Summer 2013

Approximate Algorithms for the Combined arrival-Departure Aircraft Sequencing and Reactive Scheduling Problems on Multiple Runways

Gulsah Hancerliogullari
Old Dominion University

APPROXIMATE ALGORITHMS FOR THE COMBINED ARRIVAL-DEPARTURE AIRCRAFT SEQUENCING AND REACTIVE SCHEDULING PROBLEMS ON MULTIPLE RUNWAYS

by

Gulsah Hancerliogullari
B.S. May 2008, Bilkent University, Turkey
M.S. July 2010, Bilkent University, Turkey

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

ENGINEERING MANAGEMENT

OLD DOMINION UNIVERSITY
August 2013

ABSTRACT

APPROXIMATE ALGORITHMS FOR THE COMBINED ARRIVAL-DEPARTURE AIRCRAFT SEQUENCING AND REACTIVE SCHEDULING PROBLEMS ON MULTIPLE RUNWAYS

Gulsah Hancerliogullari
Old Dominion University, 2013
Director: Dr. Ghaith Rabadi

The problem addressed in this dissertation is the Aircraft Sequencing Problem (ASP) in which a schedule must be developed to determine the assignment of each aircraft to a runway, the appropriate sequence of aircraft on each runway, and their departing or landing times. The dissertation examines the ASP over multiple runways, under mixed mode operations with the objective of minimizing the total weighted tardiness of aircraft landings and departures simultaneously. To prevent the dangers associated with wake-vortex effects, separation times enforced by Aviation Administrations (e.g., FAA) are considered, adding another level of complexity given that such times are sequence-dependent. Due to the problem being NP-hard, it is computationally difficult to solve large scale instances in a reasonable amount of time. Therefore, three greedy algorithms, namely the Adapted Apparent Tardiness Cost with Separation and Ready Times (AATCSR), the Earliest Ready Time (ERT) and the Fast Priority Index (FPI) are proposed. Moreover, metaheuristics including Simulated Annealing (SA) and the Metaheuristic for Randomized Priority Search (Meta-RaPS) are introduced to improve solutions initially constructed by the proposed greedy algorithms. The performance (solution quality and computational time) of the various algorithms is compared to the optimal solutions and to each other.

The dissertation also addresses the Aircraft Reactive Scheduling Problem (ARSP) as air traffic systems frequently encounter various disruptions due to unexpected events such as inclement weather, aircraft failures or personnel shortages rendering the initial plan suboptimal or even obsolete in some cases. This research considers disruptions including the arrival of new aircraft, flight cancellations and aircraft delays. ARSP is formulated as a multi-objective optimization problem in which both the schedule's quality and stability are of interest. The objectives consist of the total weighted start times (solution quality), total weighted start time deviation, and total weighted runway deviation (instability measures). Repair and complete regeneration approximate algorithms are developed for each type of disruptive events. The algorithms are tested against difficult benchmark problems and the solutions are compared to optimal solutions in terms of solution quality, schedule stability and computational time.

This dissertation is dedicated to my mom Kaytan and my dad Ahmet.

ACKNOWLEDGEMENTS

Even though writing a dissertation is an individual work, I could never have succeeded in completing this dissertation without the help, support, guidance and efforts of many people.

First and foremost, I would like to express my deepest gratitude to my advisor, Dr. Ghaith Rabadi, for his endless patience, guidance, support and encouragement. Throughout my doctoral study, he always believed in me and always made me feel that I could do better and better. His way of thinking, solving problems, teaching, and understanding, led me to admire him. Not only has Professor Rabadi shaped who I am as a scientist, but he also shaped my teaching, communication and collaboration skills. I have always said this to everyone, but I would like to express it here, one more time that I couldn't have finished this work without him. Thank you very much for everything Professor!

I would like to thank other members of my dissertation committee, Dr. Resit Unal, Dr. Ariel C. Pinto and Dr. Mecit Cetin for providing me support, feedback and encouragement during the improvement of this dissertation.

I especially would like to thank Dr. Resit Unal for his numerous avenues of support which include recommending opportunities to me on countless occasions. To me, he was not only the department chair of EMSE but also a supportive role model who is concerned with all of his students' success.

I would like to extend my special appreciation to Dean of Frank Batten College of Engineering & Technology, Dr. Oktay Baysal and Associate Dean, Dr. Osman Akan for providing me the opportunity to pursue my Ph.D. I also would like to thank Dr. Ahmed Ghoniem and Farbod Farhadi from University of Massachusetts Amherst, Dr. Mohamed Kharbeche and Dr. Ameer Al-Salem from Qatar University and Dr. Hanif D. Sherali from Virginia Tech for their support and feedback during the collaborative research project.

I am deeply grateful to my family for their encouragement and unbending love not only during this study, but also throughout my life. I feel very lucky to have such a wonderful family. My very precious thanks to my mom Kaytan, my dad Ahmet, my sister, Basak, my brother, Dr. Kadir Oymen, my nephew Burak Emirhan and my sister-in-law Nihal. Whenever I needed courage, talking to them helped ease everything.

I am very grateful to all of my friends and family here in the States and in Turkey. In this long journey, miles away from home, people I have met here have made me feel like I am home again. Many thanks to Dr. Mujde Erten Unal, Deniz Unal, Mariam Kotachi, Dr. Ersin Ancel, Dr. Berna Eren Tokgoz. I also owe a special note of gratitude to Cansu Kandemir, Elnaz Dario and Alireza Shahvari who I've known for a couple of months but it feels like an eternity; they always cheered me up when I was in a bad mood with their big smiles and warm friendship. Everytime I visited Turkey, all of my friends there made me feel like I never left. These countless number of friends, who are now all around the world, especially, Esra Agca, Ardic Kocaman, Ece Zeliha Demirci, Hatice Calik, Pelin Damci Kurt, Gokce Akin deserve big thanks from me.

Last and most importantly, Emrah Koksalmis, who has given me the biggest support during my doctoral study deserves the most special thanks. He always listened to my research ideas patiently and gave me useful feedback. There are no words that can express my gratitude to him.

NOMENCLATURE

AATCSR	Adapted Apparent Tardiness Cost with Separation and Ready Times
ASP	Aircraft Sequencing Problem
ATSP	Asymmetric Traveling Salesman Problem
B&B	Branch-and-bound
CPS	Constrained Position Shifting
CSH	Cheapest search heuristic
ERT	Earliest Ready Time
Eurocontrol	European Organisation for the Safety of Air Navigation
FAA	Federal Aviation Administration
FCFS	First-Come First-Served
FPI	Fast Priority Index
GA	Genetic Algorithm
GRASP	Greedy randomized adaptive search procedure
ICAO	International Civil Aviation Organization
Meta-RaPS	Meta-heuristic for randomized priority search
MILP	Mixed-integer Linear Programming
SA	Simulated Annealing
TMA	Terminal Maneuvering Area

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The current capacity of airports is becoming insufficient due to growing air transportation demand and a huge increase in air traffic during the last decade. Therefore, some aircraft cannot land or depart at their preferred target-time. In order to achieve an efficient use of critical resources such as runways, devising appropriate methods for aircraft sequencing problem (ASP) is of great importance and is the main aim of this dissertation. Airport terminal maneuvering area (TMA) is of great interest to decision-makers since it is a critical link of air traffic operations chain. TMA includes managing air traffic control operations, runway scheduling and taxiway operations. Among these operations, runway scheduling is the one that affects the performance of the TMA the most (Sherali et al., 1992).

Although there are many ongoing studies related to the operations that take place in the TMA, the cost of the flight delays are still extremely high. In order to effectively use the low capacity resources such as runways, effective decision support systems are required, air traffic policies have to be identified, and wise planning strategies have to be proposed which require huge amount of time and investment. Researchers have been examining several approaches on the efficient use of runways while considering safety constraints. Since constructing new airports or additional runways is not a near term solution, decision-makers ought to examine competent schedules of aircraft landings and departures to improve the runway throughput capacity. In other words, already existing resources should be evaluated and utilized judiciously. This way, the current and the estimated inefficiencies in operations can be eliminated, and significant advantages in performance can be gained.

According to the U.S. Department of Transportation and Eurocontrol, the flight delays statistics were worrisome due to the fact that in the United States compared to 2006, a 15% increase in flight delays was observed in 2007 which cost $8.1 billion in terms of direct operating costs. Because of weather conditions, traffic volume and airport operations, the Air Travel Consumer Report ATCR (U.S. Department of Transportation

2008) and "Challenges of Growth 2008" by Eurocontrol stated that more than 20% of commercial flights were late by more than 15 minutes.

Scheduling is a decision-making process that concentrates on allocation of resources to tasks over given time periods. Therefore, scheduling approaches can be considered for more effective air traffic operations while maintaining up safety and efficiency. The ASP concurrently determines the assignment of aircraft to runways, the appropriate sequence of aircraft on each runway, and the departing or landing time on a chosen runway. It is assumed that each runway can accommodate at most one aircraft at any time that runways are reliable, and that they operate independently. The problem can then be modeled as an identical parallel machine scheduling problem with the runways being machines and the aircraft being jobs that have ready times (release times), target times (due dates), deadlines, tardiness penalties (weights), and sequence-dependent separation (setup) times.

As all aircraft generate wake vortices, a minimum time or a distance is set between aircraft to prevent the adverse effect; this safety buffer is referred as the separation time. Careful sequencing and scheduling can reduce the long separation and operating times. Minimum separation times between consecutive and certain nonconsecutive operations, and specified time-windows during which operations must take place are two of the major requirements of this scheduling effort. Minimizing the total weighted tardiness is a reasonable objective function to schedule landings and departures as close as possible to their target times. A mixed integer linear programming (MILP) model is provided to find optimal solutions. However, since minimizing the total weighted tardiness even for a single machine with all weights being equal is NP-hard (Lawler, 1982), ASP is also NP-hard, which means that it is computationally difficult to solve large scale instances in a reasonable amount of time. Therefore, it is necessary to develop appropriate methods to reach good quality solutions in reasonable computational times.

Passenger satisfaction is one of the key considerations for airline companies that can be maximized by minimizing flight delays. Throughout the course of daily operations, an airline is faced with the potential of deviations in the planned flight schedule as a result of various unexpected events such as severe weather conditions and unexpected aircraft or personnel failures. Unlike most literature, we consider in this dissertation aircraft related disruptive events as they occur often and are much more frequent than other type of disruptions in air traffic operations. Aircraft reactive scheduling approach is taken in this research and schedule repair algorithms are developed to deal specifically with arrival of new aircraft, flight cancellations and aircraft delays due to irregular events. One of the goals of this dissertation is to propose and validate solution methodologies and heuristic procedures to reschedule the planned flights in the event of irregular and disruptive operations.

1.1 Fundamental Concepts in Aircraft Sequencing

The Aircraft Sequencing Problem (ASP) is an operations research problem, whose goal is to assign the arrival and departure aircrafts to runways and sequence them on each runway simultaneously. Minimum separation times between consecutive and certain nonconsecutive operations, and specified time-windows during which operations must take place are two of the major requirements of this scheduling effort. The time-windows, which are required for the landing and departure operations, specify the earliest and latest times of an aircraft become available on a runway for operation. So as to reduce the wake-vortex effect risk, minimum separation times between consecutive and certain nonconsecutive operations are required. Aircraft operation types such as landing or departure, aircraft weight-class such as heavy, medium, light, and sequence of the operations decided by the air traffic controller affect the magnitude of the separation times.

In the academic literature, ASP has attracted the researchers for over the 20 years as surveyed by Bennell et al. (2011). The tools of operational research and management science, and solution techniques including dynamic programming, branch and bound, heuristics and metaheuristics for aircraft landing and take-off scheduling have been comprehensively reviewed. According to this survey, it is noted that significantly more attention has been dedicated to aircraft landings or departures in contrast to combined arrival-departure cases.

1.1.1 Time Windows

Depending on conditions such as fuel restriction, maximum allowed delay, the airspeed, runway availability, or meeting a connecting flight, the landing time of an aircraft must be within a time window that consists of its earliest and latest possible landing time. This time window should be treated as a hard constraint.

Some aircraft have target departure times, a Calculated Time of Take-off (CTOT), are calculated for smooth congestion at busy destination airports. According to Atkin et al. (2010), the CTOT limits the time that aircraft enters the congested areas to smooth the traffic in the airspace and at the airports. The CTOT defines a fifteen minutes time window for which the goal of the air traffic controllers is to assign a scheduled departure time to each aircraft from five minutes before CTOT to ten minutes after the CTOT, which is a soft constraint (Bennell et al., 2011).

1.1.2 First-Come-First-Served

One of the most commonly used heuristics for aircraft sequencing problem is First-Come-First-Served (FCFS). In terminal areas, planning specialists use estimated landing time for calculation of delays. Depending on the estimated landing time, which is based on the route and speed of aircraft, scheduled landing time is assigned to each aircraft (Neuman and Erzberger 1991). Although it is not always preferred for departure sequence, the order of the aircraft queuing is the FCFS order, which provides an estimated departure time (Carr et al. 2000).

1.1.3 Runway Capacity and Assignment

The runway capacity, which is a crucial constraint in an airport system, is the maximum rate of aircraft arrivals or departures that can be accommodated by a single or multiple runways. The factors that affect it are aircraft type, runway operation type (segregated or mixed), runway occupancy time, availability of taxiways, and weather conditions (Bazargan et al. 2002). In segregated-mode, the runway is merely used for either arrival or departure of the aircraft, whereas mixed-mode allows both landing and departure on the same runway. Atkin (2008) states that mixed-mode is more efficient than segregated-mode; likewise Newell (1979) shows that airport capacity is greater when runways are operated in mixed-mode.

The objective of air traffic controllers is to increase the throughput from the available runways while satisfying safety and operational constraints. As increasing the number of runways is not a practical solution, air traffic controllers should consider different methods while they assign a runway to the landing/departure aircraft. The runway assignment depends on the airport configuration (single runway, parallel or intersecting runways or combination of these), the direction of arriving aircraft, and departure route of the aircraft (Brinton 1992). So as to balance the number of arrivals and departures on a runway, runway allocation, that is affected by airlines' preferences, controllers' considerations such as safety and shorter flight times, is done (Isaacson et al., 1997).

1.1.4 Separation

Minimum separation between aircraft landing and departure has a great impact on the runway throughput at an airport. Careful sequencing and scheduling can reduce the number of long separation and operating times; therefore, the separation time makes aircraft sequencing and scheduling problem an important and non-trivial. Reason for setting minimum separation is to prevent the adverse effect of vortices. Because of the rolling moment it can impose on a following aircraft, a wake vortex (WV) is dangerous. The illustration of wake vortex effect is shown in Figure 1.

In order to have safe flight operations, the International Civil Aviation Organization (ICAO) regulates and puts into action separation standards between the leader and the follower aircraft for both landings and departures. The separation standard for landing is based on distance; however, for departure it is based on time (Beasley et al. 2001).

The U.S Federal Aviation Administration (FAA) has established minimum spacing requirements between landing aircraft to prevent the turbulence from wake vortices (Aeronautical Information Manual/Federal Aviation Regulation, 2003).

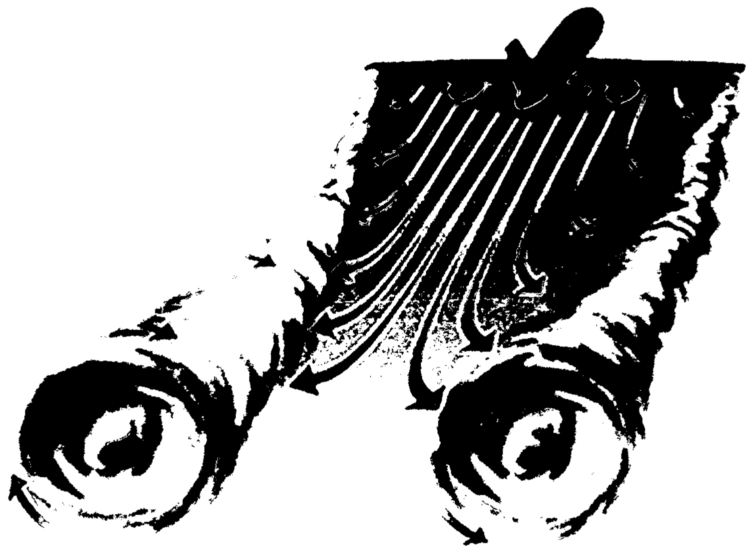


Figure 1. Graphic of wake vortices (http://www.tc.gc.ca/eng/innovation)

For safety reasons, landing/departing a large aircraft necessitates longer time delay before other aircraft can land or depart. On the other hand, a small aircraft generates little air turbulence and therefore it needs only a short time delay. This is illustrated by the International Civil Aviation Organization (ICAO) as follows in Figure 2.

For separation purposes, the FAA divides aircraft into three weight classes, based on the maximum take-off weight capability. *Heavy* aircraft class is capable of having a maximum takeoff weight of 255,000 lbs or more, *Large* aircraft class can have more than 41,000 lbs and up to 255,000 lbs maximum takeoff weight, and *Small* aircraft class is incapable of carrying more than 41,000 lbs takeoff weight. A sample of separation times on landing are shown in Table 1.

Leading Aircraft	Time for Trailing Aircraft (seconds)		
	Heavy	Large	Small
Heavy	96	157	196
Large	60	69	131
Small	60	69	82

Table 1. Minimum time separation (in seconds) landings (FAA, 2003)

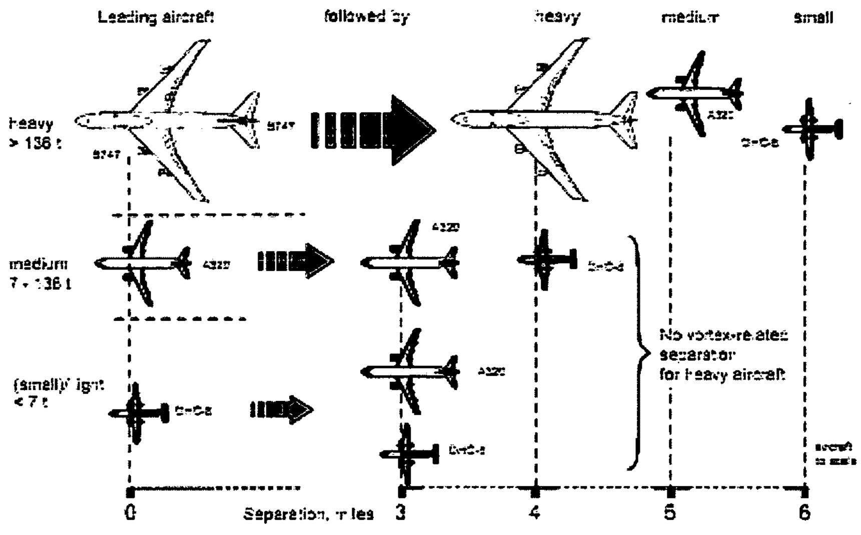


Figure 2. ICAO separations (http://www.liv.ac.uk/flightscience)

Generally, the WV separation rules depend on sequence airspeeds, landing/departure routes, size and types of aircraft, and are asymmetric; i.e., $s_{ij} \neq s_{ji}$. Note that $s_{ij} = s_{ji}$ only if *i* and *j* belong to the same weight class and same operation type. For consecutive operations, separation requirements satisfy the triangle inequality; that is $s_{ik} \leq s_{ij} + s_{jk}$, $\forall i, \forall k \neq i, \forall j \neq i, k$, where s_{ij} is the WV separation between aircraft classes, if the leading aircraft belongs to class *i*, and the trailing aircraft belongs to class *j* (Balakrishnan and Chandran 2006). However, note that in practice separation rules do not obey triangle inequality; it is not adequate just consider the separation from the immediately preceding aircraft. In this research, we will consider separation standards not only for consecutive but also for nonconsecutive operations, which means that the separation times may not necessarily satisfy the triangle inequality.

1.2 Mapping the ASP to Machine Scheduling Problem

Many real life scheduling problems can be modeled as parallel machine scheduling problems. In the classical parallel machine scheduling problem, there are *n* jobs and *m* machines. Each job needs to be executed on one of the machines during a fixed processing time. A parallel machine scheduling problem involves both resource allocation and sequencing. It allocates jobs to machines and determines the sequence of jobs on allocated machine. Machines may be identical, on which the processing time of each job is independent of the assigned machine; uniform where each machine may be have a different speed with known speed factor; or unrelated where the processing time of each job is dependent on the assigned machine without a particular relationship. The aim is to find a schedule that optimizes a certain performance measure(s) such as makespan, maximum lateness or weighted tardiness.

The Aircraft Sequencing and Scheduling Problem can be defined as determining the assignment of each aircraft (job) to runway (machine) and the start time of the operation (landing or departure) for the aircrafts. In order to map this problem to a classical scheduling problem, the following assumptions have to be considered (Blazewicz et al., 2007):

1. Any job can be processed on at most one machine at any time.
2. Preemption is not allowed, meaning that once an operation is started, it must be completed without interruption.
3. Ready times of all jobs are zero, i.e. all jobs are available at the commencement of processing.
4. Machines are always available and reliable.
5. Each machine can process at most one job at any time.
6. Sequence dependent setup times, weights, technological constraints and due dates are deterministic and known in advance where appropriate.

Consequently, the following assumptions are considered for the ASP:

1. Any aircraft can operate on at most one runway at any time.
2. There exists a non-preemptive system where a process cannot be interrupted until it is finished.
3. Ready time can be defined as the earliest available time to take-off at runway end or to land where the taxi time is not included (i.e., ready times are not necessarily zero).
4. Runways are always available and reliable.
5. At most one aircraft is allowed to operate on each runway at any time

6. Sequence dependent separation times, ready times, target times, deadlines, technological constraints, operation type, aircraft sizes and associated penalty weights are deterministic and known in advance.

1.3 Problem Statement

One can imagine a set of aircraft to land/depart, and a decision problem can be stated as which aircraft should land/depart next using which runway. Over the course of a working day, this problem is one that has to be solved repeatedly. If a decision support tool could be developed to assist the controller in making this decision then perhaps more effective use of runway capacity could be made. Actually, one has to do more than decide which aircraft lands/departs next. The air traffic controller has to think ahead and (implicitly or explicitly) form the set of aircraft waiting to land/depart; namely, decide the order in which the aircraft will land/depart as well as their landing/departure. The controller has to guarantee that an aircraft has time to safely reach to the runways so as to land/depart at the proper position in the sequence that an aircraft does not run low on fuel while airborne and that aircraft do not land/depart too close together. The first two conditions imply that for each aircraft, there is a window of time within which it must land/depart, and the final condition means that a reasonable amount of time or distance must elapse between successive landings/departing.

The following the problem elements and definition are used throughout this dissertation:

$J=\{1,2,\ldots, n\}$: A set of n aircraft (landing or departing).

$M=\{1,2,\ldots, m\}$: A set of m identical runways.

Ready time (r_j): The earliest time that aircraft j is ready to take-off at runway end or to land (taxi time is not included). Thus, an aircraft cannot be scheduled before r_j.

Target time (δ_j): Planned time for aircraft j to take-off at runway end or land (taxi time is not included). Landing/departing after the target time is allowed, but then a weighted tardiness penalty is incurred.

Deadline (d_j): Allowable latest time that aircraft *j* to take-off or land after which the operation is infeasible. It is the upper bound of the target-to-deadline window, where $r_j \leq \delta_j \leq d_j$, $\forall j \in J$.

Operation type (Oj): Operation type of aircraft *j*, being a landing or a departure

Size class (C_j): Size class of aircraft *j*, e.g., heavy, large, or small

Weight (w_j): Penalty cost/weight per unit of tardiness for aircraft *j*. It is assigned to aircraft *j* based on its operation type (landing or departure) and its size class (heavy, large, or small). In particular, higher priority is usually assigned to landings over departures and to heavy aircraft over large and small ones. Moreover, in the test-bed $w_{j1}=w_{j2}$ if $O_{j1}=O_{j2}$ and $C_{j1}=C_{j2}$.

Sequence-dependent separation time (s_{kj}): Minimum separation time required between aircraft *k* and *j* if they are respectively the leading and the following aircraft, $\forall\ k, j \in J, k \neq j$. This separation time is dependent on sequence, aircraft *k* and *j* in terms of operation type and aircraft weight class, and independent of the runway.

Start time (t_j): The start time of aircraft *j*. (i.e., the time for departure or landing). That is, $r_j \leq t_j \leq D_j$, and it is desirable to have t_j as close to $\boldsymbol{\delta}_j$ as possible.

Piecewise tardiness (T_j): piecewise tardiness of aircraft *j* with respect to its target-time, $\forall\, j \in J$.

Target time-to-Deadline window (d_j - δ_j): Missing the target time is allowed but not preferred, and a weighted tardiness cost is incurred; on the other hand, missing the deadline is not allowed. For the problem where $d_j > t_j > \delta_j$, the tardiness cost is $T_j = \max(t_j - \delta_j, 0)$. An aircraft is labeled *infeasible* and *not scheduled,* if its start time misses the deadline, where $t_j > d_j$. A time horizon that illustrates the nature of problem is provided in Figure 3.

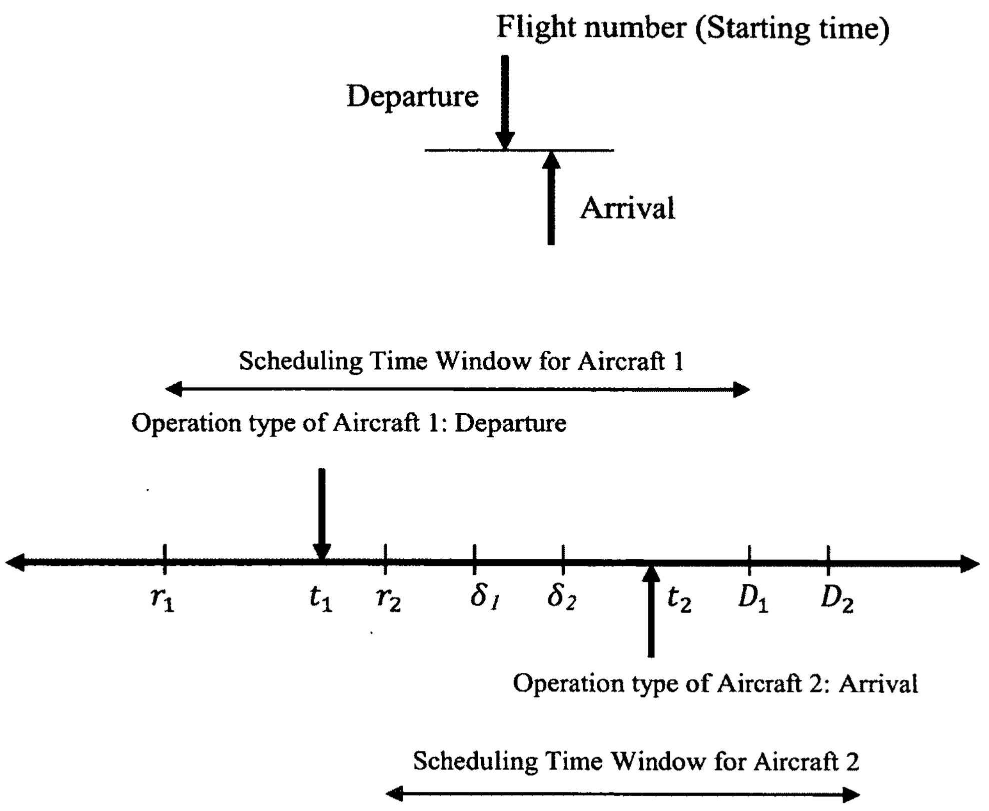

Flight number (Starting time)
Departure
Arrival
Scheduling Time Window for Aircraft 1
Operation type of Aircraft 1: Departure
r_1
t_1
r_2
δ_1
δ_2
t_2
D_1
D_2
Operation type of Aircraft 2: Arrival
Scheduling Time Window for Aircraft 2

Figure 3. Scheduling Time Horizon

Hancerliogullari et al. (2013) studied the ASP over multiple runways, under mixed mode operations with the objective of minimizing the total weighted tardiness of aircraft landings and departures simultaneously. A scheduling problem is described by a triplet $\alpha \mid \beta \mid \gamma$. The α field describes the machine environment, the β field provides details of processing characteristics and constraints, and the γ field describes the objective to be minimized and often contains a single entry (Pinedo, 2008). Using the $\alpha \mid \beta \mid \gamma$ notation of Lawler et al. (1982), the representation of the problem being researched is $Pm|r_j, \boldsymbol{\delta}_j, d_j, s_{kj}, time\ window|\Sigma wjTj$. The ASP can be defined as scheduling n aircraft (jobs) on m identical runways (machines). Each aircraft ($j= 1, \dots, n$) has a penalty weight w_j, becomes ready to operate on a runway at ready time r_j (i.e., aircraft cannot be scheduled before r_j), ought to start its operation (land or depart) by target time $\boldsymbol{\delta}_j$ (planned latest time of an aircraft to operate) and before deadline d_j. A sequence-dependent separation time s_{kj} is enforced to avoid the dangers of wake-vortex effects when aircraft j operates after aircraft k. s_{kj} values depend on aircraft operations (departures, arrivals) and the size-class of the aircraft (small, large, heavy) (Federal Aviation Administration, 2003, Rabadi et al., 2012, Hancerliogullari et al., 2013). For instance, a heavy aircraft requires a larger separation time before a smaller aircraft can land/depart; on the other hand, a small aircraft generates little air turbulence and, therefore, less separation time is necessary if it is scheduled ahead of a larger aircraft. Note that the wake-vortex separation requirements for departures-only or arrivals-only operations satisfy the triangular inequality, which is $s_{ab} + s_{bc} \geq s_{ac}$, if the separation time required between leading aircraft a and trailing aircraft b is s_{ab}. The implication is that when the spacing requirements between successive aircraft are ensured, the spacing requirements for all pairs of aircraft are met. However, the triangle inequality does not necessarily hold when both arrivals and departures are scheduled simultaneously (Balakrishnan and Chandran, 2010), which makes the problem harder to solve.

The start time of the operation for aircraft j is denoted by t_j, and the tardiness by $T_j = \max(t_{j-} \delta_j, 0)$. Missing the target time for aircraft j is possible at a weighted tardiness cost of w_jT_j if it misses its target-time. Missing the deadline, however, is not permitted where if aircraft j misses d_j, it will not be assigned to a runway, and the aircraft in such case is labeled as "unscheduled" resulting in an infeasible schedule. Target time-to-deadline window is the time window during which weighted tardiness cost is incurred; on the other hand, ready time-to-deadline window is the scheduling window in which aircraft have to operate. The scheduling objective is the minimization of the total weighted tardiness (TWT) which is expressed as $\sum_{j=1}^{n} w_jT_j$.

The minimum separation times adopted in this dissertation are specified in Table 2. These minimum safety separation times are enforced by the Federal Aviation Administration (FAA), the national aviation authority of the United States. This precaution is necessary because the triangle inequality does not systematically hold for the separation times. It has been noted in Sherali et al. (2010) and Balakrishnan and Chandran (2010) that the separation times in Table 2 do not automatically ensure proper separation between any pair of aircraft having the same operation type that are interspersed with an aircraft operation of the opposite type (e.g., two landings separated by a departure or two departures interspersed with a landing).

Arrival → Departure Case			
Leading\Following	Heavy	Large	Small
Heavy	40	40	40
Large	35	35	35
Small	30	30	30

Arrival → Arrival Case			
Leading\Following	Heavy	Large	Small
Heavy	99	133	196
Large	74	107	131
Small	74	80	98

Departure → Departure Case			
Leading\Following	Heavy	Large	Small
Heavy	60	90	120
Large	60	60	90
Small	60	60	60

Departure → Arrival Case			
Leading\Following	Heavy	Large	Small
Heavy	50	53	65
Large	50	53	65
Small	50	53	65

Table 2. Minimum Separation Times (seconds) from Sherali et al. (2010)

Due to the specific separation times used in this dissertation, which are similar to those in Sherali et al. (2010), it is necessary to ensure the separation of an aircraft between at most four consecutive aircraft.

By denoting the start time of the aircraft in the k^{th} position by $t_{[k]}$ and the separation time between aircraft at positions k_1 and k_2 by $s_{[k_1,k_2]}$, the start time of an aircraft operation k for up to 4 positions can be obtained by Equations (1) through (4).

$$t_{[1]} = r_{[1]}; \qquad (1)$$

$$t_{[2]} = \max\{r_{[2]}, t_{[1]} + s_{[1,2]}\}; \qquad (2)$$

$$t_{[3]} = \max\{r_{[3]}, t_{[1]} + s_{[1,3]}, t_{[2]} + s_{[2,3]}\}; \qquad (3)$$

$$t_{[k]} = \max\{r_{[k]}, t_{[k-1]} + s_{[k-1,k]}, t_{[k-2]} + s_{[k-2,k]}, t_{[k-3]} + s_{[k-3,k]}\}, \ \forall k = 4, \dots, n \ (4)$$

In order to illustrate the problem, sample data is given in Table 3.

j	r_j	δ_j	d_j	o_j	C_j	w_j
1	19	79	619	1	3	1
2	30	90	630	0	3	4
3	54	114	654	0	3	4
4	64	124	664	1	3	1
5	135	195	735	1	2	2
6	26	86	626	0	1	6
7	78	138	678	0	3	4
8	130	190	730	1	1	3
9	128	188	728	0	1	6
10	177	237	777	1	1	3

s_{kj}	1	2	3	4	5	6	7	8	9	10
1	30	65	65	60	60	50	65	60	50	60
2	30	30	98	30	30	74	98	30	74	30
3	30	98	30	30	30	74	98	30	74	30
4	60	65	65	30	60	50	65	60	50	60
5	90	65	65	90	40	50	65	60	50	60
6	40	196	196	40	40	40	196	40	99	40
7	30	98	98	30	30	74	30	30	74	30
8	120	65	65	120	90	50	65	50	50	60
9	40	196	196	40	40	99	196	40	40	40
10	120	65	65	120	90	50	65	60	50	50

Table 3. Sample Data

In the problem, there are two identical parallel runways (i.e. runways #1 and # 2), 10 aircraft, and each aircraft has its own ready time, target time, deadline, operation type, size class, weight/penalty and sequence dependent separation time values. A feasible schedule for the problem set is provided in Figure 4 where the notation A(B) refers to Runway number (Start time).

