algorithm would generate 84% less labels when routing departing aircraft compared to the QPPTW algorithm, whereas

when a part of the total number flights were routed (reducing the traffic and the delays) 87% less labels were generated with the ASQPPTW algorithm. Similarly, for the arriving aircraft the ASQPPTW algorithm would outperform the QPPTW algorithm when routing all the flights by 76%, whereas an 80% improvement was recorded when a part of the total number the flights were routed. This provides more evidence that ASQPPTW performs even better compared to QPPTW when there is more traffic and the routing problem gets more complicated. Moreover, it proves that the number of labels that are generated is indeed affected by the delays that can happen in the airport.

8.5.5 Label Generation and Airport Characteristics

In order to have a better understanding of the ASQPPTW algorithm and the resulted decrease in label generation, the improvement in the number of “labels that are generated” was correlated (separately) with two parameters. The first parameter is the distance of the shortest path which is the minimum distance that an aircraft needs to traverse in order to reach its destination and the second parameter is the total taxi time that was required for the aircraft to reach its destination. These two parameters were chosen in order to investigate if the ASQPPTW algorithm performs better than the QPPTW algorithm as the distance and/or the time that the aircraft is moving around the airport increases. After performing a regression analysis on the previously mentioned variables the results are displayed in Figure 8.10 and Figure 8.11.

Figure 8.10 shows the improvement (as a percentage) in the number of labels that are generated by ASQPPTW compared to QPPTW in relation to the time duration that the aircraft was travelling - including all the delays that may have occurred. As the graph shows the results are quite scattered, with the majority of the observations (duration of travel) for departing aircraft being around 201 seconds and an 85% improvement in the number of labels generated and 190 seconds and a 75% improvement for arrivals. After performing a linear regression, the functions of which are displayed in the graph for arrivals and departures (separately), the coefficient of determination (R^2) for departures is around 53% and for arrivals around 5%. As the travel duration increases it is observed that the improved performance of ASQPPTW gradually decreases for departing aircraft. However, it is important to mention that the coefficient of determination is not very high so as the travel duration increases it is harder to predict how much less labels ASQPPTW will generate compared to QPPTW. This is more apparent for the arriving aircraft where the coefficient of determination is very low indicating that there is a very low correlation between the travel duration of an arriving aircraft and the improvement of the ASQPPTW algorithm in label generation. This is expected though, as the duration of the arrivals is greatly affected by the

departing aircraft that are prioritised. This causes many of the shortest routes to be blocked, making it harder for both algorithms to solve the problem and less predictable when it comes to the number of labels that will be generated.

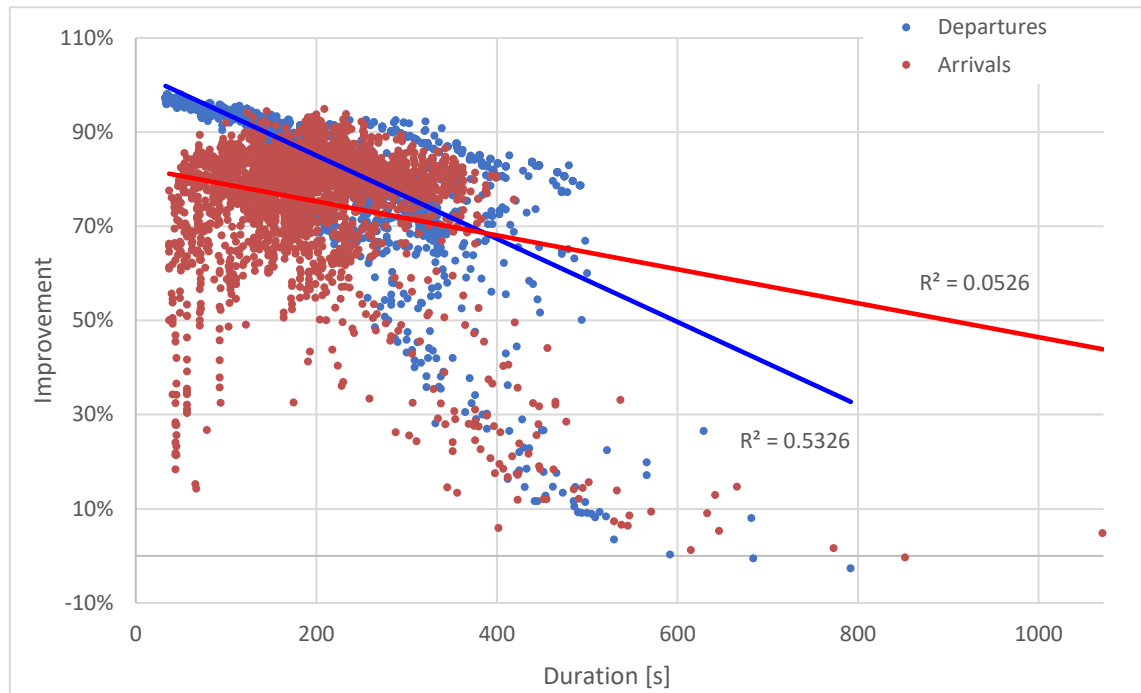


Figure 8.10: The improvement in the number of labels generated in relation to the time duration travelled

Similarly, for Figure 8.11 the graph shows the improvement in number of labels that are generated by ASQPPTW compared to QPPTW in relation to the distance that the aircraft has travelled. In this graph, the results are even more scattered, with the majority of the observations (distance travelled) being 1519 meters and an 85% improvement in label generation for departures and 1456 meters and a 76% improvement for arrivals. After performing a linear regression (see functions on the graph), the coefficient of determination (R^2) for departures is around 19% and for arrivals around 3%. As the distance that is travelled increases it is observed that the improved performance of ASQPPTW gradually decreases for departing aircraft. The coefficient of determination however, is quite low so it is hard to tell if the two variables are correlated. For the arriving aircraft, the coefficient of determination is even lower demonstrating again the unpredictability of the improvement that the ASQPPTW algorithm can have when there are many delays and the shortest path is often not available.

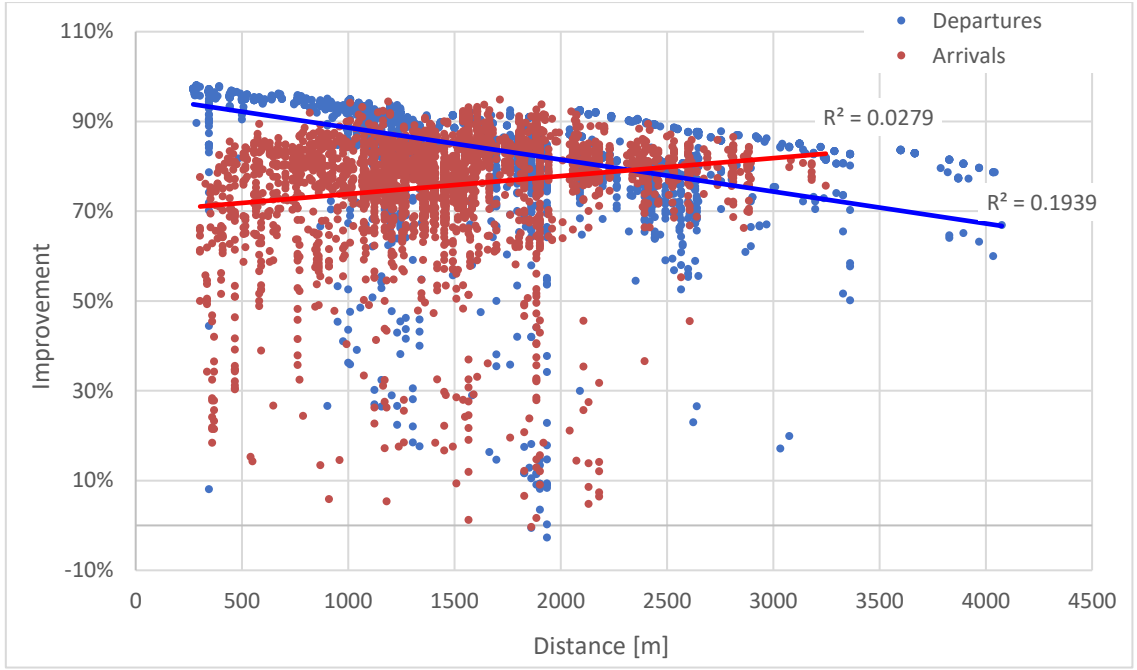


Figure 8.11: The improvement in the number of labels generated in relation to the distance travelled

As the results show, there is not a strong correlation between the travelled distance or the travel duration of a flight and the improvement that the ASQPPTW algorithm can provide in reduced numbers of generated labels compared to QPPTW. For this reason, the airport characteristics were not investigated in more depth. Evidently the traffic in the airport and the delays that happen can be a more crucial factor for the increased performance of ASQPPTW against the QPPTW algorithm as demonstrated in the previous subsections.

