University of Sheffield

Optimal scheduling of planes at an airport

Luziyang Chen

Supervisor: Joab R Winkler

A report submitted in fulfilment of the requirements
for the degree of MSc in Advanced Computer Science

in the

Department of Computer Science

September 9, 2019

Declaration

All sentences or passages quoted in this report from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations that are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this project and the degree examination as a whole.

Name: Luziyang Chen

Signature:

Date: 08/09/2019

Abstract

The aircraft landing scheduling problem (ALSP) and the airport gate assignment problem (AGAP) have been an important and challenging problem for a long time. The aim of this project is to design strategies to schedule the planes as they approaching an airport and allocate the departure gate to them after they land. This project designed a system to solve these problems. The key points of the system are the branch and bound algorithm and backtracking algorithm for ALSP and AGAP problem respectively. The branch and bound algorithm use the value of the aircrafts waiting expenses to optimize the function. Also, it uses the feasible landing time interval as the constraint. The backtracking algorithm is an optimization algorithm. In addition, the test results show that these algorithms are more directive and they accelerate the computing process. Meanwhile, the efficiency of them has met up with the requirements.

Acknowledgements

I would like to thank Dr. Winkler for giving me opportunity to work on this project, and his guidance. I also would like to thank my friends for their help.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the rapid development in economics, the globalization has become changed the framework of business. More and more transnational enterprises have to conduct their business globally. As a result, the aviation business has become an important part of the economic. The number of airlines has increased rapidly in recent years. However, this larger number of flights may cause some problems. Mahmud demonstrated that a large number of aircraft departures and landings within a short period has the possibility to cause risk air traffic [11]. If this problem occurs, the planes may be delayed or canceled. These circumstances affect the schedules of passengers and have a bad influence on the profit of airline companies. To deal with these problems, the airline companies have to plan a reasonable framework for their airlines. A good framework is able to improve service efficiency and reduce expenses [12]. The framework contains two essential issues [12]. The first issue is the airline management which manages the routes and time schedule of different flights. The second issue is the air traffic control, including the aircraft landing schedule. The aircraft landing schedule is to generate a landing plan for the planes approaching the airport. The landing schedule affects the expenses of airport and airline companies. As a result, a good landing schedule is an essential factor affecting profit. Meanwhile, for the airport, there is a contingency problem after the aircraft land, which is the departure gate assignment problem. In some circumstances, some flights are connected to next flights letting passengers transfer to other destinations. Thus, the passenger needs to get off the planes as quickly as possible. If passengers do not get off on time, they may miss the next plane and affect their schedules.

1.1 Aims and Objectives

The aims of the project are to design a system to generate the schedule for the planes approaching the airport and the corresponding departure gate assignment. The system contains two main parts. The first part is modeling the aircraft landing schedule problem and designing the algorithm to calculate the schedule according to the model. The second part is to design another algorithm to allocate the departure gate to the landed planes. The objectives of this project are to design a feasible and stable landing order for the planes

firstly. This order will manage the landing order automatically. Secondly, the algorithm has to consider some real factors and adjust the sequence based on these factors. For example, a passenger is ill, or the plane is late, and there are many passengers who must transfer to other aircraft. These planes have a priority to land. The third objective is to design an algorithm to allocate the departure gate stably and efficiently. The algorithm has to consider the different processing speed of different departure gate types.

1.2 Overview of the dissertation

This dissertation will introduce the system designed for the project. This article has seven main chapters. The first chapter is the introduction which is this chapter. The second chapter is a literature review. This chapter will introduce background information research. It shows the main factors affecting the landing sequence. It also shows the previous work to solve the aircraft scheduling problem and the airport gate assignment problem. The third chapter is the requirements and analysis. This chapter will analyze the requirements of the project and the ethical issues of the project. Meanwhile, this chapter will convert the requirement into the detailed function. The fourth chapter is named as design. This chapter will introduce the detailed design, including the model and algorithm design. The fifth chapter is implementing and testing. It explains the code design in the program, including the data structure, strategy, and procedures. This chapter also displays the test results of the system and the comparison with other algorithms. The sixth chapter is the result and discussion. It summarizes the finding of the project, the fulfillment of the goals, and further work. The last chapter is the conclusion.

Chapter 2

Literature Review

2.1 Previous research

Yu et al. proposed an algorithm named cellular automata optimization for the aircraft landing scheduling problem (ALS) [14]. This algorithm is aimed to optimize the First-Come-First-Server strategy (FCFS), which is the most commonly used method to deal with the aircraft landing scheduling problem in airports. However, the aircraft landing scheduling problem has been proved to be an NP-hard problem which means the problem will become more and more complex and difficult as the number of planes increases. The First-Come-First-Server strategy is an effective solution for the limited number of aircraft, but for a large number of planes at the busy time this strategy may not generate the landing schedule in a short time and the schedule maybe not the best solution. Yu et al. believe that FCFS is effective and easy to implement by using cellular automata optimization [14]. In real circumstances, the first plane detected by the airports radar may not still arrive at the airport in the first order because of some factors like wind speed or the engine power of the aircraft. As a result, Yu et al. decided to divide this optimization problem into two parts [14]. The first part is to optimize the landing order of aircraft. The second part is to optimize the exact landing time of each landing plane based on the landing sequence. The cellular automata optimization algorithm is used to provide a landing sequence for the first part. It applies one-dimensional cellular automation to the FCFS model to optimize an effective landing order after the airport generate the order by FCFS model [14]. At this phase, the algorithm would simulate the arrival time considering the real-time factors. If the estimated arrival time is equal to the result of the cellular automata optimization algorithm, this plane will keep the original landing order in the system. If the original estimated arrival time is later than the result of the cellular automata optimization algorithm, the system will let the plane land ahead of schedule. For the planes that would delay, the system would postpone the landing order. The figure 2.1 shows the flow chart of the cellular automata optimization algorithm. The optimization of the second part is consisting of a genetic algorithm and some other optimization algorithm such as using the cellular automata model to simulate the landing. Yu et al. believe that cellular automata model is a dynamical model [14]. The

optimization algorithm using this model to simulate the very dynamical movement in the airport. This model is able to simulate the final results and compute the landing time of aircraft base on the sequence. Yu et al. also run lots of tests to compare with other aircraft landing schedule algorithms including LP-based tree search, a heuristic method, ant colony optimization, scatter search and bionomical algorithm [14]. The test results show that these algorithms usually cannot generate a suitable answer in a short time period. For the cellular automata optimization, it can generate the solution in a very short period.

Mahmud et al. suggested using Embedded Flower Pollination algorithm (EFPA) to solve the aircraft landing schedule problem [11]. Embedded Flower Pollination algorithm consists of the flower pollination algorithm (FPA) and the Runway Balance Strategy (RBS). This strategy is aimed to optimize the service efficiency of the airport runways by designing the landing order. In Mahmuds strategy [11], it takes many real conditions into consideration. For instance, the different models of aircraft including small, large and heavy planes. Mahmuds theory is to limit the total deviation of the plan landing time and the real landing time. Because of the deviation of the plan and the real situation will cost many resources. This algorithm has two preconditions. The first precondition is different models of aircraft has different interval time which means after one aircraft landed there is a period that other aircraft are not able to land. Another precondition is that other aircraft cannot land on the same runway in a period after one landed on that runway. Mahmud divided the embedded flower pollination algorithm into two steps. The first step of the algorithm is to initialize every possible landing sequence. It will simulate the landing order by assigning aircraft to any available runways. This step repeats until the possible solutions are produced. Then, the algorithm would share these solutions like honey bees share their experience of nectar resources and bio-tic pollination. To find out the best solution, this algorithm uses the mutation operator of differential evolution to search the better results [11]. Mahmud et al. also tested the algorithm [11]. Tests compared the EFPA with other algorithms in the respects of computation time and efficiency. The EFPA algorithm shows good performance in those tests [11].

Zhou et al. claimed that aircraft landing problem had been proved to be an NP-hard problem [16]. They declare that the kernel of the aircraft landing problem is to find out the most suitable landing schedule for the planes to land on difficult runways and the sum of the corresponding landing time. Their model has two criteria. The first criterion is named window which represents the time interval between the earliest and latest landing time of the aircraft to the assigned runway. The second criterion is named separation time which represents the time interval between two planes is allowed to land by the airport control tower. Meanwhile, the aircraft may not land at the airport on time actually. Thus, they designed penalization for both the early landing and the landing after the scheduled time. On the base of these conditions, Zhou et al. proposed a mathematical formulation of this problem [16].

$$F = \min \sum_{i=1}^{n} \vec{t_i}\vec{c_i} + \vec{T_i}\vec{C_i} \tag{2.1}$$

The window and separation time can affect the cost of landing. The propose of this formula is to minimize the cost of early and late landings. They also regulated a set of constraints to fit the model to the real situation. To solve the problem, Zhou et al. apply the flower pollination algorithm into the model [16]. This algorithm is similar to Mahmuds solution. The algorithm needs to initialize the number of runways. Secondly, the algorithm will calculate all the suitable solutions firstly. The last step which is also the most important part is to optimize the previous solutions to find out a better solution. The flower pollination algorithm uses a method called context cognitive learning to filtrate the solutions [16]. The algorithm will search for better choices which means the shortest time interval of the best landing time and the real landing time. It will search for the better nodes to replace the original nodes among their neighbors. The algorithm will iterate this process until very nodes have been searched and the solution at this time is the final solution. Zhou et al. stated that the solution might have the problem of local optimal [16]. It may not be the best choice but the solution is also an effective schedule. Meanwhile, their simulations and tests prove that the flower pollination algorithm can generate this schedule very quickly and can deal with a very large number of aircraft.

