Modelling and Solving Decentralized and Dynamic Aircraft Landing Scheduling Problems

A Thesis

In

The Department

Of

Mechanical, Industrial and Aerospace Engineering (MIAE)

Presented in Partial Fulfillment of the Requirements

For the Degree of Master of Applied Science at

Concordia University

Montréal, Québec, Canada

November, 2017

This is to certify that the thesis is prepared:

By: **Mr. Hamidreza Rezaei**

Entitled: **Modelling and Solving Decentralized and Dynamic Aircraft Landing Scheduling Problems**

and submitted as in partial fulfillment of the requirements for the degree of:

Master of Applied Science in (Industrial Engineering)

Complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

Dr. W. Xi	Chair
Dr. A. Bulgak, MIAE	Examiner
Dr. A. Awasthi, CIISE	External Examiner
Dr. M. Chen	Supervisor
Dr. C. Wang	Co-Supervisor

Approved by: ______________________________

Chair of Department or Graduate Program Director

____

Dean of Faculty

Date: ______________________________

Abstract:

Aircraft landing problem (ALP) is considered as a scheduling problem where aircrafts are sequenced and allocated with appropriate time slots. In this thesis ALP problem is investigated where several constraints such as aircraft's landing time windows, minimum separation time and position shifting constraints are taken into consideration. Existing approaches such as optimized solution based methods and heuristic methods to tackle different aspects of the problem are reviewed, and a static mathematical model is studied. The mathematical model is solved and verified using random data generated from simulation. The data are generated based on Pierre Elliott Trudeau International Airport (YUL) in Montreal, Quebec, Canada as well as from relevant data base library. AnyLogic^{tm} software was used to simulate aircraft landing operations in a runway environment. An agent based simulation was designed to include the dynamic event of aircrafts arrivals to the runway system.

In the agent based system, an iterative bidding framework is used to generate flight landing schedule in a decentralized environment. In the decentralized environment, we consider each flight as a self-interest agent competing with other flights to get the most appropriate landing time. The efficiency of the decentralized approach is also studied. The results of the decentralized approach are compared with the centralized ALP solution. The results show that the agent based solution approach is able to generate reasonable landing comparing to optimal aircraft landing schedule from the centralized ALP model.

Acknowledgments:

I would like to thank God whom without him nothing would have been possible. I am pleased to thank following persons who have supported me both technically and emotionally during this study.

I would like to thank Dr. Mingyuan Chen and Dr. Chun Wang, my thesis supervisors, for their supervision, technical support and valuable guidance as well as for their patience and comments that helped me through this research. It was my pleasure to work under their supervision as a master student.

My friend Zhijie for his technical guidance and support and sharing his experiences and knowledge to complete my thesis.

My lab partners Jair, Armaghan, Omar, Cesar, and Yasser Ghahremany for sharing all the pleasant moments during this research.

Last but not least I would like to express my gratitude to my family for their continuous support and love.

Table of Contents

List of Tables:

List of Charts and Figures:

Chapter 1: Introduction and Motivation

In this chapter Aircraft Landing Problem (ALP) is described. The challenges of the problem are discussed. The approaches to tackle the problem and the main contributions of this thesis are provided. Later, the overview of the thesis outline is explained.

1.1 Aircraft Landing Problem (ALP)

According to International Air Transport Association (IATA), demand for air travel increases on average by 6.3% each year (IATA, 2016). This growth in demand results in traffic congestion in landing aircrafts, takeoff aircrafts, taxi way at the runway. To tackle this problem some changes like enlarging the en-route traffic capacity have taken palace resulting in a shifting of the bottleneck from en-route to airports (Soomer, Franx, 2008).

One option to reduce traffic congestion at airports is to build new runways. Building a new runway can be highly costly or impossible due to airport configurations. Another option is to make the best use of the current resources. ALP optimization is an elective tool to find the best solution in minimizing total related overall cost using available resources.

While considering safety factors, Air Traffic Control System (ATC) must take several criteria into account. Runway utilization is an important criteria to be considered in airport management scheme. One way to achieve best usage of runway is to use aircraft landing delay. Since runway scheduling, en-route scheduling, gate assignment and baggage handling are linked with each other, aircraft delays have a great impact on whole airline operations. Furthermore, punctuality is an important interest of airlines companies.

1.2 Challenges and motivation

When schedule landing time of the aircrafts, ATC attempts to establish the minimum separation time between two aircrafts. Because of wake vortex effect depending on aircraft weights and sizes, each aircraft must maintain a standard separation distance from following and to leading aircraft. Wake vortex is caused by warm air from aircraft engine leaving air turbulence behind causing air instability for the following aircrafts. These challenges raise a need for optimized landing schedules meeting safety standards and considering stakeholder interests.

In the literature, most mathematical models for aircraft landing scheduling consider a fixed number of aircrafts that are already in the system ready to land at any time. Some of these models may not consider some critical issues for practical implementation.

Another factor that affects runway scheduling is dynamic events such as aircraft new arrival to the system, weather conditions, poor visibility and unforeseen technical problems. These dynamic events may have a great impact on the scheduling process and should also be considered

1.3 Contributions

This research considers a mathematical model for ALP the mathematical model is based on that in Beasley et al (2000). Various objective functions such as minimizing total aircraft delay time, makespan, and overall flight costs are considered. A position shifting constraint with binary variable is also added to the model. The model was verified with the generated data observed from YUL airport.

A simulation was developed with numerical experiments to investigate the impact of new aircraft arrival to the system and its impact on the optimal solution of the ALP model. The model is designed to reschedule the optimal solution by arriving new aircraft to the runway area and assign the best landing time. The model was tested with several new aircrafts arrival rate including rush hour and regular time at the runway, and the solutions was compared with each other.

The main contribution of the thesis is to develop a negotiation framework for solving ALP problem in a distributed optimization fashion. Considering decentralized environment of the aircraft landing scheduling, an iterative bidding procedure is proposed to schedule landing aircrafts in a wider time horizon where aircrafts negotiate for the best landing time slot. In this thesis the distributed ALP problem is concerned with overall cost reduction of the landing aircrafts. Furthermore, the efficiency of the decentralized ALP model and the mathematical model was tested and compared with each other.

1.4 Outline of the thesis

In the next chapter, the Aircraft Landing Problem (ALP) research literature is reviewed. In Chapter 3, a mathematical model is introduced for solving ALP. In Chapter 4, a simulation model considering dynamic ALP decentralized models is introduced. In Chapter 5, decentralized model for ALP problem is presented and the efficiency of the proposed framework is investigated. Chapter 6 presents the summary and conclusions.

Chapter 2: Literature review

In this chapter, common approaches used to solve aircraft landing problem are explained. Generally, ALP problem solving can be divided into two categories: exact optimization and approximate solutions. Approximate solutions such as heuristics and meta-heuristics are usually used for solving larger instances or when problems need to be solved with a larger instances in real time.

2.1 Optimization Based Solution

In this section, optimization based approaches commonly used to solve ALP problems are introduced.

2.1.1 Mixed integer programming

Mixed integer programming (MIP) is widely used to tackle ALP problems. Solving MIP by using optimization software is often time consuming. MIP models have been used as base models to be solved by other methods such as dynamic programming or heuristic algorithms. Beasley et al (2000) presented a mixed integer model for solving ALP with single runway and extended it to solving multiple runways problem. In his model, minimum separation time between landing aircrafts and time windows with preferred landing time are considered. To minimize deviation between target and actual landing time a solution method based on linear programming relaxation was developed. The author also introduced a heuristic method to determine the upper bound on optimal solution for tightening the time windows for each aircraft. Dirk Briskorn (2014) developed several MIP models for both single and multiple runways. The author solved each model in polynomial time but is not fast enough to be used for real time application. Farhadi and Ghoniem (2014) studied Doha International Airport and proposed a heuristic method based on a MIP model considering First Come First Served (FCFS) sequencing approach with focus on delay and fuel cost reduction. Artiouchine (2007) proposed a MIP model considering different holding patterns with airport tower controllers making certain aircrafts to wait for certain amount of time before landing.

2.1.2 Dynamic Programming

Dynamic programming (DP) is a method which breaks the problem down into a sequence of simpler problems and tackle the problems one by one using the answer of one stage to get the answer of the next stage. It continues solving the problem until the best optimal solution is reached. It divides the problem into different inter related stages where each stage is solved independently from the next stage. DP models can be solved either backward or forward. In the forward approach the first stage is the initial stage to be solved while in the backward approach the first stage is considered as the final stage of the problem.

DP method is a common approach for solving scheduling and sequencing problems, and is widely used in solving aircraft landing and scheduling problem Psarafti (1976) first used a backward dynamic algorithm for ALP solving problem in a single runway considering constraint position shifting. The author first considered a number of aircrafts waiting for landing where the preferred landing time for all aircrafts is equal to zero. Likewise he proposed a dynamic programming algorithm for travel sales man problem.

He considered each aircraft as a node and minimum time separation as a link between the nodes. The goal of his approach is to find aircraft sequencing where either aircrafts sum landing time or latest landing time is minimized. Then he extended the model into two runways case without considering CPS.

Bayen (2004) considered ALP as a single job shop problem with a given processing time and deadline where the objective functions are minimizing the sum of starting time, and minimizing the time of the last assigned job (makespan). He considered a holding time for a single class of aircraft then solved the model with a combination of dynamic programming and linear programming.

Brentell (2006) developed a forward DP model of Psarafti by considering earliest arrival time for each aircraft. As an objective function, Brentell considered minimizing sum of landing times. Each aircraft based on their classes should be sequenced in a non-decreasing order of earliest arrival time. He also proposed a forward DP for holding time where all aircrafts are circling in several holding patterns (stack) to be landed. Brentell (2009) studied Stockholm Arlanda airport comparing FCFS sequencing policy with several approaches. He implemented the algorithm into discrete event simulation for ALP problem in a single runway. Using several statistical methods, he analyzed the effect of sequencing algorithm on the ALP model, delay-sharing strategy, arrival rate and wake-vortex mix.

Balakrishnan (2006) solved the aircraft landing and takeoff scheduling under constraint position shifting, and introduced an acyclic directed graph where nodes represent a sub sequence of aircraft landing order. She considered maximum position shifting which is solved as a shortest path problem using DP algorithm. According to FCFS order aircrafts are labeled *1* to *n*. There are n stages representing the final position of an aircraft in the final sequence. Each stage includes several nodes with the length of min {*2k* + *1*, *P*} where k is maximum position shifting and p represents one stage. For example, for *k*=1 and *p*=5 then length of subsequence is 3 with the ending point of 5. An arc (*i* , *j*) from node *i* in stage *p* to node j in stage *p+1* represents minimum time separation between the last and the first aircraft in their subsequence. In the subsequence the first {*2k*, *p*} aircrafts of node *j* is the same as the last min {*2k*, *p*} aircrafts of node *i*. For instance, for *k*=1 and *p*=5 there is a link between subsequence (2-4-6) in stage 5 and (4-6-5) in stage 6 (Figure 2.1).

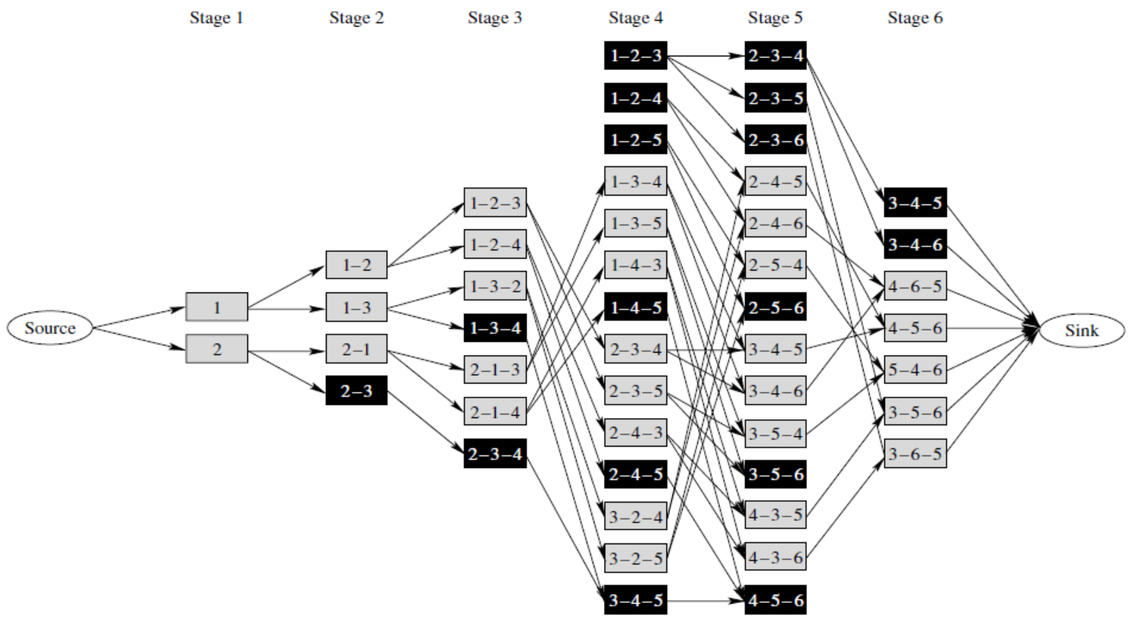


Figure 2.1.Directed graph network for *P*=6 and *k*=1 (Balakrishnan, 2006)

Nodes in each stage are all possible subsequences with the maximum position shifting of 1. Black nodes do not belong to any final sequence and is pruned from the network. The main objective function is to minimize makespan of the CPS graph.

$$T_j^* = \max\{\, e(j)\,, \min(T_i^* - \delta_{ij})\} \;\; , \qquad\qquad i \,\epsilon\, p(j) \colon T_i^* \; \leq L(i)$$

The objective function finds the shortest path from the beginning(source) to each stage *j* to the final node (sink) with respect to earliest *e(j)* , and latest *L(j),* landing time of aircraft *j*, and the minimum separation time between node *i* and *j*, δ_{ij} . p *(j)* represents set of subsequences that are feasible to node *i*. She also solved the problem for minimizing the sum of delay of all aircrafts.

Lieder et al (2015) developed a dynamic programming algorithm for large instances with different classes of aircraft in a multiple independent runways case. The model is solved for a number of aircrafts up to 100 with a positive target time and time window. As an objective function minimization of total delay subject to necessary separation time between two landings are also considered.

2.1.3 Branch and Bound:

Ernest (1999) proposed a branch and bound method by developing a basic Mixed Integer Programming (MIP) model of landing problem. The method minimizes penalty cost for aircrafts that land ahead or after their target time. The model is subject to the time window and necessary separation time constraints. They solved the problem by a set of partial landing orders of aircrafts. A space search heuristic to determine an upper bound for the branch and bound method is also proposed.

Beasley et al (2000) used linear programming tree search to solve the mixed integer binary model. He showed that their model can deal with several issues such as: different objective functions, precedence constraint and runway workload balancing.

2.1.3 Branch and Price:

Wen et al (2005) first developed a column generation exact algorithm for aircraft landing problem. They proposed a mixed integer programming model, and reformulated it as a set partitioning problem. Using the set partitioning problem an exact branch and price algorithm is developed. Applying Beasley et al (2000) instances they tested their algorithm for 50 aircrafts and four runways. The optimal solutions with less than 500 columns generated are produced in a reasonable computation time.