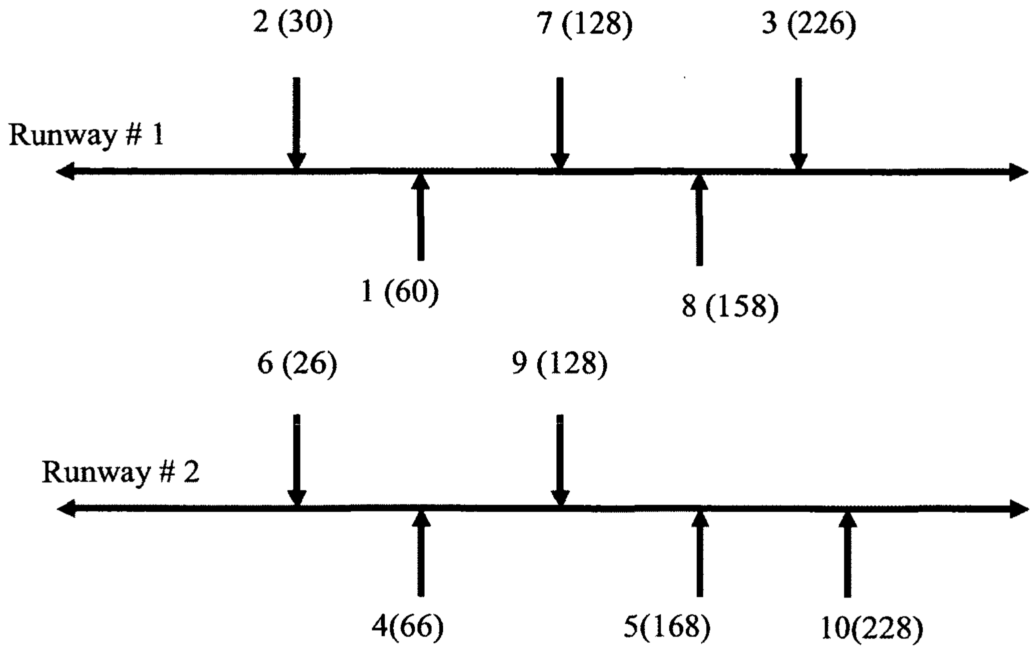


Figure 4. A Feasible Schedule

Utilizing the Equations (1)-(4), the start times for a given schedule are calculated as follows:

On the 1st runway:

$t_2 = r_2 = 30;$

$t_1 = \max\{r_1, t_1 + s_{21}\} = \max\{19, 30+30\} = 60;$

$t_7 = \max\{r_7, t_1 + s_{17}, t_2 + s_{27}\} = \max\{78, 60+65, 30+98\} = 128;$

$t_8 = \max\{r_8, t_7 + s_{78}, t_1 + s_{18}, t_2 + s_{28}\} = \max\{130, 128 + 30, 60 + 60, 30 + 30\} = 158;$

$t_3 = \max\{r_3, t_8 + s_{83}, t_7 + s_{73}, t_1 + s_{13}\} = \max\{54, 158 + 65, 128 + 98, 60 + 65\} = 226;$

On the 2nd runway:

$\mathrm{t}_6 = \mathrm{r}_6 = 26;$

$\mathrm{t}_4 = \max\{\mathrm{r}_4, \mathrm{t}_6 + \mathrm{s}_{64}\} = \max\{64, 26+40\} = 66;$

$\mathrm{t}_9 = \max\{\mathrm{r}_9, \mathrm{t}_4 + \mathrm{s}_{49}, \mathrm{t}_6 + \mathrm{s}_{69}\} = \max\{128, 66+50, 26+99\} = 128;$

$\mathrm{t}_5 = \max\{\mathrm{r}_5, \mathrm{t}_9 + \mathrm{s}_{95}, \mathrm{t}_4 + \mathrm{s}_{45}, \mathrm{t}_6 + \mathrm{s}_{65}\} = \max\{135, 128 + 40, 66 + 60, 26 + 40\} = 168;$

$\mathrm{t}_{10} = \max\{\mathrm{r}_{10}, \mathrm{t}_5 + \mathrm{s}_{5-10}, \mathrm{t}_9 + \mathrm{s}_{9-10}, \mathrm{t}_4 + \mathrm{s}_{4-10}\} = \max\{177, 168 + 60, 128 + 40, 66 + 60\} = 228$

Once the start times are determined, the piecewise tardiness and the total weighted tardiness are calculated as follows:

$\mathrm{T}_1 = \max\{0, 60\text{-}79\} = 0;$

$\mathrm{T}_2 = \max\{0, 30\text{-}90\} = 0;$

$\mathrm{T}_3 = \max\{0, 226\text{-}114\} = 112;$

$\mathrm{T}_4 = \max\{0, 66\text{-}124\} = 0;$

$\mathrm{T}_5 = \max\{0, 168\text{-}195\} = 0;$

$\mathrm{T}_6 = \max\{0, 26\text{-}86\} = 0;$

$\mathrm{T}_7 = \max\{0, 128\text{-}138\} = 0;$

$T_8 = \max\{0, 158\text{-}190\} = 0;$

$T_9 = \max\{0, 128\text{-}188\} = 0;$

$T_{10} = \max\{0, 228\text{-}237\} = 0;$

$\sum_{j=1}^{10} w_j T_j = w_1 T_1 + w_2 T_2 + \cdots + w_{10} T_{10} = 4 \times 112 = 448$

1.4 Research Scope and Objectives

The research scope of this dissertation is framed around two problem areas: the aircraft sequencing problems and the aircraft reactive scheduling problems. The scheduling and reactive scheduling environments consist of a finite set of aircraft and a finite set of runways. It is assumed that aircraft have unequal ready times, target times and deadlines. There is a ready time-to-deadline window is the scheduling time window that aircraft have to operate. Moreover, there is a target time-to-deadline window between the ready time and deadline during which weighted tardiness cost is incurred. The solution quality of the ASP is measured through minimizing the total weighted tardiness (i.e., minimizing the total weighted delay from target time). From a modeling perspective, it is advantageous to draw an analogy between the aircraft sequencing problems and specially-structured parallel machine scheduling problems. Using this metaphor, runways and aircraft are interpreted as machines and jobs, respectively, where it is desirable to minimize a pertinent cost function. This dissertation research formulates a scheduling problem from the air traffic environment as a parallel machine scheduling problem. Effective solution methodologies are proposed for ASP by taking into account the social and economic benefits across ever-increasing air traffic volume.

Air traffic systems frequently encounter disruptions such as bad weather conditions, technical failures, etc. so that the schedule cannot be executed as planned; therefore, the air traffic controllers have to update flight operations. It is assumed that a continuous reactive scheduling approach is used to update the initial schedule when a disruption occurs. Aircraft reactive scheduling problem is studied, and schedule repair and rescheduling algorithms are developed to deal with aircraft related disruptions, specifically, flight cancellations, aircraft delays and arrival of new aircraft. These diverse disruptions possibly cause failure in process quality, or continuity of the process. In order to limit the negative consequences of the disruptions, a proper course of action should be taken. However, it is not preferable to change the existing decision significantly while making a new decision. We prefer to maintain conformity to the initial decision, and be unwilling to perturb it much. For this reason, the reactive scheduling problem considers both minimizing total weighted start times and minimizing schedule instability. The objective in the ARP takes into account not only the primary measure of schedule performance but also the stability measure. The stability can be measured based on the difference between the initial and final schedule. In this dissertation, the measure of stability can be defined as differences in start times of the operations, and the runway assignment deviation. Consideration of these objectives leads us to formulate the ARP as a multi objective reactive scheduling problem.

The objectives of the research can be summarized as follows:

1. To formulate a scheduling problem from the air transportation environment as a parallel machine scheduling problem, which is mostly common in production environment.
2. To model the ASP under a mixed mode of operations where both landing and departure flows are considered simultaneously.
3. To examine the problem of scheduling aircraft arrivals and departures over multiple runways.
4. To develop effective and efficient solution methods to obtain initial schedules which satisfy the constraints of the problem in a reasonable amount of time. This is achieved by introducing greedy algorithms for the problem.
5. To improve initially constructed solutions and to find near optimal solutions using metaheuristics including Simulated Annealing and Meta-RaPS.
6. To address the *rescheduling* problem by formulating the problem as a multi-objective optimization problem where total weighted start time and instability are minimized (i.e., total weighted start time deviation and total weighted runway deviation). This is achieved using optimization models and approximate

algorithms to repair and to reschedule the disrupted schedules satisfactorily regarding multi-objectives.

The rest of this dissertation is organized as follows. In Chapter 2, related research is summarized. Solution methodologies proposed for the aircraft arrival and departure sequencing problem on multiple runways are provided in Chapter 3. A Computational study for different problem sizes is described and their results are analyzed in Chapter 4. Reactive scheduling mechanisms and solution methods for aircraft rescheduling problem are presented in Chapter 5 followed by a computational study in Chapter 6. Finally, conclusions and future research are presented in Chapter 7.

CHAPTER 2

LITERATURE REVIEW

In the light of problem defined earlier, related literature including solution approaches (i.e., exact algorithms, heuristics and metaheuristics) for aircraft sequencing problem, parallel machine scheduling problem and aircraft reactive scheduling problem are reviewed to display practical and intellectual contributions of the dissertation research.

2.1 Aircraft Sequencing Problem

In the literature, both exact and heuristic algorithms have been proposed for the ASP, with approximate algorithms recently gaining attention due to the fact that for large problems it may take a long time to reach optimal solutions. Bennell et al. (2011) provides a recent survey on ASP where a comprehensive review of operations research techniques such as dynamic programming, branch and bound, heuristics and metaheuristics that have been used to schedule aircraft landing and departures were surveyed.

2.1.1 Exact Algorithms

Early work on ASP dates backs to the early 80s where Psaraftis (1980) investigated a single machine scheduling problem for which a dynamic programming approach was developed and applied in the context of sequencing aircraft arrival operations. For computational convenience, it was assumed that groups of identical jobs are sequenced with the objective of minimizing the total processing cost.

Bianco et al. (1987, 1997) used integer programming to sequence arriving aircraft inside the Terminal Maneuvering Area (TMA). The authors showed that the ASP, combinatorial optimization problem was NP-hard. Moreover, when the ready times are zero, it was found that the problem reduced to the Asymmetric Traveling Salesman Problem (ATSP). As a solution procedure, they suggest a branch-and-bound strategy using Lagrangian lower bounding techniques, and a partitioning approach based on the characteristics of the subsequences obtained in the solution process. However, this work ignored time restrictions for aircraft operations and necessary separation times between certain nonconsecutive operations. In fact, for a triplet of aircraft, enforcing separation times between consecutive operations may not automatically satisfy the separation requirements between the first and the third operations.

Beasley et al. (2000) proposed a mixed integer linear program (MILP) model for the single and multiple runways aircraft sequencing problem, and applied a heuristic algorithm which is a version of First-Come-First-Serve (FCFS) for the aircraft landing problems (ALP).

Wen et al. (2005) developed a column generation-based exact decomposition algorithm for the ALP. They formulated the problem as a mixed integer program, and then reformulated it, using Dantzig-Wolfe decomposition, as a set partitioning problem with side constraints. Based on the set partitioning formulation, a branch-and-bound algorithm was developed to obtain the exact solution for the problem. The objective of this problem was to minimize the total (weighted) deviation from the target landing time for each plane. The decomposition algorithm was implemented in Matlab and they compared the computational performances of the same problem to some other papers in the literature; the running time in this research was substantially larger.

Due to the relative priority of landings over departures, the literature mostly focuses on the single runway aircraft landing problem. However, Gupta et al. (2009) presented a MILP for aircraft departures based on operations at Dallas-Fort Worth International Airport. The model was generic and addressed various scenarios of departure queue handling. The objective function included multiple objectives pertaining to throughput; system delay and maximum individual delay. Constraints for wake vortex separation and departure fix restrictions were considered. Multiple objectives relating to throughput, efficiency and equality were taken into account. Computational improvements to the basic MILP were provided, and tests indicated that system delay minimization has faster solution times than for throughput. For the purpose of the comparison, randomly-generated problems of varying sizes were used.

The combined arrival-departure ASP was studied over a single runway by Sherali et al. (2010). The problem was modeled as an asymmetric traveling salesman problem where the authors minimize makespan, subject to proper separation time and time-windows restrictions. The MILP model for our problem was developed in Al-Salem et al. (2012) in which they provided valid inequalities and symmetry-defeating constraints.

2.1.2 Approximate Algorithms

ASP is a key problem in air traffic control operations and it is well known that it is a NP-Hard problem since minimizing the total weighted tardiness even for a single machine with all weights being equal is NP-hard (Lawler, 1982). Within a polynomial amount of time, there is not an efficient algorithm to find global optimal or near-optimal solutions. In the early 1990s, researchers started focusing on the approximate algorithms for the ASP.

2.1.2.1 Greedy Algorithms

Dear et al. (1989, 1991) presented a Constrained Position Shifting (CPS) heuristic for the static and dynamic ALP. Termed Constrained Position Shifting (CPS) methodology is examined and its effectiveness is tested. CPS has two steps: first it searches for those sequences which maximize throughput, then the maximum throughput solution with minimum delay is selected. To show the effectiveness of the algorithm, a performance comparison between CPS and FCFS is conducted using fast-time simulation. Computational results involving up to 500 aircraft and one runway show that smaller delays are obtained under the heuristic than for a FCFS approach.

Neuman and Erzberger (1990) evaluated the performances of the FCFS approach, time advanced (TA) technique and CPS heuristic, which are used in air traffic control systems. Firstly, as an initial ordering, one of the most straightforward sequencing strategies for arrivals, FCFS is used. For an optimization step, to maximize the throughput by speeding up certain key aircraft during periods of heavy traffic, time advanced (TA) technique is presented. CPS algorithm orders the aircrafts taking advantage of different separation requirements for different aircraft classes is also used for optimization purpose. To determine the statistical characteristics of the algorithms, randomly chosen traffic samples are generated. It is found that FCFS establishes a fair order; TA method provides reasonable results for reducing the delay of each scheduled aircraft; and CPS is the most effective for heavy traffic with large groups of aircraft, and is successful at minimizing the average delay per aircraft.

Venkatakrishnan et al. (1993) were interested in using existing capacity more efficiently by improving air traffic control procedures. Their focus was air traffic delays for landing aircraft under the assumption of a single runway at Logan Airport Boston. An empirical model Landing Time Intervals (LTI) between aircraft in terms of two factors: landing runway configuration and the weight-class categories of the aircraft was presented. Furthermore, static and dynamic models were presented for ASP. In terms of the static model, they applied the work of Psaraftis (1980), but modified for the time window constraints. For the dynamic case, two dynamic models DASP-1 and DASP-2 were presented with fixed and shrinking time windows respectively.

A deterministic job shop scheduling model with sequence-dependent setup times and release dates for scheduling aircraft in TMA with multiple runways is proposed in the paper of Bianco et al (1997). Moreover, a fast dynamic local heuristic algorithm called the cheapest search heuristic (CSH) is developed. The performance of the model and the algorithm are analyzed on real data sets for the TMAs of Milan-Malpensa and Rome-Fiumicino airports. The numerical analyses indicate that on the average, delay is reduced at least 40%, and the capacity of TMA increases by about 30%when compared to to a FCFS based control policy.

Carr et al. (1998) introduced the concept of priority scheduling, which considers airline arrival preferences in sequencing and scheduling algorithms for air traffic control automation. The priority scheduling is a method of scheduling a bank of arriving aircraft according to a preferred order instead of FCFS sequence based on estimated time of arrival at the runway. To evaluate the feasibility of the method, fast-time simulation is used. The numerical analysis shows that, for certain traffic conditions, the proposed scheduling method is more successful than FCFS scheduling in reducing deviations from the preferred bank arrival order though causing little or no decrease in scheduling efficiency.

Bauerle et al. (2007) examined the queuing process of aircrafts arriving at an airport and the implications of it for the capacity of the airport. They use the general assumption that arrival times can be modeled by a Poisson process. An M/SM/1 queue (with dependent service times) is used to model a single runway. Then, they concentrated on the two runways case with a number of heuristic routing strategies such as fair coin flipping, random splitting, round robin and variants of the join-the-least-load rule. The performance of these strategies is compared with respect to the average delay they cause. It turns out that join-the-least-load strategy gives the best, and simple splitting rule gives the worst results.

In another related work, Soomer and Franx (2008) solved the single runway arrival problem in which an arrival schedule must be determined taking airlines cost into account. Having provided the mixed integer programming formulation of the model, as well as a local search heuristic in which the initial feasible solution is obtained by sorting the flights according to expected arrival times. In order to improve the initial solution, two swap and shift neighborhoods were used. The numerical experiments conducted showed that the heuristic is able to solve instances with over 100 flights in a few minutes and large cost savings for the airlines compared to a schedule that resembles current practice.

Balakrishnan and Chandran (2010) proposed that the CPS method helps to maintain fairness among aircraft operators and increases the predictability of landing times. They present dynamic programming algorithms for runway scheduling under CPS and other system constraints such as time-window restrictions and precedence constraints that had not been modeled by previous approaches. As an important objective, maximizing runway throughput or equivalently, minimizing the completion time of a sequence of aircraft is considered. They provide approaches to the multiple runway condition, although their study is on single runway.

Yu et al. (2011) proposed an algorithm called Cellular-Automata-based Optimization (CAO) for ALP with a single runway. The algorithm has two major steps. First, to efficiently obtain a good landing sequence, where the aircraft landing process is simulated using Cellular Automation (CA) model. Second, further optimization is carried out via a stochastic local search to the landing sequence obtained in previous step. They compare the method with LP-based Tree Search, a Heuristic method, Ant Colony Optimization (ACO), Scatter Search (SS) and Bionomical Algorithm (BA). It is observed that CAO is superior in terms of both the quality of the solution and the speed of the computation on most cases.

Different from many studies in the literature, Boysen and Fliedner (2011) researched the impact of the landing schedule with single runway on the workload of ground staff. In order to level the workload of ground staff, three different objectives were presented to minimize: (1) number of passengers carried by landing aircraft, (2) landing per airline, and (3) number of passengers per airline. In the study, separation time owing to turbulence effect is assumed to be equal. Earliest and latest landing times for each aircraft are not taken into consideration. For each objective function, mathematical models (including dynamic programming), complexity results and heuristic solution procedure were presented.

Recently, Hancerliogullari et al. (2013) worked on the ASP over multiple runways, under mixed mode operations with the objective of minimizing the total weighted tardiness of aircraft landings and departures simultaneously. The ASP is modeled as a parallel machine scheduling problem with unequal ready-times, target-times and deadlines. Furthermore, sequence-dependent separation times on each runway are considered to prevent the dangers associated with wake-vortex effects. The greedy algorithms, namely the Adapted Apparent Tardiness Cost with Separation and Ready Times (AATCSR), the Earliest Ready Time (ERT) and the Fast Priority Index (FPI) are proposed.

2.1.2.2 Metaheuristics

Population-based (Genetic Algorithm (GA), Memetic Algorithms, Ant Colony Optimization (ACO))

Some studies applied population based metaheuristics including Genetic Algorithm (GA), Ant Colony Optimization (ACO) and Scatter Search (SS) for the ALP.

Two approaches for solving the problem of scheduling aircraft landing times were presented in Abela et al. (1993). They looked at the arrivals problem for a set of aircraft with landing time windows. A GA is proposed to obtain an approximate solution. Moreover, for an exact solution, a branch and bound algorithm was formulated as a 0-1 mixed integer programming problem. They tested the algorithms on a randomly generated large data set, and according to the results, it is concluded that for small problem instances, the approximate algorithm performed reasonably; on the other hand, the branch and bound algorithm consumed a large amount of time to solve larger problems.

Ciesielski and Scerri (1997) investigated the applicability of genetic algorithm (GA) to the problem of real time scheduling of aircraft arrival times at airports. It was determined that computation time could be decreased and the quality of the solutions could be improved by seeding the GA from a previous population. The experiments were performed data from the one of the busiest days of the year at Sydney airport. Their preliminary results indicate that GA can produce high quality schedules in real time.

A specialized simplex lower-bounding method based on the simplex algorithm, was presented to evaluate the landing times rapidly by Ernst et al. (1999). For single and multiple runway problems, this method was used in both problem space search (PSS) heuristic and branch-and-bound method. PSS heuristic is a metaheuristic that combines a simple constructive heuristic with a GA. According to their computational studies, the heuristic manages to produce solutions in a reasonable amount of CPU time for both single and multiple runway problems. However, the solution quality is less consistent in the multiple-runway case.

Practical applications are to be found in Beasley et al. (2001) where a GA was presented to schedule aircraft arrivals at London Heathrow airport, whereas Atkin et al. (2007) proposed a TS algorithm for aircraft departures at that same airport. The algorithm that Beasley et al. (2001) proposed mainly solves the problem of deciding landing times which lie in aircraft time windows and meet the separation distance criteria, whilst optimizing proper objective. In the short term, it is expected that less delays for passengers as aircraft would land quicker; in the long term, improved scheduling would give potential for increasing the number of flights scheduled. It is concluded that the developed algorithm is able to quickly (in a matter of seconds), and effectively schedule aircraft landings with single runway.

Although many decision problems assume a static operational environment, Beasley et al (2004) considered dynamic landing times. They defined a generic decision problem for the displacement problem. This problem arises when sequences of decisions have to be made and each new decision that must be made has an explicit link back to the previous decision that was made. In order to solve the displacement problem, they adapt three solution approaches: an optimal (DALP-OPT) and two heuristics (DALP-H1, DALP-H2), given previously in the literature for the static aircraft landing problem (ALP) for multiple runway. One can expect that an optimal algorithm to always produce a solution superior to that produced by a heuristic algorithm. However, it is observed that for 2 of the 39 problems, population based (genetic algorithm) heuristic DALP-H2 produces a better solution than DALP-OPT due to time limit considerations.

Capri and Ignaccolo (2004) introduced a dynamic model for departing flights to take into account time-varying variables, and built a GA to solve the ASP on single runway. It is observed that the algorithm is proved to be quick and efficient.

Hansen (2004) examined the segment of air traffic control, termed traffic management adviser (TMA) that is concerned with the complex task of scheduling arriving aircraft to the available runways. The purpose was to investigate the utility of the genetic search approach using features of TMA problems. The reason for choosing genetic search is its applicability to solve complex problem in domains characterized by discontinuous, non-convex, or nonlinear problems. Four different genetic search methods were tested and several empirical tests were included. Method 1 used two separate genomes to represent the flight landing sequence and the runway assignment. Method 2 used a single genome definition for the complete runway assignment, sequencing and scheduling problem.

Model 3 used a randomized approach that emphasizes desirable fitness values that is similar to the standard GA approach. Method 4 incorporated genetic programming (GP) operators to define a metric, which is then used in a recursive algorithm to derive an efficient schedule. For problems of realistic size (i.e., 12 aircraft/3 runway) in real time, optimal or near-optimal assignments were achieved, and GP method yielded the best fitness values.

In order to solve the problem of position-shifting-based arrival scheduling and sequencing (ASS), Hu and Chen (2005) introduced the concept of Receding Horizon Control (RHC) into a GA. RHC is a N-step-ahead online optimization strategy. Within this framework, decisions are made by looking ahead for N steps in terms of a given cost/criterion, and only the decision for the first step is actually implemented. Then, the implementation result is checked, and a new decision is made by taking account of updated information and looking ahead for another N steps. Simulation studies which were done to check the robustness of the proposed model, indicate that RHC-based GA has much better performance than a pure GA, while requiring much less computational time.

Pinol and Beasley (2006) addressed the multiple runway static ALP where the set of aircraft that are waiting to land is known. They presented two population-based metaheuristics (Scatter Search and Bionomic Algorithm). Primarily, there were two objective functions, non-linear and linear, that are based on deviation from target times. The idea behind the non-linear objective depends on the deviation of scheduled landing time from the aircraft target time, and the difference between the scheduled landing time and the target time. The linear objective used a cost for each aircraft linearly independent on deviation of assigned landing time from target time. Computational results involving up to 500 aircraft and 5 runways were presented. It is indicated that Bionomic algorithm outperforms Scatter Search for the non-linear objective; on the other hand, for the linear objective, the reverse is observed.

Hu and Paolo (2008) designed a GA based on a binary representation rather than a permutation representation in order to solve arrival sequencing and scheduling problem in single runway. In order to construct a chromosomes for the GA, the neighboring relationship between each pair of aircraft in an optional arriving queue was used and constructed as 0-1 valued matrices. Based on the binary matrix, a highly efficient uniform crossover operator was designed. A simulation study was conducted, which showed that binary representation based GA outperformed the permutation based GA.

The effects of the airport landing sequencing algorithms on Air Traffic Control (ATC) are notes and compared in the study of Brentall and Cheng (2009). For efficiently sequencing aircraft landings, FCFS method is widely used in practice. The FCFS method is compared with alternative algorithms and its robustness under many conditions was studied by utilizing statistical methods. The minimization of makespan and total tardiness were considered. A data collection is exercised by the Eurocontrol Experimental Centre (EEC) at Stockholm Arlanda Airport to model aircraft arrivals with single runway.

Hu et al. (2009) aimed to design efficient GAs for the aircraft arrival sequencing and scheduling (ASS) in multi-runway systems. To do so, a highly efficient crossover operator- uniform crossover was attempted since it is usually effective and efficient to identify, to inherit, and to protect common genes in GAs.

Wang (2009) developed a hybrid algorithm that integrated Bee Evolutionary Genetic Algorithm (BEGA) with modified clustering method (CM), for ALP in single runway systems. In order to observe the effectiveness of the BEGA – CM, the experiments were carried and compared with GA. It is noticed that at the same time, the computational cost of the hybrid method was by far lower than GA, which concludes that BEGA – CS has a better optimization performance than GA.

Similarly, Bencheikh et al. (2009) proposed a hybrid method for ALP with multiple runway based on Job Shop Scheduling Problem (JSSP). There were three steps in this study; firstly, a mathematical programming model, whose objective was to minimize the cost deviation between the actual time of landing of all aircraft and the target time, was proposed. Secondly, based on a graphical representation, ALP was formulated as a JSSP. Finally, a hybrid resolution method, called ACOGA that combines ant colony optimization (ACO) with genetic algorithm (GA), was presented. The numerical results showed that ACOGA takes less time than GA. Moreover, in the majority of the cases, the hybrid algorithm found the optimal solution or approached a near optimal one.

Atkin et al. (2010) provided an overview, comparison and critical examination of the various ground movement models and solution methods in the literature. It was observed that there are significant differences between both the objectives and the constraints that are utilized in previous research because of differences between airports and various stakeholder aims. It was determined that in these studies, the state-of-the–art approaches use mixed integer linear programming or genetic algorithm. They suggested that since runway sequencing for both arrivals and departures and gate assignment are highly connected to the problem of airport ground movement, it would be beneficial to handle them simultaneously.

Liu (2010) developed genetic local search (GLS) to solve runway dependent aircraft landing problem. Primarily, GLS is an extension of genetic algorithm (GA) that is obtained by integrating local search into a GA context. In order to assign the landing sequence and schedule a landing time, aircraft safety regulations are met by satisfying the separation requirement. Here, the objective is to minimize the sum of squared deviations of scheduled landing time and the minimum earliest landing time of each aircraft. Numerical results, that were obtained to investigate the effectiveness of the proposed algorithm, were compared with GA, scatter search (SS) and a binomial algorithm (BA). The results support the superiority of GLS specifically when the large number of aircrafts (i.e., 12) and large number of runways (i.e., 5) are involved.

Bencheikh et al. (2011) considered the ALP on single and multiple runways as well. The mathematical formulation of the problem with a linear and nonlinear objective function, which is similar to Bencheikh et al. (2009), was presented. Then a heuristic was proposed for the single runway problem, and the heuristic was incorporated into an ant colony algorithm to solve the multiple runway case. Several priority rules were compared with the proposed heuristic. Two types of improvement heuristics were defined: parallel improving and global improving. The computational results reflect that up to 50 aircrafts and five runways, the solutions of developed algorithm coincide with the optimal solutions in 80% of the total number of instances, with an average deviation of 5% from the optimal solutions for 20% of instances that remained.

The joint sequencing of aircraft arrivals and departures over a single runway in the TMA was addressed in the study of Sherali et al. (2010) in which the objective function aimed to minimize the total processing time, subject to minimal nonconsecutive safety separation rules. A basic mathematical programming for the combined arrival –departure ASP was presented. They introduced two heuristic procedures and benchmark them against the proposed exact approaches as well as against FCFS method that is typically adopted in practice. The first proposed heuristic was an optimization-based approach (OBH) that exploits the proposed mathematical programming formulations. The second heuristic was a tour construction and improvement procedure (TCIH) that takes advantage of the problem structure. So as to test the effectiveness of the different models and heuristic procedures, several realistic flight data sets were generated by varying the number of aircraft included. Over a test-bed of 50 problem instances, the FCFS heuristic resulted in a 9% deviation from optimality. The proposed heuristic TCIH, produced near-optimal solutions within an average of 3% from optimality, and OBH was able to overcome the inherent combinatorial complexity of the problem and yielded an average 0.57% deviation from optimality.

Trajectory – based (Simulated Annealing (SA), Tabu Search (TS))

Though not as commonly applied to the ASP as evolutionary heuristics, trajectory based metaheuristics were also applied in some researches.

Atkin et al. (2007) presented a hybrid metaheuristic approach to the reordering of aircraft that consider the physical holding-point structure. Tabu search (TS) is used as a metaheuristic to search good take-off orders. Real-world constraints, for instance partially fixed schedules and fixed routes through the holding points, maintaining required separations have been considered. They presented that although at each iteration of the test system has knowledge of only a subset of the aircraft, better overall schedules can be obtained. Moreover, they have shown that if a computerized system is used in ordering the aircraft at the holding points, then it must find good take-off schedules in real time.

Atkin et al. (2008) proposed a metaheuristic based solution for determining good sequences of departure flights to help the air traffic controllers at London Heathrow Airport which has single runway for use by departures. The objective was to increase the throughput of the departure runway subject to several constraints, such as holding point constraints and minimum separation times. The search heuristics, which are the first descent, SA, steepest descent and TS, were investigated and tested. It has been concluded that both the SA and TS algorithms perform well in very short search time period.

2.2 Machine Scheduling Problem

The multiple runway ASP has similarities with the parallel machine scheduling problem. Therefore, it is worthwhile to focus on research addressing the scheduling of jobs on identical parallel machines. The literature on scheduling identical parallel machines with ready times to minimize total weighted tardiness problem is limited.

Lee and Pinedo (1997) proposed the Apparent Tardiness Cost with Setups (ATCS) heuristic to find an initial schedule for the jobs with ready-times and sequence-dependent setup times on identical parallel machines.

Mönch et al. (2005) presented two decomposition approaches to minimize the total weighted tardiness on parallel machines with unequal ready times. In the first approach, once fixed batches are formed, the batches are assigned to the machines by GA. The batches are then sequenced on each machines at the end. In the second approach, once jobs are assigned to the machines by the GA, batches on each machine are formed; again the batches are sequenced on each machine at the end. For the sequencing of batches, they considered modifications of the ATC dispatching rule. By using stochastically generated test data, it was concluded that the first approach usually outperforms the second with respect to solution quality and computation time.

Pfund et al. (2008) extended the ATCS by Lee and Pinedo (1997) by allowing non-ready jobs to be scheduled providing an opportunity to a machine to be idle for a high priority job arriving at a later time. They minimized the total weighted tardiness in the identical parallel machine problem with ready times and setups. A grid approach was developed which evaluates multiple values for the scaling parameters and chooses the best schedule among the multiple solutions.

Driessel and Mönch (2009) proposed the Variable Neighborhood Search (VNS) scheme to minimize the total weighted tardiness for an identical parallel machine scheduling problem with ready times, precedence constraints and sequence dependent setup times. It was shown that ATC greedy rule does not perform as well as VNS.