Chougdali et al. proposed a real-time scheduling system to optimize the First Come First Server strategy (FCFS) [6]. They applied four algorithms into their system and the system can use its inference engine to analyze the feasibility, constraints and efficiency of those algorithms. Then, the inference engine would choose the most suitable algorithm to deal with the task. The four algorithms they used are Rate Monotonic (RM), Deadline Monotonic (DM), Earliest Deadline First (EDF) and Least Laxity First (LLF) [6]. The RM algorithm is suitable for independent tasks with deadlines while unsuitable for sporadic tasks. The DM algorithm is similar to the RM algorithm and very easy to implement. The EDF algorithm has the highest efficiency for the tasks with deadlines but it is very hard to implement. The LLF is very suitable for a single CPU and has good performance when processing multi-processors architecture. The real-time scheduling system accepts a sequence of aircraft as input. Then, the inference engine selects an algorithm to generate a landing schedule considering the rules and factors from the database. Finally, the system outputs the landing time, costs and sequence. Chougdali et al. also propose an expert system for the aircraft landing schedule problem [6]. The system consists of an expert interface, a referred user interface, a knowledge base or rules base, an inference engine, a fact base and a journal of reasoning. The expert interface is designed for the experts of aircraft management. They can control the aircraft manually at this interface. The referred user interface allows the air traffic controller to enter the sequence of aircraft and also shows the results of aircraft landing schedule from the system. The knowledge base or rules base restore all the algorithms and rules of scheduling. The inference engine is used to pick the most suitable algorithm by artificial intelligence. The fact base is the intermediate used by the inference engine. Finally, the journal of reasoning is used to record the former results. Chougdali et al. believe that this system is very suitable for practical application [6].

Bai and Zhang suggested a new model for the airport which only has one runway and

used the genetic algorithm to optimize the results [3]. The new model overcomes the flaw of traditional scheduling methods which did not consider the different flying ability of aircraft. The aim of this model is to improve the aircrafts velocity. The aircraft arrive at the airport in different directions and height. Thus, the speed and the estimated landing time is different because of these factors. The aircrafts velocity represents these criteria. The algorithm would adjust the direction to reduce the time between the plane detected by radar and landing on the runway [3]. Then, Bai and Zhang use a genetic algorithm (GA) to improve the results of the model [3]. The figure 2.2 shows the process of a genetic algorithm. The genetic algorithm is a search heuristic which motivated by the natural evolution. To accelerate the algorithm, Bai and Zhang add a fitness function to each gene [3]. The bigger value of fitness function has, the more conflicts between genes have. The fitness function represents the quality of solutions. After setting this function, the algorithm will pick some nodes in different solutions randomly. If the value of fitness function is very large, these nodes will not go into the next generation and those suitable nodes will combine with other suitable nodes to generate the next generation. This process will iterate until the algorithm gets the optimal solution. However, the solutions of the genetic algorithm may not be the perfect solution but it can generate the solution very quickly even there is a large number of aircraft. As a result, Bai and Zhang suggest using genetic algorithm when the number of aircraft is very large [3]. Their tests prove that the results of this algorithm become steady with the number goes bigger than 50.

Bouras et al. investigated the airport gate assignment problem (AGAP) [5]. They studied many mathematical formulations on AGAP. They first looked into the integer linear programming formulations (IP) to this problem. It considers this question an integer programming model. The aim of this model is to reduce the passengers walking distance and luggage transporting time. The second model Bouras researched is the binary integer programming (BIP). This model takes the time factors into consideration. The third model is named mixed-integer linear programming (MILP). The model considers the time interval of one airport gate after serving an aircraft. However, this model may have a problem that system may assign the same gate to two aircraft at the same time. The mixed-integer nonlinear programming is to minimize these conflicts [5]. But in real situations, there are many restricted conditions. Thus, they suggested a model called binary quadratic programming. It takes the landing time, size of plane and size of airport gate into consideration and uses a greedy algorithm to generate the solution. Because the number of the airport gate is limited. The number of the gate cannot be very large. In this situation, the greedy algorithm is an effective method to compute the optimal solution. The greedy algorithm will search for the most effective node at every branch. After the traversal of the whole branches, the algorithm will generate the optimal solution for the airport gate assignment problem.

Ding et al. proposed two different solutions to the over-constrained airport gate assignment problem [7]. Their objective of airport gate assignment problem is to maximize the service efficiency of the airport gate and minimize the passengers walking distance. Ding designed a model with four airport gates. When all the gates are occupied, the passengers will be

assigned to the apron. They applied the greedy algorithm into this model which is the first solution. The algorithm sorts the aircraft by arrival time and then assigns the gate one by one. If the gate are all busy, the plane will land on the apron which is far away from the airport. However, only using the greedy algorithm is not able to solve the over-constrained problem. Ding et al. introduced the simulated annealing approach (SA) and applied the neighborhood search method of SA into their algorithm [7]. In the simulated annealing framework, the algorithm will generate the solution created by the greedy algorithm firstly. Then, it will set the annealing temperature. This temperature can control the number of iterations. After the iterations start, the algorithm will begin the neighborhood moving. There are three types of neighborhood moves in the simulated annealing framework, including Insert Move, Exchange I Move and Exchange II Move [7] These movements are to move the flights to different airport gates. The Insert Move is to move one flight to another gate. The Exchange I Move is to exchange two flights and their gates. Exchange II Move is to exchange two consecutive flights and their gate. These movements may improve the effectiveness of airport gates and may also decrease it. Therefore, after every movements the algorithm will compute the cost and efficiency in the framework. If the value of cost and efficiency decrease, the algorithm will subtract the annealing temperature to reduce the iterations. Otherwise, the value of annealing temperature remains. The iteration stops when the temperature reached the setting value and generate the final solution. The test results of this solution show that the performance of simulated annealing is not always better than the greedy algorithm. However, the simulated annealing can generate the solution faster than other algorithms [7].

Khakzar and Rahmani suggested using harmony search and NSGA-II algorithms to solve the airport gate scheduling problem [2]. The model designed by Khakzar and Rahmani considers the following factors. The first factor is the layout of the airport, including the layout, planning period and the time interval of the gate maintenance. The second factor is the size of aircraft including the size, the number of passengers and the processing time. Thirdly, the gate can only process one flight and is not allowed to breakdown. The last factor is the size of the airport gate. In this model, the objective function is to assign the available gate to the landed aircraft as soon as possible [2]. Then, they used the NSGA-II algorithm. The progress of this can be divided into six steps [2]. The first step is initialization. Secondly, the algorithm checks the feasibility and perform the fast non-dominated sorting approach [2]. In the third step, it will generate the next population. The fourth step is to mix the two generations. Fifthly, the algorithm checks the feasibility and implement the fast non-dominated sorting approach. The last step is to loop step 3, 4 and 5 until reaching the maximum loop times. The solution produced by harmony search is similar to NSGA-II. But harmony search uses the harmonys concept instead of chromosome [2].

Etschmaier et al. proposed a system for the airport and airline companies and they claim that this system can grow in an evolutionary process [8]. They studied the previous work about the airline schedule and aircraft landing schedule. They find out that the models of these questions can be formulated by mathematics methods and can be solved by objective functions based on the type of aircraft, operating costs, and some other restrictions.

Although these methods can generate the optimal solutions, they cannot solve the aircraft scheduling problem. Because many significant choices in the real situation are made by humans. Meanwhile, computers and algorithms may not consider all the real conditions on account of some factors affecting the operating costs keep changing. In order to find a balance between the computer and human, Etschmaier et al. invented this system [8]. There are three key elements in the system. The first element is a set of database. The second factor is a man-machine interactive environment. The last element is the clear definition as well as the allocation of responsibilities [8]. The computers responsibility is to choose the most suitable and effective algorithm according to the real conditions. After the computer generates the simulated results, the airport controller can adjust the schedule based on those results as well as their experiences. Etschmaier et al. also state the directions for future work [8]. They believe the situation in the air may become more complex to deal with because of the increase of new airlines. Meanwhile, the model of aircraft scheduling problem needs to adapt to the dynamic conditions.

Matthew E.Berge and Hopperstad propose a method to deal with problems of dynamic aircraft, including capacity assignment, models and algorithms [4]. Their model is formulated based on the landing costs. The objective of this model is to reduce the cost as many as possible. They also apply the dynamic network into the framework. The dynamic network has the capacity of dealing with the dynamic change of aircraft. Because in the normal conditions, the aircraft will not arrive at the airport in the same order as they appeared on the airport radar. The order will change all the time on account of weather and other conditions. The dynamic network can show the movements in real time. Matthew E.Berge and Hopperstad also suggest two heuristic algorithms [4]. The first algorithm is a sequential minimum cost flow method [4]. This algorithm can only deal with problems with two types of aircraft. The main function of the algorithm is to reduce the problem to a single commodity minimum cost problem [4]. The second method is named as delta profit method (DELPRO). Delta profit method is an optimization algorithm. It begins with finding the feasible nodes and paths in the network. At each feasible node, the algorithm can select a better path which takes less cost to replace the original path. The algorithm will traverse all the nodes and paths and finally optimize the solution. Matthew E.Berge and Hopperstad also test these two methods to verify the accuracy and efficiency. The results prove that the accuracy of the methods has reached up to 99.9%. Meanwhile, these methods solved the test samples in less than 4 minutes.