8.5.6 Examination of the Heuristic

In favour of providing a better insight into the benefits of the heuristic that was used in the ASQPPTW algorithm, another heuristic was also tested. As discussed in Section 8.2.2 another option would be to use a heuristic that is based on distance.

While the experiments were executed, the extra execution time and memory that was required for the ASQPPTW with the alternative heuristic was also tracked. After running the experiments, the results show that the use of storing the value of the cost from moving from each node to each stand or runway did not affect the execution time of ASQPPTW. For each of the 317 nodes which the graph consists of, the algorithm would load and store the “minimum time duration” that is needed by an aircraft to reach a stand or a runway, for each of the 110 stands and runways (34,870 values in total). The loading of these values would happen in less than 10 milliseconds which is a very low increase considering that the

execution time of the ASQPPTW algorithm can be faster by at least 7 seconds compared to the QPPTW algorithm.

The memory requirements would increase by approximately 200 Megabytes when ASQPPTW is used compared to QPPTW but overall the ASQPPTW algorithm needs less than 700 Megabytes to solve the problem, which is not a significant requirement for any modern personal computer.

After solving the same problem using two different heuristics, the heuristic chosen for ASQPPTW performed significantly better than the distance heuristic. Table 8.13 shows the number of labels that are generated with each set-up (QPPTW, ASQPPTW and QPPTW with a distance based heuristic). The results show that even though the distance heuristic provides a significant improvement compared to QPPTW, it cannot reach the efficiency of ASQPPTW. On average, the ASQPPTW algorithm would generate 43% less labels compared to the QPPTW with the distance based heuristic.

Table 8.13: Number of labels that are generated with each set-up

	QPPTW		ASQPPTW		Distance Heuristic	
	Total labels	Average labels/flight	Total labels	Average labels/flight	Total labels	Average labels/flight
Day 1	363322	444	77806	95	135766	166
Day 2	352302	437	74294	92	127706	158
Day 3	334396	428	63640	81	117136	150
Day 4	369063	441	81662	98	139236	166
Day 5	386553	469	69472	84	133099	161
Day 6	328415	435	65500	87	114497	152
Day 7	340136	432	62979	80	115924	147
Total	2474187	441	495353	88	126195	157

As mentioned in Section 8.2.2 the reason that the heuristic of ASQPPTW outperforms the distance based heuristic and any other heuristic that can be used for A* type algorithms is because ASQPPTW heuristic provides a more accurate estimation. The results show that the accuracy of the heuristic is important, as the distance heuristic will almost always expand to more nodes and generate more labels, eventually increasing the execution time.

8.6 Conclusions

In this chapter, a new algorithm that can solve the routing and scheduling problem of aircraft faster was introduced. The new approach is based on the A* algorithm and is designed to provide a directed search to the quickest path problem with time windows. The algorithm extends the popular QPPTW algorithm for finding the quickest path for aircraft and takes into consideration important parameters such as the stand holding process and the pushback process. The presented approach is based on finding the remaining time that is needed to traverse the shortest path instead of using an estimation of the distance to the target destination, making it possible to have a compatible and advantageous use of time windows, that are necessary for routing multiple aircraft that interact with each other.

Experiments were run to test the efficiency of the new algorithm compared to QPPTW. Both algorithms solved 7 days of the routing process of aircraft in Zurich airport as well as Arlanda airport. The execution time for both of the algorithms was measured for solving the full problem as well as solving the arrival and departure parts of the problem separately. The results have been very encouraging and show that the new A* approach of solving the problem (ASQPPTW) outperforms the QPPTW algorithm on average by 46% to 67% (in Zurich and Arlanda airport respectively) in execution time when used for solving the full routing problem. When the performance of the program was tested, ASQPPTW could route the aircraft of a full week (which is 5609 aircraft in this case) on a simple notebook computer in 7.2 seconds. That is on average less than 1.3 milliseconds per flight which makes it more than adequate for real time routing but also provides a useful routing method that can be used in conjunction with other processes that happen in the airport such as the gate allocation process and the runway sequencing process. This can be very useful for optimising the full problem instead of finding the optimal solutions of smaller problems and lead to a solution that will increase the fuel efficiency of aircraft that are moving on airports and increase the capacity of airports.

Moreover, the way that the ASQPPTW algorithm expands was investigated and the number of labels that are generated for routing each aircraft were calculated. The results show that the ASQPPTW algorithm generates far less labels compared to the QPPTW algorithm, which explains why there is an improved efficiency in execution times. The differences in number of generated labels between arriving aircraft and departing aircraft was also investigated in Zurich and Arlanda airport. Routing the departing aircraft before the arriving aircraft considerably affects the speed of which both the QPPTW and ASQPPTW algorithms can find the route for departing aircraft. However, the ASQPPTW algorithm is much more efficient in solving the problem even in this case. This proves that using a heuristic that calculates the

remaining time can outperform a routing algorithm that expands towards all directions (such as QPPTW) in almost every case. Moreover, it is shown that an increase in the number of delays results in more labels being generated for both algorithms and that ASQPPTW performs even better in situations where there are more delays (high traffic).

Integration of the Ground Movement Problem in a Flight Simulator Cockpit

9.1 Introduction

In this chapter, the real-world difficulties of applying the research that has been presented in the previous chapters, is investigated. The developed project demonstrates how this research can be used in practice in a wider and more integrated framework. The ground movement model that was developed in Chapters 4 and 8 is applied within a flight simulator cockpit, and the various integration and applicability issues are discussed.

There has been extensive research in the ground movement problem as was presented in Chapter 2, but so far, the overwhelming majority has not been tested in airports. Many processes happen in parallel while aircraft are moving on the ground. The ground movement problem affects and is affected by processes from other disciplines outside the traditional area that the current literature is focusing on. Two of these processes that have arisen from the research that is taking place in the Institute of Aerospace Technology of the University of Nottingham and have been underlined by the INNOVATE project are the navigation and positioning of the aircraft and monitoring the human workload of pilots. Significant research is happening on these areas and testing the ground movement model within the latest advances on this area is therefore very useful. Creating a physical demonstrator that combines new technologies and the latest research can provide useful insight into how to apply this research.

The flight simulator cockpit was developed as part of this research and in conjunction with the research of two other PhD students from the INNOVATE project. The purpose of the project is to demonstrate the applicability of novel aerospace technologies and how they can

be integrated in an operational system. The project consists of three different processes that work in parallel and with each other. The first process is the navigation and positioning of the aircraft, the second process is the real-time routing of aircraft and the third process is the physiological monitoring of human performance to assess the impact of the other two processes on mental demand.

9.2 Advanced Receiver Autonomous Integrity Monitoring

The first process (navigation and positioning of the aircraft) uses the Advanced Receiver Autonomous Integrity Monitoring (ARAIM) system, which is a new aircraft based augmentation system technique. The ARAIM technique uses the global navigation satellite system receivers to navigate aircraft around the world with a high precision. The ARAIM system aims to reduce the dependence of the support systems that are needed from the ground or satellite based systems that are currently in use. This method provides a real-time positioning method for the aircraft. More details of the system that is used can be found on the paper of Paternostro et al. (2016).

The ARAIM algorithm that was used for this project is connected to a commercial flight simulator software (called X Plane 10) through a tool (X-Plane Connect) that was developed by NASA (Teubert, 2008). X-Plane Connect (XPC) is an open source research tool that makes it possible to interact with the X-Plane software. As stated on the NASA website (NASA 2017), through the XPC tool it is possible for the user to receive real time information from the simulated aircraft in X-Plane using functions written in various programming languages such as MATLAB, Python, Java, C or C++. By using this tool, it is possible to visualise flight paths, test control algorithms, simulate an active airspace, or generate out-the-window visuals for in-house flight simulation software.

For this research, the XPC tool has been used for extracting information that is associated with the real-time position and attitude of the aircraft. This information is used by the ARAIM tool that has been developed by Paternostro et al. (2016) for the navigation of the aircraft.

9.3 Solving the Ground Movement Problem and Applying the Solution

In this section, the model that was developed for the physical demonstrator is explained. In the first part, the way that the location of the aircraft on the graph can be found by using GPS

locations is described. In the second part, the real-time routing and navigation of the aircraft is presented.

