CHALMERS | UNIVERSITY OF GOTHENBURG

JEPPESEN
A BOEING COMPANY

Development and Evaluation of Novel Algorithms for Enhanced Aircraft Routing on Ground

Master of Science Thesis in Computer Science

ACHILLEAS F. KATSAROS

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Göteborg, Sweden, March 2012

Development and Evaluation of Novel Algorithms for Enhanced Aircraft Routing on Ground

Achilleas F. Katsaros

Examiner: Peter Ljunglöf

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Cover: The logo of the company Jeppesen, where the present work was done, combined with a picture of aircraft routing on the ground.

Department of Computer Science and Engineering
Göteborg, Sweden March 2012

Abstract

The process of Aircraft Routing on Ground corresponds to the surface movement of aircraft on an airport's taxiway network, from a runway exit to a parking stand (arrivals) and from a parking stand to a runway entrance (departures). Given such a pair of terminal points and depending on the taxiway network structure, there can be one or more alternative routes that the aircraft can follow to its destination. From a functional perspective, this surface movement is the link between the airborne movement and the turnaround process for each aircraft. As the size, the operational complexity and the traffic of an airport increases, the sequence of "landing-taxiing-turnaround-taxiing-taking-off" can become tight and the satisfaction of temporal constraints can become critical in terms of cost-effectiveness, both for the airlines and the airport operators.

From the discussion above we can deduce that the choice of an optimal route on the taxiway network for each aircraft depends also on the dimension of time, since the traffic load on each taxiway is a dynamic parameter. The problem that this thesis is concerned with can therefore be called "4D Taxi Routing on Ground". The work for this thesis was done at Jeppesen GmbH in Neu-Isenburg, Germany.

The main focus of this work is the definition, analysis and modeling of the problem of "4D Taxi Routing on Ground". The entities that constitute the essence of the problem are described: the Taxiway Network as a directed, bimodal graph with weights that are functions of time; the Aircraft as moving agents with a finite set of states; and the Airport Operations as the environment. A set of objectives that determine the quality of a routing solution is also defined: minimization of taxi time, hold time and speed changes for each aircraft. Based on these definitions, a mathematical model is built.

In order to test the soundness of the defined model and its applicability to the problem of "4D Taxi Routing on Ground", an optimization algorithm is designed and implemented, based on a combination of Dijkstra's SPP algorithm and a Linear Programming formulation. The evaluation of the algorithm is realized using simulations of a 2-day actual flight schedule and the results support the assumption of correctness for most of the model design choices, while they also show the inefficiencies of a static SPP algorithm when applied to a dynamic or "4D" routing problem, thus indicating directions for further research or potential areas of improvement.

Acknowledgements

The work for this thesis was done at the Advanced Research department of Jeppesen GmbH, in Neu-Isenburg, Germany from May to October 2011. My thesis supervisor and advisor at Jeppesen, as well as the person who formulated and proposed the subject of this thesis, is Dr. Nima Barraci, to whom I express my acknowledgement and gratitude. I learned a lot during those six months thanks to his broad knowledge of Aviation and Optimization among others and his well-structured way of working. His focus on the scientific soundness, his way of communicating ideas and his enthusiasm about the outcome of this thesis guided my efforts. I would also like to express my acknowledgement and gratitude to my university supervisor and examiner, Dr. Peter Ljunglöf, who reviewed my thesis document and came up with a number of pointed remarks that improved the academic quality of this document. His contribution to the sound formulation of the mathematical model and his proposal on the restructuring of the graph is especially highly appreciated.

Regarding Jeppesen, I would also like to thank the Advanced Research team for making me feel welcome and providing a great working environment, as well as for their comments and feedback during the internal presentations of the progress of my thesis. Alicia Grech from the TAAM team supplied me with the summer 2010 flight schedule for Stockholm-Arlanda airport (the case study for this work) and with scientific papers about TAAM. Jan-Olof Roos, ATC at Arlanda airport, provided me with information on the utilization of runways, terminals and parking stands. Pilot Paul Gaede supported me with an interview which enhanced or corrected my assumptions on a variety of operational features. The information I obtained from these people was crucial for the model design and implementation and for the present document. I owe a great thank to them.

Regarding the university, I want to thank my thesis opponent, Malin Ahlberg, for her thorough review, her cooperation and comments during my presentation. As this thesis concludes these two years of my master studies, a decision that definitely changed my life in many aspects, I feel the need to acknowledge the academic community of Chalmers and GU, students I worked with and teachers that inspired me, especially Björn von Sydow and Jan Jonsson. I already miss being there.

Last but definitely not least, my warmest thanks to my closest people, my family - father Fragkiskos, mother Konstantina, sister Christina and brother Vassilis – as well as my girlfriend, Elisabet Tsirkinidou, for their support and encouragement during many ups-and-downs of the recent years. The best is yet to come.

Table of Contents

List of Acronyms

A-CDM	Airport Collaborative Decision Making
AGC	Aircraft Ground Controller
AIBT	Actual In-Block Time
AM	Arrival Management
AMDB	Airport Mapping Database
AOBT	Actual Off-Block Time
ATC	Air Traffic Control
ATCO	Air Traffic Control Officer
CAED	Coordinated Airport through Extreme Decoupling
CFMU	Central Flow Management Unit
CTOT	Calculated Take-Off Time
DM	Departure Management
GA	Genetic Algorithm
GM	Gate Management
IATA	International Air Transport Association
ICAO	International Civil Aviation Organization
LP	Linear Programming
PLC	Product Life Cycle
SESAR	Single European Sky ATM Research
SMAN	Surface Manager
SOA	Service Oriented Approach
SPP	Shortest Path Problem
STN	Simple Temporal Networks
TAAM	Total Airspace and Airport Modeler
TIS	TITAN Information Service
TITAN	Turnaround Integration in Trajectory and Network
TPO	Taxi Planning Optimization
VTTC	Variable Taxi Time Calculation

Chapter 1 **Introduction**

The cycle of an aircraft's operation consists of a number of consecutive phases. Starting from a parking stand on an airport's apron, an aircraft will be routed on the taxiway, enter the runway from a predefined runway exit (in this case it is a "runway entrance"), speed up and take off, climb to a specific altitude and follow the instructed airways and altitudes flying to its destination airport. When it approaches the destination, the aircraft will be commanded to descend and land to the airport's runway (or a specified one in case of airports with more than one runway), use a runway exit to start routing on the taxiway towards the predefined stand on the destination airport where it will be parked. The turnaround process will follow; unloading passengers and baggage or cargo, refueling, cleaning, performing various checks, loading again and finally being pushed back to the taxiway to start routing towards its new destination [6]. According to [7], the phases of flight are the following:

- Standing
- Pushback / Towing
- Taxi
- Take-off
- Initial climb
- En route
- Approach
- Landing

Based on the aircraft's movement, these operations can be grouped into three categories. The first group consists of the operations from taking off to landing, which correspond to the airborne movement, the second group corresponds to the movement on the ground – from the apron to the runway via the taxiway and vice versa (pushback and taxi flight phases) - and the third group is the above mentioned turnaround process or an overnight stop, when the aircraft is not moving but stands parked on the apron. The operation cycle can be extended with the regular maintenance checks, which could be considered as part of the turnaround process but not of the same frequency; for example, refueling takes place between two flights, while maintenance can occur from every 500-800 flight hours to 3-6 months, depending on the type of maintenance checks performed [32].

The reason for this grouping is the high degree of independence among these three operational categories. There is of course a time-dependence because of the sequential ordering; each group of operations follows the other in this cycle. However, an aircraft that has already taken off does not affect the situation on the taxiway anymore, as well as an aircraft routing on the taxiway does not affect the unloading of cargo from another aircraft already parked at the apron.

The high degree of independence allows the Operations Research to focus on a specific category, as the present work is focusing on the taxiway routing, but this does not imply full independence. For example, traffic congestion on the taxiway can affect aircraft approaching the airport, if they are directed to delay their descent and fly a holding pattern around the airport instead. Another example is that an aircraft on the taxiway routing towards a parking area may be forced to hold and wait for another that just finished the turnaround process and is pushed back to

the taxiway, if these two aircraft share the same parking area. This holding is an important factor of taxiway delays and can propagate to other aircraft moving on the taxiway [8].

The purpose of this discussion is to place the operation of taxiway routing into the greater context of the whole operational cycle of an aircraft and point out relations and complications, constraints and objectives, all the participating factors that can serve as a basis for first describing and then defining the model of the problem of 4D Taxi Routing on Ground.

Before proceeding, a clarification on the term "4D" is necessary. This term is used to emphasize that the problem cannot be defined using the 3 spatial dimensions, but the introduction of the dimension of time (commonly referred to as the 4th dimension) is also essential. On the other hand, from the 3 spatial dimensions only 2 are sufficient for describing a ground movement. Therefore, one could say that the term "4D" is redundant. The reasons for naming the problem "4D Taxi Routing on Ground" are the following:

- An exact title like "(2+1)D Taxi Routing on Ground" introduces more ambiguity than solves the problem
- The 3rd dimension (altitude) can be as well included and considered constant

1.1 Airline Operations Research

The Industrial Evolution of the 19th century and the so-called Informational Revolution of the end of the 20th century have changed and keep on changing our world. Scientific evolution, innovative ideas and research lead to technological advances, new products and services. New markets and professions emerge to meet our needs, while others become obsolete and are gradually phased out or drastically reformed in order to survive. The airline industry began in the 1920s and nowadays, 90 years later, it is reaching maturity and will continue until air travel becomes obsolete. According to the author of [20]:

> "*In the 1950s and 1960s the world's air traffic grew on average at around 14-15% each year. In the decade 1970-79 the annual growth was close to 10%. This still meant that air traffic, and the airlines with it, doubled in size every seven years or so. In the following ten years to 1989 growth declined to around 6% annually and in the decade up to 1999 growth was down slightly at 5.2%. In absolute terms, because of the much higher base, a 5% jump in recent years represents a much greater surge in demand than a 10% annual growth thirty years ago*" [20].

However, "*the financial performance of the world's airlines taken as a whole has been very marginal, even in the years when the industry was highly regulated and largely protected from internal competition*" [20]. This is mainly due to the high operating (fuel, crew, maintenance, handling etc) and other costs (airport taxes, depreciation) airlines have to face. Therefore, there is a lot of operations research by the related actors (airlines and airport operators) towards optimizing their processes, which means minimizing costs and/or maximizing revenues. Due to the large economic size of the airline industry, a cost reduction of a seemingly trivial percentage – 1-2% - can be translated into large absolute numbers. For example, Lufthansa announced a total revenue of 27.3 billion € for 2010 and the operating result for the same year was 876 million € [10].

The evolution of computer processing power and capacity, in the greater context of the evolution of Computer Science mainly during the last 20 years of the previous century and until nowadays, has provided the industry operations research (including the aviation industry) with the essential tools, in order to translate the various operations into mathematical models that consist of constraints and objectives, solve them with the aid of computers and use the solutions towards optimizing the way they utilize their resources. Many problems have been well-studied and become prototypes of combinatorial optimization problems. Some of them, related to the airline operations, are the following:

- The Timetabling problem: Given a network of operating airports, flight costs and estimations of passenger demands, produce an optimal (daily-weekly-monthly) time-schedule of flight operations [14]. This can be combined with the next problem.
- The Fleet Assignment problem: Given a time-schedule of flight legs, assign specific aircraft types to each flight leg, in order to match the aircraft capacities with the passenger demands [15].
- The Aircraft Routing problem: Given an assignment of flight legs to aircraft types and timing, maintenance and availability restrictions, create optimal routes for each aircraft of the airline's fleet [16].
- The Crew Scheduling problem, which can be divided into two sub-problems. First, to create working patterns (pairing) and then to assign these to individual crew (rostering), with respect to the rules limiting the hours the crew is allowed to work in a specific period of time [17].

For the above mentioned problems, the decisions are solely taken by the airlines. Other stakeholders, like the airport operators, do not have any direct profit from the way airlines allocate and assign their fleet and crew. On the other hand, there exist other optimization problems, which concern both the airport operator and the airlines. The decisions are the responsibility of the airport operators, but their results have a financial effect on the airlines, so a good solution constitutes a profit for both sides. These problems include:

- The coordination of the Turnaround process, which can be defined as the set of services required from the time the flight arrives at its stand (AIBT – Actual In-Block Time) until the time it leaves it (AOBT – Actual Off-Block Time). As briefly discussed in the previous section, these services include passenger and baggage/cargo loading and unloading, aircraft refueling, cleaning, security checks etc. There are a number of different stakeholders participating in this process and they have to be efficiently coordinated, so that the set of services is finished on-time for the aircraft to depart [6], [19].
- The optimization of Aircraft Routing on the Ground, which is the subject of the present thesis work. An overview of the problem follows in the next section.

1.2 Aircraft Routing on the Ground

The routing of aircraft on the ground or taxi routing process is the generation of a route on the taxiway for each aircraft to follow after landing – from the runway exit to the parking stand – or before taking off – from the parking stand to the runway entrance. The problem of deciding on the

route that the aircraft will follow can be from trivial to highly complicated, depending on specific factors that will be presented in the next paragraphs. Arrival and departure taxi routings are not the only possible surface movements of an aircraft, even though they can be considered as the most common. An aircraft can move to a maintenance hangar, to an overnight parking position or to the de-icing facility as well. In the next chapters there will be further discussion about these cases.

1.2.1 Characteristics and Restrictions

First of all, the problem depends on the airport itself. Each airport has its own taxiway network structure. There can be intersections like the crossroads on city road networks, some wide and long straight taxiways, where the aircraft can develop a higher taxi speed; there can also be 90° turns into smaller taxiways (see Appendix I). Some of the taxiways can be one-way directed, in the sense that both arriving and departing aircraft must traverse them in the same defined direction. Some taxiways can be bidirectional, where arriving aircraft use the one direction and departing aircraft the other. Finally, there are some typical limits on the taxiway routing speed, dictated by safety rules. A typical taxi speed is 15 to 20 knots[1] on a straight taxiway and 7 to 12 knots on 90° turns, while the usual maximum speed is 25 knots and is reached in cases when an aircraft is delayed and hurries to reach the parking position [8].

The structural characteristics and restrictions of an airport's taxiway, as well as the speed limits, where applicable, do not depend on the aircraft routing on the taxiway; they are uniform. However, there are other restrictions that affect only certain types of aircraft. One of them is the maximum wingspan allowed on certain (narrower) taxiways. Another is the maximum weight, which can be met as a restriction on airports that include bridges in their taxiway structure.

All the above characteristics of an airport's taxiway network are structural, therefore more or less static. Apart from these, there are also operational characteristics that impose further restrictions on the choice of a feasible taxiway route and that are changing with time. An airport can have one or more runways with possibly different orientations. The choice of the runway(s) to be used for landings and for takeoffs depends on the weather conditions, especially the wind direction, so it can change during the same day. There are also environmental issues, as well as the need to reduce noise in case of the existence of residential areas close to the airport. Lastly, the runway lighting and the visibility during certain hours of the day or under certain circumstances can affect the choice of the runway(s). Consecutively, this choice also defines which runway exits are used at a certain time, which in turn affects the taxiway routing process [5], [8].

Other constraints are related to the business and legal aspects of an airport's operations. These have an impact on the terminal / gate, thus the parking stand that an aircraft will be routed to. For example, in European airports there are different terminals for passengers coming from Schengen and non-Schengen countries. There can also be different terminals for domestic and international flights. Moreover, within the same terminal, it is usually the case that certain airlines or airline alliances have allocated a specified range of gates, depending on contracts, agreements or business relations between the airline and the airport operator. Some airports are hubs or bases of the same country's national carrier. For example, the airports of Kastrup in Copenhagen, Arlanda in

[1] A knot is a unit of speed equal to one nautical mile per hour. 1 knot ≈ 1.852 km/h [38]

Stockholm and Gardermoen in Oslo are hubs of the Scandinavian Airlines (SAS) [33], while Lufthansa uses the airports of Frankfurt, Düsseldorf and Munich in Germany and Zürich in Switzerland as its hubs [9]. It is obvious that these airlines have their reserved gates at their hub airports.

The description of the taxiway structural and operational constraints places the problem into context. As most of the already mentioned problems (fleet assignment, crew scheduling etc), the problem of aircraft routing on the ground falls into the category of combinatorial optimization: given an environment that imposes certain constraints and a finite set of resources, the problem is to allocate / distribute / use these resources in such a way that guarantees feasibility according to the constraints and optimizes an objective function, which is either a cost function to be minimized or a profit function to be maximized.

The combinatorial optimization problems are considered "difficult" to solve optimally, because of their combinatorial nature, where the number of possible solutions grows rapidly (usually exponentially) with the growing of the problem size. This makes finding the optimal solution by enumeration practically not feasible in most cases that correspond to real-world problems. In order to have all the necessary information to define the problem of 4D Taxi Routing on Ground as a sound mathematical model, a discussion about the objectives of this problem must be made and a number of research questions that will be attempted to be answered in the present work must be set.

1.2.2 Objectives and Research Questions

What measures are to be optimized during the process of routing aircraft on the taxiway and why should an airport operator be concerned about optimizing them?

It has been shown [6] that the airports are becoming the bottlenecks of the air transport network. The quality of service that an airport operator offers to the airlines using it is determined among others by the lack of delays while arriving to and departing from this airport. The turnaround process and the need of coordination of the different stakeholders has already been mentioned and is quite important, but what about the time that an aircraft spends on the ground taxiing to the parking stand or back to the runway?

In low to normal taxiway traffic conditions, the possibility of two aircraft meeting on a taxiway intersection during their routings is approximately 20% and in such cases there are explicit directions from Air Traffic Control (ATC) on which aircraft has the highest priority. Usually the aircraft that does not have the high priority is informed well beforehand, so that a speed reduction is enough and the overhead is just a number of seconds. A speed reduction, if possible, is preferable to a full stop on the taxiway, because the second is more time and fuel consuming [8]. Nevertheless, it is not uncommon at large airports where aircraft land and take off every minute (or every few minutes) that a high load of traffic on the taxiway is observed similar to the traffic jams on the streets of big cities during peak hours. This results to more frequent meetings of aircraft on intersections, lower taxi speed or even aircraft holding on the ground and waiting for their route to clear in order to continue.

From this discussion, it becomes evident that the sooner an aircraft reaches the parking stand or the runway, i.e. exits the taxiway network, the lower the traffic becomes for the rest of the aircraft

routing. Therefore an objective is the minimization of taxi time $\boldsymbol{tt}(\boldsymbol{a_k})$ for each aircraft $\boldsymbol{a_k}$, so that the taxiway network is less combusted and delays are avoided.

Another objective is the minimization of hold time $\boldsymbol{ht}(\boldsymbol{a_k})$ for each aircraft $\boldsymbol{a_k}$. The frequency or possibility of holding on taxiway intersections increases with the taxiway traffic load. Another important factor that causes holdings on the taxiway is when an aircraft is about to park at a gate that is still occupied or cannot enter a parking area because it must wait for another aircraft to be pushed back from the same area [8]. These are results of ineffective gate occupancy coordination or even of a delayed turnaround process.

Holding on the ground with the engines on is very costly in terms of fuel consumption. For example the aircraft engine CF6-80C2 (manufacturer: GE Aviation) consumes 0.206 kg of kerosene per second in idle power [11] and the average price of kerosene (October 2011) is $4.10 per gallon [2] which is approximately $1.32 per kg[2], so every minute of standing idle with one engine on costs about $16.4 only for the fuel. If this number is multiplied with an average number of minutes that each aircraft holds on the ground and with the number of an airline's fleet, the resulting fuel cost for an airline rises substantially.

There are also other costs that can be added to the fuel costs, like crew overtime payment or even an overnight stay at an airport hotel for passengers of connecting flights missed because of a delayed arrival [8]. This is a worst-case scenario and definitely not a common situation. However, it shows how a delay caused by holding on the taxiway can have a knock-on effect with high costs for the airline. Last but not least, there are environmental issues with having aircraft standing on the ground with their engines running.

An aircraft is a massive object, has therefore much bigger inertia than a car for example. It requires spending much energy in order to start routing after being on hold or in order to change speed while taxiing. The fuel costs are one of the main costs airlines face and try to reduce, but there is also the customer – passenger – satisfaction that must be taken into consideration. This is not a numerical metric; however it can be translated into numbers because a satisfied passenger is more likely to prefer the same airline in a future trip. And the assumption that a routing on the taxiway with a steady speed is much more comfortable than feeling the inertia of constant speed changes - accelerations and decelerations – gives one more reason to introduce a third objective which is the minimization of the number of speed changes $\boldsymbol{sc}(\boldsymbol{a_k})$ for each aircraft $\boldsymbol{a_k}$ while taxiing.

In a trivial case where an aircraft lands on an airport and there is no other aircraft routing on the taxiway, the meeting of the objectives set is quite simple. A direction can be given to the aircraft to follow the shortest path from the runway exit to the parking area, with a steady speed. The calculation of the shortest path for a static graph with given weights (the lengths of the different paths) is a well-studied and efficiently solved problem, e.g. the algorithm of Dijkstra [21], [39] and the A* algorithm [40]. The combinatorial nature of the problem emerges when the load of traffic increases. A path that is available at a given time point, might not be available after a few seconds because another aircraft has just entered from the opposite direction. This is where the dimension of time enters the problem: the graph of the airport taxiway network is a time-dependent one. A

[2] 1 gallon ≈ 3.785 lt and the density of kerosene is 0.82 kg/lt, so 1 gallon of kerosene weights approximately 3.104 kg [35]

seemingly shortest path can turn out to be a bad choice if every aircraft follows it. The occurrence of a deadlock, where two aircraft wait for each other to move in order to free a path on the taxiway is also a possibility we cannot rule out.

It becomes clear that the efficient routing of aircraft on the ground can be of great importance for an airport operator. Minimization of the taxi and the hold time is energy and time saving for the airlines, making the airport a preferable one. Delays and passenger complaints are avoided; safety and environmental restrictions are met. The less time the aircraft spend on the taxiway, the more time becomes available for the turnaround process and the sooner they can leave the airport, thus increasing the availability of parking stands and the aircraft capacity in terms of gate services, resulting to higher revenues for the airport operator.

The research questions that will be attempted to be answered in terms of the present thesis work are the following:

1. How can the above described problem be formulated into a mathematical model?
2. Which algorithms (of which algorithmic classes) are more suitable to deal with this problem?
3. How efficiently can the objectives be met? How much can the mean taxi time, hold time etc be decreased?
4. What is the relation between the decrease of the taxi and hold time and the increase in the airport's capacity / throughput?
5. How robust can a solving algorithm be, i.e. how well can it deal with last moment unforeseeable changes?
6. What is the traffic limit for a specific airport, after which the system crashes, i.e. is unable to recover to a normal operation?

1.3 Structure of this Thesis

This thesis is organized in six chapters and is concluded by three appendices. The present chapter is an introduction to the problem of 4D Taxi Routing on Ground; a first discussion on the characteristics and restrictions of the problem and its placement into the context of Airline Operations Research. The chapter concluded with a set of research questions, which determine the focus and direction of this thesis. An overview of the current state-of-the-art, regarding commercial products and research work on this and similar topics, is given in chapter 2.

Chapter 3 presents the concept for a system that captures the essence of 4D Taxi Routing on Ground and supports the development of algorithms with the purpose of optimizing the objectives set. The chapter starts with a detailed discussion and a formal definition of the problem and results to a mathematical model. Chapter 4 describes the realization of the concept and gives an overview on the class architecture and the optimization algorithms. The environment setup for the evaluations and their respective results are presented in chapter 5.

Finally, chapter 6 concludes this thesis with a discussion on how satisfactorily the research questions set above were answered, what improvements can be made and which further directions of study and research on this topic can be followed.

Complementary information can be found in the appendices. Appendix I contains the maps of the taxiway structure of Stockholm-Arlanda airport as of October 2011. Appendix II contains two distance matrices for each pair of “runway exit – parking stand area” terminal points for this airport. The first table starts from the runway exits and the second starts from the parking stand areas. Appendix III contains the data tables of aircraft types, airlines and destination airports which are currently related to this airport and were therefore used as test data in our model implementation.

Chapter 2 The State of the Art

When an aircraft is landing on an airport runway, the direction on which runway exit to use is already communicated to the pilot by the Air Traffic Control Officer (ATCO) who is responsible for the landing traffic management [8]. Once the aircraft is on the ground, the control is handed over to the Aircraft Ground Controller (AGC), who is responsible for the taxi routing process [22]. If the aircraft has not received the taxi clearance, i.e. the permission to start taxiing and the route - the sequence of taxiways - to follow, it must stand still on the runway exit and wait. The taxi route can be fully communicated a priori - the most usual case in small and medium sized airports - or it can be communicated in parts while the aircraft moves on the taxiway. In the second case, which is usual in large airports (e.g. Frankfurt am Main International airport), the ground controller commands the aircraft to taxi up to a certain point via a specified sequence of taxiways and wait for further directions [8].

The procedure is similar when an aircraft is departing. In order to be pushed-back and start routing on the taxiway towards the runway for taking-off, an aircraft must have first obtained the permission to do so by the ATCO who is responsible for the take-off traffic management. Then the AGC assumes control. The purpose of this discussion is to point out the interconnection and the tight timing sequence of aircraft flow from the responsibility of one section of ATC to another. Moreover, the sequences of landing-taxiing-parking or pushing-back-taxiing-taking-off are not the only links in the operation chain of an airport and ATC is not the only stakeholder. Different airport partners are involved in different operations and their objectives and interests are sometimes conflicting with each other. The decisions of the management area for one operation depend on the outcome of the preceding operations and accordingly affect the following ones.

In this complex environment, the taxiway routing process cannot be analyzed and presented without a reference to its surrounding operations. The current state-of-the-art in coordinating the airport operations is called Airport Collaborative Decision Making (A-CDM) and its application to the turnaround process is the subject of section 2.1. In section 2.2 we present products and tools related to the taxiway routing process and section 2.3 closes the chapter with a discussion about classes of algorithms that are developed for solving different types of network routing problems.

2.1 A-CDM and the Turnaround Process

"*The concept of Airport CDM endeavors to bring all the main airport partners (ATC, Aircraft Operator, Airlines, CFMU and Ground Handlers) together and share operational data in a transparent way. Information sharing is essential for achieving common situational awareness. Enhancing decision making processes will lead to achieving maximized operational efficiency and best use of the available airport infrastructure and resource management*" [12].

The above statement defines the term of Airport CDM and the rationale for its introduction. Airport CDM is a concept, not an implementation nor a protocol. It appeared as the answer to the inefficiencies of standalone information systems at the different management areas within the airport operations. Each management unit tries to optimize the utilization of their resources, in order to maximize profits or minimize costs. Their decisions are facilitated by their information systems,

which provide them with frequently updated data of the current situation in their area of interest or responsibility. The gate manager watches the current gate occupancy on the screen and decides which gate will be allocated for an aircraft due to arrive within the next minutes. In the same way, the arrival manager has a view of the situation on the runway and the ground manager has a view of the taxiway network and so on.

So, the managers make their decisions based on a restricted view of the airport and this leads to suboptimal solutions. It is like a "short-sighted" algorithm that gets stuck in local optima because it fails to explore the whole landscape and find the global optimum. A-CDM addresses this problem by the sharing of information in a transparent and systematic way, so that the airport partners collaborate on making decisions which enhance the overall operational efficiency of the airport. One of the airport operations where many stakeholders are involved and where the application of the concept of A-CDM is quite important is the Turnaround process.

2.1.1 The TITAN Project

TITAN is an abbreviation for "Turnaround Integration in Trajectory and Network" and it is an ongoing project (October 2011) partially funded through the Seventh Framework Program of the EU and fully compatible to the SESAR Concept of Operations [42]. The SESAR concept extends the flight operation management to include the turnaround process with the rationale that when an aircraft is parked on the ground, "*its trajectory is not evolving in the spatial dimensions but it continues to evolve in the time dimension*" [6]. The turnaround process is also in a time-sequence between the previous and the next flight, so a delay in the turnaround process will result to a late departed flight.

According to [6], "*the airport delays in 2008 accounted for around 26.6% of total delays, with an increasing trend*" and the principal origin of airport delays is the turnaround process. There are a lot of tasks to be coordinated during the turnaround process; the unloading of passengers and baggage / cargo, the inspection, cleaning and refueling of the aircraft, catering and potable water replenishment, security checks and the loading of passengers and baggage / cargo for the next flight are the most significant. There are a lot of factors that can cause a delay in one or more of these tasks and this delay can propagate to the following tasks thus delaying the whole process.

The TITAN project builds upon the A-CDM concept of sharing information among the stakeholders by extending this information to include landside and off-airport data. For example, a train arriving late at the airport terminal or traffic congestion on the highway leading to the airport may result to passengers coming late for check-in. This information is obtained by automated sensing facilities and is fed to the TIS (TITAN Information Service) module together with A-CDM data about the airside situation, in order to proactively assess possible delays. For this purpose, TITAN follows a service oriented approach (SOA) using the notion of "milestones" as sets of temporal checkpoints to provide the following services:

- Passenger Flow Information Service - PFIS
- Baggage Flow Information Service - BFIS
- Cargo/Mail Flow Information Service - CMFIS
- Aircraft Status Report Service - ASRS
- Airport Information Report Service – AIRS

2.1.2 The CAED Project

Another significant work giving a different perspective on how to coordinate the turnaround process efficiently is the CAED project developed by Delft University of Technology for Eurocontrol. CAED stands for "Coordinated Airport through Extreme Decoupling" and the main assumption of this approach is that "*local parties are in the best position to plan their resources and activities*" [19]. This work recognizes the usefulness of A-CDM but points out that it is mostly focused on the airport processes around ATC and proposes the methodology of "Extreme Decoupling" as "*a means of integrating the ground services and turnaround management with the overall airport planning*" [19].

Regarding the coordination of the actors participating in the turnaround process, two solutions have been proposed; fully centralized planning and fully distributed planning. The first one disregards the fact (stated above) that local parties have their own organization, business model and know-how in order to plan their resources and the second one introduces much complexity for resolving planning conflicts. The CAED project aims to entirely decouple the planning functions using ideas from both centralized and distributed approaches. The whole planning procedure is divided into three steps:

1. An overall decoupling of tasks to be performed based on time dependencies. A decoupling algorithm using Simple Temporal Networks (STNs) will distribute and assign time slots to the local actors.
2. The local planners will make their planning for their assigned time slots independently and without the need for communication with other parties.
3. The resulting local plans will be merged together into an overall operational plan, which, according to properties of the STN, is guaranteed to be conflict-free.

For example, assuming that the tasks to be completed during turnaround are the ones described in the previous section (passenger and baggage unloading, fueling, cleaning etc), the actors are the cargo loading, the fueling, the cleaning, the catering and the passenger boarding operator. Some of their tasks follow a temporal sequence and some others are independent. Cleaning and fueling must take place after all the passengers have evacuated the aircraft, but they are independent of each other, so they can take place simultaneously. Each task has a specified duration (with some variations possible), depending on the aircraft type. Everything must be finished before the estimated departure time. This way, a STN is built showing work flows and temporal constraints in the form of inequalities. The decoupling algorithm iteratively finds the solution that satisfies this set of inequalities and accordingly assigns the time slots during which the local actors will do their tasks.

The creation and consecutive decoupling of a Simple Temporal Network starting from a ground handling example is presented in the main reference of this section, which is the second deliverable of the CAED project [19]. Extensive theory about Temporal Constraint Satisfaction Problems, a special case of which is the Simple Temporal Problem, can be found at the first deliverable of the CAED project [18], where a STN can alternatively be represented as a directed edge-weighted graph, called a Distance Graph, which then can be examined for consistency, i.e. having at least one solution, using the Floyd-Warshall algorithm [43].

2.2 Tools and Products for the Taxiway Routing Process

After the presentation of the concept of A-CDM and the current research on its application to the turnaround process, the focus moves back to the taxiway routing process. The works and commercial software tools described here are Jeppesen Total Airspace and Airport Modeler (TAAM), Taxi Planner Optimization (TPO) and ATRiCS Surface Manager (SMAN). The purpose of presenting these different approaches is to provide an as-complete-as-possible image of the subject of the present thesis.

2.2.1 Jeppesen Total Airspace and Airport Modeler (TAAM ®)

Jeppesen TAAM is a simulation software tool developed to support and facilitate planning, analysis and decision making for Civil Aviation Authorities, Airport Operators and Airlines. TAAM provides 4D models of airports and the airspace. The basic features of this tool, that determine its usefulness and advantages compared to other modeling software, are the following:

- It can be configurable to any airport or airspace
- It can run accurate and detailed simulations in fast-time
- It employs a number of parameters that can be set in order to simulate a wide range of scenarios
- It offers a rich and comprehensive graphical interface to monitor the movement of aircraft

Being a fast-time simulation tool, TAAM can be combined with optimization or planning techniques and evaluate their proposed solutions by modeling and simulating scenarios based on these solutions. A prototype for the automated optimization of taxiway placement using TAAM and a Genetic Algorithm [31] can be found in [23]. This work addresses the problem of increasing the throughput of an airport - i.e. the number of aircraft it can serve – based on higher traffic demand. The airport operator wants to assess the degree of throughput increase as a result of building new taxiways on the airport. How many taxiways and in which part of the existing taxiway network structure should they be added so that the airport can serve more aircraft?

For this purpose, a TAAM model of Sydney International Airport was created and a realistic number of flights were simulated on different configurations of the airport, adding up to 50 new taxiways and their combinations. The genetic algorithm (GA) was chosen as the optimization technique because its qualities are suitable for the nature of this problem. The GA can generate different configurations of the airport by adding various numbers of taxiways on different locations. Then it can evaluate these configurations using the results of TAAM simulations and combine the "fittest" solutions to create new generations of improved configurations, searching for the optimum.

The use of TAAM is not restricted to the taxiway network structure of an airport; it is a much more generic tool. It simulates the 4D aircraft trajectory from gate to gate, both airborne and on the ground. According to the paper [23] referenced in the previous paragraph, TAAM can facilitate the reconfiguration of an airport. It can also measure the effect of introducing new flights operating on this airport or the impact of disruptions, such as construction works on the runways or taxiways. Civil Aviation Authorities can use TAAM to simulate situations of difficult weather conditions or analyze and redesign the use of different airways. Airlines can simulate and evaluate their operations thus leading to changes that can reduce costs and delays by a more effective use of their resources.