Ahmadbeygi et al. analyze the potential for delay propagation in the airline network [1]. The delay propagation is caused by mechanical problems, weather conditions, the ground holds and other conditions [1]. These elements can affect the taking off and landing time. Therefore, ground control has to postpone the flights. In a network structure, this postponement will affect all the afterward flights. This phenomenon is called delay propagation. The delay propagation can cost many resources and affect the profits of airline companies. Ahmadbeygi et al. create a propagation tree in their framework [1]. With the propagation tree, they can better understand and analyze how a root delay can propagate through the

network. In the propagation tree, the first node is the first delay flight. If the ground control can eliminate the delay by bringing forward next flight, the node will not generate new branches. Otherwise, the first branch will generate a branch to the second node which represents the second delayed flight. The whole propagation tree is created by the rules above. To analyze and evaluate the tree, Ahmadbeygi et al. suggest focusing on the maximum depth and maximum magnitude of the propagation tree [1]. The maximum depth represents the entire delay caused by the first plane. The maximum magnitude represents the total number of flights affected by the first delayed plane.

Barnhart studied how to optimize the model for the integrated flight scheduling and fleet assignment when the airport is under congestion [12]. Barnhart searched a large number of materials and found out that the airport congestion and flight delays lead to a huge cost to the airline companies [12]. In order to analyze the problem, Barnhart presents a mixed-integer linear optimization model for the airport congestion and flight delays problem [12]. The models objective is to reduce the possibility of congestion and delays by changing some flights at peak period to the off-peak hour. The mixed-integer linear model is mainly used to predict the profit corresponding to the number of flights. The airline companies will adjust the flights according to the model.

Goyal and Vin present a fair airport scheduling algorithms for aircraft scheduling problems [9]. They apply the rate controlled service discipline (RCSD) into the algorithm. The fair of fair algorithm reflected in this rate controlled service discipline. This discipline can find a balance between rate and delay. Goyal and Vin invent a new method to find the balance by only allocate rate or separate the rate and delay [9]. In order to implement this method, they add a rate regulator and an assistant service queue into the data.

2.2 Summary

All the previous work on the aircraft landing scheduling problem (ALS) indicate that a good solution to the aircraft landing scheduling problem needs to consider the factors of the real situation as many as possible and optimize the landing sequence to minimize the cost of early or late landing. The deviation between the real landing and the schedule will waste many resources. As a result, the main objective is to improve the efficiency of the landing schedule. Chougdali et al. [6] and Yu et al. [14] optimized the First-Come-First-Server strategy. However, this is only a basic strategy. For a large number of aircraft, this method may take a very long time to process. Mahmud et al. [11] added the runway as a condition into its model and mixed the flower pollination algorithm with the runway balance strategy. Zhou et al. also proposed the flower pollination algorithm to optimize the First-Come-First-Server strategy. The real time and dynamic change have also been taken into consideration. The literature about the airport gate assignment problem (AGAP) is all aimed to improve the service efficiency of airport gates. Khakzar and Rahmani [2] proposed a thoughtful model and applied NSGA-II algorithms to optimize the results. The previous research also indicates that modeling the problem, producing the primary result and optimizing the result is the

best approach to solve the airport gate assignment problem.

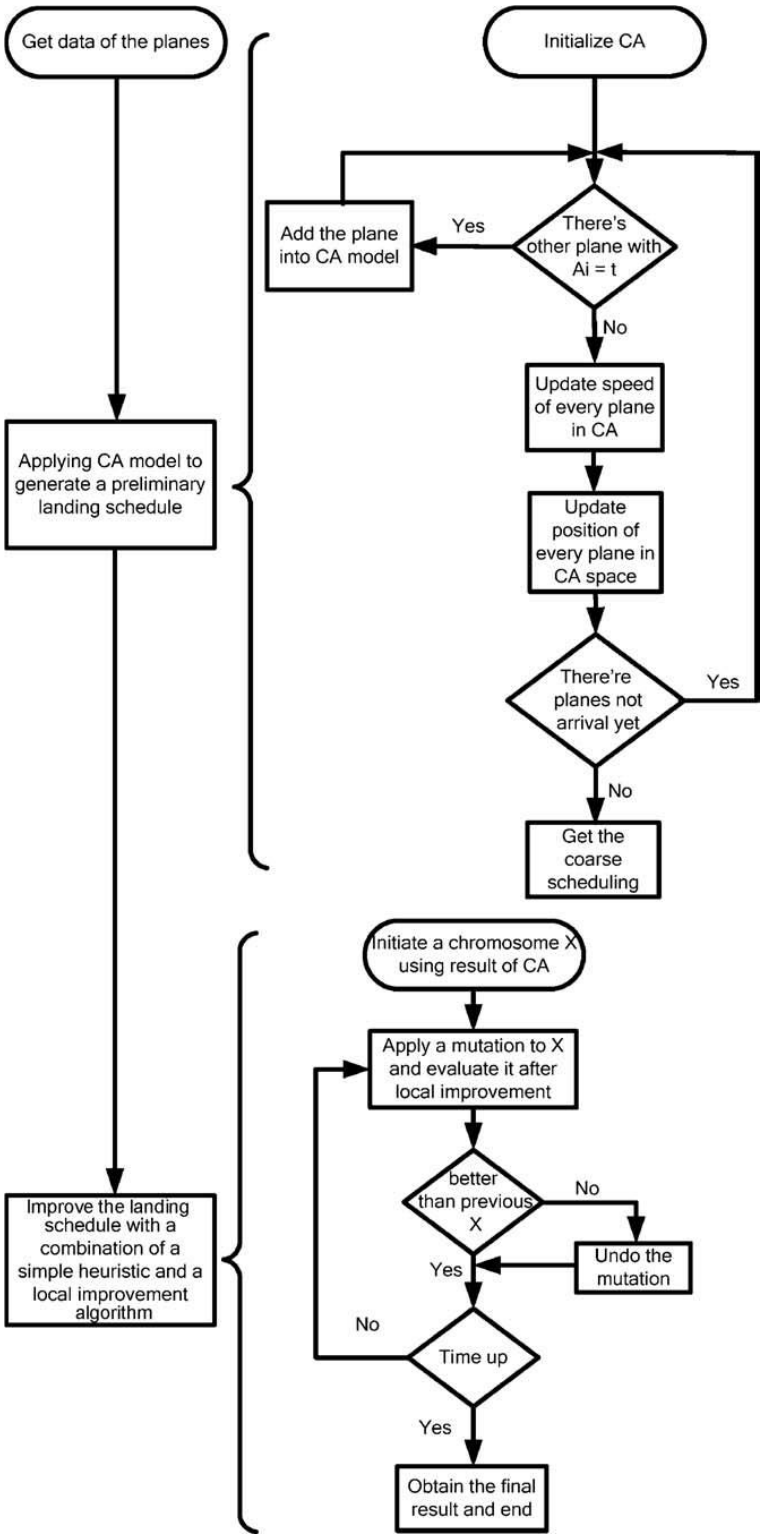

Figure 2.1: Flow chart of the cellular automata optima. Reproduced from with the permission of the copyright owner.

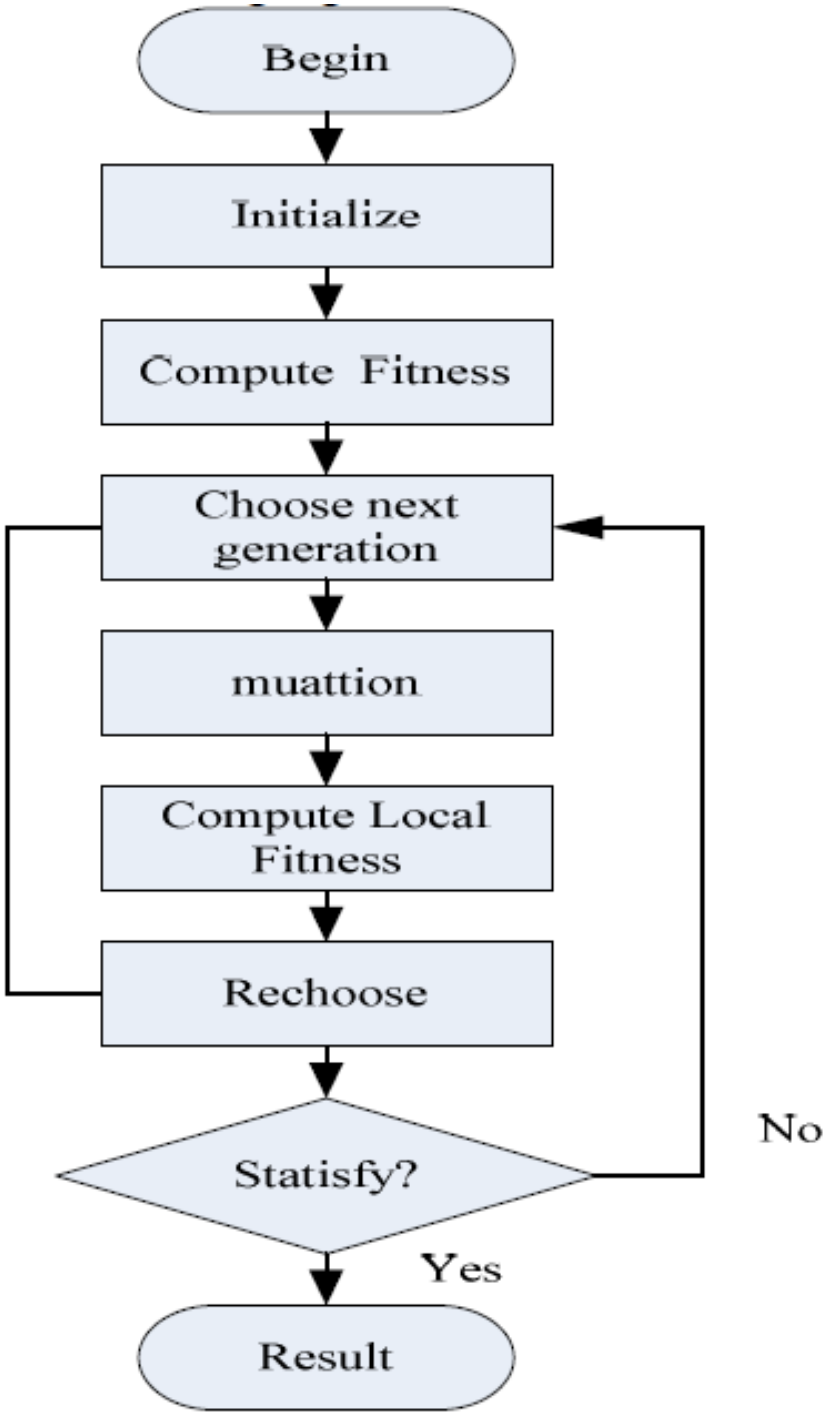

Figure 2.2: Flow chat of the GA algorithm. Reproduced from with the permission of the copyright owner.