Reichelt et al. (2006) introduced multi-objective optimization problem that minimizes total weighted tardiness and makespan. They suggested a hybrid multi-objective GA. Three phase scheduling approach including a batch formulation, a batch assignment and a batch sequencing were introduced. NSGA-II metaheuristics based on GA was proposed. Then, the NSGA-II was combined with a local search algorithm to improve the results further.

Similar to Reichelt et al. (2006), Gharehgozli et al. (2009) considered a multi-objective optimization problem which minimizes the total weighted flow time and total weighted tardiness for a parallel machine scheduling problem with release and sequence-dependent setup times. A mixed integer goal programming (MIGP) was proposed. They considered the problem under the assumption of fuzzy processing times.

Rabadi et al. (2006) studied unrelated, parallel machine scheduling problems with setup times and developed a metaheuristic called Meta-RAPS (Metaheuristic for Randomized Priority Search) to solve the problem with the objective of minimizing the makespan. The effectiveness of the Meta-RaPS algorithm was also tested by comparing it to an existing heuristic called Partitioning Heuristic (PH), developed by Al- Salem (2004). For small sized problems, that problem instances ranging from six to nine jobs and two to four machines were randomly generated for which Meta- RaPS found all optimal solutions. For large problems, ranging from twenty to hundred and twenty jobs, and two to twelve machines, Meta-RaPS outperformed solutions obtained by the PH.

Helal et al. (2006) and Arnaout et al. (2009) developed TS and Ant Colony Optimization (ACO) algorithms respectively, to solve the same problem and to further improve the quality of the solutions produced in Helal et al. (2006). According to their computational results, TS outperformed the Partitioning Heuristic algorithm in most cases. Nevertheless, for small sized problems they observed less robustness. In Arnout et al. (2009), the performance of ACO results showed that the ACO performed better than Meta-RaPS, which ranked second and the TS third while the PH ranked fourth.

For the just-in-time single machine scheduling problem with setup times in which the objective was to minimize the sum of total earliness and total tardiness, Rabadi et al. (2007) developed two algorithms to find near-optimum solutions for large-sized problem instances and compared the results to a local search method that was originally developed by Rabadi et al. (2004).The first algorithm is the Shortest Adjusted processing Time (SAPT) heuristic, which consists of two phases: a schedule constructive phase and a local neighborhood search phase. Secondly, a Simulated Annealing (SA) algorithm was developed. A hybrid algorithm, SAPT-SA, which was based on both of the SAPT and SA, was also introduced. It was shown that SA provides solutions with slightly better quality while SAPT is significantly faster than SA. SAPT-SA reached to high quality solutions with low computational cost. In addition, Lee and Sherali (1994) proposed effective algorithms for unrelated machine scheduling problems having time-window and machine unavailability constraints.

2.3 Reactive Scheduling Problem

A significant amount of computational time and effort is invested in developing efficient operational schedules for airlines which are impacted by unforeseen events. The first priority for the airline is to restore the flight schedule as much as possible by minimizing the number of cancellations and total delays.

One of the extensive studies including the rescheduling concepts, reviews of the rescheduling literature, and how rescheduling affects the performance of a system is provided by Vieira et al. (2003). According to them, the rescheduling literature includes three major types of studies: methods for repairing a disrupted schedule, methods for creating a robust (immune) schedules, and on how rescheduling policies' impact on the performance of the manufacturing systems. The framework for the rescheduling research, which includes rescheduling environments, strategies, policies and methods, is presented in Figure 5. The rescheduling *environment* identifies the set of jobs that needs to be rescheduled and their nature, the *strategies* categorize whether or not schedules are completely generated or repaired, the *policies* classify when scheduling should occur, and the *methods* determine how schedules are generated and updated (Vieira et al., 2003).

Rescheduling Environments				
Static (finite set of jobs)		Dynamic (infinite set of jobs)		
Deterministic (all information given)	Stochastic (some information uncertain)	No arrival variability (cyclic production)	Arrival variability (flow shop)	Process flow variability (job shop)

Rescheduling Strategies				
Dynamic (no schedule)		Predictive-reactive (generate and update)		
Dispatching rules	Control-theoretic	Rescheduling policies		
		Periodic	Event-driven	Hybrid

Rescheduling Methods				
Schedule Generation		Schedule Repair		
Nominal schedules	Robust schedules	Right-shift scheduling	Partial rescheduling	Complete regeneration

Figure 5. The framework for the rescheduling research

Rescheduling has attracted researchers after 1990s due to it's the problem's practical significance. A rescheduling problem with release times, and machine disruptions was considered by Bean et al. (1991) who proposed a match-up scheduling method after a machine is disrupted. The match-up approach attempts to compensate for the disruption by matching-up with the pre-schedule. They pointed out that disruptions include machine breakdowns, tool unavailability, unexpected new jobs arrival, new lot release and deviation in release or target times. In addition to the match-up scheduling algorithm, they provide integer programming, and priority rule dynamic assignment heuristic. Although the cost of match-up scheduling is close to lower bounds, it is applicable only if there is enough idle time existing in the original schedule.

Church and Uzsoy (1992) considered the disruption as random job arrivals with the objective to minimize the maximum lateness on single-stage production systems involving both single and parallel machines. They indicated that continuous rescheduling approaches take rescheduling action at each time an event is recognized by the system; whereas, periodic rescheduling defines a time interval between rescheduling actions that are taken at periodic time points, also referred to as rescheduling points. Therefore, until the following point, any events occurring between rescheduling points are ignored. As a solution methodology, they define event-driven rescheduling as rescheduling action that can be taken upon the recognition of a disruption event. Moreover, worst-case error bounds for the periodic approach was developed assuming that at each scheduling point, an optimal algorithm is used to schedule available jobs.

Wu et al. (1993) presented single-machine rescheduling problem on occurrence of an unforeseen disruption serves as a model for machine breakdown. In order to satisfy conflicting goals, minimizing the makespan and the deviation from the original schedule, they used a bicriterion approach. They developed two sets of local search heuristics considering the right-shift rescheduling; the first set is pairwise swapping methods with weighted combination of the objectives, and the second set is based on GA.

Unal et al. (1997) considered single machine rescheduling problem with part-type dependent setup times and deadline for the newly arrived jobs to the existing schedule. They proposed two heuristics where the objective was to minimize either the total weighted completion time or makespan of the new jobs given that the existing jobs satisfy the deadline constraint.

Akturk and Gorgulu (1999) extended the study of Bean et al. (1991) to the concept of modified flow shop problem where they proposed rescheduling procedures for machine failures that are designed to match-up with a long term original schedule. They studied multi-objectives; minimizing the tardiness of the jobs, and the match-up point to guarantee stability of the schedule.

For unrelated parallel machine systems, Vieira et al. (2000) provided analytical models to detect the performance measures for rescheduling strategies and determine the trade-offs between performance measures. It is considered that jobs are dynamically arriving and setup times occur when production changed from one job type to another. They provided periodic, event-driven, and hybrid strategy based rescheduling heuristics. The primary performance measures determined as average flow time, machine utilization and setup frequency by the experimental results.

Alagoz and Azizoglu (2003) presented procedures for identical parallel machine rescheduling problem with an objective function of minimizing the flow time and number of disrupted jobs under machine eligibility restrictions. They proposed linear programming model for optimal solution to the problem. A polynomial time, and two branch and bound based heuristics are provided while considering right-shift strategy as well.

Hall and Potts (2004) studied a single machine rescheduling problem and considered the arrival of multiple new jobs as a disruption type. They presented a polynomial algorithm in order to minimize the total cost including original schedule cost and cost of deviation because of the disruption.

Curry and Peters (2005) considered the arrival of new jobs as a disruption type and they focused on the identical parallel machine scheduling with stepwise increasing tardiness cost objectives, non-zero machine ready times, and machine reassignment costs. They observed the tradeoff between schedule nervousness when a scheduling procedure reassign several planned operations to different machines or start times and tardiness in single and multiple period dynamic problems. They solved this problem within a simulation model with a branch and price algorithm.

Azizoglu and Alagoz (2005) considered rescheduling of identical parallel machine with machine disruptions and the schedule has to be updated to recover the effects of the disruptions. Similar to the Alagoz and Azizoglu (2003), they measured efficiency in terms of the total flow time, and as a stability measure, the number of disrupted jobs was considered where a disrupted job is one that is processed on different machines in the original and revised schedules. The optimal solution to the problem was provided and a polynomial time algorithm was presented while considering right-shift strategy that found efficient set of schedules.

Yang et al. (2006) studied identical parallel-machine problem with uncertain job arrival and sequence dependent setup time. They developed a parallel insertion algorithm which was implemented with rescheduling criterion for makespan minimization. A probabilistic model was provided to estimate the makespan when the inter-arrival time of jobs was exponentially distributed. As a solution approach, a dispatching rule, FCFS was proposed.

Lee et al. (2006) presented two machine scheduling problems in a machine related disruption environment. A polynomial algorithm was provided for optimal solution for each problem, and pseudo-polynomial algorithm was provided for the NP-hard problems. They assumed that if jobs, which are assigned to the disrupted machines, have not been processed yet, they have two options: they can be moved to other available machines by processing with additional cost and time, or can be processed by the current machine after the disruption. The objective function contains original cost function such as total weighted completion time and weighted deviation cost, transportation costs, and disruption cost.

Duenas and Petrovic (2008) proposed a predictive-reactive approach to the parallel machine scheduling problem to minimize the makespan. Material shortage and new job arrival are the two types of disruption. The starting time deviations between predictive and reactive schedules are the stability measure. Left-shifting and building new schedules were applied as rescheduling methods.

Arnaout and Rabadi (2008) considered unrelated parallel machine environment and developed repair and rescheduling algorithms, which are right shift repair, fit job repair, partial rescheduling, and complete rescheduling, for different rates of machine breakdown and delays. These rescheduling methods were evaluated based on the efficiency measure (makespan) and stability measure (number of shifted jobs).

Itayef (2009) examined multi-objective bicriteria flow shop scheduling problem with new job arrivals. A multi-objective simulated annealing algorithm, MOSA, was implemented. The procedure composed of two steps: first, given a fixed order of jobs, a conventional heuristic is proposed for job-machine assignment, and second, a simulated annealing algorithm is applied.

There are three rescheduling methods which are right shift scheduling, partial rescheduling and schedule regeneration. Keeping the defined sequences of jobs the same, right shift scheduling postpones each remaining operation by the amount of time needed to obtain a feasible schedule. Partial rescheduling algorithm reschedules only the operations affected by the disruptions. Therefore, match-up scheduling is a type of partial rescheduling. Regeneration solves the problem from the scratch for the remaining operations and reschedules them (Church and Uzsoy, 1992).

Research on parallel machine rescheduling is summarized in Table 4. In the reviewed research, parallel machine rescheduling is required due to different disruptions and events such as new job arrivals, machine breakdowns, order cancellations (Shi-jin et al., 2007), material shortage, tool unavailability, changes in due date (Jain and ElMaraghy, 1997), and changes in order priority. Subramaniam et al. (2005) provided about 20 types of disruption; however, majority of the rescheduling literature has focused on two primary types of disruptive events which are the job related (e.g. arrival of new jobs) and machine breakdowns (Bean et al.,1991, Church and Uzsoy, 1992, Vieira et al., 2000, Alagoz and Azizoglu, 2003, Curry and Peters, 2005, Azizoglu and Alagoz, 2005, Yang et al., 2006, Lee et al., 2006, Duenas and Petrovic, 2008, Arnout and Rabadi, 2008, Cheng et al., 2009) .

In addition to parallel machine rescheduling literature, there are several papers studying the application of disruption management to the airline industry. Clausen et al. (2001) discussed the developments in disruption management and their Operations Research application to telecommunications, ship-building and airline industry. Restrictive weather conditions, maintenance problems, and staff shortages are the primary disruptive events in airline operations that cause delays, or cancellations of a flight, affect not only the passengers but also the next planned activity and the crew.

Teodorovic and Guberinic (1984) considered the situation when there are one or more aircraft out of commission. Due to technical reasons, and when a stand-by aircraft is not available, disruptions arise in the planned schedule and delays occur. They used branch-and-bound technique to minimize overall passenger delay and attempted to find the least expensive aircraft routings assuming that the capacity of the aircrafts is the same.

Jarrah et al. (1993) focused on flight delays and cancellations because of aircraft shortages. The reasons for such shortages were determined as weather conditions that make flight unacceptable, mechanical problems, and delays in the schedule of incoming flights. They developed decision support system for United Airlines by providing two network models; one for delays, one for cancellations.

Luo and Yu (1997) considered the airline schedule perturbation problem, which is caused by the ground delay program, and was modeled as an integer program with the objective of minimizing maximum delay among out-flights. The schedule perturbations were classified into three groups: perturbations caused by temporary shortage of resource, perturbations caused by shortage of resource permanently, and perturbations that result from change of accessibility to airport facility. In addition to exact solutions, for finding good feasible solutions, a heuristic procedure was proposed.

Yu et al. (2003) developed a decision support system for crew scheduling and crew recovery problem, CrewSolver, based on optimization models for Continental Airlines.

They mentioned that during the day of operations, inclement weather, mechanical problems and crew unavailability prevent an airline's ability to execute its schedule as planned.

Source	Title	Objective	Disruption type	Solution Method
Bean et al. (1991)	Match-up scheduling with multiple resources, release dates and disruptions	Minimize total tardiness	Machine breakdowns, tool unavailability, unexpected new job arrival, deviation in release or target times	Match-up scheduling approach, integer programming, priority rule dynamic heuristic
Church and Uzsoy (1992)	Analysis of periodic and event driven rescheduling policies in dynamic shops	Minimize maximum lateness	Random job arrivals	Hybrid event driven rescheduling approach
Wu et al. (1993)	One-machine rescheduling heuristics with efficiency and stability as criteria	Minimize makespan, deviation from original schedule	Machine breakdown	Bicriterion approach, local search heuristics considering right-shift rescheduling
Unal et al. (1997)	Rescheduling on a single machine with part-type dependant setup times and deadlines	Minimize makespan	New job arrival	Heuristic based solution
Akturk and Gorgulu (1999)	Match-up scheduling under a machine breakdown	Minimize the tardiness, match-up point	Machine breakdown	Reactive hierarchical scheduling approach
Vieira et al. (2000)	Predicting the performance of rescheduling strategies for parallel machine systems	Minimize average flow time, maximize machine utilization	New job arrival	Periodic, event-driven, and hybrid strategy based heuristics
Alagoz and Azizoglu (2003)	Rescheduling of identical parallel machines under machine eligibility constraints	Minimize flow time, number of disrupted jobs	Machine eligibility	Linear Programming, branch and bound based heuristic considering right-shift
Azizoglu and Alagoz (2005)	Parallel-machine rescheduling with machine disruptions	Minimize flow time, number of disrupted jobs	Machine breakdown	Polynomial time algorithm
Curry and Peters (2005)	Rescheduling parallel machines with stepwise increasing tardiness and machine assignment stability objectives	Minimize tardiness cost, reassignment cost	New job arrival	Branch and price algorithm
Yang et al. (2006)	A comparative study to minimize the makespan of parallel-machine problem with job arrival in uncertainty	Minimize makespan	New job arrival	Probabilistic insertion algorithm, FIFO dispatching rule
Lee et al. (2006)	Current trends in deterministic scheduling	Minimize total weighted completion time, weighted deviation cost, deviation from completion time	Machine related	Polynomial and pseudo polynomial algorithms
Duenas and Petrovic (2008)	An approach to predictive-reactive scheduling of parallel machines subject to disruptions	Minimize makespan, starting time deviation	Material shortage, New job arrival	Predictive-reactive, left shifting methods
Arnout and Rabadi (2008)	Rescheduling of unrelated parallel machines under machine breakdowns	Minimize makespan, number of shifted jobs	Machine breakdown	Right shift repair, fit job repair, partial rescheduling, complete rescheduling
Itayef (2009)	Rescheduling a permutation flow shop problem under the arrival a new set of jobs	Minimize makespan, maximum tardiness, stability (time and sequence disruption)	New job arrival	Multi-objective metaheuristic method (MOSA) Conventional heuristic, SA algorithm

Table 4. Summary of the research on parallel machine rescheduling problem

Clausen et al. (2010) offered an overview of the network models for airline disruption management of resources, including aircraft rerouting, and crew and passenger recovery. When a disruption occurs, airlines follow a sequence to react to the problem. After resolving the infeasibilities in the aircraft schedule, they work on crewing problems. Then, ground problems are attended, and finally, the impact on passengers is evaluated. Often, because of the adverse weather conditions, scheduled flights have to be delayed, or cancelled.

As a summary of the papers that dealt specifically with airport disruptions, restrictive weather conditions, maintenance problems, aircrafts shortages, staff shortages, crew unavailability, delays in the schedule of incoming flights are the major disruptive events which can cause flight delays, flight cancellations, and runway closures. Research on airport disruption management are summarized in Table 5. If we map such disruptions to the identical parallel machine scheduling problem, flight cancellations can correspond to departure of an existing job from the original schedule. Flight delays may match up changes in ready time, target time, separation time and deadline; and because of the delays from the leading schedule, an unscheduled flight from the previous schedule may have to be scheduled with the existing schedule, or an unexpected flight operation has to be scheduled (e.g. emergency landing). This situation can correspond to arrival of new jobs in the parallel machine scheduling environment. Even though it is rare, runway closures occur in extreme cases, e.g. a snow storm, this may be mapped to machine breakdown.

In conclusion, there is a gap in the corresponding area of aircraft sequencing problem and reactive scheduling problem with unequal ready time, target time, deadline, sequence-dependent separation time. In this dissertation, we make the following contributions. First, contrary to most existing studies that treat departures as separate from landings (i.e., segregated mode), we model the ASP under a mixed mode of operations where both landing and departure flows are considered simultaneously.

Source	Title	Disruption type	Solution Method
Clausen et al. (2001)	Disruption management	Weather conditions, maintenance problems, staff shortages	
Teodorovic and Guberinic (1984)	Optimal dispatching strategy on an airline network after a schedule perturbation	Aircraft shortages (technical reasons, unavailable stand-by aircraft)	Branch and bound algorithm
Jarrah et al. (1993)	A decision support framework for airline flight cancellations	Aircraft shortages (weather conditions, mechanical problems, delays in the schedule of incoming flights)	Decision support system by providing network models
Luo and Yu (1997)	On the airline schedule perturbation problem caused by the ground delay program	Resource shortages, accessibility to airport facility	Integer programming, heuristic procedure
Arguello et al. (1997)	A GRASP for aircraft routing in response to groundings and delays	Aircraft shortages, delays (weather conditions), flight cancellations	Greedy randomized adaptive search procedure (GRASP)
Yu et al. (2003)	A new era for crew recovery Continental Airlines	Weather conditions, mechanical problems, crew unavailability	Decision support system for crew scheduling and crew recovery
Clausen et al. (2010)	Disruption management in the airline industry-concepts, models and methods	Flight cancellations, delays (weather conditions)	Overview of the network models for airline disruption (aircraft rerouting, crew and passenger recovery)

Table 5. Summary of the research on airport disruption management

In general, a landing aircraft in the air has more risk than a departing aircraft on the ground; therefore, for instance, when two aircraft (i.e., one landing, one departing) belong to the same weight class, we assign higher priority, w_j, to the aircraft that is landing. Sequence-dependent separation times for the different operation types (landing, departure), and consequently the calculation of the value of a start time, t_j, increases the complexity of the problem, which is explained in detail in Section 2. Second, besides the single runway problem, we examine the problem of scheduling aircraft arrivals and departures over multiple runways. Even though the runways are assumed to be identical, the complexity of the multiple runway problem increases when compared with the single runway system. When a single runway is considered, one merely has to determine the sequence of the aircraft allocated to a runway. On the other hand, scheduling over multiple runways is a two-step process; first, one has to determine the assignment of aircraft to runways, then the sequence of the aircraft on each runway. It is well known in the scheduling literature that parallel machine scheduling problems are in general more complex than a single machine with the same objective and constraints (Pinedo, 2008, Koulamas, 2010). Third, and to our knowledge, we are considering more aspects to the problem than any other previous work where we propose greedy algorithms (AATCSR, ERT and FPI) for the combined arrival-departure ASP with unequal ready-time, target-time, deadline, and sequence-dependent separation time. Finally, two metaheuristics (SA and Meta-RaPS) are introduced for the problem for the first time to improve initially constructed solutions by the proposed greedy algorithms.

The research that address multi-objective optimization problem in aircraft reactive scheduling problem that are liable to flight related disruptions is very limited. To fill this research gap, this dissertation updates mixed integer linear programming with normalized objective function to find optimal solutions, and proposes approximate algorithms and potential decision support system to obtain near optimal schedules efficiently. The trade-off between the objectives is evaluated; the components of the multi-objective function are the total weighted start time which represents solution quality, and the total weighted start time deviation and total weighted runway deviation which represent solution stability. Unlike the studies in the literature which focus on one disruption type at a time, in this dissertation, different types of disruptions with multiple disruptive events are considered simultaneously. Therefore, the sequential evaluation methodology is developed to treat the disruptions and revise the schedules periodically. Alternative reactive scheduling approaches for different disruptions are proposed in which the model itself dynamically select the most appropriate from several candidate solution methods with respect to (conflicting) objectives of quality and stability.

CHAPTER 3

AIRCRAFT SEQUENCING PROBLEM SOLUTION METHODOLOGY

Consider a system that schedules *n* aircrafts on *m* identical runways where each aircraft *j* has a priority weight w_j, it becomes ready to operate at ready time r_j, should start its operation (land or depart) by target time $\boldsymbol{\delta}_j$ and before deadline d_j. Furthermore, sequence-dependent separation time s_{kj} is required when an aircraft *j* operates (lands or departs) after an aircraft *k* to prevent the dangers of wake-vortex effects. If the start time of the operation for aircraft *j* is denoted by t_j, the tardiness is represented as $T_j = \max(t_j - \boldsymbol{\delta}_j, 0)$. The objective function of the aircraft sequencing problem is the minimization of the total weighted tardiness, $\sum_{j=1}^{n} w_j T_j$. Therefore, The ASP can be represented as $Pm|r_j, \boldsymbol{\delta} j, d_j, skj, time\ window|\Sigma wjTj$. Recall that, ASP has the following assumptions: any aircraft *j* can operate on at most one runway at any time; a scheduling discipline is non-preemptive (i.e., once an operation is started, it is executed until complete); runways are always available and reliable; each runway can allow at most one aircraft at any time; parameters (i.e., ready times, target times, deadlines, operation type, aircraft sizes, priority weights) and constraints of the problem are deterministic and known in advance.

In this chapter, a mixed integer linear programming (MILP) formulation of the problem, which was developed in Al-Salem et al. (2012) to find optimal solutions for the ASP, is provided. It is known that minimizing the total weighted tardiness even for a single machine with all weights being equal is NP-hard (Du & Leung, 1990), and when the jobs have different weights, the problem is strongly NP-hard (Lawler et al., 1982); hence ASP is also NP-hard combinatorial optimization problem. Consequently, it is necessary to develop efficient and effective solution approaches with reasonable computation times. Therefore, the ASP proposed in this dissertation mainly belong to a set of difficult optimization problems. When the problem size is low, it is sensible to use exact solution methods for solution quality and efficiency; however, in order to solve larger instances in reasonable computational time, it is necessary to develop efficient and effective solution approaches with reasonable computation times.

3.1 Mathematical Model

The mixed-integer 0-1 programming formulation of the problem was provided in Al-Salem et al. (2012) involving multiple runways with both immediate and general precedence decision variables. The complete MILP model for the problem is presented below:

3.1.1 Index Sets and Notation

$M=\{1,2,\dots, m\}$: A set of m identical runways.

$J=\{1,2,\dots, n\}$: A set of n aircraft (landing or departures).

r_j: ready time for aircraft j to take-off at runway end/to land (taxi time is not included), $\forall j \in J$

δ_j : target time for aircraft j to take-off at runway end/to land (taxi time is not included), $\forall j \in J$

d_j: deadline for aircraft j to take-off at runway end/to land (taxi time is not included), $\forall j \in J$

Oj: operation type of aircraft j, being a landing or a departure, $\forall j \in J$.

C_j: weight class of aircraft j, e.g., heavy, large, or small, $\forall j \in J$.

w_j: weight assigned to aircraft j based on its operation type and its weight class, $\forall j \in J$. In particular, higher priority has been assigned to landings over departures and to heavy aircraft over large and small ones. Moreover, in the test-bed $w_{j1}=w_{j2}$ if $O_{j1}=O_{j2}$ and $C_{j1}=C_{j2}$.

s_{kj}: minimum separation time required between aircraft k and j if they are respectively the leading and the following aircraft, $\forall\ k, j \in J, k \neq j$.

3.1.2 Decision Variables

t_j: the start time of aircraft j, $\forall j \in J$.

T_j: piecewise tardiness of aircraft j with respect to its target-time, $\forall j \in J$

$$z_{ij} = \begin{cases} 1, & \text{if aircraft } j \text{ is assigned to runway } i, \forall i \in M, j \in J. \\ 0, & \text{otherwise} \end{cases}$$

$$y_{kj} = \begin{cases} 1, & \text{if aircraft } k \text{ and } j \text{ are assigned to the same runway and } t_k > t_j, \forall k,j \in J, k \neq j. \\ 0, & \text{otherwise} \end{cases}$$

3.1.3 A Mixed Integer Programming Formulation

Minimize $\sum_{j \in J} w_j \; T_j$ (5)

$$\sum_{i \in M} z_{ij} = 1, \forall j \in J \qquad (6)$$

$$1 \leq \sum_{j \in J} z_{ij} \leq \left\lceil \frac{n}{m} \right\rceil, \forall i \in M \qquad (7)$$

$$r_j \leq t_j \leq d_j, \forall j \in J \qquad (8)$$

$$t_j \geq t_k + s_{kj} - (1 - y_{kj})(d_k - r_j + s_{kj}), \forall k,j \in J, k \neq j \qquad (9)$$

$$y_{kj} + y_{jk} \geq z_{ik} + z_{ij} - 1, \forall i \in M, \forall k,j \in J, k \neq j \qquad (10)$$

$$T_j \geq t_j - \delta_j, \forall j \in J \qquad (11)$$

$$0 \leq T_j \leq d_j - \delta_j, \forall j \in J \qquad (12)$$

$$z_{ij}, y_{kj} \text{ binary } \forall i \in M, \forall k,j \in J \qquad (13)$$

The objective function (5) minimizes the total weighted tardiness. Constraint (6) assigns every aircraft to exactly one of the m runways, whereas Constraint (7) introduces lower and upper bounds on the number of aircraft assigned to any runway in order to balance the loads across runways. Constraint (8) specifies allowable time-window restrictions.

Constraint (9) ensures that proper separations between any pair of aircraft are assigned to the same runway. Constraint (10) activates the sequencing variables between any pair of aircraft that are assigned to the same runway. Constraint (11) expresses aircraft tardiness, with respect to target-times. Constraint (12) enforces non-negativity restrictions and upper bounds on aircraft tardiness. Constraint (13) defines binary decision variables.

3.2 Review of Solution Methods

Exact algorithms and approximate algorithms are commonly used to find qualified solutions for optimization problems. The optimal solution for a small sized problem can be obtained by exact algorithms. On the other hand, in order to solve large sized instances, approximate algorithms are developed to yield a quick and reasonable solution to the problem although the approximate algorithms cannot guarantee optimality.

3.2.1 Exact Algorithms

Exact methods aim to find an optimal solution in a polynomial amount of time for every finite size of a combinatorial optimization problem. Mixed integer programming, branch and bound algorithms, and decomposition methods are the most common exact methods for the scheduling problems. Nevertheless, it is difficult to obtain optimal solutions for the problem in a reasonable time especially as the problem size becomes large. Consequently, it becomes necessary to develop qualified approximate solutions for the ASP.

3.2.2 Approximate Algorithms

Approximate algorithms are generally classified as constructive algorithms, local search and metaheuristics.

3.2.2.1 Constructive Algorithms

Although some exceptional implementations may need high computational times, constructive algorithms are generally the fastest approximate algorithms. Starting from scratch, the solutions are generated by adding parts of the solution in the constructive algorithms. Due to their easy implementation and low computational requirements, dispatching (priority) rules are the most widely used constructive algorithms. They are functional to start with a reasonably good schedule with regard to a single objective (Pinedo, 2008). Moreover, some priority rules generate the optimum solution for certain problems such as Shortest Processing Time (SPT) Rule is used to minimize the total completion time of a single machine scheduling problem. A number of dispatching rules such as the Shortest Processing Time (SPT), Minimum SLACK (MSLACK), and Slack per Remaining Processing Time (S/RPT) have been applied to solve total tardiness, weighted tardiness, and maximum tardiness related problems (Lee and Pinedo, 1997). Although some constructive algorithms perform very well in certain cases, there is not any specific rule that can be applied to all problems and perform satisfactorily.

3.2.2.2 Local Search Algorithms

Starting from an initial solution, that may be generated randomly or via a constructive algorithm, local search algorithms try to replace part or the whole solution with a better one iteratively. The solution or solutions with the best objective function value in those neighborhoods of solutions are called local optimum solutions. Getting easily trapped in local optima is the primary drawback of the local search algorithms. Local search algorithm with proper moves can be very helpful in exploring a neighborhood of an initial solution; however, there is not such a mechanism that searches other far neighborhoods of the solution space in which the global optimum may exist. To overcome this issue, new modern search methods have been developed with embedded meta-strategies to guide the search process.

3.2.2.3 Metaheuristic Algorithms

A metaheuristic is a heuristic procedure that is applied to difficult combinatorial optimization problems to achieve reasonable solutions. The purpose of the methodology is to explore the search space effectively by logical movements that can help avoiding local optimum solutions; allowing worsening moves is one way to do that. Another approach is that rather than just providing random initial solutions, one can generate new starting solutions for the local search in a more intelligent way. Metaheuristics iteratively obtain better solution until a stopping criterion such as total number of iterations or number of consecutive iterations without any improvement is met. Candidate solutions are evaluated; a record of the best solution obtained so far is maintained. Although the implementation of the metaheuristics is more difficult than simpler heuristics, they are superior in terms of solution robustness. Simulated annealing, tabu search, genetic algorithms, ant colony optimization, and metaheuristic for randomized priority are examples of metaheuristic algorithms which are going to be discussed.

Genetic Algorithm (GA)

It was first introduced by Holland (1975). Genetic Algorithm (GA) is a population-based metaheuristic rooted in natural selection and evolutional theory in order to find a good solution. A solution is represented by knowledge structures (i.e., chromosomes) that are composed of genes. A set of solutions contains a population that evolves over time through competition. Each member of the population (i.e., individual) is evaluated and assigned a fitness value and then the next population is formed in two steps. Firstly, individuals with high fitness values are selected for reproduction and a crossover operator generates two offspring from two parents. The crossover operator forms new fit individuals from fit parents. Then, a mutation operator changes one or more components of a selected individual. The mutation operator serves as a secondary search that guarantees that the points in the search domain are reachable. Until a stopping criterion is met, the incumbent solution is expected to improve as populations are generated. GA differs from SA and TS that at each iterative step, several schedules are generated and carried over the next step. On the other hand, in simulated annealing and tabu search, single schedule is generated and it is carried over from one iteration to another. Therefore, the neighborhood search concept of GA is dependent on a set of schedules, rather than a single schedule (Hazir et al., 2008).

Ant Colony Optimization (ACO)

Similar to GA, the ant colony optimization (ACO) algorithm is population-based metaheuristic. It is motivated by the collective behavior of ants for the continued existence of their colonies. For the food sources, ants deposit pheromone on their trail. The capability of other ants to recognize this substance enables them to find the shortest path between their nest and the food. When more ants cooperatively follow a trail, the trail becomes more attractive for being followed in the future. The ability of a single ant to place food is limited; however, the shared information helps the colony to locate efficient paths to a food source. Dorigo and Gambardella (1997) introduced this feature in solving combinatorial optimization problems. By applying a stochastic local search policy, each ant could construct a solution. A tour ends when all ants of the colony produce solutions of dissimilar quality. The knowledge gathered at the end of each tour is updated through a global pheromone updating rule. By using the information in the next tour, it is expected that the ants generate better solutions.

Metaheuristic for Randomized Priority Search (Meta-RaPS)

Meta-RaPS is a generic, high-level strategy used to modify greedy algorithms based on the insertion of a random element. Meta-RaPS integrates priority rules, randomness, and sampling in each iteration to avoid getting stuck in local optima. The general steps in applying the Meta-RaPS methodology to any combinatorial problem are as follows: study the structure of the problem to be solved, find priority rules that construct feasible solutions, modify priority rules to incorporate randomness, construct feasible solutions using priority rule and randomness, improve selected solutions, keep the best solution found by Meta-RAPS for both construction and improvement stages. Report the best solution found at the end of maximum number of iterations (Moraga, 2002).

Simulated Annealing (SA)

SA was first proposed by Kirkpatrick et al. (1983). Simulated annealing (SA) is one of the oldest and most frequently used metaheuristics to find global optimum or near-optimum to combinatorial optimization problems. SA is a robust random search technique, improvement heuristic with a clever mechanism to avoid getting trapped at local optima. The SA algorithm generates a new solution in the neighborhood of an initial (current) solution constructed by a greedy heuristic. Initial solutions can be constructed by greedy algorithms. Although improving moves are preferred, a particular structure sometimes allows moves to worse solutions to improve the search domain and avoid getting trapped at local optima. It can deal with nonlinear models and many constraints

(Hazir et al., 2008). There is a clear tradeoff between the quality of the solutions and the time required to compute them. Subsequently, Johnson et al. (1989, 1991) examined the effects of various parameters on the solution quality and computational requirements.