Ghoniem (2015) formulated a mixed integer problem which is solved by Branch-&-Price algorithm that outperforms in the standard solvers. In his model they both considered aircraft landing and takeoff time for a multiple runways problem.

2.2 Heuristic Based Solutions:

Heuristic based solutions are designed to find a close answer to the optimal solution in a quick timing when the problem is too large or needs to be solved in a real time. In this section most common heuristics approaches for the ALP are introduced.

2.2.1 Evolutionary Algorithm:

Genetic Algorithms (GAs) is one the most common approaches originated from evolution theory which is based on natural selection process and genetics. It mimics biological evolution. The basic process of GAs is divided into several steps: Initialization, Fitness, Selection, Recombination and Evaluation. In the process, decision variables are converted to set of string binary digits (gens), and an initial population (parents) is generated. By applying genetic operators such as: mutation, cross over and selection new generation of solution (children) is generated from the initial population.

In the crossover, the children will inherit/reject the common genes from the parents in an equal chance. In mutation, new solution is reached by filliping binary variables in an opposite way of the population with a certain probability, and selection operator chooses new generation based on the fitness function.

Pinol and Beasley (2006) considered multiple runways for aircraft landing problem. They presented two population heuristic algorithm Scatter search and bionomic algorithm. Scatter search relies on deterministic process instead of randomness. Opposite to the Genetic Algorithm, scatter search is not limited to binary variables. Bionomic algorithm is less used algorithm among other heuristics. It is based on a graph which represents parents and population structure. In their research two different objective functions are considered: a non-linear and a linear objective function. Linear objective function penalizes deviation of earliness and lateness from the target time. The nonlinear objective function penalizes a quadratic of positive and negative deviation from landing after and before target time. They used Operation Research (OR) data base of Beasley et al (2000) for a large number of 500 aircrafts.

Hu and Di Paolo (2008) used the neighboring relationship between every two aircrafts to build a chromosome as a binary zero-one matrix. The matrix can be adopted as a uniform crossover operator. Using simulation study they showed that the binary-representation-based GA outperforms the permutation representation-based GA algorithm.

Hu and Chen (2005) proposed a new genetic algorithm in a dynamic environment. The algorithm is solved using Receding Horizon Control (RHC) for the problem of arrival scheduling and sequencing (ASS).The goal is to minimize the total delay. The total delay is deviation of the actual landing time from the earliest landing time. To test their algorithm, problem instances of Bianco et al (1997) is used, and they compared their GA performance with the approach of Bianco et al (1997).

2.2.2 Local Search:

Local search is one of the common heuristic methods that find the best solution in a search space. Local search tries to make improvement by comparing the solution with other search spaces. The common approach is to start with an initial solution and replace it with a better solution by exploring other neighborhoods. The procedure is to start with an initial solution to find the local optimal solution. Then, it moves to another neighborhood until no better solution can be found. The drawback of the local search algorithm is that the solution is in the local optimal solution and it may be trapped to a search space while there exist a better solution. To avoid this problem some efficient approaches based on the problem’s structure were introduced.

Soomer and Franx (2007) implemented a local search heuristic for landing problem in a single runway. Their algorithm is designed to find an optimal sequence of flights and landing times. Furthermore, a MIP model for aircrafts sequencing is provided. Using neighborhood of the flights sequence, local search tries to improve it in a way that a new sequence does not change significantly from the previous solution.

Liu (2015) proposed a Genetic local Search (GLS) algorithm. GLS implements local search into genetic algorithm framework. In the GLS local search is used to improve the initial population that is randomly generated in the first step where aircrafts landing times is assigned.

2.2.3 Tabu Search:

Tabu Search (TS) is an extension of local search algorithm with the difference that it has capability of not being trapped in a local optimal solution. To find the best possible solution Tabu Search iteratively moves from one neighborhood to another. Using a memory based structure it creates a list of tabu moves that already investigated. TS has the ability of searching for a better solution in other areas that classic local search algorithm might not be able to explore. Similar to the local search algorithm, the exploring procedure can be terminated by a predefined function.

In the runway scheduling, TS algorithm is mostly used for takeoff and rerouting problem. For example, Sema et al (2014) used TS algorithm for aircraft rerouting problem for two runways in Milan Malpensa Airport. As an objective function minimization of maximum delay is considered.

2.2.4 Simulated Annealling:

Simulated Annealling is a stochastic neighborhood search algorithm where it starts by randomly generating an initial solution and compare the cost with its neighborhood. The process is terminated until the optimal or near optimal solution is reached. One of the advantages of simulated annulling is the capability of not being stuck in the local optimal points.

Salehipour et al (2009) used simulated annealing and Variable Neighborhood Search (VNS) method to tackle ALP problem and runway assignment. To acquire more efficient result the initial solution is reached by genetic algorithm.

2.2.5 Ant Colony:

Ant Colony algorithm is based on real ant colony behavior where the ants try to find the shortest path to their nest or source of food, and it’s been used wildly for combinatorial optimization problems. In the literature review, several approaches based on ant colony optimization were proposed for solving ALP problem. For instance, Zhan et al (2010) used ant colony optimization algorithm to solve ALP problem in a congested situation. As an objective function their algorithm tries to reach the optimal sequence with minimization of total aircrafts delay time.

2.3 Distributed Optimization

Distributed optimization is a decentralized approach where there is neither global control nor global data storage. In the decentralized approach tasks are usually distributed between different parts of the system such that the work load of the optimization and control is spread out during different parts of the system. Distributed optimization approaches can results to a higher speed and reliability of the problem solving process. Distributed optimization can be referred as Agent Based model when there are certain numbers of collaborating agents. The agents have their own goals or tasks to accomplish.

G.SMITH (1990) developed the Contract Net Protocol to determine a distributed problem solver which includes negotiation process. In his approach nodes with specific tasks are defined in a way that none of the nodes has sufficient data to solve the problem and negotiation between nodes is necessary to achieve the global solution. The negotiation process is designed that each node is capable of communicating with every other nodes. In his algorithm each node can take either controller role, manager or they can get both rolls at the same time for different contracts. Manager is in charge of processing the results of the execution where controller is in charge of the actual execution.

A contract is built by mutual communication between managers and controllers. Managers announce tasks to be evaluated by controllers. Controllers send a bid to those tasks that suits them. The manager evaluates the bids and awards contracts to the nodes that are most appropriate. Then the negotiation process may be iterated. The controllers play a roll of manager and assign tasks to the other nodes. The whole process can be divided into several steps including: Task announcement, task announcement processing, bidding, bidding process, contract processing, and termination.

In the task announcement a node announces a task including information such as: a brief description of the task, eligibility that nodes have to meet to be able to participate in the negotiation process, biding criteria and expiration date of the task which is a deadline for the nodes to bid the task. In the announcement processing part each node checks for the eligibility once that it receives the announcement. Then, it ranks the task according to its criteria. In the bidding once a node receives different tasks, it selects the one that is most attractive and submit the bid. In the bidding process, a node has a list of bids from different nodes after collecting all the bids. Then, the node awards the one that is most appropriate for the task. After winning the contract, the controller and the manager communicate with each other to give the sufficient information for the task to be done. The manager can terminate the task processing after or before the job is done.

In the agent based approach, one of the main issues is to define the structure of the system. Defining the structure of the system facilitates communication and task distribution between agents to get the most appropriate result. Shen (2006) provided a comprehensive agent based review on the works that have been done in the manufacturing process planning and scheduling. He compared agent based approach with the traditional approaches. Tumer (2007) developed an agent based system for air traffic flow management and tested the model in the FACET – an air traffic flow simulator developed at NASA. In their multi agent algorithm they considered "fixes" ground aircraft controller, as an agent. The agents are responsible for aircrafts that are in their area to keep safe distance between each other. Through a reward processing, each agent learns the best value for the safe distance between other aircrafts such that it benefits both individual agents and overall system.

Patrice Godin (2005) used MAS as a decentralized model for dynamic outpatient scheduling problem where some booked timeslots are canceled. The manager allocates time slots to the patients. They introduced three kinds of agents of outpatient agent, directory facilitator and diagnostic service. The outpatient agents represent patients assistant that keep the schedule updated. Directory facilitator agent collects information from outpatients and provides registration. Diagnostic service agent plays a role of the clinic secretary. The agent works through the negotiation protocol to achieve an overall solution for the time slot allocation problem.

Sousa (1999) introduced a Holonic Architecture for the dynamic scheduling of manufacturing systems. Using the contract net protocol, the author developed a negotiation protocol. The negotiation protocol is able to adopt dynamic changes and some conflicts between agents.

There are several papers used auction based theory to develop airlines networking and scheduling. For example, George L. Donohue et al (2003) studied an auction based model for allocation of system airspace resources. Dian Sheng et al (2015) developed auction based system to allocate time slots to the airlines where there is an uncertainty in demand. S.J. Rassenti et al (1982) introduced sealed-bid combinatorial auction for airlines which allows the airlines to compete over landing or takeoff slots.

3. Mathematical Model

In this chapter, ALP problem is studied. General definitions and constraints are described. In order to implement the model in dynamic optimization and decentralized algorithms in chapter 4 and 5, a MIP model based on Beasley et al (2000) model is introduced. The model considers a single runway for landing only. In the MIP model, we assume that there is a fixed set of aircrafts reaching the runway requesting a landing time to be allocated by the controller in the control tower. In addition, we assume that we already know all the information (such as aircraft preferred landing time, velocity, time window, etc.) that is needed to generate landing schedule for each aircraft. The set of aircraft is known and fixed when the aircraft reach the Extended Terminal Maneuvering Area (E-TMA) 30 to 40 minutes before their landing. The aircraft approaching the runway are under control of the Terminal Radar Approach Control (TARCON). Finally, the landing time is assigned to an aircraft when it reaches the final path 20 minutes before their target landing time.

3.1Definitions:

In this section common terms and definitions used in the ALP problem model are described.

3.1.1 Holding and Maneuvers

Holding an aircraft in the air may happen when an aircraft arrives earlier than the scheduled time or due to the traffic congestion, bad weather conditions, and emergency situations. Several aircrafts can be on hold at the same time. ATC rules require that the late aircrafts go on hold in a descending order of altitude. The holding patterns have several restrictions that each aircraft must follow depending on its type, speed, and runway situation. To hold an aircraft, there are four main techniques used by the controller including vector for space (VFS), holding pattern (HP), detour, and shortcut. Figure 3.1 illustrates the techniques commonly used to hold an aircraft.

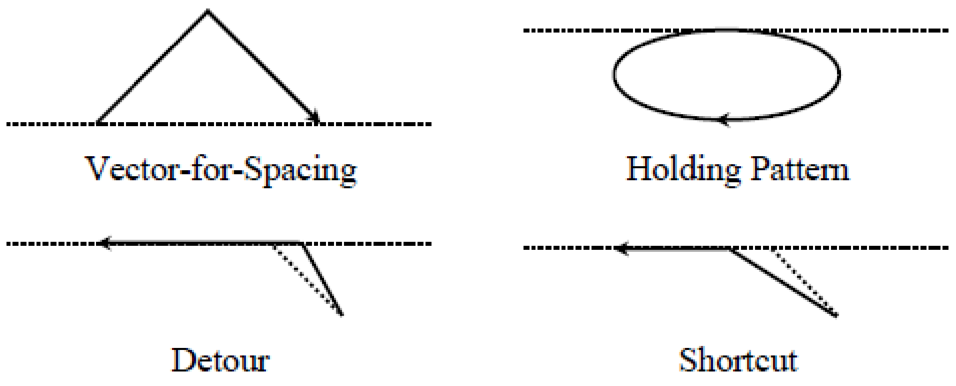


Figure 3.1 different maneuver techniques (Mesgarpour, 2006)

3.1.2 Minimum Time Separation:

The most important safety factor to schedule aircraft landing is minimum time separation which is established by the International Civil Aviation Organization (ICAO). Because of wake vortex effect between each pair of aircrafts, the trailing aircraft should maintain a minimum separation space to avoid danger of instability that is caused by the leading aircraft. Minimum time separation mainly depends on aircraft weight as shown in table 3.1.

According to the ICAO weight category, aircrafts are classified in three types of heavy, medium and light type. For example, aircrafts Boeing B777 and Airbus A330 are heavy aircrafts and aircraft Boeing B737 and Airbus A320 models are medium and Bombardier CRJ700 and Embraer Z145 are light aircrafts.

Let a, b and c represent the aircraft classes heavy, large and small respectively, and δ is the time separation between each two classes. Equation 1, triangle inequality, ensures that the time separation between sequencing aircrafts from different classes is met.

For all aircraft class a, b and c: $\delta_{ac} \leq \delta_{ab} + \delta_{bc}$(Equation 1)

Landing (second)	**Trailing Aircraft**		
Leading Aircraft	**Heavy**	**Large**	**Small**
Heavy(greater than 300 000 *lb*)	96	157	196
Medium(between 15 500 and 300 000 *lb*)	60	69	131
Light(less than 15 500 *lb)*	60	69	82

Table 3.1 Minimum separation time between landing aircrafts (ICAO)

3.1.3 Time Window Constraints:

In aircraft landing schedule, each aircraft has earliest landing time (ELT) and latest landing time (LLT). The scheduled landing time (LT) of an aircraft, should be between ELT and LLT. The earliest landing time is based on the maximum safe speed at which an aircraft can land. The latest landing time is based on the amount of fuel carried by the aircraft and the maximum time for which it can be on hold before landing.

3.1.4 Precedence Constraint

The precedence constraint is usually determined by airlines in order to apply particular operations, and manage urgent flights. The constraint restricts some flights from being overtaken by other flights on the same route.

3.2 Objective Function:

In the ALP problem, a variety of stakeholder interests are involved. In order to consider stakeholder interests different objective functions are considered in the related literature. The stakeholders are Air Traffic Control (ATC), Airports, Airlines, and Governments. Each of the stakeholders has their own specific objectives that may be in conflict with others. The most used ALP objective functions in the literature are:

-Minimizing makespan

-Maximizing runway utilization

-Minimizing total delay

-Minimizing deviation from planed landing time

-Minimizing environmental effect (noise and pollution)

In our model, we considered three objectives. The first objective is to minimize makespan which minimizes the last actual landing time of the aircraft. Minimizing the makespan is equivalent to maximizing the runway throughput. To consider airline's point of view, we set the second objective to minimize the total delay of all aircrafts. The last objective includes minimizing of the unfairness. Minimizing unfairness means to spread out the delay time caused by minimizing the make span through all aircrafts.

3.3 Mathematical Model:

In this section we introduced mathematical ALP model which is based on the Beasley (2000). Beasley modeled the scheduling landing problem using a mixed integer zero-one formulation. The objective function is to minimize the makespan (Equation 1). He also considered the objective function to minimize deviation from the aircraft's target time (Equation 2). The target time is the time when an aircraft lands in its most economical speed known as cruise speed.

LT_i represents the actual landings time of aircraft i where there is a fixed set of p aircrafts waiting in line.

Minimize maximum $[LT_i\ ,\ i = 1,..,p]$ (Equation 1)

To consider deviation from target time of all aircrafts (Equation 2) is implemented.

TW = $\sum_{j=1}^{n} \max\{ TG_i - LT_i\ , 0 \} + \sum_{j=1}^{n} \max\{LT_i - TG_i\ , 0\}$ (Equation 2)

The given TG_i represents the preferred landings time of aircraft i. Figure 3.2 shows the variation in cost of aircraft's time window from equation 2.