Chapter 3

Requirements and analysis

The brief introduction of the project can be summarized as follows. The project is concerned with the scheduling of planes as they approach an airport. The planes approach the airport in a specified order, but it may be necessary for some planes at the front of the queue to be delayed while planes behind them are given priority. Because some emergency may occur. For example, a passenger is ill or the plane is late and there are many passengers who must connect with other aircraft. Also, the distance behind the different model of the plane should be considered. For instance, the distance behind the double-decker A380 is much greater than the distance behind a smaller plane with only two engines. Meanwhile, the airport gate should be assigned to the planes after they landed. The assignment of the airport gate should consider the efficiency and the type of the gate.

The aim of the project is designing a system to schedule the planes as they approach an airport. The model must consider two main factors. The first factor is the order of planes approaching and the second factor is how to assign the airport gate to the landed planes. The objectives of the project are to design a reasonable and stable landing order for the planes firstly. This order will manage the landing order automatically. Secondly, a function which can adjust the landing order is needed for emergencies. Like the examples listed above, a passenger is ill or the plane is late. The third objective is to design an algorithm to manage the airport gate efficiently. The algorithm should consider the different types of the airport gate. In real conditions, the airport has different kinds of gates which has different capacity. For instance, the small gate can only be assigned to the small planes such as Boeing 737. However, the big gate which can be assigned to the big plane like A380 can also be assigned to other kinds of planes. As a result, the main objective is to generate a flexible plan to allocate the airport gate.

3.1 Detailed Requirements Analysis

3.1.1 The first requirement analysis

In order to implement the first requirement, the progress of aircraft landing must be fully analysed. The first requirement is an aircraft landing schedule problem. For the aircraft which appear on the aircraft radar, it needs the controller of air traffic to assign the landing route and the landing time for them. In the real situation, all the aircraft is approaching the airport at a different speed. Meantime, the aircraft still has the distance between the airport. If the ground control arranges the landing schedule to the plane, the plane has to accelerate or decelerate to control itself to land on the runway at the specific time. Because in many circumstances, the plane will not arrive at the airport by the scheduled time without changing the cruising speed r12. Barnhart has proved that the main factor the influences the profit of airline companies is the air traffic congestion including the delays [12]. The air traffic congestion and delay always force aircraft to accelerate or decelerate in the air. The accelerate or decelerate will consume aircraft fuel and increase the abrasion. Although the abrasion is very slight, the repairing and maintenance still cost plenty of money. Meanwhile, the fuel used to accelerate or decelerate is another essential expenditure. According to Barnharts research, the criterion of a landing schedule is the total cost of every plane to land on the airport according to the schedule [12]. As a result, the landing schedule has to reduce the deviation between the landing time and best landing time as many as possible to reduce the cost. Therefore, the evaluation of a schedule can base on the number of total landing cost. For the real aircraft landing problem, there must be an interval time between two continuous aircraft. This time interval is caused by the airflow created by planes and the transferring time for the plane moving from the runway to the airport gate or parking apron. The time interval is related to the aircraft model. For example, the airflow behind the Boeing 747 is much greater than the Boeing 737. Therefore, the safe distance behind the Boeing 747 is longer than the Boeing 737 and the time interval is longer than Boeing 737. This time interval is corresponding to the distance mentioned in the brief project introduction. This is all the elements needed to solve the aircraft landing schedule problem.

The second step to implement the system is modeling. A model must be designed to convert this problem from a real situation to the math problem. Modeling is an essential part of this project. All the essential elements should be taken into consideration. Firstly, the basic elements represent the variable in the real situation. To formulate the total cost of landing, the following elements should be taken into the model. The first factor should be the type of aircraft. The different type of aircraft has different performance parameter. Different type of aircraft has different cruising speed which costs the least fuel. Meanwhile, to describe the expenditure change corresponding to the accelerating or decelerating of aircraft, they need a function to formulate the expenditure. The speed is corresponding to the x-axis and the expenditure is corresponding to the y-axis. The function can be a linear function or a power function depending on the parameters of aircraft. At the cruising speed, the expenditure reaches its minimum value. Apart from the cruising speed, the model should consider the

maximum speed and the minimum speed. According to these two borderline-speeds, the model can get the earliest and the latest landing time. These two values set the available landing time interval. Because in normal conditions, not every plane can be scheduled to land at their best landing time which is the estimated time arriving at the airport if the plane keeps the cruising speed. Therefore, every aircraft must land in this time interval. After all, the model needs at least three elements to describe the arrival time which are the best arrival time, earliest arrival time and latest arrival time. These three times is depending on cruising speed, maximum speed and minimum speed. Another factor is the interval time caused by airflow mentioned above. This element is named as crashing interval and each kind of aircraft has its unique crashing interval. In order to implement the requirement of dealing with the emergency, the model has to consider the special situation. The model can add a binary flag element which represents whether there is an emergency on the aircraft such as fuel shortage, engine malfunction. If the value of flag equals 1 one, this plane has a high priority to land. The algorithm has to put this flight in front of the queue.

The third step is to design the objective function and use an algorithm to calculate the optimal solution. The objective function is to calculate the summary of expenditure according to the deviation time and the cost of accelerating or decelerating. Therefore, the objective of the algorithm is to minimize the value of the objective function by arranging aircraft to land by a suitable sequence. Zhou et al. have proved that this problem to be an NP-hard problem [16]. Liu et al. explained the concept of NP-hard problem (non-deterministic polynomial) [10]. They take an example to describe this question. Assuming there are some different intermediate nodes between point A and B. Therefore, there are plenty of different suitable routes to go to point B. The problem is to find a route which cost the fewest money. Apparently, it is easy to find a solution but it cannot ensure that this is the cheapest route. Because it is impossible to traverse all the routes. The previous work on aircraft landing schedule problem which is an NP-hard problem used some effective algorithms including first-come-first-serve algorithm, genetic algorithm and flower pollination algorithm.

The first-come-first-serve algorithm is the most direct method. This method can only generate a feasible schedule. The method is designed to deal with the plane in the same sequence as they appeared on the airport radar. Apparently, this method cannot guarantee the efficiency and the expenditure may be very high. But this method is able to generate the schedule in a very short time because of its simple process mode. The first-come-first-serve algorithm is a very basic method [15]. It provides the simplest idea for the problem. If this method be optimized appropriately, it may become a suitable algorithm.

The genetic algorithm is a common method to solve the NP-hard problem. Sirohi et al. applied this algorithm to solve their problem and displayed the procedures of implement [13]. The genetic algorithm is studied from the theory of evolution. In nature, one population keeps evolving to adapt to the environment. The population always evolve new characters to survive. For example, the giraffe evolved the long nick to eat the leaves on the high trees. The core idea of the genetic algorithm is similar to this. Applying the algorithm to the aircraft landing schedule problem, the environment is the total expenditure of landing. The

environment requires the expenditure to be as low as possible. The schedule is the population and the cost is the populations character in this problem. Only the population with low expenditure can survive in the environment. The algorithm initializes various population with different characters at the beginning. These populations has to copulate to generate the next generations and evolve the specific character. At this stage, the algorithm needs to evaluate every generation. If the character of one generation is not evolved as required, this generation will be discarded. However, the normal copulation is too random and it may generate many useless generations. As a result, the genetic algorithm takes a special method to copulate which is called roulette wheel selection method. It can generate the next generation very quickly by selecting two parents with suitable character at one time. The algorithm will let the population keep copulating and evolving until meeting the terminal conditions. The terminal conditions can be the number of iteration or the character becomes stable or the character has met requirements. The schedule generated by the genetic algorithm is always a feasible and suitable solution to the requirements.

The flower pollination algorithm is another feasible algorithm to deal with the NP-hard problem. The flower pollination algorithm is studied from the bees pollinating between flowers. When bees pollinating between flowers, their will exchange the information after they pollinated. They will tell each other the shorter route between flowers. Applying this model to the NP-hard problem. The flowers are corresponding to the nodes in the problem. The objective is to find the shortest or the least cost route. The procedure of the flower pollination can be divided into the following steps. The first step is that the algorithm has to get every possible landing sequence. The program will simulate the landing sequence by traverse method or other feasible. This step repeats until the possible solutions are produced. Then, the algorithm would share these solutions like honey bees sharing their experience of nectar resources and biotic pollination. The algorithm will traverse all the nodes and replace some routes between nodes based on the last steps information.