9.3.1 Using the GPS Locations of the Aircraft to Find their Position on the Graph

The routing system uses a directed graph model of the airport. To use this, it needs to locate each aircraft in a node of the graph. In order to find the location of the aircraft on the graph, using real time GPS coordinates (such as those from XPC) it is important to convert and project the geographic coordinates to the x and y coordinates on the graph that the model is using (see airports and graphs section in Chapter 3)

Since the area that is considered (the airport) is rather small compared to the size of the earth, these calculations do not take into consideration the curvature of the earth. The (x, y) positions of the nodes are therefore calculated in a linear way, since the distance between two (x, y) points and two actual locations on the earth is almost identical. The error that is generated when comparing the distance between a point and a nearby node is less than a centimetre, not practically affecting the accuracy of finding the closest node at which the aircraft is located at any time.

Table 9.1: Table of Notation and Definitions

Constants	Explanation
W	The width of the airport graph picture in pixels
H	The height of the airport graph picture in pixels
φ_m	The coordinate of the most northern location of the area that is considered
λ_m	The coordinate of the most western location of the area that is considered
φ_M	The coordinate of the most southern location of the area that is considered
λ_M	The coordinate of the most eastern location of the area that is considered
Variables	Explanation
φ	The latitude of the aircraft
λ	The longitude of the aircraft
x	The x coordinate of the aircraft on the graph/screen
y	The y coordinate of the aircraft on the graph/screen

The x, y coordinates are calculated using Equations 9.2 and 9.3, and Table 9.1 shows the definitions of the notation that was used. For this calculation, the dimensions of the picture

(airport graph representation) are needed as well as the geographic coordinates of the upper left corner and the bottom right corner of the area that the picture is going to include.

$$x = \frac{W(\varphi - \varphi_m)}{\varphi_M - \varphi_m} \quad (9.2)$$

$$y = \frac{H(\lambda_m - \lambda)}{\lambda_M - \lambda_m} \quad (9.3)$$

Once the location of the aircraft on the graph is found, the node that is the closest to the aircraft needs to be identified. Since, however, the aircraft can travel from node A to node B, but there can be a node C that the aircraft is closest to, finding the node that has the minimum distance from the aircraft can be misleading. An example of this situation can be seen in Figure 9.1. where the aircraft is traveling from node A to node B and is closer to node C which is not on its path. In this case node B should be considered as the closest node, rather than node C.

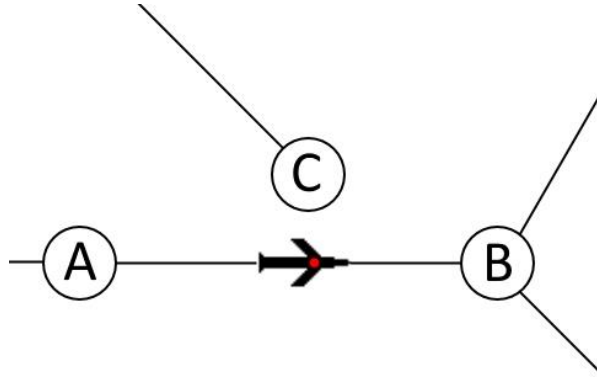


Figure 9.1: Finding the closest node

For this reason, the algorithm that was created to find the node which the aircraft is closest to is searching for the path segment (rather than node) that the aircraft is closest to. When the closest segment is found, the node which the aircraft is closest to (start node or end node of the segment) is found, by comparing the distance of the aircraft and the two nodes of the segment.

$$l^2 = (x_b - x_a)^2 + (y_b - y_a)^2 \quad (9.4)$$

$$t = \frac{(x - x_a)(x_b - x_a) + (y - y_a)(y_b - y_a)}{l^2} \quad (9.5)$$

$$d = \sqrt{(x_a + t(x_b - x_a) - x)^2 + (y_a + t(y_b - y_a) - y)^2} \quad (9.6)$$

Equations 9.4, 9.5 and 9.6 show how the distance (d) between the aircraft and a node is found. This distance is calculated for all of the nodes of the graph, and the node that has the smallest value denotes the node closest to the aircraft. In case the shortest distance to a node is greater than 100m, the aircraft is considered to be out of the range of the airport.

9.3.2 Real Time Navigation of an Aircraft on the Ground

The previously mentioned algorithm finds the locations of all of the aircraft that are in the airport. Then the aircraft that is controlled by the user/pilot is routed to a runway for take-off by taking into consideration the location of other aircraft. The model uses the ASQPPTW algorithm to route the aircraft from the node which the aircraft is currently at, to the runway that is used for aircraft that are taking-off for at that time. This process takes into consideration other aircraft by blocking the nodes on which other aircraft are. When the ASQPPTW algorithm is executed the nodes that are occupied by other aircraft are not available for expanding. It is important to mention that this methodology considers that the aircraft have full control of their routing process and are not being scheduled by the air traffic controllers. Although the optimal available path is found, this is not a globally optimal solution.

The above process is implemented within its own thread and is executed approximately every 2 seconds. The position of the aircraft is updated each time, as well as the available path for the aircraft to reach the runway.

The available path and the positions of the aircraft are highlighted on the map to advise the controller on the path that has to be followed in order to reach the runway. Figure 9.2 shows an example of the output for an aircraft traveling from a gate in terminal A to runway 14/32 in Zurich airport.

9.4 Physiological Monitoring of Human Performance

The third process of the integrated project monitors the human performance to assess the impact of the other two processes on mental demand. It uses the FLIR A65sc thermal infrared camera which can monitor the thermal features of the person's face that is using the flight simulator. The developed model for this process can track the face of the user and monitors the temperature difference of specific points in the face such as the pupils and the nose.

Marinescu et al. (2017) show that there is a high correlation between the decrease of the temperature of the nose and the mental workload.

This process evaluates the effects that the previously mentioned processes (see Sections 9.2 and 9.3) have on the pilot. Since these processes provide new information to the pilot, it is useful to monitor his mental workload in order to be able to evaluate how safe, and how demanding for the pilot, the implementation of these processes is.

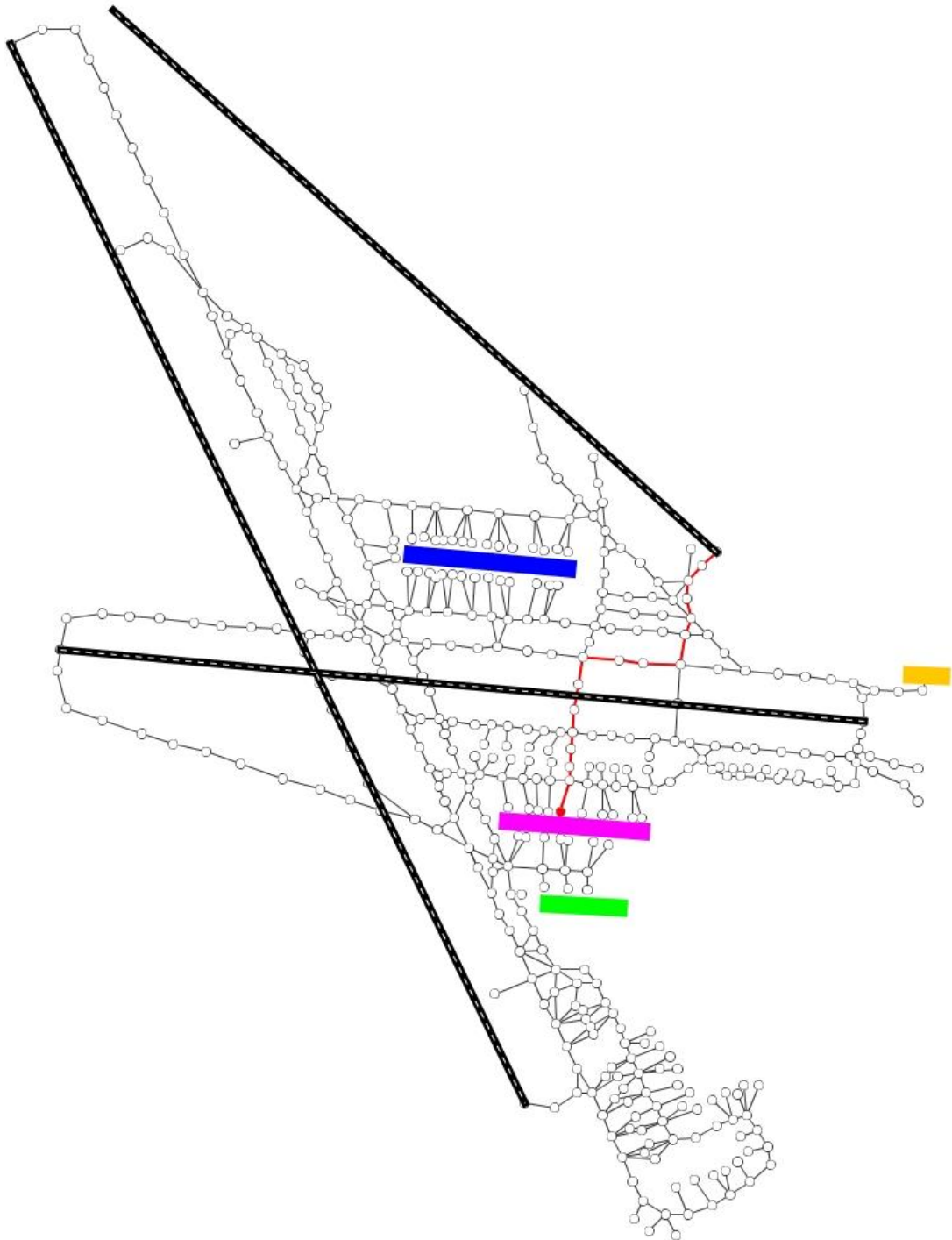


Figure 9.2: Graph representation of Zurich airport (quickest path highlighted)

9.5 The Physical Demonstrator

The developed model integrates the processes that were described in the previous sections in a physical demonstrator. The physical demonstrator is a flight simulator cockpit that uses several instruments and controllers.