Tabu Search (TS)

Similar to SA, Tabu search (TS) is a local-search improvement heuristic that tries to avoid a local minimum by punishing the moves which creates cycling among previously observed solution points. These forbidden moves are called "tabu". TS algorithm keeps a list of such moves for a specific number of iterations. The cycling occurrence depends on the length of the tabu list. If the length of the tabu list is small, the process may have a high possibility of cycling (Lee et al., 1997). The major advantage of TS is the use of memory which accelerates the solution space search process (Zobolas et al., 2008). Two commonly used strategies to obtain good solutions are diversification and intensification. Diversification is used to direct the search into less visited regions of the search space, whereas intensification is used to fully explore a certain region. Glover's studies on TS have attracted numerous researchers to use the metaheuristic to solve problems from various fields due to its potential to solve difficult combinatorial optimization problems. The technique is straightforwardly applied to continuous functions by choosing a discrete encoding of the problem. Many of the applications in the literature involve integer programming problems, scheduling, etc. (Hazir et al., 2008).

Comparison of the Metaheuristics

Decision areas, problem size, available time to develop are the factors that affect the choice of which metaheuristic algorithm to use. In order to have a rational approach to perform the metaheuristics for effective ASP solutions, a review for comparative studies in scheduling problems would be useful. There are two types of metaheuristic algorithms: population based and single point search. Population-based metaheuristic methods such as GA and ACO work with a group of solutions to generate new solutions; on the other hand, single point search methods such as SA and TS start with a single solution and try to improve it via neighborhood search.

The single point search methods start with an initial feasible solution and generate more solutions iteratively. The time needed to implement SA or simple TS is relatively short compared to the population-based metaheuristics. Both TS and SA have usually been found to provide comparable quality in acceptable time. SA however is simpler to implement and requires less computer memory. SA and TS may be considered as special forms of GA with the number of individual in each generation limited to one. The GA keeps track of multiple solutions at each iteration which makes it slower. In GA, the neighborhood concept is not based on a single solution, but rather on a set of solutions. A new solution can be constructed by combining different parts from different schedules within the set. Therefore, GA fails to intensify the search to the most promising regions of a neighborhood (Lee et al., 1997).

Kim et al. (1996) reviewed several approximate algorithms including SA and GA with the objective of minimizing mean tardiness. Simulated annealing algorithm considering insertion neighborhood search algorithm, outperformed the remaining metaheuristics. Parthasarathy and Rajendran (1998) proposed SA where the initial solution was provided by a specific rule. Results were compared to the other heuristics including TS, and the SA algorithm produced the best rules. Vallada et al. (2008) provided comprehensive review of heuristic and metaheuristic approaches for the flowshop scheduling problem with the objective of minimizing total weighted tardiness. The SA algorithm outperformed all the other methods evaluated including GA and TS.

The performances of SA, TS, GA and ACO metaheuristics on the customer order scheduling problem were compared in Hazir et al. (2008). Their results indicate that the output quality of a metaheuirstic were dependent on the problem size; among the algorithms, SA performs the best. Moreover, it was mentioned that the implementation of SA was easier than the implementation of the others. Jungwattanaakit et al. (2009) investigated SA, TS, GA algorithms for minimizing the convex combination of makespan and the number of tardy jobs with unrelated parallel machines and setup times. They investigated the performance of the algorithms for the recommended SA, TS, GA parameters, and found that SA based algorithm outperformed the other algorithms.

To sum up, the parameter setting, the problem type, the platform where the study is conducted affect the outcome of each comparative study. However, it is clear that simulated annealing is easy to implement while obtaining good solutions.

3.3 Approximate Algorithms for the ASP

Exact solution methods can be used to find optimal solutions for scheduling problems. However, as minimizing the TWT with a single machine $(1||\Sigma wjTj)$ is NP-hard (Lawler, 1982), minimizing the TWT with parallel machines with ready times $(P_m|r_j|\Sigma wjTj)$ is also NP-hard. According to the complexity of hierarchy (Pinedo, 2008), the problem $Pm|rj, \delta_j, d_j, s_{kj}, time\ window|\Sigma wjTj$, which is a general case of the single machine problem, can also be considered NP-hard. Therefore, it is necessary to develop appropriate methods to reach good quality solutions in reasonable computational times.

In this section, we propose greedy algorithms and metaheuristics to solve the problem. The greedy algorithms, namely the Adapted Apparent Tardiness Cost with Separation and Ready Times (AATCSR), the Earliest Ready Time (ERT) and the Fast Priority Index (FPI) are proposed. Moreover, metaheuristics, specifically SA and Meta-RaPS are introduced for the ASP to improve the initially constructed solutions by greedy algorithms. The AATCSR is constructed by extending the ATCS rule, which is commonly used for TWT problems. The FPI rule is a modification of AATCSR, and the ERT is a version of the FCFS. SA and Meta-RaPS have been applied to different scheduling problems in order to find near-optimal solutions, and are applied for the first time to the ASP. Another reason for proposing metaheuristics is the possibility of observing a condition where the greedy algorithms cannot find a feasible schedule for a particular instance. This indicates that at least one aircraft is "unscheduled" since its start time exceeds its dedicated deadline. In such a case, the neighborhood search structures in the SA and Meta-RaPS would suffice to efficiently generate feasible solutions.

3.3.1 Adapted Apparent Tardiness Cost with Separation and Ready Times (AATCSR)

We introduce here the AATCSR composite greedy algorithm for the ASP as an extension of the ATCS rule that was introduced by Lee and Pinedo (1997). In the AATCSR, we include new terms to take into account the deadlines and ready-times. The proposed AATCSR heuristic is dynamic in a sense that after each aircraft is assigned to a runway, the remaining aircraft are prioritized according to the priority index given in Equation (14). While the deadline constraint is satisfied, the priority index is computed for each aircraft that has not been scheduled yet and the one with the highest index value is assigned to the runway on which the aircraft can start to operate the earliest.

$$\pi_j(t,k) = w_j \times exp(-\max(r_j - t, 0)) \times exp(-s_{kj}) \times exp(-\max(\delta_j - t, 0)) \times exp(-\max(d_j - t, 0)) \quad (14)$$

where $\pi_j(t,k)$ is the index for aircraft j at time t given that k is the last aircraft operated on the runway that was just freed, and t is the decision time for an assignment. The urgency of an aircraft is measured by the slack factors for the ready times max $(r_j - t, 0)$, separation times (s_{kj}), target times max $(\delta_j - t, 0)$, and deadlines max $(d_j - t, 0)$. The rationale of this rule is to exponentially increase aircraft priorities as they approach their ready times, target times, and deadlines, as well as for those whose separation times are short. Once the slack factors have negative values, they will not have an impact on the priority as exp (0) =1. In other words, target-time δ_j for job j does not have any impact on the index value when the aircraft's target time is already behind the assignment decision time (i.e.,$\delta_j < t$). In this problem, whenever an aircraft is assigned to a runway, it is assumed that the "job is completed" because the processing times are considered negligible. That is, the start times, which are the decision variables, and the completion times are the same. The pseudo code for AATCSR is given below.

t: decision time for assignment
t_j*:* the start time for aircraft j, $\forall$ j $\in$ J
$PST_i(t)$*:* potential start time for aircraft j on runway i to take-off/land at time t
$Cmax_i(t)$*:* throughput/makespan of the runway i at time t
$\pi_j(t,k)$: priority of aircraft j at time t given that k is the last aircraft operated on the runway that was just freed
M: set of runways

J: set of aircraft that are not scheduled yet

n: number of aircraft

m: number of runways

D_j: deadline for aircraft j to take-off at runway end/to land (taxi time is not included), $\forall j \in J$

```
1. Set $t=0, t_j = 0, J = \{1,2,..., n\}, M = \{1,2,..., m\}, Cmax_i(t) = 0, \forall j \in J, \forall i \in M$
2. Calculate $\pi_j(t,k)$ $\forall j \in J$ by using Equation (14)
3. while $J \neq \emptyset$
4.   while $t_j < d_j, \forall j \in J$
5.     Find $j = \{j \in J: \pi_j(t,k) = \max_{k,l \in J}\{\pi_l(t,k)\}\}$
6.     Find $i = \{i \in M: PST_i(t) = \min_{m \in M}\{PST_m(t)\}\}$ where PST is calculated according
       to Equations (1)-(4)
7.     Update $t_j = PST_i(t)$
8.     Update $Cmax_i(t) = t_j$
9.     Update $t = \min_{m \in M}\{Cmax_m(t)\}$ and remove aircraft $j$ from $J$
10.  end while
11. end while
12. Calculate and display the total weighted tardiness
```

3.3.2 Earliest Ready Time (ERT)

In the ERT rule, aircraft are assigned to the runways in increasing order of their ready times. This rule resembles the FCFS rule which is the most widely used heuristic in terminal areas for the aircraft sequencing. In addition to the total weighted tardiness minimization, ERT rule has also been applied to problems with different objective functions such as minimization of the makespan and the maximum lateness (Larson and Dessouky, 1978, Damodaran and Gallego, 2010). In the ERT, an aircraft to be scheduled with the earliest ready time is assigned to the runway on which the aircraft can start the operation the earliest. Such an approach has been used in the literature (e.g., Tsai and Lee, 1996 and Jeong and Kim, 2008) to successfully construct a good initial solution.

Moreover, the solution quality and computational times of the ERT rule can be used as a baseline to which other methods can be compared.

3.3.3 Fast Priority Index (FPI) Rule

The FPI index in equation (15) is computed for each aircraft to be scheduled when a runway is free at time *t* and the aircraft with the highest priority index value is assigned to the runway on which the aircraft can start the operation the earliest.

$$FPI_j(t,k) = w_j \times \frac{1}{\max(r_j - t, 1)} \times \frac{1}{\max(\delta_j - t, 1)} \times \frac{1}{\max(d_j - t, 1)} \times \frac{1}{s_{kj}} \quad (15)$$

where $FPI_j(t,k)$ is the index for aircraft *j* at time *t* given that *k* is the last one operated on the runway that was just freed. Different from the AATCSR, in FPI the urgency of scheduling aircraft in FPI is treated in a linear manner rather than exponential, which makes it much faster in terms of computational time. Note that the maximum of slack or 1 is used in the denominators to avoid dividing by zero.

3.3.4 Simulated Annealing (SA) Algorithm

SA is one of the well-known metaheuristic algorithms which is based on the work of Metropolis et al. (1956) that simulated the energy levels in cooling solids by producing a sequence of physical states. Kirkpatrick et al. (1983) applied this approach to solve combinatorial optimization problems for finding the global optimum, or near-optimum, of a cost function. Since it generally provides good solution and statistically guarantees finding an optimal solution, it is considered a robust metaheuristic. It has a mechanism to escape local optima by sometimes accepting a worse neighborhood move with an acceptance probability of $e^{-\Delta/T}$, where Δ is the difference in the objective function values of the current solution and candidate solution, *T* is a temperature control parameter corresponding to the temperature in the analogy of physical annealing. When T is high, most moves (better and worse) will be accepted, and as T is reduced, worse moves will more likely be rejected. Therefore, to prevent getting trapped in a local minimum, a relatively high value of T is set at the beginning of the algorithm. Specifically in our problem, instead of a constant value, the initial temperature is defined as a function of the objective function value of the current solution, which enables the initial temperature to be more flexible and to take reasonable values. While the SA goes through k drops in temperature according to the function $T_k = \alpha T_{k-1}$, at each temperature, it explores the neighborhood of the current solution.

The logic behind the neighborhood search in SA is to avoid getting stuck in local optima. *Aircraft exchange_1* and *Aircraft exchange_2* functions are used to perturb the current solution locally.

Aircraft exchange_1: When there is a randomly selected unscheduled aircraft j, it is exchanged with another randomly selected aircraft i such that $r_j < r_i$ and $d_j < d_i$.

Aircraft exchange_2: If there are no unscheduled aircraft, then randomly selected aircraft j is exchanged with randomly selected aircraft i. This neighborhood is applied to all aircraft across the runways.

The difference between both is that *Aircraft exchange_1* is applied when there is an unscheduled aircraft (i.e.,$d_j < t_j$); otherwise *Aircraft exchange_2* is executed. The performance of the SA depends on several parameters: the maximum number of inner loop iterations (i_{max}), the maximum number of iterations (t_{max}), the initial temperature coefficient (k), and the temperature cooling coefficient (α). These parameters are tuned in Section 4.2. The SA algorithms proposed in this dissertation generate new solutions in the neighborhood of the initial/current solution constructed by the proposed greedy algorithms. Therefore, the SA algorithm that integrates the AATCSR is called SA_{AATCSR}. Similarly, the implementation with the ERT is called SA_{ERT} and when integrated with FPI is called SA_{FPI}. The pseudo code for the SA is given below.

S: search area
θ *:* current solution
θ'*:* neighbor solution of the current solution
θ^**:* best solution
$f(\theta)$ *:* objective function value of the current solution (the total weighted tardiness value)

N(θ) : neighborhood of θ

M: memory set of current best solution and objective function value

i: inner loop iteration counter

i_{max}: max number of inner loop iterations

c : iteration counter

t_{max}: max number of iterations

T: temperature

k : initial temperature coefficient

α: temperature cooling coefficient

```
1. Get ERT, FPI or AATCSR solution as an initial solution $\theta$ from $S$
2. Calculate $f(\theta)$
3. Initialize memory, $Memory\ MO = \{(\theta, f(\theta))\}$
4. Set iteration counters $i = 0$, $c = 0$
5. Set initial temperature $T = k.f(\theta)$
6. while $c < t_{max}$
7.    while $i < i_{max}$
8.          Choose $\theta' \epsilon N(\theta) \subseteq S\ where$ MO= $\{(\theta, f(\theta'))\}$, do neighborhood search
            algorithm
            if there is at least one unscheduled aircraft, use Aircraft exchange_1
            else use Aircraft exchange_2
9.          Calculate $f(\theta')$
10.         if $f(\theta') - f(\theta) \leq 0$ or $rand\ [0,1] \leq e^{-\frac{\Delta\theta}{T}},\ where\ \Delta\theta = f(\theta') - f(\theta)$ then
11.         MO= $\{(\theta, f(\theta'))\}$
12.         end if
13.         $i = i+1$
14.         $c = c+1$
15.    end while
16.   Update temperature $T = \alpha.T$
17. end while
18. Output $\theta^*$ and $f(\theta^*)$
```

3.3.5 Metaheuristic for Randomized Priority Search (Meta-RaPS) Algorithm

Meta-RaPS is based on the work by DePuy and Whitehouse (2001) as the result of research conducted on the application of a modified version of Computer Method of Sequencing Operations for Assembly Lines (COMSOAL) approach that was developed by Arcus (1966). Meta-RaPS was then formally introduced by Moraga (2002) who defined it as a generic, high-level strategy used to modify greedy algorithms based on the insertion of a random element, which integrates priority rules, randomness and sampling.

Meta-RaPS is composed of two phases: a constructive phase and an improvement phase. In the constructive phase, feasible solutions are generated through randomized priority rules until a stopping criterion is met. In the improvement phase, the solutions obtained at constructive phase might be improved if they pass a specific criterion. In this dissertation, ERT, AATCSR, FPI greedy algorithms are independently used as the priority rules in initial stage. Therefore, the Meta-RaPS algorithm that integrates the ERT rule as a basis for selecting the next aircraft to schedule is called Meta-RaPS$_{ERT}$. Similarly, the implementation with the AATCSR is called Meta-RaPS$_{AATCSR}$ and when combined with the FPI is called Meta-RaPS$_{FPI}$. To prevent getting trapped in local optima, Meta-RaPS modifies the constructive algorithm such that the next aircraft to schedule does not always have to be the one with the best priority value. As an alternative, an aircraft is sometimes selected randomly from a candidate list (CL) of feasible aircraft.

Meta-RaPS involves the use of four parameters where the performance of the algorithm depends on: the number of iterations (*I*), the priority percentage (*p%*), the restriction percentage (*r%*) and the improvement percentage (*ip%*). The number of constructed feasible solutions is determined by the number of iterations. The percentage of time the aircraft with the best priority value is added to the solution is called the priority percentage. Implicitly, 100%-*p*% determines percentage of time the aircraft is randomly selected from a CL, which is a set of aircraft whose priority values are within the restriction percentage of the best priority value. The improvement percentage is used to decide if the solution created at the constructive stage is worthy of being improved in the improvement phase. Therefore, only solutions with promising values are improved by using neighborhood search algorithms, which are similar to those used in the SA algorithm. The rationale of selectively improving solutions is not to waste computational time on very inferior solutions, but rather improve solutions that have the potential to reach a global optimum. The pseudo code for Meta-RaPS$_{AATCSR}$ is given below.

θ: solution from constructive phase
$f(\theta)$: objective function value of the constructive phase
I: number of iterations
p%: the priority percentage
r%: the restriction percentage
ip%: the improvement percentage

```
1. Set $t=0, t_j = 0, J = \{1,2,..., n\}, M = \{1,2,..., m\}, Cmax_i(t) = 0, \forall j \in J, \forall i \in M$
2. Calculate $\pi_j(t,k)$ $\forall j \in J$ by using Equation (14)
3. while $J \neq \emptyset$
4.   while $t_j < d_j, \forall j \in J$
5.     Find $j = \{j \in J: \pi_j(t,k) = \max_{k,l \in J}\{\pi_l(t,k)\}\}$
6.     Find $i = \{i \in M: PST_i(t) = \min_{m \in M}\{PST_m(t)\}\}$ where PST is calculated according
       to Equations (1)-(4)
7.     $P = RND(0,1)$
8.     if $P \leq p\%$ then
9.     Set aircraft to schedule index, $s = j$
10.    else
11.    Randomly choose aircraft $l$ from Candidate List = $\{l: l \in J \mid \pi_l(t,k) \geq \pi_j(t,k).r\%\}$
12.    Set aircraft to schedule index, $s = l$ and Candidate List = $\emptyset$
13.    end if
14.    Update $t_j = PST_i(t)$
15.    Update $Cmax_i(t) = t_j$
16.    Update $t = \min_{m \in M}\{Cmax_m(t)\}\}$ and remove aircraft $j$ from $J$
17.  end while
18. end while
```

```
19. Calculate $f(\theta)$
20. if $f(\theta) \leq f_{best} + (f_{worst} - f_{best}) .\ ip\%$ then
20. Find $\theta' \epsilon N(\theta)$ do neighborhood search algorithm
    if there is at least one unscheduled aircraft, use Aircraft exchange_1
    else use Aircraft exchange_2
21. Calculate $f(\theta')$
21. end if
22. Update Memory, Report $f_{best}$
```

In this chapter, several solution methodologies and heuristic procedures are proposed for the Aircraft Sequencing Problem. Three greedy algorithms, namely the Adapted Apparent Tardiness Cost with Separation and Ready Times (AATCSR), the Earliest Ready Time (ERT) and the Fast Priority Index (FPI) were developed to construct initial solutions. In addition to greedy algorithms, metaheuristics including Simulated Annealing (SA) and the Metaheuristic for Randomized Priority Search (Meta-RaPS) were introduced to improve solutions initially constructed by the proposed greedy algorithms. The performance (solution quality and computational time) of the various algorithms is compared to the optimal solutions and to each other in Chapter 4.

CHAPTER 4

COMPUTATIONAL STUDY FOR SOLUTION METHODOLOGIES

Computational study for the ASP heuristic algorithms aims to determine the performance of the proposed algorithms for a wide range of problem sizes. The effectiveness and efficiency of the ERT, AATCSR, FPI, SA, Meta-RaPS algorithms are evaluated for problems with a number of aircraft $n = 15, 20, 25$ and number of runways $m = 2, 3, 4, 5$; 15-aircraft instances with 2,3,4 runways, 20 and 25 aircraft instances with 2,3,4,5 runways. For each combination of n and m, 5 instances were generated totaling 55 problem instances that are publicly available at http://ahmed.ghoniem.info/download/MASP-SET.txt. The proposed algorithms were implemented in C and run on an Intel Core 2 Duo 2.10 GHz CPU with 4.00 GB of RAM laptop. According to Ghoniem and Farhadi (2012), the optimal solutions were obtained by mixed-integer formulations which were coded with AMPL and solved using CPLEX 12.4 on Intel Core i7-2600 CPU with 3.40 GHz and 12 GB RAM laptop. For the optimal solutions, they imposed a time limit of 1 CPU hour on the solver.

4.1 Data Generation

In the data used, each aircraft is characterized by its operational type (i.e, arrival or departure), weight-class (i.e, heavy, large, or small), priority (aircraft tardiness penalty), ready time, target time, deadline, and separation times. Data were generated as in Ghoniem and Farhadi (2012) as follows:

1. Aircraft operation types were randomly generated as 0 or 1 to represent an arrival and departure respectively.

2. Aircraft weight classes were randomly generated as 1, 2, 3 to represent heavy, medium or light aircraft respectively.

3. The aircraft tardiness penalty (priority) w_j varies between 1 and 6 and was introduced as a function of the aircraft weight class and its operation type, where the least weight of

1 was assigned to small departures and the greatest weight of 6 was given to heavy arrivals.

4. The ready-times r_j were randomly generated using a discrete uniform distribution over the interval $(0, \gamma\frac{n}{m})$, where γ is a parameter that was randomly selected between 30 and 90.

5. Every aircraft was prescribed a time-window of 600 seconds. Therefore, deadlines d_j were calculated by $\boldsymbol{r}_j + 600$.

6. Target times $\boldsymbol{\delta}_j$ were calculated by $\boldsymbol{r}_j + 20$.

7. As presented in Sherali et al. (2010) and shown in Table 1, the minimum separation times s_{kj} are given and range between 30 and 200 seconds depending on aircraft type (light, medium or heavy), and the type of operation (landing vs. departure) that the actual values are enforced by aviation authorities.

4.2 SA Parameter Setting

The solution quality and the speed of the metaheuristics depend on the values of their parameters. One of the Design of Experiments (DoE) methods, Taguchi design, was used in order to tune the parameters of the SA algorithm. A subset of problems from the entire problem set of instances up to 25 aircraft were selected randomly. For each of the four design factors (i.e., parameters of the SA algorithm) two possible levels were defined, and the experiments were conducted to obtain appropriate parameter values while considering the qualities of the average relative error and average CPU times. For the SA algorithm, two levels were determined as follows: maximum number of iterations (t_{max}= 1000, 2000), maximum number of inner loop iterations (i_{max}=10, 20), initial temperature coefficient (k = 1, 10) and temperature cooling coefficient (α = 0.8, 0.95). Considering not only the main effects (e.g.,t_{max}, i_{max}) but also the interaction effects (e.g.,$t_{max} \times i_{max}$), we solved five random problem instances 20 times using SA starting from initial solutions constructed by the AATCSR, ERT and FPI.

After completing regression analysis of the average relative error values and average CPU, the R^2 values were 0.92 and 0.88 respectively for the AATCSR case; R^2= 0.92 and 0.88 respectively for the FPI case; and R^2= 0.91 and 0.88 respectively for the ERT case. Accordingly, the following parameter levels were selected for best performance: maximum number of iterations t_{max}= 1000, maximum number of inner loop iterations i_{max}=10, initial temperature coefficient k = 1 with the initial temperature being the corresponding weighted tardiness of the initial solution, and temperature cooling coefficient $\alpha = 0.8$ for SA_{ERT}, SA_{AATCSR} and SA_{FPI}.

4.3 Meta-RaPS Parameter Setting

A similar DoE approach was used to set the appropriate parameter values for Meta-RaPS for its four parameters. Two-level factorial design was used and the levels are identified in Table 6. For each experiment, we selected 5 problem instances randomly from the entire problem set, and we solved them 20 times by Meta-RaPS using the AATCSR, ERT and FPI as priority rules in its construction stage.

Level	Coded	Parameter			
		I	*r%*	*p%*	*ip%*
Low	-1	2000	0.25	0.75	0.75
High	+1	5000	0.5	0.9	0.5

Table 6. Coded values of factor levels

Regression analysis of the average relative error values and average CPU time values revealed R^2= 0.97 and 0.98 respectively for the AATCSR case; R^2= 0.97 and 0.97 respectively for the FPI case; and R^2= 0.96 and 0.97 respectively for the ERT case. Accordingly, the following parameter levels were selected for best solution quality: number of iterations *I*=5000, the priority percentage *p%*=0.75, the restriction percentage *r%*=0.25 and the improvement percentage *ip%*=0.9. The same parameters were used for Meta-$RaPS_{ERT}$, Meta-$RaPS_{AATCSR}$ and Meta-$RaPS_{FPI}$.

4.4 Effectiveness of the Greedy Algorithms (AATCSR, ERT, FPI)

The performance of the ERT, AATCSR and FPI algorithms are evaluated in terms of average relative error, number of times each heuristic solution reaches optimal solutions and CPU times. The relative error (i.e., deviation from optimal) is calculated via Equation (16) for each test problem.

$$Relative\ Error = \frac{TWT_{ALG} - TWT_{Optimal}}{TWT_{Optimal}} \quad (16)$$

where TWT_{ALG} is the objective function value (i.e., total weighted tardiness) of the proposed greedy algorithm and $TWT_{Optimal}$ is the objective function value of the optimal solution for a test problem.

Optimal solutions were obtained using the MILP formulation presented in Section 4. Average relative errors for the greedy algorithms and average CPU times are obtained by averaging the values for the five instances for each aircraft-runway combination. The results in Table 7 show that the performances of the three greedy algorithms are similar in terms of average CPU time. The ERT performs slightly better than the AATCSR and the FPI in terms of the frequency of reaching optimal solutions. However, the average relative error metric indicates that it does not perform as well as the AATCSR and the FPI. Finally, it is worth noting that CPU times of the proposed solution methods are considerably shorter than the optimal solutions.

The relative error data and CPU time performance for each combination are not normally distributed according to Anderson-Darling normality test as stated in Figure 6. Therefore, the performances of the algorithms are analyzed statistically by nonparametric test, Kruskal-Wallis.

n	m	Average Relative Error from Optimal			# of optimal solutions			Average CPU (s)			
		AATCSR	ERT	FPI	AATCSR	ERT	FPI	Optimal	AATCSR	ERT	FPI
15	2	0.728	0.888	0.844	0	1	0	25.37	0.000	0.000	0.000
	3	0.873	1.172	1.171	0	0	0	74.222	0.000	0.000	0.000
	4	0.446	0.563	0.563	0	3	0	1684.228	0.000	0.000	0.000
20	2	0.726	0.834	0.834	0	0	0	4.456	0.001	0.000	0.000
	3	1.304	1.507	1.505	0	0	0	583.98	0.000	0.000	0.000
	4	1.708	1.708	1.708	0	0	0	2458.94	0.000	0.000	0.000
	5	0.819	0.825	0.819	0	0	0	2596.48	0.000	0.000	0.000
25	2	1.183	1.205	1.205	1	1	1	1.28	0.000	0.000	0.000
	3	0.577	0.947	0.940	0	0	0	723.514	0.000	0.000	0.000
	4	1.297	1.370	1.297	0	0	0	495.628	0.000	0.000	0.000
	5	0.540	0.540	0.540	0	0	0	1308.014	0.000	0.000	0.000

Table 7. Results of the Greedy Algorithms

The statistical software program, Minitab 15.1 is used for analysis.

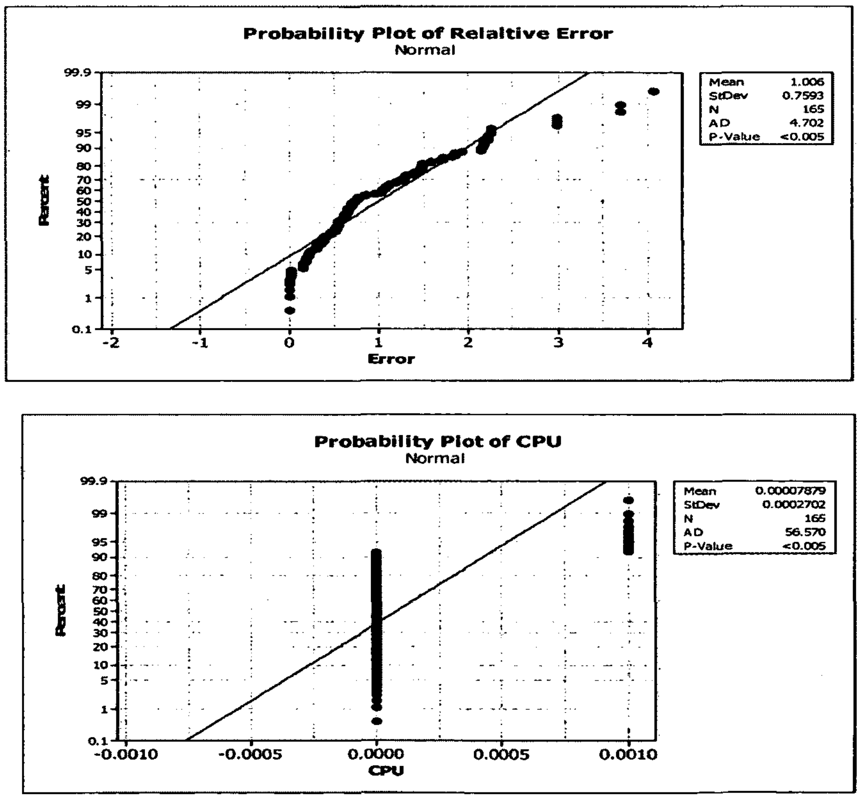


Figure 6. Normality Test for the Greedy Algorithms

Table 8 shows that AATCSR has performed better since the median of AATCSR relative errors is less than the median values of the ERT and FPI. On the other hand, AATCSR, ERT and FPI have similar CPU time performance.

```
Kruskal-Wallis Test on Relative Error

Method    N  Median  Ave Rank   Z
AATCSR   55  0.7031    76.9    -1.15
ERT      55  0.7659    86.4     0.64
FPI      55  0.7613    85.7     0.51
Overall  165            83.0

H = 1.33  DF = 2  P = 0.514
H = 1.33  DF = 2  P = 0.514  (adjusted for ties)
```

```
Kruskal-Wallis Test on CPU

Method     N     Median    Ave Rank    Z
AATCSR   55  0.000000000     93.0  1.90
ERT      55  0.000000000    78.0  -0.95
FPI      55  0.000000000    78.0  -0.95
Overall  165                 83.0

H = 3.61  DF = 2  P = 0.164
H = 16.60  DF = 2  P = 0.000  (adjusted for ties)
```

Table 8. Comparison of effectiveness of the Greedy Algorithms

4.5 Effectiveness of the SA and Meta-RaPS Algorithms

Three versions of the SA and the Meta-RaPS were applied to the problem with the main difference being the way that the initial solutions are generated using ERT, AATCSR, or FPI algorithms. The performances of the SA and the Meta-RaPS are compared in terms of the relative error, the number of times each heuristic solution reached optimal solutions and the CPU times. Equation (16) is used to calculate the relative error for each test problem with TWT_{ALG} being the objective function value of the proposed metaheuristic.

Hancerliogullari et al. (2013) introduced SA and the Metaheuristic for Randomized Priority Search (Meta-RaPS) to the ASP to improve the initially constructed solutions by greedy algorithms. The algorithms' solutions are compared to optimal solutions and their performances are evaluated in terms of solution quality and CPU time. According to Table 9, the average relative error and the number of optimal solutions reached by the metaheuristics indicate that SA_{AATCSR} performs better than SA_{FPI} and SA_{ERT}. Also, the average CPU times of the algorithms are less than one second on average. The performance measures reflect that solving the problem with the SA is advantageous since in the majority of time the SA could find optimal solutions.

n	M	Average Relative Error from Optimal				# of optimal solutions				Average CPU (s)			
		SA_{AATCSR}	SA_{ERT}	SA_{FPI}	SA_{Random}	SA_{AATCSR}	SA_{ERT}	SA_{FPI}	SA_{Random}	SA_{AATCSR}	SA_{ERT}	SA_{FPI}	SA_{Random}
15	2	0	0.002	0	0.004	5	3	4	1	0.417	0.412	0.406	0.422
	3	0.024	0.055	0.05	0.063	3	2	3	0	0.438	0.371	0.445	0.441
	4	0	0.004	0	0.005	5	3	3	1	0.502	0.371	0.381	0.384
20	2	0.018	0.037	0.02	0.045	2	1	2	0	0.583	0.549	0.619	0.556
	3	0.006	0.045	0.02	0.053	4	0	1	0	0.62	0.486	0.523	0.595
	4	0.004	0.044	0.01	0.050	4	0	3	0	0.58	0.561	0.431	0,571
	5	0.002	0.013	0.01	0.009	4	2	3	1	0.43	0.484	0.438	0.49
25	2	0	0.009	0.01	0.009	4	2	3	1	0.756	0.678	0.828	0.692
	3	0.005	0.151	0.03	0.082	2	0	1	0	0.762	0.699	0.725	0.701
	4	0.018	0.092	0.05	0.095	3	0	2	0	0.538	0.577	0.585	0.581
	5	0.021	0.089	0.08	0.102	2	0	2	0	0.513	0.53	0.513	0.542

Table 9. Results of the SA Algorithm

Moreover, the performance of the SA algorithm starting with a randomly generated initial solution (SA_{Random}) is evaluated. The corresponding average relative error, the number of optimal solutions reached by the SA_{Random}, and average CPU times are also summarized in Table 9. The results show that SA_{AATCSR} still performs the best among the SA_{ERT}, SA_{FPI}, SA_{Random} in terms of solution quality. On the other hand, the CPU time performance of the algorithms are close to each other and considerably low. It is concluded that generating new starting solutions for local search improvement heuristic, SA, using greedy algorithms (AATCSR, ERT, FPI) is more intelligent way than just generating random initial solutions. There is an advantage in starting from a better initial solution than random solution in problems where good solutions cannot be easily obtained through a small number of elementary transformations of a random solution (Pirlot, 1996).