These three methods are common solutions to the NP-hard problem as well as the aircraft landing schedule problem. These three methods have their advantages and disadvantages individually. The first-come-first-serve is a direct method, but it is time-consuming and the solution is not always meet the requirements. Nevertheless, this method provides very useful thinking. The genetic algorithm is a feasible and mature method. It can always generate a suitable schedule. But the genetic algorithm is most suitable for only some kinds of NP-hard problem with the specific restriction of expenditure. The aircraft landing schedule problem is to find the least-cost solution. Therefore, the genetic algorithm may not be suitable for the project. The flower pollinating algorithm is also feasible. But the concept of this algorithm is a little bit complex. As a result, this algorithm is a bit difficult to implement.

3.1.2 The second requirement analysis

In order to implement the second requirement, the progress of the airport gate assignment must be fully analysed. The second requirement is an airport gate assignment problem. After aircraft landed, the project requires the airport to assign a gate for them. The passengers can

get off the planes by those gates. Bouras et al. demonstrate that the efficiency of airport gates is essential to the airport [5]. The poor efficiency is very likely leading to airport congestion. Too much time spending on the airport gate has a high possibility leading to the delay of the following flights. Because if all the airport gates are busy, the following landing flights have to wait for an unoccupied gate. In this project, there is no parking apron. Therefore, passengers aboard have the only choice to get off by airport gates. Meanwhile, the project requires the airport has different types of gates which is accord with the real situation. The different types of airport gates have different processing speed. Meanwhile, the aircraft have different types which have different passenger capacity. In order to evaluate the gate assignment, an objective function should be designed. According to the different passenger capacity of aircraft and the different processing speed of airport gates, the objective function should calculate the cost of these gates to process all the landed aircraft.

After designing the objective function, the variables needed to be modeled. First of all, the type of plane has been modeled in the first requirement. The passenger capacity has to be taken into consideration. For the airport gate, it must contain the type of gate, processing speed and the cost of gate per minute. Comparing with the aircraft landing schedule, the airport gate assignment problem is also an NP-hard problem.

The next step to element the requirement is designing the algorithm. The objective of the algorithm is to find out the assignment of airport gate which cost money as less as possible. The previous work dealing with the airport gate assignment problem utilized the genetic algorithm and simulated annealing algorithm. The genetic algorithm has been introduced in the first part of requirement analysis. The simulated annealing algorithm is another method to deal with the NP-hard problem. This algorithm is studied from the principle of solid annealing. Solid annealing is a process when a solid cooling down slowly after being heated to a very high temperature. The molecules in the solid move acutely at the high temperature. With the temperature going down, the movements of molecules become orderly and the interval energy drops to the bottom. The interval energy represents the value of the objective function. The algorithm is to stimulate the annealing process. When the interval energy achieves the minimum value, the value of the objective function is also the smallest. To simulate the process of temperature dropping and the movements of molecules, the algorithm has to find a feasible solution firstly. Then, it needs to find another solution. The algorithm calculates the value of the objective function (interval energy) and replaces the high-cost solution with a low-cost solution. This is the main process of simulated annealing. The algorithm keeps selecting alternative solutions and replaces the origin solution with a better solution until the number of iterations reaches the restriction. The number of iterations is the temperature in the solid annealing process. This algorithm can generate a solution in a very short period. However, the quality of the solution cannot be guaranteed.

Although the simulated annealing algorithm is not the best choice, the step to find the feasible solution is worth learning. The data sample in this project may not be very large. Therefore, an algorithm which can generate the lowest cost schedule should be considered preferentially.

3.2 Evaluation criterion and Testing

The aim of this project is to design feasible and efficient algorithms producing the landing sequence and assigning airport gate to the landed plane. Thus, the feasibility is one of the most important evaluation criteria. The landing sequence must have no conflicts and reliable. The other evaluation criterion is the performance of the algorithm. Both algorithms objective function is to calculate the total cost. Therefore, the value corresponding to the schedule and assignment is the evaluation criterion. If these two algorithms are successful, the cost should not be very high. Meanwhile, the performance of the algorithm includes efficiency. The time spent on computing should not be very long.

The program will run several tests to show the efficiency and feasibility of the algorithm. The test data will use a set of cases which have simulated plane performance data. The test for aircraft landing schedule algorithm will use the real data to compare with the schedule arranged by ground control and other algorithms. The test for the airport gate assignment algorithm will use the data set in advance to verify the feasibility.

3.3 Ethical, Professional and Legal Issues

This project does not need to review the ethical problems. The algorithms designed for aircraft landing scheduling and airport gate assignment in this project are only for the theoretical problems. The model designed int his project does not cover all the conditions in the real world. Therefore, the algorithm will not be applied to real situations. According to the BCS code of conduct, I must consider the public health, privacy, security and well-being of others and the environment. In this project, I must consider real conditions into my model as many as possible. I must respect and value alternative viewpoints and seek, accept and offer honest criticisms of work. Meanwhile, I shall develop my professional knowledge and skills, maintaining awareness of technological developments, procedures and standards that are relevant to my field. Finally, I must accept my duty to uphold the reputation of the profession and not take any action which could bring the profession into disrepute such as plagiarism.

Chapter 4

Design

This chapter will describe the detail design of the project. The chapter is divided into three parts. The first part describes the detailed model design of the aircraft landing schedule problem as well as the algorithm to deal with the objective function. The second part explains the model design and the algorithm of the airport gate assignment problem. The last part introduces the programming language, compiler software and the environment.

4.1 The algorithm for the aircraft landing schedule problem

According to the analysis in the last chapter, the model should contain the necessary elements and create the objective function. In this project, the model only considers the static situation. The project chooses static conditions because the dynamic conditions require a more complex network to modeling. Meanwhile, the data in the dynamic model keeps changing. Although the dynamic model is more accord with the real situation, it is very difficult to deal with. The static conditions are easier to dispose. The data is static in the model. As a result, the project only considers the scheduling problem for N planes arriving at the airport at the same time in the static conditions. N planes arriving at the airport represent the planes appeared on the radar at the same time.

The different types of aircraft are denoted as TYPE. For different types of planes, they have different parameters including the earliest landing time, the best landing time and the latest landing time. They are represented by t^e , t^b and t^l. The value of i is between 1 and N. Meanwhile, the model has to formulate the cost of accelerating and decelerating. The accelerating cost of plane number i per minute is denoted as S_i. The decelerating cost of plane number i per minute is denoted as H_i. The expenditure change function of accelerating and decelerating corresponding to the time is displayed in the figure 4.1.

To simplify the problem, the model assumes the cost of accelerating is equal to the cost of decelerating. The advantage of this assumption is that it reduces the complexity of computation. In addition, the expenditure function is assumed to be a linear function. This assumption is also for simplifying the complexity. As a result, the cost of accelerating S_i and decelerating H_i is equal to the parameter C_i. The value of i is from 1 to N.

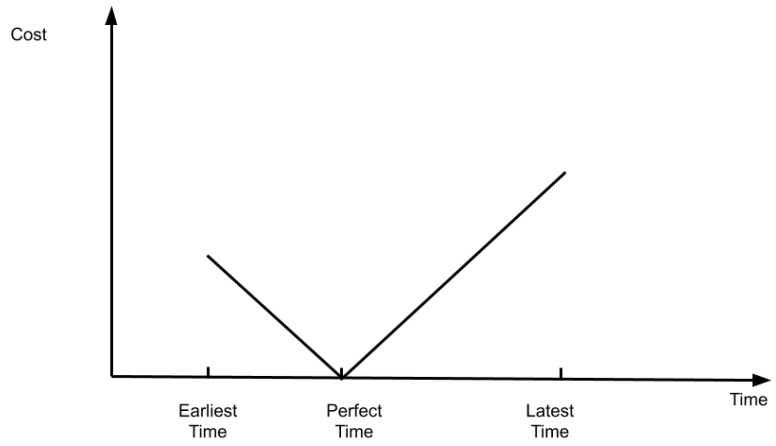

Figure 4.1: The expenditure change function

Assuming the X_i is the estimated landing time, this time is not a fixed value. The value of X_i can be adjusted by the algorithm. The final value of X_i is the landing time for plane i. According to the objective function, the algorithm has to reduce the time interval between the best landing time and the estimated time. Therefore, the time interval in the model is denoted as d_i. $d_i = |X_i - t^p|$. The objective function is the product of the cost C_i and the time interval d_i. The following formula is the objective function.

$$\min \sum_{i=1}^{n} C_i d_i = \min \sum_{i=1}^{n} |X_i - t^p| \tag{4.1}$$

$$t_i^e \leq X_i \leq t_i^l \qquad i = (1, 2, ..., n) \tag{4.2}$$

$$X_i - X_k \geq l_i \qquad i, k = (1, 2, ..., n) \tag{4.3}$$

Formula 4.2 and 4.3 are the restrictions of the objective function. Formula 4.2 represents that the estimated time must be earlier than the latest landing time and later than the earliest landing time. Formula 4.3 represents that the time interval between two continuous planes to land must longer than the crashing time interval of the first plane.

This article selects the branch and bound algorithm to solve the objective function. The branch and bound algorithm are a systematical method to search the solution space. For the aircraft landing schedule problem which is an NP-hard problem, it has plenty of feasible solutions. These solutions make up the solution space. The branch and bound only generate the state space tree of solutions based on some restrictions. In other words, the algorithm

generates the optimal solution by computing a fraction of feasible solutions. In the state space tree of solutions, the algorithm will deal with the possible nodes. Every live node which represents the feasible schedule has only one chance to become an E- node. An E- node can generate branches to every possible new node. From these new nodes, the algorithm will abandon those nodes which are impossible to generate the optimal or feasible solutions. The rest of the new nodes will be added into the table and become the live nodes. The algorithm then chooses another node from the table and let it become the new E- node. These procedures keep iterating until the table of live nodes is empty or the optimal solution has been generated.