The demonstrator is located in the Aerospace Technology Centre and consists of the following hardware:

- Yoke
- Throttle quadrant
- Rudder Pedals
- Switch Panel
- Instrument Panel
- Radio Panel
- Multi Panel
- High Specification Research PC.
- FLIR Infra-Red camera4 LED screens



Figure 9.3: The flight simulator cockpit

Figure 9.3 shows the flight simulator cockpit without the thermal camera.

The physical demonstrator runs the X-Plane flight simulator software. Using the instruments described above it is possible for the user to control an aircraft and complete various tasks such as routing, take-off, navigation and landing.

The routing process that was described in Section 9.3 of the physical demonstrator uses as an input the geographical coordinates that is the output of the ARAIM algorithm (see Section 9.2). The ARAIM algorithm finds the exact location of the aircraft that is being controlled by the operator/pilot of the cockpit simulator. These coordinates are then passed to the ground movement program. The output of this program is a graph representation of the airport, with the quickest path being highlighted (see Figure 9.2).

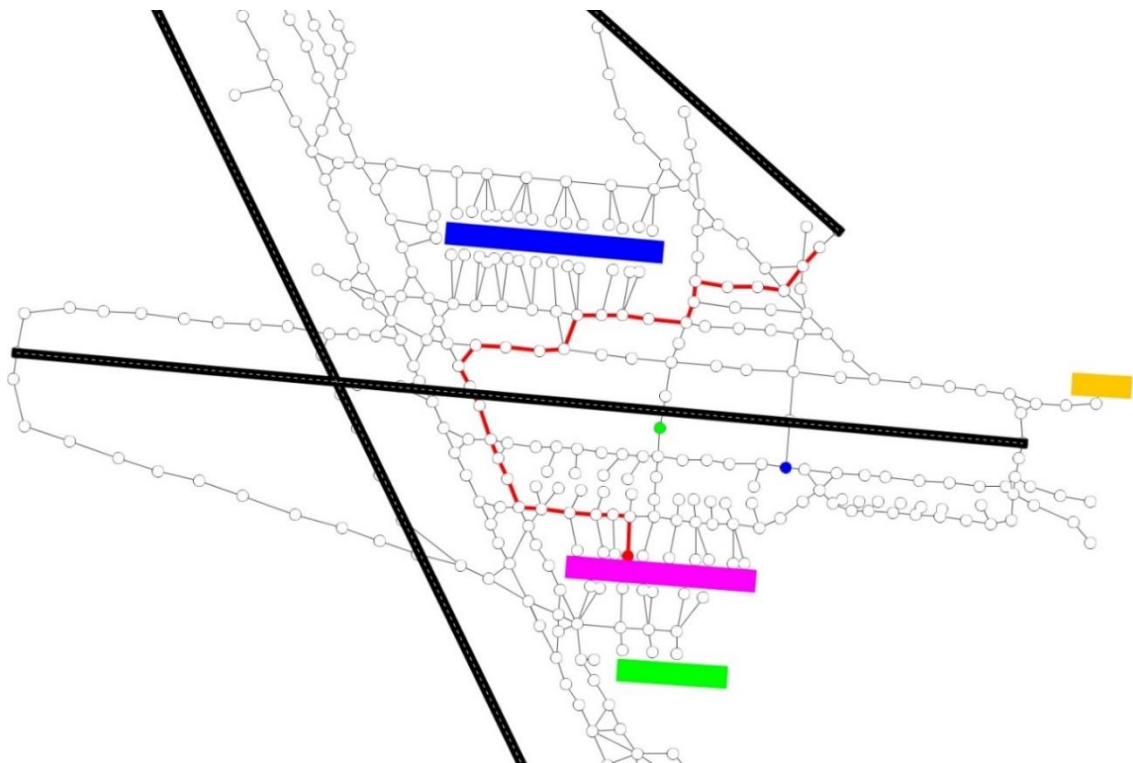


Figure 9.4: Quickest path highlighted after taking into consideration the positions of other aircraft

The highlighted path is updated every second and guides the user to the runway where the take-off procedure can initiate. The ground movement program takes into consideration the locations of other aircraft that are moving on the airport. The algorithm will highlight the fastest path that leads to the runway avoiding the paths that are blocked by other aircraft. Figure 9.4 shows the quickest path for the same aircraft that has to go from a gate in terminal A to runway 14/32 in Zurich airport, but this time taking into consideration two aircraft that are blocking shorter paths (aircraft are marked in green and blue).

While the user follows the path that is presented by the ground movement tool the human performance and the mental workload are monitored by the thermal camera as described in Section 9.4.

9.6 Conclusions

In this chapter, the integration of the ground movement problem in a flight simulator cockpit and the real-world difficulties of applying this research has been presented. This integration project works as a proof of concept for integrating the algorithms for the routing process into a real system. Various integration issues come up, such as the question of automating the identification of the node to place an aircraft in the model.

The fast execution speed for solving the ground movement process makes it possible to provide the path that the aircraft needs to follow in order to reach its destination in real time. The position of other aircraft on the ground is always taken into consideration when the quickest path is calculated. The path is constantly updated with the latest positions of the all of the aircraft on the ground (including the aircraft that is being routed), in order to guarantee that the path that is shown to the pilot is accurate and always the quickest.

The ground movement process uses as an input the real time GPS coordinates of the aircraft, which is provided by the Advanced Receiver Autonomous Integrity Monitoring system and as an output, provides the necessary information to the pilot in order to be able to navigate around the airport. At the same time, the human performance and the mental workload are monitored. All these processes are integrated into one system that provides an improved way to navigate around the airport. This system could be integrated in aircraft, in order to provide a more automated, safer and optimised movement of aircraft on the ground.

Conclusions

10.1 General Summary

As mentioned throughout this thesis, theoretical models can ignore some practical issues when modelling the ground movement process of aircraft. There is always a level of abstraction when modelling a real-world problem and some parameters that are considered less important are ignored in order to simplify the problem. This thesis considers some of these practical issues and investigates their importance. Moreover, it considers the practicality of implementing them into airport systems and integrating with positioning software.

This thesis has focused on the improvement of the realism of ground movement methods using operational research methodologies. More specifically, processes and parameters that affect the ground movement process have been investigated in order to understand how delays during the routing process of aircraft in airports happen, and how they can be reduced.

Two methodologies have been used for modelling and solving the ground movement problem. After comparing the two approaches, the one that was more appropriate for solving the ground movement problem was chosen, as it was more effective in taking into consideration both arriving and departing aircraft and being able to use alternative paths. Various features were added to the selected approach in order to explicitly implement the pushback process and to use stand holding for departing aircraft. Later the routing methodology was redesigned in order to improve the execution time of the ground movement process. Finally, the methodology for routing and scheduling aircraft was integrated with other processes in a physical demonstrator as proof of concept.

The effects of the pushback process and the delays that can happen were investigated in Chapter 4. In Chapter 5 the stand holding process plus pushback process were implemented within the routing process and the priority between arrivals and departures was examined. A

model with a mixed prioritisation was created with various degrees of prioritisation settings and the delays that can happen when aircraft were routed for each setting were examined. The effects of the layout of the airport were also examined in Chapter 6, providing a better understanding of when and how delays can happen. Chapter 7 introduces an integration framework for solving the gate allocation process while taking into consideration the delays that can happen during the ground movement process. A new algorithm was also introduced in Chapter 8, that can solve the ground movement problem faster. Finally, an implementation of this research was integrated with other novel technologies in a flight simulation cockpit in order to demonstrate the applicability of this research.

In the following sections of this chapter, the key results are presented and the areas where future research seems promising are highlighted.