2.2.2 Taxi Planner Optimization (TPO)

This work of A. Marin and J. Salmerón "*introduces taxi planning optimization (TPO) as a methodology to guide airport surface management operations*". The purpose of this work is to "*improve aircraft taxiing routes and their schedule in situations of congestion, minimizing overall taxiing time (TT), and helping taxi planners to meet pre-specified goals such as compliance with take-off windows, TT limits and trajectory conflicts*" [22] (abstract). This tool was developed as part of the European Commission project "LEONARDO" and used the Barajas Airport of Madrid as the base for implementation.

In their paper the authors assume that the primary management tasks in the operation of an airport are the following:

- *Arrival management (AM)*, which estimates the landing time and runway exit a few minutes before the aircraft touches ground
- *Departure management (DM)*, which estimates the time that an aircraft is pushed-back from its parking position and establishes calculated take-off time windows (CTOTs)
- *Gate management (GM)*, which assigns arriving aircraft to the available gates

The process which interacts with all of the above mentioned ones is the process of taxiway routing and the successful operation of the runways and the gates depends also on the efficient operation of the taxiway. The management tool that is presented in their work, TPO, must be coordinated with the tools for AM, DM and GM with the continuous periodical exchange of updated data depicting the situation from the present and within a certain look-ahead planning timeframe at each management area. The proposed and efficiently tested operation pattern is to iteratively use data from the AM, DM and GM modules, execute for 1-2 minutes in order to respectively optimize the taxiway routes for the next 15-30 minutes, feed the output back to the other modules, receive updated data after 3 minutes and so on.

The model described in the paper is a "*large-scale space-time multi-commodity network with capacity constraints*" [22]. The definition of *space-time* network can be considered as an alternative name for *4D* network and the foundations of the TPO model (the structure of the network graph, the definition of the origin and destination) have similarities with the model developed for the present work. The objective function is a weighted sum of the minimization of the overall taxi time and of penalties for delayed take-offs that don't meet their respective CTOTs, thus providing the flexibility to the planner to establish tradeoffs between these goals. Finally, the model is solved as a mixed-integer optimization problem using Branch & Bound.

2.2.3 ATRiCS Surface Manager (SMAN)

The works presented in the previous two sections are related to the taxiway routing process but not in the same degree and not from the same point of view. On the one hand, Jeppesen TAAM is a 4D modeling and simulation tool, adaptable to different airport and airspace configurations – not only taxiways - and scenarios. It is not an optimization tool though, but it can be combined with such tools producing considerable results. TPO on the other hand is a research work focused on the taxiway and performs optimization as its name denotes, but it is not a commercial product, at least not presently. ATRiCS Surface Manager is related to the taxiway routing process from yet another perspective, it is neither similar to TAAM nor to TPO.

ATRiCS Surface Manager is a software system which automates taxi time calculation, routing, guidance and control services in one application. It can be configured for basic and advanced implementation levels, depending on the size of the airport. Based on the current information, SMAN is operational at Incheon Airport in Seoul, Korea and Kuala Lumpur Airport in Malaysia and has been tested in field trials at Frankfurt am Main, Munich and the under construction new airport of Berlin-Brandenburg. The SMAN system supports the controller who is responsible for the ground traffic by suggesting or assigning optimized taxi routes to the aircraft and by guiding the aircraft accordingly. The system is integrated into the airport's surveillance and lighting systems and uses them in order to continuously check and adjust the situation on the runways, taxiways and apron. The subsystems of SMAN are the following:

- *Variable Taxi Time Calculation*: This is a forecasting system which dynamically computes taxi time estimates up to 60 minutes in advance. This work acknowledges that "*advanced taxi time calculation considerably improves the accuracy of predicted on-block and take-off times*" [13]. It uses updated estimates for landing and off-block times and plans a surface trajectory, from which the taxi time is deduced. To ensure robustness, VTTC uses a stochastic model of the airport where it incorporates factors like traffic, preferred taxi routes and visibility and estimates taxi times for different combinations of these factors.
- *Routing*: The routing system has two main functions, the creation of a taxi route and the assignment to the aircraft. The system can be used in a manual, semi-automatic or automatic mode, depending on who creates and who assigns the route. In semi or fully automatic mode, SMAN dynamically proposes a route to the controller, which it creates from scratch and in regard to preferences and feasibility constraints. Of course, this route must also be efficient, i.e. minimize taxi time and distance.
- *Guidance*: Once a route has been assigned to an aircraft, the guidance system takes over. "*During taxi, SMAN automatically switches the taxiway centerline lights to unambiguously indicate the assigned taxi route to the pilots. At any time, the controller can manually control stop bars and illuminate taxi route sections to statically indicate admissible taxiways to pilots and drivers*" [13]. The taxiway lights can also be adjusted to turn off after the aircraft has passed from the corresponding taxiway, thus saving energy.
- *Control*: The control system uses the surveillance systems of the airport to detect high traffic and ensure provision of spacing, both lateral (when aircraft converge at intersections) and longtitudal (when one aircraft follows another). In both cases, when the distance falls below a certain level, an alert is generated. Alerts are also generated in cases of excessive taxi speed, route deviations and deadlocks, as well as when an assigned route does not comply with wingspan restrictions. SMAN can also control the runway stop bars to protect arriving and departing aircraft from other mobiles entering the runway.

Concluding this section, the presentation of SMAN shows how an existing software system can be implemented at an airport with the purpose of increasing safety and reducing various operating costs by the automated and efficient routing of aircraft on the ground. Airport operators and authorities are nowadays looking to this direction.

2.3 Algorithms for Routing Problems

This section closes the state-of-the-art chapter by presenting a number of algorithms that have been applied to problems similar to 4D Taxi Routing on Ground, as well as some general purpose heuristics which could potentially be adapted to the particularities of this problem. Starting with the heuristics, the following are well-known and extensively applied algorithmic methods based on the imitation of natural processes:

- The Genetic Algorithm [31] imitates the natural process of selection and reproduction of the fittest individuals in every generation. The success of this method depends on how well a potential solution of the specific problem can be encoded in a data structure such that the operations of crossover and mutation can be performed with meaningful results. A balance between "exploitation of the best solutions" and "exploration of the search area" is also a key factor, as in most heuristics.
- The Ant Colony System imitates the way ants tend to follow the shortest paths on their movements among their colonies and how these paths tend to stabilize towards optimality even after changes in the "search area". Some very interesting applications which show the efficiency and robustness of the Ant Colony Systems as optimization algorithms can be found in [25] and [26].

The problem of 4D Taxi Routing on Ground is a shortest path problem enhanced with the dimension of time. The prototype algorithm for solving the basic single-source shortest path problem is the algorithm of Dijkstra [39]. A variation of this algorithm was implemented for the present thesis work and will be described in the fourth chapter. An efficient heuristic which assumes the existence of a distance table between the destination and each intermediate vertex of the network graph in order to be applied is the A* Algorithm [40].

There are also numerous and various extensions of the shortest path problem, which were formulated in order to model the increasing complexity and diversity of contemporary real-world network and routing problems. The K-Shortest Path Problem models the need to find and use the best alternative solutions when one shortest path might not be enough. A description of the problem and a presentation of algorithms that aim to solve it can be found at [27].

A taxiway network can be hierarchical in the sense that it can consist of a number of primary and a number of secondary taxiways. The object of the Hierarchical Network Design Problem is "*to identify the least cost, two-level hierarchical network*", which "*must include a primary path from a predetermined starting node to a predetermined terminus node*" and also "*each node not on the primary path must be connected to some node on that path by means of a secondary path*" [28].

Another generalization of the shortest path problem is the Language Constrained Shortest Path Problem. This problem assumes the existence of "*an alphabet Σ , a graph G whose edges are weighted and labeled in Σ and a regular language $L \subseteq \Sigma$*". The problem "*consists of finding a shortest path p in G such that the concatenated labels along p form a word of L*" [29]. This problem applies on multimodal networks, where roads are differentiated by categories. Last but not least, the Canadian Traveler Problem is "*a stochastic shortest path problem in which one learns the cost of an edge only when arriving at one of its endpoints. The goal is to find an adaptive policy (adjusting as one learns more edge lengths) that minimizes the expected cost of travel*" [30].

At this point, the problem of 4D Taxi Routing on Ground has been described but not analyzed or formulated into a model. However, and based on these descriptions, we can speculate on the applicability of the previously presented algorithms to this problem and identify possible limitations. The heuristics have the advantage of being general purpose methods, therefore applicable to a wide range of problems. The success of the Genetic Algorithm lies on the encoding of the solutions, and it can combine (crossover) different parts of the taxiway network into overall routing solutions, so it can be an efficient method to deal with this problem. The Ant Colony System seems to be even more promising, due to its flexibility and stabilization ability; it could be executed dynamically while the aircraft are routing, by periodically sending agents (ants) to the destination to find the shortest path at that point in time and adapt the aircraft routes accordingly.

The algorithm of Dijkstra applies by definition to static graphs and does not incorporate the notion of time, so it should somehow be enhanced in order to be applied to a 4D routing problem. The implemented variation of this algorithm shows certain inefficiencies too, which will be discussed in the conclusion chapter of the present thesis. Moreover, the A* algorithm assumes that there exists a coordinate distance map for all pairs of vertices in a graph and this distance map is not always feasible to obtain or calculate. Therefore, the applicability of A* is generally more limited. On the other hand, having a list of alternative routes for each aircraft and destination at a given time point would be an effective way to handle dynamic changes on the taxiway, so a K-Shortest Paths algorithm could be a good candidate for our problem.

The rest of the routing problems described in the present section were chosen in order to present some concepts that could be applicable to the 4D Taxi Routing on Ground problem, perhaps combined or extending an already existing model. A taxiway network could have the form of a hierarchical network, with primary and secondary taxiways, so this topology could serve in such cases. There could also be distinctions among vertices or edges of a taxiway graph imposing constraints that are expressed with the aid of a regular language. Finally, a different approach would be a stochastic one, similar to the Canadian Traveler that aims to deal with the uncertainty and the dynamic nature of the problem by finding adaptive policies.

Chapter 3 Conception and Model Definition

After introducing the problem of 4D Taxi Routing on Ground and placing it into the context of Airline Operations Research and after a presentation of the current state-of-the-art regarding research on this topic and algorithmic classes that have been applied to similar problems, the next step is to formulate a structured definition that will lead to a sound mathematical model. This chapter is divided into two main sections. In the first section we formulate and analyze the problem of 4D Taxi Routing on Ground by distinguishing and describing the three main entities that compose it. The rationale for this distinction is explained in the beginning of the section. Following and based on the definitions of the first section, we formulate the mathematical model of the problem, including the measures for the objectives, in the second section.

3.1 The Entities

ICAO defines the following concepts [1]: An airport (or aerodrome) is "*a defined area on land or water (including any buildings, installations and equipment) intended to be used either wholly or in part for the arrival, departure and surface movement of aircraft*". Depending on factors like size and traffic, an airport can have one or more runways. A runway is "*a defined rectangular area on a land aerodrome prepared for the landing and take-off of aircraft*". Another important part of an airport is the apron, "*a defined area, on a land aerodrome, intended to accommodate aircraft for purposes of loading or unloading passengers, mail or cargo, fuelling, parking or maintenance*". Finally, a taxiway is *a "defined path on a land aerodrome established for the taxiing of aircraft and intended to provide a link between one part of the aerodrome and another*".

An airport usually has a network of taxiways, resembling a city road network in the sense that there might be intersections, 90° turns, speed limits and defined directions that an aircraft must follow when moving on the taxiway. So, the taxiways are where the "surface movement" mentioned in the previous paragraph takes place. On arrivals, this movement starts from the runway where the aircraft lands and ends at a specific parking position on the apron. On departures it is the other way around. Therefore, a taxiway usually provides a link between the runway and the apron. The other cases of surface movement (maintenance, overnight parking, deicing) are out of the scope of the present work. However, the model described in the second part of this chapter can be extended in order to include such movements.

The runway and the apron serve as the source and the destination (and vice-versa) in the taxiway routing process. Nevertheless, the operations taking place on the runway and the apron are not a concern of the taxiway routing process, even though they may affect it. For example, if an aircraft fails to brake effectively in order to use the runway exit which was commanded by ATC, it will have to use another exit, so the taxiway route will have to be redefined using the new runway exit as a starting point. However, this can be considered as different input data provided to the taxiway network system and not as a change to the system itself. Concluding this discussion, the first main entity that composes the problem of 4D Taxi Routing on Ground is the **Taxiway Network**.

The taxiway structure of an airport is more or less static; it does not change frequently. As stated in the introduction chapter, the generation of a path connecting a runway exit and a parking

stand on the apron would be a trivial task on an empty taxiway. What provides the dynamic aspect to the whole procedure is the concurrent movement of other aircraft on the taxiway network, making the availability of certain paths a function of time. The second main entity of the problem is thus the **Aircraft**.

From an abstraction level, the Taxiway Network supplied with Aircraft moving on it can be considered as a separate system within the surrounding system of the whole airport. These two entities can define the model of the taxiway network system, as it will be shown later in this chapter. The interaction of the taxiway network system with its surroundings is the information of the source-destination pair and the time that an aircraft appears at the source, as an input, and the time that the aircraft reaches its destination (and possibly other metrics), as an output. This information depends on several different factors, which can be grouped together under the term **Airport Operations**. This is the third entity of the problem. The airport operations include:

- The daily / weekly / seasonal flight schedule that provides the expected times of arrivals and departures.
- The runway usage pattern, especially in airports with more than one runway. The usage of the runway(s) and the direction of arrivals and departures on them affect the choice of the runway exits as starting or ending points of the taxiway routing.
- The airlines operating on the given airport and the corresponding airports of the flights departing from or arriving at the given airport. The combination "airline - source airport" defines the terminal of the given airport where the aircraft will be parked, so it affects the choice of the parking stands.

As a conclusion, one can say that for the problem of the 4D Taxi Routing on Ground, the taxiway network provides the structure or topology (the spatial dimensions), the movement of aircraft adds the 4th dimension (time) to the system and the airport operations stand for the environment, connecting the system to the real world by providing data and operational constraints.

3.1.1 The Taxiway Network

The Taxiway Network is the set of taxiways of an airport. Taxiways serve as links between the runway and the apron, at least for the purposes of the present work. Apart from the runway exits and the exits to the parking areas, which are the terminal points of this network, other significant places are the taxiway intersections, defined as "*junctions of two or more taxiways*" [1]. An intersection can be a *crossing*, where two roads meet but an aircraft moving on the one road cannot enter the other, a *merging*, where two or more roads merge into one, or a *splitting*, where one road splits into two or more. An intersection can also be a combination of the above. It is obvious that intersections are important on the taxiway routing, because either a check to avoid conflicts (crossing, merging) or a routing decision (splitting) has to be made.

The most intuitive and general purpose mathematical structure that serves as a basis for modeling networks (among others) is the Graph. Using a general definition, a graph is "*an abstract representation of a set of objects where some pairs of the objects are connected by links. The interconnected objects are represented by mathematical abstractions called vertices, and the links that connect some pairs of vertices are called edges*" [36]. A taxiway network for an airport can be

modeled as a graph, where the vertices are the terminal points (runway exits and exits to the parking areas) and the intersections. The edges of this graph are the taxiways or parts of them that connect any two of these significant points (vertices). A graph can have many specializations or extensions, and for the purposes of a taxiway network the edges must be directed, according to the directions of the respective taxiways. The edges can also be supplied with a weight or cost that corresponds to the length of the respective taxiways (from a spatial perspective) or the time to traverse them (from a temporal perspective).

After defining the main entities that participate in the 4D Taxi Routing process, a crucial step towards modeling the problem in order to develop algorithms that attempt to deal with it, is to create a graph that corresponds to the taxiway network and captures its structural foundations and particularities. Having separated the taxiway network from its operational environment, the only information needed in order to create the graph, is a full map of a given airport plus a database containing numerical representations of the airport dimensions (lengths of taxiways) and structural constraints (maximum wingspan and weight allowed on specific edges).

For the present work, all the necessary information was extracted from the Airport Mapping Database (AMDB) of Jeppesen. In the next subsection a case study will be presented in order to illustrate the transition of an airport map and relevant information to a graph representing the taxiway network.

3.1.1.1 The Taxiway Network of Stockholm-Arlanda Airport

The airport we chose to use as a case study for this work is the airport of Stockholm-Arlanda (IATA: ARN – ICAO: ESSA). It is the largest airport in Sweden, the third largest airport in the Nordic countries, and the second busiest in terms of international passengers [37]. The choice was mainly based on the size of the airport and the complexity of its taxiway network. As of October 2011, Stockholm-Arlanda has 3 runways and 4 terminals and its traffic density can be categorized as follows:

According to [1] (pages 1-1, 1-2), "*aerodrome traffic density is medium where the number of movements in the mean busy hour is of the order of 16 to 25 per runway or typically between 20 to 35 total aerodrome movements.*" Also, "*the number of movements in the mean busy hour is the arithmetic mean over the year of the number of movements in the daily busiest hour. Either a take-off or a landing constitutes a movement.*" The published statistics for Stockholm-Arlanda airport [3] show that for the year 2010, the daily busiest hour was from 17:00 to 17:59 with an average of 39 movements (20 take-offs and 19 landings). According to these facts, Stockholm-Arlanda can be considered as a medium to heavy traffic density airport.

For the purposes of the present work, the airport that would serve as a case study should be fairly large and complex, in order to help exploring the problem dimensions; a regional airport with one runway and one taxiway would not serve. On the other hand, a very large airport (for example Frankfurt am Main International airport) would require a substantial amount of time to be modeled. The maps of Stockholm-Arlanda airport are displayed in Appendix I. The resulting graph for the taxiway network is accordingly displayed in figures 3.1 and 3.2 on pages 20-21, while the necessary remarks / clarifications are given right after.

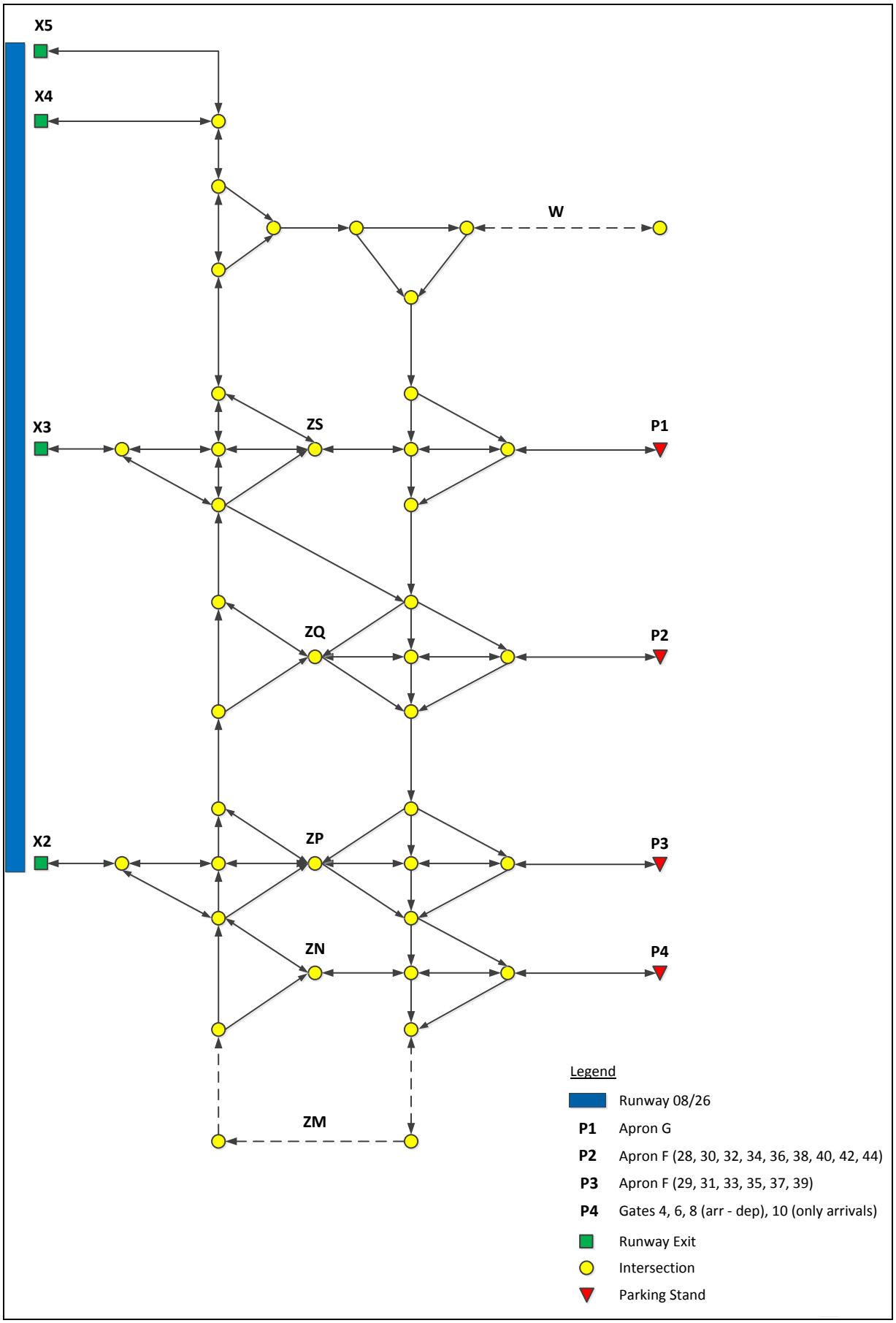


Figure 3.1: The graph of the northern part of Stockholm-Arlanda airport taxiway network

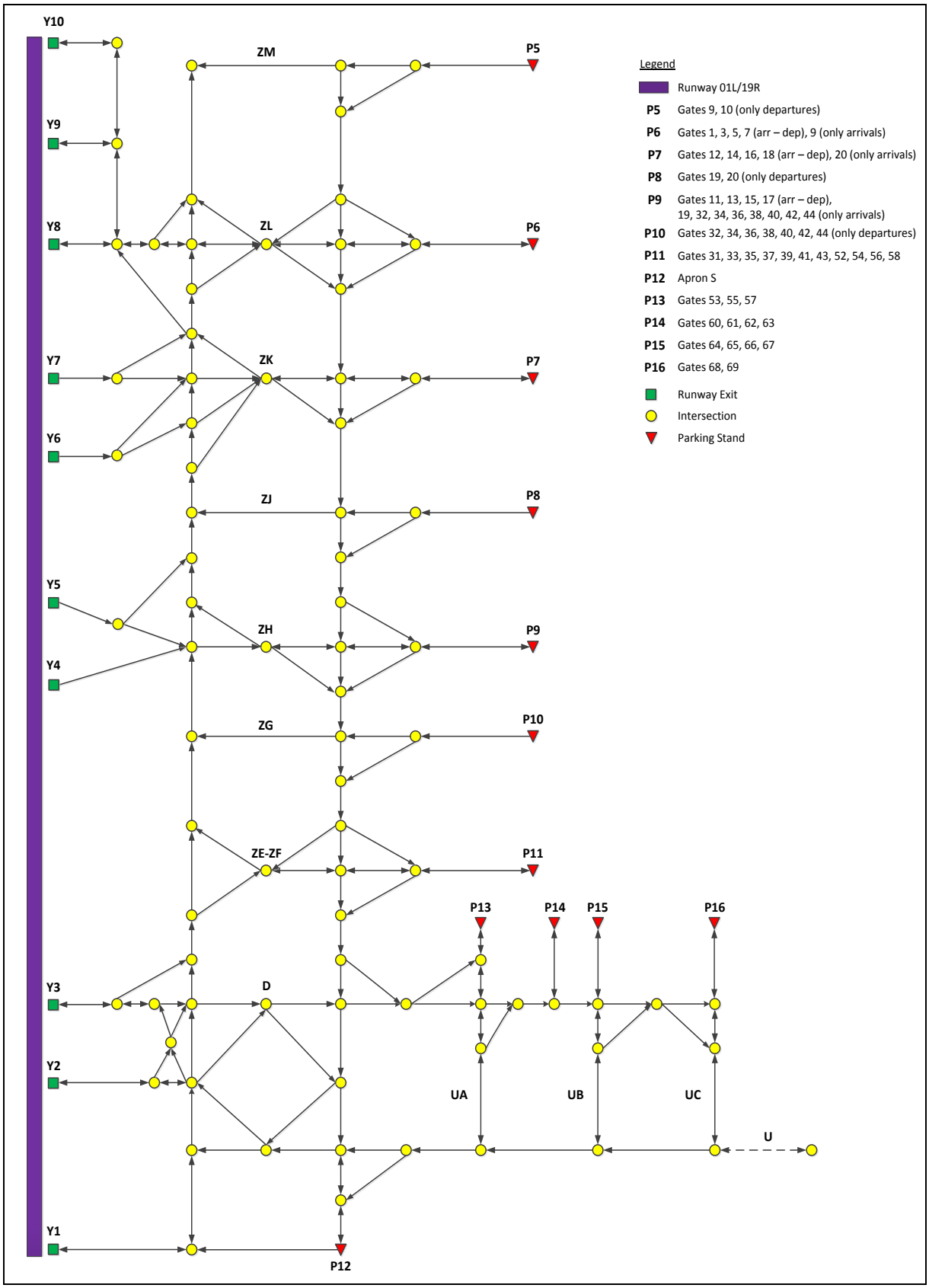


Figure 3.2: The graph of the southern part of Stockholm-Arlanda airport taxiway network

The figures 3.1 and 3.2 represent the sub-graphs for the northern and southern part of the taxiway network of Arlanda airport. The taxiway identifiers (ZS, ZQ, ZP etc) are shown as references to the airport map, so that the correspondence between the graph and the map can be more evident. It is also evident (see identifier ZM on both graphs), that these two sub-graphs can be merged into one overall graph that displays the whole taxiway structure, apart from the taxiways leading to the third runway - 01R/19L (see maps B and D in Appendix I). This part of the airport was left out of the figures 3.1 and 3.2 but was included in the implementation of the model. The long taxiways leading to and from the region of the third runway - 01R/19L – are named U and W and are shown in figures 3.1 and 3.2 with dashed lines.

The runways included in the figures are drawn as long rectangles (see legends) and the runway exits / entrances are the rectangular green vertices of the graph. The triangular red vertices (P…) are the exits / entrances of the taxiway leading to and from parking areas on the apron. The specific gates that can be reached from a given taxiway exit are also shown on the legend of the graph. It should be noted here that the parking areas / gates are located on the apron, so they are not a part of the taxiway structure. The rectangular green (X..., Y...) and triangular red vertices are the terminal points of the graph. The circular yellow ones stand for the taxiway intersections; note the existence of crossings, merges, splits and combinations of them.

A detailed comparison of the figures 3.1 and 3.2 with the respective maps can lead to the conclusion that some parts of the apron are excluded from the graph, i.e. there are no taxiway exits leading to and from them. The reason for the exclusion is that these locations are used for other purposes. They can be maintenance hangars or serve cargo aircraft, but anyway they are not used for commercial flights. Detailed information about the local regulations at aircraft parking stands can be found at [4]. We also obtained relevant information by exchanging emails with people working at the ATC of Arlanda airport.

The edges of the graph are directed. Edges with arrows on both ends represent taxiways that can be traversed both ways. In this case, one direction is for arrivals and the other is for departures. A comparison between the maps that display the same part of the airport but in different modes (northern part: map A for arrivals and C for departures / southern part: map B for arrivals and D for departures) leads to the conclusion that there are significant differences. There are edges used only for arrivals, edges used only for departures, edges used for both modes in the same direction and finally edges used for both modes in opposite directions. The last of these four “edge types” can be distinguished on the figures 3.1 and 3.2 by the existence of arrows on both ends, as mentioned at the beginning of this paragraph. However, the graph does not provide further information about the unidirectional edges and if they used for arrivals, departures or both.

There are also edges whose one vertex is a runway exit. One can notice that some of these edges are unidirectional and some others are bidirectional. For example, runway exit Y7 can be used only for arrivals, while Y10 can be used for both arrivals and departures, depending on the runway exit usage pattern which belongs to the entity of Airport Operations and will be analyzed in section 3.1.3. The same remark can be made for the exits to parking areas too.

At this point it is important to introduce the notion of mode – arrival or departure – and suggest that the graph displayed in the previous pages is not sufficient for modeling all the aspects of taxiway routing. This graph is actually a merging of two graphs, the arriving and the departing one.

This can be shown on the figures 3.3, 3.4 and 3.5 below, where a fraction of the taxiway network is used as an example.

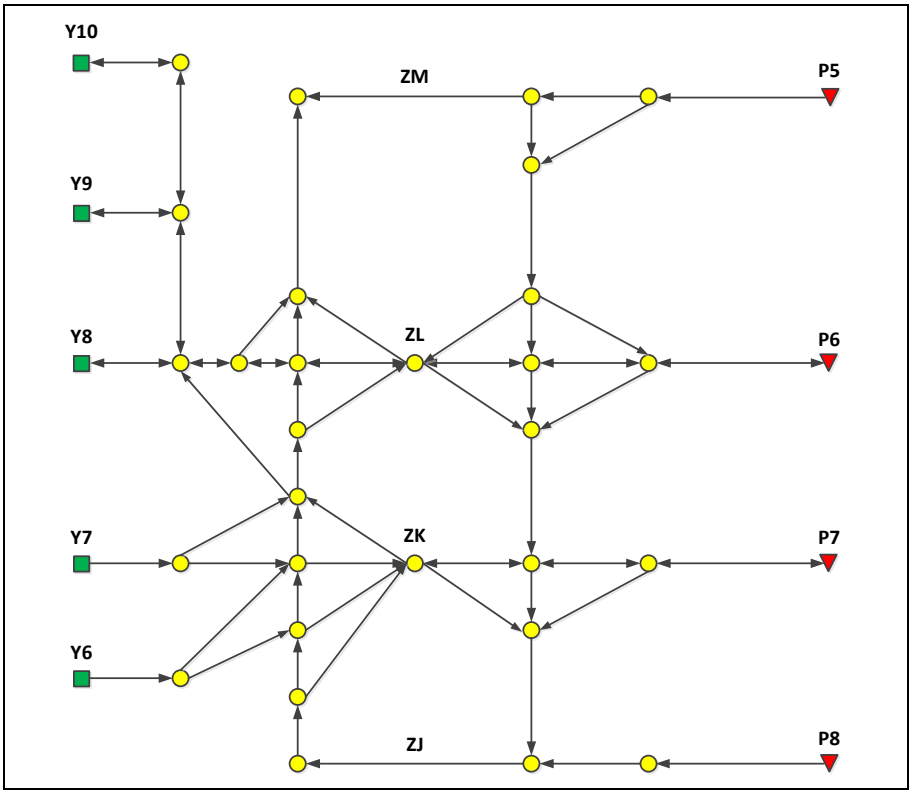


Figure 3.3: A fraction of the graph of Figure 3.2, used as an example

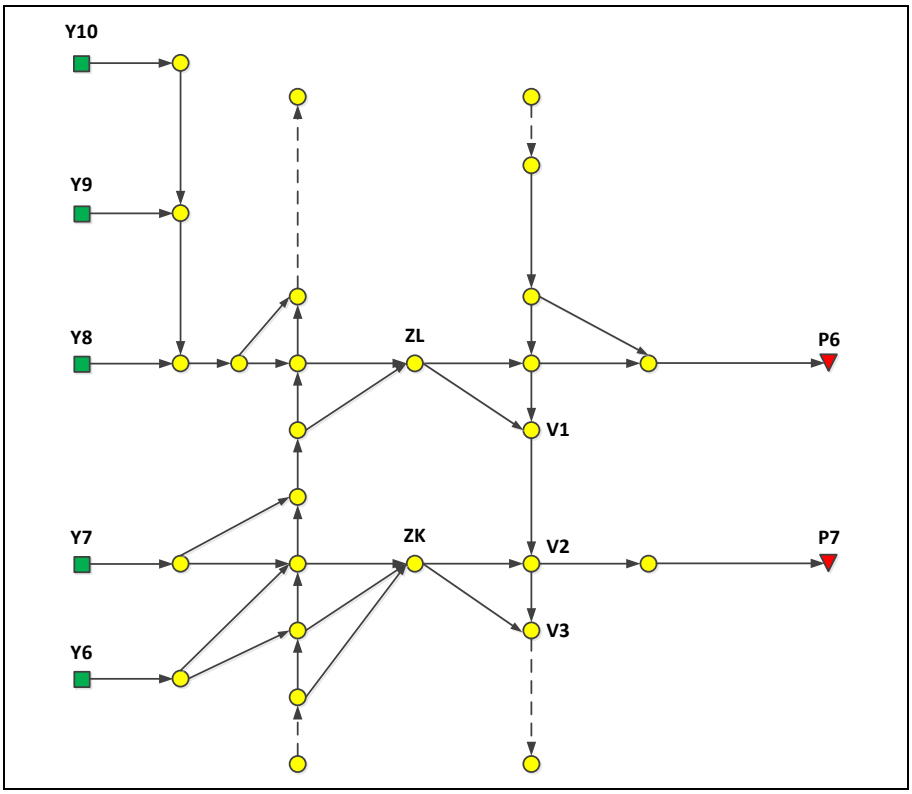


Figure 3.4: The arrival "component" of the graph of Figure 3.3

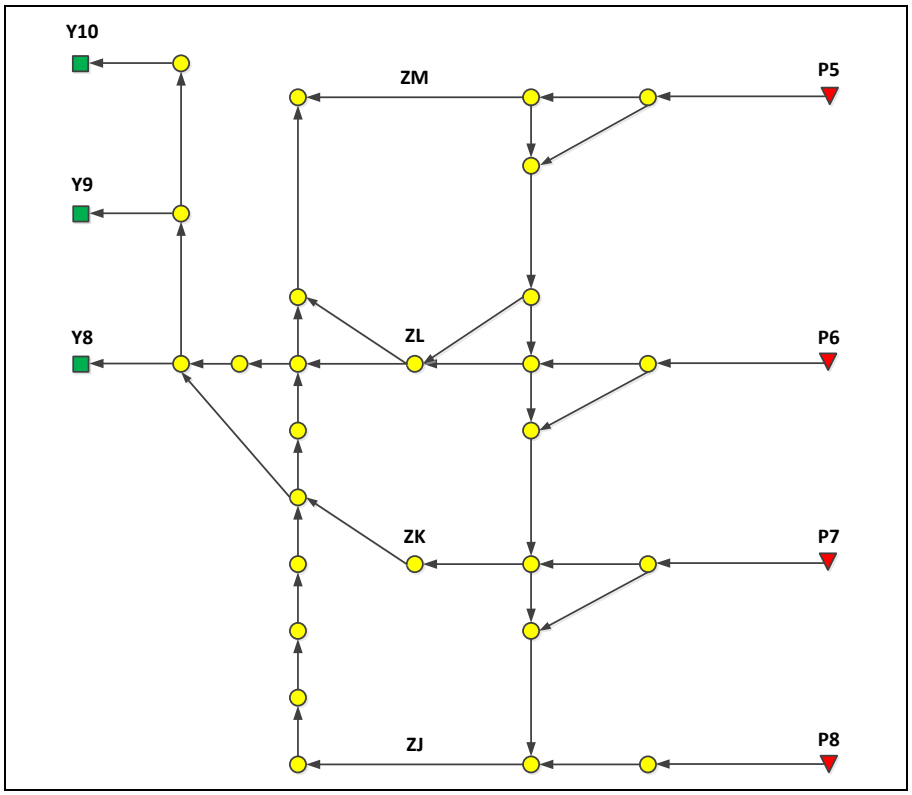


Figure 3.5: The departure "component" of the graph of Figure 3.3

The example figures 3.3, 3.4 and 3.5 can be used for distinguishing the four edge types. The edges starting from runway exits Y6 and Y7 are used only for arrivals. The edges starting from parking area exits P5 and P8 are used only for departures. Runway exits Y8, Y9 and Y10, as well as parking area exits P6 and P7 are used for both modes (bimodal) in opposite directions. Finally, the two vertical axes – named Y and Z on the maps – are used for both modes but in one direction.