There are two common methods to generate the new E- node. The first method is the first-in-first-out method. The other method is the lowest-cost method. This article selected the lowest-cost method. The first-in-first-out method takes the nodes from the table in the same order of these nodes been added into the table. This is a simple way to dispose these nodes. But it may be very time-consuming if the order of nodes is not well. The least-cost method gives every node a corresponding value of cost. In this project, the aim is to find the lowest cost solution. The lowest-cost method creates the live-node table by the minimum root heap method. By this method, the next E- node must be the lowest-cost live node.

The branch and bound algorithm can accelerate the process of generating a feasible solution. For the planes approaching the airport, if the project analyzes all the feasible sequence to find the lowest-cost solution, it will certainly generate the best solution. But it costs too much time and the planes may not have enough time to wait. The branch and bound algorithm accelerate the process by adding some restrictions to limit the sequence deviating from the optimal direction. To apply this algorithm to the project, we assume that the plane whose best landing time t^p is the closest to the current time ($Curtime$) can be put in the front of the sequence. If this plane is planned to land, other planes have to wait in the air which cost the resources. Meanwhile, there is a crashing time after a plane landed which do not allow another plane to land in this time interval. Thus, the total landing time is the current time plus the crashing time. The algorithm computes other planes direction of cost change during this planes landing time to estimate the direction of total cost change and then decide whether this plane should land before other planes. If the total cost increases, this plane cannot land before other planes and the algorithm have to consider other planes first. On the contrary, if the total cost decrease, this plane is allowed to land. The data structure used in the algorithm is described below:

public String AT; public int ET; public int PT; public int LT; public int IT; public int Cost; public int pass_number; public int emergency;

AT, ET, PT, LT, IT represents the type of aircraft, earliest landing time, perfect landing time, latest landing time and crashing time respectively. The cost is the expenditure of planes waiting in the air. The emergency represents whether the plane has an urgent situation.

The description of the algorithm:

(1) Divide planes approaching the airport into different groups according to the types of planes. In each group, sort the planes in order of the arrival time.

(2) Select all the first arrival planes in each group and put them into a new temporary group. In this group, sort the planes in order of the arrival time.
(3) If the temporary group is empty, jump to step (7). Otherwise, compute the time difference of every plane in the temporary group. The time difference is $|CurTime - t^p|$.
(4) Select the plane with the lowest time difference to land. This plane is named as First and its crashing time is l_I.
(5) $CurTime = CurTime + l_i$. For all the planes in the temporary group except the First, compute the time difference which is $|NextTime - t_i^p|$. According to the results, there three following possible situations.
1) For the planes whose waiting cost is higher than the First, if the value of cost is reducing, First can land before other planes.
2) For the planes whose waiting cost is higher than the First, if there is one plane whose value of cost is rising, then consider these two following situations.
I. If the $CurTime$ is larger than this planes perfect landing time, let this plane land first. Jump to step (6).
II. If the $CurTime$ is less than this planes perfect landing time, compute the waiting cost of all the planes whose waiting cost is larger than First while the value of time difference is rising. Let the lowest-cost plane among these planes to land first. Jump to step (6).
3) For the planes whose waiting cost is higher than the First, if there is more than one plane whose value of cost is rising, then consider these two following situations.
I. If the $CurTime$ is larger than these planes best landing time, the plane with higher waiting cost should land first. Jump to step (6).
II. If the $CurTime$ is less than these planes best landing time, compute the waiting cost of all the planes whose waiting cost is higher than First while the value of time difference is rising. Let the lowest-cost plane among these planes to land first. Jump to step (6).
(6) Save the results into a linked list. Delete the landed plane from the temporary group. Move the next plane from the same group where the landed plane was into the temporary group. Let $CurTime$=$CurTime$. Jump to step (3).
(7) End of the algorithm.

4.2 The algorithm for the airport gate assignment problem

According to the analysis in the last chapter, the model should contain the necessary elements and create the objective function. The conditions have to be considered less than the conditions in the aircraft landing schedule problem. The objective function is to calculate the cost of all the passengers getting off by the airport gate. The total expenditure is the product of time and cost. In this project, there are N airport gates. The type of airport gate is denoted as $TYPE$. Different type of airport gate has different processing speed and cost. The processing speed of gate number i is represented as S_i. $(i = 1, 2, , N)$ The cost of the gate is denoted as C_i. Meanwhile, the aircraft structure created in the first part contains

another element which is the number of passengers (N_P). Thus, the time passengers need to take off is denoted as T_i. $Ti = (N\_P)/S_i$. The formula 4.4 is the objective function.

$$\min \sum_{i=1}^{n} C_i d_i = \min \sum_{i=1}^{n} C_i \frac{N\_P}{S_i} \tag{4.4}$$

The project selects the backtracking method to solve the objective function. According to the results of previous work, there are two alternative methods. The first method is traversing. Traversing will compute the cost of all the possible solution and select the lowest cost solution. Traversing can ensure the solution is the optimal solution. But it is very time-consuming. If there are a large number of planes and gates, there may be billions of solutions. In real situations, planes cannot wait that long on the parking apron. As a result, this method is not suitable for the project. The second optional method is a simulated annealing algorithm. This method has been introduced in the requirement analysis chapter. The advantage of this algorithm is high efficiency. It can generate a feasible answer in a very short period. But this method cannot ensure the quality of the solution. Thus, it is not the best choice for the project. Compared with these two methods, the backtracking method combines the advantages of traverse and simulated annealing algorithm. It can generate an optimal solution in a very short period if the number of planes is not very huge.

The backtracking algorithm is an optimal search method which is also a trial method. The backtracking method searches the solution tree according to the optimal conditions to achieve the solution. However, when the algorithm searches to a certain node and finds that the original choice is not optimal or cannot meet up the requirements, the algorithm will track back and choose the nodes again. The point of a certain state satisfying the backtracking condition is called backtracking point. The data structure used in the algorithm is described below:

public int NO; public int style; public int speed; public int Cost; public int busy;

NO is the number of the airport gate. The style, speed, cost represent the type, processing speed and expenditure of the airport gate respectively. The busy shows that whether the airport gate is occupied. Depending on the real conditions of the project, the algorithm deal with the planes in the same order as they landed. The algorithm is modified from the traversing algorithm and inspired form the backtracking method. It firstly creates a formula to compute the value of the objective function. The algorithm sorts the service order based on this value. Then, the algorithm creates a temporary group. The first N planes will be put into the group. The number N is the number of the airport gates. Afterward, the optimal gate assignment of these planes is generated by traversing the feasible solutions. Meanwhile, the gates status is set to the busy status. Then, the next plane will be put into the temporary group. For this plane, it has to wait for a vacant gate. Thus, the algorithm has to select the lowest waiting cost gate for this plane. After processing all the planes, it will output the final assignment.

4.3 Programming Language and Compiler Software

The project selects C# as the programming language and uses the SharpDevelop 4.4 as the compiler. C# is an object-oriented, high-level programming language released by Microsoft that runs on the.net Framework and.net Core. C# looks surprisingly similar to Java; It includes things like single inheritance, interfaces, almost the same syntax as Java, and the process of compiling intermediate code and then running it. But C# differs significantly from Java. The C# is directly integrated with COM (component object model) and is the backbone of Microsoft's.net Windows network framework. C# is a secure, stable, simple, elegant, object-oriented programming language derived from C and C++. It inherits the power of C and C++ while removing some of their complex features. It combines VB's simple visual operation with C++ 's high operating efficiency. With its powerful operation ability, elegant syntax style, innovative language features and convenient support for component-oriented programming, C# has become the preferred language for .net development. It is also an object-oriented programming language. It allows programmers to quickly write applications based on the Microsoft.net platform, which provides a range of tools and services to maximize the exploitation of the computing and communications space. Thus, the project uses windows as the operating system. Meanwhile, C# allows C++ programmers to develop programs efficiently by calling native functions written in C/C++ without losing the power of C/C++. This is the most important reason why the project selects C# as the programming language and the SharpDevelop is an assorted compiler for this language.

Chapter 5

Implementation and testing

This chapter explains the design of the code and the tests to verify the feasibility and efficiency of the system. The first part introduces the detailed design of code to implement the model and algorithm. The second part introduces the tests and the data used in the test.

5.1 The design of code

The design of code can be divided into four parts including the algorithm of aircraft landing schedule, the algorithm of airport gate assignment, the data input and the user interface.