10.2 Key Results

Better understanding of the pushback process. The effects of the pushback process, when explicitly implemented, have been investigated in this thesis. The delays that can happen due to aircraft that are pushing back or by aircraft that are not allowing other aircraft to push back have been identified and examined. This is an important issue, as many delays do happen during the pushback process, that could pass unnoticed if the process is not explicitly implemented. Knowing where and when these delays can happen is vital in order to be able to minimise them.

Better understanding of the prioritisation effects between arriving and departing aircraft. The delays that happen when arrivals or departures were prioritised were also examined. Furthermore, a mixed prioritisation as well as different levels of mixed prioritisation were also examined. The priority between aircraft can also have a significant effect on the delays that can happen during the routing process and using different prioritisation methods provides an important insight into how these delays happen when the pushback process is explicitly implemented and how they can be reduced.

Improved model for aircraft routing including pushback delays. A model that can route aircraft on the ground has been extended to include the pushback process while the stand holding process is in use. The majority of the experiments in this thesis use these extensions in order to have a more accurate mapping of the delays that can happen during the ground movement process. These extensions provide a more realistic tool for solving the ground movement process of aircraft.

Better understanding of the effects of airport layout. Chapter 6 analysed how the airport layout can affect the routing process of the aircraft and the delays that happen during taxiing. Moreover, the use of alternative paths has been investigated and layouts that contain extra or fewer taxiways were created to observe the delays that can arise or the improvement in reducing long delays.

Integration framework between the routing process and the gate allocation process. An integration framework that can solve the gate allocation problem, by explicitly taking into consideration the delays that can happen during the ground movement process of aircraft has been created and tested. The model tries different allocations and receives detailed feedback with the combinations of aircraft that can produce a delay. This makes it possible to add in the gate allocation process an extra objective for the individual delays that can happen, and attempt to minimise them.

The integrated model takes the wider problem - of optimising the ground processes at an airport - into consideration and can provide an overall good solution instead of optimal solutions for the individual problems. In practice, it can significantly decrease the number and the duration of delays that happen when the aircraft move on the ground, as the results in this thesis shows.

Finding conflicting combinations. An algorithm that can find all the conflicting combinations in a particular allocation (flights to stands) has been created. The algorithm solves multiple combinations of aircraft on an empty graph in order to isolate the aircraft that cause a delay. Delays that could happen between combinations of aircraft if other aircraft of the allocation were missing are also found in order to guarantee that the list of conflicting aircraft contains all the possible combinations of aircraft that can arise between any number of aircraft with that particular allocation to stands. This makes it possible to provide important feedback to the gate allocation process that can be used for avoiding allocations where aircraft conflict with each other.

Faster execution times. The QPPTW algorithm has been modified from a Dijkstra's based algorithm to incorporate an A* based algorithm. A heuristic that can significantly reduce the number of expansion labels that are generated has been implemented. The heuristic finds the remaining time that is necessary for the aircraft to reach its destination if there are no delays to the remaining path. This can help the algorithm only expand towards the node of the quickest path if no delays occur, significantly reducing the search space. The execution time was shown to be reduced by 46% to 67%, depending on the airport. Having a model with a fast execution time makes it more adequate for real time routing but also provides a useful routing method that can be used in conjunction with other processes that happen in the airport such as the gate allocation process and the runway sequencing process.

Practical implementation. A practical implementation of the routing process has also been presented in Chapter 9. The ground movement process has been integrated with processes from other disciplines – such as positioning and navigation process and the mental workload monitoring of the pilot – for the first time, as a proof of concept of the practicality of integration. This provides an insight into the practical implementation issues that can arise when integrating the ground movement process in a physical flight simulator cockpit.

10.3 Future Work

During this research, a number of areas with further research potential have been identified. Some of these areas have been summarised below.

More historic flight data. Executing experiments using historic data is very important for drawing accurate conclusions. However, the available data for this thesis was limited. Having more datasets can make the arguments that are drawn from the conclusions stronger and more reliable.

Use of more airports. This research considers experiments in three different airports. As was observed in Chapters 6 and 8 the layout of the airport significantly affects the efficiency of the algorithm that is used and the execution times. Moreover, the duration of delays, the taxi duration, the number of delays and the use of alternative paths were also significantly affected by the airport that was used. This shows that running experiments in one or only a few airports may not provide generally acceptable results for the ground movement process in general, so there is value in considering new layouts.

A faster integrated model with gate allocation. Even though the ground movement solver and the algorithm that finds conflicting combinations of aircraft in the integrated model in Chapter 7, can route all the aircraft multiple times and find all the bad combinations within seconds, the gate allocation model can currently find a day's allocations only for one terminal in a reasonable time. Since the gate allocation solver is executed multiple times in order to find the best solution, the integrated model takes a considerable amount of time to execute. As mentioned in Neuman's thesis (Section 7.6 of her thesis), using a receding horizon decomposition might improve the execution times. Moreover, some parameters may not be so important for the integrated model - especially the ones that try to reduce the delays that happen during taxiing – since the feedback that the integrated model receives from the ground movement solver might be sufficient for minimising taxiing delays.

Integration with other processes. As was seen in this thesis, integration of airport processes that affect each other can provide an improved overall solution. In this research, the pushback

process was explicitly implemented within the ground movement process, the stand holding process was also used for departing aircraft to guarantee the sequence of arrival of aircraft at the runway and the ground movement process was integrated with the gate allocation process. However, more processes could be integrated with the ground movement process, such as the runway sequencing/take-off scheduling. The ASQPPTW algorithm that has been presented in Chapter 8 provides an excellent tool for fast solution of the ground movement process, making it possible in future work to construct integrated models that were previously thought to be impractical due to long execution times.

Integration with electric aircraft taxi. Another novel technology that has been investigated by the INNOVATE project has been electric taxiing. Since an electric motor is sufficient for pushing back an aircraft and only consumes electricity when the aircraft is moving, the parameters of the ground movement problem of aircraft are affected significantly. Stand holding is not as necessary as it is for aircraft that taxi using their engines and the duration of the pushback process might be reduced without the need to start up the engines at the stand. A model that will take into consideration the effects of an electric motor will be very useful for identifying the benefits of this technology.

References

- Anagnostakis, I. and Clarke, J.P., 2003, January. Runway operations planning: a two-stage solution methodology. In *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on* (pp. 12-pp). IEEE.
- Anderson, K., Carr, F., Feron, E. and Hall, W., 2000. Analysis and modeling of ground operations at hub airports. In *Proceedings of the 3rd USA/Europe Air Traffic Management R&D seminar, Napoli, Italy*.
- Anderson, R. and Milutinović, D., 2013. An approach to optimization of airport taxiway scheduling and traversal under uncertainty. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of aerospace engineering*, 227(2), pp.273-284.
- Apice, C., De Nicola, C., Manzo, R. and Moccia, V., 2014. Optimal scheduling for aircraft departures. *Journal of Ambient Intelligence and Humanized Computing*, 5(6), pp.799-807.
- Atkin, J.A., Burke, E.K., Greenwood, J.S. and Reeson, D., 2006, June. The effect of the planning horizon and the freezing time on take-off sequencing. In *Proceedings of 2nd International Conference on Research in Air Transportation (ICRAT'06)*.
- Atkin, J.A., Burke, E.K., Greenwood, J.S. and Reeson, D., 2007. Hybrid metaheuristics to aid runway scheduling at London Heathrow airport. *Transportation Science*, 41(1), pp.90-106.
- Atkin, J.A., Burke, E.K., Greenwood, J.S. and Reeson, D., 2008. A metaheuristic approach to aircraft departure scheduling at London Heathrow airport. *Computer-aided Systems in Public Transport*, pp.235-252.
- Atkin, J.A., Burke, E.K., Greenwood, J.S. and Reeson, D., 2009. An examination of take-off scheduling constraints at London Heathrow airport. *Public Transport*, 1(3), pp.169-187.
- Atkin, J.A., Burke, E.K. and Ravizza, S., 2010, June. The airport ground movement problem: Past and current research and future directions. In *Proceedings of the 4th International Conference on Research in Air Transportation (ICRAT), Budapest, Hungary* (pp. 131-138).
- Atkin, J.A., Burke, E.K. and Ravizza, S., 2011. A statistical approach for taxi time estimation at London Heathrow Airport. In: *Proceedings of the 10th Workshop on Models and Algorithms for Planning and Scheduling Problems (MAPSP), Nymburk, Czech Republic*, pp. 61-63.