Since the relative error data for each combination does not follow a normal distribution according to Anderson-Darling normality test in Figure 7, the statistical significance of performance between algorithms is analyzed using non-parametric Kruskal-Wallis test.

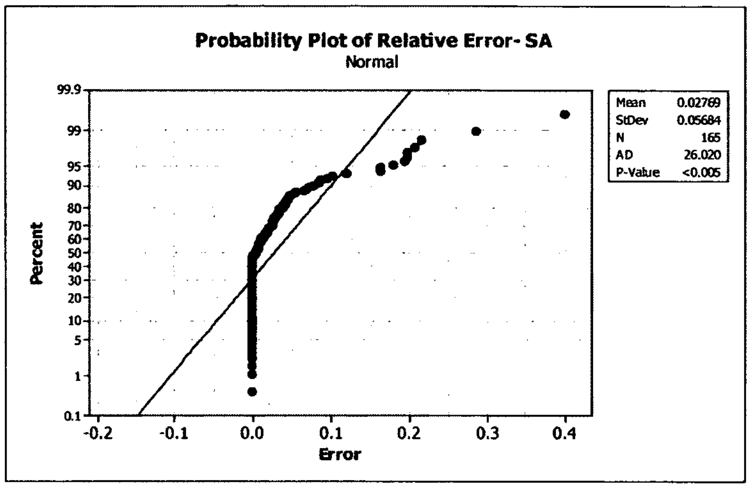


Figure 7. Normality Test for the SA Algorithm: Relative Error

Table 10 shows that there is a statistical difference between the population mean values of SA_{AATCSR}, SA_{ERT} and SA_{FPI} in terms of relative error (p=0.000<α=0.05). It can be inferred that SA_{AATCSR} has the best performance because median of relative errors is less than median value of SA_{ERT} and SA_{FPI}; moreover, SA_{FPI} performs better than SA_{ERT}. On the other hand, the CPU time performances of the algorithms are almost indifferent.

Another important measure is Levene's test when the data is continuous but not necessarily normally distributed. In order to measure the robustness of the proposed algorithms, test of equal variance is used. Table 11 shows that there exists significant difference between variances for the relative errors (p=0.002<α=0.05). SA_{AATCSR} algorithm has lower confidence intervals for relative error which indicates that it is more robust for the problem compared to SA_{ERT} and SA_{FPI}.

```
Kruskal-Wallis Test on Relative Error

Method       N    Median     Ave Rank    Z
AATCSR-SA   55  0.000000000    60.4   -4.30
ERT-SA      55  0.024637681   106.5    4.47
FPI-SA      55  0.004854369    82.1   -0.17
Overall    165                 83.0

H = 25.63  DF = 2  P = 0.000
H = 28.53  DF = 2  P = 0.000  (adjusted for ties)
```

```
Kruskal-Wallis Test on CPU

Method      N     Median     Ave Rank    Z
AATCSR     55  0.000000000    93.0     1.90
ERT        55  0.000000000    78.0    -0.95
FPI        55  0.000000000    78.0    -0.95
Overall   165                 83.0

H = 3.61  DF = 2  P = 0.164
H = 16.60  DF = 2  P = 0.000  (adjusted for ties)
```

Table 10. Comparison of effectiveness of the SA Algorithm with different initial solutions

Figure 8 also confirms that SA_{AATCSR} has the best performance in terms of relative error.

```
Test for Equal Variances: Relative Error
95% Bonferroni confidence intervals for standard deviations

Method       N    Lower       StDev      Upper
AATCSR-SA   55   0.0169324   0.0208642  0.026979
ERT-SA      55   0.0632537   0.0779418  0.100786
FPI-SA      55   0.0402559   0.0496037  0.064142

Levene's Test (Any Continuous Distribution)
Test statistic = 6.29, p-value = 0.002
```

Table 11. Test for Equal Variances: Relative Error versus Method

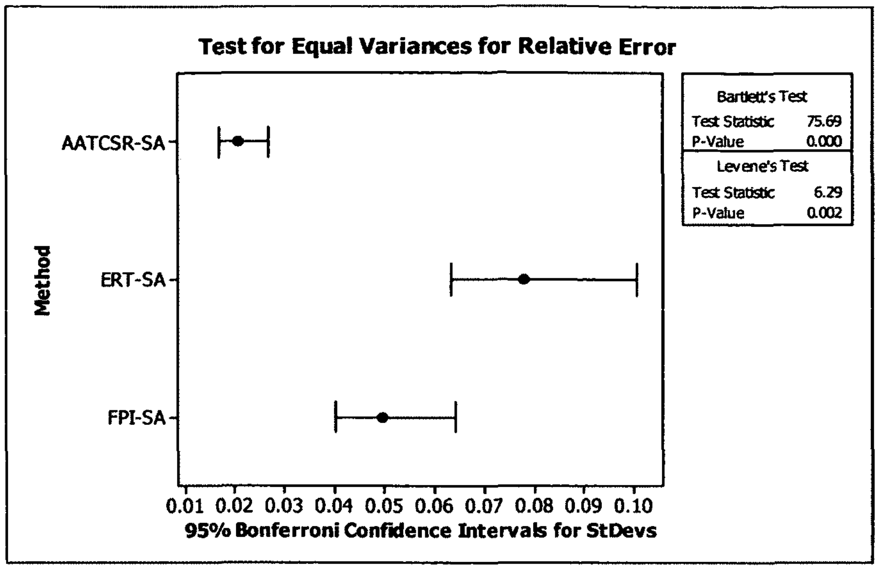


Figure 8. Test for Equal Variances: Relative Error versus Method

According to Table 12, the solution quality of Meta-RaPS$_{AATCSR}$ outperforms that of Meta-RaPS$_{ERT}$ and Meta-RaPS$_{FPI}$ in terms of the relative error and the number of optimal solutions reached by the Meta-RaPS. However, the drawback of Meta-RaPS$_{AATCSR}$ is that it requires longer CPU time.

n	M	Average Relative Error from optimal			# of optimal solutions			Average CPU (s)		
		Meta-RaPS$_{AATCSR}$	Meta-RaPS$_{ERT}$	Meta-RaPS$_{FPI}$	Meta-RaPS$_{AATCSR}$	Meta-RaPS$_{ERT}$	Meta-RaPS$_{FPI}$	Meta-RaPS$_{AATCSR}$	Meta-RaPS$_{ERT}$	Meta-RaPS$_{FPI}$
15	2	0.001	0.261	0.160	4	0	2	7.394	1.327	0.171
	3	0.047	0.225	0.187	3	0	3	8.400	1.399	0.191
	4	0.112	0.216	0.184	0	0	0	8.873	1.399	1.702
20	2	0.021	0.464	0.264	1	0	1	12.841	2.149	0.271
	3	0.029	0.644	0.535	0	0	0	16.261	2.559	0.230
	4	0.853	1.041	0.969	0	0	0	17.202	2.749	2.816
	5	0.456	0.482	0.456	1	0	0	13.760	3.413	2.954
25	2	0.008	0.418	0.276	3	2	3	23.068	3.151	0.374
	3	0.009	0.411	0.262	1	0	1	23.265	3.522	0.280
	4	0.431	0.848	0.565	1	0	0	20.690	4.133	4.116
	5	0.234	0.311	0.274	0	0	0	21.065	3.891	4.223

Table 12. Results of the Meta-RaPS Algorithm

Similar to SA, since the relative error data for each combination in Meta-RaPS does not follow a normal distribution according to Anderson-Darling normality test in Figure 9, the statistical significance of performance between algorithms are analyzed using non-parametric Kruskal-Wallis test Table 13 shows that there is a statistical difference between the population mean values of Meta-RaPS $_{AATCSR}$, Meta-RaPS $_{ERT}$ and Meta-RaPS $_{FPI}$ in terms of relative error (p= 0.000<α=0.05). Meta-RaPS $_{AATCSR}$ has the best performance because median of relative errors is less than median value of Meta-RaPS $_{ERT}$ and Meta-RaPS $_{FPI}$; moreover, Meta-RaPS $_{FPI}$ performs better than Meta-RaPS $_{ERT}$.

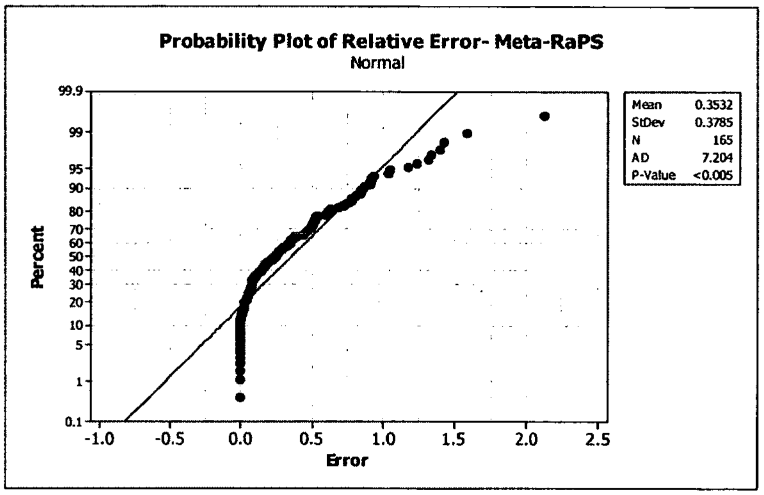


Figure 9. Normality Test for the Meta-RaPS Algorithm: Relative Error

Furthermore, it can be determined that there is a statistical difference between the CPU time values of Meta-RaPS algorithms that use different dispatching algorithms as priority rules in the initial stage. Meta-RaPS $_{AATCSR}$ has worse CPU time performance compared to Meta-RaPS $_{ERT}$ and Meta-RaPS $_{FPI}$.

```
Kruskal-Wallis Test on Relative Error

Method               N  Median  Ave Rank    Z
AATCSR-METARAPS   55 0.03884     54.1    -5.49
ERT-METARAPS       55 0.36946    104.0    3.99
FPI-METARAPS       55 0.26413     90.9    1.50
Overall            165              83.0

H = 32.16  DF = 2  P = 0.000
H = 32.21  DF = 2  P = 0.000  (adjusted for ties)
```

```
Kruskal-Wallis Test on CPU

Method                 N  Median   Ave Rank     Z
AATCSR-METARAPS   55  15.0010    138.0     10.46
ERT-METARAPS       55   2.7300    67.1        3.02
FPI-METARAPS       55   0.3910    43.9       -7.44
Overall       165         83.0

H = 115.85  DF = 2  P = 0.000
H = 117.22  DF = 2  P = 0.000  (adjusted for ties)
```

Table 13. Comparison of effectiveness of the Meta-RaPS Algorithm with different initial solutions

The performances of both SA and Meta-RaPS are compared when they use the same greedy algorithm in their initial stages. As expected, the general trend of the results indicates that SA and Meta-RaPS that use greedy algorithms as initial solutions perform better than the greedy algorithms alone (i.e., SA_{AATCSR} is superior to AATCSR, and $Meta\text{-}RaPS_{ERT}$ is superior to ERT). When Tables 14 and 15 are compared, it is observed that the solution quality of SA is usually better than that of the Meta-RaPS when the same greedy algorithms are considered for initial solutions (i.e., SA_{AATCSR} is superior to Meta-$RaPS_{AATCSR}$, SA_{FPI} is superior to $Meta\text{-}RaPS_{FPI}$).

The statistical significance of performance between algorithms is analyzed using non-parametric Kruskal-Wallis Tables 14, 15 and 16 show that there is a statistical difference between the population mean values of Meta-RaPS $_{AATCSR}$, SA_{AATCSR} and AATCSR; Meta-RaPS $_{ERT}$, SA_{ERT} and ERT; and Meta-RaPS $_{FPI}$, SA_{FPI} and FPI in terms of relative error ($p = 0.000 < \alpha = 0.05$). SA_{AATCSR} has the best performance because median of relative errors is less than median value of Meta-RaPS $_{AATCSR}$ and AATCSR. Similarly, SA_{ERT} performs best among Meta-$RaPS_{ERT}$ and ERT because median of relative errors is less than median value of Meta-RaPS $_{ERT}$ and ERT; and SA_{FPI} has the best performance because median of relative errors is less than median value of Meta-RaPS $_{FPI}$ and FPI (0.004<0.761).

```
Kruskal-Wallis Test on Relative Error

Method             N   Median       Ave Rank    Z
AATCSR-METARAPS    55  0.038844622   79.1    -0.74
AATCSR-SA          55  0.000000000   42.3    -7.73
AATCSR             55  0.703125000  127.6     8.47
Overall           165                83.0

H = 88.07  DF = 2  P = 0.000
H = 92.07  DF = 2  P = 0.000  (adjusted for ties)
```

Table 14. Comparison of effectiveness of the AATCSR and metaheuristic algorithms

```
Kruskal-Wallis Test on Relative Error

Method          N   Median   Ave Rank    Z
ERT             55  0.76587   123.2    7.65
ERT-METARAPS    55  0.36946    92.6    1.83
ERT-SA          55  0.02464    33.1   -9.48
Overall        165             83.0

H = 101.20  DF = 2  P = 0.000
H = 101.26  DF = 2  P = 0.000  (adjusted for ties)
```

Table 15. Comparison of effectiveness of the ERT and metaheuristic algorithms

```
Kruskal-Wallis Test on Relative Error

Method          N    Median    Ave Rank    Z
FPI             55   0.761347   126.8     8.33
FPI-METARAPS    55   0.264126    89.1     1.15
FPI-SA          55   0.004854    33.1    -9.48
Overall        165               83.0

H = 106.99  DF = 2  P = 0.000
H = 107.57  DF = 2  P = 0.000  (adjusted for ties)
```

Table 16. Comparison of effectiveness of the FPI and metaheuristic algorithms

According to the Levene's test, Figures 10, 11, and 12 show that there exists significant difference between variances for the relative errors ($p=0.002<\alpha=0.05$).

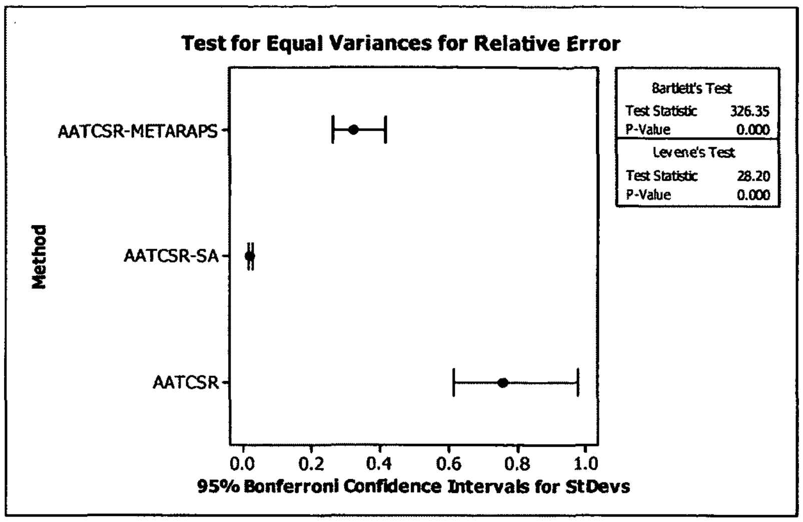


Figure 10. Test for Equal Variances: Relative Error versus Method-AATCSR

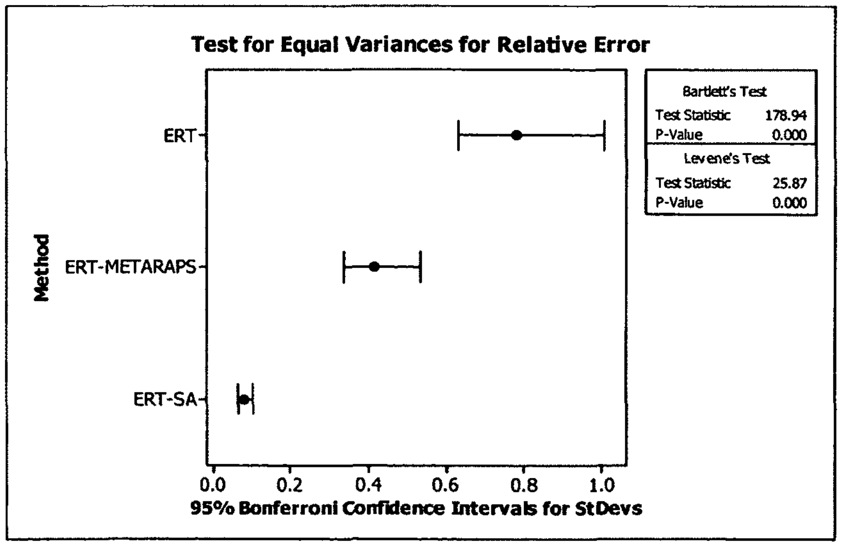


Figure 11. Test for Equal Variances: Relative Error versus Method-ERT

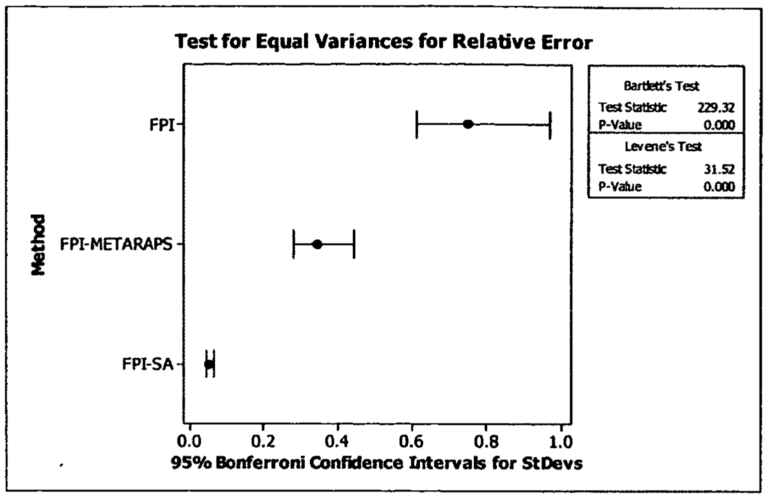


Figure 12. Test for Equal Variances: Relative Error versus Method-FPI

To sum up, Figures 13 and 14 illustrate that the mean plots of all problem instances at 95% confidence level. Figure 13 clearly shows that there are statistically significant differences between the relative error values of the greedy algorithms and the metaheuristic algorithms since there is no overlap between them (e.g., AATCSR does not overlap with SA_{AATCSR} and Meta-$RaPS_{AATCSR}$).

Figure 14Figure 14 illustrates that the CPU times of the proposed algorithms are quite similar, and they are considerably low compared to the optimal solution.

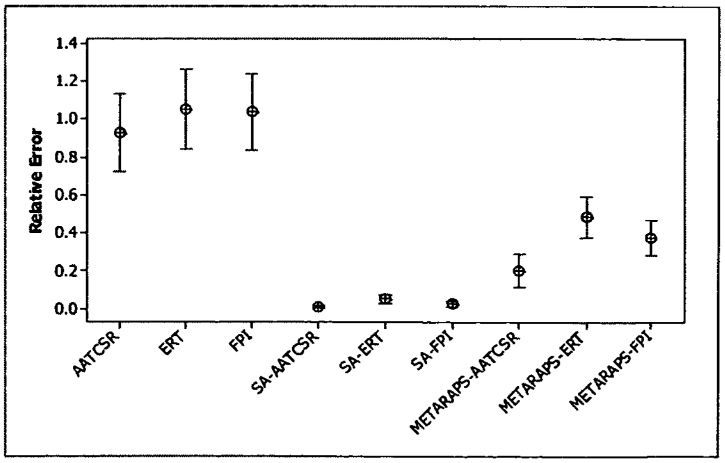


Figure 13. Mean plots intervals at the 95% confidence level of Relative Errors

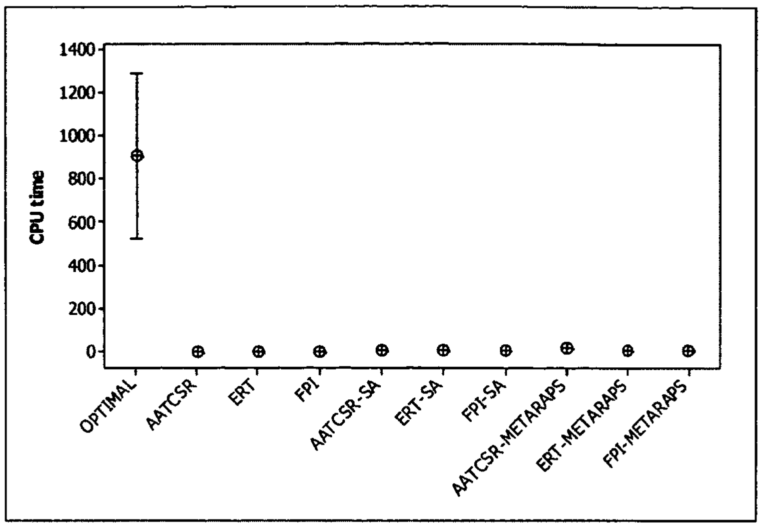


Figure 14. Mean plots intervals at the 95% confidence level of CPU Times

4.6 Further Analysis of the Algorithms' Effectiveness with respect to Feasibility

In this section, we focus on analyzing the algorithms' effectiveness in terms of producing initial feasible solutions especially for congested schedules. Table 17 summarizes the objective function values obtained by MILP (i.e. optimal), each greedy algorithm (i.e., AATCSR, ERT, FPI) and each corresponding integrated metaheuristic (i.e., SA_{AATCSR}, Meta-$RaPS_{AATCSR}$, SA_{ERT}, Meta-$RaPS_{ERT}$, SA_{FPI}, Meta-$RaPS_{FPI}$) for 55 instances generated in Ghoniem and Farhadi (2012); except that each aircraft was prescribed a time-window of 300 seconds. By reducing the time-window by half (from rj+600 to rj+300), we obtain more congested instances for which it is more difficult to obtain feasible solutions. The greedy heuristics generated more infeasible solutions than before (denoted as "-"). The table shows that out of 55 instances, AATCSR, FPI and ERT could not find any feasible solution for three, six and eleven instances respectively. SA and Meta-RaPS are applied to the problem with specific neighborhood schemes to efficiently generate feasible solutions (e.g., instance #9), improve initially constructed solutions by the proposed greedy algorithms (e.g., instance #1), and reach the optimal solutions (denoted as *) (e.g., instance #3). Therefore, it can be concluded through Table 17 that even though some of the initial solutions obtained by the greedy heuristics are infeasible, with the help of problem specific neighborhood search structures, the metaheuristics can find feasible, and sometimes optimal, solutions.

			Objective Function Value									
n	m	Instance	Optimal	AATCSR	SA_{AATCSR}	Meta-$RaPS_{AATCSR}$	ERT	SA_{ERT}	Meta-$RaPS_{ERT}$	FPI	SA_{FPI}	Meta-$RaPS_{FPI}$
15	2	1	2139	3137	2139*	2139*	3415	2139*	2346	3415	2139*	2311
		2	2173	4384	2173*	2173*	5232	2173*	2647	5232	2173*	2374
		3	6606	-	6606*	6631	-	6631	7782	-	6631	7774
		4	3352	4816	3352*	3352*	4886	3372	3765	4886	3352*	3352*
		5	1240	3322	1240*	1240*	3050	1240*	1999	3050	1240*	1240*
	3	6	581	1222	581*	581*	-	581*	627	1222	581*	581*
		7	1366	2406	1366*	1366*	-	1381	1699	2406	1366*	1366*
		8	761	1620	761*	962	1895	761*	1026	1895	761*	761*
		9	2349	3618	2519	2798	3831	2837	2888	3831	2855	2855
		10	753	1393	788	853	1803	787	918	1803	787	787
	4	11	2910	3447	2910*	2952	3463	2931	3225	3463	2931	3225
		12	3061	3107	3061*	3253	3746	3107	3513	3586	3107	3513
		13	2310	2768	2310*	2434	3019	2310*	2480	3019	2310*	2498
		14	452	936	452*	607	935	452*	648	935	452*	655
		15	468	732	468*	508	948	508	606	948	468*	542
20	2	16	5383	8208	5425	8040	-	5495	8757	-	5495	5495
		17	2186	4268	2186*	2187	-	2197	3360	5045	2186*	2947
		18	669	774	669*	669*	774	669*	774	774	669*	669*
		19	1004	1743	1058	1332	1743	1043	1080	1743	1043	1043
		20	1024	1744	1067	1278	1744	1068	1524	1744	1068	1068
	3	21	471	1326	471*	475	-	497	878	1326	475	475
		22	1575	4616	1575*	2391	5007	1616	2676	5007	1616	2654
		23	1783	3248	1839	1844	-	1862	2635	-	1839	1839
		24	1044	2287	1044*	1949	2287	1079	1994	2287	1115	1115
		25	2080	3673	2080	2670	3667	2148	2670	3667	2098	2098
	4	26	354	1157	361	849	1157	361	821	1157	361	859
		27	632	2046	657	1478	2046	681	1290	2046	657	1633
		28	666	1120	666*	1008	1120	972	972	1120	689	972
		29	521	1608	563	796	1608	600	1034	1608	586	1080
		30	587	1464	587*	1034	1464	587*	1022	1464	591	1012
	5	31	1645	2271	1685	1685	2271	1685	1819	1685	1645*	1819
		32	340	697	340*	340*	697	340*	604	697	340*	604
		33	690	1118	690*	1115	1118	707	1047	1118	707	1047
		34	352	869	357	648	869	356	648	869	356	648
		35	571	934	571*	587	917	571*	609	917	571*	609
25	2	36	51	51*	51*	51*	51*	51*	51*	51*	51*	51*
		37	1801	2705	1803	1803	-	1850	2711	-	1803	1850
		38	203	808	203*	203*	808	203*	203*	808	203*	203*
		39	1232	1952	1232*	1232*	2032	1238	1535	2032	1232*	1232*
		40	2798	5792	2798*	5401	5921	2829	4558	5921	2829	2829
	3	41	888	952	894	944	-	944	1352	1042	924	944
		42	235	354	302	317	360	259	354	360	259	345
		43	75	237	75*	75*	267	105	120	237	75*	75*
		44	4442	6613	4487	5295	7330	4460	6083	7330	4519	4519
		45	1602	-	1602*	1855	-	1679	1855	-	1644	1644
	4	46	356	365	356*	365	365	365	365	365	365	365
		47	345	938	375	624	938	412	751	938	407	702
		48	412	2087	412*	790	1935	451	1290	1935	412*	790
		49	2084	2923	2090	2681	2923	2114	2627	2923	2139	2681
		50	125	205	125*	146	205	140	199	205	125*	155
	5	51	1094	1690	1094*	1161	1690	1117	1219	1690	1094*	1219
		52	123	186	125	162	186	143	168	186	143	168
		53	2978	-	1689	1772	-	1745	1772	-	1689	1772
		54	1379	2144	1387	1857	2144	1420	1857	2144	1420	2114
		55	857	1458	930	1196	1458	1027	1279	1458	1027	1279

Table 17. Computational Results for the Algorithms with Congested Instances

CHAPTER 5

AIRCRAFT REACTIVE SCHEDULING PROBLEM METHODOLOGY

5.1 Introduction

We generally tend to assume the operational environment for decision making problems is stationary. Nevertheless, as the time passes by, a latest update on the situation requires reviewing the existing decisions that have been previously made. Such conditions are dynamic in a sense that as the operational settings change, the projected plans have to constantly be revised.

Diverse disruptions may occur during the execution of already planned service process due to the dynamic and uncertain operational environment, which could possibly cause failure in process quality, or continuity. In order to limit the negative consequences of disruptions, one should take a proper course of action. However it is quite clear that, it is not preferable to alter the existing decision significantly. For this reason, when we adopt a course of action, the existing decision has to be taken into account since we prefer to maintain conformity to the initial decision, and not perturb it much.

In industrial setting, the preplanned production schedules rarely remain fixed due to unexpected events. It frequently happens in manufacturing environment that an initial schedule is usually exposed to disruptions, and this makes rescheduling inevitable. Rescheduling is a dynamic approach where an original production schedule is updated in response to disruptions (Vieira et al., 2003). Examples of common disruptions are the arrival of new orders, order cancellations, due date changes, rush orders, machine breakdowns, resource unavailability, etc. Kanet and Sridharan (1990) stated that a production planning method which supports an effective rescheduling is necessary if one is looking for a quick and efficient response after disruptions or other changes.

In order to cope with dynamic structure in a scheduling environment, two scheduling approaches are considered: reactive scheduling and proactive scheduling.

Reactive Scheduling

Reactive Scheduling is an improvement of pre-computed predictive schedules which revises the schedule after the disruptions occur. In general, the reactive scheduling action is based on two strategies: *rescheduling* and *schedule repair.* A possible intuitive approach while reacting to the disruptions is generating a new schedule from scratch, which is called rescheduling. It is expected that this form of reactive scheduling generates a solution whose solution quality in terms of efficiency measure is superior to any repaired schedule. On the other hand, in practice, it is encouraged to ensure conformity to the initial schedule; therefore, generating a new schedule that is very dissimilar to the old one is not preferable. Schedule repair; on the other hand, modifies the existing initial schedule and provides relatively similar schedule to the old one. Heuristic based schedule repair algorithms may involve simple rules such as right-shift rule. When the disruptions are minor, the initial schedule can be adapted to the new conditions by schedule repair strategy. However, if the disruptions are frequent and large, rescheduling is may be better to apply.

Proactive Scheduling

Proactive Scheduling considers the uncertainty in generating schedules; where it is developed prior the start of the operations. Since proactive scheduling takes into account potential future disruptions, it tries to minimize their effect on performance measures while constructing an initial solution. The constructed schedule does not have to be optimal; however, it has to perform well in uncertain environment. Contrary to reactive scheduling, one of the main concerns while executing a proactive scheduling is optimizing the robustness (Wu et al.1993).

Due to the random events that change the system's state frequently, the schedules have to be reviewed at some points in time. Therefore, the timing, and the way that the review have to be done are the important parameters of reactive scheduling decision, which are going to be discussed in detail. There are different approaches to make a decision on timing of reactive scheduling. The system is periodically considered for scheduling in the *periodic scheduling*; the length of period can be either variable or constant. The reactive scheduling policies are considered at the beginning of each fixed-time in the constant-time interval. On the other hand, in the variable case, decisions are made after a certain amount of schedule is realized; however, in practice, constant-time interval method is more preferable. The schedule is updated after following a certain number of disruptions in the *continuous scheduling.* For instance, it takes a reactive scheduling action each time a random event occurs (Raman, Rachamadugu and Talbot, 1989). Another approach is *adaptive scheduling* where the schedule is reviewed after a predetermined amount of deviation from the initial schedule is observed (Sabuncuoglu, Karabuk, 1999). *Event-driven scheduling,* which is a hybrid method, revises the schedules both periodically and continuously as applied in periodic scheduling and continuous scheduling respectively.

The way that the schedules are revised consists of several components, and it is as crucial as timing for the reactive scheduling decision. The scheduling scheme, which can be *off-line, on-line* or *quasi-online* (i.e., *hybrid,* combination of off-line and on-line), is one of the components. Off-line scheduling method schedules all activities for the whole scheduling period providing a global perspective; alternatively, on-line scheduling scheme makes a decision one by one (e.g., dispatching rules), and it accommodates flexibility. Quasi-online scheduling method schedules a subset of the activities, and leaves the rest of the activities for future (Wu, Byeon and Storer, 1999).

The second component is amount of data used while generating a schedule. Forecast window, the time span of job release data; and simulation window, length of the simulation runs that scheduling decisions is made were defined in Kutanoglu and Sabuncuoglu (2001). The value of the simulation window may be equal to or smaller than the forecast window. If the simulation window is equal to the forecast window, it indicates that all available information is used. When the simulation window is smaller than the forecast window, only a part of the available information is used, and it is called partial scheduling.

Another component of the way that the schedules should be revised is the response types, which are used to manage unexpected events. The responses are categorized as *do* *nothing, reschedule all operations from scratch* and *repair the schedule by making alteration to the initial schedule.* However, the weakness and strengths of each response type are not analytically studied.