To implement the aircraft's landing model and the branch and bound algorithm, the project firstly creates an Aircraft class to restore the information of the aircraft. The Aircraft class contains eight elements including the aircraft type (AT) which is a String data type, the earliest landing time (ET) which is an integer data type, the perfect landing time (PT) which is an integer data type, the latest landing time (LT) which is an integer data type, the crashing time interval (IT)which is an integer data type, the waiting cost of aircraft (Cost) which is an integer data type, the number of passengers on the plane (pass_number) which is an integer data type and the emergency situation (emergency) which is also an integer data type. After creating the data structure, a class which is used to sort has been created (class Com). This class compares two planes perfect landing time. Then, the main class for the main body of the algorithm (class Algorithm). For the emergency situations, the project creates the emergencyFirst class. In this class, if there is one planes value of emergency equals to one, this class will put this plane into the front of the list. The list is the final schedule of aircraft landing and will be output at last. The next step is to divide aircraft into different groups according to the types of planes. The lists to store the planes is named as group_airList[k]. The character k represents different types of plane and the value of k is one two three . After that, the tempList is created to store the planes whose perfect landing time are the earliest in their group. The algorithm will calculate the difference between the current time and the perfect landing time of all elements in the tempList. This difference is named as C_P which is an integer data type. Meanwhile, there is a list to store the waiting cost of every plane in the tempList. Then, the algorithm has to select the lowest cost plane from the

tempList which is variable First and record its crashing time interval (IT). The crashing time interval plus the perfect landing time is the value of the NextTime. This NextTime is the earliest landing time for the next time. Afterward, the algorithm will compute the summary of waiting cost of other planes is the tempList. This is for the algorithm to estimate the trend of cost change. If the cost decreases while all the planes can land after the NextTime, the plane First will be added into the schedule list. If there is one planes cost increases while the current time is later than this planes perfect landing time. The algorithm will put this plane into the schedule list rather than the First plane. If there is one planes cost increases while the current time is earlier than this planes perfect landing time, the algorithm has to select the lowest cost plane from all the planes whose cost are higher than the First and their cost changing trend is rising. Then, the algorithm adds this plane into the schedule list rather than the First. If there are more than one planes cost increases while the current time is later than this planes perfect landing time. The algorithm will select the highest cost plane from the tempList and add it to the schedule list. If there are more than one planes cost increases while the current time is earlier than this planes perfect landing time, the algorithm has to select the lowest cost plane from all the planes whose cost are higher than the First and their cost changing trend is rising. Then, the algorithm adds this plane into the schedule list. According to the restrictions above, the algorithm will select one plane to land and add it into the schedule list. The cost of this plane will also be recorded. Afterward, this plane will be removed from the tempList and the next plane from this planes original group will be added into the tempList. Meanwhile, the current time equals to the value of NextTime. The program will loop the procedures until all the planes have been added into the schedule list. Finally, the algorithm will output the landing schedule and the total cost of landing.

To implement the airport gate model and the backtracking algorithm, the project firstly creates an Gate class to restore the information of the airport gates. The Gate class contains five elements including the number of airport gate (NO) which is an integer data type, the type of airport gates (style) which is an integer data type, the processing speed of the gate (speed) which is an integer data type, the cost of the gate (cost) which is an integer data type and the status of airport gate (busy) which is also an integer data type. The state of the gate has two status. One status is that the gate is empty and another status is that the gate is on service. The empty status is represented by number zero while the busy status is represented as number one. The algorithm has to compare the costs of different choices and decides whether it should go back to make a new choice. To accurate the comparison, the algorithm creates a formula. The formula is the summary of the time and the product of the gate cost with a constant. The time means how much time all the passengers need to get off from the plane. The constant is a double data type. The value used in the program is 0.05. This value can convert the cost from integer data to more accurate data. The result of this formula is the criterion to compare. The algorithm begins to compute with the same sequence of landing. It computes the criterion mentioned above and selects the gate of which value is the lowest. When the gate has been selected, the status of the gate will be set to the busy status. After all the passengers get off at this gate, the status will become empty.

To every plane in the queue, the algorithm computes the value of that formula. If there is a less costly value, the algorithm will give up an original choice and go back to the node where have a better choice. After processing all the planes, the algorithm will generate the plan of the airport gate assignment and the total cost of it.

The user interface is designed for the user to input the data and get the final results of the landing schedule as well as the airport gate assignment. For the function of inputting data, there are two methods. The first method is to add the data manually from the Add data button. The Add data menu has two tabs. The first tab is to add the information of planes while the second tab is to add the information of airport gates. Users can add all the necessary elements into the system. The second method is to add the data into a .txt file and the program can read the data from the file. The main interface uses the grid view framework. There are three text boxes with columns. The first and second text box displays information about the planes and the airport gate. The third text box displays the results of the two algorithms. It shows the sequence of the landing and the gate assignment. Meanwhile, there are three simple text boxes to show the total cost of landing, the total cost of gate assignment and the total time spent on the gates. Finally, there are two buttons below these text boxes. One of them is to start computing and another button is to exit the system.

5.2 Testing

The test ran on the personal computer. The operating system is Windows 10. The development software is SharpDevelop 4.4. It is a compiler for the programming language C#. To verify the feasibility and efficiency of two algorithms designed above, the projects used several sets of data. These data are collected from the real cases of the airports ground control. Here is one set of data. The data is displayed in the table 5.1 and table 5.2.

Aircraft Type	Earliest	Perfect	Latest	Interval	Cost	Emergency	passenger
B737-200	26	30	60	10	700	0	104
B737-200	0	0	90	10	700	0	104
B737-200	30	44	105	10	700	0	104
A310-200	43	47	107	10	1000	0	240
A310-200	2	6	66	10	1000	1	240
B747-400	15	18	108	10	1500	0	416
B747-400	37	39	99	10	1500	0	416

Table 5.1: Information of aircraft.

According to the data above, the tests used three different methods to solve the aircraft landing schedule problem and the airport gate assignment problem. The detailed schedule as well as the cost of two problem haven been recorded. These results will be compared to each other. This method can clearly prove which one is the best solution. The first method is to use the branch and bound methods for the aircraft landing schedule problem while the

Number	Type	Speed	Cost
1	1	100	120
2	2	70	70
3	2	50	50

Table 5.2: Information of departure gate.

backtracking method for the airport gate assignment problem. The second method is to use the plans given by the ground controller and the compute the cost of their plans. The third method is to use the first come first serve algorithm to generate the landing schedule and use the traverse algorithm to solve the airport gate assignment problem. The landing sequence and the cost of two problems have also been recorded. The results and the cost is shown in table 5.3.

Aircraft Type	Landing Time	Landing Cost	Gate	Cost of gate
B737-200	0	0	2	104
B737-200	10	4000	1	288
B737-200	20	3000	2	416
A310-200	30	1000	1	124
A310-200	40	1500	2	416
B747-400	50	3000	3	240
B747-400	60	11200	2	104

Table 5.3: The landing sequence and the assignment of departure gates.

Method	Cost
The branch and bound algorithm	22700
Ground control	34000
First come first serve method	42000

Table 5.4: Comparison with other methods.

As the results shown in the table 5.3 and table 5.4, the branch and bound algorithm have a good performance. Firstly, in terms of feasibility, the schedule is workable. There is little difference between the schedule generated by the branch and bound algorithm and the schedule made by the airport controller. Meanwhile, the landing schedule made by the first method meets the restrictions of real situations. There are no planes landing in the time interval of other planes crashing time interval. Also, all the plane land between their earliest landing time and the latest landing time. These two results show that the branch and bound algorithm is feasible to generate the workable plan. Meanwhile, the table also displays the cost and time spent on computing. The results prove that the branch and bound algorithm has the best efficiency among these three methods. The first come first serve method cost the least time while the landing cost is the highest. The plans arranged by the airport ground controller take too much time and the cost is higher than the cost of the branch and bound

algorithm. These two methods prove that the branch and bound algorithm is the best choice among these three methods for the aircraft landing schedule problem. For the airport gate assignment problem, the first and second method is similar to the method to test the landing schedule algorithm. The only difference is the first method using the backtracking algorithm. Meanwhile, the third method uses the traversing algorithm. First of all, the results of the first method are similar to the third method. Meanwhile, the airport gate assignment plan generated by the backtracking algorithm is almost the same as the plan generated by the traversing algorithm. This result shows that the backtracking algorithm is feasible because the traversing algorithm is certain to calculate the optimal solution. As a result, the feasibility of the algorithm has been proved. In respect of efficiency, the backtracking spent the least time among three methods. The artificial method cannot guarantee the time spent. It is longer than the other two methods. The traversing algorithm takes the longest time to search for the optimal answer. However, the backtracking algorithm shows good performance both in the feasibility and efficiency. In summary, the tests prove that these two algorithms are feasible and efficient.

Chapter 6

Results and discussion

This chapter summarizes the results during the period of designing and implementing the project, including the findings and the task achievements of the project. Meanwhile, this chapter also proposes ideas for improving some aspects of the project.

6.1 Results and Findings

During the literature search section, lots of previous work have been researched. Firstly, the problem of aircraft landing schedule has been analyzed. There are two different kinds of model can be applied to the problem. The first model is a static model and another one is a dynamic model. By analyzing the previous work, the static model shows more steady performance in those algorithms based on the static model. Meanwhile, the algorithm complexity of the algorithms using the static model is lower than those algorithms using the dynamic model. Because the dynamic model needs too many parameters and the algorithm have to process all these parameters. Although the dynamic model is more likely to simulate the real conditions, the static model can also deal with some complex situations and the previous tests prove that static model has a good performance. As a result, the project selects the static model. Also, the analysis shows that the aircraft landing problem is an NP-hard problem. Because the number of solutions to the landing schedule can be huge. If there are ten planes approaching the airport, there can be 10! different sequences. Thus, the project should select an efficient algorithm. There are four alternative algorithms including the first come first serve algorithm, genetic algorithm, flower pollination algorithm and the branch and bound algorithm used in the project. The analysis finds out the advantages and disadvantages of each algorithm. The first come first serve algorithm is the simplest method to implement while the other three methods have good efficiency. The genetic algorithm and flower pollination algorithm have good efficiency. They can generate a suitable schedule in a short period. The branch and bound algorithm combine good efficiency with the easy implement.

Secondly, the analysis of the airport gate assignment problem finds out the advantages and disadvantages of the suitable algorithm. The traversing method always generates the optimal while it takes a very long time. On the contrary, the simulated annealing algorithm generate

a feasible answer in a short time period. As a result, the project creates a backtracking algorithm based on the traversing algorithm. The backtracking algorithm can generate a more accurate answer than the simulated annealing algorithm and faster than the traversing algorithm.