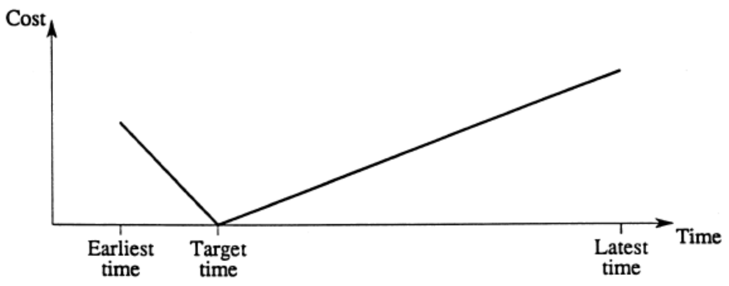


Figure 3.2 cost variation in aircraft time window (Beasley et al, 2000)

To include fairness in actual landing time assignment to all the aircraft the last objective (Equation 3) is considered.

$\sum_{i=1}^{p} \frac{LT_i}{p}$ (Equation 3)

3.3.1 Constraint Position Shifting (CPS):

Constraint Position Shifting (CPS) is used to prevent the model from placing an aircraft far away from its initial position in the FCFS order. The CPS constraint first was introduced by Dear (1989) and developed by Balakrishnan (2007).

CPS constraint maintains fairness among all aircraft by not allowing them too much deviation from their position in FCFS order. It implies that aircrafts cannot move more than a certain number from their first position in FCFS order. For example, if the number of allowed moves is 2, for an aircraft with 4th position in FCFS order, only position of 2th, 3th, 4th, 5th, and 6th can be taken. In our model in order to define such a constraint we used the binary decision variable x_{ij} indicating the order of actual landing time. To indicate initial position of each aircrafts in FCFS order we defined F_{ij}. We defined the constraint in a way that all aircrafts have the flexibility of moving one position backward or forward from their initial position in the FCFS order.

Decision variables:

$$x_{ij} = \begin{cases} 1 & \text{if aircraft } i \text{ lands before aircraft } j \\ 0 & \text{otherwise} \end{cases}$$

LT_{ib} $\quad$ *integer landing time of aircraft i type b*

Objective Function:

Minimize max LT_i + TW + $\sum_{i=1}^{n} \frac{LT_i}{n}$

Subject to:

1) $x_{ij} + x_{ji} = 1 \qquad \forall\, i,j \in N, i \neq j$

2) $LT_{ia} + \delta_{ab} \leq LT_{jb} + M(1 - x_{ij}) \qquad \forall\, i,j \in N,\ i \neq j \quad a,c \in C$

3) $LT_{ib} + \delta_{ba} \leq LT_{ja} + M(x_{ij}) \qquad \forall\, i,j \in N,\ i \neq j \quad a,b \in C$

4) $ELT_i \leq LTi \leq LLTi \qquad \forall i \in N$

5) $LT_i p_{ij} \leq LT_j \qquad \forall\, i,j \in N \quad i \neq j$

6) $x_{ij} = F_{ij} \qquad \forall\, i,j \in N \quad i \neq j + 1\ \&\ i \neq j - 1$

$x_{ij} \in \{1, 0\} \qquad \forall\, i,j \in N \quad i \neq j$

$LT_i \geq 0 \qquad \forall i \in N$

The objective function minimizes the last assigned landing time to the aircraft, the deviation from all aircraft's actual landing time, and unfairness among all the aircrafts. Constraint (1) ensures that either aircraft i lands before aircraft j or aircraft j lands before i. Constraint (2) and (3) consider minimum time separation between each pair of aircrafts. Constraint (4) represents aircraft's time window such that each aircraft should land between its earliest and latest possible landing time. Constraints (5) represents precedence constraint where p_{ij} is a binary variable with the value of 1 if aircraft i must land before aircraft j, and 0 otherwise. This constraint ensures that if aircraft i must lands before j then aircraft i is scheduled earlier than aircraft j. Sources of such constraints are the airlines themselves, which have precedence constraints due to priority flights. In addition, arrivals on the same jet route are constrained to not overtake each other. Precedence constraints can also represent the restricted freedom available to taxiing departures that are not allowed to overtake each other (Balakrishnan, 2010) (Carr, 2004). The last constraint ensures that aircrafts can only move one position from FCFS order.

In this model aircrafts occupancy time, the time from runway touchdown until leaving the runway and required maneuvering time are not considered. The model was solved and verified in OplCplex Optimization Studio 1263 software, and the results are discussed in the next chapter.

Chapter 4. Dynamic Scheduling

In this chapter, in order to study the impact of new aircraft arrival on the aircraft landing scheduling, dynamic ALP optimization problem using AnyLogic simulation software is investigated. A brief overview of the AnyLogic environment is described and the implementation of the model into the software is explained.

4.1 Challenges and Approach:

Runways are highly dynamic areas which directly affects the scheduling. These dynamic events that happen at the airport are one of the issues of implementing the ALP model. One of the most dynamic events that affects the aircraft landing scheduling is the new arrival of aircrafts to the airport. While the static models consider a fixed number of aircrafts, there is a new aircraft appearing to the radar system every few minutes. Due to the obligatory minimum time separation, the utilization of the runway mainly depends on the order of aircraft's landing. For instance, the makespan for five heavy types of aircrafts following by another five but light weight aircrafts is much less than the makespan of a heavy type followed by one light weight in the sequence.

After scheduling a fixed set of aircrafts and putting them in an optimal order, when a new aircraft comes to the system it may be more efficient to put the new aircraft in the last or second last in the optimal order. For example, if we already scheduled ten aircrafts, scheduling the new aircraft in the 9th or 10th position may results in a better utilization of the runway, instead of ignoring the new arriving aircraft. The traditional way to tackle this problem is to run the static model over and over waiting for the new set of aircrafts to arrive to the system. This approach is not very efficient because it does not consider the minimum time separation between the last aircraft in the optimal sequence and the first aircraft in the new set. Furthermore, the number of aircrafts in the set to solve the problem can be an issue itself. To include this dynamic event of aircraft new arrival, we implemented ALP problem using AnyLogic simulation software.

4.2 AnyLogic Implementation

AnyLogic is a simulation tool that covers three methodologies framework including: Agent Based models, Discrete Event Simulation models and System Dynamics. Each of these features is used for different level of abstraction and complexities. The software was written based on Java Script and gives the flexibility to structure environments with different abstraction level. In addition, there is a solver engine OptQuest that is used for linear programming optimization purposes.

4.3 Simulation

To study the impact of the dynamic events at the runway, we simulated the ALP model where aircrafts appear on the system making the controller reschedule the model with updated information. To have more flexibility on communication between aircrafts and the controller, we used the agent based framework where there is a main environment agent, aircraft agent, and the controller agent. The main agent is the environment of the runway including aircraft agent and controller agent. Aircraft agent is responsible for generating new aircraft arrival every few minutes and assigning attributes to each of them randomly. The aircraft's attributes include the type, earliest and latest landing time and aircraft's number.

4.3.1 Procedure:

Once an aircraft arrives on the system, it sends its information to the controller agent. The controller agent receives the aircraft's information and puts them in the collection list. The collection list is a module that keeps necessary information for the optimization part. Figure (4.1) shows aircrafts approaching on the runway from different zone. Zone C is the place for those aircrafts that have not appeared in the controller's radar yet. Aircrafts in zone B are those who send their information and participate in rescheduling every time a new aircraft arrives to this area. Zone A is for aircrafts who are too close to the runway to participate in rescheduling. These aircrafts are not allowed to get new landing times and are excluded from the collection list.

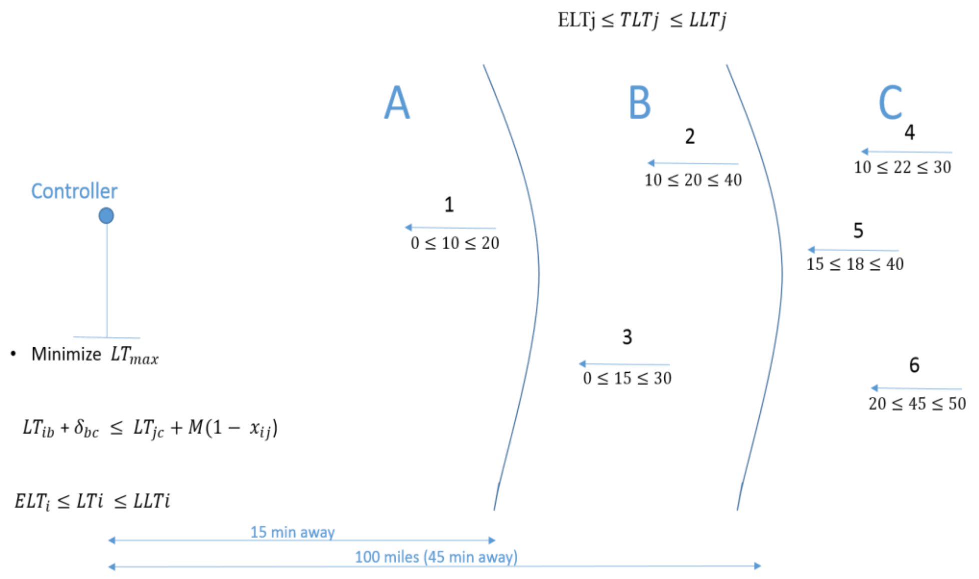


Figure 4.1 Approaching aircrafts to the runway

As it is shown, the controller runs the model with an objective function of minimizing the maximal landing time with the constraints, which was mentioned in the previous section. A new aircraft appearing on the radar in zone B makes the controller run the optimization model to assign temporary landing time to the aircraft. Those aircrafts that are too close to the runway will keep their last assigned landing time as their actual landing time and is removed from the system.

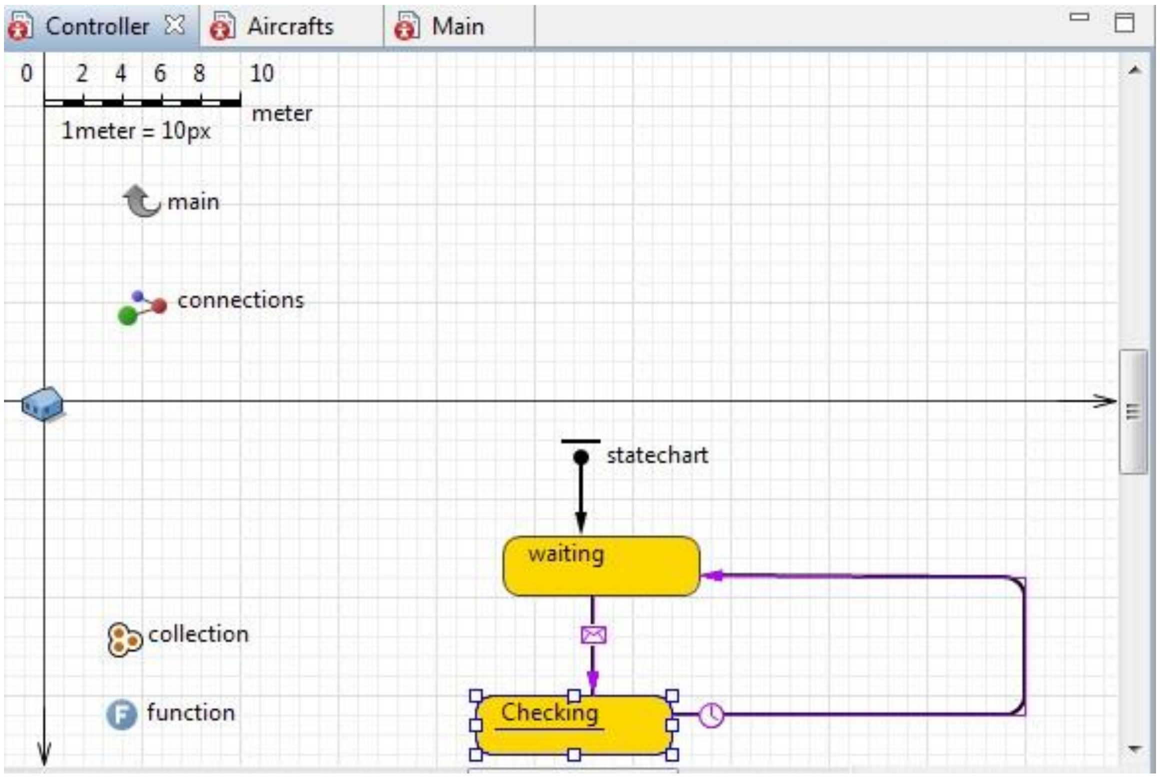


Figure 4.2 Controller chart

The figure (4.2) shows the environment of the controller agent. The communication between agents is adjusted through the Connection module. To show the behavior of the controller, the state chart is used. The normal state of the controller agent is to wait for a new aircraft arrival. Once a new aircraft arrives on the system, a massage is sent from the aircraft agent to the controller. Then the controller changes its status to the checking state. In the checking state, the controller looks for those aircrafts who are not eligible for the scheduling optimization, and removes them from the collection list. Having updated information, the controller agent sends a message to the Function module to run the optimization model. Function is the module where the OptQuest solver is called. Once the model is solved the controller announces the landing times to the aircrafts and returns to the waiting status looking for new aircrafts to join the system.

The language of the solver engine is based on Java Script and the optimization model is written by calling the OptQuest library. The results of the simulation are shown in the next section.

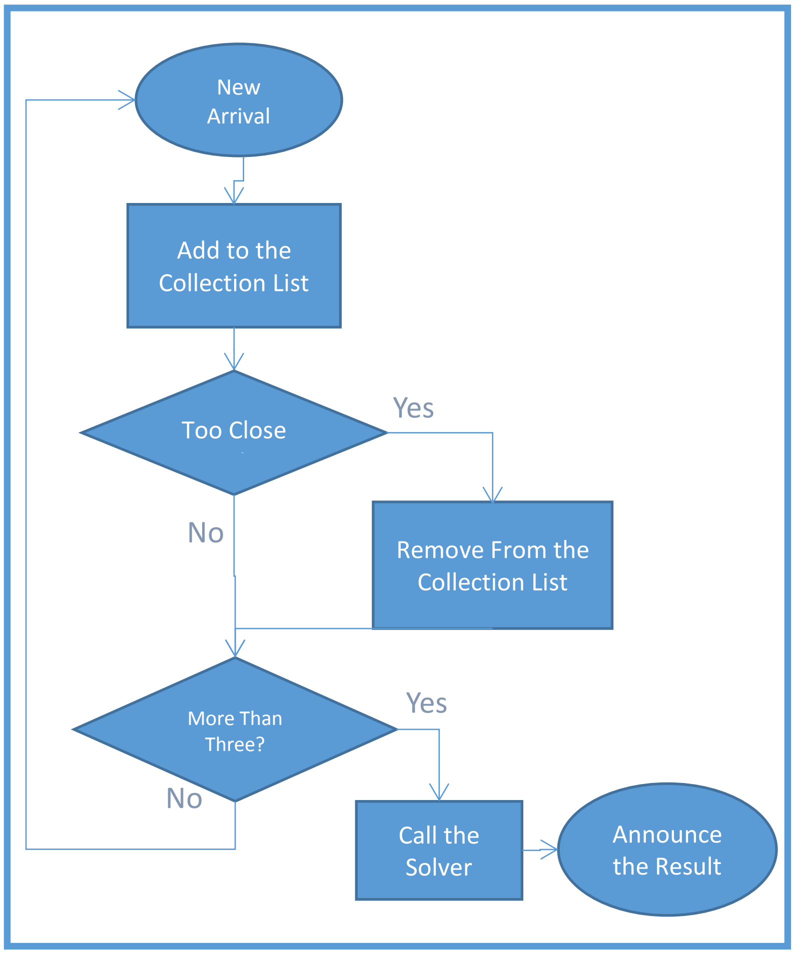


Figure 4.3 Dynamic Scheduling Algorithm Flowchart

4.3.2 Termination:

To maintain the minimum separation time between aircrafts, one cannot be removed from the set when time separation between them is applied. To avoid such a problem the participation in scheduling optimization model can be terminated when a new arrival aircraft's time window does not overlap with the other aircraft's time windows. The largest minimum separation time between two aircrafts is three minutes (Table 3.1). When the time window of the new aircraft does not overlap with other aircrafts, the minimum separation time between the new aircraft and other aircrafts is met. The landing time of the new aircrafts will be at least as big as the separation time between all other aircrafts and the new aircraft. Checking the aircrafts time window, the controller can terminate the optimization and assign the actual landing time to the aircrafts.