The next section summarizes the necessary steps for the transition from an airport map to a graph that captures all the essence of taxiway routing and serves as a basis for the mathematical model. Before proceeding to that, a last remark that can be derived from the figures above is that when an aircraft reaches a vertex on the graph, the possible next movements are not determined only by the graph itself, but also by some transition table. For example, on figure 3.4, an aircraft reaching vertex ZL can be directed towards the parking area of P6 or can turn to V1. From V1 the only choice is to continue to V2, which corresponds to a crossroad, so in this case the aircraft must continue to V3. In order to reach the parking area of P7, an aircraft must be coming from ZK. This example shows that the existence of an edge starting from a given vertex does not necessarily mean that this edge can be followed, but the additional information of where the aircraft comes from is also of high importance.

3.1.1.2 Summing-up: From an Airport Map to the Taxiway Network Graph

Stockholm-Arlanda airport was used as a case study for the present work in order to get some insight into the processes regarding taxiway routing. However, the conclusions must be independent of a specific airport and the model, which is one of the main goals of this work, must be generic enough

and applicable to all types of airports with the minimum of adjustments. The purpose of section 3.1.1 is to define the steps for formulating a graph, including any accompanying information, for the entity of the Taxiway Network, using a mapping database of a given airport. After the discussion that preceded and the identification of the problem dimensions with the help of a case study airport, the map-to-graph procedure can be summarized in steps as follows:

1. Obtain and study the maps showing the taxiway network structure. Using airport regulation information, limit the taxiway network to the set of taxiways used for arrivals and departures.
2. Identify the runway exits and the exits to parking stand areas and define the modes on which they are operated. These shall be the terminal vertices of the graph.
3. Identify all the taxiway intersections in the area of interest. These shall be the inner vertices of the graph.
4. Connect the vertices with edges according to the taxiways on the maps. For each edge record the crucial information: its type (one of the four types defined in the discussion above), its length (in meters or some other base unit) and any existing constraints, like maximum speed wingspan and weight. The maximum speed is usually not stated explicitly, but there are some empirical limits depending on the form of the taxiway (see chapter 1).
5. For each edge and allowed direction record the possible next edges, thus creating a list of allowed transitions.

An example of tables (some sample rows) with all the necessary data for a Taxiway Network graph is displayed below. Table 3.6 shows some edges with the data of step 4. For example, the edge [27, 28] has a length of 59 meters, the maximum speed is a fraction 0.6 of the global maximum for the specific airport (because this edge corresponds to a turn on the taxiway, so the speed limit must be lower), the maximum wingspan is 65 meters and the edge is used only for departures. The edge [29, 30], on the other hand, is used on both modes in opposite directions, i.e. from vertex 29 to 30 on arrivals and from vertex 30 to 29 on departures, and has a length of 100 meters etc. Also, the edge [31, 33] is used on both modes but in the same direction and has no wingspan limit (equals zero).

vFrom	vTo	length	speed	wingspan	arrival	departure
27	28	59	0,6	65	0	1
27	31	34	1	0	1	1
27	32	54	0,6	65	1	0
28	30	28	0,6	65	0	1
28	31	21	0,6	0	1	0
29	30	100	0,8	65	1	-1
29	33	120	0,8	0	1	0
30	31	26	0,8	65	1	0
31	32	51	1	65	1	0
31	33	89	1	0	1	1
32	120	48	1	65	1	0
32	122	82	0,6	24	1	0

Table 3.6: Representation of the edge information for a Taxiway Network graph

Table 3.7 is related to table 3.6 and shows the transitions for edges that support arrivals. Edge [27, 28] does not exist on this table, since it is used only for departures. Edge [29, 30] exists and the extra information is that when reaching vertex 30 coming from vertex 29, the only next step is to move to edge 31, then to edge 32 and then there are two choices: 120 and 122. In general, there can be more than two choices as "next edges" depending on the form of the taxiway network. This way one can traverse the graph from a terminal point and, depending on the mode and the respective transition table, a tree-like structure of paths can be generated, where the root is a runway exit and the leaves are all the parking areas accessible from this runway exit (arriving mode), or the root is a parking area exit and the leaves are all the runway entrances accessible from this parking area exit (departing mode).

vFrom	vTo	vNext1	vNext2	length	speed	wingspan
27	31	33		34	1	0
27	32	120	122	54	0,6	65
28	31	33		21	0,6	0
29	30	31		100	0,8	65
29	33	34		120	0,8	0
30	31	32		26	0,8	65
31	32	120	122	51	1	65
31	33	34		89	1	0
32	120	121		48	1	65
32	122	123		82	0,6	24

Table 3.7: Representation of the arrivals transition for a Taxiway Network graph

3.1.2 The Aircraft

Up to this point, the first entity of the problem of 4D Taxi Routing on Ground, the Taxiway Network, is described in detail, together with its particularities and a number of well-defined steps for constructing the corresponding taxiway graph. The foundations are set in order to describe the second entity, the Aircraft moving on the taxiway. An aircraft has many features, but for the purpose of the present work the focus is on a small subset of them. The airborne features, the ones that define the main operation of an aircraft (flying), do not concern the taxiway routing process and the aircraft can be abstractly considered a large vehicle, with the difference that because of its mass, speed changes – accelerations and decelerations – are undesirable in regards of fuel cost and passenger comfort.

The dimensional features of an aircraft that concern the taxiway routing process are its length, wingspan and weight. The length of an aircraft in combination with its current position determines its distance from other aircraft routing on the taxiway at the same time, which is obviously important for the avoidance of conflicts and the safety of routing. The wingspan and weight of an aircraft, as discussed both in the first chapter and earlier in the present one, determine whether it can enter specific taxiways, by comparing their structural restrictions with the according features of this type of aircraft.

From these three features, the one used in the model is the aircraft's wingspan. The weight restrictions are enforced in a similar way to wingspan restrictions; the aircraft's value is compared to the taxiway limit and an "enter"/"don't enter" decision is made accordingly. Furthermore, instead of following a very detailed approach of using the length of each aircraft, the alternative approach followed was to consider each aircraft a moving point (thus just record its position at each time) with a constant safety distance (say 100 meters) as a "tail" moving with it. Once again and similar to the discussion of section 3.1 about the other kinds of aircraft movements on the ground apart from arrivals and departures, a weight restriction and an explicit aircraft length consideration are left out of the problem model, with the rationale that:

a) the purpose of this work is to create a model for 4D Taxi Routing on Ground that is fairly simple and compact, yet covers the essential principles and most of the realistic cases, and
b) the model can be extended later to incorporate such movements and restrictions

The conclusion is that the dimensional feature of an aircraft that is used for the purpose of taxiway routing is its **wingspan**. A table containing all the aircraft types used in the Stockholm-Arlanda model implementation – according to the aircraft that actually use this airport – with their names, lengths and wingspans can be found in Appendix III. The **airline** where an aircraft belongs to is another useful feature, not for the model itself, but for the determination of the terminal and thus the parking area where the aircraft will be routed to in case of arrival. More about this process will be discussed in section 3.1.3.

Being the dynamic entity of the taxiway, an aircraft has also some features that change with time. One of them is the **position** on the taxiway. This position could be recorded using the coordinates of an aircraft at each time unit, but since the taxiway network is modeled as a graph, the position of the aircraft on the graph is a triple [from-vertex, to-vertex, distance]. For example, if the current position of an aircraft is [29, 30, 45], the aircraft is currently situated on edge [29, 30] – with direction from vertex 29 to vertex 30 – and its frontal part (its "nose") is located 45 meters from the beginning of the edge (from vertex 29). Obviously, the "distance" variable of the position triple must be less than or equal to the length of the current edge.

Another feature that changes with time is the **speed** of the aircraft. The speed is usually in the range from 0 (when the aircraft is on hold) to a maximum allowed speed for the specific taxiway – edge, where the aircraft is currently situated. For example, using the data from table 3.6 and assuming that the usual maximum taxiway speed for an airport is 25 knots [8], an aircraft currently situated on edge [29, 30] – where the speed is set to the 0.8 of the overall maximum - should taxi with a speed no more than $25 * 0.8 = 20$ knots. The aircraft speed is especially important for the taxi routing process, because it has the key-role of "binding the model together". The speed determines the distance covered at each time unit, so it determines the position of the aircraft too. The speed together with the length of the route that an aircraft will follow determines the total taxi time, which is the central metric of the problem of 4D Taxi Routing on Ground. Finally, the changes of speed are themselves recorded and used as another metric of the quality of an assigned route, as stated in section 1.2.2.

As already stated and described, the aircraft is the entity that enhances the taxiway model with the dimension of time because of its movement that by definition makes its important features

(position, speed) functions of time. The aircraft is the entity that also keeps the metrics of the problem objectives: **taxi time** $\boldsymbol{tt(a_k)}$, **hold time** $\boldsymbol{ht(a_k)}$ and **speed changes** $\boldsymbol{sc(a_k)}$.

An aircraft also has a **state** which changes during the routing process, as well as a number of operations / commands that enable the state transitions. The possible states of an aircraft regarding the taxiway process are shown on table 3.8 and the state transition model is displayed on figure 3.9.

State	Description
0	Landed, initialized
1	Waiting at the runway exit to start taxiing (a first-come first-serve queue)
2	Arrival - taxiing with speed sp>0
3	Arrival - on hold (speed sp=0)
4	Parked on the apron - turnaround process
5	Waiting at the parking stand to start taxiing (a first-come first-serve queue)
6	Departure - taxiing with speed sp>0
7	Departure - on hold (speed sp=0)
8	Reached the runway to take-off / finish

Table 3.8: The aircraft states during taxiway routing

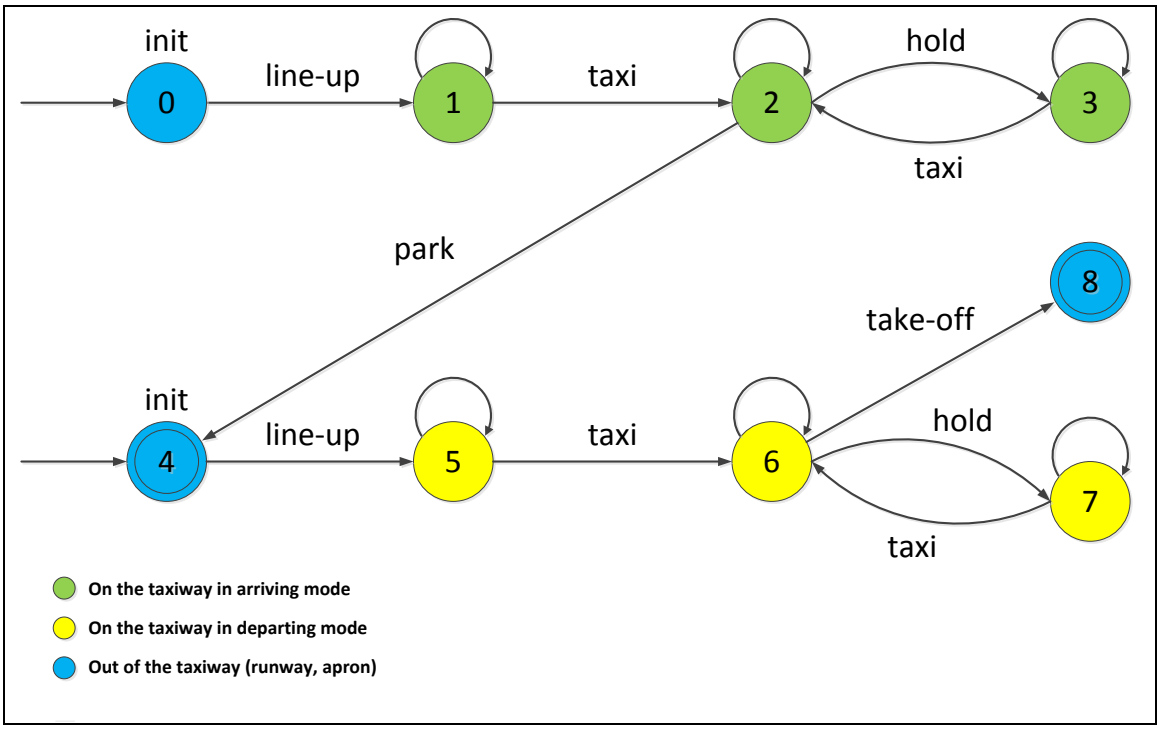


Figure 3.9: The aircraft state transition model

On figure 3.9, the names of some commands are also displayed. These commands – init, line-up, taxi, hold, park and take-off – along with a few others are responsible for the routing of the aircraft and their execution depends on the situation on the taxiway at a given point in time. The complete listing of the aircraft taxiway commands, their resulting transitions (based on figure 3.9) and the conditions that trigger them are displayed on table 3.10.

Command	Transition	Condition
Init	$0, 4$[3]	The time has come for a new aircraft to be introduced according to the flight schedule
Line-up	$0 \rightarrow 1, 4 \rightarrow 5$	The aircraft has taken its position at the runway exit or the parking area exit and must check its priority on the waiting queue
Update-priority	$1 \rightarrow 1, 5 \rightarrow 5$	Another aircraft left the queue, so the remaining update their priorities and check which is the first
Hold-on	$1 \rightarrow 1, 5 \rightarrow 5$	The aircraft is not the first on the waiting queue, must hold on
Taxi	$1 \rightarrow 2, 3 \rightarrow 2$ $5 \rightarrow 6, 7 \rightarrow 6$	The aircraft is the first on the waiting queue or is on hold and the taxiway is clear in order to start taxiing again
Update-position-speed	$2 \rightarrow 2, 3 \rightarrow 3$ $6 \rightarrow 6, 7 \rightarrow 7$	The aircraft is on the taxiway and at each time unit must update its current position and speed
Change-speed	$2 \rightarrow 2, 6 \rightarrow 6$	There is another aircraft ahead or at a crossroad, must change speed accordingly
Hold	$2 \rightarrow 3, 6 \rightarrow 7$	The aircraft is approaching a stopped aircraft ahead or an edge (taxiway) that is currently unavailable, must stop completely
Park	$2 \rightarrow 4$	The arrival routing is finished, the aircraft reached its parking position
Take-off	$6 \rightarrow 8$	The departure routing is finished, the aircraft reached the runway entrance to take-off

Table 3.10: The aircraft commands during taxiway routing

The commands of table 3.10, apart from changing the aircraft's state, have an impact on the variable features and metrics that each aircraft keeps. For example, commands "*Hold*" and "*Hold-on*" result in the aircraft standing still on the taxiway, so they increase the hold time $\boldsymbol{ht}(\boldsymbol{a_k})$ metric accordingly. The command "*Line-up*" starts the routing process of the aircraft, so the time that it takes place is

[3] It is also possible that an aircraft appears directly at state 4. This will be explained in chapter 4.

recorded and when the command "*Park*" or "*Take-off*" is given, thus the routing process is finished, this time is also recorded and their difference is the taxi time $\boldsymbol{tt}(\boldsymbol{a_k})$ of the aircraft.

Another conclusion which can be deduced from the state transition model of figure 3.9 is that the state of an aircraft determines its **mode**, i.e. states 1, 2 and 3 correspond to arriving mode and states 5, 6 and 7 correspond to departing mode. The mode of the aircraft is quite important, because the taxiway network graph is also bimodal, as discussed in section 3.1.1, and its form, therefore the taxiways that the aircraft can follow, depends among others on the respective mode. To make it simple: an arriving aircraft "sees" an arriving taxiway and a departing aircraft "sees" a departing one.

3.1.3 The Airport Operations

The Airport Operations constitute the third entity of the 4D Taxi Routing on Ground problem. Unlike the Taxiway Network and the Aircraft, they are not part of the model itself but act as the "plug-in" between the model and the outside world. An airport has many different types of operations, but the ones that affect or interact with the taxiway routing system can be grouped into the categories briefly mentioned at the beginning of this chapter, according to the kind of input they provide to the taxiway routing process.

3.1.3.1 The Time-Schedule of Arrivals and Departures

The time that an aircraft enters the taxiway depends on the time that the flight operated by this aircraft is scheduled to arrive or depart. Moreover, the flight schedule contains information, like the airline that operates the flight, the corresponding airport and the type of the aircraft, which is absolutely necessary for the generation of the route. The form of a flight schedule used in the implementation (sample rows from the summer 2010 schedule for Stockholm-Arlanda, provided by the Jeppesen TAAM team in contact with the airport authorities of Arlanda) is displayed on table 3.11.

time	airline_code	flight_no	airport_code	aircraft type	arr/dep
09:00	U2	1574	GVA	319	D
09:05	BA	776	LHR	321	A
09:05	DY	3702	UME	733	A
09:05	OK	492	PRG	735	A
09:05	DL	203	JFK	752	D
09:05	QR	92	DOH	332	D
09:05	SK	1042	KRN	736	D
09:10	JZ	428	HAD	F50	A
09:10	JZ	474	KID	F50	A
09:10	SK	531	LHR	73W	D
09:20	LH	3014	MUC	320	A
09:20	TK	1793	IST	320	A
09:25	SK	8	LLA	736	D
09:30	SK	157	GOT	736	D

Table 3.11: Sample of the flight schedule for Arlanda, 2010-11-08, from 09:00 to 09:30

The way to read this schedule is quite easy once one knows which airlines and airports correspond to the codes of table 3.11. For example, an Airbus A321 operating flight BA-776 (British Airways) from London-Heathrow is scheduled to arrive at 09:05 and a Boeing 737-600 operating flight SK-157 (SAS) to Gothenburg-Landvetter is scheduled to depart at 09:30. The tables containing the airlines operating at Stockholm-Arlanda and the destination airports - according to the flight schedule of summer 2010 – can be found in Appendix III.

3.1.3.2 The Airlines and corresponding Airports

The flight schedule provides the time that each aircraft appears on the taxiway. It also provides the airline and the corresponding airport of the flight, but the taxiway routing process is not concerned directly about this information. The input to the Taxiway Routing System is the time and the pair of terminal vertices that correspond to the source and destination of the route. In case of departures, the aircraft is already parked at a gate, so it will start routing from there. In case of arrivals, however, the aircraft must be routed to the appropriate parking area and this depends on the airline and the airport it comes from.

It is already mentioned in the previous and the present chapter that an airport has a number of terminals (not to be confused with the "terminal vertices" mentioned in the previous paragraph) and there are specific parking stand areas on the apron which lead to and from each terminal. Therefore, in order to find the destination vertex in case of arrivals, the knowledge of the terminal is necessary, even though it is not sufficient in airports where a terminal has more than one parking stand areas. The approach followed in the present work, regarding the determination of the destination vertex given the terminal, will be presented in the next chapter.

So, the challenge is to find the airport terminal – if the airport has more than one terminal - given the airline and the corresponding airport. For the purposes of this work, this function is just an intersection. The separation of the passenger area of an airport in terminals is due to legal, handling and administrative reasons. Different terminals are used for domestic flights, others for international flights. Within the international terminals, there can be different groupings of destination countries, depending on passport/visa controls and regulations. Table 3.12 shows the distinction of the terminals of Stockholm-Arlanda airport [37].

Terminal	Type	Remarks
2	International	Mostly used by low-cost airlines
3	Domestic	Serves smaller aircraft from small domestic airports
4	Domestic	The main domestic terminal (SAS operates here)
5	International	The main international terminal, including non-Schengen arrivals

Table 3.12: The terminals of Stockholm-Arlanda airport

This table, together with the tables in Appendix III, gives an impression of how the terminal is defined given the airline and the airport from which the aircraft departed. Usually, knowledge of this airport is enough to define the terminal. For example, arrivals from non-Schengen countries will park at terminal 5. Arrivals from regional domestic airports (Växjö, Karlstad, Halmstad, Mora, etc) will park at terminal 3. A flight of SAS from Gothenburg will arrive at terminal 4. If there is more than one terminal for a corresponding airport, an intersection with the set of terminals for the specific airline is applied. For example, flights from Budapest can arrive at terminal 2 or 5. So, if the flight is operated by a low-cost airline like Norwegian, it will arrive at terminal 2 and if it is operated by Malév (the national carrier of Hungary) it will arrive at terminal 5.

3.1.3.3 The Runway Usage Pattern

At this point, the Airport Operations have defined the time and the terminal vertex of the parking area. The last input needed by the taxiway routing process is the terminal vertex of the runway exit. No matter if the aircraft is arriving or departing, the position where it will exit or enter the runway must be generated. An airport has one or more runways and a runway has usually more than one exit. In a non-trivial case, there will be a substantial number of runway exits to choose from. For example, Stockholm-Arlanda airport has 3 runways with (10 + 4 + 8) = 22 exits (see maps in Appendix I). The runways can be parallel or have different orientations or there can be a combination of parallel and non-parallel runways [41]. The latter case can obviously be met only in airports with more than two runways and is the case with Stockholm-Arlanda airport. The choice of which runway(s) to operate at a given time depends on different factors like the load of traffic and the weather conditions (wind direction, visibility), as well as safety and other regulations like the hour of the day – at night for example the aircraft must use airways to land or take-off that are not close to residential areas, in order to avoid noise disturbances [5], [8].

Based on these factors, a number of different runway usage patterns is created and at each time one pattern is chosen and applied. As of October 2011, the patterns for Stockholm-Arlanda are displayed on table 3.13 [5].

Pattern	Arrivals	Departures	Traffic load
1	01L / 01R	01L / 01R	High
2	19L / 19R	19L / 19R	High
3	01L / 01R	01L / 08	High
4	26	19R	Medium
5	19R	08	Medium
6	01L	08	Medium
7	26	01L	Low
8	01R	01L	Low

Table 3.13: The current runway usage patterns of Stockholm-Arlanda airport

Depending on the pattern used at a given time, the runway exits and entrances are determined. Obviously these are exclusive subsets of the set of all the runway exits / entrances; the same way cannot be used for the exit and entrance of aircraft at the same time. Once again, the approach followed in the present work, regarding the choice of a runway exit / entrance from the available subsets defined by the used pattern, will be presented in the next chapter.

3.1.3.4 Overview

The grouping and description of the airport operations that constitute the environment of the taxiway routing process completes the presentation of the problem entities. Before closing this section and proceeding to the formulation of the mathematical model, the following figure (3.14) summarizes the discussions and definitions up to this point and serves as an overview of the subject of the present work.

The Taxiway Routing System has the main focus; it is composed by the taxiway network and the aircraft moving on it. The aircraft appear, dynamically transform and disappear from the system within a moving time-frame. The airport operations define when (arrival or departure time) and where (source vertex) an aircraft appears in the system, as well as where it will finish its route and disappear (destination vertex), thus they provide the input to the system. The model of the system acts as the mathematical abstraction where the optimization algorithm(s) will be applied in order to propose a routing solution to the aircraft. When each aircraft reaches its destination, it has recorded actual values for the metrics – taxi time, hold time and speed changes. These are the output of the system and their evaluation will define the quality of the routing solution proposed by the respective algorithm and test the performance of the algorithm to the problem of 4D Taxi Routing on Ground.

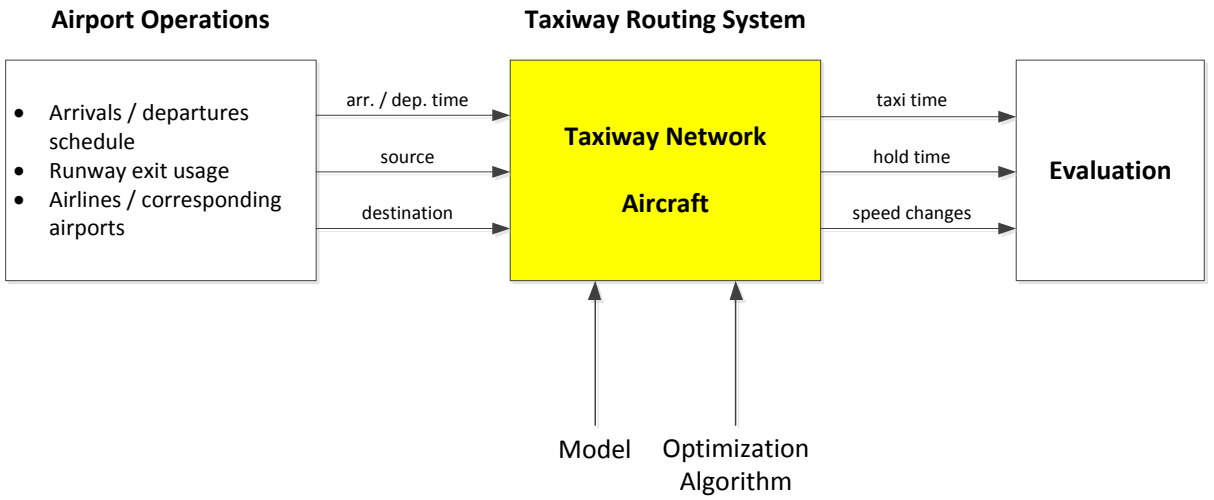


Figure 3.14: Overview of the Taxiway Routing System

3.2 The Model

The problem of 4D Taxi Routing on Ground is about optimizing a set of objectives applied on the Taxiway Routing System. According to our conception of the Taxiway Routing System analyzed in the previous sections of the present chapter, this system is composed of the Taxiway Network and the Aircraft.

3.2.1 The Taxiway Network

The Taxiway Network is modeled as a *time-dependent*, *labeled*, *bimodal* and *directed* graph. The notion of time-dependence on the graph is expressed with a non-static weight function $w_{ij}(t)$ on its edges, where time is assumed to be discrete, expressed as an integer time unit (for example seconds). The graph is defined as follows:

$$\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{A}, \boldsymbol{c_V}, \boldsymbol{c_E}, \boldsymbol{c_A})$$

where the set

$$\boldsymbol{V} = \{v_i\}, \qquad 1 \leq i \leq N$$

is the set of vertices of the graph and corresponds to certain positions on the Taxiway Network; the runway exits, the parking stands and the intersections of taxiways (crossings, merges and splits). The labeling function of the vertices is therefore:

$$\boldsymbol{c_V}: V \to L, \qquad where\ \ L = \{r, p, i\} \quad (\mathbf{1})$$

corresponds to 'runway exit', 'parking stand' and 'intersection' respectively. The vertices of the graph are interconnected via edges which correspond to uninterrupted taxiways, in the sense that an aircraft can traverse them without implications once it is permitted to. The set of edges is defined as:

$$\boldsymbol{E} = \{e_{ij}\} \subseteq V * V, \qquad 1 \leq i, j \leq N, \qquad i \neq j$$

and $$\boldsymbol{c_E}: E \to [\, l, sp, ws, md]$$

is the static feature function, returning for each edge e_{ij} the length l_{ij}, the maximum speed sp_{ij} and wingspan ws_{ij} allowed on the respective taxiway, as well as the supported modes for this edge, a subset of {arrivals, departures} - where $md_{ij} \equiv md(e_{ij})$ and $md_{ji} = \{-m : m\ in\ md_{ij}\}$ - defined as follows:

- $md_{ij} = \{arr\}$ only arrivals from i to j (one-way one-mode edge)
- $md_{ij} = \{dep\}$ only departures from i to j (one-way one-mode edge)
- $md_{ij} = \{arr, dep\}$ arrivals and departures from i to j (one-way two-modes edge)
- $md_{ij} = \{arr, -dep\}$ arrivals from i to j, departures from j to i (two-way two-modes edge)

The set of possible next edges that can be followed after an edge e_{ij} depends also on the supported modes for this edge. If $md_{ij} = \{arr\}\ or\ md_{ij} = \{dep\}$, there is only one mode supported, so there is one set of next edges. Otherwise, there are two sets of next edges, one for each mode. This can be modeled as the transition function:

$$\boldsymbol{next_E}: (E, md) \rightarrow E^*$$

Each set can have one or more next edges, but there can also be no next edges – if the edge leads to a terminal point at the given mode. Data examples of the static feature function $\boldsymbol{c_E}$ and the transition function $\boldsymbol{next_E}$ have already been displayed on tables 3.6 and 3.7 respectively.

3.2.2 The Aircraft

The Aircraft are the agents moving on the graph. The set of aircraft, which has no predefined size restriction, and the static feature function for each aircraft are defined as follows:

$$\boldsymbol{A} = \{a_k\}, \qquad k \geq 1$$

$$\boldsymbol{c_A}: A \rightarrow [ws]$$

The function c_A returns the dimensional features of each aircraft a_k, i.e. its length l_k, wingspan ws_k and weight wg_k. For the present model only the wingspan is considered, even though the model can be extended to include the other features as well (see the reasoning of section 3.1.2). In that case, the feature function of the edges would have to be extended accordingly. For example, if the weight feature is added as the output of function $\boldsymbol{c_A}$, a *maximum* weight feature would have to be added as the output of function $\boldsymbol{c_E}$ and the latter would restrict the allowance of aircraft on the edges based on the comparison with the former.

The speed, state, mode and position of an aircraft are dynamic features, because they are functions of time. Their definitions are presented below:

- $\boldsymbol{speed}: (\mathrm{A}, \mathrm{t}) \rightarrow \mathbb{R}_+$
- $\boldsymbol{state}: (\mathrm{A}, \mathrm{t}) \rightarrow \{0, 1, 2, \dots, 8\}$
- $\boldsymbol{mode}: (\mathrm{A}, \mathrm{t}) \rightarrow \{\mathrm{arr}, \mathrm{dep}\}$
- $\boldsymbol{position}: (\mathrm{A}, \mathrm{t}) \rightarrow E \cup \{r, p\}$ $(\mathbf{2})$

Based on these definitions, we can further elaborate:

- The speed $speed(a_k, t)$ of an aircraft is a real number greater than or equal to zero.
- The state $state(a_k, t)$ of an aircraft is an enumerative type from 0 to 8, according to the discussion of section 3.1.2. The aircraft can be considered as a Finite State Machine, given the specified number of states and the transition table described also in the same section.
- The mode $mode(a_k, t)$ of an aircraft is derived implicitly from its state. States 1, 2 and 3 correspond to arriving mode, states 5, 6 and 7 to departing mode, while states 0, 4 and 8 do not correspond to any mode, since these are the "non-taxiway" states.
- The position $position(a_k, t)$ of an aircraft while it is taxiing (or being on hold on the taxiway) – states 2, 3, 6, 7 – is defined by the current edge and the position on this edge, so it can be modeled as a duple $\{e_{ij}, pos\}$ or alternatively as a triple $\{v_i, v_j, pos\}$, where pos is the current distance from vertex v_i.

- The position of an aircraft while it is parked or pushed back or waiting at a runway exit to start taxiing – states 1, 4, 5 – is just the vertex of the corresponding parking stand or runway exit, so (2) in this case can be defined using (1) as: $position(a_k, t) = v_i$ where $\{c_{v_i} = r \mid p\}$.

3.2.3 The Weight Function

The weight of an edge has a "3-level" definition. On the first level, the weight is static and depends on the features of the edge, namely its **length** and **maximum speed**, so the weight actually stands for the minimum time to traverse the edge. On the second level, the weight is still static but also depends on the aircraft that is about to traverse the edge. The **wingspan** of the aircraft must not exceed the maximum wingspan allowed on this edge and the **mode** of the aircraft must be supported by the edge. On the third level the weight becomes dynamic and depends on the current **occupancy** of the edge. The occupancy of an edge can be defined as follows:

$$\boldsymbol{occupancy}: (\mathrm{E}, \mathrm{t}) \rightarrow \{true, false\}$$

$$\boldsymbol{occupancy_{ij}(t)} = \begin{cases} true, & if\ \exists a_k \in A:\ position(a_k, t) \text{ in } \{e_{ij}, e_{ji}\} \\ false, & otherwise \end{cases} \quad (\mathbf{3})$$

In the present model we assume that an occupied edge cannot be entered by another aircraft. This restriction can be relaxed – and actually it is relaxed in the realization of the model (see chapter 4) - in order to allow an aircraft a_k enter an edge e_{ij} occupied by another aircraft a_l, if both aircraft are following the same direction and if the position of a_l in the edge (i.e. the distance from vertex v_i) is greater than a specified minimum safety distance.