Finally, the project created a system to generate the landing schedule and the gate assignment schedule for planes approaching the airport. The system contains the branch and bound algorithm as well as the backtracking algorithm mentioned above. Meanwhile, the system has a user interface for users to add data into the system and display the results of the algorithms. The figure 6.1 shows the user interface of the system.

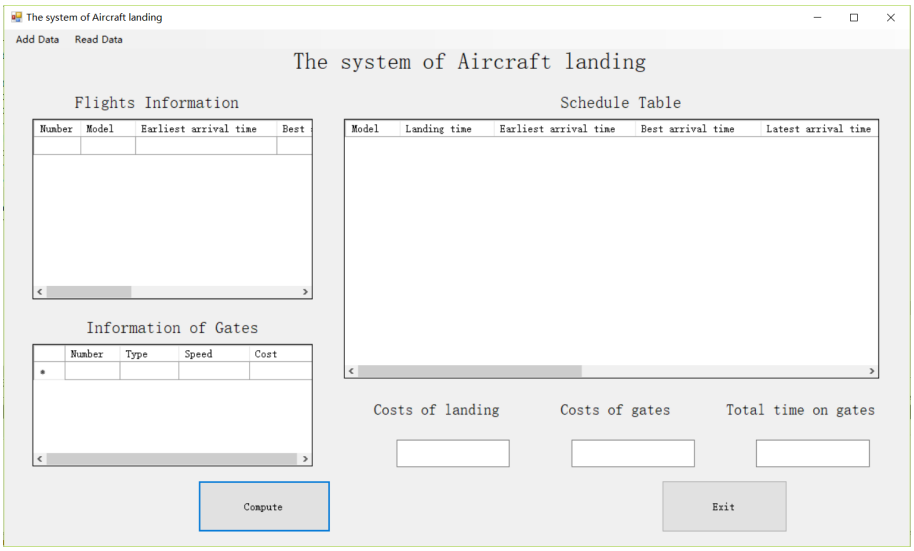

Figure 6.1: The user interface of system

Users can input the information of flights and the airport gate into the system by clicking the add data button in the optional bar. The data is displayed in the two text boxes on the left as the picture shows. By clicking the compute button, the system begins calculating. Afterward, the landing schedule and gate assignment plan will be displayed in the text box on the right. The detailed costs of the plan are shown in the three small text boxes on the bottom.

6.2 Goals achieved

Compared with the requirements of the project, all of the requirements have been met. Firstly, the project requires planes to land in a reasonable sequence. The branch and bound algorithm generate the landing sequence based on the cost of waiting to land. The algorithm generates a feasible and efficient landing schedule. Secondly, the project requires to priority process the planes with emergencies such as the passenger is ill or the plane delays. The system set a parameter to represent the emergencies. If the value of this parameter equals

to one, this plane has the priority to land. Thirdly, the project requires the system should consider the different distance behind the plane according to their types. The system converts this element into the time interval after the plane landed. In this time interval, other planes are not allowed to land. This is the time interval represents the distance behind the plane. Because the distance behind the plane is also a safe distance as well as the safe time interval. Finally, the project requires an efficient and suitable plan for the departure gate assignment. Thus, the backtracking algorithm has been created. This algorithm generates a plan based on the efficiency and expenditure of different departure gates. It is able to generate the lowest cost plan as quickly as possible. Meanwhile, the project has some nonfunctional requirements. These two algorithms need to have good efficiency and the schedule should cost money as low as possible. The tests prove that the branch and bound algorithm guarantee both the efficiency and the feasibility. The backtracking algorithm is certain to generate the lowest cost plan but it may spend lots of time calculating if there is plenty of data. In addition, there are some parts in the system which not quite meet the requirements. The parameters of time are the system are integer data type while the time data type is the best choice. The system uses the integer date type because the integer data type is very easy to deal with in the algorithm. As a result, the time in the final schedule is not in the form of time. It is just a number. This data type needs improvements. In summary, the system achieves most of the goals of the project.

6.3 Further Work

The further work for this project is to optimize the algorithm and the user interface. Firstly, the project can consider the element of the runway. The model in the system only considers one runway. Secondly, the system can consider trying the dynamic model. Although it is complex, it can cover more real conditions. The system now is an experimental system and it used the idealized model. By applying the dynamic model into the system, it may be capable to deal with the real problem as well as being applied into the real life. Thirdly, some elements in the model need to be optimized such as the parameter of time mentioned before. Fourthly, the backtracking algorithm for the airport gate assignment problem can be replaced by other better algorithms. The backtracking algorithm used in the system is adapted from the traversing algorithm. Thus, it is still a time-consuming plan, although it is more efficient than traversing algorithm.

Chapter 7

Conclusions

This project designed a system for planes approaching to the airport. The main parts of this system are two algorithms. The first algorithm is the branch and bound system and the second algorithm is the backtracking algorithm. The first algorithm is to solve the aircraft landing schedule problem. According to the results of previous work, this problem has been proved to be an NP-hard problem. This problem has plenty of potential suitable solutions. Therefore, the optimal solution is very hard to be found among those solutions. The previous work applied a genetic algorithm and flower pollination algorithm to search for the optimal answer. Although, these algorithms are efficient but their process is not direct. They restrain the calculating by constantly exchange nodes (flower pollination) or cross the solutions to generate new solutions (genetic algorithm). These methods take time to iterate and generate the final solution. On the contrary, the branch and bound algorithm uses more directive strategy than other algorithms to accelerate the computing process. This algorithm is to search the optimal answer at every node. It avoids the high-cost solutions. The second algorithm is backtracking algorithm. This algorithm is also a directive method. This method is adapted from traversing algorithm. Because the number of departure gate cannot be very large and the landing sequence has been decided by the first algorithm. But the traversing algorithm is still time-consuming. Thus, the backtracking algorithm optimizes the strategy to exclude high-cost solutions. The previous algorithm applied to this problem is a simulated annealing algorithm. This algorithm is to generate a suitable solution when there is plenty of data. Compared with this algorithm, the backtracking algorithm has better efficiency when the number of samples is not very large. These two algorithms designed in the project have improved some aspects of previous algorithms and met the requirements of project. The results of the test have proved the feasibility and efficiency of two algorithms. Besides, there are some aspects of the system need to be improved. The dynamic framework and more elements can be taken into the model. In summary, the system has fulfilled the project and met up with the requirements.

Bibliography

[1] Ahmadbeygi, S., Cohn, A., Guan, Y., and Belobaba, P. Analysis of the potential for delay propagation in passenger airline networks. *Journal of Air Transport Management 14*, 5 (2008), 221–236.

[2] Bafruei, M. K., Khatibi, S., and Rahmani, M. A bi-objective airport gate scheduling with controllable processing times using harmony search and nsga-ii algorithms. *Journal of Optimization in Industrial Engineering 11*, 1 (2018), 77–90.

[3] Bai, C., and Zhang, X. Aircraft landing scheduling in the small aircraft transportation system. In *2011 International Conference on Computational and Information Sciences* (2011), IEEE, pp. 1019–1022.

[4] Berge, M. E., and Hopperstad, C. A. Demand driven dispatch. a method for dynamic aircraft capacity assignment, models and algorithms. *Operations Research 41*, 1 (1993), 153–168.

[5] Bouras, A., Ghaleb, M. A., Suryahatmaja, U. S., and Salem, A. M. The airport gate assignment problem: a survey. *TheScientificWorldJournal 2014* (2014), 923859–923859.

[6] Chougdali, S., Roudane, A., Mansouri, K., Youssfi, M., and Qbadou, M. New model for aircraft landing scheduling using real time algorithms scheduling. In *2015 Intelligent Systems and Computer Vision (ISCV)* (2015), IEEE, pp. 1–7.

[7] Ding, H., Lim, A., Rodrigues, B., and Zhu, Y. The over-constrained airport gate assignment problem. *Computers and Operations Research 32*, 7 (2005), 1867–1880.

[8] Etschmaier, M. M., and Mathaisel, D. F. X. Airline scheduling: An overview. *Transportation Science 19*, 2 (1985), 127–138.

[9] Goyal, P., and Vin, H. Fair airport scheduling algorithms. In *Proceedings of 7th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV '97)* (1997), IEEE, pp. 257–265.

[10] Liu, Y., Gao, C., Zhang, Z., Lu, Y., Chen, S., Liang, M., and Tao, L. Solving np-hard problems with ¡i¿physarum¡/i¿-based ant colony system. *IEEE/ACM*

Transactions on Computational Biology and Bioinformatics (TCBB) 14, 1 (2017), 108–120.

[11] Mahmud, A., Satakshi, W., and Jeberson, W. Aircraft landing scheduling using embedded flower pollination algorithm. *International Journal of Parallel Programming* (2018), lt;xocs:firstpage xmlns:xocs=quot;quot;/gt;.

[12] Pita, J., Barnhart, C., and Antunes, A. Integrated flight scheduling and fleet assignment under airport congestion. *Transportation science 47*, 4 (2013), 477–492.

[13] Sirohi, R., Singh, A., Tarafdar, A., and Shahi, N. Application of genetic algorithm in modelling and optimization of cellulase production. *Bioresource Technology 270* (2018), 751–754.

[14] Yu, S.-P., Cao, X.-B., and Zhang, J. A real-time schedule method for aircraft landing scheduling problem based on cellular automation. *Applied Soft Computing Journal 11*, 4 (2011), 3485–3493.

[15] Zeller, R., and Deschamps, J. First come, first served. *Nature 420*, 6912 (2002), 138–139.

[16] Zhou, G., Wang, R., and Zhou, Y. Flower pollination algorithm with runway balance strategy for the aircraft landing scheduling problem. *Cluster Computing 21*, 3 (2018), 1543–1560.