4.5 Results:

In this section the data and the results of the ALP model and simulation implementation is shown.

4.5.1 Data Description:

The aircrafts time window data is set according to the OR library ALP problem. Each aircraft time window's length varies from 6 to 8 minutes, and target time is on average 2 minutes after earliest arrival time. For the aircrafts type we studied Pier Eliot Montreal Airport (YUL) aircrafts arrivals within a week of April 2017. According to the data that we collected 59% of the aircrafts were medium weight, 31% light weight, and 10% of them were heavy weight. The average arrival rate of aircraft in a busy day was 0.25

aircrafts per minute or one aircraft every 4 minutes, and it varies to one aircraft every 7 minutes (www.skyscanner.ca).

4.5.2 Experimental Results

The static model is run with three different objective functions up to 50 aircrafts as larger instances cannot be solved in a reasonable time. We run the model for each objective separately as each objective function can be used in a certain situation. Objective number one considers minimizing the makespan which is applicable during rush hours at the airport which happens during high seasons in the weekends. Objective number two considers minimizing total delay, and Objective number three consider both objectives number one and two in addition to the fairness.

Table 2 shows the impact of using different objective functions to the aircrafts landing time. The model is solved for 20 aircrafts and the data, and the actual landing time for each objective is shown. The number in the first column shows the position in the FCFS order. The actual landing times is set to an increasing order. For instance, second row shows that aircraft with the third position in the FCFS lands at 51.04 as a second aircraft in the set.

No. Aircrafts	Obj 1 (min)	Time (sec)	Obj 2(min)	Time(sec)	Obj 3(min)	Time(sec)
10	81.34	0.12	4.29	0.12	151.02	0.25
20	116.19	0.14	9.39	0.12	208.651	0.20
30	273.97	0.18	9.83	0.24	401.89	0.23
40	379.31	0.26	11.68	0.32	584.06	0.33
50	455	0.32	15.25	0.33	697.83	0.39

Table 4.1 ALP with different Objective functions with up to 50 aircrafts

Charts 4.1 to 4.3 show the deviation from actual landing time and target time for 50 aircrafts .The black line represents target time and the points represent the actual

Position (FCFS)	Type	Earliest Landing Time(min)	Latest Landing Time(min)	Target Time(min)	Actual Landing Time Obj 1	Actual Landing Time Obj 2	Actual Landing Time Obj 3
1	Medium	38.81	45.78	39.83	38.81	39.83	39.83
3	Medium	44.45	51.04	45.77	51.04	45.77	45.77
2	Small	54.66	61.46	55.66	57.82	55.66	55.66
4	Small	58.82	65.02	61.06	58.82	61.06	59.77
5	Medium	60.60	66.85	62.39	61.44	63.68	62.39
6	Heavy	67.13	73.78	70.09	72.24	70.09	70.09
7	Medium	68.82	75.73	70.83	73.39	72.61	72.61
8	Heavy	69.39	75.57	71.49	75.57	71.46	71.46
9	Small	79.85	86.83	82.56	79.85	80.13	80.13
10	Medium	80.34	87.22	82.75	82.47	82.75	82.75
11	Medium	81.25	87.37	83.48	83.62	83.9	83.9
13	Medium	81.98	88.06	84.34	87.43	86.38	86.05
12	Small	83.62	90.05	85.38	90.05	85.38	85.05
14	Small	88.36	95.13	90.22	91.05	90.22	90.22
15	Medium	95.96	101.99	97.87	98.06	97.87	97.87
16	Heavy	100.24	107.17	102.42	100.24	101.02	101.02
17	Medium	100.78	107.70	102.17	105.95	102.17	102.17
18	Heavy	102.02	108.13	104.54	108.13	104.54	104.54
19	Small	110.18	116.95	111.75	110.18	111.75	111.75
20	Medium	116.19	122.77	118.30	116.19	118.3	116.19

Table 4.2 ALP position shifting with 20 aircrafts

Landing time.

Chart 4.1 shows the makespan and the actual landing time of 50 aircrafts. The model was solved with objective function 3 of minimizing the total tardiness, makespan and unfairness. As the chart indicates, 41 aircrafts land according to their target time where the total tardiness is 15.59 with the makespan of 457.32.

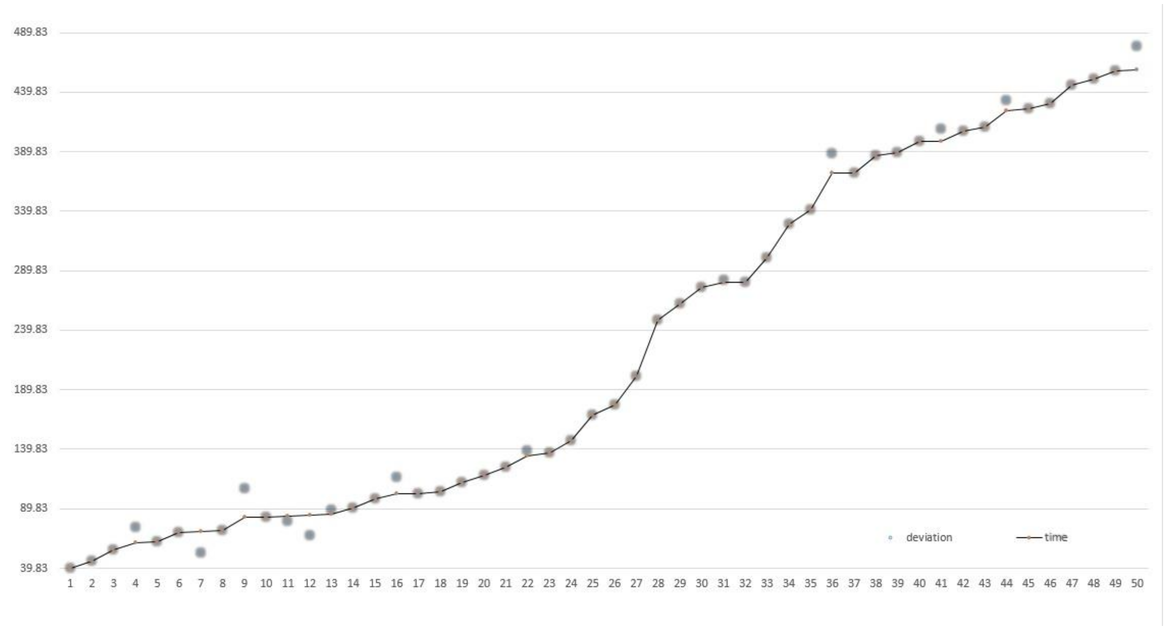


Chart 4.1 Deviation of aircrafts target time for objective 3

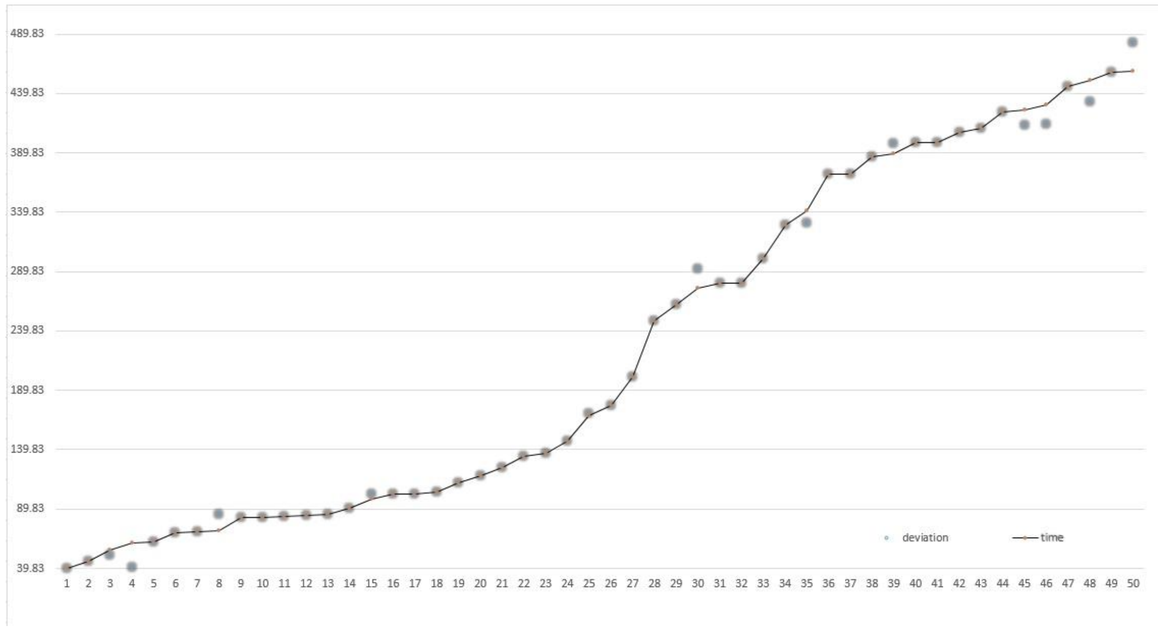


Chart 4.2 Deviation of aircrafts target time for objective 2

Comparing chart 4.1 and 4.2, obviously using objective function 2 of minimizing total tardiness results into less deviation in target landing time where the total tardiness stands at its minimum amount of 15.25 and the makespan of 459.94 has only 2 minutes and a few seconds difference compared with objective function 2.

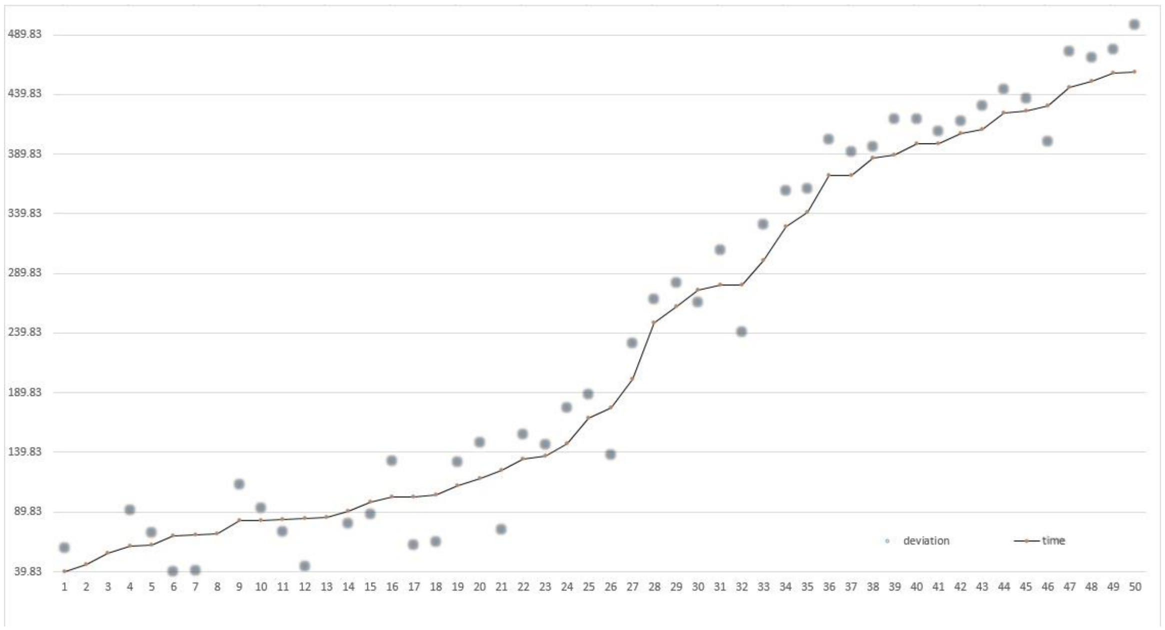


Chart 4.3 Deviation of aircrafts target time for objective 1

As chart 4.3 illustrates the propagation of aircrafts actual landing time is at its higher level when most of the landing time occurred toward earliest landing time. In this case the Air Traffic Controller (ATC) makes the best usage of the runway whiles the total tardiness stands at its highest rate 130 with the makespan of 455.44.

Table 4.3 shows the results observed from running simulation with three arrival rates. Based on our observation from Pier Eliot Montreal Airport (YUL) during the busiest time of the year in the weekend our minimum arrival rate is set to 4 minutes which changes to 10 minutes during regular time in the weekdays. Based on the dynamic optimization algorithm the maximum number of aircrafts participates in the scheduling optimization is

16 when the arrival rate is set to be 4 minutes where the last landing times occurs at 142.67.

Mean Inter Arrival Time (minute)	**Maximum Number of Aircrafts**	**Objective Maximize Utilization**
4	16	142.67
5	13	112.86
10	5	67.93

Table 4.3 simulation result with different inter arrival time

4.6 Conclusion:

In this chapter a static model has proposed. The model includes three different objective functions. Each objective function applies for different situations. As the results show the best utilization of the runway can be achieved by using the first objective function. The disadvantage of using the first objective function is large deviation from aircrafts target time. Similarly using the second objective function leads to the minimum total delay time of the aircrafts making the makespan much larger. And the last objective function results into a reasonable total delay time and the runway utilization. As a conclusion the first objective function maximizing runway utilization can be used during rush hours at the airport and the last one can be used during regular hours. In addition, applying Constraint Position Shifting makes the ALP problem more practical. As it is shown in the table 2 according to the FCFS order there is no more than one movement in the optimal sequence.

To implement the static model AnyLogic simulation software has been used. The dynamic feature of the aircraft landing problem for different aircrafts arrival rate has been studied. For the simulation part the first objective function, maximizing the runway utilization, has been used. Based on Table 4.3 the result is as the same as what we expected in the static ALP model.

Chapter 5. Distributed Optimization

In this chapter we introduce an agent based scheduling system to minimize the overall cost of aircraft landing problem in a decentralized fashion. To develop a decentralized scheduling algorithm, iteration bidding frame work is implemented and the result of the algorithm is compared with ALP centralized model introduced in chapter 3 and 4.

5.1 Distributed Optimization for ALP

During last two decades distributed optimization gained more attention for large problems. Distributed optimization is coordination of different agents among where each agent is responsible for its own decision making problem. In the distributed optimization the workload of the problem is distributed between the agents, and opposite to the centralized model there is no data storage at one place where local decision making in a decentralized approach contributes to a global solution in the system. In the following section the landing problem for decentralized aircraft landing is described, procedure steps and the mathematical model is explained, and an example is illustrated.