Having defined the occupancy function, the definition of the edge weight function can follow. The first two levels are merged into one to provide the definition of the *static* weight:

$$\boldsymbol{stWeight}: (\mathrm{E}, \mathrm{A}) \rightarrow \mathbb{R}_+ \cup \mathrm{Inf}$$

$$\boldsymbol{stWeight_{ij}(a_k)} = \begin{cases} {l_{ij}}/{sp_{ij}}, & if\ ws(a_k) \leq ws_{ij}\ \wedge md(a_k)\ \in\ md_{ij} \\ \infty, & otherwise \end{cases} \quad (\mathbf{4})$$

Based on the static weight (4) and the occupancy function (3), the *dynamic* weight concludes this section:

$$\boldsymbol{dnWeight}: (\mathrm{E}, \mathrm{A}, \mathrm{t}) \rightarrow \mathbb{R}_+ \cup \mathrm{Inf}$$

$$\boldsymbol{dnWeight_{ij}(a_k, t)} = \begin{cases} stWeight_{ij}(a_k), & if\ not\ occupancy_{ij}(t) \\ \infty, & otherwise \end{cases}$$

3.2.4 Routing Metrics and Objectives

The problem of 4D Taxi Routing on Ground can now be formulated. There is a graph described in section 3.2.1 and aircraft – described in section 3.2.2 as finite state agents - moving on the graph. The edges of the graph are supplied with dynamic weights which are functions of the edge features, the aircraft features and time – described in section 3.2.3. The input to this model consists of:

1. A scheduled time that each aircraft enters the graph:

$$\boldsymbol{start}: A \to t, \qquad \boldsymbol{start_k} = start\ of\ routing\ of\ aircraft\ a_k$$

2. A source vertex:

$$\boldsymbol{source}: A \to V, \ \boldsymbol{source_k} = \begin{cases} v_i, & such\ that\ \ c_{v_i} = r, \quad if\ md(a_k) = arr \\ v_j, & such\ that\ c_{v_j} = p, \quad if\ md(a_k) = dep \end{cases}$$

3. A destination vertex:

$$\boldsymbol{destination}: A \to V, \ \boldsymbol{destination_k} = \begin{cases} v_j, & such\ that\ c_{v_j} = p, \quad if\ md(a_k) = arr \\ v_i, & such\ that\ c_{v_i} = r, \quad if\ md(a_k) = dep \end{cases}$$

The problem of 4D Taxi Routing on Ground is the assignment of a route for each aircraft a_k:

$$\boldsymbol{route_k} = [\{\boldsymbol{source_k}, \boldsymbol{v_1}\}, \{\boldsymbol{v_1}, \boldsymbol{v_2}\}, \dots, \{\boldsymbol{v_{n-1}}, \boldsymbol{v_n}\}, \{\boldsymbol{v_n}, \boldsymbol{destination_k}\}]$$

which is consistent according to the structure and constraints of the graph and whose execution will evaluate the following metrics:

$$\boldsymbol{finish}: A \to t, \qquad \boldsymbol{finish_k} = \sum_{\mathrm{e_{ij}} \in \mathrm{route_k,t}} dnWeight_{ij}(a_k, \mathrm{t})$$

1. $\boldsymbol{taxitime}: A \to \mathbb{N}, \ \ \boldsymbol{tt(a_k)} = finish_k - start_k$
2. $\boldsymbol{holdtime}: A \to \mathbb{N}, \ \boldsymbol{ht(a_k)} = |\{\ start_k \le t_i < finish_k \mid speed(a_k, t_i) = 0\ \}|$
3. $\boldsymbol{speedchanges}: A \to \mathbb{N},$

$$\boldsymbol{sc(a_k)} = |\{\ (t_i, t_{i+1}),\ start_k \le t_i < finish_k \mid speed(a_k, t_i) \neq speed(a_k, t_{i+1})\ \}|$$

A routing is *feasible* if it is consistent and additionally the aircraft does not enter an occupied edge, thus if $f_k < Inf$. A feasible routing is *optimal* if it satisfies the problem objectives, thus minimizing the overall taxi time, hold time and speed changes:

$$min\left[\sum\nolimits_{a_k \in A} \boldsymbol{tt(a_k)}\right], \ min\left[\sum\nolimits_{a_k \in A} \boldsymbol{ht(a_k)}\right], \ min\left[\sum\nolimits_{a_k \in A} \boldsymbol{sc(a_k)}\right]$$

Chapter 4 **Model Realization**

The present chapter is an overview of the classes and algorithms built as a realization of the concepts of the model described in the previous chapter. From a procedural aspect, the whole routing process follows the sequence displayed on figure 4.1.

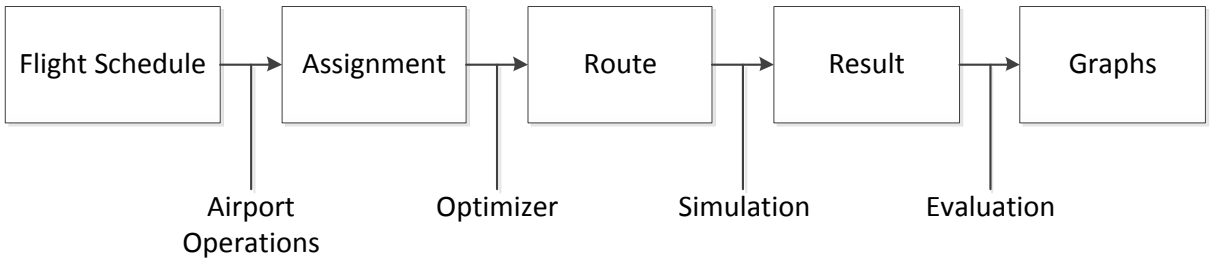


Figure 4.1: Overview of the routing process

The **flight schedule** of an airport is the "driving force" of the routing process. Each arrival or departure triggers a new routing on the taxiway. A sample of a flight schedule for Stockholm-Arlanda is already displayed on table 3.11; here we use one of its rows as an example:

09:05	BA	776	LHR	321	A

At 09:05 flight BA-776 is expected to arrive. The **airport operations** will define the **assignment** of the two terminal points on the taxiway – source and destination – of the route for this aircraft. Suppose that at 09:05 runway usage pattern 4 is applied (for runway usage patterns see table 3.13). This means that arrivals take place on runway 26. Based on the airport maps and runway exit specifications, when an aircraft lands on runway 26, it can follow exit X2 or X3. X2 is randomly chosen for this example. Flight BA-776 comes from London Heathrow, which means that the aircraft is about to park on terminal 5. Terminal 5 is serviced by several parking stand areas, for this example P6 is also randomly - and depending on its occupancy – chosen. So now the source-destination pair is generated: [X2, P6].

With arrival time, source and destination already known, an **optimization** algorithm can be used to generate the whole **route** – a sequence of edges or vertices on the taxiway – that the aircraft will follow towards its destination. The resulting route, corresponding to the shortest path between X2 and P6, is marked with thick green arrows on figure 4.2 and it is the following:

Route: [X2(11) 61 62 63 91 92 94 95 97 98 100 P6(158)]

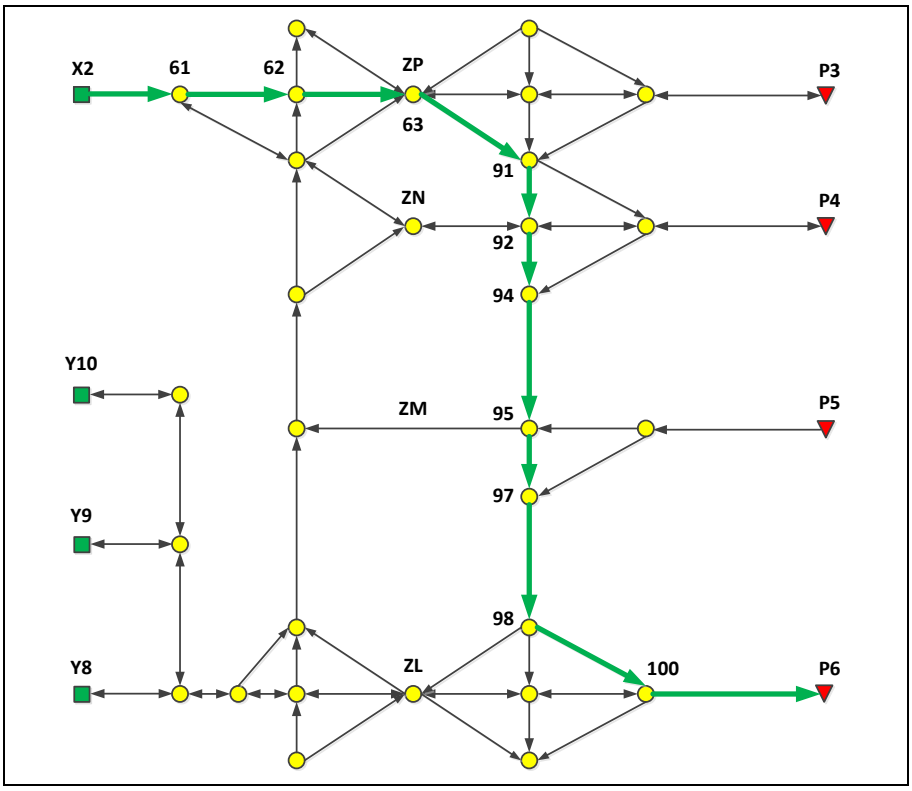


Figure 4.2: The arrival route followed by flight BA-776

The aircraft will taxi according to the assigned route. At the time that it exits the taxiway, the **results** for this routing are recorded; taxi time, hold time, speed changes. For this example, the values of the routing metrics were: taxi time = 150 seconds, hold time = 0, speed changes = 1. The total distance covered was 831 meters, a rather short routing.

This procedure is followed for every aircraft arriving or departing. At the end, each flight is just a set of values for the routing metrics. All these **evaluations** are gathered, aggregated and displayed in various **graphs**, so that conclusions about the performance of the optimization algorithms can be drawn. The classes that compose the realization of the model and the functionalities they support are presented in section 4.1. The routing algorithm that builds upon the class functionalities and binds the airport, the schedule and the optimization together is described in section 4.2. Finally, the implemented optimization algorithms close this chapter in section 4.3.

4.1 The Classes

The classes that compose the realization of the model are eight and their distinction is based on the three main entities defined in the previous chapter. The entity of the Taxiway Network is realized by a class named *AirportGraph*, which is composed of vertices and edges. This class actually contains all the information displayed on the graphs of figures 3.1 – 3.5 and 4.2. So, the distinction of the vertices in runway exits, parking stand areas and intersections is made in the class of the *AirportGraph*. The edges and the parking stands have their own structure and functionalities, so they are modeled as separate classes that compose the *AirportGraph*: *Edge* and *ParkStand*. In the class diagram of figure 4.3, these three classes are distinguished by having green labels.

The entity of the Aircraft is realized by two classes, labeled with orange color: the *AircraftType* which holds the static features of an aircraft (airline, model name, length and wingspan) and the *Aircraft* itself which inherits from the *AircraftType* and additionally keeps all the dynamic features and metrics (speed, position, state, taxi time, hold time, speed changes) as well as the commands that route an aircraft as methods (taxi, hold, line-up, park, take-off – see table 3.10). There is also an *Airport* class, labeled blue, which inherits from the *AirportGraph*, contains *Aircraft* and includes the entity of the Airport Operations, thus binding everything together.

Finally, there are two more classes that realize the driving force of the application, which is the flight schedule. The first is the *SchedLine* that contains the information for one flight, such as the flight number, the airline and type of aircraft, the arrival-or-departure flag, the origin or destination airport and the time the aircraft is scheduled to arrive or depart. The second is of course the *Schedule* class, composed of schedule lines, which resembles the entity of a whole flight schedule. The flight schedule classes are labeled with yellow color on the class diagram displayed below:

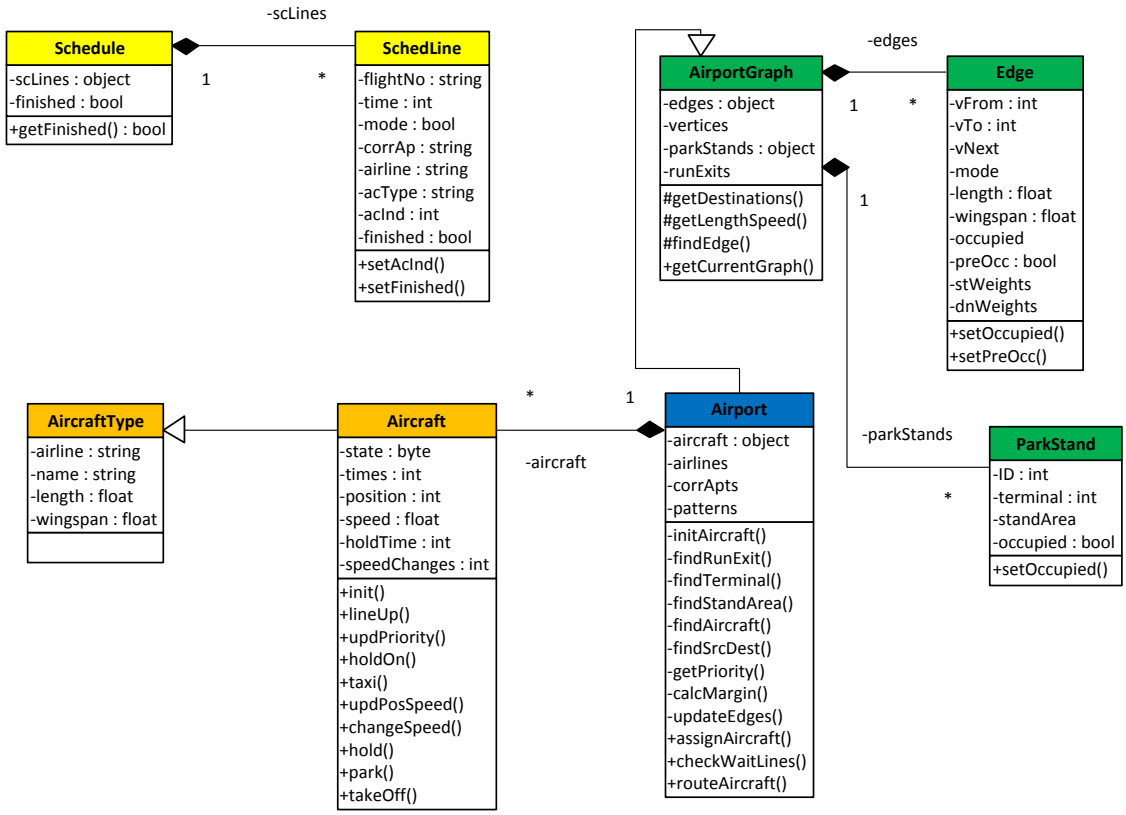


Figure 4.3: The class diagram

The above listings of properties and methods for the implemented classes are not exhaustive. There are still some more hidden or auxiliary features, but the purpose of this section is to present the essential functionality rather than getting into every detail. This functionality, the way it is structured and the design choices are developed in the coming section.

4.2 The Routing Algorithm

The routing algorithm is based on a specified airport, a flight schedule to be executed on this airport and an optimization algorithm (or a suite of algorithms) that decide on the 4D routes that the aircraft will follow on the taxiway. Everything must be interchangeable; the airport is just a set of data files with a given structure loaded as input before the routing starts, the schedule is also an input file with the only restriction being its compliance with the airport. Most significantly, the routing process must be independent of the optimization (see figure 4.1), it should just execute and evaluate the routings generated for each aircraft. A rough overview of the routing algorithm in pseudo-code is given on figure 4.4. The main procedures are emphasized in bold letters and analyzed right afterwards.

```
Execute Route
        Initialize airport; Load schedule
        Set time unit; Set current time t = 0
        While schedule is not finished loop
                If time for a new schedule line: Assign aircraft
                Check the waiting lines
                Route the aircraft currently on the taxiway
                t = t + time unit;
        End loop
End
```

Figure 4.4: Overview of the routing algorithm

The main conclusion regarding the routing process is that, after all initializations, it traverses time and iteratively creates “snapshots” capturing the situation on the taxiway with a constant period defined as an input parameter (time unit). This time unit should intuitively be a relatively small number of seconds, so that there is a frequent “refresh” of the taxiway situation. At every time point - every “time unit” seconds - and until all flights of the schedule are routed:

- If the time has come for the next scheduled arrival or departure, this routing is assigned to an aircraft which is then put on the waiting line (runway exit or parking stand area)
- If there are aircraft on the waiting lines from the previous iteration, the first of them can enter the taxiway
- For all aircraft currently routing on the taxiway, the position and speed for this snapshot are calculated and further decisions are made if necessary

4.2.1 Initialize the Airport

The airport initialization is the creation of an Airport object with all the data for the structural and operational features already described and analyzed. In this implementation there are two input files, one for the structural and one for the operational features of the airport. The first input file is used to create the Airport Graph object with the following properties (see the class diagram - figure 4.3):

- An array of vertices with the distinction of whether they are runway exits, parking stand areas or intersections
- A composite structure (an array of cells) for the runway exits, containing the runway they are attached to and the mode-direction pairs they support
- An array of Edge objects
- An array of Parking Stand objects

Each Edge object has a pair of vertices (vFrom, vTo) and one or two sets of next edges (vNext), depending on its supported modes, as well as a length, a maximum allowed speed and wingspan. The edges are initially not occupied and their static weights are equal to their dynamic weights and computed as functions of the mode, length and wingspan. As the present chapter is the realization of the entities and the model of chapter 3, the present discussion is based completely on the definitions of section 3.2. In the class constructor there are several checks performed for data consistency within an edge (the existence of vertices, the length and wingspan are positive numbers etc) and among the created objects (no duplicates).

Each Parking Stand object corresponds to a parking position on the apron. A parking stand belongs to a terminal and is part of a parking stand area, so it can be reached through a stand area terminal vertex of the Airport Graph. The vertex where an aircraft will finish its arrival routing is not always the same as the vertex where this aircraft will enter the taxiway again in order to start departure routing. For example, compare the maps A (arrivals) and C (departures) in Appendix I. There is parking stand - or gate - 10 which belongs to terminal 5 of Stockholm-Arlanda airport. An arriving aircraft aiming to park there will exit the taxiway from ZN (P4 on figure 3.1), but an aircraft parked there will enter the taxiway from ZM (P5 on figure 3.2) in order to depart. Apart from these properties, the implementation of a parking stand should also consider its occupancy, in order to route aircraft only to available parking stands at each time.

The Airport object inherits the structural features from an Airport Graph object and adds the operational ones using the second input file. The operational features, as described in section 3.1.3, are the following:

- A composite structure (an array of cells) for the airlines that operate on the specific airport, containing the 2-character IATA code (for example “LH” for Lufthansa, “SK” for SAS etc) for each airline and the set of terminals it uses
- A similar structure for the source / destination airports containing the 3-character IATA code for each airport and the set of terminals used for flights to and from this airport
- An array of runway usage patterns with the respective runways and directions used under each pattern (see table 3.13)

There is one last “piece” of information that can be added to the Airport object, if this information is available. An airport may have aircraft parked on its apron before the time that the schedule starts.

So, initializing the airport can also include the introduction of aircraft at certain parking stands (overnight parking positions), which are accordingly set as occupied. This means that a number of Aircraft objects are created given their types; their current state is set directly to 4 (parked) and their position is the stand area vertex that corresponds to the parking stand where they are located.

4.2.2 Assign Aircraft

The airport is initialized, possibly with a number of aircraft at overnight parking positions. A runway usage pattern is chosen to be the current and a time unit for the iterations is set. Then time starts running until the scheduled time for the first arrival or departure is reached. Apart from the time and the mode, a Schedule Line object has the airline, the source / destination airport and the aircraft type as properties and it has to be assigned to an aircraft. Supplied with all this information, *assignAircraft* calls the following methods:

1. ***findRunExit***: according to the current runway usage pattern and the mode (arrival, departure) of the schedule line to be assigned, a set of possible runway exits is created (see also section 3.1.3.3). A runway exit rx and an alternative rx' are chosen randomly.
2. ***findTerminal***: given the source airport and the airline, the terminal where the aircraft must be parked is found (see also section 3.1.3.2).
3. ***findStandArea***: first a list of unoccupied parking stands is produced, given the terminal. Then a stand area sa and an alternative sa' are chosen randomly according to the parking stands.
4. ***findAircraft***: given a terminal and an aircraft type, an aircraft of this type parked at a stand of this terminal is found.
5. ***findSrcDest***: calls methods 1-4. In case of an arrival, the source is a runway exit (*findRunExit*) and the destination is a stand area (*findStandArea*). In case of a departure, the source is a stand area where an already existing aircraft is parked (*findTerminal* - *findAircraft*) and the destination is a runway "entrance" (*findRunExit*). In both cases, there is a check performed whether a path between these two vertices exists and if it does not exist, the alternative vertices are also checked.

The outcome of this procedure is to generate a source-destination pair of vertices which fulfill the operational requirements of the given flight schedule line and for which there exists a valid path on the taxiway that connects them.

In case of an arrival, a new aircraft must be introduced to the system, so an Aircraft object is created (state = 0). In case of departures, an already existing aircraft parked on the apron (state = 4) is chosen. The source-destination pair is given to the optimization algorithm that produces the full route - including the respective speed vector – to be followed. The route is assigned to the aircraft and the start-routing time is set, so that the total taxi time can be computed when the aircraft reaches its destination. Finally, the aircraft is placed (lined-up) in its initial position, with its state set from 0 to 1 or from 4 to 5. Therefore, according to the table 3.10, procedure *assignAircraft* invokes the commands "*Init"* and "*Line-up"*.

4.2.3 Check Waiting Lines

While *assignAircraft* described in the previous section initializes aircraft and assigns scheduled arrival or departure routings to them, thus being responsible for the state transitions $[0 \rightarrow 1]$ and $[4 \rightarrow 5]$, *checkWaitLines* is responsible for the next step, which is to give the permission to the aircraft to enter the taxiway and start routing. So, *checkWaitLines* results in the state transitions $[1 \rightarrow \{1, 2\}]$ and $[5 \rightarrow \{5, 6\}]$.

Each terminal vertex of the Airport Graph is assumed to have a queue or waiting line for the aircraft that aim to enter the taxiway from this source. The notion is quite simple: when an aircraft a_k appears at a runway exit or a parking stand area ready to start routing, a method called ***getPriority*** is called to assign a priority to this aircraft on the waiting line of the specific vertex. Priority $p = 1$ means that there is no other aircraft ahead on the same waiting line, so the command "*Taxi*" is invoked, the aircraft state changes to 2 or 6 and the aircraft a_k starts routing at the next iteration. Priority $p > 1$ means that there are currently $(p - 1)$ aircraft ahead on the same waiting line, so the command "*Hold-on*" just increases the hold time of the aircraft a_k. Finally, when an aircraft leaves the waiting line and starts routing, the command "*Update-priority*" is invoked for all the others, decreasing their priority by 1.

4.2.4 Route Aircraft

Up to this point all the procedures described are pre-routing; first the airport is initialized and then time starts running and the scheduled arrivals and departures are assigned to aircraft, which in turn line-up until they can enter the taxiway. The most important part is obviously the actual routing of the aircraft which is described in the present section. The purpose of the *routeAircraft* procedure is to route each aircraft from its source to its destination via a sequence of taxiways with respect to the following assumptions / restrictions that attempt to make this routing as realistic as possible:

- The aircraft moves with a speed that must not exceed the maximum speed allowed on the edge it traverses.
- When an aircraft enters a new edge: if the edge has speed limit lower than the current speed of the aircraft, it must decelerate. If the edge has speed limit higher than the current speed of the aircraft, it can accelerate. However, many speed changes must be avoided according to the discussion of the first chapter.
- There is a maximum acceleration and deceleration which is realistically acceptable. These are supplied as input parameters to the routing algorithm.
- The movement of an aircraft is considered to be linear, so the distance covered is equal to the product of speed (v) and time (dt), $\boldsymbol{ds = v * dt}$, and the relation between speed and acc-deceleration (a) is $\boldsymbol{dv = a * dt}$. For example, suppose that an aircraft is accelerating from $v_o = 6\, m/sec$ to $v_1 = 10\, m/sec$ and that the maximum acceleration allowed is $a = 2\, m/sec^2$. This means that the speed change will take $\boldsymbol{dt = \frac{v_1 - v_0}{a}} = 2$ seconds during which the aircraft will cover a distance of $\boldsymbol{ds = \frac{v_0 + v_1}{2} * dt} = 16$ meters.
- The restriction of entering an occupied edge is relaxed in the following manner: if an aircraft a_k is about to enter an edge e_{ij} with direction $i \rightarrow j$ occupied by another aircraft a_m which moves in direction $j \rightarrow i$, then obviously a_k must stop and wait for the edge to become free,

otherwise there will be a frontal crash or a mutual blocking (deadlock) situation. However, if the edge is occupied by an aircraft a_m which moves in the same direction $i \rightarrow j$, then a_k can enter, as long as the distance between the two aircraft is greater than a minimum safety distance, supplied to the routing algorithm as an input parameter (like time unit and maximum acc-deceleration).

- Generally when an aircraft follows another aircraft, the distance between them must always be greater than the minimum safety distance. The following aircraft must therefore adapt its speed to the speed of the preceding one.
- When two aircraft are about to meet at an intersection (a crossing or a merging), priority is given to the one that arrives first at this point. The other aircraft must decelerate or hold.
- No overtaking is allowed.

Under these assumptions and at each time point a new snapshot of the situation on the taxiway is created by executing the procedure *routeAircraft* for each aircraft that is located on the taxiway at that time point. This procedure consists of the following steps:

1. Method ***calcMargin*** is called, which calculates the time margin from the current position of an aircraft until it reaches an obstacle, which might be a preceding aircraft or an occupied edge.
2. Depending on the margin and the current state of the aircraft, the appropriate command is given: "*Change speed*": $[2 \rightarrow 2]\ or\ [6 \rightarrow 6]$
   "*Hold*": $[2 \rightarrow 3]\ or\ [6 \rightarrow 7]$
   "*Taxi*": $[3 \rightarrow 2]\ or\ [7 \rightarrow 6]$
3. As a result of steps 1 and 2, the position and speed of the aircraft are updated to reflect what has happened since the last snapshot. This is command "*Update-position-speed*".
4. If the aircraft reached its destination, command "*Park*" or "*Take-off*" is invoked, the state is changed to 4 or 8 and the aircraft is not considered anymore, because it is out of the taxiway.
5. If the aircraft is still on the taxiway and depending on its new position, the edges of the graph are updated (set occupied or unoccupied) using method ***updateEdges***.

This is an overview of the routing algorithm. Certain inefficiencies have been found while running the simulations and they will be presented in chapter 5. The presentation of the optimization algorithms can now follow.

4.3 The Optimization Algorithms

The optimization algorithms implemented for the present work were an iterative version of Dijkstra's Shortest Path Problem algorithm [39] combined with a linear programming formulation [34]. The algorithms are not "mixed"; they are rather used in a complementary way and a sequential ordering. Dijkstra's algorithm is used for generating a shortest path on the taxiway network graph between the two terminal vertices. This path is a set of edges and each edge has a maximum speed allowed. The linear formulation is applied on this "speed vector" in order to find a balance between two objectives:

minimize taxi time and minimize speed changes. A more detailed presentation of both algorithms is given in the following sections.

4.3.1 The Iterative Version of Dijkstra's Algorithm

The algorithm of Dijkstra is "*a graph search algorithm that solves the single-source shortest path problem for a graph with nonnegative edge path costs, producing a shortest path tree*" [39]. The algorithm starts from a given source vertex of the graph and finds the minimum cost path between this vertex and every other vertex of the graph. The detailed presentation of how Dijkstra's algorithm works is out of the scope of the present work, since this algorithm is well-studied and extensively referenced. The input to Dijkstra's algorithm is a graph and a source vertex. Optionally, a destination vertex can be given as an input and in that case the algorithm is slightly modified in order to stop once it assigns a final cost to this destination vertex. This is the case in our implementation, since the source-destination pair is already known and the purpose is to find the minimum cost path that connects the two vertices.

The Airport Graph class provides a method called ***getCurrentGraph***, which returns a graph suitable for algorithms like Dijkstra's by keeping only the arrival or the departure "component" of the whole graph. For example, suppose that the airport graph is the one displayed on figure 3.3. A valid path for an arriving aircraft should not include edges used only for departures, so the method ***getCurrentGraph*** will return the graph of figure 3.4, which is unidirectional and where Dijkstra's algorithm can be applied without further modifications. This method can result to an even smaller graph by cutting off edges where the aircraft cannot enter because of wingspan or current occupancy restrictions. So, the method ***getCurrentGraph*** takes the total graph, the mode and wingspan of the aircraft and a static-or-dynamic decision flag as inputs and outputs the according component sub-graph.

Now Dijkstra's algorithm has all the input it needs in order to create a deterministic shortest path, which will be assigned to the aircraft. However, there is one feature of the model created for the problem of 4D Taxi Routing on Ground that can lead the application of the algorithm of Dijkstra to invalid results. This is the transition table mentioned in section 3.1.1.2, which dictates the next vertices (vNext) that can be reached from each edge. The problem can be illustrated with an example. Figure 4.5 shows the taxiway network graph of the southern part of Stockholm-Arlanda airport. This is actually the arrival component of the graph of figure 3.2. Parking stand areas P5, P8 and P10 and some edges which are only used for departures are not present here and all remaining edges are unidirectional. The flow of traffic is clearly directed from the runway exits to the parking stand areas.

In this example, we suppose that an aircraft has just arrived at runway exit Y1 and is about to park at a stand serviced by P13. Dijkstra's algorithm has a compatible graph (figure 4.5), a source (Y1) and a destination (P13). The question is what the output of the algorithm will be.

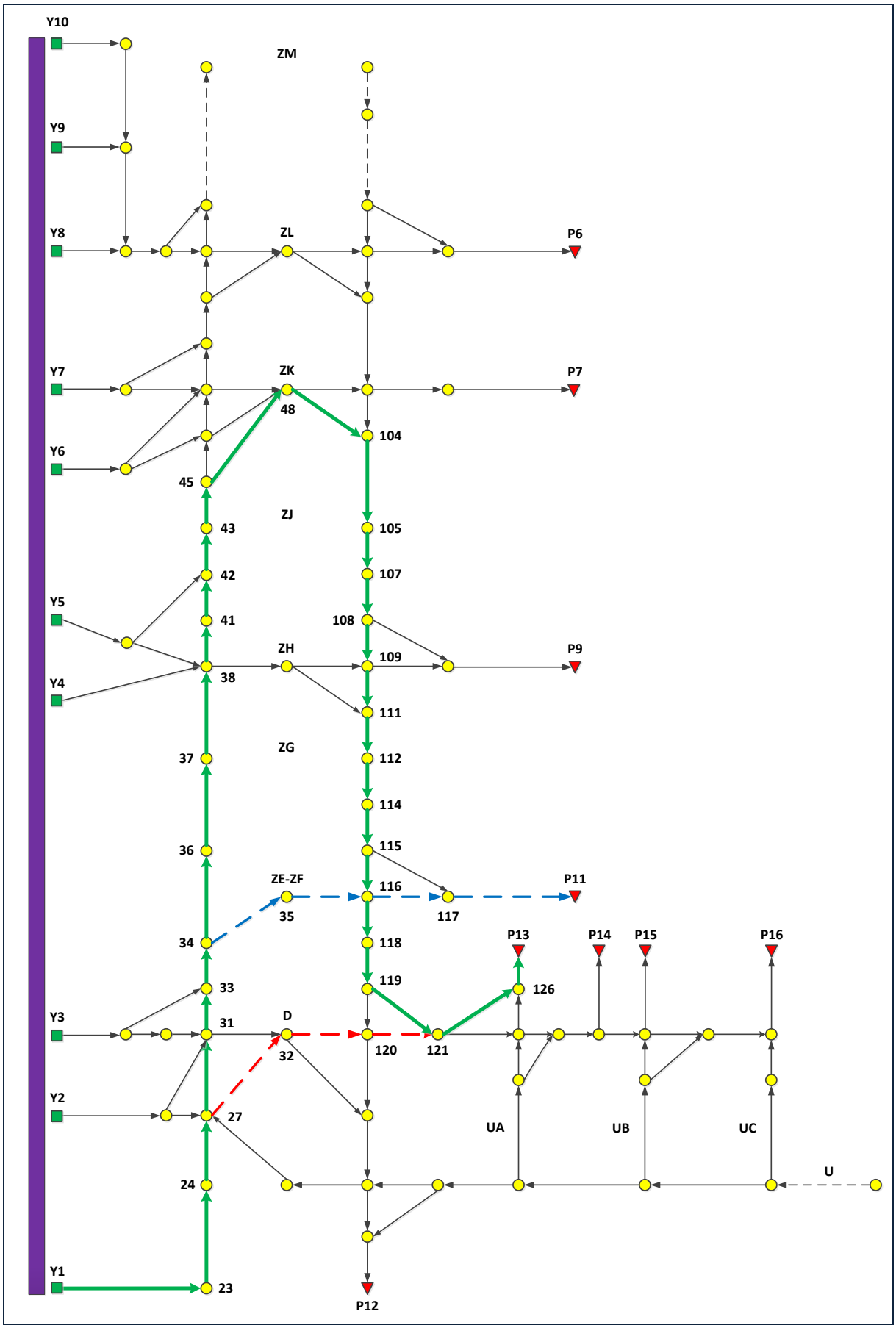


Figure 4.5: The arrival "component" of the southern part of Stockholm-Arlanda airport

The “pure” version of the algorithm does not consider any invalid transitions when searching the graph for shortest paths. An edge e_{ij} directed from v_i to v_j can always be followed within a path, so in this example the algorithm will produce this sequence of vertices:

[Y1 23 24 27 32 120 121 126 P13]

However, based on the airport taxiway maps, an aircraft reaching vertex 27 coming from vertex 24 is not permitted to turn to 32; it can only proceed straight to 31. The part of the first path that is not valid is marked with dashed red arrows.

The algorithm of Dijkstra iterates by creating a list of unvisited vertices from a current vertex which is then considered visited and assigned a final and minimal distance. The addition to the “pure” version, which created a first “tailored” version, was to intersect the list of unvisited vertices with the list of valid transitions from the current vertex coming from its previous one. For example, when starting the algorithm of Dijkstra from vertex Y1, it assigns a minimum distance to 23, 24 and 27 and creates the path [Y1, 23, 24, 27]. At the next iteration, the current vertex is 27 and its previous is 24. The list of unvisited neighbors contains vertices 31 and 32, according to the direction of the edges. This is intersected with the list of valid transitions from edge [24, 27] which contains only 31.