- Atkins, S., Brinton, C. and Jung, Y., 2008, September. Implication of variability in airport surface operations on 4-D Trajectory Planning. In *Proceedings of the American Institute of Aeronautics and Astronautics (AIAA) Aviation Technology, Integration, and Operations (ATIO) Conference*.
- Babić, O., Teodorović, D. and Tošić, V., 1984. Aircraft stand assignment to minimize walking. *Journal of Transportation Engineering*, 110(1), pp.55-66.
- Baik, H., Sherali, H. and Trani, A., 2002. Time-dependent network assignment strategy for taxiway routing at airports. *Transportation Research Record: Journal of the Transportation Research Board*, (1788), pp.70-75.
- Baik, H. and Trani, A.A., 2008. Framework of a time-based simulation model for the analysis of airfield operations. *Journal of Transportation Engineering*, 134(10), pp.397-413.
- Balakrishna, P., Ganesan, R., Sherry, L. and Levy, B.S., 2008, October. Estimating taxi-out times with a reinforcement learning algorithm. In *Digital Avionics Systems Conference, 2008. DASC 2008. IEEE/AIAA 27th* (pp. 3-D). IEEE.
- Balakrishna, P., Ganesan, R. and Sherry, L., 2009, June. Application of reinforcement learning algorithms for predicting taxi-out times. In *Proceedings of the 8th ATM R&D Seminars, Napa, USA*.
- Balakrishna, P., Ganesan, R. and Sherry, L., 2010. Accuracy of reinforcement learning algorithms for predicting aircraft taxi-out times: A case-study of Tampa Bay departures. *Transportation Research Part C: Emerging Technologies*, 18(6), pp.950-962.
- Balakrishnan, H. and Jung, Y., 2007, August. A framework for coordinated surface operations planning at Dallas-Fort Worth International Airport. In *AIAA Guidance, Navigation, and Control Conference* (Vol. 3, pp. 2382-2400).
- Balakrishnan, H. and Chandran, B.G., 2010. Algorithms for scheduling runway operations under constrained position shifting. *Operations Research*, 58(6), pp.1650-1665.
- Beasley, J.E., Krishnamoorthy, M., Sharaiha, Y.M. and Abramson, D., 2004. Displacement problem and dynamically scheduling aircraft landings. *Journal of the operational research society*, 55(1), pp.54-64.
- Benlic, U., Brownlee, A.E. and Burke, E.K., 2016. Heuristic search for the coupled runway sequencing and taxiway routing problem. *Transportation Research Part C: Emerging Technologies*, 71, pp.333-355.
- Bennell, J.A., Mesgarpour, M. and Potts, C.N., 2011. Airport runway scheduling. *4OR: A Quarterly Journal of Operations Research*, 9(2), pp.115-138.

- Bhadra, D., Knorr, D.A. and Levy, B., 2011. Benefits of virtual queuing at congested airports using ASDE-X: A case study of JFK airport. *Air Traffic Control Quarterly*, 20(3), p.225.
- Bianco, L., Dell'Olmo, P. and Giordani, S., 1999. Minimizing total completion time subject to release dates and sequence-dependent processing times. *Annals of Operations Research*, 86, pp.393-415.
- Bihr, R.A., 1990. A conceptual solution to the aircraft gate assignment problem using 0, 1 linear programming. *Computers & Industrial Engineering*, 19(1-4), pp.280-284.
- Bolat, A., 2000. Procedures for providing robust gate assignments for arriving aircrafts. *European Journal of Operational Research*, 120(1), pp.63-80.
- Brinton, C., Krozel, J., Capozzi, B. and Atkins, S., 2002, August. Improved taxi prediction algorithms for the surface management system. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference* (pp. 5-8), Monterey, CA, USA.
- Brownlee, A.E., Atkin, J.A., Woodward, J.R., Benlic, U. and Burke, E.K., 2014. Airport ground movement: real world data sets and approaches to handling uncertainty. In *Proceedings of the 10th International Conference on Practice and Theory of Automated Timetabling (PATAT 2014)* (pp. 462-464).
- Burgain, P., Pinon, O.J., Feron, E., Clarke, J.P. and Mavris, D.N., 2012. Optimizing pushback decisions to valuate airport surface surveillance information. *IEEE Transactions on Intelligent Transportation Systems*, 13(1), pp.180-192.
- Carr, F., Evans, A., Clarke, J.P. and Feron, E., 2002, May. Modeling and control of airport queueing dynamics under severe flow restrictions. In *American Control Conference, 2002. Proceedings of the 2002* (Vol. 2, pp. 1314-1319). IEEE.
- Chandran, B. and Balakrishnan, H., 2007, July. A dynamic programming algorithm for robust runway scheduling. In *American Control Conference, 2007. ACC'07* (pp. 1161-1166). IEEE.
- Chen, J. and Stewart, P., 2011a, July. Planning aircraft taxiing trajectories via a multi-objective immune optimisation. In *Natural Computation (ICNC), 2011 Seventh International Conference on* (Vol. 4, pp. 2235-2240). IEEE.
- Chen, J., Ravizza, S., Atkin, J.A. and Stewart, P., 2011b. On the utilisation of fuzzy rule-based systems for taxi time estimations at airports. In *OASICS-OpenAccess Series in Informatics* (Vol. 20). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.
- Chen, J., Weiszer, M. and Stewart, P., 2015, September. Optimal speed profile generation for airport ground movement with consideration of emissions. In *Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on* (pp. 1797-1802). IEEE.

- Chen, J., Weiszer, M., Stewart, P. and Shabani, M., 2016a. Toward a More Realistic, Cost-Effective, and Greener Ground Movement Through Active Routing—Part I: Optimal Speed Profile Generation. *IEEE Transactions on Intelligent Transportation Systems*, 17(5), pp.1196-1209.
- Chen, J., Weiszer, M., Locatelli, G., Ravizza, S., Atkin, J.A., Stewart, P. and Burke, E.K., 2016b. Toward a more realistic, cost-effective, and greener ground movement through active routing: A multiobjective shortest path approach. *IEEE Transactions on Intelligent Transportation Systems*, 17(12), pp.3524-3540.
- Cheng, Y., 1997. A knowledge-based airport gate assignment system integrated with mathematical programming. *Computers & Industrial Engineering*, 32(4), pp.837-852.
- Cheng, Y., 1998. Solving push-out conflicts in apron taxiways of airports by a network-based simulation. *Computers & industrial engineering*, 34(2), pp.351-369.
- Cheng, V.H. and Foyle, D.C., 2002, August. Automation tools for enhancing ground-operation situation awareness and flow efficiency. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference* (Vol. 4856).
- Cheng, V.H., 2003, August. Airport surface operation collaborative automation concept. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference* (pp. 11-14).
- Cheng, V.H., 2007, April. Research progress on an automation concept for surface operation with time-based trajectories. In *Integrated Communications, Navigation and Surveillance Conference, 2007. ICNS'07* (pp. 1-13). IEEE.
- Clare, G., Richards, A. and Sharma, S., 2009a. Receding horizon, iterative optimization of taxiway routing and runway scheduling. In *Proceedings of the AIAA Guidance Navigation, and Control Conference, Chicago, USA*.
- Clare, G. and Richards, A., 2009b. Airport ground operations optimizer. In *8th Innovative Research Workshop & Exhibition Proceedings, Eurocontrol*.
- Clare, G. and Richards, A.G., 2011. Optimization of taxiway routing and runway scheduling. *IEEE Transactions on Intelligent Transportation Systems*, 12(4), pp.1000-1013.
- Clelow, R.R., Simaiakis, I. and Balakrishnan, H., 2010. Impact of arrivals on departure taxi operations at airports. In *AIAA Guidance, Navigation, and Control Conference. Toronto, Ontario Canada*.
- Confessore, G., Liotta, G. and Grieco, R., 2005, December. A simulation-based architecture for supporting strategic and tactical decisions in the apron of Rome-Fiumicino Airport.

In *Proceedings of the 37th conference on Winter simulation* (pp. 1596-1605). Winter Simulation Conference.

De Maere, G. and Atkin, J.A., 2015. Pruning rules for optimal runway sequencing with airline preferences. *Lecture Notes in Management Science*, 7, p.77.

Deau, R., Gotteland, J.B. and Durand, N., 2008, June. Runways sequences and ground traffic optimisation. In *ICRAT 2008, International Conference on Research in Air Transportation*.

Deau, R., Gotteland, J.B. and Durand, N., 2009, June. Airport surface management and runways scheduling. In *ATM 2009, 8th USA/Europe Air Traffic Management Research and Development Seminar*.

Diepen, G., Van Den Akker, J.M. and Hoogeveen, J.A., 2009. Integrated gate and bus assignment at Amsterdam Airport Schiphol. In *Robust and Online Large-Scale Optimization* (pp. 338-353). Springer Berlin Heidelberg.

Ding, H., Lim, A., Rodrigues, B. and Zhu, Y., 2004a, January. Aircraft and gate scheduling optimization at airports. In *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on* (pp. 8-pp). IEEE.