We consider the ALP problem in a wider time horizon where aircrafts are willing to schedule their landing time at the airport. This situation happens during bad weather situation when there is a cancelation so the airlines need to reschedule their landing time at the runway.

We focus on a setting where aircrafts may have different cost of landing within different time windows. The ATC needs to schedule the flights in a way that aircrafts minimum separation time constraints are satisfied, all the time windows are assigned in runway and at the same time, the overall costs are minimized.

We formulate the ALP model with the objective of reducing the overall flight costs. The model is basically the same as the ALP model introduced in chapter 3 and used in chapter 4 for dynamic scheduling.

In the decentralized approach, we designed aircraft landing problem in a way that the flights are modeled as self-interest agents and the cost of landing time window is their private information, which is not known to the ATC. The objective of the airline company is to maximize its profit which is the difference between the flight revenue and the cost for landing within the assigned time window. On the other hand, the objective of the ATC agency is to maximize the overall price from covering all flights while the cost of each flight is unknown.

We design a negotiation framework to solve the ALP. The framework is implemented by an iterative bidding procedure. It also provides flight scheduling process automation, which allows the ATC agency and the flights to construct efficient service schedules.

The rest of the chapter is organized as follows. In section 3, ALP model is described. Section 4 presents the structure and components of the proposed iterative bidding framework. Section 5 evaluates its performance through a computational study.

5.2 Flight Scheduling Problem

As it was mentioned before, we consider the ALP problem in a decentralized environment in which the costs of flights are unknown to the ATC. In order to compare the results with the decentralized approach. We reformulate the ALP model in a wider time horizon with the objective function of overall cost reduction. Since we consider ALP in a wider time horizon we refer it as a landing flight scheduling problem. In this section, we first introduce the binary model in a centralized environment to demonstrate the newly introduced features.

5.3 Centralized Formulation

For the centralized flight scheduling problem we formulate ALP with binary decision variable. The flight scheduling problem consists of a set of n aircrafts and an air traffic controller. The aircrafts land in a runway and the minimum time separation between aircraft class a and aircraft class b is denoted by δ_{ab}. An aircraft i has an earliest possible landing time $elt_i$and a latest possible landing time llt_i. The aircraft will not land before its earliest possible landing time or after the latest possible landing time. We assume that$r_i$is the flight revenue of aircraft i which lands before.C_{it}represents the cost of aircraft i landing at time t. Let Z_{it}be 1 if aircraft j lands at time $t \in T$and 0 otherwise. Consequently the actual landing time of aircraft i is $Z_{it}.LT_t$. The ALP involves the scheduling of exact landing times such that all the scheduling constraints are satisfied and, at the same time, the over profit is maximized. Let x_{ij} be a binary decision variable with the value of 1 if aircraft i lands before aircraft j and 0 otherwise. With this assumption, we can conveniently model the problem as a mixed integer program.

$$min \sum_i^n \sum_t^T (Z_{it} * C_{it})$$

Subject to:

$$x_{ij} + x_{ji} = 1 \qquad \forall\, i,j \in A\ i \neq j \quad (1)$$

$$elt_i . \sum_t^m Z_{it} \leq \sum_t^m Z_{it} . LT_t \leq llt_i \qquad \forall\, t \in T, \quad \forall\, i \in A \ (2)$$

$$\sum_t^m Z_{ita} . LT_t + \delta_{ab} \leq \sum_e^m Z_{jeb} . LT_e + M\left(1 - x_{ij}\right) \quad \forall\, t \in T,\ \forall\, i,j \in A\ i \neq j \quad (3)$$

$$\sum_e^m Z_{jeb} . LT_e + \delta_{ba} \leq \sum_t^m Z_{ita} . LT_t + M\left(x_{ij}\right) \qquad \forall\, t \in T,\ \forall\, i,j \in A\ i \neq j \quad (4)$$

$$x_{ij}, Z_{ita} \in \{1,0\} \qquad \forall\, i,j \in A\ i \neq j, \forall\, t \in T \quad (5)$$

The objective function minimizes the total cost of landing aircrafts. Constraint (1) represents either aircraft i lands before aircraft j or aircraft j lands before i.Constraint (2) represents aircrafts time window such that each aircraft should land between its earliest and latest possible landing time. Constraint number (3) and (4) consider minimum time separation between aircraft i from type a and aircraft j from type b, where M is a positive large number which is used for constraint formulation logic. Constraints (5) and (6) are binary and non-negative integer constraints respectively.

5.4 Revenue and Cost Structure:

A flight revenue mostly comes from passengers ticket (around 75 percent) and 15 percent comes from cargo and 10 percent comes from other transports services (Airline-

Economic.asp). Flight agencies play an important role for selling the ticket to the passengers. The revenue of a flight can be calculated by the amount of cargoes and the price of a sold ticket to the passengers.

There are different factors associated with a flight cost including: fuel cost, flying operation cost, maintenance cost, crew costs, passenger services and travel agencies. Labor costs are common to nearly all of those categories. Some flight costs are fixed while others depends on flight duration time. For instance, if a flight missed its target time missing connection cost between connected flights might happen where the airline are responsible to provide the passengers with proper services until they reach to the next available flight. Similarly, fuel cost increases when they are hold above the runway waiting in the line to land. The latter case mostly happens during rush hours. For our model we only consider those costs that increase by the flight landing time as time passes: Fuel cost, flying operation, Crew cost, Missing connection cost.

5.5 Iterative Bidding Framework

The proposed iterative bidding framework is a price mechanism based on Chun Wang et al (2011) in which the airport and aircrafts negotiate the landing time by adjusting the price on time windows. In this section, we first introduce the requirement-based bidding language used by the aircrafts to express the landing requirements in the bid. Then we describe the iterative bidding procedure which consists of four components, namely initialization, price update and bidding, bid screen and termination and winner determination. The iterative bidding framework allows the airport and aircrafts interact in a systematic way to generate the landing schedule collaboratively. It also allows negotiation over price and landing time concurrently.

5.6 Requirement-Based Bidding Languages

In this problem, the cost of aircrafts is based on the landing time window. During the negotiation with airport, an aircraft expresses its preference in a conditional statement consists of two elements: landing time window and price. In this subsection, we propose a requirement-based language to represent the preferences.

The airline company could be indifferent to landing within a certain time window. We define the atomic bid (C-Bid) to represent an aircraft's value over a time window defined by elt and llt. A C-Bid is a 4-tuple $< C, elt, llt, p >$ in which C is the category of the aircraft, elt is the start time of a landing time window, llt is the end time of a landing time window, and p is the price that the aircraft is willing to pay for landing within this time window $elt < alt \leq llt$, where alt stands for the actual landing time of this aircraft. C-Bids can be connected by XOR connective as an XOR-C-Bid to represent the values that an aircraft has on different time windows. For example, $< C_i,\ elt_{i,1}, llt_{i,1}, p_1 > XOR < C_i, elt_{i,2}, llt_{i,2}, p_2 >$ means that aircraft i is willing to pay p_1 if its landing time is allocated to $elt_{i,1} < alt_i \leq llt_{i,1}$, and p_2 if it's landing time is allocated to $elt_{i,2} < alt_i \leq llt_{i,2}$. The aircrafts only needs one landing time, and there is no overlap between the two time windows.

Assume that an aircraft has m_k time windows within the acceptable time window. Accordingly, an XOR-C-Bid with m_k C-Bids can represent the aircraft's valuations within the acceptable time window. The full valuation of an aircraft can be represented by$< C_i, elt_{i,0}, llt_{i,0}, p_0 > XOR < C_i, elt_{i,1}, llt_{i,1}, p_1 > XOR < C, elt_{i,2}, llt_{i,2}, p_2 >$ $XOR, \ldots, XOR < C_i, elt_{i,m_k}, llt_{i,m_k}, p_{m_k} >$, where$elt_{i,0} = EL_i$,$llt_{i,0} = LL_i$, and $p_0 = r_i$. The time windows are adjacent, i.e., $llt_{i,k-1} = elt_{i,k}$for$1 \leq k \leq m_k$.

5.7 Iterative Bidding

In this section the iterative bidding process is explained. It includes four steps of Initialization, Bidding, Termination Checking and Winner determination.

5.7.1 Initialization

Before submitting the first bid, the aircrafts need to initialize a reserve price for their landing requirements, which is between its preferred landing time and any other delay time. The reserve price is set to 0 in order to maximize the profit. With the given fixed revenue, an aircraft's profit for a time window is the remainder of deducting bidding price, fixed cost and variable from the revenue at each round of bidding. Then the aircrafts bid for the time window with the highest profit. The bidding price of the first round equals the reserve price of that time window.

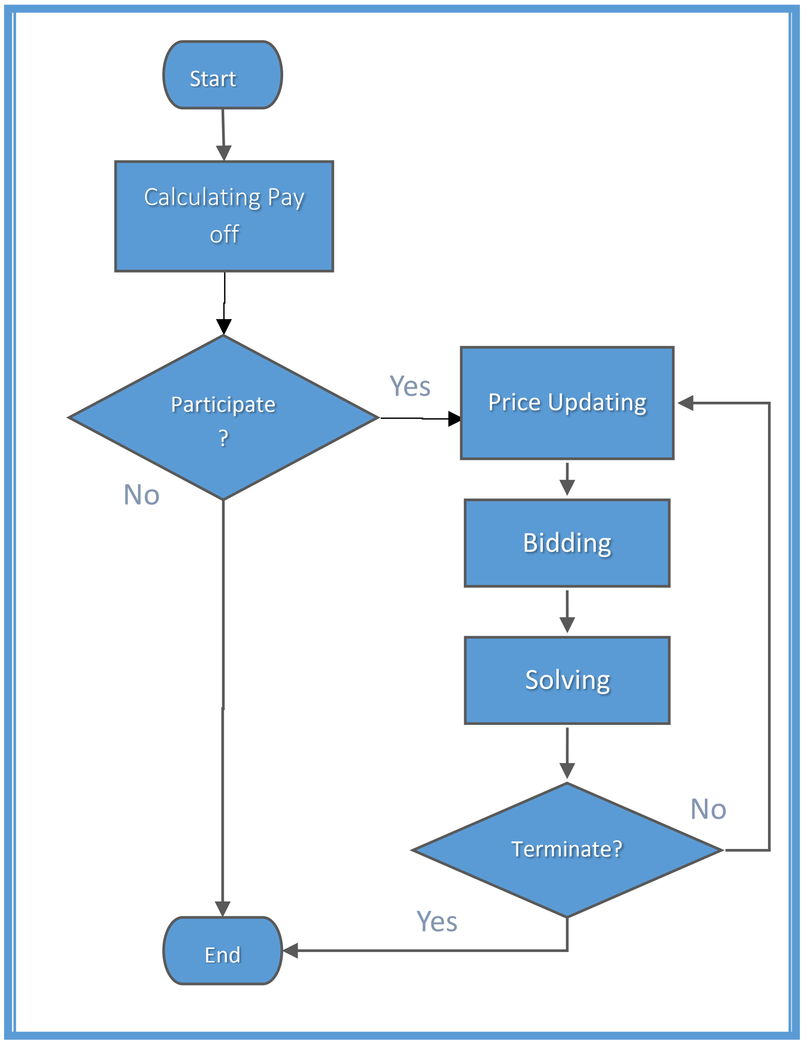


Figure 5.1 Iteration Bidding Protocol

5.7.2 Price Update and Bidding

At each round$t$ ($t > 1$), aircrafts start from updating their bidding prices for the time window submitted at round $t - 1$. There are three different scenarios for aircrafts to act out at round t depending on the provisional allocation status determined at round $t - 1$: (1) if an aircraft's bid was not awarded in the provisional allocation at round $t - 1$, it can increase its bidding prices by ε on the time window it bids for at round $t-1$ or rounds before $t-1$, where ε is the minimum price increment imposed by the airport. Since aircrafts are assumed to be rational in maximizing their profits, they in general do not bid with an increment more than ε. (2) If an aircrafts is provisionally assigned a time window at round $t-1$, they may want to keep their bidding price unchanged at next round, which means they are allowed to repeat the same bids at round t. However, the aircrafts are not prevented from entering a higher bid in future rounds in this scenario, and (3) we consider an ultimate time window for those aircrafts who are not assigned a schedule at the final round. The cost for ultimate time window is set higher than other time windows. The model is run one more round to assign these aircrafts a landing time within that ultimate time window. The latter case may happen when the fuel will run out at the end time of the time window and the profits of every other time windows become negative.

After updating bidding prices, an aircraft needs to compute the profit of time widows again based on the updated bidding prices to determine the maximum profit time window. In computing such a time window or time windows, an aircraft i solves a maximization problem $max_{i\in\{1\leq k\leq m_k\}}[r_i^t(llt_{i,k})-p_i^t(llt_{i,k})]$ and obtains a set of C-Bids with equally maximum profit, where $p_i^t(llt_{i,k})$represents the bidding price for $llt_{i,k}$ at round t.Then the aircraft randomly choose one from the set of C-Bids with maximum payoff and bid for updated bidding price. In the scenario that an aircraft has entered into final bid status, it is no longer allowed to increase its bidding price. However, the aircraft can repeat its final bid in future rounds until termination. We setup this final bid repeating arrangement here is to allow the temporarily excluded bids to come back to the game to further increase the airline company's profit. In the iterative bidding process, some bids can be temporarily “excluded” from the provisional allocation because, in a specific round, there is a particular combination of allocation constraints and resource requirements with higher overall profit. Along with the bidding continues, that particular situation may have changed to allow the previously excluded bids back to the bidding process. However, those bids will not be submitted again without this setting if their costs have been reached during the “excluded” periods, which means the aircraft will not choose to bid them, even though the time windows become available in subsequent provisional allocations.

5.7.3 Bids Screening and Termination Checking

In this stage, the airport first screens out the invalid bids from all bids received from aircrafts. Those bids will not enter into winner determination procedure. There are two types of invalid bids: (1) any bids with bidding price lower the highest one for that same C-Bid received in previous rounds, (2) bids with increased prices from aircrafts who have already declared their final bidding status previously.

The aircraft then checks the termination condition against the valid bids. The bidding will terminate at the round with no price updates for all valid bids, which means all aircrafts participating bidding in this round have repeated their bids. After the bidding terminates, the airport allocates landing time to the aircrafts according to the final allocation at their bidding prices. If the termination condition is not satisfied and the procedure continues, the winner determination model will take the set of valid bids as input and solve the problem. The auction goes back to price update and bidding after the winner determination.