This way, the restriction of the list of neighbors that the algorithm of Dijkstra considers, seems to overcome the path validity problem. But it does not and this can be shown with a second example based on the same graph and scenario. After 27, the current vertex is 31, as discussed in the previous paragraph. The only possible transition from edge [27, 31] is 33; 32 could only be reached if the aircraft used runway exit Y3. Vertex 33 is followed by 34 and then there are two next vertices, both valid: 35 and 36. If vertex 35 is followed, the concluding path will be: [Y1, 23, 24, 27, 31, 33, 34, 35, 116, 117, P11]. This is a completely valid path and the optimal from Y1 to P11. However, within this path, vertex 116 is assigned a final distance and considered visited. This means that vertex 118 and the whole graph below it will never be reached and their resulting distances will be infinite.

The conclusion is that - on the one hand - the “pure” algorithm of Dijkstra cannot be applied to this type of graph, because it can generate invalid shortest paths. On the other hand, the restriction of applying a list with valid transitions can lead the algorithm to not exploring parts of the graph thus not finding paths that actually exist. The solution which was finally adopted was a third, “iterative” version of Dijkstra’s algorithm. This version assumes that a path from a given source to a given destination exists and, if the application of the “tailored” version (the one using the list of transitions) fails to find the path, it uses the second, third, etc vertex of the distance vector as intermediate sources and iteratively runs the algorithm until such a vertex is found from which the destination can be reached. In the case of the graph of figure 4.5 and the source-destination pair [Y1, P13], this intermediate vertex is 36: If vertex 36 is used as a source, a shortest path to P13 is found. The concatenation of $shortest\_path(Y1, 36)$ and $shortest\_path(36, P13)$ produces the following:

[Y1, 23, 24, 27, 31, 33, 34, 36, 37, 38, 41, 42, 43, 45, 48, 104, 105, 107, 108, 109, 111, 112, 114, 115, 116, 118, 119, 121, 126, P13]

This path is marked with thick green arrows. The worst case is when the path indeed does not exist and the “iterative” version runs *N* iterations of Dijkstra’s algorithm to conclude that. This imposes a polynomial overhead of one order of magnitude to the algorithm’s complexity.

4.3.2 The Linear Programming Formulation

The iterative version of Dijkstra's algorithm presented in the previous section produces a sequence of vertices or edges, which is the route to follow. Each edge is a taxiway or a fragment of a taxiway and has a speed limit according to the form of this taxiway, as already mentioned. It is also discussed that the speed of an aircraft at each time determines all the metrics for this aircraft (taxi time, hold time, speed changes), as well as its position at each time, which plays an important role on the routing of other aircraft that happen to taxi simultaneously. The speed of the aircraft makes the problem of Taxiway Routing a 4D one and much more complex than a variation of a shortest path problem.

Acknowledging the importance of the aircraft speed, a linear programming formulation is implemented in order to provide a "speed pattern" to the aircraft. This pattern can be considered as a set of directions, like for example: "taxi from point (vertex) 23 and up to 34 with a constant speed of 18 knots and then turn right to 35 with a speed of 12 knots…" So, this pattern - the output of solving the LP formulation - is just a vector with a speed value for each edge of the route that the aircraft will follow. The input to the LP solver is a same size vector with the maximum allowed speed for each edge, i.e. the set of constraints.

So, we have a sequence of edges, each with a (possibly different) maximum allowed speed. The purpose of this optimization is to assign a routing speed to each edge that is less than or equal to the speed limit and additionally finds a balance between the two conflicting objectives: minimize taxi time and minimize speed changes. There are two extreme solutions, each taking into account only one of the objectives. The first is to route the aircraft with the maximum speed at each edge; i.e. use the input vector as output. This solution minimizes taxi time, because the aircraft will taxi with the maximum allowed speed, but it ignores the speed changes and is actually not a realistic solution; an aircraft is not a race car. The second extreme is to route the aircraft with a completely steady speed, which is the minimum value of the input vector. This minimizes speed changes and produces a totally smooth routing but also a very slow one.

The idea for the LP formulation that was implemented was taken from section 2.5 of the book [24]. The LP problem described at that section is called "Ice-Cream All Year Round" and is about an ice-cream factory that wants to setup the production plan for the coming year based on sales predictions for each month. One solution is to produce just-in-time, but this may lead to great variations in the produced amount, which have significant costs. On the other hand, a completely flat production can lead to storage costs for the surpluses of some months. The purpose is to find a compromise that minimizes the cost both from changes in production and from storage of surpluses.

Chapter 5 **Evaluation**

The classes and algorithms described in chapter 4 were implemented for evaluation and testing in MATLAB. The airport used as a case study was Stockholm-Arlanda. According to the procedure described in section 3.1.1.2, the resulting taxiway graph for this airport has 22 runway exits, 16 parking stand areas and 130 intersections, which make a total of 168 vertices. The number of parking stands corresponding to the 16 stand areas is 88. The edges of the graph are 256, divided in the four edge types as:

Mode	Operation	#Edges
[1, 0]	Only arrivals	53
[0, 1]	Only departures	46
[1, 1]	Both modes in the same direction	69
[1, -1]	Both modes in opposite directions	88

Table 5.1: The number of edges per type

There are two matrices in Appendix II, displaying the time-distances for each pair of "runway exit – parking stand area", computed by iterative executions of the n-Dijkstra algorithm as presented in chapter 4. It is not obvious that the distance between - for example – runway exit Y1 and parking stand P1 is the same as the distance [P1, Y1] and the reason is that the graph is not the same in arriving and departing mode, so the shortest path can vary. Moreover, there are some runway exits used only for arrivals, so if they are considered destinations the distance is infinite. Concluding, it is possible that a path [A, B] exists, while the path [B, A] does not exist.

The airlines and origin/destination airports used for determining the terminal and therefore the parking stand area as a destination vertex for aircraft in arriving mode can be found in Appendix III, where the types of aircraft flying to Arlanda (according to the flight schedule of summer 2010) are also listed.

5.1 Simulation Setup

The simulation routings were based on the actual flight schedule of Stockholm-Arlanda airport for summer 2010. The initial thought was to execute this flight schedule for a chosen day, in order to test the model implementation and the optimization algorithms on realistic data. However, at the beginning of this chosen day the airport would not have any aircraft parked on its apron and this does not correspond to reality. Moreover, there would be a consistency issue if an early departure according to the schedule could not find any aircraft to be assigned to.

At most medium to large airports, including Arlanda which is also a hub of Scandinavian Airlines, there is a number of aircraft in overnight parking positions to depart at the next or one of the next days. A realistic airport implementation should consider this fact and since there was no way of knowing the number, fleet type and airline of aircraft parked at Arlanda at a given night in summer 2010, an alternative approach was to execute the flight schedule for two consecutive days; the first day would be a "preparation" day at the end of which there would be aircraft in overnight parking

positions for the execution of the "regular" day schedule. The days chosen were the 11th and 12th of August 2010 because they were the ones with the maximum number of flights, 628 in total, within the specified period. The routing parameters, as presented in chapter 4, were given the following values:

- The time unit for the execution loop would have to be small enough to keep a detailed control of the routing process. However, a time unit of 1 second would make the process very slow, as there would be 48*60*60 = 172800 loop iterations for a 2-day schedule. The final choice was a time unit of 10 seconds.
- The minimum safety distance was set to 100 meters.
- The maximum acceleration / deceleration of an aircraft were set to $3\ m/sec^2$. This is equal to approximately $0.3 * g$, where g - the g-force "*associated with an object is its acceleration relative to free-fall*" [44].

A number of 100 test runs of the 2-day schedule for different runway usage patterns were performed at first; not all of them were successful. The most common errors were the following (error rates are displayed on table 5.2):

1. A flight could not be assigned because there was no aircraft available at that time.
2. An aircraft entered an edge occupied by another aircraft.
3. An aircraft "crashed" with another aircraft, i.e. their distance at some point became less than the minimum safety distance.
4. Two aircraft fell in a deadlock from which they could not escape and as the time proceeded more aircraft were gathering behind them thus forming two increasing rows of holding aircraft.

The first 3 errors can be considered as problems deriving from the design choices of the routing algorithm. There were modifications / improvements made accordingly, but the occurrence of these errors was not ruled out completely. On the other hand, the deadlock is a rather useful outcome, as it shows the inefficiencies of the applied version of Dijkstra's algorithm to the problem of 4D Taxi Routing on Ground, which is one of the conclusions of the present thesis.

The outcomes of the first test runs of the routing process led to the finally adopted simulation setup. Different runway usage patterns lead to the utilization of different parts of the taxiway and the results vary accordingly. So there is a point in grouping the simulations by the runway usage pattern, as it can also be concluded from the results presented in the next section.

5.2 Presentation of the Results

The 2-day flight schedule for Stockholm-Arlanda airport was simulated for each of the eight runway usage patterns. A successful simulation of this schedule, which produces 628 taxiway routings, needs about 18-20 minutes to be completed on a PC with a Pentium® Dual-Core T4300 CPU running at 2.10GHz. An unsuccessful simulation obviously needs less than this time, since the execution stops when the error is produced. However, there is no error explicitly produced when the deadlock occurs because this is no violation of the routing rules; the aircraft just gather in rows and hold on the taxiway. The identification of this situation was made possible with the following assumption:

- There is a holding time limit for each aircraft, after which the aircraft is assumed to be blocked. This limit was set to 600 seconds = 10 minutes (of simulation time).
- The number of aircraft being blocked is constantly recorded. If at some time this number exceeds a limit, the simulation execution stops and the "deadlock" result is returned. This limit was set to 5 aircraft, so if more than 5 aircraft have been holding on the taxiway for more than 10 minutes, we assume that the blocking is permanent.

Under these assumptions and using the set of parameters as described above, a number of up to 20 successful or up to 40 total simulations were executed for each runway usage pattern. The aggregate outcomes are displayed in table 5.2.

	success	error	deadlock	total	succ %
pattern 1	4	6	30	40	10,00%
pattern 2	9	20	11	40	22,50%
pattern 3	20	14	0	34	58,82%
pattern 4	20	0	0	20	100,00%
pattern 5	12	27	1	40	30,00%
pattern 6	20	18	0	38	52,63%
pattern 7	20	3	0	23	86,96%
pattern 8	12	28	0	40	30,00%
	117	116	42	275	42,55%

Table 5.2: The aggregate outcomes of simulations per runway usage pattern

Considering the successful simulations, the aggregate results for the routing metrics are displayed in table 5.3.

	pattern 1	pattern 2	pattern 3	pattern 4	pattern 5	pattern 6	pattern 7	pattern 8
Average speed (m/sec)	7,81	7,71	6,97	6,84	6,79	6,50	6,85	7,28
Average speed (knots)	15,18	14,98	13,54	13,29	13,20	12,63	13,31	14,15
Average distance (m)	2978	2830	2013	1883	1692	1627	1867	2413
Average taxitime (sec)	381,39	367,28	288,98	275,46	249,14	250,30	272,69	331,56
Average # of sp. changes	3,52	3,41	2,62	2,60	2,34	2,26	2,56	3,31
Maximum hold time (sec)	207	407	124	148	169	206	193	159

Table 5.3: The aggregate results of successful simulations per runway usage pattern

Based on tables 5.2 and 5.3, some remarks can be made and some conclusions can be drawn, which provide reasoning on the correctness and effectiveness of the model and the algorithms.

1. Runway usage patterns 1 and 2 use runways 01L/19R and 01R/19L for mixed operations (arrivals and departures simultaneously). Runway 01R/19L – see maps B and D in Appendix I – is the most distant runway from the terminal area and every aircraft taxiing to or from this runway must be routed via taxiway U or W. These are long and straight taxiways, so an aircraft can taxi with a high speed. This fact is supported by the results: patterns 1 and 2 lead to the highest average speed and the longest average distance, followed by pattern 8, where runway 01R/19L is also used.
2. Taxiways U and W support both modes in opposite directions and runway usage patterns 1 and 2 use both modes. This combination, supported by the length of these 2 taxiways, leads aircraft to deadlocks, which justifies the fact that simulations resulted in deadlocks only under patterns 1 and 2. Pattern 8 also uses taxiways U and W but in only one mode, so it does not result in deadlocks. In general, patterns 1, 2 and 8 have the lowest success rates.
3. The runway usage patterns where the number of successful simulations reached the target number 20 were 3, 4, 6 and 7. These use one runway only for arrivals and one runway only for departures and the directions of exiting or entering these runways, which respectively define the sources and destinations of the taxiway routes, result to rare meetings of aircraft at intersections, thus minimal conflicts.
4. The average taxi time varies, depending on the runway usage pattern, from 250 seconds = 4:10 minutes to 380 seconds = 6:20 minutes. According to [8], typical taxi time at Stockholm-Arlanda airport is between 5 and 7 minutes, excluding delays on the runway exit in order to obtain taxi clearance or delays outside the parking stand area if another aircraft is pushed back at the same time. On average the results of the simulations are therefore comparable to reality.

In the following pages, the results are illustrated in the form of graphs. For each runway usage pattern there are 4 graphs:

1. The distribution of the number of aircraft according to their taxi time, grouped in ranges of 30 seconds
2. The taxi time per distance covered, including a tendency line whose angle with the horizontal axis corresponds to the average taxi speed ($y = a * x \rightarrow s = v * t$)
3. The hold time per distance covered
4. The speed changes per distance covered

In the third and fourth graph (hold time, speed changes) for each pattern there is no clear tendency or relation between the respective metrics and the distance covered, even though one can notice that longer distances are more likely to result to more changes of speed and longer hold times. In contrast to the taxi time, which naturally depends also on the distance to be covered, the other two metrics can be considered more as qualitative aspects of the routing solutions. A “good” or efficient solution can result to no speed changes and no hold time if an aircraft follows a route that avoids conflicts completely, no matter how long the covered distance is. A final discussion based on these evaluations will follow in the next chapter, which concludes this work.