The last component is the performance metric. Generally, scheduling research has been concentrated around the schedule quality which includes objective functions such as the total weighted tardiness, makespan, earliness, and tardiness among others. However, when the operational environment is dynamic, *stability* and *robustness* performance measures must also be considered. These measures are related to the difference between the initial schedule and revised one. Uncertainty and unforeseen disruptions degrade schedule performances and cause variability. Robustness is concerned with the performance of the realized schedule. The revised schedule is robust if the objective function value of the new schedule does not deteriorate much (Wu, Storer and Chang, 1993). Stability, on the other hand, is concerned with the difference between the intial and realized schedules themselves, not just their performance metric. The revised schedule is called stable if it does not deviate much from the initial schedule when there is a disruption. Moreover, there is a trade-off between stability and robustness performance measures, and observing the conditions under which a measure is more important than another may lead to better understanding, comparison and selection of reactive scheduling methodologies (Wu et al., 1993).

As was mentioned, reactive scheduling problems consider both primary measure of schedule performance and stability measure of disruption caused by the rescheduling. Therefore, the objective in the aircraft reactive scheduling problem (ARSP) is not only minimizing the schedule's quality metric (the total weighted start times in this research) but also minimizing the schedule instability. The reason for using the total weighted start times as an objective function is that we would like to schedule the aircraft as early as possible to the runways. By this way, we can also minimize the start time of the latest positioned flight. The stability can be measured based on number of disrupted operations, differences between operation start times, or number of schedule changes made. In this research, the measure of stability can be defined as the difference in operations' start times, and deviation of runway assignment. Consideration of these objectives leads us to formulate the ARSP as a multi objective reactive scheduling problem.

5.2 Disruptions in Air Traffic Operations

Passenger satisfaction is one of the key considerations for airline companies that can be maximized by minimizing flight delays. Throughout the course of daily operations, an airline is faced with the potential of deviations in the planned flight schedule as a result of various unexpected events such as severe weather conditions and unexpected aircraft or personnel failures. However, very little research has been conducted on the problem disruptive events are considered. The main goal of this chapter is to propose and validate solution methodologies and heuristic procedures to reschedule the planned flights in the event of irregular operations. A mathematical formulation introduced by Ghoniem and Kharbeche (2013) of the aircraft rescheduling problem is utilized to obtain optimal solutions.

One of the most important aspects of tactical planning is to develop an airline's published flight schedule which requires a major effort. Generally, the schedule's planning process starts several months ahead of the actual operation of a given flight; and it is dependent on a broad array of information. Airlines are constantly faced with disruptions which may cause great variations from its planned flight schedules. The first priority for the airline is to restore the initial flight schedule as much as possible by minimizing the effects of numerous cancellations and delays. Therefore, real-time decisions should be made to treat the overall operations over some period of time.

In this dissertation, aircraft related disruptive events are studied because they occur much more frequent than other type of disruptions in air traffic operations. Aircraft reactive sequencing is studied and schedule repair algorithms are developed to deal specifically with the arrival of new aircraft, flight cancellations and aircraft delays due to unexpected events such as severe weather conditions, unexpected aircraft or personnel failures.

According to U.S. Department of Transportation, bad weather conditions are cited as the major cause of disruptions in the airline system accounting for 10% of the disruptions on the average. Due to Hurricane Sandy in 2012 for example, airlines cancelled over 20,000 flights in North America, which cost the U.S. carriers around $300 million.

At the majority of airline operation centers throughout the world, flight disruptions are mainly dealt with manually. With a reliance on the air traffic controller or responsible decision makers, and their past experience, the assessments are done, and decision about the schedules is made. Although it might be sufficient to deal with the irregularity for now, given the complexity of the problem, the airline operations control center needs effective real-time decision making tools and strategic schedules that minimizes the overall impact of disruptions on profitability, and operations.

5.3 Multi-Objective Optimization

Multi-objective optimization (MOO) is the process of optimizing a collection of objective functions systematically and simultaneously (Marler and Arora, 2004). A general multi-objective optimization problem is stated as follows:

Minimize $F(x)=[F_1(x), F_2(x), \ldots, F_k(x)]^T$

subject to $g_j(x) \leq 0, j = 1,2, \ldots, i$

$$h_l(x) = 0, j = 1,2, \ldots, e \tag{17}$$

where k is the number of objective functions, i is the number of inequality constraints, and e is the number of equality constraints.

In contrast to single-objective optimization problem, there is no single global solution in the multi-objective problem that minimizes all objectives simultaneously because the objective functions usually conflict with each other. When the solution quality improves, the other objective function, stability in our case, typically deteriorates. Pareto optimality is defined to describe the solutions for multi-objective optimization problem (Pareto, 1906).

Pareto Optimality

A point $x^* \in X$ is Pareto optimal if and only if no other feasible x$\in X$ exists such that $(x) \leq F(x^*)$, and $F_i(x) < F_i(x^*)$ for at least one objective function.

Sometimes, algorithms provide solutions that satisfy some criteria but may not be Pareto optimal but weakly Pareto optimal.

Weakly Pareto Optimal

A point $x^* \in X$ is weakly Pareto optimal if and only if no other feasible x$\in X$ exists such that $F(x) \leq F(x^*)$. It can be inferred that weakly Pareto optimal points are not Pareto optimal but Pareto optimal points are weakly Pareto optimal (Marler and Arora, 2005).

5.3.1 Multi-Objective Optimization Methods

There are four basic multi-objective optimization methods that are categorized according to the preferences of the decision maker in the solution process: no-preference methods, priori methods, posteriori methods, interactive methods.

No-preference Methods

If there is not any decision maker and his/her preference information available, it can be considered as no-preference method. It can obtain some unbiased compromised solution without any additional preference information.

Priori Methods

In order to obtain a single Pareto optimal point, the decision makers express the relative importance of the objective functions and then the method looks for a Pareto optimal solution satisfying the preferences as much as possible (Miettinen and Hakanen, 2009). Hierarchical approaches and simultaneous approaches are two primary approaches used with this method. When the decision maker arranges the objectives according to their importance for subsequent solution by a single objective optimization method (Azizoglu and Alagoz, 2005), it is called a lexicographic optimization. The lexicographic optimization would be an example for hierarchical approach. A simultaneous approach contains value function methods that include the original objectives and preferences of the decision makers for optimization, and then a single objective optimization problem is solved. The weighted sum method, which will be discussed in this dissertation, is an example for simultaneous approach. Since a scalarized objective function is used in the rescheduling problem, the weighted sum method can be used as a posteriori method so that different weights are set to generate different Pareto optimal solutions, and then the decision maker can select the most satisfactory one.

Posteriori Methods

Population based methods such as multi-objective simulated annealing, differential evolution and nondominated sorting genetic algorithm belong to the posteriori methods (Rangaiagh, 2008). A representation of the entire Pareto optimal set is generated and a single solution is selected from a set of mathematically equivalent solutions which satisfy the preferences. This can be obtained by solving a series of MOO problems by changing the coefficients of the objective functions (Marler and Arora, 2005). Posteriori methods may provide many Pareto optimal solutions, which may be computationally expensive, to the decision makers who review and select one for implementation (Miettinen and Hakanen, 2009). In this way, the decision makers get an overview of the solutions; however, it might be difficult for them to analyze a large amount of information.

Interactive Methods

In these methods, a solution pattern is formed and repeated, and the decision maker can specify the preference information progressively during the solution process. During the solution of the multi-objective optimization problem, these methods require interaction with the decision makers. Several interactive methods exist in the literature, and none of them is superior to all the others; however, some of them may fit different decision makers and problems better than the others. Examples of these methods are reference point approaches, satisficing trade-off method, the NIMBUS method (Miettinen and Hakanen, 2009).

5.3.2 Weighted Sum Method

The most common approach to multi-objective optimization is the weighted sum method in which, a convex combination of functions is reformulated. The general multi-objective optimization problem is stated below:

$\min \sum_{i=1}^{k} \lambda_i F_i(x)$

s.t. $g_j(x) \leq 0; j=1,2,\ldots, J$

$h_l(x) \leq 0;\ l=1,\ 2,\ldots,\ L$

$\sum_{i=1}^{k} \lambda_i = 1$ (18)

where λ_i is the weighting factor for the ith objective function, the weights are non-negative $\lambda_i \geq 0$ for all $i=1,2,\ \ldots,\ J$. Changing the weighting factor's relative values changes the orientation of the contours for the weighted sum. Minimizing the weighted sum can yield Pareto optimality (Miettinen and Hakanen, 2009). Mathematically, the weights are related to the decision maker's preference function in Steur (1999).

5.3.3. Normalization in the weighted sum method

For the sake of consistent evaluation of objective function values, it is advantageous to transform the original objective functions; this is especially true with scalarization methods, which involve a priori expression of preferences. When the objective functions for a problem have significantly different orders of magnitude, determining suitable weighting factor is difficult. Therefore, the objective functions should be transformed by the normalization techniques such that functions are dimensionless. Some common normalization methods are summarized by Marler and Arora (2005) as follows:

Normalization by the minimum of the objective functions

This approach yields non-dimensional objective function value with a lower limit of one. It is referred to as the *lower-bound approach*, and a common approach to function transformation is given as:

$$f_i^{normal} = \frac{f_i(x)}{|f_i^{min}|} \tag{19}$$

where $f_i^{min}$is the minimum value for objective *i,* and can be defined as an ideal point such as $f_i^{min} = f_i(x^*)$ where $f_i(x^*)$ is the optimum value of objective i when it is optimized individually disregarding the other objectives .The upper limit of the $f_i^{normal}$is unbounded; on the other hand, the lower limit of $f_i^{normal}$is restricted to non-negative when $f_i^{min} > 0$.

Alternatively, the numerator may also be modified which also provides a non-dimensional objective function in which case, the lower limit of $f_i^{normal}$is restricted to zero. It is referred to as the *alternate lower-bound approach*, and transformation is given as follows.

$$f_i^{normal} = \frac{f_i(x) - f_i^{min}}{|f_i^{min}|} \tag{20}$$

Normalization by the maximum of the objective functions

Known as the *upper-bound approach*, is a variation on lower-bound approach that uses the maximum value of the function in the denominator rather than f_i^{min} as follows:

$$f_i^{normal} = \frac{f_i(x)}{f_i^{max}} \tag{21}$$

where $f_i^{max}$is the maximum value for objective *i*. This approach yields non-dimensional objective function value such that $f_i^{normal} \leq 1$ with no restriction on the lower value.

The most robust approach to normalize the objective functions, regardless of their original range is called the *upper-lower bound approach*, and is given as follows:

$$f_i^{normal} = \frac{f_i(x) - f_i^{min}}{f_i^{max} - f_i^{min}} \tag{22}$$

The denominator of the formulation is the length of the intervals over which the objective functions vary within the Pareto optimal set. In this case, f_i^{normal} gets values between zero and one, and unlike previous approaches, the denominator is guaranteed to be positive since$f_i^{max} > f_i^{min}$.

According to Grodzevich and Romanko (2006), since the objective functions are normalized by the true values of their variation over the Pareto optimal set, upper-lower bound approach provides relatively robust normalization. The following scalarized objective function is used for the multi-objective optimization problem (Rangaiah, 2008).

$$Min\ \lambda \frac{f_1(x)-f_1(x^*)}{f_1^{max}-f_1(x^*)} + (1-\lambda)\frac{f_2(x)-f_2(x^*)}{f_2^{max}-f_2(x^*)} \tag{23}$$

where $0 < \lambda < 1$ is a weighting factor. $f_1(x^*)$ and $f_2(x^*)$ are the optimal values of the individual objectives when they are optimized individually, and $f_i^{max}$and $f_i^{min}$are the respective maximum and minimum values of the individual objectives. The provided upper-lower boung approach will be used for the normalization since it is one of the most robust approaches to normalize the objective functions.

5.4 The Mixed Integer Linear Programming Model for Aircraft Rescheduling Problem

5.4.1 The Initial MILP Model

The initial MILP model for ARSP is provided in Ghoniem and Kharbeche (2013) where runway re-assignment and constrained position shifting (CPS) are allowed but penalized. In addition to existing scheduling constraints discussed in section 3.1 the objective function, new parameters, decision variables and constraints are modified. Initial runway assignments and start times are used as input in the revised model. In order to consider both the solution quality and stability in the objective function, a multi-objective model is constructed.

Index Sets and Notation

$M = \{1,2,\ldots, m\}$: A set of m identical runways; runways are indexed using $i=1,2,\ldots, m$.

$\bar{J}=\{1,2,\dots,n\}$: A set of n aircraft (i.e., landing or departing) from the initial schedule which can be rescheduled.

$D \subseteq \bar{J}$: A set of delayed aircraft.

$E \subseteq \bar{J}=\{1,2,\dots,n\}$: Set of cancelled aircraft.

A: a set of new aircraft arrivals

$J \equiv (\bar{J} - E) \cup A =$ A set that includes all aircraft that are considered for rescheduling.

r_j: ready time for aircraft j to take-off or land (taxi time is not included), $\forall j \in J$

δ_j: target time for aircraft j to take-off or land, $\forall j \in J$

d_j: deadline for aircraft j to take-off or land, $\forall j \in J$

Oj: operation type of aircraft j, being a landing or a departure, $\forall j \in J$.

C_j: weight class of aircraft j, e.g., heavy, medium, or light, $\forall j \in J$.

w_j: weight assigned to aircraft j based on its operation type and its weight class, $\forall j \in J$. In particular, higher priority has been assigned to landings over departures and to heavy aircraft over medium and light ones. Moreover, in the test-bed $w_{j1}=w_{j2}$ if $O_{j1}=O_{j2}$ and $C_{j1}=C_{j2}$.

α_j: Penalty cost of deviation from the initial start time of aircraft j, $\forall j \in \bar{J} - (D \cup E)$

β_j: Penalty cost of deviation from the initial assignment of the aircraft j, $\forall j \in \bar{J} - (D \cup E)$

s_{kj}: sequence-dependent minimum separation time required between aircraft k and j if they are respectively the leading and the following aircraft, $\forall\ k, j \in J, k \neq j$.

π_1: Preemptive coefficient for the deviation from the initial start times and the initial runway assignment for all aircraft that are being rescheduled but not experiencing any disruption.

π_2: Preemptive coefficient for the sum of the total weighted start times of all aircraft that are being rescheduled.

The solution associated with the initial schedule are now input for the ARSP

$\bar{t}_j$: the start time of the operation for aircraft j in the initial schedule, $\forall j \in \bar{J}$

$$\bar{z}_{ij} = \begin{cases} 1, \text{ if aircraft } j \text{ has been assigned to runway } i \text{ in the initial schedule}, \forall\, i \in M, \forall j \in \bar{J}. \\ 0, \text{ otherwise} \end{cases}$$

Decision Variables

t_j: the start time of aircraft j (i.e. the time for departure or landing), $\forall\, j \in J$.

$$z_{ij} = \begin{cases} 1, \text{ if aircraft } j \text{ is assigned to runway } i, \forall\, i \in M, j \in J. \\ 0, \text{ otherwise} \end{cases}$$

$$y_{kj} = \begin{cases} 1, \textit{if both aircraft } k \textit{ and } j \textit{ are assigned to the same runway} \\ 0, \textit{otherwise where } t_k > t_j, \forall\, k, j \in J, k \neq j \end{cases}$$

$$\text{Min } \pi_1\left(\textstyle\sum_{j \in \bar{J}-(D \cup E)} \alpha_j (g_j + q_j) + \sum_{j \in \bar{J}-(D \cup E)} \beta_j (u_j + o_j) \right) + \pi_2 \textstyle\sum_{j \in J} w_j t_j \tag{24}$$

$$\text{s.t.} \quad \textstyle\sum_{i \in M} z_{ij} = 1, \forall\, j \in J \tag{25}$$

$$r_j \leq t_j \leq d_j, \forall j \in J \tag{26}$$

$$t_j \geq t_k + s_{kj} - (1 - y_{kj})(d_k - r_j + s_{kj}), \forall\, k, j \in J, k \neq j \tag{27}$$

$$y_{kj} + y_{jk} \geq z_{ik} + z_{ij} - 1, \forall\, i \in M, \forall\, k, j \in J, k \neq j \tag{28}$$

$$t_j + g_j - q_j = \bar{t}_j, \forall\, j \in \bar{J} - (D \cup E) \tag{29}$$

$$\textstyle\sum_{i \in M} i z_{ij} + u_j - o_j = \sum_{i \in M} i \bar{z}_{ij}, \forall\, j \in \bar{J} - (D \cup E) \tag{30}$$

$$y, z \textit{ binary}, g, q, u, o \geq 0. \tag{31}$$

The objective function (24) minimizes respectively the components, the sum of the total weighted deviation from the initial start times, the total weighted deviation from the initial runway assignments for all aircraft are rescheduled but not experiencing any disruption, and the total weighted start times of all aircraft. The aircraft that are experiencing any disruption (i.e., cancelled), are not included in the calculation of total weighted start time deviation and total weighted runway deviation since for instance, the start time deviation of a cancelled flight is impractical to measure. Constraint (25) satisfies that every aircraft is assigned to exactly one of the *m* runways. Constraint (26) specifies allowable ready time-deadline time-window restrictions. Constraint (27) satisfies minimal separation times between aircraft that are assigned to the same runway. Constraint (28) specifies that if two aircraft are assigned to the same runway, then one must operate before the other. Constraint (29) reflects the deviation from initial start times. Constraint (30) reflects the deviation from initial runway assignments. Constraint (31) defines the binary and sequencing decision variables.

5.4.2 The Revised MILP Model

In order to have a fair analysis, and study the contribution of each objective function component on the total objective function, a set of three new coefficients is introduced for the three different components as follows:

π_1: coefficient/weighting factor for the total weighted start time deviation (TWSD) from the initial start times for all aircraft that are being rescheduled but not experiencing any disruption.

π_2: coefficient/weighting factor for the total weighted runway deviation (TWRD) from the initial runway assignment for all aircraft that are being rescheduled but not experiencing any disruption.

π_3: coefficient/weighting factor for the sum of the total weighted start times (TWS) of all aircraft that are being rescheduled.

Referring to Section 5.3.3, we have revised the MILP model to incorporate the normalized objective function. At first, the MILP model is updated as follows:

$$\text{Min}\ \pi_1 \sum_{j\in \bar{J}-(D\cup E)} \alpha_j(g_j + q_j) + \pi_2 \sum_{j\in \bar{J}-(D\cup E)} \beta_j(u_j + o_j) + \pi_3 \sum_{j\in} w_j t_j \qquad (32)$$

where $\pi_1 + \pi_2 + \pi_3 = 1$. Then, the objective function is rewritten in a normalized format to minimize a linear convex combination of normalized instability and total weighted start time objectives where each objective function component has an allocated weighting factor indicating its relative importance. The objective function of the ARSP is as follows:

$$\text{Min}\, \pi_1 \left(\frac{f_{TWSD}(x) - \widetilde{f_{TWSD}}(x^*)}{\widetilde{f_{TWSD}^{max}} - \widetilde{f_{TWSD}}(x^*)}\right) + \pi_2 \left(\frac{f_{TWRD}(x) - \widetilde{f_{TWRD}}(x^*)}{\widetilde{f_{TWRD}^{max}} - \widetilde{f_{TWRD}}(x^*)}\right) + \pi_3 \left(\frac{f_{TWS}(x) - \widetilde{f_{TWS}}(x^*)}{\widetilde{f_{TWS}^{max}} - \widetilde{f_{TWS}}(x^*)}\right) \quad (33)$$

Or

$$\text{Min}\, \pi_1 \left(\frac{\sum_{j \in \bar{J} - (D \cup E)} \alpha_j (g_j + q_j) - \widetilde{f_{TWSD}}(x^*)}{\widetilde{f_{TWSD}^{max}} - \widetilde{f_{TWSD}}(x^*)}\right) + \pi_2 \left(\frac{\sum_{j \in \bar{J} - (D \cup E)} \beta_j (u_j + o_j) - \widetilde{f_{TWRD}}(x^*)}{\widetilde{f_{TWRD}^{max}} - \widetilde{f_{TWRD}}(x^*)}\right) +$$
$$\pi_3 \left(\frac{\sum_{j \in} w_j t_j - \widetilde{f_{TWS}}(x^*)}{\widetilde{f_{TWS}^{max}} - \widetilde{f_{TWS}}(x^*)}\right) \quad (34)$$

where $\widetilde{f_{TWSD}}(x^*), \widetilde{f_{TWRD}}(x^*), \widetilde{f_{TWS}}(x^*)$, $\widetilde{f_{TWSD}^{max}}$, $\widetilde{f_{TWRD}^{max}}$, $\widetilde{f_{TWS}^{max}}$ are the estimates of the optimal and the maximum values of the individual objective components. Each represent a theoretical optimistic or a pessimistic objective value for an individual objective. In order to find these values, it is required to implement the MILP model three times by using different combinations of π_1, π_2, π_3. For instance, $\widetilde{f_{TWSD}}(x^*)$ is the estimate value of the minimum total weighted start time deviation for a given instance, and can be estimated after the first run by setting

$\pi_1 = 1 - \pi_2 - \pi_3$, if $\pi_2 = \pi_3 = \varepsilon$, where ε is a very small positive real number.

Similarly, $\widetilde{f_{TWSD}}(x^*)$ is the estimate value of the minimum total weighted start time deviation for a given instance, and can be estimated after the first run by setting

$\pi_2 = 1 - \pi_1 - \pi_3$, if $\pi_1 = \pi_3 = \varepsilon$, where ε is a very small positive real number.

Finally, $\widetilde{f_{TWS}}(x^*)$, which is the estimate value of the minimum total weighted start time for a given instance, and can be estimated after the second run by setting $\pi_3 = 1 - \pi_1 - \pi_2$, where $\pi_1 = \pi_2 = \varepsilon$.

$\widetilde{f_{TWSD}}(x^*) = 0$, optimistic TWSD value of the case that start times of the aircraft are not affected by the disruption.

$\widetilde{f_{TWRD}}(x^*) = 0$, optimistic TWRD value of the case that the runway assignments of the aircraft are not affected by the disruption.

$\widetilde{f_{TWS}}(x^*)$ = optimistic TWS value of the case that start times of the aircraft are minimum.

$\widetilde{f_{TWSD}^{max}}$ = pessimistic TWSD value of the case that start times of aircraft get worst after the disruption.

$\widetilde{f_{TWRD}^{max}}$ = pessimistic TWRD value of the case that the runway assignments of aircraft get worst after the disruption.

$\widetilde{f_{TWS}^{max}}$ = pessimistic TWS value of the case that start times of all aircraft get worst after the disruption.

In order to clarify the methodology of estimating the minimum and maximum values of objective functions, a sample data for the problem with 15 aircraft and 2 runways is given as follows:

Aircraft = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]

Ready Times = [35, 51, 76, 218, 237, 264, 203, 57, 109, 165, 122, 252, 354, 441, 276]

Target Times = [95, 111, 136, 278, 297, 324, 263, 117, 169, 225, 182, 312, 414, 501, 336]

Deadlines = [635, 651, 676, 818, 837, 864, 803, 657, 709, 765, 722, 852, 954, 1041, 876]

Weights = [6, 2, 6, 4, 1, 5, 3, 2, 2, 2, 3, 6, 1, 5, 5]

For the given problem instance, the best total weighted start time value is 177 obtained by exact solution. The initial schedule is shown in Figure 15 with the following operation start times and runway assignments.

Start times = [35, 75, 107, 231, 299, 264, 261, 57, 147, 207, 135, 311, 406, 478, 371]

Runway Assignment = [[1 0], [1 0], [0 1], [1 0], [0 1],[0 1], [1 0], [0 1],[0 1], [0 1], [1 0],[1 0], [0 1], [0 1],[0 1]] where [1 0] means that an aircraft is assigned to runway 1 and [0 1] is that it is assigned to runway 2.

Runway #1	1	2	11	4	7	12			
Runway #2	8	3	9	10	6	5	15	13	14

Figure 15. Initial Schedule Before Disruption

Suppose that there is an aircraft cancellation, which is aircraft # 3, and only the aircraft that might be affected by the disruption are going to be repaired. Since the aircraft on the 1st runway are not affected by the cancelation, and aircraft 8 is assigned prior to 3, they are then removed from the aircraft set that need repair leaving the set of aircraft to repair as aircraft # 5,6,9,10,13,14,15. In order to estimate $\widetilde{f_{TWSD}}(x^*), \widetilde{f_{TWRD}}(x^*), \widetilde{f_{TWS}}(x^*)$, $\widetilde{f_{TWSD}^{max}}, \widetilde{f_{TWRD}^{max}}$ and $\widetilde{f_{TWS}^{max}}$, initially, the data set is updated based on the disruption information (i.e., the parameter values for aircraft #3 are omitted since it is cancelled).

Then, by setting

i) $\pi_1 = 1 - \pi_2 - \pi_3$ when $\pi_2 = \pi_3 = \varepsilon = 8.854 \times 10^{-12}$ regarding the total weighted start time deviation.

$\widetilde{f_{TWSD}}(x^*) = 0$, where there is no start time deviation

ii) $\pi_2 = 1 - \pi_1 - \pi_3$ when $\pi_1 = \pi_3 = \varepsilon$ regarding the total weighted runway deviation.

$\widetilde{f_{TWRD}}(x^*) = 0$, where there is no runway assignment deviation

$\widetilde{f_{TWSD}^{max}}=10752$

iii) $\pi_3 = 1 - \pi_1 - \pi_2$ when $\pi_1 = \pi_2 = \varepsilon$

$\widetilde{f_{TWS}}(x^*) = 10752$, where the total weighted start times is minimized

$f_{TWRD}^{max} = 32$, $f_{TWS}^{max} = 21314$

As a summary, it is estimated that the $[\widetilde{f_{TWSD}}(x^*)\,,\ \widetilde{f_{TWRD}}(x^*)\,,\widetilde{f_{TWS}}(x^*)] = [0, 0, 10752]$ and $[f_{TWSD}^{max} f_{TWRD}^{max} f_{TWS}^{max}] =] = [10752, 32, 21314]$.

Accordingly, the objective function for this problem is updated as follows:

$$\text{Min } \pi_1 \left(\frac{f_{TWSD}(x)-0}{10752-0}\right) + \pi_2 \left(\frac{f_{TWRD}(x)-0}{32-0}\right) + \pi_3 \left(\frac{f_{TWS}(x)-10752}{21314-10752}\right) \qquad (35)$$

The optimal schedule obtained for the ARSP by using the objective function with $\pi_1 = \pi_2 = 0.5,\ \pi_3 = 0$ is shown in Figure 16.This solution has the objective value of 0 with TWSD=0, TWRD=0, and TWS=11430.

Runway #1	1	2	11	4	7	12			
Runway #2	8		9	10	6	5	15	13	14

Figure 16. Optimal Solution After Disruption

In the computation study chapter, the experiments will be conducted with various weighting factor for analysis.

5.5 Reactive Scheduling Algorithms

Suppose that the initial schedule (i.e., initial assignment of each aircraft to a runway, and operation start times) is given and that we periodically receive disruption information of flight cancellations, flight delays, and new flight arrivals. If there is a flight cancellation, the number and indices of the cancelled flights are provided. If there is a flight delay, number of delayed flight and amount of delay are provided. Finally, if there is a new unexpected flight arrival, the number of the new aircraft, index of the new aircraft, the parameter information (e.g., ready time, target time, deadline, etc.) of the new flight are provided. The data is updated based on the disruption information. Once the status is updated, the response strategies to each corresponding disruption are executed; and the revised schedules are obtained. The general control framework of the rescheduling strategy problem is provided in Figure 17.

In this dissertation, unlike the studies in the literature which focus on one type of disruption at a time, different types of disruptions with multiple disruptive events are considered simultaneously. Therefore, the sequential evaluation methodology is developed to treat the disruptions and revise the schedules periodically. When there is at least one disruption, firstly, we check whether a flight cancellation is observed or not. If that is the case, a response algorithm is executed for cancellation, and a revised schedule is generated as the current schedule. Then, the next step is to check whether a flight delay is observed in which case a response algorithm is executed for delay, the schedule is repaired based on a response, and the obtained new schedule is updated as a current schedule. Finally, we check whether a new unexpected flight arrival is observed or not, and respond accordingly. The detailed framework of the control phase employed in this research is depicted in Figure 18.

Church and Uzsoy (1992) state that the disrupted schedules can be repaired by three reactive scheduling methods: right shift rescheduling, partial rescheduling and complete regeneration. In the right shift rescheduling, each remaining operation is postponed by the amount needed to obtain a feasible schedule. Partial rescheduling algorithm reschedules only the operations that are affected. Complete regeneration algorithm reschedules the entire set of operations from scratch.

The problem addressed here is a multi-objective optimization problem where both solution quality (i.e., total weighted start time) and stability (i.e., total weighted start time deviation, total weighted runway deviation) of the solution are considered by a decision maker.