5.7.4 Winner Determination

In the following winner determination model, we take XOR-C-Bids from aircrafts as input.We modeled the ALP to winner determination linear programming where the output would be a timeslot assigned to an aircraft with a certain price. In order to consider different time slots for winner determination model we defined a binary variable Z_{it} = 1 if aircraft i assigned to a time slot before its landing time t, and $Z_{it} = 0$ otherwise. The goal of objective function is to maximize the total price of the agents in the timeslots they bid for.

$$max \sum_i^n \sum_t^T (Z_{it} * P_{it})$$

Subject to:

$$\sum_t^T Z_{it} \leq 1 \qquad \forall\, t \in T,\ \forall\, i \in A \qquad (1)$$

$$x_{ij} + x_{ji} \ \leq 1 \qquad \forall\, i,j \in A\ i \neq j \qquad (2)$$

$$elt_i. \sum_t^m Z_{it} \leq \sum_t^m Z_{it} \,.\, LT_t \leq llt_i \qquad \forall\, t \in T, \quad \forall\, i \in A \qquad (3)$$

$$\sum_t^m Z_{ita} \,.\, LT_t + \delta_{ab} \leq \sum_e^m Z_{jeb} \,.\, LT_e + M(1 - x_{ij}) \qquad \forall\, t \in T\ \ \forall\, i,j \in A\ i \neq j\ (4)$$

$$\sum_e^m Z_{jeb} \,.\, LT_e + \delta_{ba} \leq \sum_t^m Z_{ita} \,.\, LT_t + M(x_{ij}) \qquad \forall\, t \in T, \forall\, i,j \in A\ i \neq j \qquad (5)$$

$$x_{ij} , Z_{ita} \ \in \{1,0\} \ , \forall\, i,j \in A \quad i \neq j\ , \qquad \forall\, t \in T \qquad (6)$$

The objective function maximizes the total price on the assigned time slot. Constraint 1 ensures that an aircraft can only be assigned to one of its time windows. Constraint 2-6 is basically as the same as constraint in the centralized model.

5.8 Example:

In this section, a worked example of assigning landing time to five flights is explained. For the sake of simplicity, we considered maximum number of three time windows for each flight, and the number of the aircrafts was set to five. However, the main model does not have this restriction. The revenue for each flight is fixed during scheduling. The cost of each time window varies from one to another, in an increasing order. The data of the worked example is illustrated in table 5.1. For example for aircraft number one we considered two time windows of (1, 3) and (3, 12) with cost of $33 and $47 respectively. The third column of the table indicates the revenue of each flight.

Aircraft No.	**C Bid**	**Revenue**
1	(1,(1,3),33), (2,(3,12),47)	$100
2	(1,(3,4),45), (2,(5,6),50)	$130
3	(1,(4,8),30), (2,(11,13),35),(3,(18,30),37)	$130
4	(1,(6,10),40), (2,(11,12),44)	$150
5	(1,(6,10),40), (2,(11,12),50),(3,(13,14),70)	$120

Table 5.1 Iterative Bidding Data Set Example

All the flights are willing to be scheduled in a time window with the minimum cost. Assume that the price increment ($\varepsilon = \$5$) set by the ATC agent. At the beginning, when the scheduling process starts, all flights calculate their payoff and bid for the one with the minimum cost. In the first iteration, flights 1 to 5 bid for the first time window with the lowest cost with the starting price of 0. The ATC agent receives the bids and calculates the winner determination model. After several iterations and price increments, at the end of the last round, exact landing time of the aircrafts are assigned with the related cost. The final result of the iteration bidding model for the example is shown in table 5.2. The second column of the table shows aircraft type. The third column shows the number of assigned time window. The forth column indicates the exact landing time of the aircraft regarding to the minimum time separation. The fifth and the last column show the cost and the price related to the assigned time window to the aircraft.

Aircraft No.	**Type**	**Time window**	**Landing time**	**Cost**	**Price**
1	Medium	2	12	$47	$97
2	Light	2	5	$50	$130
3	Light	1	4	$30	$130
4	Medium	2	9	$20	$130
5	Medium	1	8	$40	$120

Table 5.2 Iterative Bidding Example Result

In order to compare the results of the iterative bidding example with the ALP centralized model, we solved the centralized model with the same data set. As Table 5.2 shows, the overall cost for iterative bidding model is $ 187 which is close to the optimal solution achieved by centralized model of $158.

5.9 Data and Experimental Result

To represent practical output of the experiment, real scaled data for aircrafts time window is used. The type of aircraft that is used in the model is as the same as data used in chapter

3 and 4. For the aircrafts type we studied Pier Eliot Montreal Airport (YUL). Based on OR library data base, the length of aircraft time window is considered from 6 to maximum 9 minutes. For the winner determination model, the revenue of each flight is estimated based on the number of the tickets sold of a flight and cargo expenses, the latter of which is considered as a fixed amount for each flight. We considered two types of cost: fixed cost, and variable cost. Fixed costs include total crew cost, maintenance cost and fuel cost. And missing connection cost is considered as a variable cost which changes with different time window within the time horizon. Since the fuel and other associated costs do not change significantly they are categorized as a fixed cost, when compared with missing connection costs due to the delayed flights. The models is coded in ILOG OPLStudio1263 (Optimization Programming Languages, IBM) and solved using ILOG CPLEX.

After running the model all aircrafts are assigned with a proper landing time which maintains the safety factors. The model is tested with ten different group of instances up to 20 aircrafts. For each group ten different instances are randomly generated. The experimental result is shown in (Table 5.3). The effectiveness of the decentralized model is confirmed by comparing it with the centralized ALP model with ten different instances. Table 5.3 shows the optimal cost for iterative model framework and centralized ALP model. The first column of the table shows groups of instances. For each group the average of ten different instances are considered. Second column shows the number of aircrafts used to solve the model.

We tried minimum number of five aircrafts and solved the model up to 20 aircrafts. For

Group	Number of Aircraft	Bidding solution cost	Bidding solution Payment	Optimal ALP Solution
1	5	$175	$425	$135
2	5	$140	$245	$105
3	10	$295	$490	$210
4	10	$345	$770	$300
5	15	$495	$1100	$415
6	15	$535	$1250	$405
7	20	$570	$1060	$475
8	20	$645	$1320	$510
9	20	$560	$1150	$420
10	20	$630	$1455	$545

Table 5.3 experimental result of iteration bidding model

the 20 aircrafts the model is solved less than two seconds. As we increase the number of aircraft to 25 the model is not solved in a reasonable time as it may takes more than one minute to reach the solution.

Third column indicates the bidding solution cost which is the overall cost of assigning the aircrafts with proper landing time within selected time window. Forth column shows bidding solution payment which is the overall cost and total price of the time windows that the aircraft bid for. The last column shows the optimal solution of the centralized ALP with the same data set as its decentralized model. Comparing the iterative bidding model with centralized ALP solution, we observed iterative bidding solution achieves on average of 78% of the optimal solution obtained by ALP solution. The epsilon value is set to 10$ and the initial price for each iteration is set to 0.

Comparing centralized model with our agent based model, the centralized model reaches the optimal solution, and the agent based model reaches a solution close to optimal solution. Our model’s advantages is that the agent based model considers the decentralized environment of the ALP problem with the price of not being optimal solution. Next chapter discuss the conclusion and related research future work.

6. Conclusion and Future Work

In this thesis the issue of Aircraft Landing Problem was investigated. In chapter three, a static model was introduced and verified. The CPS constraint was added to the static model. In chapter four, to study dynamic impact of the new arrival aircraft to the ATC system, the ALP model was implemented and tested in Any Logic simulation software. In chapter five, a distributed mathematical model using iterative bidding framework for ALP in a wider horizon was introduced.

In the implementation of the ALP model the impact of the new arriving aircrafts with different arrival rates has been tested. The inter arrival rates during busy time and regular time have a great impact on the model size and controller work load. Considering dynamic events of arriving new aircraft and landing aircrafts, decreases the computational workload as it excluded the aircrafts that has reached the final approach at the runway and ready to land.

In chapter five, we proposed an iterative bidding framework for flights landing scheduling using ALP with a binary mathematical model. The iterative bidding framework facilitates the negotiation between flights and the ATC. The negotiation model assigns an exact landing time to each flight while it minimizes the total cost at the same time. As a result of comparing centralized and decentralized models, the decentralized model has reached a proper solution close to the optimal solution of the centralized model. Furthermore, developing ALP as an agent based model, considers negotiation between aircrafts and the ATC. In the iteration bidding framework, each aircraft is self-interest agent making decisions according to their own benefit. Designing agent based ALP model, consider communication between agents in a decentralized environment and achieve an efficient solution to the problem. It also add time complexity to the problem where we solved the decentralized model up to 20 aircrafts in a real time. Larger number of aircrafts couldn't be solved in a reasonable time by ILOG CPLEX. As a future work a heuristic can be developed to solve the decentralized model with larger number of aircrafts in a real time.

Another future research direction to the ALP agent based model is to improve the efficiency of the solution. Current decentralized ALP model does not allow aircrafts to bid for their previous timeslots they bid in the previous iterations. The efficiency of the algorithm can be enhanced by allowing the aircrafts to be able to bid for all the time windows at the same time.

Appendix I

Main Iterative Bidding. Mode: Cplex

```
stringrunID="ALP";

main
{
	thisOplModel.settings.mainEndEnabled=true;

	varreturnedTotalSolutionValue;
	varreturnedProviderRevenue;

	functioniterativeRun(inputFile,epsilon)
	{
		writeln("Started iterative run");
		varsource=newIloOplModelSource("iterativeBidding.mod");
		vardef=newIloOplModelDefinition(source);
		varCplex=newIloCplex();
		vardata=newIloOplDataSource(inputFile);

		varepsilonData=newIloOplDataElements();
		epsilonData.epsilon=epsilon;

		varbaseOpl=newIloOplModel(def,Cplex);
		baseOpl.addDataSource(data);
		baseOpl.addDataSource(epsilonData);
		baseOpl.generate();
		varopl;
		vartotalNumberOfBids=0;
		for(vari=1;i<=20000;i++)
		{
			if(i==1)opl=baseOpl;
			else{
				opl=newIloOplModel(def,Cplex);
				opl.addDataSource(baseOpl.dataElements);
				opl.generate();
			}
			Cplex.solve();
			opl.postProcess();

			if(i%1==0) {
				printNextInput(opl,inputFile+"- iteration
"+i+".dat");
			}


		for(varoinopl.one) {
			for(vartinopl.T){
			writeln("------ZZ: "+opl.ZZ[o][t]);
			}
		}
```

```
                    totalNumberOfBids+=opl.numberOfBids;

                    // termination conditions

                    varallAircraftAssigned=true;

                    for(varfinopl.one) {
                                 //writeln("every services got assigned?" +
c3.isAssigned);
                                 if(f.isAssigned==0)
                                  {
                                        allAircraftAssigned=false;
                                        break;
                                  }
                    }
                    if(allAircraftAssigned)
                          break;

                    //check if no customer can bid in the next iteration (only
check bidding customer)
                    varanyAircraftHasBid=false;
                    for(varfinopl.one) {
                          if(f.flightTB.tw!=null!=null&&
                          f.isAssigned==0&&
                          f.isInFinalState==0)
                          {
                                 anyAircraftHasBid=true;
                                 break;
                          }
                    }
                    if(!anyAircraftHasBid)
                          break;

                    if(i>1)
                          opl.end();
             }


      baseOpl.end();data.end();epsilonData.end();def.end();Cplex.end();source.
end();
      }

      functionprintNextInput(opl,outputFileName)
      {
             varnextInput=newIloOplOutputFile(outputFileName);

             nextInput.writeln("one=");
             nextInput.write(opl.one);
             nextInput.writeln(";\n");

             nextInput.writeln("s=");
             nextInput.write(opl.s);
             nextInput.writeln(";\n");
```

```
                nextInput.writeln("P=");
                nextInput.write(opl.P);
                nextInput.writeln(";\n");

                nextInput.writeln("feasiblePackages=");
                nextInput.write(opl.feasiblePackages);
                nextInput.writeln(";\n");

                nextInput.writeln("ZZ=");
                nextInput.write(opl.Z);
                nextInput.writeln(";\n");

                nextInput.close();
        }


        iterativeRun("biddingData.dat",30);


}
```

Appendix II

Iterative Bidding. Mode: Cplex

```
intepsilon=...;

intn=5;
rangep=1..n;// number of flights

rangeA=1..3;//class type



tupletimeWindow
{
        floatelt;
        floatllt;
}

tupletimeBundle
{
        timeWindowtw;
}

tupleQ{
        keyintNo;
        intClass;
        intflightTBID;
        timeBundleflightTB;
        intisAssigned;
        intisInFinalState;
}

tupleFeasiblePackage
{
        keyintfpid;
        timeBundlepackageTB;
        floatbc;
        floatrevenue;
        intisBid;

}

floats[A][A]=...;//minimum sepration time
intP[p][p]=...;//precedence 0/1 number
{Q}one= ...;
{FeasiblePackage}feasiblePackages[one]=...;


dvarbooleanX[one][one];
rangeT=1..15;
intTS[T]=...;

dvarbooleanZ[one][T];
intZZ[one][T]=...;

intnumberOfBids=0;

executeinit
{

        //generate bids
        for(vari=1;i<=one.size;i++) {

                if(one.get(i).isAssigned==0)
                {
                        //check if it's the first round
                        if(one.get(i).flightTB.tw.llt!=0)
                        {

        varoldFP=feasiblePackages[one.get(i)].get(one.get(i).flightTBID);
                //            writeln("oldFP is "+oldFP.visitBundle + " ,price is:
" + oldFP.price);

                              oldFP.bc+=epsilon;

                        writeln("after subtracting,price is "+oldFP.bc);
                        }

                        varmaxUtility= -1;
                        varmaxUtilityFP;

                        for(varfp2infeasiblePackages[one.get(i)])
                        {
                              writeln(fp2.packageTB+" : revenue is 
"+fp2.revenue+", price+cost is "+fp2.bc);
                              varutility=fp2.revenue-fp2.bc;
                              writeln("                max utility is 
"+maxUtility+", utility is "+utility);

                              if(utility>maxUtility)
                              {
                                     maxUtilityFP=fp2;
                                     maxUtility=utility;
                              }
                        }

                        if(maxUtility!= -1) {
                              numberOfBids++;
                              maxUtilityFP.isBid=1;

                              one.get(i).flightTBID=maxUtilityFP.fpid;

        one.get(i).flightTB.tw.elt=maxUtilityFP.packageTB.tw.elt;

        one.get(i).flightTB.tw.llt=maxUtilityFP.packageTB.tw.llt;
                              writeln(one.get(i).No+" bid 
"+one.get(i).flightTBID+", with "+one.get(i).flightTB+" with price 
"+maxUtilityFP.bc+" and utility = "+maxUtility);
                        }else{
                              one.get(i).isInFinalState=1;
                              //With ultimate timeWindow setting
                              for(varinT) {
                                     if(ZZ[one.get(i)][t]==0){
                                            one.get(i).flightTBID=1;

        one.get(i).flightTB.tw.elt=feasiblePackages[one.get(i)].get(1).packageTB
.tw.elt;

        one.get(i).flightTB.tw.llt=feasiblePackages[one.get(i)].get(1).packageTB
.tw.llt;
                                     }else{
                                            ///Without ultimate timeWindow setting
                                            oldFP.bc-=epsilon;
                                            one.get(i).flightTBID=oldFP.fpid;

        one.get(i).flightTB.tw.elt=oldFP.packageTB.tw.elt;

        one.get(i).flightTB.tw.llt=oldFP.packageTB.tw.llt;

                                            writeln(one.get(i).No+" is in final 
bid: "+one.get(i).flightTBID+" with "+oldFP.bc);
                                     }
                              }
                        }
                }
        }
}
```

```
//Winner Determination Model Start

//dvar float+ LT[one];


//tuple decision{
//     Q q;
//     FeasiblePackage fp;
//}
//{decision} d = {<i,j> | i in one, j in feasiblePackages[i]};

//dvar boolean Z[z in d];


maximizesum(zinone,tinT)(Z[z][t]*(item(feasiblePackages[z],
<z.flightTBID>).bc));

subjectto{

       cons00://////XOR

forall(finone)// forall(f in one)
sum(tinT)Z[f][t] <=1;// sum(z in d : z.q == f)Z[z] <= 1;

cons01://////sequence
forall(iinone,jinone:i!=j)
       X[i][j] +X[j][i] <=1;

       cons021:///////time window
       forall(finone)//forall(f in one)
       sum(tinT) (Z[f][t]*TS[t]) <=f.flightTB.tw.llt;      //LT[f] <=
f.flightTB.tw.llt * sum(z in d : z.q == f)Z[z];

       cons022:///////time window
       forall(finone)//forall(f in one)
             sum(tinT)Z[f][t]*TS[t]
>=f.flightTB.tw.elt*sum(tinT)Z[f][t];//LT[f] >= f.flightTB.tw.elt * sum(z in d
: z.q == f)Z[z];

cons03://///minimum separation

forall(iinone,jinone:i!=j)// forall(i in one , j in one : i!=j)
       sum(tinT) (Z[i][t]*(TS[t] +s[i.Class][j.Class]))
<=sum(einT)(Z[j][e]*TS[e]) +1000*(1-X[i][j]);//LT[i] + s[i.Class][j.Class] <=
LT[j] + 1000*(1 - X[i][j]);

cons4://///minimum separation
forall(iinone,jinone:i!=j)//     forall(i in one, j in one : i !=j)
       sum(einT) (Z[j][e]*(TS[e] +s[j.Class][i.Class]))
<=sum(tinT)(Z[i][t]*TS[t]) +1000*X[i][j];     //LT[j] + s[j.Class][i.Class] <=
LT[i] + 1000*X[i][j];

}

executefinish
```

```
{
       thisOplModel.settings.mainEndEnabled=true;
       for(varfinone) {
              for(vartinT) {
                     ZZ[f][t] =Z[f][t]
                     if(Z[f][t]==1)
                     writeln("Aircraft "+f.No+": Landing at "+ (t));
              }
       }
}
```