5.2.1 Pattern 1 - Mixed Operations on Runways 01L and 01R

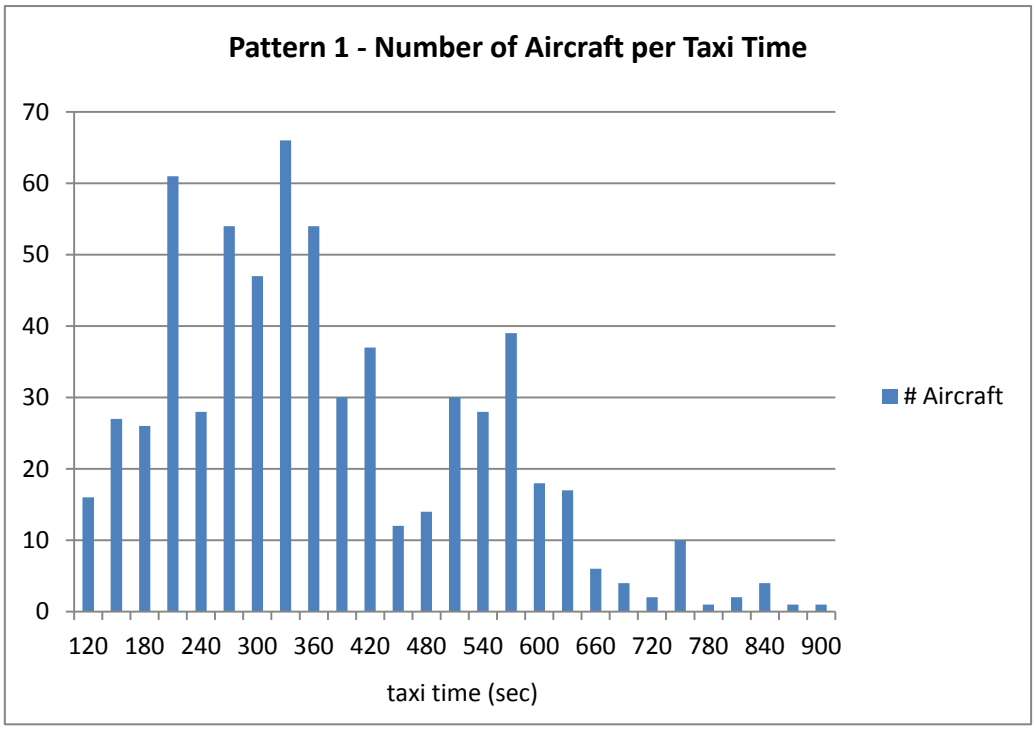


Figure 5.1: The distribution of aircraft per taxi time

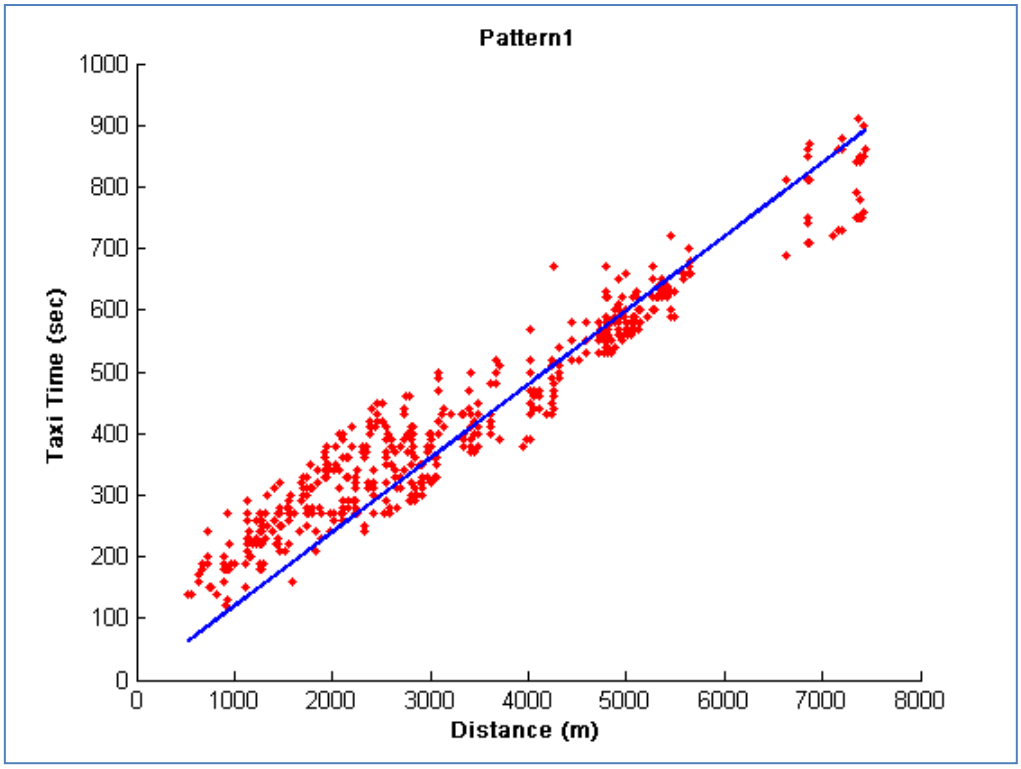


Figure 5.2: Taxi time per distance covered, including the tendency line

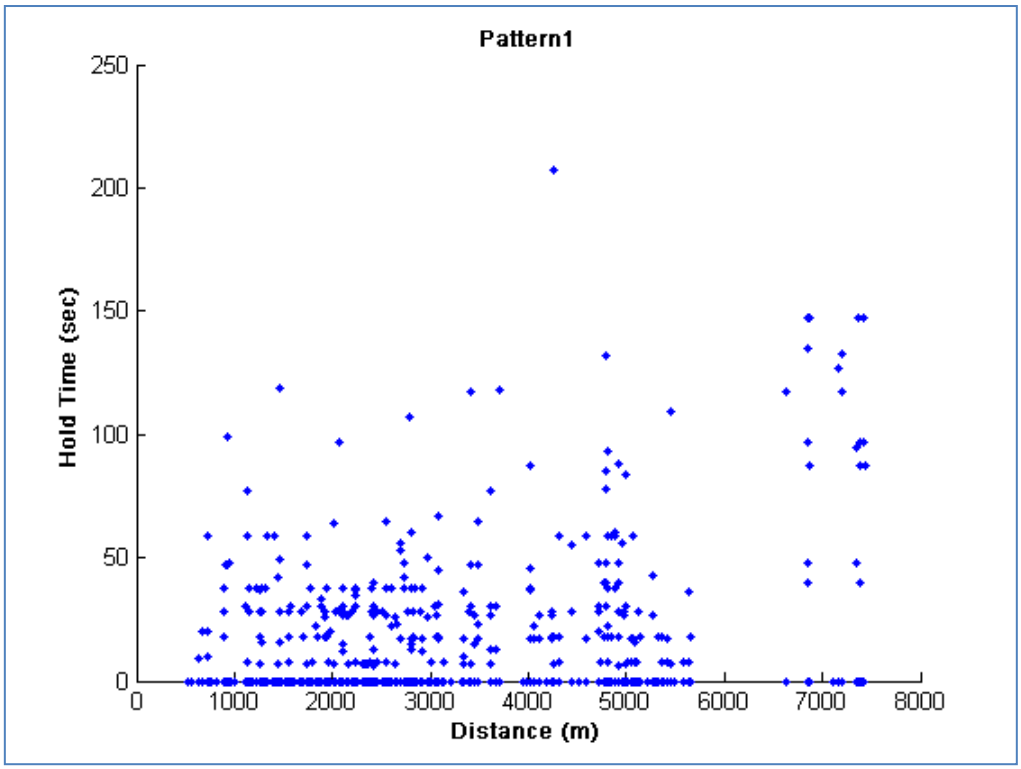


Figure 5.3: Hold time per distance covered

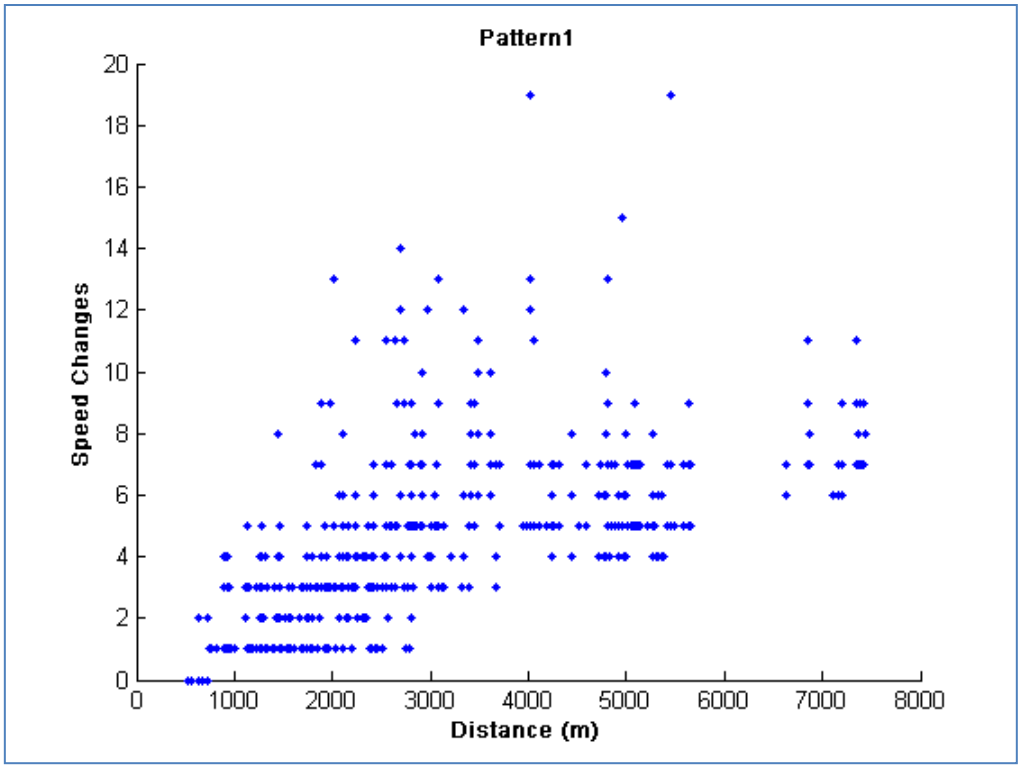


Figure 5.4: Number of speed changes per distance covered

5.2.2 Pattern 2 - Mixed Operations on Runways 19L and 19R

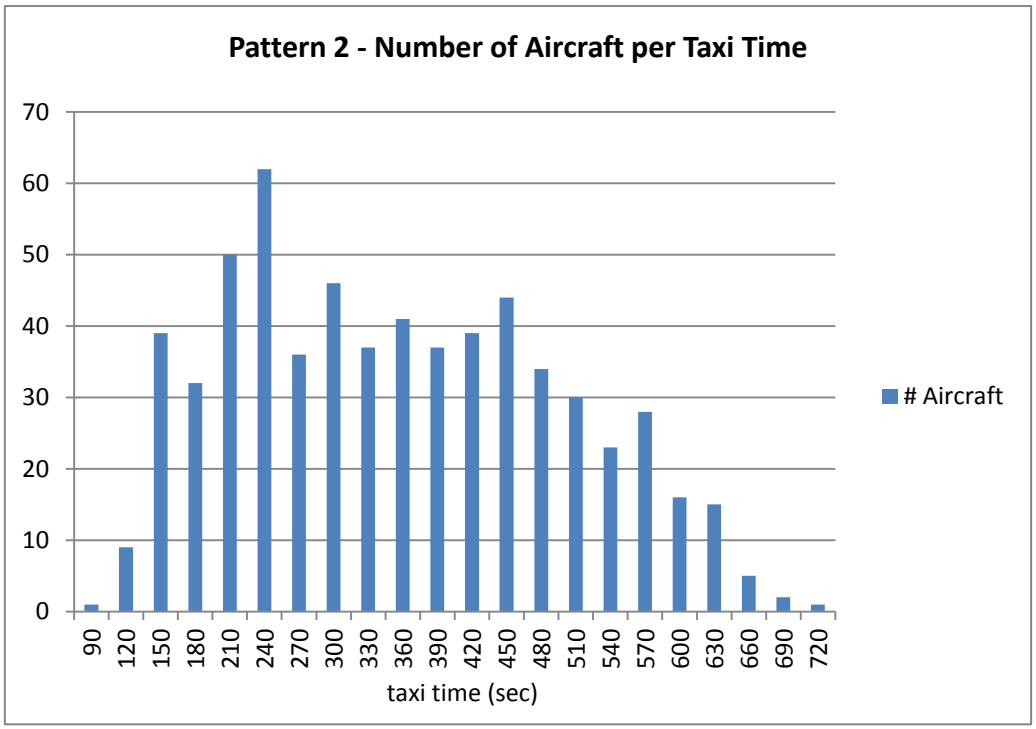


Figure 5.5: The distribution of aircraft per taxi time

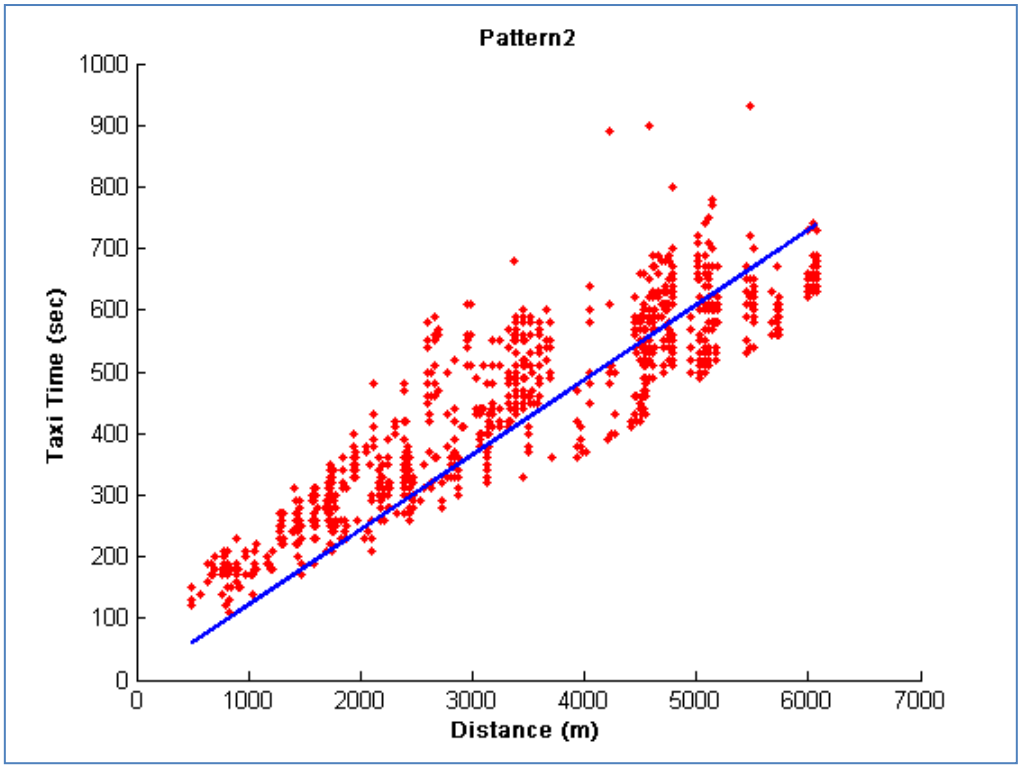


Figure 5.6: Taxi time per distance covered, including the tendency line

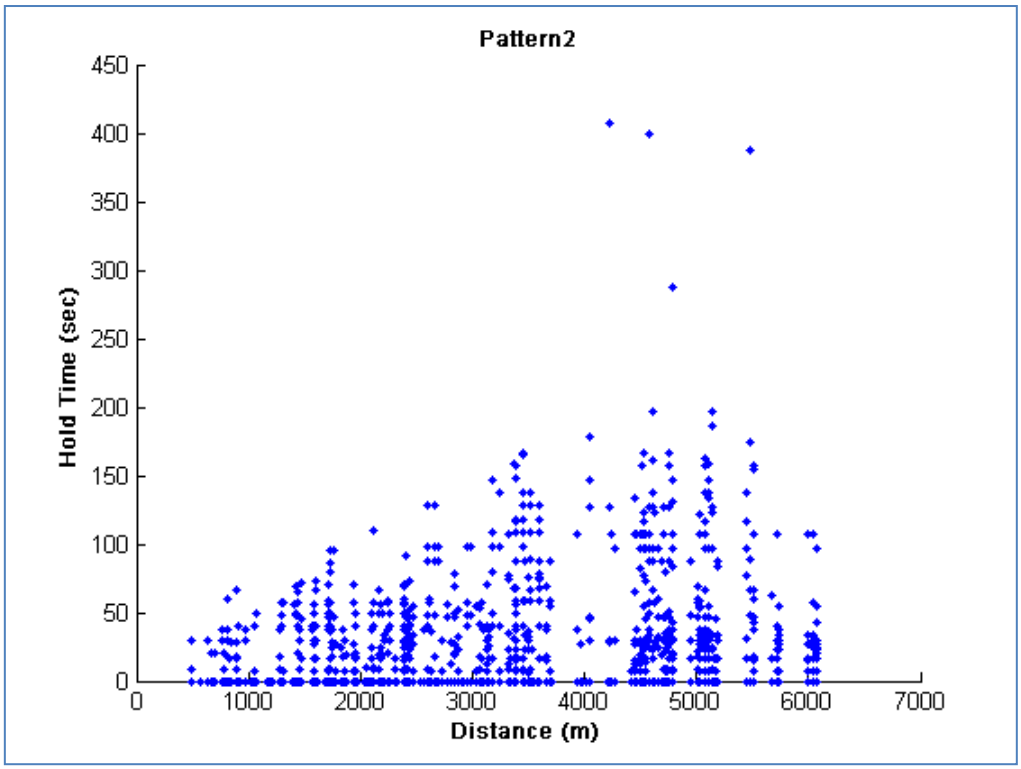


Figure 5.7: Hold time per distance covered

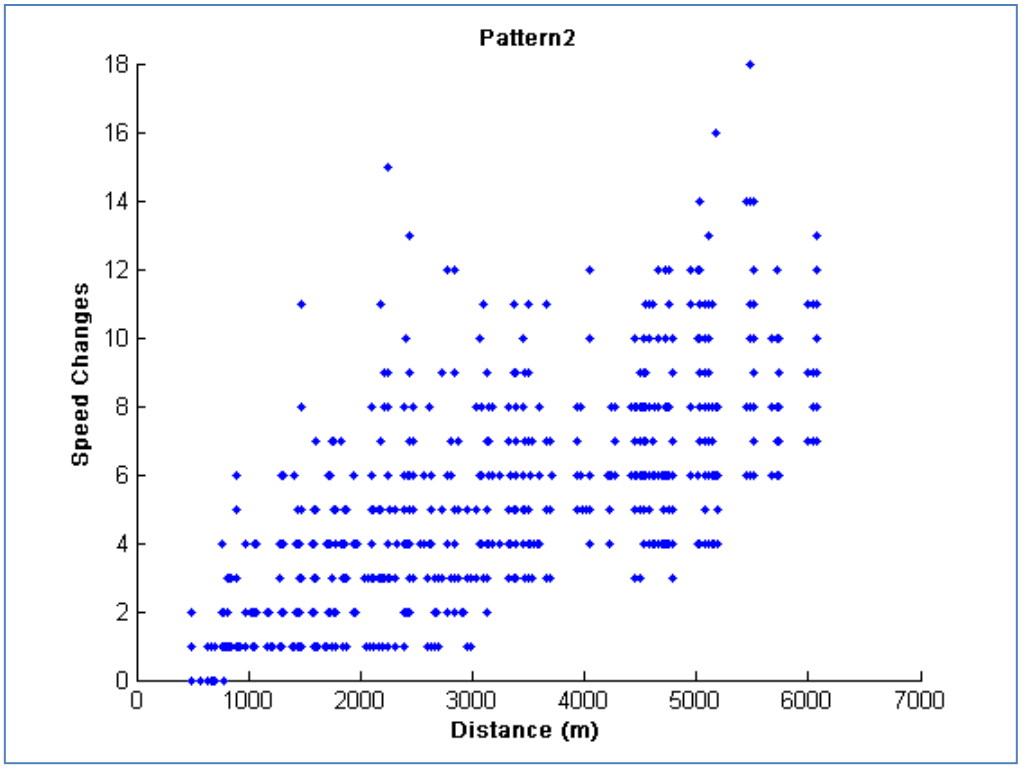


Figure 5.8: Number of speed changes per distance covered

5.2.3 Pattern 3 - Mixed Operations on 01L, Arrivals on 01R, Departures on 08

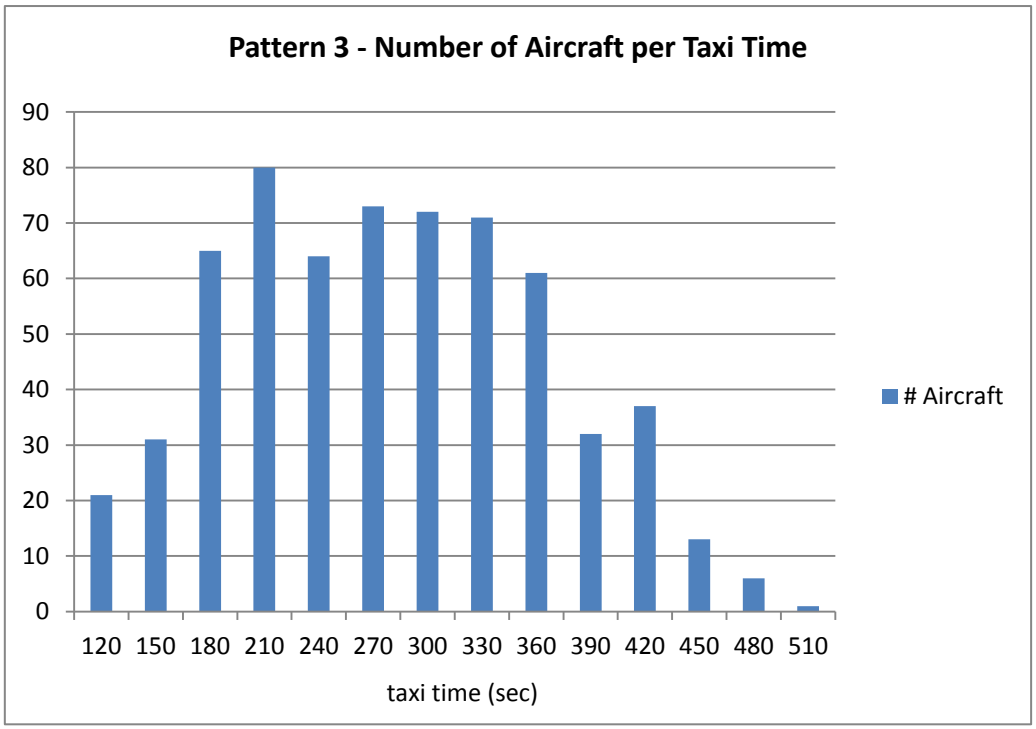


Figure 5.9: The distribution of aircraft per taxi time

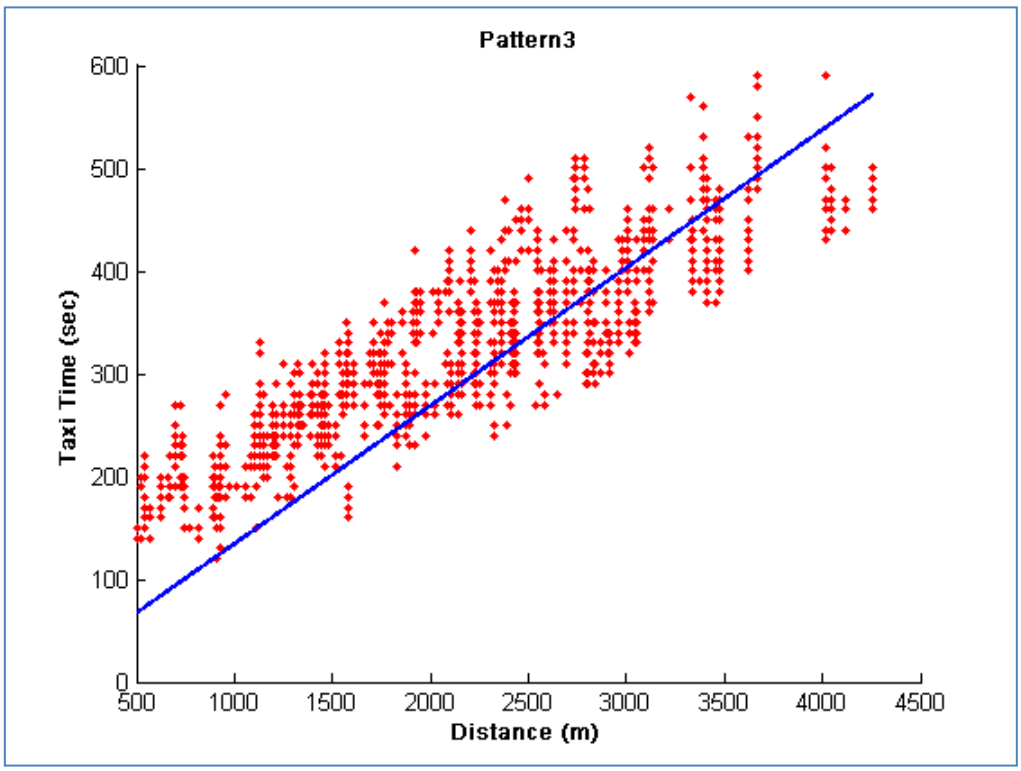


Figure 5.10: Taxi time per distance covered, including the tendency line

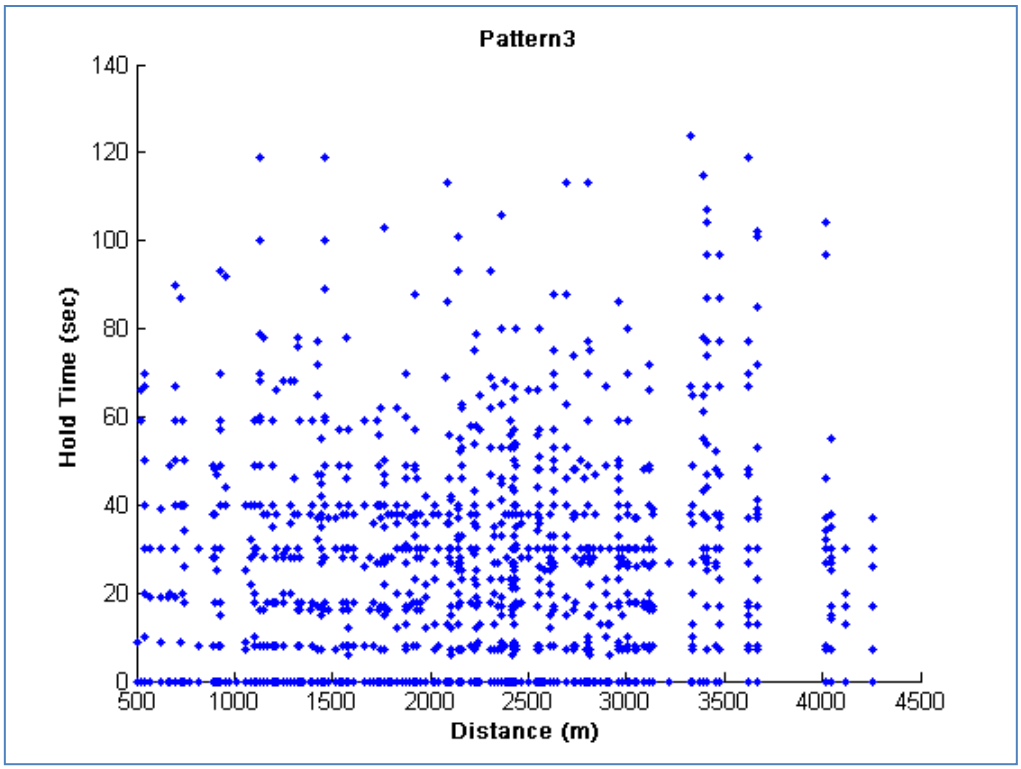


Figure 5.11: Hold time per distance covered

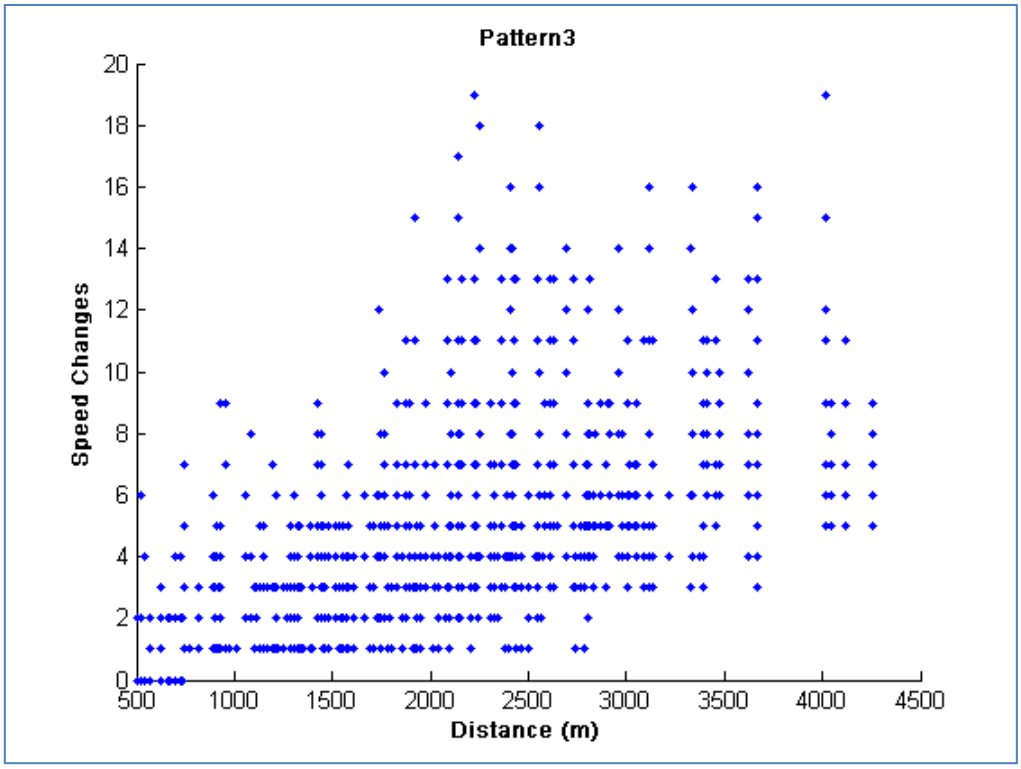


Figure 5.12: Number of speed changes per distance covered

5.2.4 Pattern 4 - Arrivals on Runway 26, Departures on Runway 19R

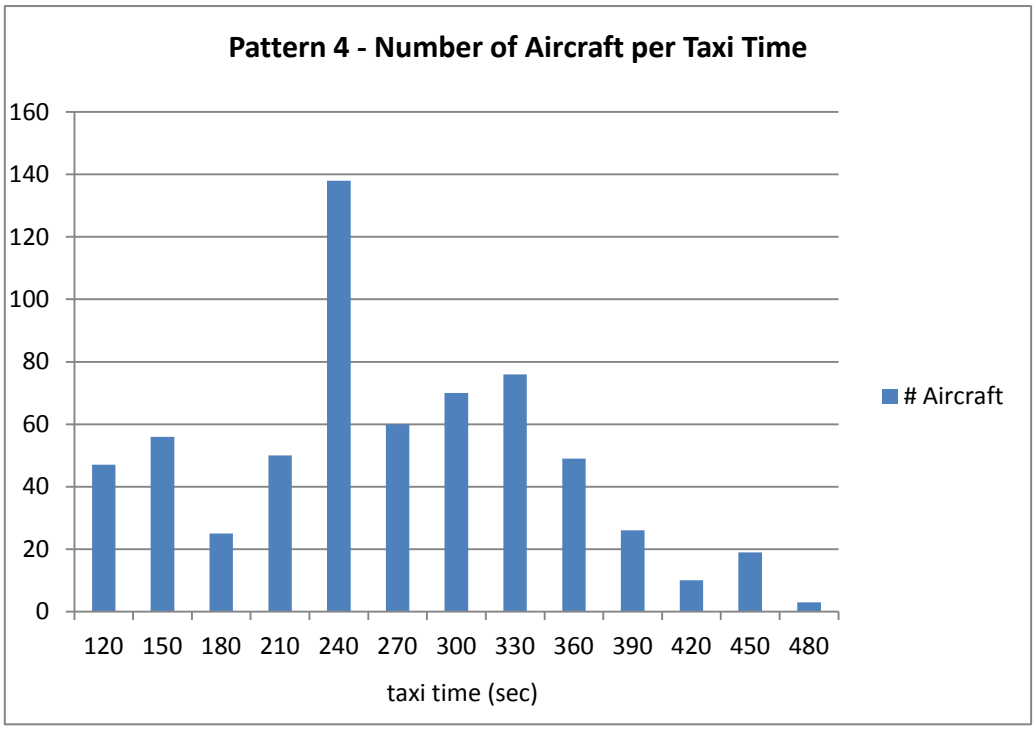


Figure 5.13: The distribution of aircraft per taxi time

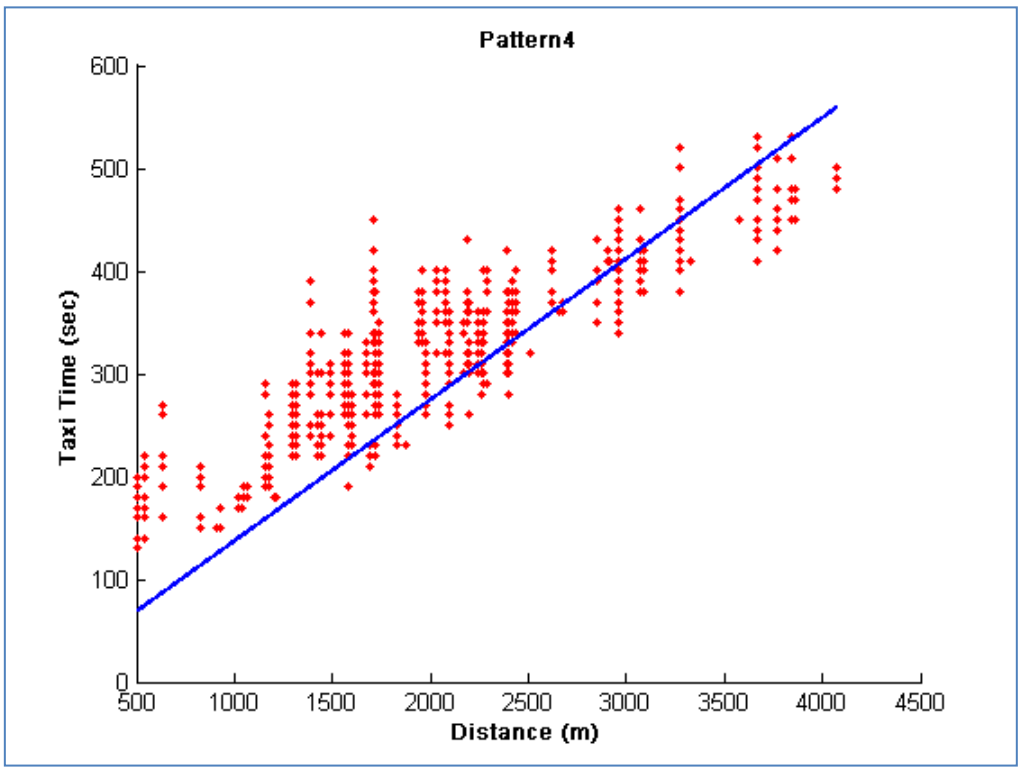


Figure 5.14: Taxi time per distance covered, including the tendency line

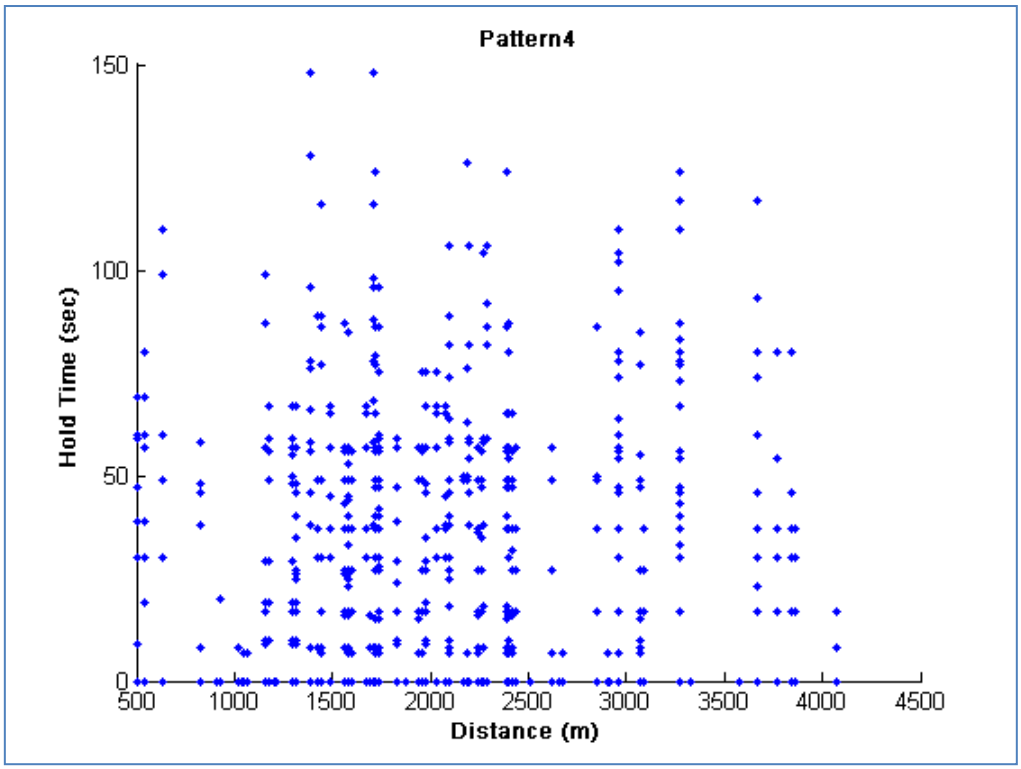


Figure 5.15: Hold time per distance covered

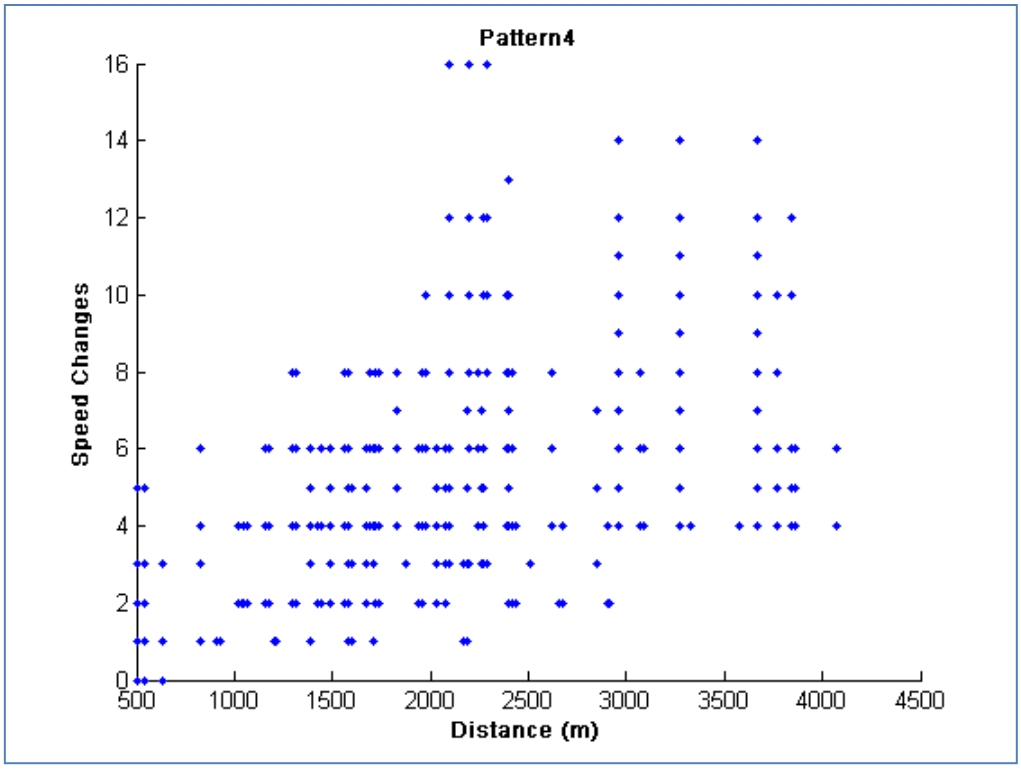


Figure 5.16: Number of speed changes per distance covered

5.2.5 Pattern 5 - Arrivals on Runway 19R, Departures on Runway 08

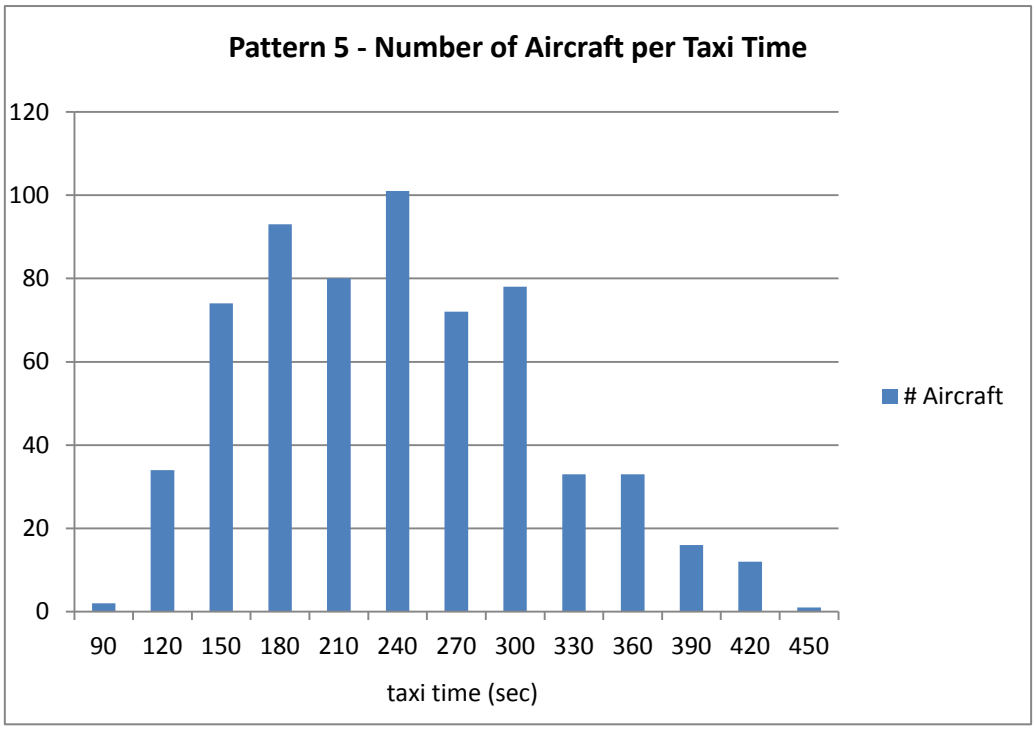


Figure 5.17: The distribution of aircraft per taxi time

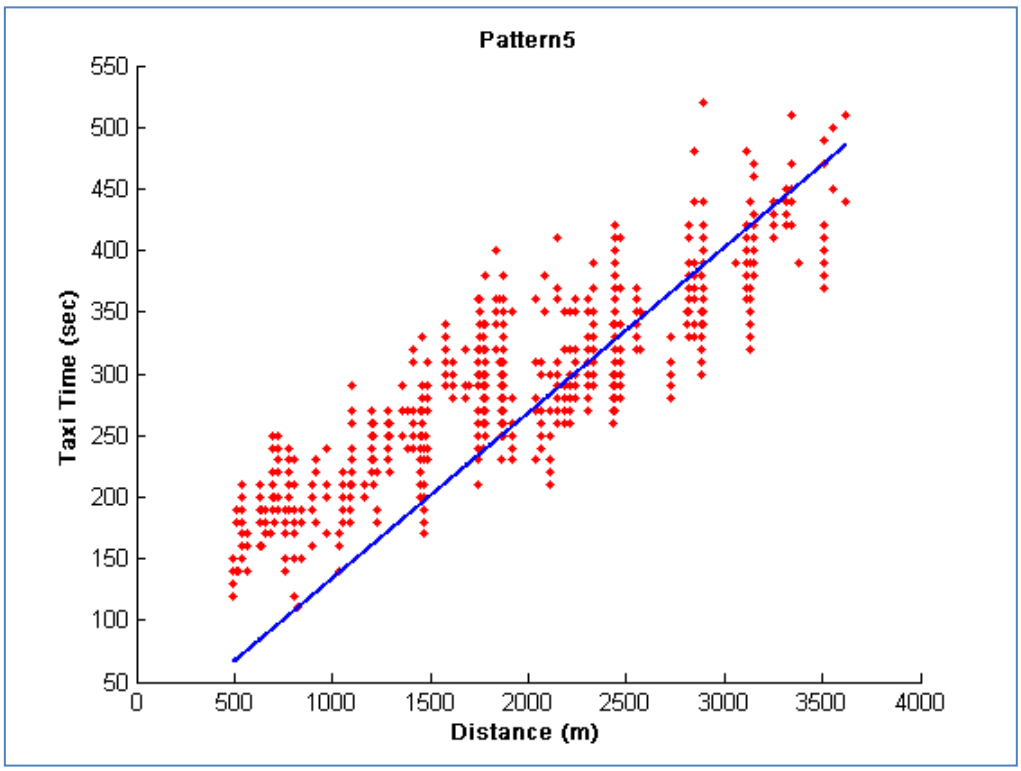


Figure 5.18: Taxi time per distance covered, including the tendency line

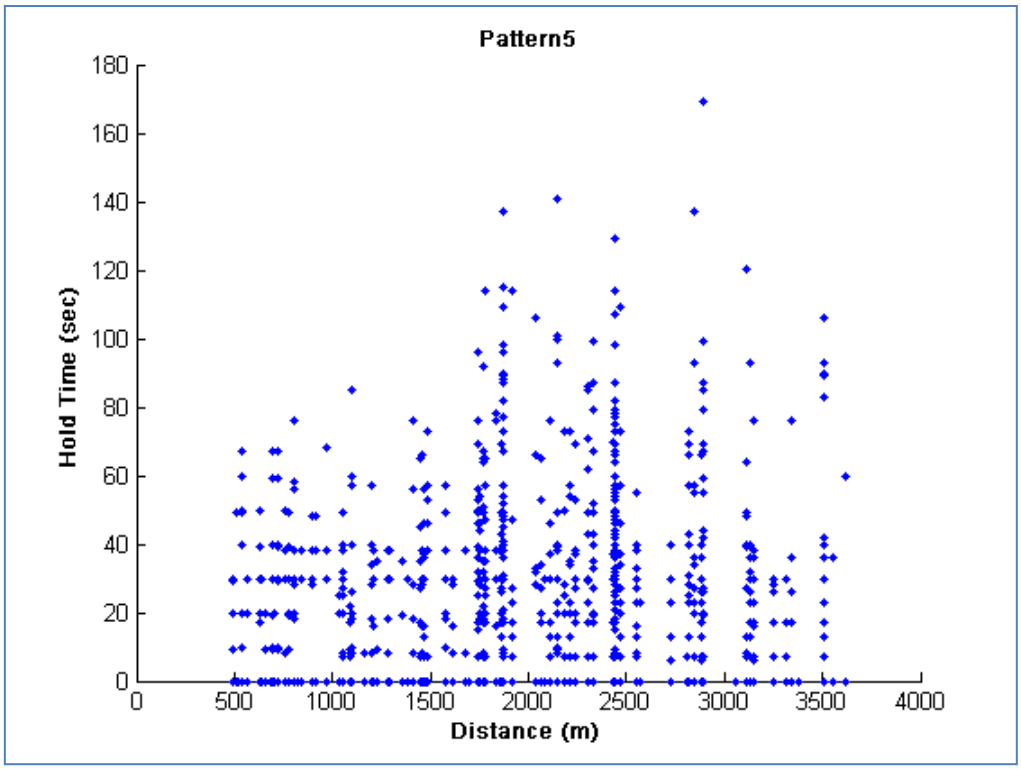


Figure 5.19: Hold time per distance covered

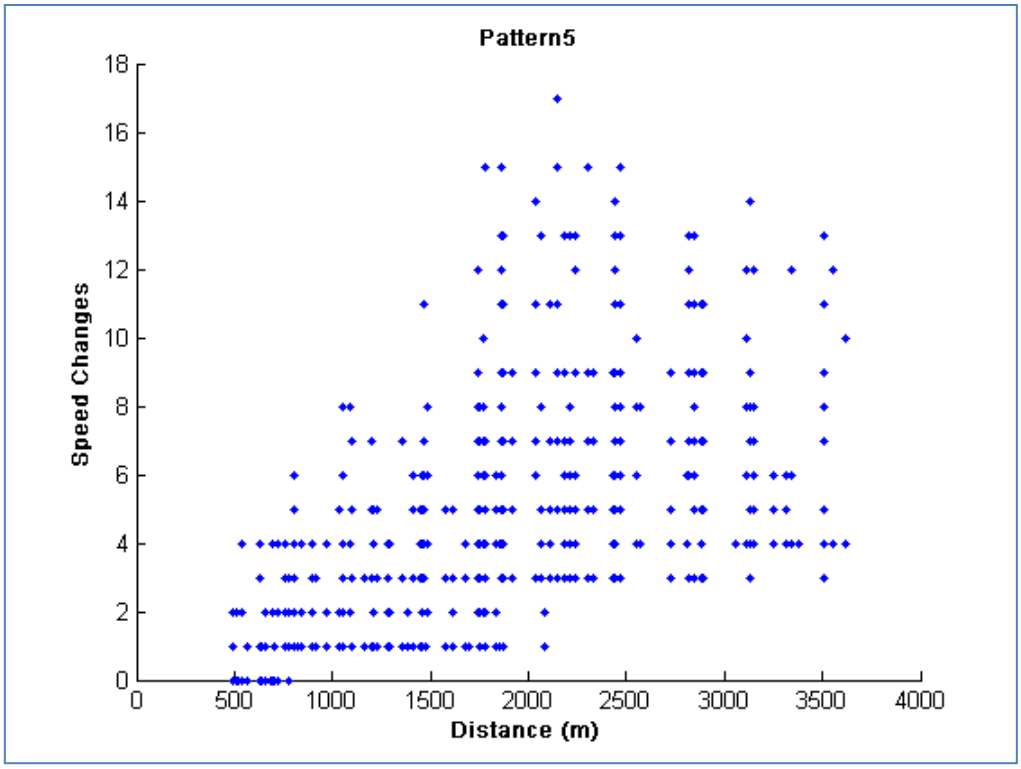


Figure 5.20: Number of speed changes per distance covered

5.2.6 Pattern 6 – Arrivals on Runway 01L, Departures on Runway 08

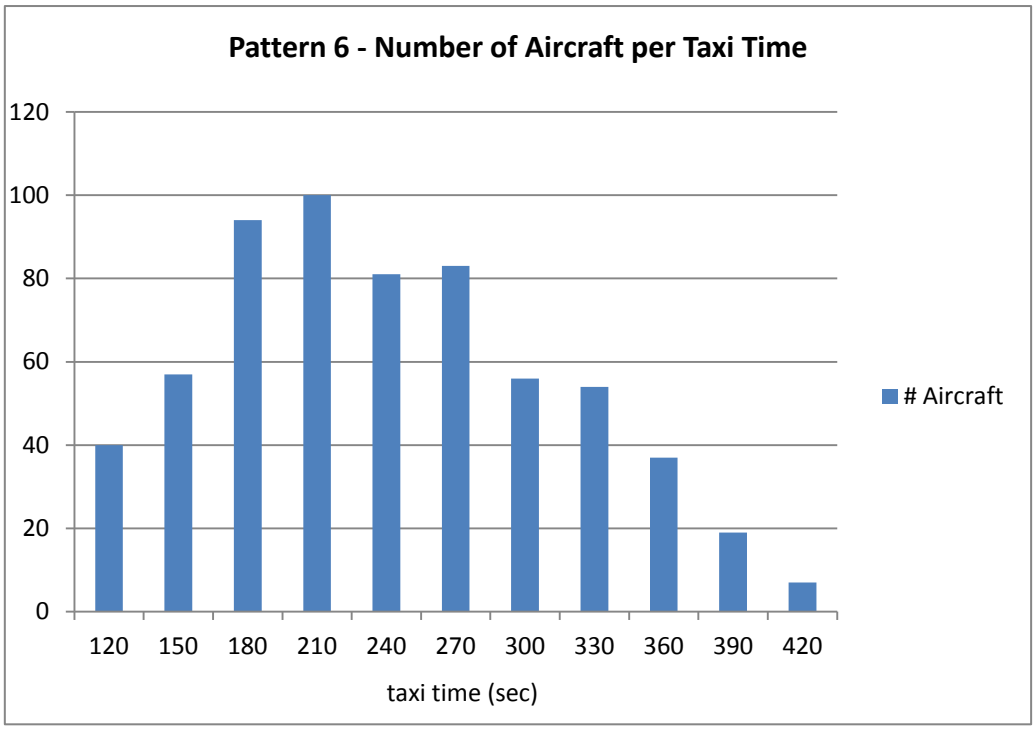


Figure 5.21: The distribution of aircraft per taxi time

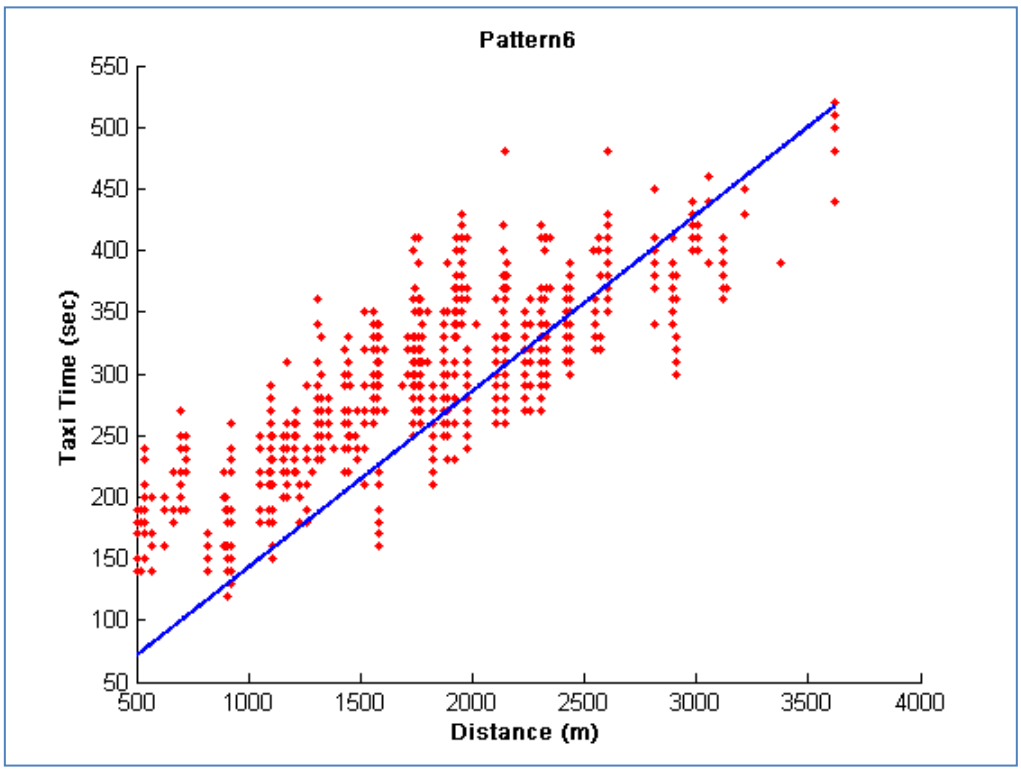


Figure 5.22: Taxi time per distance covered, including the tendency line

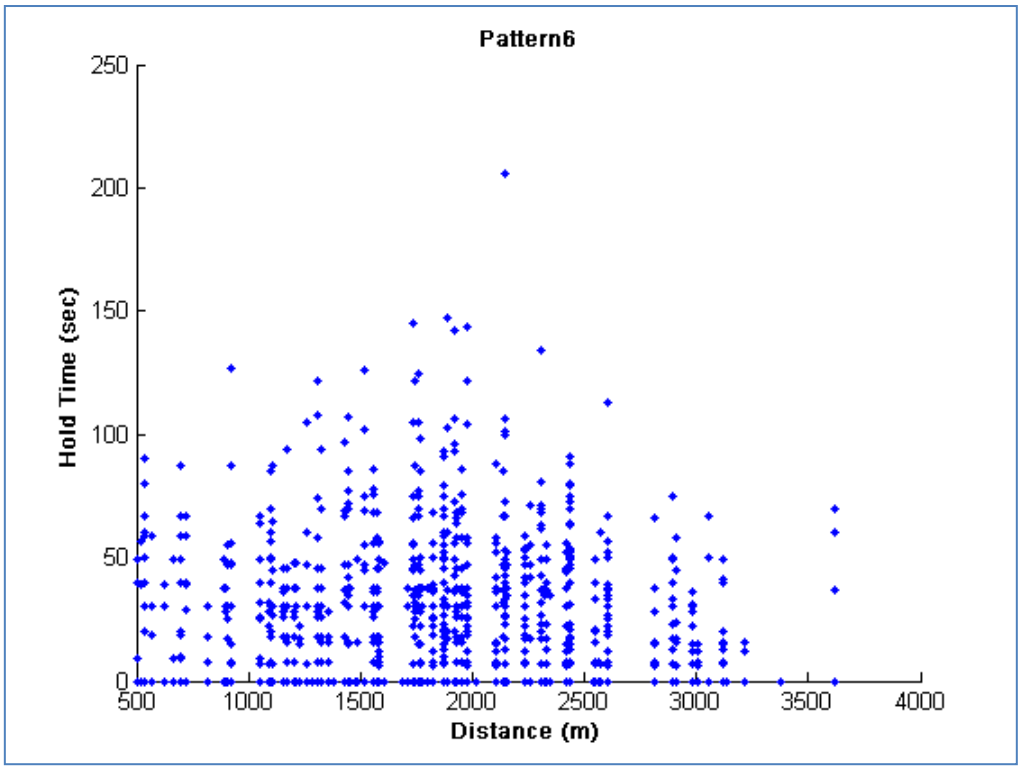


Figure 5.23: Hold time per distance covered

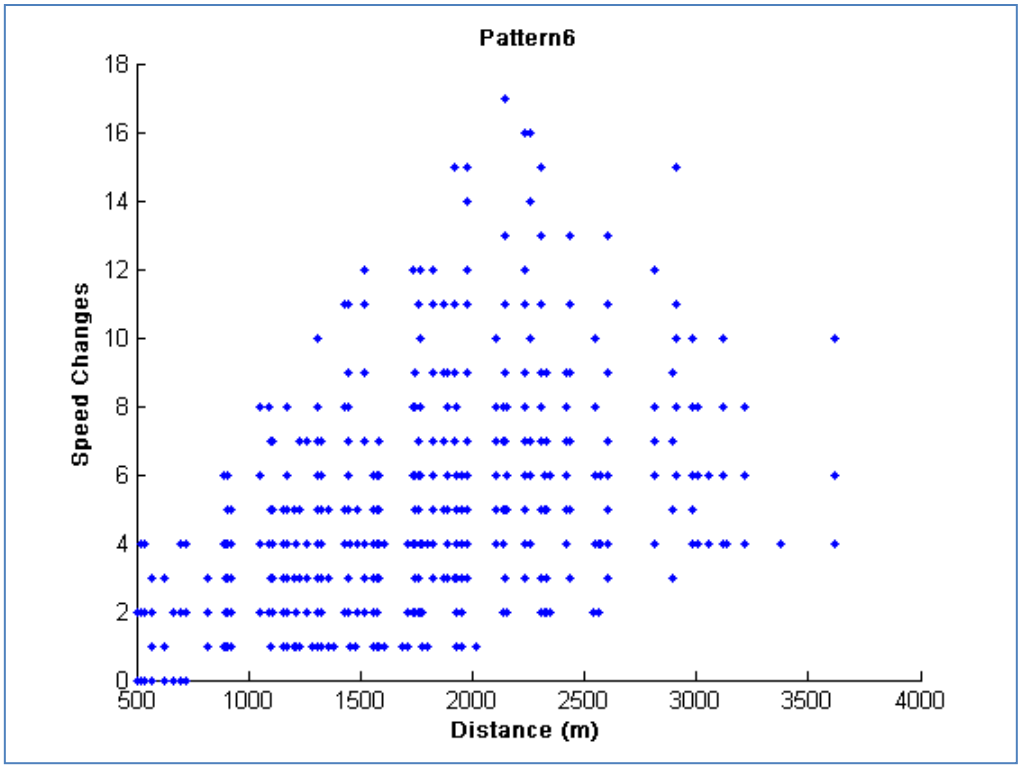


Figure 5.24: Number of speed changes per distance covered

5.2.7 Pattern 7 – Arrivals on Runway 26, Departures on Runway 01L

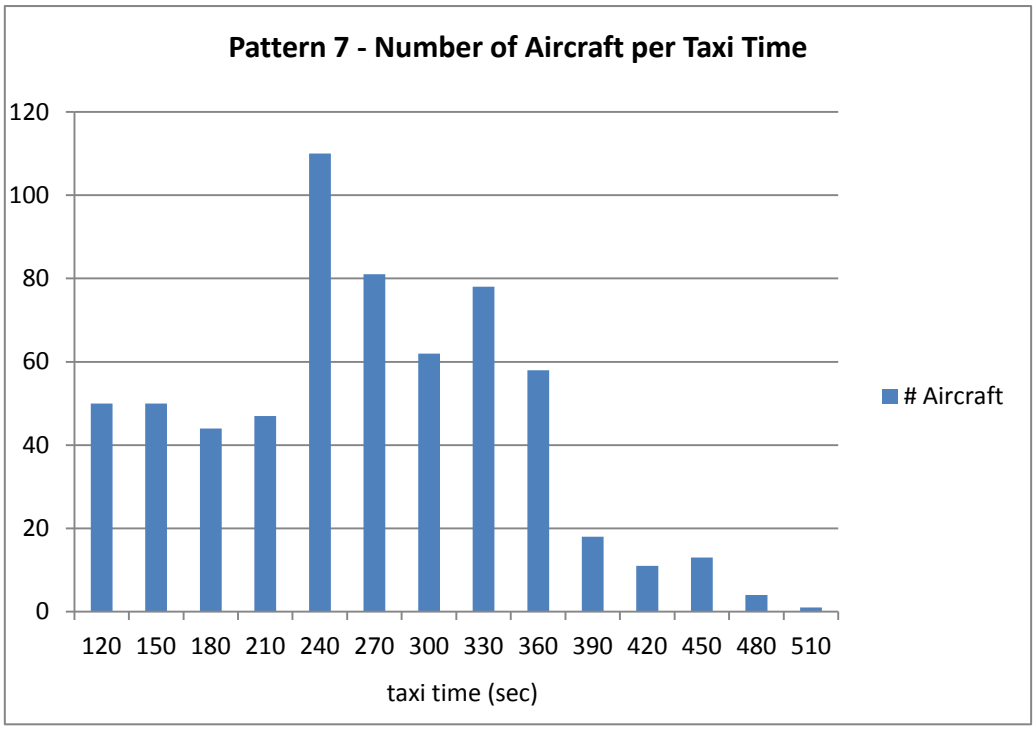


Figure 5.25: The distribution of aircraft per taxi time

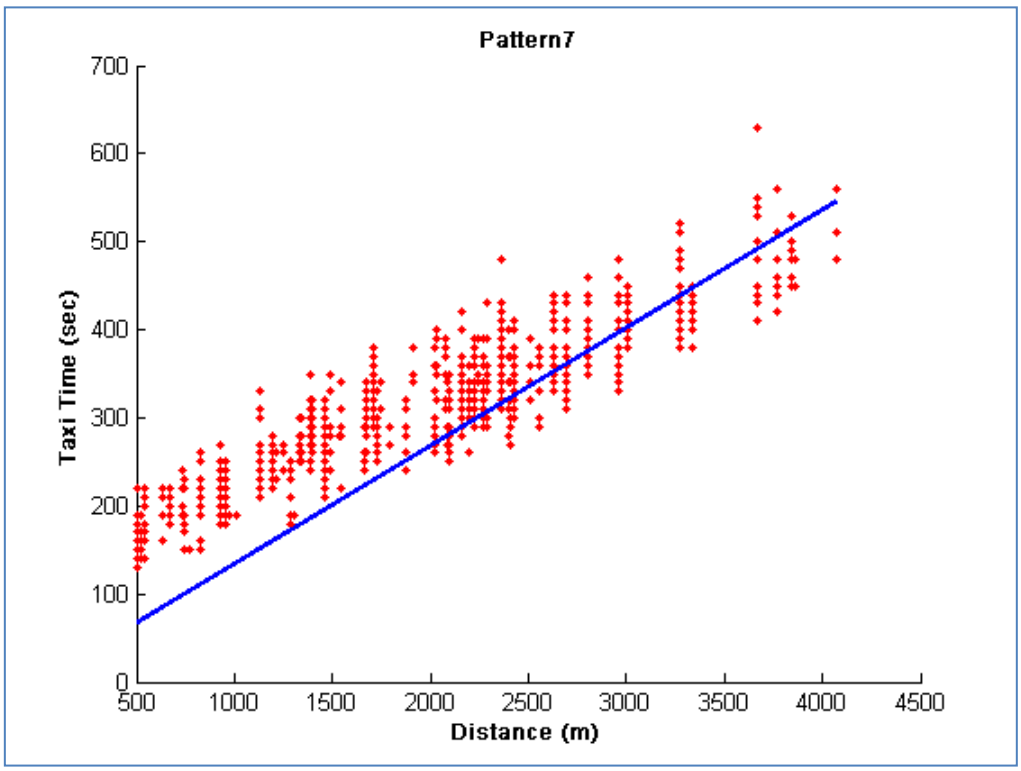


Figure 5.26: Taxi time per distance covered, including the tendency line

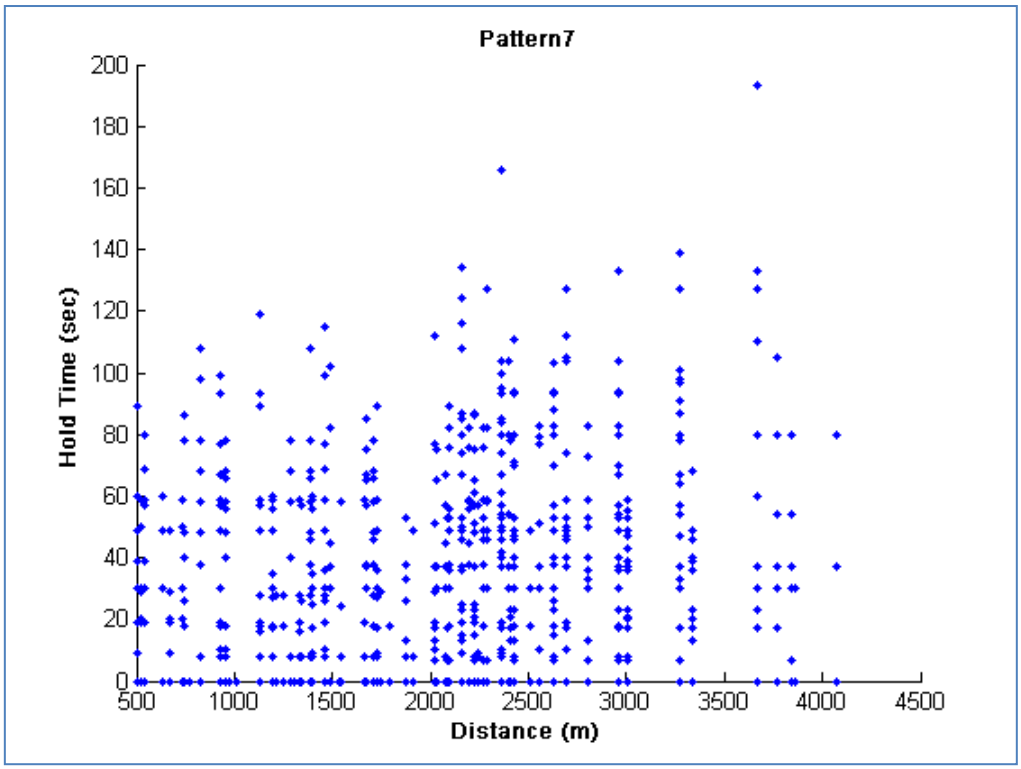


Figure 5.27: Hold time per distance covered

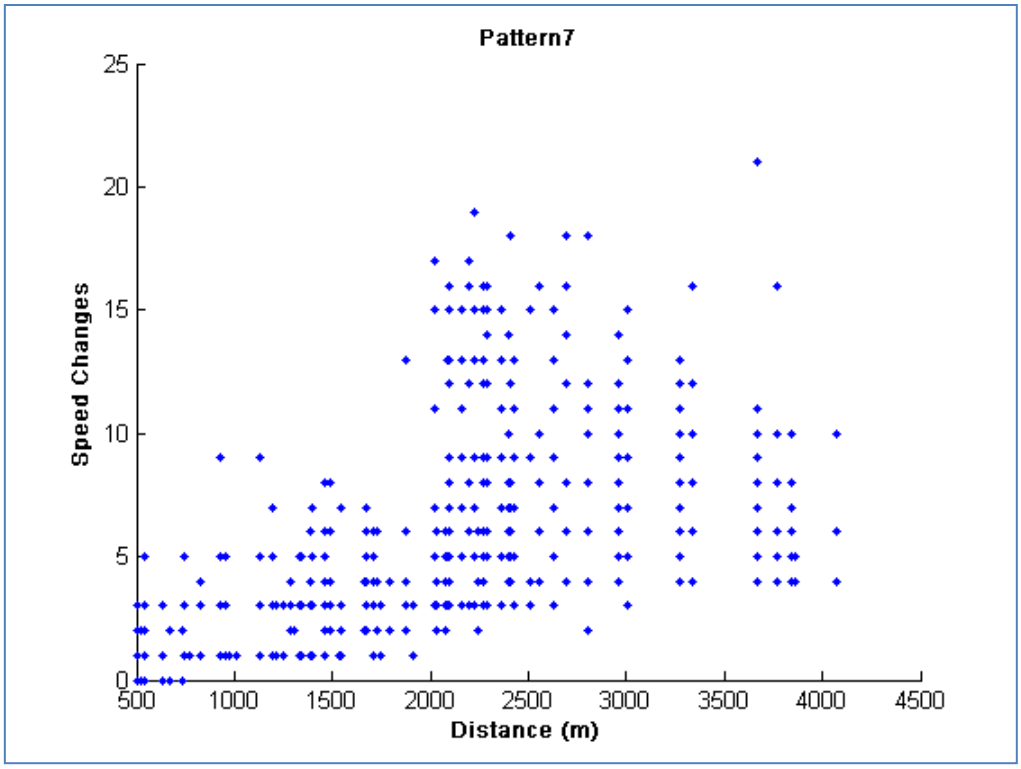


Figure 5.28: Number of speed changes per distance covered

5.2.8 Pattern 8 – Arrivals on Runway 01R, Departures on Runway 01L

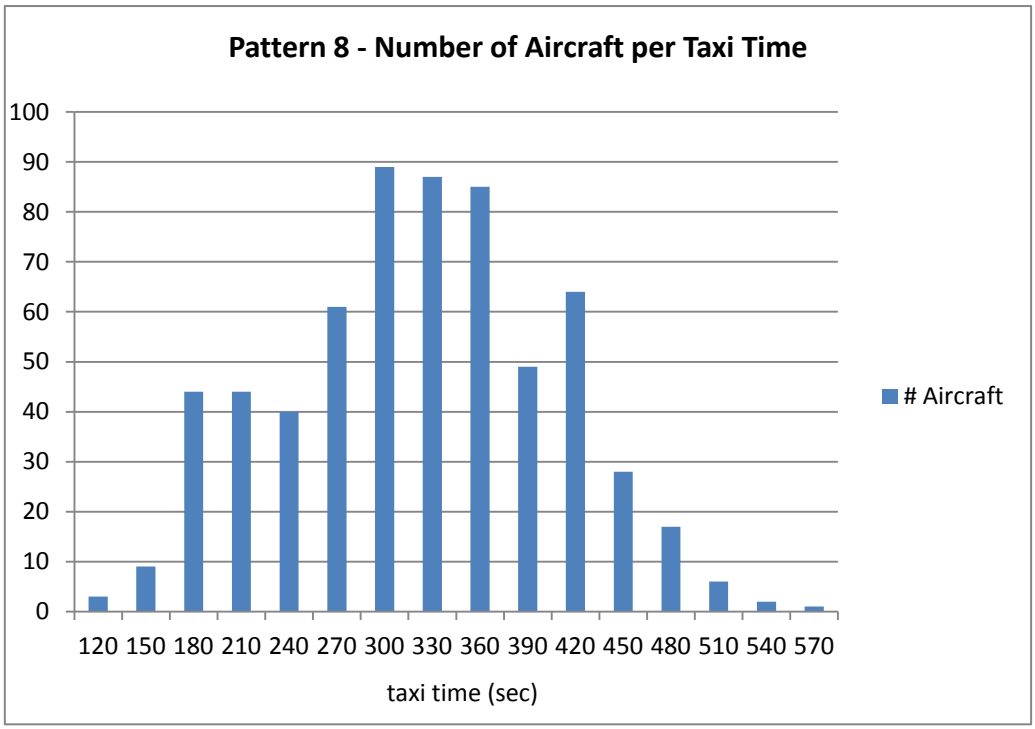


Figure 5.29: The distribution of aircraft per taxi time

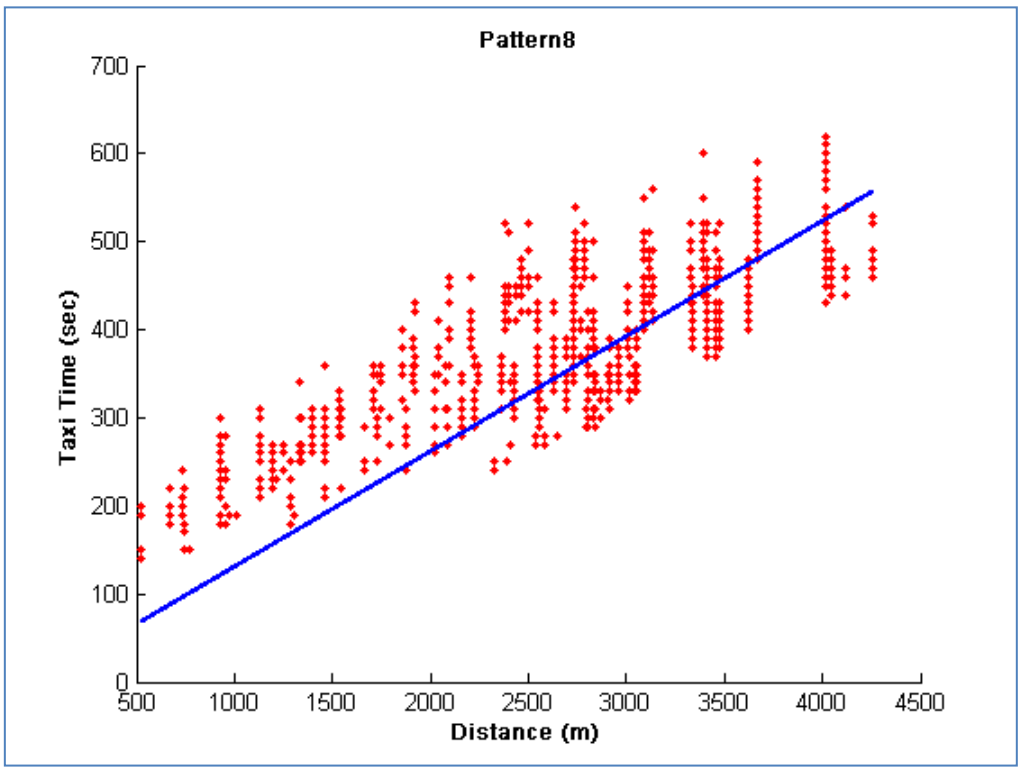


Figure 5.30: Taxi time per distance covered, including the tendency line

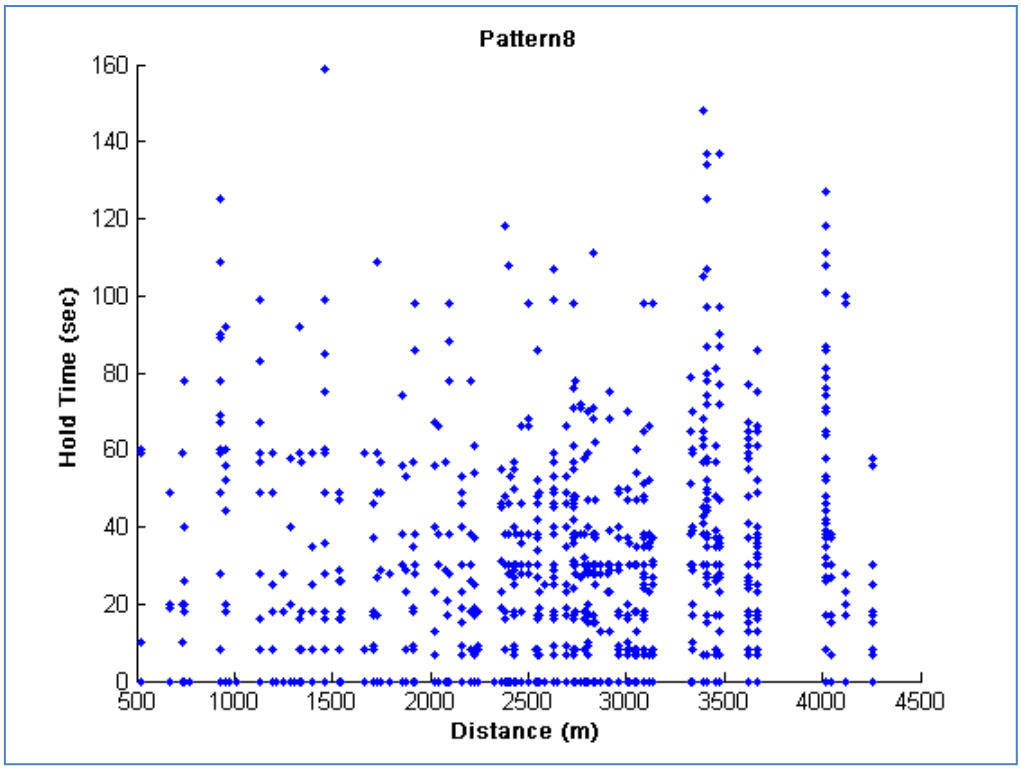


Figure 5.31: Hold time per distance covered

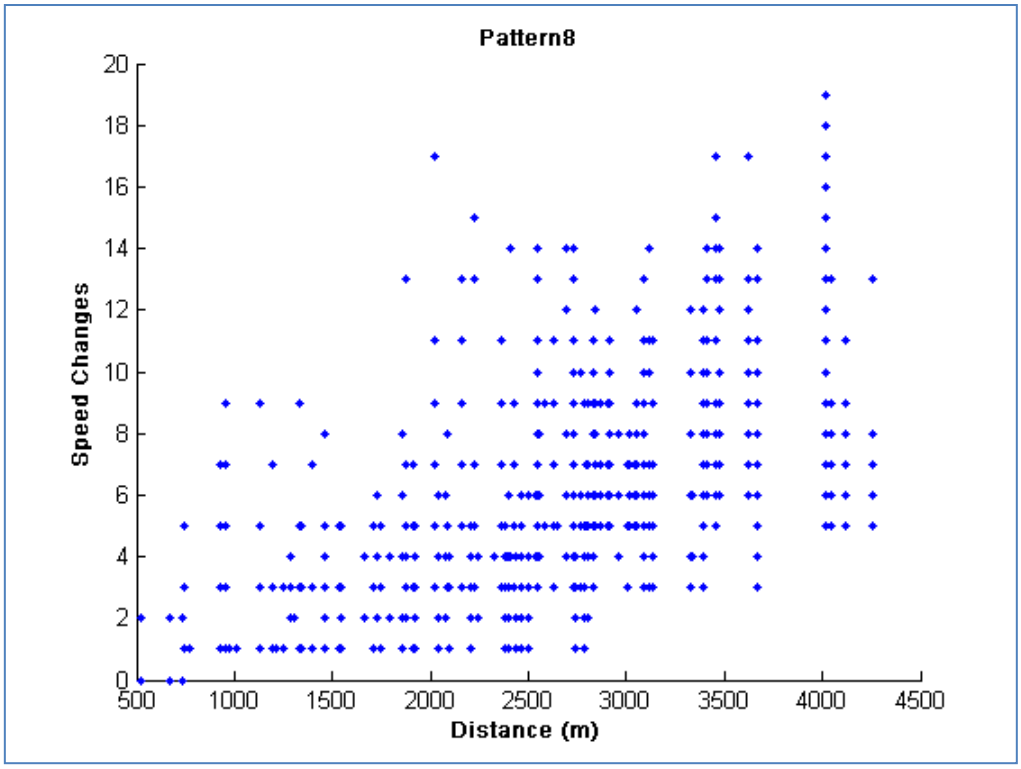


Figure 5.32: Number of speed changes per distance covered

Chapter 6 **Conclusions**

The present thesis work was intended to be a thorough study of the problem of 4D Taxi Routing on Ground. It started with a description of the taxiway routing process and an analysis of its structural and operational entities, particularities, constraints, as well as the interaction with its surrounding processes – arrival, departure and gate management. At the end of the first chapter a set of research questions was defined, which would determine the research directions throughout this work.

6.1 Answers to the Research Questions

The main part of this work was dedicated to answering the first research question: "*How can the above described problem be formulated into a mathematical model?*" A model was consecutively created with the purpose of capturing the essence of the problem; it is based on a time-dependent, labeled, bimodal and directed graph with agents – finite state machines – moving on it. This is shortly the answer to the first research question according to this work. The reasoning that concluded to this model and the model itself were described in chapter 3 and its realization in chapter 4.