Ding, H., Lim, A., Rodrigues, B. and Zhu, Y., 2004b. New heuristics for over-constrained flight to gate assignments. *Journal of the Operational Research Society*, 55(7), pp.760-768.

Ding, H., Lim, A., Rodrigues, B. and Zhu, Y., 2005. The over-constrained airport gate assignment problem. *Computers & Operations Research*, 32(7), pp.1867-1880.

Dorndorf, U., Drexl, A., Nikulin, Y. and Pesch, E., 2007. Flight gate scheduling: State-of-the-art and recent developments. *Omega*, 35(3), pp.326-334.

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

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

Drexl, A. and Nikulin, Y., 2008. Multicriteria airport gate assignment and Pareto simulated annealing. *IIE Transactions*, 40(4), pp.385-397.

Durand, N. and Alliot, J.M., 1998, July. Genetic crossover operator for partially separable functions. In *GP 1998, 3rd annual conference on Genetic Programming*.

Ernst, A.T., Krishnamoorthy, M. and Storer, R.H., 1999. Heuristic and exact algorithms for scheduling aircraft landings. *Networks*, 34(3), pp.229-241.

Evertse, C. and Visser, H.G., 2017. Real-time airport surface movement planning: Minimizing aircraft emissions. *Transportation Research Part C: Emerging Technologies*, 79, pp.224-241.

García, J., Berlanga, A., Molina, J.M. and Casar, J.R., 2005. Optimization of airport ground operations integrating genetic and dynamic flow management algorithms. *AI Communications*, 18(2), pp.143-164.

Genç, H.M., Erol, O.K., Eksin, İ., Berber, M.F. and Güleriyüz, B.O., 2012. A stochastic neighborhood search approach for airport gate assignment problem. *Expert Systems with Applications*, 39(1), pp.316-327.

Gotteland, J.B., Durand, N., Alliot, J.M. and Page, E., 2001, December. Aircraft ground traffic optimization. In *ATM 2001, 4th USA/Europe Air Traffic Management Research and Development Seminar*.

Gotteland, J.B., Durand, N. and Alliot, J.M., 2003a, June. Handling CFMU slots in busy airports. In *ATM 2003, 5th USA/Europe Air Traffic Management Research and Development Seminar*.

Gotteland, J.B. and Durand, N., 2003b, December. Genetic algorithms applied to airport ground traffic optimization. In *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on* (Vol. 1, pp. 544-551). IEEE.

Guclu, O.E. and Cetek, C., 2013. Optimization of Aircraft Taxi Movements and Gate Allocation using Hybrid Dynamic Gate Assignment. In *Proceedings of the Multidisciplinary Academic Conference*, p1

Guclu, O.E. and Cetek, C., 2017. Analysis of aircraft ground traffic flow and gate utilisation using a hybrid dynamic gate and taxiway assignment algorithm. *The Aeronautical Journal*, 121(1240), pp.721-745.

Gupta, P., Subramanian, H. and Pant, R.S., 2010a. Generation of optimized routes and schedules for surface movement of aircraft on taxiways. In *AIAA ATIO/ISSMO Conference, Fort Worth, TX, USA*.

Gupta, G., Malik, W. and Jung, Y., 2010b, August. Incorporating active runway crossings in airport departure scheduling. In *AIAA Guidance, Navigation and Control Conference* (pp. 2-5).

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

- Hayashi, M., Hoang, T., Jung, Y.C., Malik, W., Lee, H. and Dulchinos, V.L., 2015. Evaluation of Pushback Decision-Support Tool Concept for Charlotte Douglas International Airport Ramp Operations. *Eleventh USA/Europe Air Traffic Management Research and Development Seminar (ATM2015)*
- Herrero, J.G., Berlanga, A., Molina, J.M. and Casar, J.R., 2005. Methods for operations planning in airport decision support systems. *Applied Intelligence*, 22(3), pp.183-206.
- Hu, X.B. and Di Paolo, E., 2007, September. An efficient genetic algorithm with uniform crossover for the multi-objective airport gate assignment problem. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on* (pp. 55-62). IEEE.
- Idris, H., Clarke, J.P., Bhuva, R. and Kang, L., 2002. Queuing model for taxi-out time estimation. *Air Traffic Control Quarterly*, 10(1), pp.1-22.
- Jordan, R., Ishutkina, M.A. and Reynolds, T.G., 2010, October. A statistical learning approach to the modeling of aircraft taxi time. In *Digital Avionics Systems Conference (DASC), 2010 IEEE/AIAA 29th* (pp. 1-B). IEEE.
- Jung, Y., Hoang, T., Montoya, J., Gupta, G., Malik, W., Tobias, L. and Wang, H., 2011, June. Performance evaluation of a surface traffic management tool for Dallas/Fort Worth International Airport. In *Ninth USA/Europe Air Traffic Management Research and Development Seminar* (pp. 1-10).
- Karapetyan, D., Atkin, J.A., Parkes, A.J. and Castro-Gutierrez, J., 2017. Lessons from building an automated pre-departure sequencer for airports. *Annals of Operations Research*, 252(2), pp.435-453.
- Keith, G., Richards, A. and Sharma, S., 2008, August. Optimization of taxiway routing and runway scheduling. In *Proceedings of the AIAA Guidance, Navigation and Control Conference, Honolulu, Hawaii, USA*.
- Kim, S.H., Feron, E. and Clarke, J.P., 2009, August. Assigning gates by resolving physical conflicts. In *AIAA Guidance, Navigation, and Control Conference* (p. 5648).
- Kim, S.H., Feron, E. and Clarke, J.P., 2010, August. Airport gate assignment that minimizes passenger flow in terminals and aircraft congestion on ramps. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference, Toronto, Canada* (Vol. 2, pp. 1226-1238).
- Kim, S.H. and Feron, E., 2014. Impact of gate assignment on departure metering. *IEEE Transactions on Intelligent Transportation Systems*, 15(2), pp.699-709.

- Kistler, M.S. and Gupta, G., 2009, September. Relationship between airport efficiency and surface traffic. In *Proceedings of the 9th AIAA Aviation Technology, Integration, and Operations Conference, Hilton Head, SC, USA*.
- Kjenstad, D., Mannino, C., Schittekat, P. and Smedsrud, M., 2013a, April. Integrated surface and departure management at airports by optimization. In *Modeling, Simulation and Applied Optimization (ICMSAO), 2013 5th International Conference on* (pp. 1-5). IEEE.
- Kjenstad, D., Mannino, C., Nordlander, T.E., Schittekat, P. and Smedsrud, M., 2013b. Optimizing AMAN-SMAN-DMAN at Hamburg and Arlanda airport. *Proceedings of the SID, Stockholm*.
- Kumar, P. and Bierlaire, M., 2011. Multi-objective airport gate assignment problem. In *Swiss Transport Research Conference, Monte Verita, Switzerland*.
- Kumar, P. and Bierlaire, M., 2014. Multi-objective airport gate assignment problem in planning and operations. *Journal of advanced transportation*, 48(7), pp.902-926.
- Lee, H. and Balakrishnan, H., 2012, October. A comparison of two optimization approaches for airport taxiway and runway scheduling. In *Digital Avionics Systems Conference (DASC), 2012 IEEE/AIAA 31st* (pp. 4E1-1). IEEE.
- Leese, R. A., Craig, A., Ketzscer, R., Noble, S. D., Parrott, K., Preater, J., Wilson, R. E., & Wood, D. A. (2001). The sequencing of aircraft departures. In *40th European study group with industry, Keele*.
- Lesire, C., 2009. Automatic planning of ground traffic. In *AIAA Aerospace Sciences Meeting (ASM'09). Orlando, USA*.
- Lesire, C., 2010, August. An Iterative A* Algorithm for Planning of Airport Ground Movements. In *ECAI* (pp. 413-418).
- Li, L., Clarke, J.P., Chien, H.H.C. and Melconian, T., 2009, October. A probabilistic decision-making model for runway configuration planning under stochastic wind conditions. In *Digital Avionics Systems Conference, 2009. DASC'09. IEEE/AIAA 28th*(pp. 3-A). IEEE.
- Lim, A., Rodrigues, B. and Zhu, Y., 2005. Airport gate scheduling with time windows. *Artificial Intelligence Review*, 24(1), pp.5-31.
- Mangoubi, R.S. and Mathaisel, D.F., 1985. Optimizing gate assignments at airport terminals. *Transportation Science*, 19(2), pp.173-188.
- Marín, A.G., 2006. Airport management: taxi planning. *Annals of Operations Research*, 143(1), pp.191-202.