Appendix III

AnyLogic-OptQuest

```
try {

traceln("Start solving");

        // Create Engine

        Engine engine = createEngine();

        // Set stop time, initialize random number generator:

        engine.setStopTime(50);

        engine.setDefaultRandomGenerator(new Random());

 // Create optimization engine

 final COptQuestOptimization opt =
ExperimentOptimization.createOptimization(engine);

//int P = Main.controller.collection.size();

             /////////////////////////////////////////Data

double [ ] [ ] S = {{ 1.6, 2.62, 3.27 },

                    { 1, 1.15, 2.18 },

                    { 1, 1.15, 1.37 },

               };

ArrayList<Double> ELL = new ArrayList<Double>();

 for (Aircrafts ac : controller.collection){
         ELL.add(ac.EL);
 }
         for(int i=0; i<ELL.size();i++){
       traceln("ELL"+(i)+" from solver: " + ELL.get(i));
        }
        ArrayList<Integer> Type = new ArrayList<Integer>();
         for (Aircrafts ac : controller.collection){
        Type.add(ac.type - 1 );  }
         for(int i=0; i<Type.size();i++){
         traceln("Type"+(i)+" from solver" + Type.get(i));
        }
    //////////////////////////////////////////////////////Dicision Variable
//int[][] x = new int [P][A];
ArrayList<ArrayList<COptQuestBinaryVariable>> v = new
ArrayList<ArrayList<COptQuestBinaryVariable>>();
for (int i = 0 ; i < controller.collection.size(); i++){
       //define an arraylist A
       ArrayList<COptQuestBinaryVariable> A = new
ArrayList<COptQuestBinaryVariable>();
       for(int j = 0; j <controller.collection.size(); j++){
        final COptQuestBinaryVariable x = new COptQuestBinaryVariable();
        //add x to A
         A.add(x);
       }
       //add A to v
       v.add(A);
}
for (int i=0; i<v.size(); i++){
       for (int j=0; j<v.get(i).size(); j++){
              opt.AddVariable(v.get(i).get(j));
       }
}
```

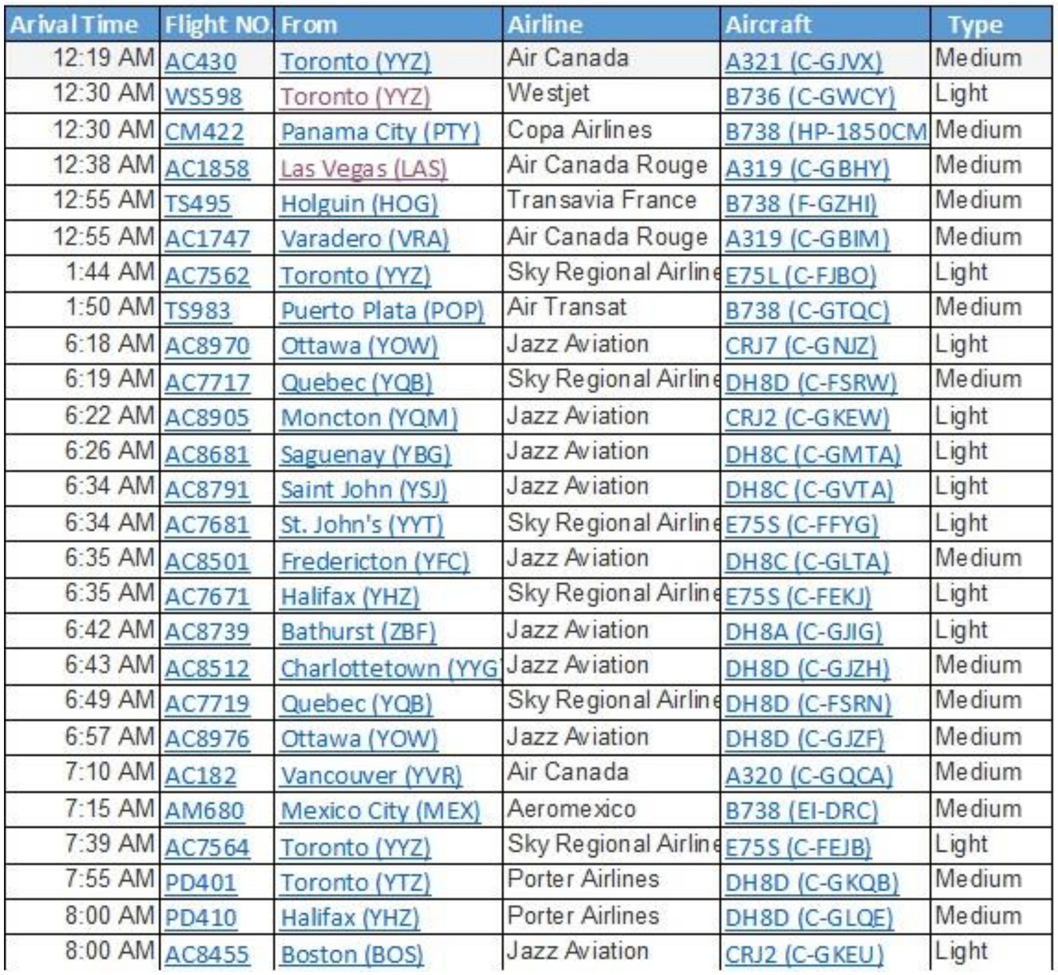
```
//double [][] LT = new double[20][3];

ArrayList<ArrayList<COptQuestContinuousVariable>> LT = new
ArrayList<ArrayList<COptQuestContinuousVariable>>();

for (int i = 0 ; i < controller.collection.size(); i++){

  ArrayList<COptQuestContinuousVariable> B = new
ArrayList<COptQuestContinuousVariable>();

       for(int j = 0; j <3; j++){

        final COptQuestContinuousVariable y = new
COptQuestContinuousVariable();

  LT.SetLowerBound(0);

LT.SetUpperBound(1000);

   B.add(y);

       }

       LT.add(B);

}

traceln(" ************** " );

for (int i=0; i<LT.size(); i++){

traceln("defining dv(row): " + LT.size());

       for (int j=0; j<LT.get(i).size(); j++){

       traceln("defining dv(column): " + LT.get(i).size());

              opt.AddVariable(LT.get(i).get(j));

       }

}

final COptQuestContinuousVariable max = new COptQuestContinuousVariable();

        max.SetLowerBound(1.0);

        max.SetUpperBound(1000.0);

opt.AddVariable(max);

///////////////////Objective Function

  final COptQuestObjectiveFunction obj = new COptQuestObjectiveFunction();

        obj.SetMinimize();

        obj.AddVariable(max,1);

         opt.AddObjective(obj);
```

```
 //////////////////////Contraints
 //1)max>= LT[i][a]    for all i in ac.NO
 traceln("----------");
 traceln("Collection in solver: " + controller.collection.size());
 traceln("type in solver: " + Type.size());
 for (int i = 0 ; i < controller.collection.size(); i++){
int a = Type.get(i);
traceln("LT in solver: " + LT.size());
  final  COptQuestGEConstraint constraint1 = new  COptQuestGEConstraint();
constraint1.AddVariable(max,1);
 traceln("a: " + a);
 constraint1.AddVariable(LT.get(i).get(a),-1);
constraint1.SetRHS(0);
     opt.AddConstraint(constraint1);
 }
 //2)x[i][j]+x[j][i]=1   for all i&j in collection.size()
 for (int i = 0 ; i < controller.collection.size(); i++){
      for(int j = 0; j < controller.collection.size(); j++){
      if(i!=j){
 final  COptQuestEQConstraint constraint2 = new  COptQuestEQConstraint();
constraint2.AddVariable(v.get(i).get(j),1);
constraint2.AddVariable(v.get(j).get(i),1);
  constraint2.SetRHS(1);
 opt.AddConstraint(constraint2);
}
}}
 //3) LT[i][a] +S[a][b] <= LT[j][b] + 1000*(1-x[i][j])
 for (int i = 0 ; i < controller.collection.size(); i++){
      for(int j = 0; j < controller.collection.size(); j++){
        if(i!=j){
        int a = Type.get(i);
```

```
          int b = Type.get(j);
   final  COptQuestLEConstraint constraint3 = new  COptQuestLEConstraint();
constraint3.AddVariable(LT.get(i).get(a),1);
constraint3.AddVariable(LT.get(j).get(b),-1);
 constraint3.AddVariable(v.get(i).get(j),1000);
  constraint3.SetRHS(1000-S[a][b]);
opt.AddConstraint(constraint3);
 }}}
  //4)LT[j][b]+S[a][b]<= LT[i][a]+1000*x[i][j]
 for (int i = 0 ; i < controller.collection.size(); i++){
for(int j = 0; j < controller.collection.size(); j++){
        if(i!=j){
        int a = Type.get(i);
         int b = Type.get(j);
  final  COptQuestLEConstraint constraint4 = new  COptQuestLEConstraint();
constraint4.AddVariable(LT.get(j).get(b),1);
constraint4.AddVariable(LT.get(i).get(a),-1);
 constraint4.AddVariable(v.get(i).get(j),-1000);
 constraint4.SetRHS(-S[a][b]);
 opt.AddConstraint(constraint4);
 }}}
 //5) LT[i][a] >= EL[i]
 for (int i = 0 ; i < controller.collection.size(); i++){
int a = Type.get(i);
  final  COptQuestGEConstraint constraint5 = new  COptQuestGEConstraint();
constraint5.AddVariable(LT.get(i).get(a),1);
constraint5.SetRHS(ELL.get(i));
 opt.AddConstraint(constraint5);
}
 //6) LT[i][a] <= LL[i]
   for (int i = 0 ; i < controller.collection.size(); i++){
```

```
int a = Type.get(i);
  final  COptQuestLEConstraint constraint6 = new  COptQuestLEConstraint();
constraint6.AddVariable(LT.get(i).get(a),1);
 //constraint6.AddVariable(-1,ELL.get(i));
  constraint6.SetRHS(ELL.get(i)+5);
  opt.AddConstraint(constraint6);
  }
// Set the number of iterations to run
opt.SetMaximumIterations(50);
// Add suggested solution (initial solution)
//COptQuestSolution suggestedSolution = opt.CreateSolution();
//suggestedSolution.SetVariableValue(v, 50.0);
//opt.AddSuggestedSolution(suggestedSolution);
// Perform optimization
opt.Optimize();
//solution.SetObjectiveValue(obj, root.objective);
// Output results
COptQuestSolution bestSolution = opt.GetBestSolution();
//variable = bestSolution.GetVariableValue(v);
traceln("//////////Best objective: " +
format(bestSolution.GetObjectiveValue(obj)));
//main.parameter = variable;
//  main.log.println();
for (int i = 0 ; i < controller.collection.size(); i++){
      for(int j = 0; j < controller.collection.size(); j++){
        if(i!=j){
             variable = bestSolution.GetVariableValue(v.get(i).get(j));
      traceln("//////////////x"+(i)+(j)+"= " +
format(bestSolution.GetVariableValue(v.get(i).get(j))));}}}
      for (int i = 0 ; i < controller.collection.size(); i++){
       int a = Type.get(i);
        variable = bestSolution.GetVariableValue(LT.get(i).get(a));
        traceln("//////////////LT"+(i)+(a)+"= " +
format(bestSolution.GetVariableValue(LT.get(i).get(a))));}

        // main.log.println("Best objective: " +
format(bestSolution.GetObjectiveValue(obj)));

} catch (COptQuestException e) {

traceln(e.Description());

//main.log.println("!!!!!!!!" + e.Description());

 }
```

Data Sample:



Arival Time	Flight NO.	From	Airline	Aircraft	Type
12:19 AM	AC430	Toronto (YYZ)	Air Canada	A321 (C-GJVX)	Medium
12:30 AM	WS598	Toronto (YYZ)	Westjet	B736 (C-GWCY)	Light
12:30 AM	CM422	Panama City (PTY)	Copa Airlines	B738 (HP-1850CM	Medium
12:38 AM	AC1858	Las Vegas (LAS)	Air Canada Rouge	A319 (C-GBHY)	Medium
12:55 AM	TS495	Holguin (HOG)	Transavia France	B738 (F-GZHI)	Medium
12:55 AM	AC1747	Varadero (VRA)	Air Canada Rouge	A319 (C-GBIM)	Medium
1:44 AM	AC7562	Toronto (YYZ)	Sky Regional Airline	E75L (C-FJBO)	Light
1:50 AM	TS983	Puerto Plata (POP)	Air Transat	B738 (C-GTQC)	Medium
6:18 AM	AC8970	Ottawa (YOW)	Jazz Aviation	CRJ7 (C-GNJZ)	Light
6:19 AM	AC7717	Quebec (YQB)	Sky Regional Airline	DH8D (C-FSRW)	Medium
6:22 AM	AC8905	Moncton (YQM)	Jazz Aviation	CRJ2 (C-GKEW)	Light
6:26 AM	AC8681	Saguenay (YBG)	Jazz Aviation	DH8C (C-GMTA)	Light
6:34 AM	AC8791	Saint John (YSJ)	Jazz Aviation	DH8C (C-GVTA)	Light
6:34 AM	AC7681	St. John's (YYT)	Sky Regional Airline	E75S (C-FFYG)	Light
6:35 AM	AC8501	Fredericton (YFC)	Jazz Aviation	DH8C (C-GLTA)	Medium
6:35 AM	AC7671	Halifax (YHZ)	Sky Regional Airline	E75S (C-FEKJ)	Light
6:42 AM	AC8739	Bathurst (ZBF)	Jazz Aviation	DH8A (C-GJIG)	Light
6:43 AM	AC8512	Charlottetown (YYG	Jazz Aviation	DH8D (C-GJZH)	Medium
6:49 AM	AC7719	Quebec (YQB)	Sky Regional Airline	DH8D (C-FSRN)	Medium
6:57 AM	AC8976	Ottawa (YOW)	Jazz Aviation	DH8D (C-GJZF)	Medium
7:10 AM	AC182	Vancouver (YVR)	Air Canada	A320 (C-GQCA)	Medium
7:15 AM	AM680	Mexico City (MEX)	Aeromexico	B738 (EI-DRC)	Medium
7:39 AM	AC7564	Toronto (YYZ)	Sky Regional Airline	E75S (C-FEJB)	Light
7:55 AM	PD401	Toronto (YTZ)	Porter Airlines	DH8D (C-GKQB)	Medium
8:00 AM	PD410	Halifax (YHZ)	Porter Airlines	DH8D (C-GLQE)	Medium
8:00 AM	AC8455	Boston (BOS)	Jazz Aviation	CRJ2 (C-GKEU)	Light