The algorithm implemented was a modified iterative version of Dijkstra's algorithm for the single-source shortest path problem, supplied with a consequent linear programming optimization of the speed vector. Since the speed determines the position of an aircraft at each time, the linear programming optimization focused on the 4^{th} dimension. However, this algorithm-set has significant inefficiencies. There is no look-ahead in time and no flexibility, two factors that intuitively seem crucial for a time-dependent problem. Each aircraft is assigned a route that does not change based on a static image of the taxiway network graph. This route corresponds to the shortest path and is the optimal solution provided that no other aircraft will be following the same or part of the same route simultaneously. As the traffic increases, the meetings of aircraft on the taxiway become more often and Dijkstra's algorithm does not provide any insight on the current traffic load, in order to balance it by using alternative routes.

The simulation routing results under runway usage patterns 1 and 2, presented in chapter 5, show clearly this lack of flexibility of the implemented algorithm. Aircraft are routed via taxiways U and W, which are long enough - thus it takes more time to traverse them - to increase the possibility that while an aircraft is taxiing on the one direction – say from A to B – another direction appears at vertex B intending to taxi on the opposite direction – B to A. According to the routing algorithm, both aircraft will stop and hold infinitely. This is the mutual blocking or deadlock situation.

Therefore, the answer to the second research question is that a classical SPP algorithm is not suitable for a 4D problem. In our case, it can be roughly stated that Dijkstra's algorithm optimizes the spatial dimensions and the LP formulation optimizes time, however these algorithms are decoupled. This is more like a stepwise or "fragmented" optimization and the problem of 4D Taxi Routing on Ground is suspected to need an overall approach where space and time are optimized together. But what could this approach look like?

In the second chapter a number of tools and algorithms were presented as the current state-of-the-art in this and similar research areas. Taxi Planner Optimization (TPO), presented in section

2.2.2, approaches the problem using a look-ahead LP formulation. The present work used Dijkstra's algorithm to gain insight into the problem and propose extensions or alternatives:

- A look-ahead approach that uses the combination of TAAM [23] as a simulation tool for different routing scenarios and a heuristic, most probably a Genetic Algorithm [31], to crossover promising candidates towards an optimum. This implies an effective coding of solutions.
- A dynamic approach that recalculates the routes while the aircraft are already on the taxiway, using some heuristic methods. The Ant Colony Systems [25] have been proven flexible and stable and could be combined with a load-balancing objective function.
- Another dynamic approach, that keeps a list of K-Shortest Paths [27] for each aircraft, evaluates them at each time and acts accordingly.
- A more "experimental" or stochastic approach, similar to the Canadian Traveler Problem [30].
- The utilization of concurrency strategies, like semaphores, to avoid deadlocks.

The algorithm of Dijkstra could as well be used in all of the above mentioned approaches, but only as the generator of initial routings, which would then be revised continuously. Robustness is a basic quality issue that has to be addressed by every proposed algorithm.

The third research question is about the problem objectives: *"How efficiently can the objectives be met? How much can the mean taxi time, hold time etc be decreased?"* The results of the simulations are by no means exhaustive. They cannot lead to concrete conclusions, but they can show some directions or some tendencies. So, these results are considered satisfactory in the sense that they are relevant to what experience has shown (comparable to the actual values of these metrics for our case study airport) [8]. There is though a lot of improvement that can be made. Other algorithms can lead to more efficient solutions, if they address the flexibility and traffic load balancing issues. This research area seems quite promising and the present work aimed at providing the foundations where different optimization algorithms can be easily "plugged-in" and evaluated.

6.2 Future Work

The rest of the research questions are intended to be answered in future extensions of the present work, together with some revisions on the basic model which can potentially make it more adaptable to the application of different optimization algorithms. A significant revision could be to replace each vertex categorized as "crossing" with two respective vertices, one for each direction. It is like building a bridge to replace a crossroad, so that vehicles on the one direction cannot turn into the other direction. This way, the use of a transition table would be redundant, since the graph would suffice for this, so the whole model would become much simpler.

A very interesting result indicative of the taxiway routing process and the performance of the algorithms is the evolution of taxi time with increasing traffic on the taxiway. Such graphs were not displayed for every runway usage pattern in chapter 5, because the simulation of the 2-day flight schedule did not produce any significant results. For example, under pattern 4 there were at most 7 aircraft on the taxiway during this schedule. Figure 6.1 below emphasizes this; there is no evident relation between the increasing traffic on the taxiway and the average taxi time.