- Marín, A.G. and Codina, E., 2008. Network design: taxi planning. *Annals of Operations Research*, 157(1), pp.135-151.
- Marín, A.G., 2013. Airport taxi planning: Lagrangian decomposition. *Journal of Advanced Transportation*, 47(4), pp.461-474.
- Marinescu, A., Sharples, S., Campbell Ritchie, A., Sanchez Lopez, T., McDowell, M. and Morvan, H., 2017. Physiological parameter response to variation of mental workload. *Human Factors*.
- Mori, R., 2010. Modeling of aircraft surface traffic flow at congested airport using cellular automata. In *Proceedings of the 4th International Conference on Research in Air Transportation (ICRAT), Budapest, Hungary* (pp. 139-145).
- NASA 2017. X-Plane Communication Toolbox (XPC), accessed 29 September 2017 <<https://software.nasa.gov/software/ARC-17185-1>>
- Neuman, U.M. and Atkin, J.A., 2013, September. Airport gate assignment considering ground movement. In *International Conference on Computational Logistics* (pp. 184-198). Springer, Berlin, Heidelberg.
- Neuman, U.M., 2015. *Modelling and analysis of real world airport gate allocation problem* (Doctoral dissertation, University of Nottingham).
- Nikoleris, T., Gupta, G. and Kistler, M., 2011. Detailed estimation of fuel consumption and emissions during aircraft taxi operations at Dallas/Fort Worth International Airport. *Transportation Research Part D: Transport and Environment*, 16(4), pp.302-308.
- Nikulin, Y. and Drexler, A., 2010. Theoretical aspects of multicriteria flight gate scheduling: deterministic and fuzzy models. *Journal of Scheduling*, 13(3), pp.261-280.
- Paternostro, S., Moore, T., Hill, C., Atkin, J. and Morvan, H.P., 2016. Evaluation of ARAIM Performance on Predicted Aircraft Trajectories. *Proceedings of ION/IEEE PLANS*.
- Pesic, B., Durand, N. and Alliot, J.M., 2001, July. Aircraft ground traffic optimisation using a genetic algorithm. In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation* (pp. 1397-1404). Morgan Kaufmann Publishers Inc.
- Pina, P. and De Pablo, J.M., 2005. Benefits obtained from the estimation and distribution of realistic taxi times. In *ATM R&D Seminar*.
- Psaraftis, H.N., 1980. A dynamic programming approach for sequencing groups of identical jobs. *Operations Research*, 28(6), pp.1347-1359.

- Rathinam, S., Montoya, J. and Jung, Y., 2008, September. An optimization model for reducing aircraft taxi times at the Dallas Fort Worth International Airport. In *26th International Congress of the Aeronautical Sciences (ICAS)* (pp. 14-19).
- Rappaport, D.B., Yu, P., Griffin, K. and Daviau, C., 2009, September. Quantitative analysis of uncertainty in airport surface operations. In *Proceedings of the AIAA aviation technology, integration, and operations conference*.
- Ravizza, S. and Atkin, J.A., 2011. Exploration of the ordering for a sequential airport ground movement algorithm. Tech. Rep. 1543, University of Nottingham, UK.
- Ravizza, S., Atkin, J.A., Maathuis, M.H. and Burke, E.K., 2013a. A combined statistical approach and ground movement model for improving taxi time estimations at airports. *Journal of the Operational Research Society*, 64(9), pp.1347-1360.
- Ravizza, S., Chen, J., Atkin, J.A., Burke, E.K. and Stewart, P., 2013b. The trade-off between taxi time and fuel consumption in airport ground movement. *Public Transport*, 5(1-2), pp.25-40.
- Ravizza, S., 2013c. *Enhancing decision support systems for airport ground movement* (Doctoral dissertation, University of Nottingham).
- Ravizza, S., Atkin, J.A. and Burke, E.K., 2014. A more realistic approach for airport ground movement optimisation with stand holding. *Journal of Scheduling*, 17(5), pp.507-520.
- Roling, P.C. and Visser, H.G., 2008. Optimal airport surface traffic planning using mixed-integer linear programming. *International Journal of Aerospace Engineering*, 2008(1), p.1.
- Roling, P., 2009, September. Airport surface traffic planning optimization: a case study of Amsterdam Airport Schiphol. In *Proceedings of the 9th AIAA Aviation Technology, Integration, and Operations Conference (ATIO)*.
- Roling, P.C., 2011. TPMagic, a universal airport surface traffic planning analysis and optimization tool. In *11th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference, including the AIA, Virginia Beach, VA, USA, 20-22 September 2011, AIAA 2011-7006*. American Institute of Aeronautics and Astronautics (AIAA).
- SESAR. 2015. The Roadmap for Delivering High Performing Aviation for Europe. *European ATM Master Plan*. [Online]. Available: <https://ec.europa.eu/transport/sites/transport/files/modes/air/sesar/doc/eu-atm-master-plan-2015.pdf>
- Schaijk, O.R. and Visser, H.G., 2017. Robust flight-to-gate assignment using flight presence probabilities. *Transportation Planning and Technology*, 40(8), pp.928-945.

- Simaiakis, I. and Balakrishnan, H., 2010. Impact of congestion on taxi times, fuel burn, and emissions at major airports. *Transportation Research Record: Journal of the Transportation Research Board*, (2184), pp.22-30.
- Smeltink, J.W. and Soomer, M.J., de Waal, P.R. and van der Mei, R.D., 2004. An Optimisation Model for Airport Taxi Scheduling. *Elsevier Science*.
- Soomer, M.J. and Koole, G.M., 2008. Fairness in the aircraft landing problem. *Proceedings of the Anna Valicek competition*.
- Srivastava, A., 2011, October. Improving departure taxi time predictions using ASDE-X surveillance data. In *Digital Avionics Systems Conference (DASC), 2011 IEEE/AIAA 30th* (pp. 2B5-1). IEEE.
- Stenzel, B., 2008. Online disjoint vehicle routing with application to AGV routing. (Doctoral dissertation, Technical University of Berlin, Germany).
- Tang, C.H., Yan, S. and Hou, Y.Z., 2010. A gate reassignment framework for real time flight delays. *4OR: A Quarterly Journal of Operations Research*, 8(3), pp.299-318.
- Tavakkoli-Moghaddam, R., Yaghoubi-Panah, M. and Radmehr, F., 2012. Scheduling the sequence of aircraft landings for a single runway using a fuzzy programming approach. *Journal of Air Transport Management*, 25, pp.15-18.
- Tu, Y., Ball, M.O. and Jank, W.S., 2008. Estimating flight departure delay distributions—a statistical approach with long-term trend and short-term pattern. *Journal of the American Statistical Association*, 103(481), pp.112-125.
- Van Velthuizen, M.A.R., 1997. Taxi planning optimization using mixed integer programming. *Delft University of Technology, Faculty of Aerospace Engineering, memorandum m-799*.
- Weiszer, M., Chen, J., Ravizza, S., Atkin, J. and Stewart, P., 2014, July. A heuristic approach to greener airport ground movement. In *Evolutionary Computation (CEC), 2014 IEEE Congress on* (pp. 3280-3286). IEEE.
- Weiszer, M., Chen, J. and Stewart, P., 2015a. A real-time active routing approach via a database for airport surface movement. *Transportation Research Part C: Emerging Technologies*, 58, pp.127-145.
- Weiszer, M., Chen, J. and Locatelli, G., 2015b. An integrated optimisation approach to airport ground operations to foster sustainability in the aviation sector. *Applied Energy*, 157, pp.567-582.

- Weiszer, M., Chen, J. and Stewart, P., 2015c, September. Preference-based evolutionary algorithm for airport runway scheduling and ground movement optimisation. In *Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on* (pp. 2078-2083). IEEE.
- Wood, E., Herndon, S., Miake-Lye, R., Nelson, D., 2008. Aircraft and Airport-Related Hazardous Air Pollutants: Research Needs and Analysis. *Transportation Research Board 87th Annual Meeting, Washington, DC*.
- Xu, J. and Bailey, G., 2001, January. The airport gate assignment problem: mathematical model and a tabu search algorithm. In *System Sciences, 2001. Proceedings of the 34th Annual Hawaii International Conference on* (pp. 10-pp). IEEE.
- Xu, L., Wang, F. and Xu, Z., 2011, January. A robust approach for the airport gate assignment. In *International Forum on Shipping, Ports and Airports (IFSPA)* (p. 15).
- Yan, S. and Huo, C.M., 2001. Optimization of multiple objective gate assignments. *Transportation Research Part A: Policy and Practice*, 35(5), pp.413-432.
- Yin, K., Tian, C., Wang, B. and Quadrifoglio, L., 2012. Analysis of Taxiway Aircraft Traffic at George Bush Intercontinental Airport, Houston, Texas. *Transportation Research Record: Journal of the Transportation Research Board*, (2266), pp.85-94.