8:00 AM	AC8455	Boston (BOS)	Jazz Aviation	CRJ2 (C-GKEU)	Light
8:00 AM	AC7631	New York (LGA)	Sky Regional Airlines	E75S (C-FEJP)	Light
8:19 AM	WS578	Toronto (YYZ)	Westjet	B736 (C-GXWJ)	Light
8:19 AM	AC400	Toronto (YYZ)	Air Canada	A320 (C-FDCA)	Light
8:24 AM	AC8709	Quebec (YQB)	Jazz Aviation	DH8D (C-GJZN)	Medium
8:25 AM	AC7502	Toronto (YTZ)	Sky Regional Airlines	DH8D (C-FSRJ)	Medium
8:27 AM	AC8750	Val-d'Or (YVO)	Jazz Aviation	DH8A (C-FGRP)	Medium
8:30 AM	AC7301	Windsor Locks (BDL)	Air Georgian	B190 (C-GHGA)	Medium
8:40 AM	PD403	Toronto (YTZ)	Porter Airlines	DH8D (C-GKQF)	Medium
8:57 AM	AA4374	New York (LGA)	American Eagle	E145 (N619AE)	Medium
9:01 AM	WG774	Quebec (YQB)	Sunwing Airlines	B738 (C-GUUL)	Medium
9:19 AM	WS580	Toronto (YYZ)	Westjet	B737 (C-GCWJ)	Medium
9:19 AM	AC402	Toronto (YYZ)	Air Canada	A321 (C-GJWI)	Medium
9:20 AM	AA3864	Philadelphia (PHL)	American Eagle	CRJ2 (N416AW)	Light
9:25 AM	AC7504	Toronto (YTZ)	Sky Regional Airlines	DH8D (C-FSRY)	Medium
9:32 AM	UA3938	New York (EWR)	United Express	E45X (N23139)	Medium
9:34 AM	AC8978	Ottawa (YOW)	Jazz Aviation	CRJ2 (C-GQJA)	Light
9:39 AM	DL5414	New York (LGA)	Delta Connection	CRJ7 (N391CA)	Light
9:40 AM	PD405	Toronto (YTZ)	Porter Airlines	DH8D (C-GLQD)	Medium
10:00 AM	AC7563	Halifax (YHZ)	Sky Regional Airlines	E75L (C-FUJE)	Light
10:10 AM	AC7633	New York (LGA)	Sky Regional Airlines	E75S (C-FEJP)	Light
10:12 AM	AC1601	Fort Lauderdale (FLL)	Air Canada Rouge	B763 (C-GHLV)	Heavy
10:13 AM	DL5234	Detroit (DTW)	Delta Connection	CRJ7 (N759EV)	Light
10:19 AM	AC404	Toronto (YYZ)	Air Canada	A321 (C-GJWD)	Medium
10:25 AM	AC7506	Toronto (YTZ)	Sky Regional Airlines	DH8D (C-FSRW)	Medium
10:29 AM	WS3474	Toronto (YYZ)	WestJet Encore	DH8D (C-GVWE)	Medium
10:35 AM	PD407	Toronto (YTZ)	Porter Airlines	DH8D (C-GLQO)	Medium
10:37 AM	AA4346	New York (JFK)	American Eagle	E145 (N836HK)	Medium
10:38 AM	AC8594	Winnipeg (YWG)	Jazz Aviation	CRJ7 (C-FTJZ)	Light
10:43 AM	AC8818	Hamilton (YHM)	Jazz Aviation	CRJ2 (C-GKER)	Light
10:54 AM	UA4362	Chicago (ORD)	United Express	E145 (N11536)	Medium
11:05 AM	AC8505	Fredericton (YFC)	Jazz Aviation	CRJ2 (C-GKEW)	Light
11:10 AM	AC8457	Boston (BOS)	Jazz Aviation	CRJ2 (C-GJZZ)	Light
11:16 AM	N5591	Toronto (YYZ)	-	733	
11:17 AM	DL4043	New York (JFK)	Delta Connection	CRJ2 (N8977A)	Light
11:19 AM	AC406	Toronto (YYZ)	Air Canada	A321 (C-FJNX)	Medium
11:21 AM	AC8687	Saguenay (YBG)	Jazz Aviation	DH8C (C-GMTA)	Medium
11:24 AM	AC8699	Sept-Iles (YZV)	Jazz Aviation	DH8A (C-GION)	Medium
11:25 AM	AC7508	Toronto (YTZ)	Sky Regional Airlines	DH8D (C-FSRZ)	Medium
11:29 AM	WS3480	Toronto (YYZ)	WestJet Encore	DH8D (C-FKWE)	Medium
11:32 AM	AC8789	Saint John (YSJ)	Jazz Aviation	CRJ2 (C-GGJA)	Light
11:35 AM	PD411	Toronto (YTZ)	Porter Airlines	DH8D (C-GLQL)	Medium
11:52 AM	DL6260	Detroit (DTW)	Delta Connection	CRJ7 (N376CA)	Light
11:54 AM	AA3128	Chicago (ORD)	American Eagle	CRJ2 (N869AS)	Light
12:05 PM	AC8711	Quebec (YQB)	Jazz Aviation	DH8C (C-GLTA)	Medium
12:09 PM	AC8429	Ottawa (YOW)	Jazz Aviation	DH8A (C-GTAI)	Medium
12:16 PM	DL5426	Atlanta (ATL)	Delta Connection	CRJ9 (N676CA)	Medium
12:19 PM	AC408	Toronto (YYZ)	Air Canada	A320 (C-FTJS)	Light
12:20 PM	AC8756	Rouyn (YUY)	Jazz Aviation	DH8A (C-GJIG)	Medium
12:20 PM	AC8491	New York (EWR)	Jazz Aviation	CRJ2 (C-GKEU)	Light
12:20 PM	AC7635	New York (LGA)	Sky Regional Airlines	E75S (C-FFYG)	Light
12:20 PM	AC875	Frankfurt (FRA)	Air Canada	B77W (C-FITU)	Heavy
12:21 PM	YN461	Montreal (YMX)	Air Creebec	DH1	
12:25 PM	AC7510	Toronto (YTZ)	Sky Regional Airlines	DH8D (C-FSRN)	Medium
12:25 PM	AC7303	Windsor Locks (BDL)	Air Georgian	B190 (C-GHGA)	Medium
12:29 PM	AC8781	Halifax (YHZ)	Jazz Aviation	CRJ7 (C-GNJZ)	Light
12:40 PM	PD420	Halifax (YHZ)	Porter Airlines	DH8D (C-GKQB)	Medium
12:42 PM	TS24	Windsor (YQG)	Air Transat	B737 (C-GTQP)	Medium
12:42 PM	AA4319	New York (LGA)	American Eagle	E145 (N623AE)	Medium
12:45 PM	AC7515	Moncton (YQM)	Sky Regional Airlines	DH8D (C-FSRJ)	Medium
12:48 PM	AA4008	Philadelphia (PHL)	American Eagle	CRJ2 (N444ZW)	Light
1:00 PM	AC7590	Chicago (ORD)	Sky Regional Airlines	E75S (C-FEKJ)	Light
1:05 PM	PD413	Toronto (YTZ)	Porter Airlines	DH8D (C-GLQD)	Medium
1:05 PM	CA879	Beijing (PEK)	Air China	B77W (B-2087)	Heavy
1:06 PM	AC8825	Washington (DCA)	Jazz Aviation	CRJ2 (C-FIJA)	Light

References:

Shen, Weiming. 2002. "Distributed manufacturing scheduling using intelligent agents." *IEEE intelligent systems, pages 88-94.*

Shen, Weiming, Lihui Wang, and Qi Hao. 2006. "Agent-Based Distributed Manufacturing Process Planning and Scheduling: A State-of-the-Art Survey." *IEEE Transactions on Systems, Man and Cybernetics. Part C, Applications and Reviews: A Publication of the IEEE Systems, Man, and Cybernetics Society* 36 (4): 563–77.

Brentnall, A. R., and R. C. H. Cheng. 2008. "Some Effects of Aircraft Arrival Sequence Algorithms." *The Journal of the Operational Research Society* 60 (7). Palgrave Macmillan UK: 962–72.

Salehipour, Amir, Leila Moslemi Naeni, and Hamed Kazemipoor. 2009. "Scheduling Aircraft Landings by Applying a Variable Neighborhood Descent Algorithm: Runway-Dependent Landing Time Case." *Journal of Applied Operational Research* 1 (1): 39–49.

Soomer, M. J., and G. J. Franx. 2008. "Scheduling Aircraft Landings Using Airlines' Preferences." *European Journal of Operational Research* 190 (1): 277–91.

Pinol, H., and J. E. Beasley. 2006. "Scatter Search and Bionomic Algorithms for the Aircraft Landing Problem." *European Journal of Operational Research* 171 (2): 439–62.

Artiouchine, Konstantin, Philippe Baptiste, and Christoph Dürr. 2008. "Runway Sequencing with Holding Patterns." *European Journal of Operational Research* 189 (3): 1254–66.

Farhadi, Farbod, Ahmed Ghoniem, and Mohammed Al-Salem. 2014/8. "Runway Capacity Management – An Empirical Study with Application to Doha International Airport." *Transportation Research Part E: Logistics and Transportation Review* 68: 53–63.

Bennell, Julia A and Mesgarpour, Mohammad and Potts, Chris N. 2013 *"Airport runway scheduling." Annals of Operations Research*

Sousa, Paulo, and Carlos Ramos. 1999. "A Distributed Architecture and Negotiation Protocol for Scheduling in Manufacturing Systems." *Computers in Industry* 38 (2): 103–13.

Tumer, Kagan, and Adrian Agogino. 2007. "Distributed Agent-Based Air Traffic Flow Management." In *Proceedings of the 6th International Joint Conference on Autonomous*

Hu, Xiao-Bing, and Wen-Hua Chen. 2005/8. "Genetic Algorithm Based on Receding Horizon Control for Arrival Sequencing and Scheduling." *Engineering Applications of Artificial Intelligence* 18 (5): 633–42.

Liu, Yu-Hsin. 2010. "A Genetic Local Search Algorithm with a Threshold Accepting Mechanism for Solving the Runway Dependent Aircraft Landing Problem." *Optimization Letters* 5 (2). Springer-Verlag: 229–45.

Lieder, Alexander, Dirk Briskorn, and Raik Stolletz. 2015. "A Dynamic Programming Approach for the Aircraft Landing Problem with Aircraft Classes." *European Journal of Operational Research* 243 (1): 61–69.

Godin, Patrice, and Chun Wang. 2010. "Agent-Based Outpatient Scheduling for Diagnostic Services." In *2010 IEEE International Conference on Systems, Man and Cybernetics*. doi:10.1109/icsmc.2010.5642281.

Smith, Reid G.1980"The contract net protocol: High-level communication and control in a distributed problem solver." *IEEE Transactions on computers. Pages 1104-1113*

Samà, Marcella, Andrea D'Ariano, Paolo D'Ariano, and Dario Pacciarelli. 2014. "Comparing Centralized and Rolling Horizon Approaches for Optimal Aircraft Traffic Control in Terminal Areas." *Transportation Research Record: Journal of the Transportation Research Board* 2449: 45–52.

Ghoniem, Ahmed, and Farbod Farhadi. 2015. "A Column Generation Approach for Aircraft Sequencing Problems: A Computational Study." *The Journal of the Operational Research Society* 66 (10). Palgrave Macmillan UK: 1717–29.

Ghoniem, Ahmed, Farbod Farhadi, and Mohammad Reihaneh. 2015. "An Accelerated Branch-and-Price Algorithm for Multiple-Runway Aircraft Sequencing Problems." *European Journal of Operational Research* 246 (1): 34–43.

Ernst, Andreas T., Mohan Krishnamoorthy, and Robert H. Storer. 1999. "Heuristic and Exact Algorithms for Scheduling Aircraft Landings." *Networks. An International Journal* 34 (3): 229.

Hu, X. B., and E. Di Paolo. 2008. "Binary-Representation-Based Genetic Algorithm for Aircraft Arrival Sequencing and Scheduling." *IEEE Transactions on Intelligent Transportation Systems* 9 (2): 301–10.

Beasley, John E and Krishnamoorthy, Mohan and Sharaiha, Yazid M and Abramson, D 2000." *Scheduling aircraft landings—the static case". Transportation science. Pages 180-197*

Balakrishnan, Hamsa. n.d. "Algorithms for Scheduling Runway Operations Under Constrained Position Shifting." doi:10.1287/opre.1100.0869.

Zhan, Z. H., J. Zhang, Y. Li, O. Liu, S. K. Kwok, W. H. Ip, and O. Kaynak. 2010. “An Efficient Ant Colony System Based on Receding Horizon Control for the Aircraft Arrival Sequencing and Scheduling Problem.” *IEEE Transactions on Intelligent Transportation Systems* 11 (2): 399–412.

Wang, Chun and Wang, Zhiguo and Ghenniwa, Hamada H and Shen, Weiming. 2011.” Due-date management through iterative bidding” *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*

Bayen, A. M., C. J. Tomlin, Yinyu Ye, and Jiawei Zhang. 2004. “An Approximation Algorithm for Scheduling Aircraft with Holding Time.” In *2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No.04CH37601).* doi:10.1109/cdc.2004.1428880.

Briskorn, Dirk, and Raik Stolletz. 2013. “Aircraft Landing Problems with Aircraft Classes.” *Journal of Scheduling* 17 (1). Springer US: 31–45.

Brentnall AR (2006) Aircraft arrival management. PhD thesis, University of Southampton, UK

Lucio Bianco, Paolo Dell’Olmo, and Stefano Giordani.1997. "Scheduling Models and Algorithms for TMA Traffic Management"

Ball, Michael and Donohue, George and Hoffman, Karla. 2006.” Auctions for the safe, efficient, and equitable allocation of airspace system resources”. *Combinatorial auctions. MIT Press Cambridge, MA*

Rassenti, Stephen J and Smith, Vernon L and Bulfin, Robert L.1982.” A combinatorial auction mechanism for airport time slot allocation”. *The Bell Journal of Economics. Pages 402-417. JSTOR*

Sheng, Dian and Li, Zhi-Chun and Xiao, Yi-bin and Fu, Xiaowen. 2015.” Slot auction in an airport network with demand uncertainty”. *Transportation Research Part E: Logistics and Transportation Review. Pages 79-100. Elsevier*

http://www.iata.org/Pages/default.aspx

https://www-01.ibm.com/software/commerce/optimization/modeling/

www.skyscanner.ca