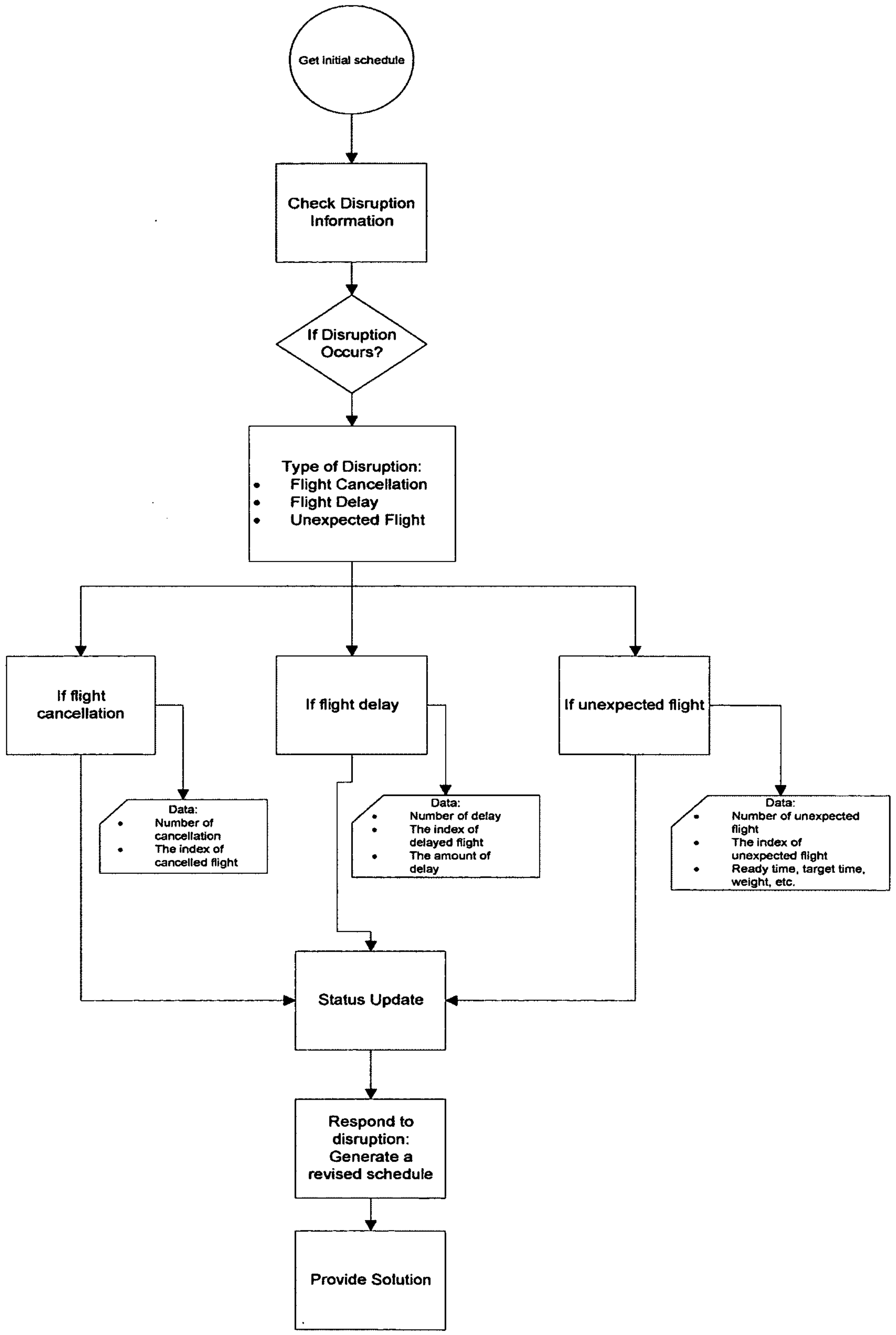
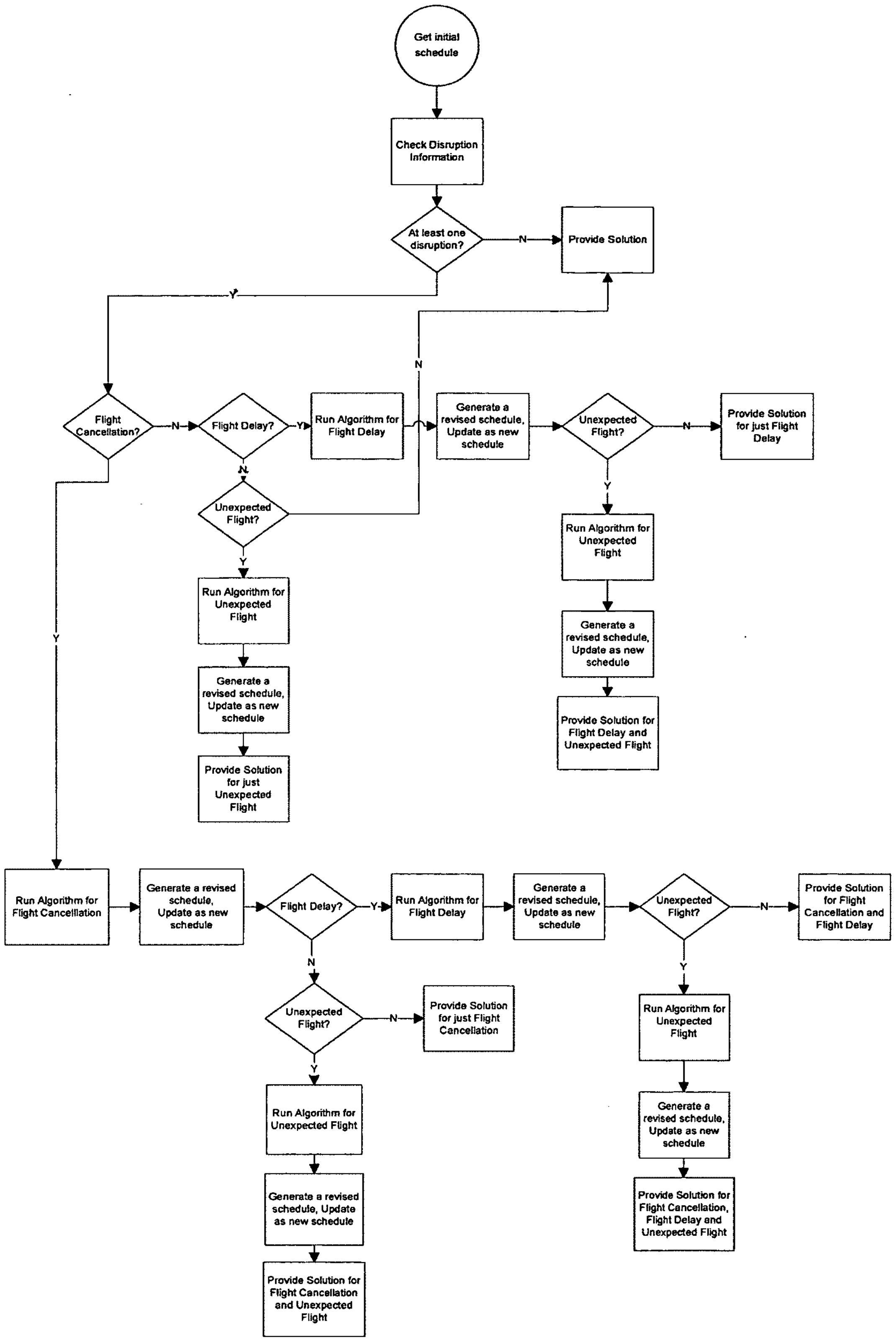


Figure 17. General Control Framework for the Rescheduling Strategy



Get initial schedule
Check Disruption Information
At least one disruption?
N
Provide Solution
Y
N
Flight Cancellation?
N
Flight Delay?
Y
Run Algorithm for Flight Delay
Generate a revised schedule, Update as new schedule
Unexpected Flight?
N
Provide Solution for just Flight Delay
Y
Run Algorithm for Unexpected Flight
Generate a revised schedule, Update as new schedule
Provide Solution for Flight Delay and Unexpected Flight
N
Unexpected Flight?
Y
Run Algorithm for Unexpected Flight
Generate a revised schedule, Update as new schedule
Provide Solution for just Unexpected Flight
Y
Run Algorithm for Flight Cancellation
Generate a revised schedule, Update as new schedule
Flight Delay?
Y
Run Algorithm for Flight Delay
Generate a revised schedule, Update as new schedule
Unexpected Flight?
N
Provide Solution for Flight Cancellation and Flight Delay
Y
Run Algorithm for Unexpected Flight
Generate a revised schedule, Update as new schedule
Provide Solution for Flight Cancellation, Flight Delay and Unexpected Flight
N
Unexpected Flight?
N
Provide Solution for just Flight Cancellation
Y
Run Algorithm for Unexpected Flight
Generate a revised schedule, Update as new schedule
Provide Solution for Flight Cancellation and Unexpected Flight

Figure 18. Detailed Control Framework for the Rescheduling Strategy

Therefore, we developed alternative reactive scheduling strategies that consist of repairing and rescheduling algorithms for each type of disruptive event to incorporate the multi-objectivity and to update the schedule. A complete regeneration method, *TWST Algorithm,* is a greedy rule which treats flight cancellation, delay and unexpected arrivals simultaneously. Another complete regeneration method is a hybrid-metaheuristic called *SA-Re Algorithm,* which gets the initial solution from the TWST Algorithm and then applies simulated annealing algorithm. In addition to complete regeneration methods, partial repair methods are also proposed. The proposed responses for each disruption are considered at every decision point, and then, the best (i.e., the one with the minimum objective function value) reactive sequencing policy is identified from several candidates. *Do-Nothing* and *Left-Shift* are the repair strategies which are considered specifically for the flight cancellation disruption. Partial repair strategies, *RepairBySlack (*Reschedule the affected flights by minimum slack time), *RepairByEDD(* Reschedule the affected flights by earliest deadline) and *InsertDelayed Algorithms (*Insert the delayed flights to the best position), are proposed to repair the schedule after the delays in particular. *RepairByTWST* and *InsertNew Algorithms* are the repair strategies for the unexpected flight arrival disruption. Figure 19 illustrates the reactive scheduling strategies for corresponding disruptive event.

These response strategies require an initialization stage to determine the rescheduling point and the set of aircraft that are affected by the disruption. It is assumed that the initial schedule is given, and at the beginning of every time period, the disruption information is updated in advance. To illustrate the problem and the solution methods, an example is provided for the instance with the number of aircraft=20 and number of runways=2. Suppose also that the initial aircraft schedule on each runway before any disruption is available.

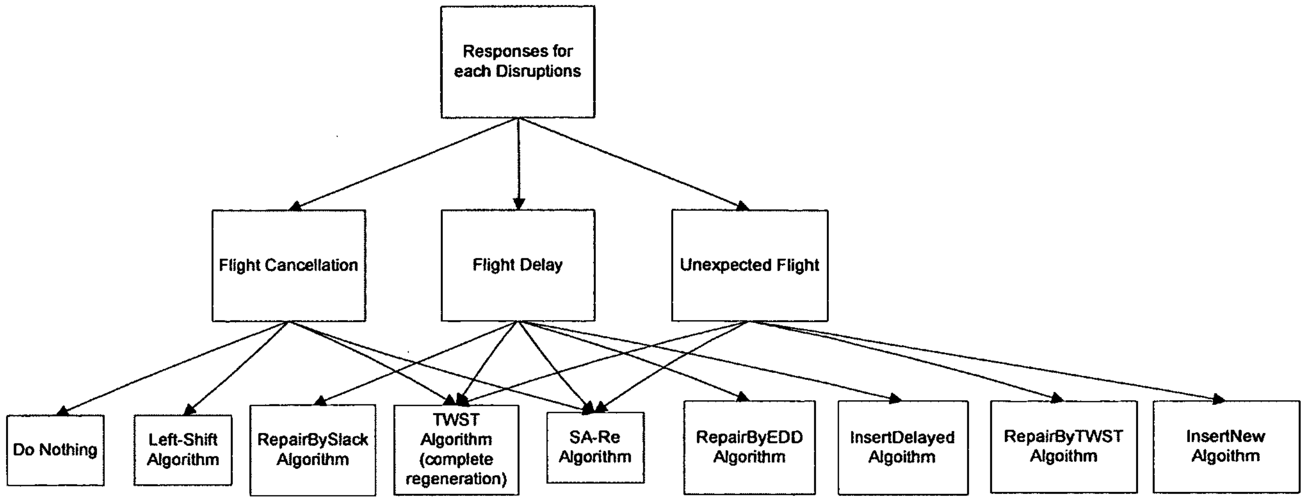
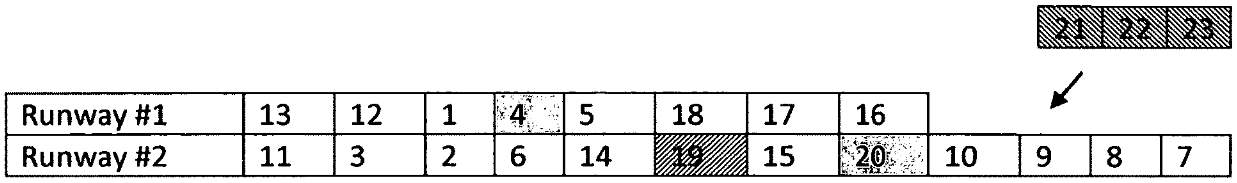


Figure 19. Reactive Scheduling Strategies for Each Disruption Type

The disruption information are as such that there are 2 cancellations (flights #4 and #20), there is one delay (flight #19), and there are 3 unexpected new flights (# 21,# 22, #23). The cancelled, the delayed and the new unexpected flights are shaded in Figure 20. The initial start times of the aircraft on each runway before any disruption are shown in Figure 21.



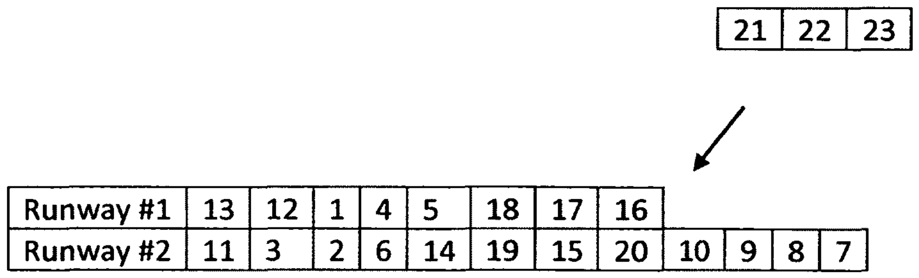
										21	22	23
Runway #1	13	12	1	4	5	18	17	16				
Runway #2	11	3	2	6	14	19	15	20	10	9	8	7

Figure 20. A Sample Initial Schedule

Runway #1	121	156	216	294	329	588	668	703				
Runway #2	71	169	199	282	328	388	453	483	536	573	643	678

Figure 21. Initial Start Times of Each Aircraft Before Any Disruptions

The set of affected aircraft; in other words, the potential set of aircraft to be rescheduled (shaded aircraft) is provided in Figure 22.



21	22	23

Runway #1	13	12	1	4	5	18	17	16				
Runway #2	11	3	2	6	14	19	15	20	10	9	8	7

Figure 22. Affected Aircraft to Reschedule

5.5.1 TWST Algorithm

TWST is a complete regeneration algorithm that reschedules all flights from scratch. The Total Weighted Start Time (TWST) Algorithm is developed to minimize TWST in a multi-objective optimization problem. The reason for proposing this algorithm is that one of the components of the total objective function is total weighted start time; therefore, it would be a considerable approach to propose an algorithm to handle TWST minimization. The procedure for TWST Algorithm is given below.

TWST Algorithm

Let J be a set of unscheduled aircraft.

Step 1. Get initial schedule (schedule before any disruption).

Step 2. Check the disruptions information (cancellation, delay, unexpected flight).

Step 3. Update the input data (i.e., ready time, target time, etc.) according to the following disruption types:

For cancellation: remove the cancelled flight and its parameter from the input data

For delay: update the ready time, target time and deadline with respect to the delay time

For new unexpected: insert new flight and its parameters to the input data

Step 4. $For\ \forall j \in \{J \cup A\}$, Calculate the ratio using $w_j/(r_j + s_{kj})$ where w_j, r_j, s_{kj} are the weight of flight j, ready time of flight j, and sequence-dependent separation time between preceding flight k and flight j respectively.

Step 5. According to the ratio that is calculated in Step 4, assign the aircraft with the largest ratio to a runway on which its operation can start earlier.

Step 6. Remove flight j from set J.

Step 7. Update the makespan of the runway where the aircraft is assigned on.

Step 8. Go to Step 4 and continue until all aircrafts are scheduled.

Step 9. Calculate the normalized total weighted start time deviation, total weighted runway deviation and total weighted start time as an objective function.

After determining the set of aircraft to reschedule, the TWST Algorithm assigns the flight by the largest ratio of weight to ready time plus separation time. The $w_j/(r_j + s_{kj})$ ratio is computed for each unscheduled aircraft, and the aircraft with the largest value is assigned to a runway on which its operation can start earliest. The rationale of using this ratio is that the weighted shortest processing time first (WSPT) is a greedy rule that is applied for minimizing the total weighted completion time when there is a single machine scheduling problem in the literature (Pinedo, 2008). Considering the unequal ready time, sequence dependent separation time and multiple resource aspect of our problem, TWST greedy algorithm is proposed. Since the contribution of the total weighted start time to the total objective function value is comparably high, stating an algorithm which takes care of the solution quality is a reasonable methodology.

5.5.2 SA-Re Algorithm

SA-Re is a rescheduling algorithm that reschedules all flights from scratch. The main algorithm is similar to the SA metaheuristic which was successfully implemented earlier in Section 3.3. SA-Re is introduced for the problem to improve initially constructed solutions by the proposed TWST algorithm. Therefore, this method primarily aims to obtain solutions to minimize the total weighted start time as well. The procedure for SA-Re is given below.

SA-Re Algorithm

Step1: Get the initial solution from TWST Algorithm, and update the objective function value.

Step 2: Set the initial temperature as a function of the current objective function value $T = k. f(\theta)$ as in Section 3.3.

Step 3: Generate a new solution in the neighborhood of the initial solution by applying neighborhood search algorithms explained in Section 3.3.

Step 4: Compare objective function value of new candidate solution to the current value. If the new value is better, accept it. If it is worse, then accept the new solution with a probability of acceptance $(rand\ [0,1] \leq e^{-\frac{\Delta\theta}{T}},\ where\ \Delta\theta = f(\theta') - f(\theta))$.

Step 5: Cool down/update the temperature ($T = a.T$).

Step 6: Go to Step 3 and continue until stopping criteria.
($c < t_{max}$ where iteration counter hits maximum number of iterations) is satisfied

Step 7: Update the solution, and objective function value.

5.5.3 Do-Nothing Algorithm

Do-Nothing strategy is a type of response that is applied when flight cancellation disruption occurs. After taking out a flight from its position on a runway, no corrective action is taken for the remaining flights assigned on that runway. Therefore, Do-Nothing keeps the initial runway assignments, flights sequence on each runway and start times as they are. The rationale behind this response is to generate a stable schedule that does not deviate much from the initial schedule after a flight is canceled. The procedure for the Do-Nothing strategy is given as follows:

Do-Nothing Algorithm

Step 1. Get initial schedule (schedule before any disruption).

Step 2. Get the canceled flight j, $\forall j \in E$.

Step 3. Take out the flight j from the initial schedule (Remove j from $\bar{J}$).

Step 4. Recalculate the normalized total weighted start time and update total objective function value

Note that the removed flight is excluded from the calculation of the objective function value. To give an example of how Do-Nothing works, assume that an initial schedule is given in Figure 20, and flights #4 and #20 are cancelled. After the first three steps, the set of flights potential to apply Do-Nothing strategy is determined as {5,7,8,9,10,16,17,18}. Consequently, after the cancellation disruption, the final schedule and the final start times are given in Figure 23 and Figure 24 respectively.

Runway #1	13	12	1		5	18	17	16				
Runway #2	11	3	2	6	14	19	15		10	9	8	7

Figure 23. Updated Schedule After Do-Nothing Algorithm

Runway #1	121	156	216		329	588	668	703				
Runway #2	71	169	199	282	328	388	453		536	573	643	678

Figure 24. Start Times After Do-Nothing Algorithm

5.5.4 Left-Shift Algorithm

Similar to Do-Nothing, Left-Shift strategy is applied when flight cancellation disruption occurs. It is a partial repair algorithm in which minor modifications are made to the particular runway after the disruptions. Left-Shift algorithm keeps the initial runway assignments, and the flights sequence on each runway. After taking out a flight from its position on a runway, the sequence of flight remains unchanged, all remaining flights assigned to that runway are shifted earlier (i.e., preponed), and the start times are updated. In general, it is expected that the value of the updated start times be smaller than the initial start times. However, due to the existence of sequence-dependent separation time constraint, this may not be the case all the time. The rationale behind this algorithm is to satisfy conformity to the initial schedule while minimizing the weighted start times after flights are canceled. The procedure for the Left-Shift strategy is given as follows:

Left-Shift Algorithm

Step 1. Get initial schedule (schedule before any disruption).

Step 2. Get the canceled flight j ($\forall j \in E$).

Step 3. Determine the position c and the runway i of the canceled flight j.

Step 3. Take out the flight j from the initial schedule (Remove j from $\bar{J}$).

Step 4. For the flights that are scheduled on runway i after the canceled flight j.

Step 5. Update the position as the initial position -1.

Step 6. Update the start time according to the Equations (1) - (4).

Step 7. Repeat until E={}.

Step 8. Calculate the combined objective function value (total weighted start time deviation, total weighted runway deviation and total weighted start time).

The earlier example can be used to illustrate the Left-Shift Algorithm. Suppose that the initial schedule is given in Figure 20, and flights #4 and #20 are cancelled. After the first three steps, the set of flights to apply Left-Shift algorithm is determined as {5,7,8,9,10,16,17,18}. The final schedule and the final start times are given in Figure 25 and Figure 26 respectively. Due to the nature of sequence-dependent separation time, the start times of the flights #5, #10, #9,# 8,#7 decrease because of the left-shift; however, the values of the flights #18, #17, #16 are not affected.

Runway #1	13	12	1	5	18	17	16				
Runway #2	11	3	2	6	14	19	15	10	9	8	7

Figure 25. Updated Schedule After Left-Shift Algorithm

Runway #1	121	156	216	276	588	668	703				
Runway #2	71	169	199	282	328	388	453	533	568	621	656

Figure 26. Start Times After Left-Shift Algorithm

5.5.5 RepairBySlack Algorithm

RepairBySlack is a partial repair algorithm that is applied to a runway in which a flight is delayed. If there is one delayed flight, the repair algorithm is considered only for the particular runway. If there are more than one delayed flights and they are scheduled on different runways in the initial schedule, the runway assignments will be same for the initial and final schedule after the disruption. The set of flights that are considered for repair consists of the delayed flights and the flights which are scheduled after the delayed flight (i.e., whose start times are greater than the delayed flight(s)) on the same runway as the delayed flights. Regarding stability, the algorithm attempts to preserve the original sequences and the initial start times on each runway. At time t, the algorithm assigns the flight with the minimum start time slack value $(\bar{t}_j - t)$ first. In other words, the prioritization among the flights is set by getting closer to the flights' initial start times. The procedure for RepairBySlack algorithm is given below.

RepairBySlack Algorithm

Step 1. Get initial schedule (schedule after cancelation).

Step 2. Get the delayed flight j ($\forall j \in D$).

Step 3. Determine the position p and the runway i of the delayed flight j.

Step 4 For the delayed flight j and flights that are scheduled on runway i after the delayed flight j.

Step 5. Set current time t as the start time of the delayed flight in the current schedule.

Step 6. While the deadline constraint satisfies $t<D_j$.

Step 7. Calculate the $(\bar{t}_j - t)$.

Step 8. Find $j = \{j \in J\text{:}\ \min_j \{(\bar{t}_j - t)\}$ and assign aircraft j to the runway i.

Step 9. Update the start time according to the Equations (1) - (4).

Step 10. Repeat until $D=\{\}$.

Step 11. Calculate the combined objective function value (total weighted start time deviation, total weighted runway deviation and total weighted start time).

In addition to the cancellation of flights # 4 and #20, flight # 19 is delayed in this example. Since the disruptions are treated sequentially, the current schedule that we should consider is not the initial schedule before any disruption; instead the schedule that provides the minimum objective function value among the candidate responses after the cancellation. Suppose that, at the first stage, *Do-Nothing* algorithm gives better normalized total objective function value than *Left-Shift*; therefore, the schedule after the execution of *Do-Nothing* is accepted as the current schedule which is shown in Figure 27 and Figure 28. Once the set of aircraft to be scheduled are determined (shaded below), the current time is updated and the slack values are calculated. The schedule and corresponding start times obtained after the *RepairBySlack* Algorithm are provided in Figure 29 and Figure 30. It is clearly observed in this example that the algorithm generates considerably stable schedules for a given instance.

Runway #1	13	12	1	5	18	17	16				
Runway #2	11	3	2	6	14	19	15	10	9	8	7

Figure 27. Updated Schedule After Do-Nothing Algorithm (Current schedule)

Runway #1	121	156	216	329	588	668	703				
Runway #2	71	169	199	282	328	388	453	536	573	643	678

Figure 28. Updated Start Times After Do-Nothing Algorithm (Current schedule)

Runway #1	13	12	1	5	18	17	16				
Runway #2	11	3	2	6	14	19	15	10	9	8	7

Figure 29. Updated Schedule After RepairBySlack Algorithm

Runway #1	121	156	216	276	588	668	703				
Runway #2	71	169	199	282	328	473	538	599	616	643	678

Figure 30. Start times After RepairBySlack Algorithm

5.5.6 RepairByEDD Algorithm

RepairByEDD is a partial repair algorithm, which works similar to RepairBySlack, except that the algorithm sorts the flights by deadline, and assigns the flight with the earliest deadline (d_j) first. In other words, the prioritization among the flights is set by getting closer to the flights' deadlines. Runway re-assignment is not permitted. Obtaining a feasible schedule after the disruptions is one of the targets of the rescheduling problems. Since the deadline constraint affects a schedule being feasible or not, the rationale behind this algorithm is obtaining a feasible schedule after disruptions. Regarding stability, the algorithm attempts to preserve the initial schedule and start times by repairing only the flights that can potentially be affected by the disruptions. The procedure for RepairByEDD Algorithm is given below.

RepairByEDD Algorithm

Step 1. Get initial schedule (schedule after cancelation).

Step 2. Get the delayed flight j ($\forall j \in D$).

Step 3. Determine the position p and the runway i of the delayed flight j.

Step 4 For the delayed flight j and flights that are scheduled on runway i after the delayed flight j.

Step 5. While the deadline constraint satisfies $t<D_j$.

Step 6. Find $j = \{j\epsilon\ J\text{: } \min_j\ \{d_j\}$ and assign j to the runway i.

Step 7. Update the start time according to the Equations (1) - (4).

Step 8. Repeat Step until $D=\{\}$.

Step 9. Calculate the combined objective function value (total weighted start time deviation, total weighted runway deviation and total weighted start time).

Similar to the previous case, again suppose that, the schedule after *Do-Nothing* is accepted as the current schedule as shown in Figure 23 and Figure 24. Once the set of aircraft to be scheduled are determined (shaded below), they are rescheduled according to their earliest deadlines. The schedule and corresponding start times obtained after running RepairByEDD Algorithm are provided in Figure 31 and Figure 32.

Runway #1	13	12	1	5	18	17	16				
Runway #2	11	3	2	6	14	15	19	10	9	8	7

Figure 31. Updated Schedule After RepairByEDD Algorithm

Runway #1	121	156	216	276	588	668	703				
Runway #2	71	169	199	282	328	440	473	566	603	643	678

Figure 32. Start Times After RepairByEDD Algorithm

5.5.7 InsertDelayed Algorithm

InsertDelayed is a partial and right shift repair algorithm that is considered only for the particular runway to which the delayed flight is initially assigned. The algorithm tries all flight insertion alternatives for the delayed flight(s) on the same runway to find the best insertion with the minimum value of the normalized objective function. Therefore, the InsertDelayed emphasizes both efficiency and stability. After inserting the flight into a position on a runway, start times of all the remaining flights assigned to that runway are shifted if necessary. InsertDelayed attempts to keep the sequence of the aircraft on the corresponding runway unchanged as much as possible. If there are more than one delayed flights on the same runway, the insertion starts with the flight that is positioned earlier. The procedure for InsertDelayed algorithm is given below.

InsertDelayed Algorithm

Step 1. Get initial schedule (schedule after cancelation).

Step 2. Get the delayed flight j ($\forall j \in D$).

Step 3. Determine the position p and the runway i of the delayed flight j.

Step 4. Determine the position p_{last} of the last flight scheduled on runway i.

Step 5. Increase the ready time, target time and deadline of the delayed flight j by the amount of delay.

Step 6. For the delayed flight j and flights that are scheduled on runway i after the delayed flight j.

Step 7. Construct an insertion set I which consists of p, $p+1$, $p+2$,..., p_{last}.

Step 8. Insert the delayed flight into place $a \in I$.

Step 9. Update the start time according to the Equations (1) - (4).

Step 10. Calculate the combined objective function value (total weighted start time deviation, total weighted runway deviation and total weighted start time).

Step 11. If the combined objective function value after the insertion into place a is better than the best objective function value so far.

Step 12. Update the best objective function value and the corresponding schedule.

Step 13. Repeat until insertion set *I*={}.

Step 14. Display the best combined objective function value and the schedule.

The earlier example can be used again to illustrate the InsertDelayed Algorithm. For a delayed flight #19, there exist 7 insertion alternatives; the existing position of the flight # 19, between flight # 15 and # 20, 20 and 10, 10 and 9, 9 and 8, 8 and 7, and after flight 7. I={14-15,15-20,20-10,10-9,9-8,8-7,7-}

Compare the combined objective function value for i=0,1,..., 6 and select the best one in terms of the objective function value. Insertion examples are given in Figures 33, 34 and 35.

Runway #1	13	12	1	4	5	18	17	16				
Runway #2	11	3	2	6	14	19	15	20	10	9	8	7

Figure 33. Existing Position of Flight #19 (i=0)

Runway #1	13	12	1	4	5	18	17	16				
Runway #2	11	3	2	6	14	15	20	19	10	9	8	7

Figure 34. Insert flight #19 between 20 and 10 (i=2)

Runway #1	13	12	1	4	5	18	17	16				
Runway #2	11	3	2	6	14	15	20	10	9	8	7	19

Figure 35 Insert flight #19 after flight 7 (i=6)

5.5.8 RepairByTWST Algorithm

RepairByTWST is a partial repair algorithm, which works similar to the proposed complete regeneration approach, TWST algorithm, and the partial repair algorithms presented earlier to treat new unexpected flights. The RepairByTWST resembles RepairBySlack and RepairByEDD algorithms in a sense that it repairs schedules of the affected flights so it does not reschedule from scratch; and it resembles TWST algorithm in a sense that the set of flights to repair are assigned to the runways by the largest $w_j/(r_j + s_{kj})$ ratio. The set of flights that are affected and considered for repair consists of the new unexpected flights and the flights whose start times are greater than the minimum ready times values of the new flights. The approach aims at preserving the initial schedule as much as possible while the total weighted start times are at minimum. Hence, the algorithm focuses on both stability and efficiency simultaneously. The procedure for RepairByTWST Algorithm is given below.

RepairByTWST Algorithm

Step 1. Get initial schedule (schedule after delay).

Step 2. Get the new unexpected flight j ($\forall j \in A$).

Step 3. For the new flight j and the flights whose start times are greater than the minimum ready times values of the new flights.

Step 4. Calculate the ratio using $\frac{w_j}{(r_j+s_{kj})}$.

Step 5. According to the ratio that is calculated in Step 4, assign the aircraft with the largest ratio to a runway on which its operation can start earlier.

Step 6. Remove flight j from set J.

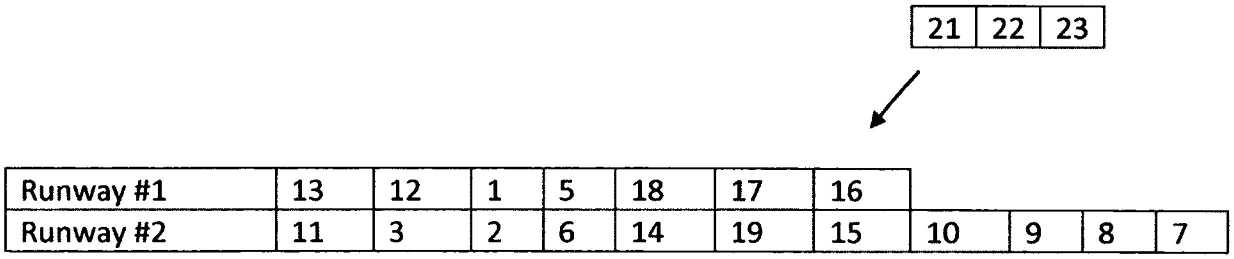
Step 7. Update the makespan of the runway where the aircraft is assigned on.

Step 8. Repeat until all aircrafts are scheduled.

Step 9. Calculate the combined objective function value (total weighted start time deviation, total weighted runway deviation and total weighted start time).

In the previous example, suppose that there are three new unexpected flights, flights # 21, #22 and #23. As the disruptions are treated sequentially, the current schedule that we should consider is not the initial schedule before any disruption; instead the schedule that provides the minimum objective function value among the candidate responses after the delay. Assume that, *RepairBySlack* strategy gives the best normalized total objective function value; therefore, the schedule after running *RepairBySlack* is accepted as the current schedule as shown in Figures 36 and 37.

Once the set of aircraft to be rescheduled are determined (shaded below), they are rescheduled according to the largest $\frac{w_j}{(r_j+s_{kj})}$ ratio. The schedule and corresponding start times obtained after the RepairByTWST Algorithm is executed are provided in Figures 38 and 39.



21	22	23

Runway #1	13	12	1	5	18	17	16				
Runway #2	11	3	2	6	14	19	15	10	9	8	7

Figure 36. Updated Schedule After RepairBySlack Algorithm (Current schedule)

Runway #1	121	156	216	276	588	668	703				
Runway #2	71	169	199	282	328	473	538	599	616	643	678

Figure 37. Start Times After RepairBySlack Algorithm (Current schedule)

Runway #1	13	12	1	5	18	21	17	22	16			
Runway #2	11	3	2	6	14	19	15	10	9	8	7	23

Figure 38. Updated Schedule After RepairByTWST Algorithm

Runway #1	121	156	216	329	588	623	668	703	763			
Runway #2	71	169	199	282	328	473	538	599	616	643	678	748

Figure 39. Start Times After RepairByTWST Algorithm

5.5.9 InsertNew Algorithm

InsertNew is a partial and right shift repair algorithm similar to the InsertDelayed algorithm. The algorithm tries flight insertion alternatives of the unexpected flights to find the best insertion with the minimum value of the normalized objective function. The set of flights that are affected and considered for repair consists of the new unexpected flights and the flights whose start times are greater than the minimum ready times values of the unexpected flights. The InsertNew emphasizes both efficiency and stability objectives. After inserting a flight into a position on a runway, start times of all remaining flights assigned to that runway are updated. If there are more than one new unexpected flight, the insertion starts with the flight whose deadline is the earliest. The procedure for InsertNew algorithm is given below.

InsertNew Algorithm

Step 1. Get initial schedule (schedule after delay).

Step 2. Get the new unexpected flight j ($\forall j \in A$).

Step 3. For the new flight j and the flights whose start times are greater than the minimum ready times values of the new flights.

Step 4. Denote the earliest position of the flight whose start times are greater than the minimum ready times values of the new flights on runway i as x_i ($\forall i \in M$).

Step 5. Denote the last position of an aircraft on runway *i* as l_i.

Step 6. Construct an insertion set *I* which consists of x_i, x_i+1, $x_i+2, \ldots, l_i$..

Step 7. Insert the delayed flight into place $a \epsilon I$.

Step 8. Update the start time according to the Equations (1) - (4).

Step 9. Calculate the combined objective function value (total weighted start time deviation, total weighted runway deviation and total weighted start time).

Step 10. If the combined objective function value after the insertion into place a is better than the best objective function value so far.

Step 11. Update the best objective function value and the corresponding schedule.

Step 12. Repeat until insertion set I={}.

Step 13. Display the best combined objective function value and the schedule.

Suppose that in addition to cancelled and delayed flights, there are 3 new unexpected flights, flights # 21, #22 and #23. Once again, suppose that the schedule after the RepairBySlack is accepted as the current schedule which is shown in Figures 36 and 37. The earlier example can be used again to illustrate the InsertNew Algorithm. For the new flight #21 for instance, there exist 6 insertion alternatives; between flight # 18 and # 17, 17 and 16, 9 and 8, 8 and 7, after flight 16, and after flight 7.

I={18-17, 17-16, 16-, 9-8, 8-7, 7-}

Compare the combined objective function value for i=0,1,..., 7 and select the one with the best objective function value. Insertion examples are given in Figure 40- Figure 45.