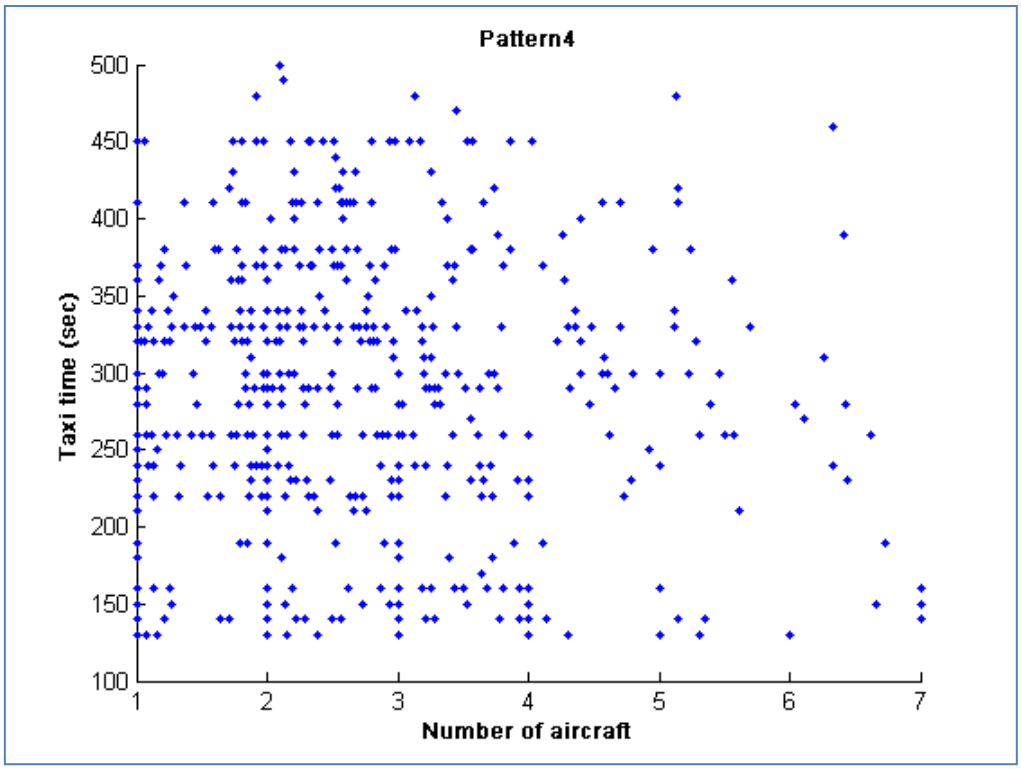


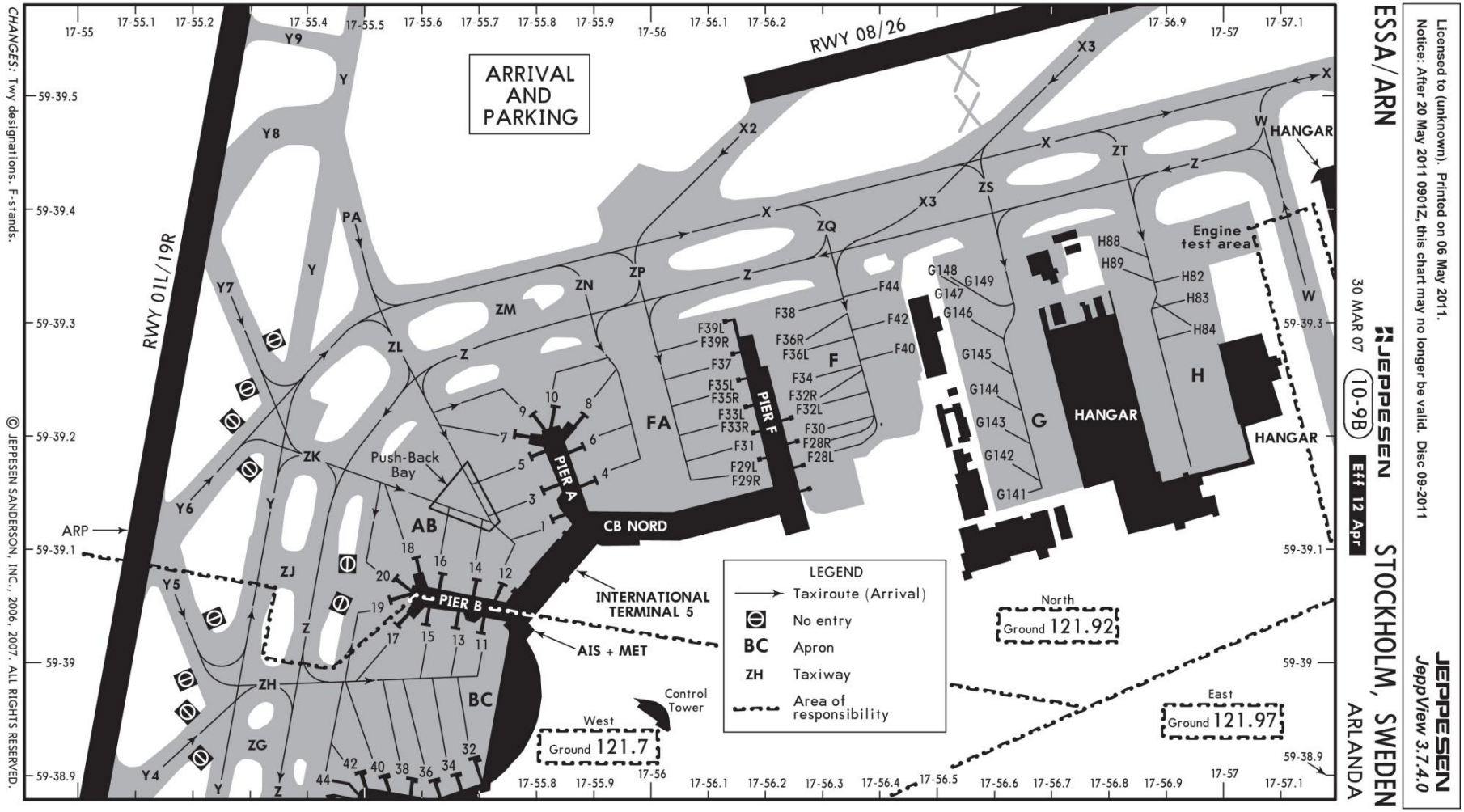
Figure 6.1: Taxi time per traffic (number of aircraft)

Nevertheless, common sense would expect that increasing traffic leads to more delays, as with any road network. Therefore a main direction of future work would be the thorough testing (Monte-Carlo simulations) of the behavior of the taxiway network under all different combinations of input parameters and increasing traffic until reaching a "breaking point", a traffic load which is the capacity limit of a given airport and the answer to the last research question.

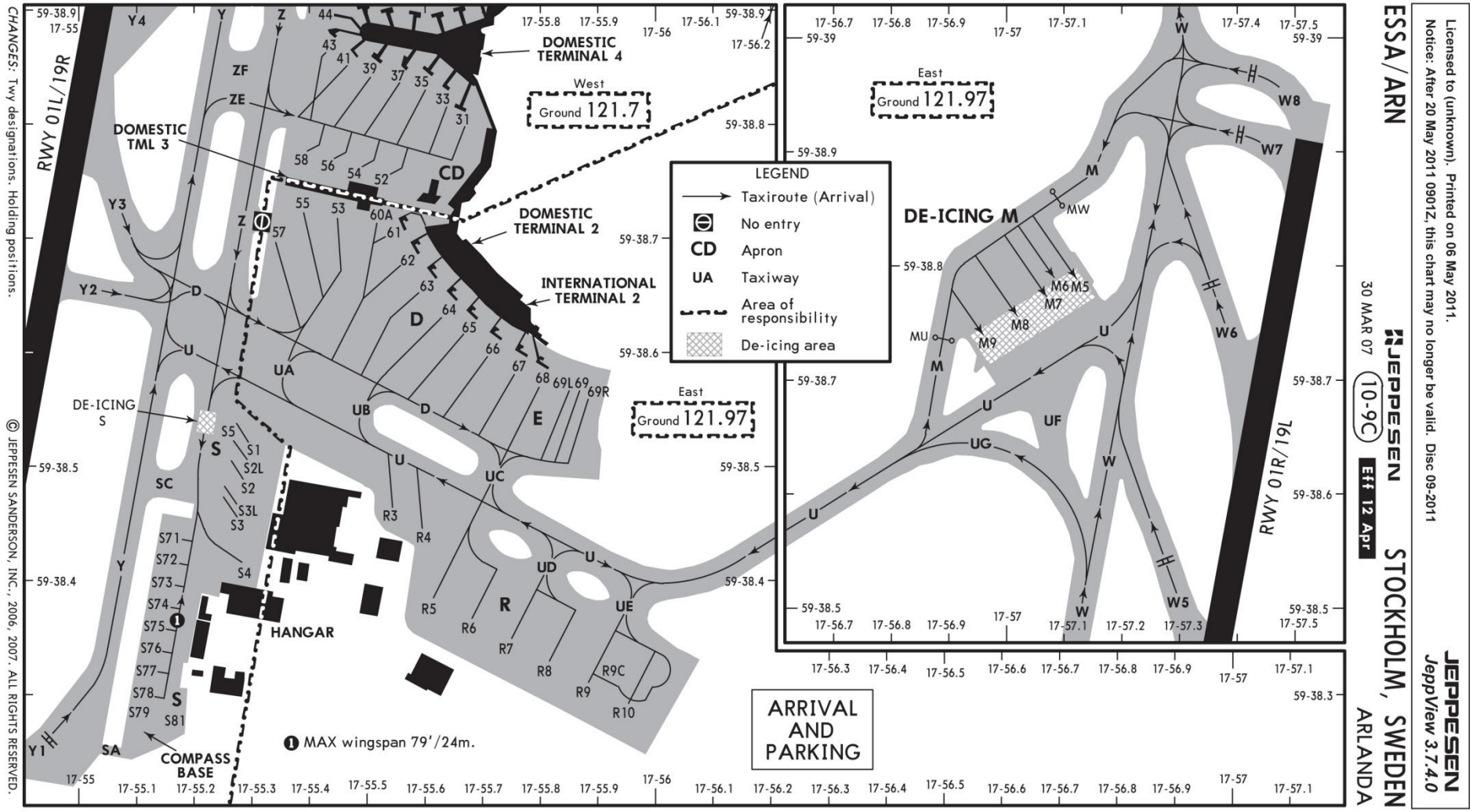
Another direction of future work is of course the design and implementation of different algorithms, as already mentioned above, so that comparisons will be possible. The model and its realization are independent of the optimization algorithm. They are independent of the airport as well, so the generation of data files for more airports would support the usefulness of this work. Apart from these extensions and based on the number of unsuccessful simulations presented in chapter 5, it is necessary to perform some restructuring on the Matlab code – the routing execution algorithm – in order to diminish the undesirable outcomes and/or improve the simulations' speed.

Appendix I – Airport maps

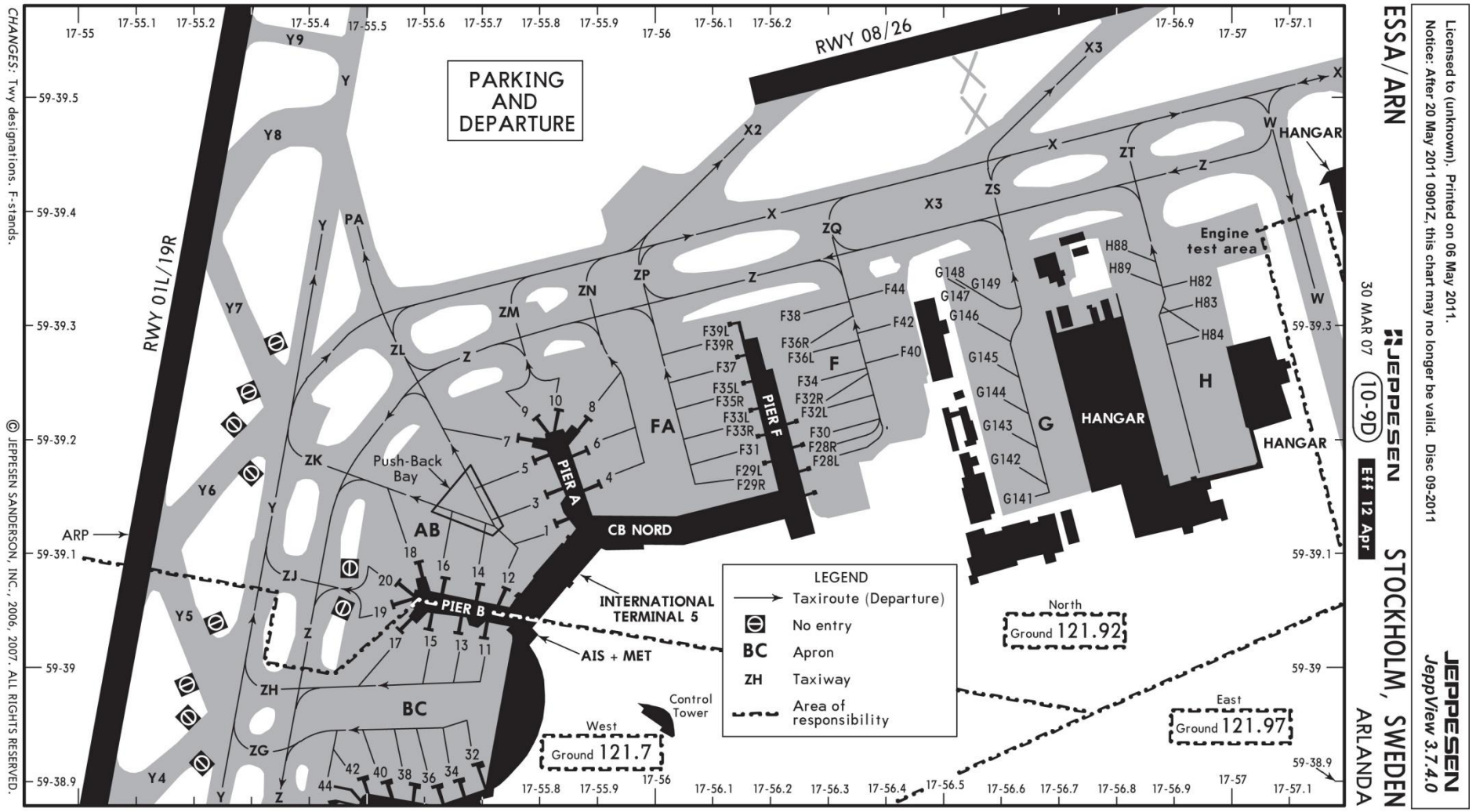
A. Stockholm-Arlanda North – Arrival and Parking



B. Stockholm-Arlanda South – Arrival and Parking



C. Stockholm-Arlanda North –Parking and Departure



D. Stockholm-Arlanda South –Parking and Departure

ESSA/ARN
STOCKHOLM, SWEDEN
ARLANDA
JEPPESEN
30 MAR 07
10-9E
Eff 12 Apr
JEPPESEN
JeppView 3.7.4.0
Licensed to (unknown). Printed on 06 May 2011.
Notice: After 20 May 2011 0901Z, this chart may no longer be valid. Disc 09-2011
CHANGES: Twy designations. Holding positions.
© JEPPESEN SANDERSON, INC., 2006, 2007. ALL RIGHTS RESERVED.
PARKING AND DEPARTURE
LEGEND
Taxiroute (Departure)
No entry
CD Apron
UA Taxiway
Area of responsibility
De-icing area
DOMESTIC TERMINAL 4
DOMESTIC TML 3
DOMESTIC TERMINAL 2
INTERNATIONAL TERMINAL 2
West Ground 121.7
East Ground 121.97
DE-ICING M
DE-ICING S
HANGAR
COMPASS BASE
RWY 01L/19R
RWY 01R/19L
MAX wingspan 79'/24m.

Appendix II – Distance Matrices

A. From the Runway Exits to the Parking Stand Areas

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16
Y1	3791	3275	3102	2865	Inf	2605	2237	Inf	2680	Inf	1710	3476	3361	3640	3623	3866
Y2	3285	2769	2596	2359	Inf	2099	1731	Inf	2174	Inf	1204	980	798	1077	1060	1303
Y3	3316	2800	2627	2390	Inf	2130	1762	Inf	2205	Inf	1235	1106	924	1203	1186	1429
Y4	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	976	Inf	1368	1789	1674	1953	1936	2179
Y5	2765	2249	2076	1839	Inf	1579	1211	Inf	983	Inf	1375	1796	1681	1960	1943	2186
Y6	2369	1853	1680	1443	Inf	1183	900	Inf	1343	Inf	1718	2139	2024	2303	2286	2529
Y7	2354	1838	1665	1428	Inf	1168	968	Inf	1411	Inf	1786	2207	2092	2371	2354	2597
Y8	2464	1948	1775	1538	Inf	1237	Inf	Inf	1885	Inf	2260	2681	2566	2845	2828	3071
Y9	2456	1940	1767	1530	Inf	1229	Inf	Inf	1877	Inf	2252	2673	2558	2837	2820	3063
Y10	2931	2415	2242	2005	Inf	1704	Inf	Inf	2352	Inf	2727	3148	3033	3312	3295	3538
X2	Inf	Inf	860	852	Inf	1188	Inf	Inf	1783	Inf	2158	2579	2464	2743	2726	2969
X3	960	923	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf
X4	1947	1971	2412	2377	Inf	2713	Inf	Inf	3308	Inf	3683	4104	3989	4268	4251	4494
X5	2697	2721	3162	3127	Inf	3463	Inf	Inf	4058	Inf	4433	4854	4739	5018	5001	5244
W1	4724	4748	5189	5154	Inf	5490	5605	Inf	6048	Inf	5078	4390	4441	4469	4136	3853
W2	4631	4655	5096	5061	Inf	5397	5512	Inf	5955	Inf	4985	4297	4348	4376	4043	3760
W3	4688	4712	5153	5118	Inf	5454	5569	Inf	6012	Inf	5042	4354	4405	4433	4100	3817
W4	4255	4279	4720	4685	Inf	5021	5136	Inf	5579	Inf	4609	3921	3972	4000	3667	3384
W5	2968	2992	3433	3398	Inf	3734	4211	Inf	4329	Inf	3684	2996	3047	3075	2742	2459
W6	2715	2739	3180	3145	Inf	3481	4252	Inf	4076	Inf	3725	3037	3088	3116	2783	2500
W7	2336	2360	2801	2766	Inf	3102	4270	Inf	3697	Inf	3743	3055	3106	3134	2801	2518
W8	2311	2335	2776	2741	Inf	3077	4437	Inf	3672	Inf	3910	3222	3273	3301	2968	2685

B. From the Parking Stand Areas to the Runway Exits

	Y1	Y2	Y3	Y4	Y5	Y6	Y7	Y8	Y9	Y10	X2	X3	X4	X5	W1	W2	W3	W4	W5	W6	W7	W8
P1	3699	3310	3587	Inf	Inf	Inf	Inf	2488	2480	2955	1597	960	1883	2633	4927	4834	4891	4458	Inf	2918	2539	2514
P2	3171	2782	3059	Inf	Inf	Inf	Inf	1960	1952	2427	1069	1007	1934	2684	4978	4885	4942	4509	Inf	2969	2590	2565
P3	2982	2593	2870	Inf	Inf	Inf	Inf	1771	1763	2238	860	1444	2371	3121	5415	5322	5379	4946	Inf	3406	3027	3002
P4	2766	2377	2654	Inf	Inf	Inf	Inf	1555	1547	2022	875	1406	2333	3083	5377	5284	5341	4908	Inf	3368	2989	2964
P5	2618	2229	2506	Inf	Inf	Inf	Inf	1407	1399	1874	973	1504	2431	3181	5475	5382	5439	5006	Inf	3466	3087	3062
P6	2468	2079	2356	Inf	Inf	Inf	Inf	1237	1229	1704	1294	1825	2752	3502	5714	5621	5678	5245	Inf	3787	3408	3383
P7	2305	1916	2193	Inf	Inf	Inf	Inf	1552	1544	2019	1633	2164	3091	3841	5551	5458	5515	5082	Inf	4126	3747	3722
P8	2099	1710	1987	Inf	Inf	Inf	Inf	1607	1599	2074	1688	2219	3146	3896	5345	5252	5309	4876	Inf	3992	3802	3777
P9	1983	1594	1871	Inf	Inf	Inf	Inf	1883	1875	2350	1964	2495	3422	4172	5229	5136	5193	4760	Inf	3876	3894	4053
P10	1924	1535	1812	Inf	Inf	Inf	Inf	2027	2019	2494	2108	2639	3566	4316	5170	5077	5134	4701	Inf	3817	3835	4002
P11	1773	1384	1661	Inf	Inf	Inf	Inf	2236	2228	2703	2317	2848	3775	4525	5019	4926	4983	4550	Inf	3666	3684	3851
P12	785	874	1151	Inf	Inf	Inf	Inf	2627	2619	3094	2708	3239	4166	4916	7210	7117	7174	6741	Inf	5201	4822	4797
P13	1527	1138	1415	Inf	Inf	Inf	Inf	2891	2883	3358	2972	3503	4430	5180	7474	7381	7438	7005	Inf	5465	5086	5061
P14	2103	1714	1991	Inf	Inf	Inf	Inf	3467	3459	3934	3548	4079	5006	5756	4235	4142	4199	3766	Inf	2882	2900	3067
P15	1502	1113	1390	Inf	Inf	Inf	Inf	2866	2858	3333	2947	3478	4405	5155	7449	7356	7413	6980	Inf	5440	5061	5036
P16	1721	1332	1609	Inf	Inf	Inf	Inf	3085	3077	3552	3166	3697	4624	5374	3853	3760	3817	3384	Inf	2500	2518	2685

Appendix III – Data Tables

Tables containing data used in the model implementation

A. Types of Aircraft using the Airport of Stockholm-Arlanda

code	name	length	wingspan
100	Fokker 100	35,6	28,1
319	Airbus A319	33,8	34,1
320	Airbus A320	37,6	34,1
321	Airbus A321	44,5	34,1
32S	Airbus A320S	37,6	34,1
332	Airbus A330-200	58,8	60,3
333	Airbus A330-300	63,7	60,3
717	Boeing 717	37,8	28,5
733	Boeing 737-300	33,4	28,9
734	Boeing 737-400	36,5	28,9
735	Boeing 737-500	31	28,9
736	Boeing 737-600	31,2	34,3
738	Boeing 737-800	39,5	34,3
73C	Boeing 737-300W	33,4	30,4
73E	Boeing 737-500W	31	30,4
73G	Boeing 737-700	33,6	34,3
73H	Boeing 737-800W	39,5	35,8
73W	Boeing 737-700W	33,6	35,8
747	Boeing 747-400	70,7	64,4
752	Boeing 757-200	47,3	38,1
753	Boeing 757-300	54,5	38,1
762	Boeing 767-200	48,5	47,6
767	Boeing 767-300	54,9	47,6
777	Boeing 777-200	63,7	60,9
AB6	Airbus A300-600	54,1	44,9
AR8	Avro RJ85	28,5	26,3
ATP	BAE ATP	26	30,6
BEH	Beech 1900D	17,6	17,6
CR2	Bombardier CRJ200	26,8	21,2
CR7	Bombardier CRJ700	32,5	23,2
CR9	Bombardier CRJ900	36,4	24,9
DH4	Bombardier DH8-Q400	32,8	28,4
E70	Embraer E-170	29,9	26
E90	Embraer E-190	36,2	28,7
ER4	Embraer ERJ 145	29,9	20
F50	Fokker 50	25,3	29
J31	Jetstream 31	14,4	15,9
M80	MD-80	45,1	32,9

M81	MD-81	45,1	32,9
M82	MD-82	45,1	32,9
M90	MD-90	46,5	32,9
S20	SAAB 2000	27,3	24,8
SF3	SAAB 340	19,7	21,4

B. Airlines operating at the Airport of Stockholm-Arlanda (summer 2010) [37]

code	name	terminal
2N	NextJet	3, 5
2Q	Air Åland	5
4P	Viking Airlines	5
4U	Germanwings	2
5R	Karthago Airlines	5
7Y	Flying Carpet Air	5
AB	Air Berlin	2
AF	Air France	5
AY	Finnair	5
B2	Belavia Belarusian Airlines	5
BA	British Airways	5
BT	Air Baltic	5
CA	Air China	5
CO	Continental Airlines	5
DC	Golden Air	4
DL	Delta Airlines	5
DY	Norwegian	2, 4
ET	Ethiopian Airlines	5
FI	Icelandair	5
FV	Rossiya Airlines	5
HG	Niki	2
IB	Iberia	5
IR	Iran Air	5
IZ	Arkia-Israeli Airlines	5
JA	B&H Airlines	4, 5
JK	Spanair	5
JP	Adria Airways	5
JU	Jat Airways	5
JZ	Skyways	3
KF	Blue1	5
KL	KLM	5
LH	Lufthansa	5
LO	LOT	5
LX	Swiss International Air Lines	5
MA	Malev Hungarian Airlines	5

OK	Czech Airlines	5
OS	Austrian Airlines	5
OV	Estonian Air	5
PC	Pegasus Airlines	5
QI	Cimber Air	5
QR	Qatar Airways	5
RB	Syrian Arab Airlines	5
RL	Royal Falcon	5
SK	Scandinavian Airlines	4, 5
SU	Aeroflot Russian Airlines	5
TG	Thai Airways	5
TK	Turkish Airlines	5
TP	TAP Air	5
U2	EasyJet	2
VV	Aerosvit Airlines	5
X9	NextJet	3
XQ	Sun Express	5

C. Destination Airports from Stockholm-Arlanda (summer 2010) [37]

IATA_code	name	terminal
ADB	Izmir Adnan Menderes Apt	5
ADJ	Amman Civil-Marka Airport	5
AGH	Ängelholm/Helsingborg Apt	4
AGP	Malaga	2, 5
ALC	Alicante	2
ALP	Aleppo	5
AMS	Amsterdam	5
ATH	Athens	2, 5
AYT	Antalya	5
BCN	Barcelona Apt	2, 5
BEG	Belgrade	5
BEY	Beirut	5
BGO	Bergen	2, 5
BKK	Bangkok Suvarnabhumi International Apt	5
BLE	Borlänge/Falun	3
BLL	Billund	5
BRU	Brussels Airport	5
BUD	Budapest	2, 5
CDG	Paris Charles de Gaulle Apt	5
CGN	Cologne/Bonn Apt	2
CHQ	Chania	2
CPH	Copenhagen Kastrup Apt	2, 5
DBV	Dubrovnik	5

DOH	Doha	5
DUB	Dublin	5
DUS	Düsseldorf International Airport	5
EDI	Edinburgh	5
EPU	Parnu	5
ESB	Ankara Esenboga Apt	5
EWR	Newark Liberty International Apt	5
FAO	Faro	2
FCO	Rome Fiumicino Apt	5
FRA	Frankfurt International Apt	5
GEV	Gällivare	3
GOT	Göteborg Landvetter Apt	4
GVA	Geneva	2, 5
HAD	Halmstad	3
HAM	Hamburg Airport	5
HEL	Helsinki	2, 5
HFS	Hagfors	3
IKA	Tehran Imam Khomeini International Apt	5
IST	Istanbul Ataturk Airport	5
JFK	New York J F Kennedy International Apt	5
JKG	Jönköping	3
KBP	Kiev Borispol Apt	5
KEF	Reykjavik Keflavik International Apt	5
KID	Kristianstad	3
KLR	Kalmar	4
KRF	Kramfors/Sollefteå	3
KRN	Kiruna	4
KSD	Karlstad	3
LED	St Petersburg Pulkovo Apt	5
LGW	London Gatwick Apt	5
LHR	London Heathrow Apt	5
LIN	Milan Linate Apt	5
LIS	Lisbon	5
LJU	Ljubljana	5
LLA	Luleå	4
LPA	Las Palmas	2
LYC	Lycksele	3
MAD	Madrid Barajas Apt	5
MAN	Manchester International Apt	5
MHQ	Mariehamn	5
MLA	Malta	5
MMX	Malmö Airport	4
MSQ	Minsk International Apt 2	5
MUC	Munich International Airport	2, 5
MXP	Milan Malpensa Apt	2, 5

MXX	Mora	3
NCE	Nice	2, 5
OER	Örnsköldsvik	4
ORD	Chicago O'Hare International Apt	5
OSD	Åre/Östersund	4
OSK	Oskarshamn	3
OSL	Oslo Gardermoen Airport	2, 5
OUL	Oulu	5
PEK	Beijing Capital Apt	5
PMI	Palma De Mallorca	2, 5
PMO	Palermo	2
PRG	Prague	5
RIX	Riga	5
RNB	Ronneby/Karlskrona	4
SAW	Istanbul Sabiha Gokcen Apt	5
SDL	Sundsvall	4
SFT	Skellefteå	4
SJJ	Sarajevo	5
SPU	Split	5
SVO	Moscow Sheremetyevo International Apt	5
SXF	Berlin Schönefeld Apt	2
TAY	Tartu	5
TKU	Turku	5
TLL	Tallinn	5
TLV	Tel Aviv Ben Gurion International Apt	5
TMP	Tampere	5
TOS	Tromso	5
TRD	Trondheim Vaernes Airport	5
TUN	Tunis	5
TXL	Berlin Tegel Apt	2, 5
UME	Umeå	4
URE	Kuressaare	5
VAA	Vaasa	5
VBY	Visby	3, 4
VHM	Vilhelmina	3
VIE	Vienna	2, 5
VXO	Växjö	3
WAW	Warsaw	5
ZRH	Zurich Airport	5

References

Websites, statistics, official documents and others (excluding Wikipedia)

[1] *ICAO - Annex 14 to the Convention on International Civil Aviation - Aerodromes Vol.1: Aerodrome Design and Operations. Fourth Edition, July 2004*

[2] *Average kerosene prices - http://www.nyserda.org/energy_information/nyeph.asp*

[3] *Arlanda Airport - http://www.arlanda.se/upload/dokument/Flygmarknad/Facts_2010.pdf*

[4] *Arlanda Airport - Local regulations at aircraft parking stands - http://www.arlanda.net*

[5] *Arlanda Airport - Preferred runways*
http://www.lfv.se/AIP/AD/AD%202/ESSA/ES_AD_2_ESSA_en.pdf

[6] *TITAN Project - Current situation analysis - http://www.titan-project.eu/index.php/library*

[7] *CAST/ICAO Common Taxonomy Team - Phases of Flight*
http://intlaviationstandards.org/Documents/PhaseofFlightDefinitions.pdf

[8] *Notes from an interview with pilot Paul Gaede*

[9] *Lufthansa Hubs- http://www.lufthansa.com/us/en/Our-hubs-in-Frankfurt-Munich-and-Zurich*

[10] *Lufthansa Annual Report 2010*

[11] *ICAO Engine Exhaust Emissions - http://www.caa.co.uk/docs/702/3GE057_01102004.pdf*

[12] *EUROCONTROL - Airport Collaborative Decision Making (A-CDM) - Introduction*
http://www.eurocontrol.int/airports/public/standard_page/APRI_Projects_ACDM.html

[13] *ATRiCS Surface Manager - http://www.atcglobalhub.com/PDF/ATRiCS_SMAN_Folder_A4.pdf*

Scientific papers, books and presentations

[14] *A. Erdmann, A. Nolte, A. Noltemeier and R. Schrader - Modeling and Solving an Airline Schedule Generation Problem*
Annals of Operations Research, Volume 107, Numbers 1-4, 117-142

[15] *J. Abara, "Applying Integer Linear Programming to the Fleet Assignment Problem"*
Interfaces 1989; 19: 20-28

[16] *Lufthansa Symposium 2008 – Aircraft Routing slides*

[17] *C.P. Medard and N. Sawhney, "Airline Crew Scheduling from Planning to Operations"*
European Journal of Operational Research, Vol. 183, No. 3, 2007, pp. 1013-1027.

[18] *Pieter Buzing, Lian Ien Oei, Léon Planken, Cees Witteveen, "CAED D1: Literature Survey on Temporal Decoupling", Delft University of Technology, Eurocontrol, 2007*

[19] *Pim van Leeuwen, "CAED D2: Modeling the Turnaround Process; the Coordinated Airport through Extreme Decoupling", NLR, Delft University of Technology, Eurocontrol, 2007*

[20] *Rigas Doganis, "Flying off course: The economics of international airlines", page 4, Routledge, 2002*

[21] *E. W. Dijkstra, "A note on two problems in connexion with graphs" Numerische Mathematik 1: 269–271, 1959*

[22] *A. Marin and J. Salmerón, 2008, "Taxi Planner Optimization: A Management Tool", Journal of Aerospace Engineering, Vol. 222, 1055-1066*

[23] *John R. Podlena and Keith G. Joshi, "Automated Optimization of Airport Efficiency Using High-Fidelity Simulation", 2006*

[24] *J. Matousek and B. Gaertner, "Understanding and Using Linear Programming", Springer 2007*

[25] *M. Dorigo, V. Maniezzo and A. Colorni, "Ant System: Optimization by a Colony of Cooperating Agents", IEEE Transactions on Systems, Man, and Cybernetics - Part B Cybernetics, Vol. 26, No 1, February 1996*

[26] *M. Dorigo and L. M. Gambardella, "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem", IEEE Transactions on Evolutionary Computation, Vol.1, No.1, 1997*

[27] *José L. Santos, "K shortest path algorithms", University of Coimbra, 2006*

[28] *J. Current, C. ReVelle and J. Cohon, "The hierarchical network design problem", European Journal of Operational Research 27 (1986) 57-66, North-Holland*

[29] *C. Barrett, K. Bisset, M. Holzer, G. Konjevod, M. Marathe and D. Wagner, "Engineering Label-Constrained Shortest-Path Algorithms", 2008*

[30] *David Karger and Evdokia Nikolova, "Exact Algorithms for the Canadian Traveler Problem on Paths and Trees", MIT Computer Science & AI Lab, 2007*

Wikipedia permanent links

[31] *Genetic Algorithm – retrieved on 2011-11-20*
http://en.wikipedia.org/w/index.php?title=Genetic_algorithm&oldid=460952229

[32] *Aircraft maintenance checks – retrieved on 2011-10-11*
http://en.wikipedia.org/w/index.php?title=Aircraft_maintenance_checks&oldid=447828403

[33] *Scandinavian Airlines (SAS) – retrieved on 2011-10-11*
http://en.wikipedia.org/w/index.php?title=Scandinavian_Airlines&oldid=454081175

[34] *Linear programming – retrieved on 2011-10-31*
http://en.wikipedia.org/w/index.php?title=Linear_programming&oldid=458076098

[35] *Gallon – retrieved on 2011-10-11*
http://en.wikipedia.org/w/index.php?title=Gallon&oldid=453848822

[36] *Graph (mathematics) – retrieved on 2011-10-11*
http://en.wikipedia.org/w/index.php?title=Graph_(mathematics)&oldid=454489646

[37] *Stockholm-Arlanda Airport – retrieved on 2011-10-11*
http://en.wikipedia.org/w/index.php?title=Stockholm-Arlanda_Airport&oldid=453710080

[38] *Knot (unit) – retrieved on 2011-10-11*
http://en.wikipedia.org/w/index.php?title=Knot_(unit)&oldid=451077447

[39] *Dijkstra's algorithm – retrieved on 2011-10-11*
http://en.wikipedia.org/w/index.php?title=Dijkstra%27s_algorithm&oldid=454569329

[40] *A* search algorithm – retrieved on 2011-10-11*
*http://en.wikipedia.org/w/index.php?title=A*_search_algorithm&oldid=454906517*

[41] *Runway – retrieved on 2011-10-13*
http://en.wikipedia.org/w/index.php?title=Runway&oldid=452186744

[42] *Single European Sky ATM Research (SESAR) – retrieved on 2011-10-23*
http://en.wikipedia.org/w/index.php?title=Single_European_Sky_ATM_Research&oldid=446074244

[43] *Floyd-Warshall algorithm – retrieved on 2011-10-23*
http://en.wikipedia.org/w/index.php?title=Floyd%E2%80%93Warshall_algorithm&oldid=453820106

[44] *g-force – retrieved on 2011-10-26*
http://en.wikipedia.org/w/index.php?title=G-force&oldid=456642592