Runway #1	13	12	1	4	5	18	21	17	16		
Runway #2	11	3	2	6	14	19	15	10	9	8	7

Figure 40. Insert Flight #21 between Flights 18 and 17

Runway #1	13	12	1	4	5	18	17	21	16		
Runway #2	11	3	2	6	14	15	19	10	9	8	7

Figure 41. Insert Flight #21 between Flights 17 and 16

Runway #1	13	12	1	4	5	18	17	16				
Runway #2	11	3	2	6	14	15	10	9	8	7	21	19

Figure 42. Insert Flight #21 between Flights 7 and 19

Runway #1	13	12	1	4	5	18	21	22	17	16	
Runway #2	11	3	2	6	14	19	15	10	9	8	7

Figure 43. Insert Flight #22 between Flights 21 and 17

Runway #1	13	12	1	4	5	18	17	21	22	16	
Runway #2	11	3	2	6	14	15	19	10	9	8	7

Figure 44. Insert Flight #22 between Flights 21 and 16

Runway #1	13	12	1	4	5	18	17	16					
Runway #2	11	3	2	6	14	15	10	9	22	8	7	21	19

Figure 45. Insert Flight #22 between Flights 9 and 8

In this chapter, the Aircraft Reactive Scheduling Problem (ARSP) is addressed. An airline is faced with the potential deviations in the planned flight schedule because of the unexpected events. In this chapter, several solution methodologies and heuristic procedures are proposed in order to update the existing aircraft schedule dynamically. Repair and complete regeneration algorithms are developed for each type of disruptive events, specifically, flight cancellations, aircraft delays and the arrival of new aircraft.

Do-Nothing and Left-Shift are the repair strategies for the flight cancellations, RepairBySlack, RepairByEDD, InsertDelayed algorithms are proposed to repair the schedule for flight delays, and RepairByTWST and InsertNew are the repair algorithms for the arrival of new aircraft. Two complete regeneration algorithms, TWST Algorithm and SA-Re Algorithm are proposed to generate schedules from scratch to treat flight cancellations, delays and unexpected arrivals simultaneously. The performance (solution quality, schedule stability and computational time) of the various algorithms is compared to the optimal solutions and to each other in Chapter 6.

CHAPTER 6

COMPUTATIONAL STUDY FOR AIRCRAFT REACTIVE SCHEDULING ALGORITHMS

The computational study for the aircraft reactive scheduling algorithms presented here aims to determine the performance of the proposed solution approaches for a wide range of problem sizes. Solutions are compared to the optimal solutions obtained by mixed-integer linear programming model discussed earlier. The solution quality and stability of the reactive scheduling algorithms are evaluated for problems with a number of aircraft $n = 15, 20, 25$ and number of runways $m = 2, 3, 4, 5$. For each combination of n and m, 5 instances were generated totaling 55 problem instances.

6.1 Data Generation

Similar to Section 4.1, each aircraft is characterized by its operation type (i.e, arrival or departure), weight-class (i.e, heavy, medium or light), priority (aircraft tardiness penalty), ready time, target time, deadline, and separation times. In addition to these, a feasible initial aircraft schedule before any disruption, set of delayed, unexpected new flights and cancelled flights, and their corresponding parameter values are generated as follows:

1. Aircraft operation types were randomly generated as 0 or 1 to represent an arrival and departure respectively.

2. Aircraft weight classes were randomly generated as 1, 2, 3 to represent heavy, medium or light aircraft respectively.

3. The aircraft tardiness penalty (priority) w_j varies between 1 and 6 and was introduced as a function of the aircraft weight class and its operation type, where the least weight of 1 was assigned to small departures and the greatest weight of 6 was given to heavy arrivals.

4. The ready-times r_j were randomly generated using a discrete uniform distribution over the interval $(0, \gamma \frac{n}{m})$, where γ is a parameter that was randomly selected between 30 and 90.

5. Every aircraft was prescribed a time-window of 600 seconds. Therefore, deadlines d_j were calculated by $r_j + 600$.

6. Target times δ_j were calculated by $r_j + 60$.

7. As presented in Sherali et al. (2010) and shown in Table 1, the minimum separation times s_{kj} are given and range between 30 and 200 seconds depending on aircraft type (small, large or heavy), and the type of operation (landing vs. departure) that the actual values are enforced by aviation authorities.

8. The aircraft penalty cost of deviation from the initial start time $\alpha_j, \forall j \in \bar{J} - (D \cup E)$ were randomly generated between 1 and 5.

9. The aircraft penalty cost of deviation from the initial runway assignment $\beta_j, \forall j \in \bar{J} - (D \cup E)$ were randomly generated between 5 and 10.

10. The number of cancelled flights is randomly generated between 5% and 10 % of the number of aircraft, and the specific cancelled flights is/are randomly generated.

11. The number of delayed flights is randomly generated between 11% and 40% of the number of aircraft, and the specific delayed aircraft is/are randomly generated.

12. The amount of delay is randomly generated as the 50% of the difference between maximum and minimum ready time values of the delayed flight(s).

13. The number of new unexpected flights is randomly generated between 5% and 15% of the number of aircraft, and the specific new unexpected aircraft is/are randomly generated.

6.2 Effectiveness of the Algorithms

The proposed algorithms were implemented in C and run on an Intel Core 2 Duo 2.10 GHz CPU with 4.00 GB of RAM laptop. The optimal solutions are obtained by MILP formulations which were coded in AMPL and solved using CPLEX 12.4 on Intel Core i7-2600 CPU with 3.40 GHz and 8 GB RAM desktop.

The performances of the reactive scheduling algorithms are measured by computing the error between the normalized combined objective function value of the algorithm and the normalized optimal solution value. The error is calculated via Equation (36) for each test problem.

$$Error = f_{ALG} - f_{Optimal} \quad (36)$$

where f_{ALG} is the normalized combined objective function value of the proposed repair and rescheduling algorithms and $f_{Optimal}$ is the normalized combined objective function value of the optimal solution for a test problem. Optimal solutions were obtained using the MILP formulation presented in Section 5.4.

Unlike most of literature that concentrates on one type of disruption at a time, in this dissertation, various types of disruptions with multiple disruptive events are considered concurrently. Therefore, as stated in Section 5.5, the sequential evaluation methodology is developed to consider the disruptions, and revise the schedules periodically. In a period, first, the response strategies executed for flight cancellation are evaluated in terms of normalized combined objective function value. Once the best strategy for each objective weight coefficient level is determined, the current schedule is updated accordingly. Then, response strategies for flight delay are compared, and the algorithm that provides the best objective function value for each objective weight coefficient combination is decided. Finally, the response strategies proposed to repair or reschedule the aircraft sequence after a disruption called new unexpected flight are evaluated.

6.2.1 Effectiveness of the Repair Algorithms for Flight Cancellation

Fifty five different unique problem instances with the initial schedule before any disruptions of cancellation, delay or new flights were generated. These instances are solved with 13 different scenarios of objective weight coefficient levels. Reactive scheduling strategies to repair flight cancelations are evaluated under different objective weight coefficient levels (π_1, π_2, π_3). Recall that π_1 is the coefficient weight factor for the total weighted start time deviation (TWSD), π_2 is the coefficient weight factor for the total weighted runway deviation (TWRD), and π_3 is the coefficient weight factor for the total weighted start time (TWS). π_1 and π_2 reflect the importance of schedule stability, while $\pi_3$relects its quality. Average errors for the reactive scheduling strategies and average CPU times are obtained by averaging the values for the instances of each aircraft-runway combination. The performance values of the Do-Nothing and Left-Shift repair strategies are compared. A sample experiment result is provided for the Do-Nothing algorithm for flight cancelation with $\pi_1 = 0.75, \pi_2 = 0, \pi_3 = 0.25$ in Table 18 in order to simplify the comparison results in Table 19.

Instance	Optimal				Do-Nothing				Error
	TWSD	TWRD	TWS	$f_{optimal}$	TWSD	TWRD	TWS	$f_{Do\text{-}Nothing}$	
1	0	13	11430	0.016	0	0	11430	0.016	0.000
2	375	107	23799	0.158	0	0	25046	0.165	0.007
3	794	121	47564	0.237	0	0	54169	0.314	0.076
4	192	104	94083	0.251	0	0	94604	0.250	0.000
5	178	81	11037	0.022	0	0	11430	0.032	0.010
								Average	0.0187

Table 18. A sample experiment result for Do-Nothing algorithm

Table 18 illustrates how the average error is calculated for a given instance. For the given five instances, the total weighted start time deviation, total weighted runway deviation, total weighted start time values and the normalized combined objective function value of the Do-Nothing strategy is compared to the values of the normalize objective function value of the optimal solution.

Average error of the normalized objective function of each response strategy for 55 unique problem instances are evaluated for 13 different combinations of (π_1, π_2, π_3) in Table 19.

π_1	0	0	0	0	0	0.25	0.25	0.33	0.5	0.5	0.75	0.75	1
π_2	0	0.25	0.5	0.75	1	0	0.75	0.33	0	0.5	0	0.25	0
π_3	1	0.75	0.5	0.25	0	0.75	0	0.33	0.5	0	0.25	0	0
Do-Nothing	0.745	0.512	0.334	0.167	**0.000**	0.400	**0.000**	0.068	0.124	**0.000**	**0.020**	**0.000**	**0.000**
Left-Shift	**0.630**	**0.426**	**0.277**	**0.139**	**0.000**	**0.335**	0.021	**0.058**	**0.109**	0.042	0.055	0.063	0.085

Table 19. Average error of the algorithms for flight cancellations

According to Table 19, for a specific weight coefficient value, the cases where an algorithm provides the smallest average error are highlighted. For instance, when $(\pi_1 = 0, \pi_2 = 0, \pi_3 = 1)$, Left-Shift algorithm has better mean performance than Do-Nothing algorithm. On the other hand, when the solution quality of the schedule is more important than the conformity to the original schedule, Do-Nothing algorithm provides higher average error than Left-Shift algorithm. When the weight coefficient value of all objective function components are equally important for a decision maker, $(\pi_1 = 0.33, \pi_2 = 0.33, \pi_3 = 0.33)$, applying the Left-Shift algorithm provides lower average error. Left-Shift algorithm is still preferable for the cases where both weight coefficient value of total weighted start time and weight coefficient value of either total weighted start time deviation or total weighted runway deviation are equal $((\pi_1 = 0.5, \pi_2 = 0, \pi_3 = 0.5), (\pi_1 = 0, \pi_2 = 0.5, \pi_3 = 0.5))$. Even though it is observed that the number of cases that the Left-Shift algorithm provides better objective function value for different weight coefficient value, Do-Nothing algorithm could find solutions that reach the optimal solutions.

According to the Figure 46, of Anderson-Darling normality test, the average error data is normally distributed.

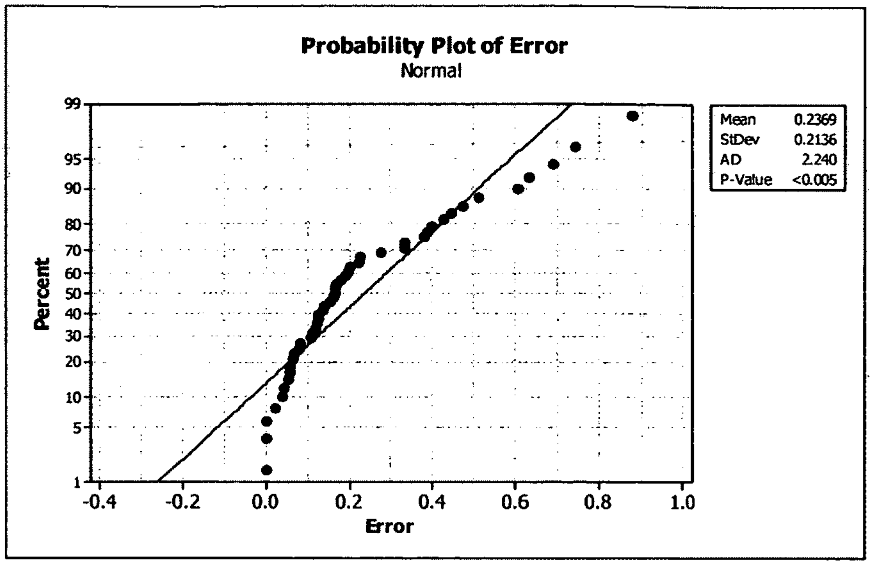


Figure 46. Normality test for average error

The performances of the response strategies are analyzed statistically by t-test using the statistical software program, Minitab 15.1. The statistical analysis is conducted under two circumstances; schedule stability and solution quality. Therefore, the analysis are focused on the weight coefficient values π_1, π_2, π_3 where schedule stability is represented by π_1, π_2 and solution quality by π_3. The runway deviation is not considered in the response strategies for flight cancellations; so π_2 is not the leading element in this analysis.

i) Note the different combinations of weight coefficients which satisfy $\pi_1 > \pi_3$. There are 5 combinations where $\pi_1 > \pi_3$ which are $(\pi_1 = 0.25, \pi_2 = 0.75, \pi_3 = 0), (\pi_1 = 0.5, \pi_2 = 0.5, \pi_3 = 0), (\pi_1 = 0.75, \pi_2 = 0, \pi_3 = 0.25), (\pi_1 = 0.75, \pi_2 = 0.25, \pi_3 = 0), (\pi_1 = 1, \pi_2 = 0, \pi_3 = 0)$. For each of the 55 problem instances, these 5 combinations are considered totaling 275 observations.

To test whether there is a significant test between the both strategies, the following hypothesis is tested using t-test:

$$H_o: \mu_1 - \mu_2 < 0$$

$$H_1: \mu_1 - \mu_2 \geq 0$$

Where μ_1 is the mean error of Do-Nothing algorithm, μ_2 is the mean error of Left-Shift algorithm when the stability is more important.

Table 20 shows that the average error of Do-Nothing strategy is statistically significantly less (p-value=0.000<$\alpha = 0.05$) from the average error of the Left-Shift algorithm when stability is of more concerned. Since $T_{\alpha,n-1} \approx 1.65$ and $T_{statistics} = -36.51$, do not reject the Null hypothesis, $H_o$and conclude that the Do-Nothing is better to apply when schedule minimizing start time deviation from the initial schedule is more important than the minimizing start time.

Paired T-Test and CI: Do-Nothing -stability, Left-Shift -stability

```
Paired T for Do-Nothing-stability - Left-Shift-stability

                         N      Mean     StDev  SE Mean
Do-Nothing-stability   275   0.00410  0.00821  0.00049
Left-Shift-stability   275   0.05341  0.02123  0.00128
Difference             275  -0.04931  0.02240  0.00135

95% CI for mean difference: (-0.05197, -0.04665)
T-Test of mean difference = 0 (vs not = 0): T-Value = -36.51  P-Value = 0.000
```

Table 20. Paired T-Test and CI: Do-Nothing-stability Left-Shift -stability

ii) There are 8 combinations where $\pi_1 \leq \pi_3$ which are $(\pi_1 = 0, \pi_2 = 0, \pi_3 = 1), (\pi_1 = 0, \pi_2 = 0.25, \pi_3 = 0.75), (\pi_1 = 0, \pi_2 = 0.5, \pi_3 = 0.5), (\pi_1 = 0, \pi_2 = 0.75, \pi_3 = 0.25), (\pi_1 = 0, \pi_2 = 1, \pi_3 = 0), (\pi_1 = 0.25, \pi_2 = 0, \pi_3 = 0.75), (\pi_1 = 0.33, \pi_2 = 0.33, \pi_3 = 0.33), (\pi_1 = 0.5, \pi_2 = 0, \pi_3 = 0.5)$ and for each of the 55 problem instances these 8 combinations are considered totaling 440 observations.

To test whether there is a significant test between the both strategies, the following hypothesis is tested using t-test:

$$H_o: \mu_1 - \mu_2 < 0$$

$$H_1: \mu_1 - \mu_2 \geq 0$$

Where $\mu_1$is the mean error of Do-Nothing algorithm, μ_2 is the mean error of Left-Shift algorithm when the stability is more important.

Table 21 shows that the average error of Left-Shift strategy is statistically significantly less (p-value=0.000<$\alpha = 0.05$) from the average error of the Do-Nothing algorithm when solution quality is of more concerned. Since $T_{\alpha,n-1} \approx 1.65$ and $T_{statistics}26.02$, reject the Null hypothesis, $H_0$and conclude that the Left-Shift is better to apply when schedule minimizing start time is more important than the minimizing start time deviation.

```
Paired T-Test and CI: Do-Nothing-quality, Left-Shift-quality

Paired T for Do-Nothing-quality - Left-Shift-quality

                      N     Mean    StDev  SE Mean
Do-Nothing-quality  440   0.2938   0.2363   0.0113
Left-Shift-quality  440   0.2468   0.1987   0.0095
Difference          440  0.04692  0.03782  0.00180

95% CI for mean difference: (0.04337, 0.05046)
T-Test of mean difference = 0 (vs not = 0): T-Value = 26.02  P-Value = 0.000
```

Table 21. Paired T-Test and CI: Do-Nothing-quality Left-Shift -quality

According to the deviation from the optimal normalized objective function information, the statistical analysis for the performance of the repair algorithms for flight cancellation is summarized in Figure 47.

Flight Cancellation								
Schedule Stability is more important					Solution Quality is more important			
$\pi_1 > \pi_3$					$\pi_1 \leq \pi_3$			
Which Has Better Mean Error Performance ?					Which Has Better Mean Error Performance ?			
	Do-Nothing	Left-Shift				Do-Nothing	Left-Shift	
Do-Nothing vs Left-Shift	✓				Do-Nothing vs Left-Shift		✓	

Figure 47. Summary of the statistical tests for flight cancellation

Table 22 provides the average CPU times of the Do-Nothing and Left Shift algorithms for 55 different unique problem instances for flight cancellations. The CPU times of the Do-Nothing and Left-Shift are indifferent and less than 0.001 second. The average CPU time does not change significantly with the change in π_1, π_2, and π_3.

π_1	0	0	0	0	0	0.25	0.25	0.33	0.5	0.5	0.75	0.75	1
π_2	0	0.25	0.5	0.75	1	0	0.75	0.33	0	0.5	0	0.25	0
π_3	1	0.75	0.5	0.25	0	0.75	0	0.33	0.5	0	0.25	0	0
Do-Nothing	0.001	0.001	0.000	0.000	0.000	0.001	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Left-Shift	0.001	0.001	0.000	0.000	0.000	0.001	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Table 22. Average CPU time of the algorithms for flight cancelations

6.2.2 Effectiveness of the Repair Algorithms for Flight Delay

Due to the sequential evaluation methodology developed to treat disruptions, once the performance analysis of the repair algorithms for flight cancellations are conducted and the revised schedule is updated as the current schedule, the performance of the repair algorithms for flight delays are evaluated next. The proposed algorithms are tested under various flight delays and objective weight coefficient levels. Average errors for the reactive scheduling strategies and average CPU times are obtained by averaging the values for 55 instances for each aircraft-runway combination. The performance values of the RepairBySlack, RepairByEDD, InsertDelayed repair strategies are compared.

6.2.2.1 Effectiveness of the Repair Algorithms for Flight Delays after Left-Shift is Applied for Flight Cancellation

Table 23 provides the average error values of the response strategies for different unique problem instances for flight delays given that Left-Shift algorithm is applied to update the schedule after being disrupted by flight cancellation.

According to Table 23, for a specific weight coefficient value, the cases where an algorithm provides the smallest average error are highlighted. For instance, when $(\pi_1 = 0, \pi_2 = 0, \pi_3 = 1)$, RepairByEDD algorithm has better average error performance than RepairBySlack and InsertDelayed algorithms.

π_1	0	0	0	0	0	0.25	0.25	0.33	0.5	0.5	0.75	0.75	1
π_2	0	0.25	0.5	0.75	1	0	0.75	0.33	0	0.5	0	0.25	0
π_3	1	0.75	0.5	0.25	0	0.75	0.00	0.33	0.5	0	0.25	0	0
RepairByEDD	**0.654**	**0.441**	**0.277**	**0.144**	**0.000**	**0.537**	0.056	**0.173**	**0.299**	0.112	0.192	0.168	0.224
RepairBySlack	0.878	0.608	0.389	0.199	**0.000**	0.688	**0.040**	0.226	0.379	**0.080**	0.200	**0.121**	**0.161**
InsertDelayed	0.745	0.509	0.322	0.166	**0.000**	0.591	0.042	0.184	0.317	0.084	**0.172**	0.126	0.168

Table 23. Average error of the algorithms for flight delays if the Left-Shift algorithm is applied for flight cancellations

When optimizing the solution quality is the primary objective of interest rather than minimizing the instability ($\pi_1 = 0, \pi_2 = 0, \pi_3 = 1$), it is more reasonable to apply RepairByEDD algorithm that generates smaller average error. On the other hand, when the stability of the schedule is more important, RepairBySlack algorithm provides smallest average error. Out of thirteen weight coefficient scenarios, InsertDelayed algorithm outperformed RepairByEDD and RepairBySlack algorithms once in terms of average error, when the coefficient value of the total weighted start time is three times more important than the coefficient value of the total weighted start time ($\pi_1 = 0.75, \pi_2 = 0, \pi_3 = 0.25$). The average error performance of RepairByEDD, RepairBySlack, InsertDelayed algorithms are same when ($\pi_1 = 0, \pi_2 = 1, \pi_3 = 0$). When all the objective function components are equally important for a decision maker, applying RepairByEDD algorithm provides the best average error value.

To further analyze the performances of the repair algorithms, they are analyzed statistically by t-test, again under two circumstances; schedule stability and solution quality. Similar to the repair algorithms for the flight cancellations, the runway deviation is not allowed in the design of response strategies for flight delays; therefore, $\pi_1$and π_3 are the foremost factors in this analysis.

When the Schedule Stability is More Important

i) The different combinations of weight coefficients which satisfy $\pi_1 > \pi_3$ are considered.

Table 24 shows that there is a statistical difference (p-value=0.000<$\alpha = 0.05$) between the population mean values of performance in terms of average error for the algorithms. RepairBySlack has better mean performance than RepairByEDD when the stability is the more important, and when the Left-Shift is applied for cancellation.

Paired T-Test and CI: left shift-RepairEDD-sta, left shift-RepairBySl-st

```
Paired T for left shift-RepairEDD-stability - left shift-RepairBySl-
stability
                           N      Mean     StDev   SE Mean
left shift-RepairEDD-sta  275   0.15063   0.05991   0.00361
left shift-RepairBySl-st  275   0.12033   0.05665   0.00342
Difference                275   0.03030   0.02497   0.00151

95% CI for mean difference: (0.02733, 0.03326)
T-Test of mean difference = 0 (vs not = 0): T-Value = 20.12  P-Value =
0.000
```

Table 24. Paired T-Test on average error of the RepairByEDD and RepairBySlack if the Left-Shift is applied for cancellation-Schedule Stability

Table 25 shows that there is also a statistical difference (p-value=0.000<$\alpha = 0.05$) between the population mean values of performance in terms of average error for the algorithms, and RepairByEDD has worse performance than InsertDelayed when the stability is more important, when the Left-Shift is applied for cancellation.

Paired T-Test and CI: left shift-RepairEDD-sta, left shift-InsertDel-sta

```
Paired T for left shift-RepairEDD-stability - left shift-InsertDel-
stability

                           N       Mean      StDev    SE Mean
left shift-RepairEDD-sta  275    0.15063    0.05991    0.00361
left shift-InsertDel-sta  275    0.11860    0.05002    0.00302
Difference                275   0.032027   0.015417   0.000930

95% CI for mean difference: (0.030197, 0.033858)
T-Test of mean difference = 0 (vs not = 0): T-Value = 34.45  P-Value =
0.000
```

Table 25. Paired T-Test on average error of the RepairByEDD and InsertDelay if the Left-Shift is applied for cancellation-Schedule Stability

Table 26 shows that there is a statistical difference (p-value=0.029<$\alpha = 0.05$) between the population mean values of performance in terms of average error for the algorithms, and InsertDelayed has better performance than RepairBySlack when the stability is the more important, when the Left-Shift is applied for cancellation.

Paired T-Test and CI: left shift-RepairBySl-st, left shift-InsertDel-sta

```
Paired T for left shift-RepairBySl-stability - left shift-InsertDel-
stability

                           N      Mean     StDev   SE Mean
left shift-RepairBySl-st  275   0.12033   0.05665   0.00342
left shift-InsertDel-sta  275   0.11860   0.05002   0.00302
Difference                275  0.001732  0.013058  0.000787

95% CI for mean difference: (0.000182, 0.003282)
T-Test of mean difference = 0 (vs not = 0): T-Value = 2.20  P-Value =
0.029
```

Table 26. Paired T-Test on average error of the RepairBySlack and InsertDelay if the Left-Shift is applied for cancellation-Schedule Stability

When the Solution Quality is more important

ii) Considering the different combinations of weight coefficients which satisfy $\pi_1 \leq \pi_3$.

Table 27 shows that there is a statistical difference (p-value=0.000<$\alpha = 0.05$) between the population mean values of performance in terms of average error for the algorithms. RepairByEDD has better mean performance than RepairBySlack, unlike the Table 24, when the quality is more important, when the Left-Shift is applied for cancellation.

Table 28 shows that it is statistically significant that (p-value=0.000<$\alpha = 0.05$), RepairByEDD has better average error performance than InsertDelayed when the stability is the more important, when the Left-Shift is applied for cancellation.

Paired T-Test and CI: left shift-RepairEDD-qua, left shift-RepairBySl-qu

```
Paired T for left shift-RepairEDD-quality - left shift-RepairBySl-
quality

                            N      Mean    StDev  SE Mean
left shift-RepairEDD-qua  440    0.3161   0.2052   0.0098
left shift-RepairBySl-qu  440    0.4220   0.2723   0.0130
Difference                440  -0.10591  0.06849  0.00326

95% CI for mean difference: (-0.11233, -0.09949)
T-Test of mean difference = 0 (vs not = 0): T-Value = -32.44  P-Value =
0.000
```

Table 27. Paired T-Test on average error of the RepairByEDD and RepairBySlack if the Left-Shift is applied for cancellation-Solution Quality

Paired T-Test and CI: left shift-RepairEDD-qua, left shift-InsertDel-qua

```
Paired T for left shift-RepairEDD-quality - left shift-InsertDel-
quality

                            N      Mean    StDev  SE Mean
left shift-RepairEDD-qua  440    0.3161   0.2052   0.0098
left shift-InsertDel-qua  440    0.3552   0.2325   0.0111
Difference                440  -0.03906  0.02928  0.00140

95% CI for mean difference: (-0.04180, -0.03631)
T-Test of mean difference = 0 (vs not = 0): T-Value = -27.98  P-Value =
0.000
```

Table 28. Paired T-Test on average error of the RepairByEDD and InsertDelay if the Left-Shift is applied for cancellation-Solution Quality

Finally, Table 29 shows that, when the Left-Shift is applied for cancellation, there is a statistical difference (p-value=0.029<$\alpha = 0.05$) between the population mean values of performance in terms of average error for the algorithms, and InsertDelayed has better performance than RepairBySlack similar to the analysis in Table 26.

According to the deviation from the optimal information, the statistical analysis for the performance of the repair algorithms for flight delays given that Left-Shift Algorithm is applied for flight cancellation is summarized through Figure 48.

```
Paired T-Test and CI: left shift-RepairBySl-qu, left shift-InsertDel-qua

Paired T for left shift-RepairBySl-quality - left shift-InsertDel-
quality

                              N      Mean     StDev  SE Mean
left shift-RepairBySl-qu    440    0.4220    0.2723   0.0130
left shift-InsertDel-qua    440    0.3552    0.2325   0.0111
Difference                  440   0.06685   0.04009  0.00191

95% CI for mean difference: (0.06310, 0.07061)
T-Test of mean difference = 0 (vs not = 0): T-Value = 34.98  P-Value =
0.000
```

Table 29. Paired T-Test on average error of the RepairBySlack and InsertDelay if the Left-Shift is applied for cancellation-Solution Quality

Flight Delay								
Given that Left-Shift is applied in Flight Cancellation								
Schedule Stability is more important					Solution Quality is more important			
$\pi_1 > \pi_3$					$\pi_1 \leq \pi_3$			
Which Has Better Mean Error Performance ?					Which Has Better Mean Error Performance ?			
	RepairByEDD	RepairBySlack	InsertDelayed			RepairByEDD	RepairBySlack	InsertDelayed
RepairByEDD vs RepairBySlack		✓			RepairByEDD vs RepairBySlack	✓		
RepairByEDD vs InsertDelayed			✓		RepairByEDD vs InsertDelayed	✓		
RepairBySlack vs InsertDelayed			✓		RepairBySlack vs InsertDelayed			✓

Figure 48. Summary of the statistical tests for flight delay – Part 1

Depending on the initially selected strategy in the flight cancellation and the decision maker's interest on solution quality and stability, the performance of the repair algorithms was evaluated. Given that Left-Shift algorithm is preferred to repair flight cancellation disruptions, when the solution quality is more important, RepairByEDD algorithm outperforms the RepairBySlack and InsertDelayed. Conversely, InsertDelayed can be preferred when the schedule stability has a higher importance.

6.2.2.2 Effectiveness of the Repair Algorithms for Flight Delays Given that Do-Nothing is Applied for Flight Cancellation

Table 30 provides the average error of the response strategies for 55 different unique problem instances for flight delays; given that Do-Nothing algorithm is applied to update the schedule after being disrupted by flight cancellation.

π_1	0	0	0	0	0	0.25	0.25	0.33	0.5	0.5	0.75	0.75	1
π_2	0	0.25	0.5	0.75	1	0	0.75	0.33	0	0.5	0	0.25	0
π_3	1	0.75	0.5	0.25	0	0.75	0.00	0.33	0.5	0	0.25	0	0
RepairByEDD	**0.693**	**0.520**	**0.347**	**0.173**	**0.000**	**0.577**	0.057	0.307	0.460	0.114	0.344	0.170	0.227
RepairBySlack	0.917	0.687	0.458	0.229	**0.000**	0.703	**0.015**	0.326	0.489	**0.031**	0.275	**0.046**	**0.061**
InsertDelayed	0.784	0.588	0.392	0.196	**0.000**	0.609	0.021	**0.290**	**0.435**	0.043	**0.260**	0.064	0.086

Table 30. Average error of the algorithms for flight delays if the Do-Nothing algorithm is applied for flight cancellations

According to Table 30, for a specific weight coefficient value, the cases where an algorithm provides smaller average error are highlighted. When the solution quality is more important than the stability ($\pi_1 = 0, \pi_2 = 0, \pi_3 = 1$), RepairByEDD algorithm is the best strategy to apply. Conversely, when the stability of the schedule is more important than the solution quality, RepairBySlack algorithm provides the lowest average error. When the weight coefficient values of all three objective function components are equal ($\pi_1 = 0.33, \pi_2 = 0.33, \pi_3 = 0.33$); InsertDelayed algorithm performs better than RepairByEDD and RepairBySlack in terms of the average error unlike the performance when the Left-Shift algorithm is applied for flight cancellations. When Table 30 and Table 23 are compared, the average error values indicate that the best strategy in each coefficient weight combination is consistent except for two scenarios (($\pi_1 = 0.33, \pi_2 = 0.33, \pi_3 = 0.33$), ($\pi_1 = 0.5, \pi_2 = 0, \pi_3 = 0.5$)). For instance, when the weight coefficient value of the total weighted start time is equal to or greater than the total weighted start time deviation, the RepairByEDD is the best strategy for flight delays when both Left-Shift algorithm and Do-Nothing algorithm are applied for flight cancellations. Another inference is that, for these cases, the average error values of the RepairByEDD are smaller if the Left-Shift algorithm is applied for flight cancellations.

Similarly, when the weight coefficient value of the total weighted start time is smaller than the total weighted start time deviation, RepairBySlack is the best strategy among the provided repair algorithms for flight delays in both Left-Shift algorithm and Do-Nothing algorithm are applied for flight cancellations. Moreover, these cases, the average error values of the RepairBySlack are smaller if the Do-Nothing algorithm is applied for flight cancellations.

Similar to Section 6.2.2.1, the detailed statistical analysis are conducted for the proposed repair algorithms developed for flight delays when Do-Nothing algorithm is applied for the flight cancellations. Appendix A includes the detailed results of the paired t-tests on average error of the RepairByEDD, RepairBySlack and InsertDelay if the Do-Nothing is applied for cancellation. The analyses are categorized into two categories: schedule stability and solution quality. According to the deviation from the optimal information, the statistical analysis for the performance of the repair algorithms for flight delays given that Left-Shift Algorithm is applied for flight cancellation is summarized through Figure 49.

Flight Delay								
Given that Do-Nothing is applied in Flight Cancellation								
Schedule Stability is more important					Solution Quality is more important			
$\pi_1 > \pi_3$					$\pi_1 \leq \pi_3$			
Which Has Better Mean Error Performance ?					Which Has Better Mean Error Performance ?			
	RepairByEDD	RepairBySlack	InsertDelayed			RepairByEDD	RepairBySlack	InsertDelayed
RepairByEDD vs RepairBySlack		✓			RepairByEDD vs RepairBySlack	✓		
RepairByEDD vs InsertDelayed			✓		RepairByEDD vs InsertDelayed	✓		
RepairBySlack vs InsertDelayed		✓			RepairBySlack vs InsertDelayed			✓

Figure 49. Summary of the statistical tests for flight delay – Part 2

Depending on the initially selected strategy in the flight cancellation (i.e. Do-Nothing vs. Left-Shift) and the decision maker's interest on solution quality and stability, the performance of the repair algorithms was evaluated. Given that Do-Nothing algorithm is preferred to repair flight cancellation disruptions, when the solution quality is more important, RepairByEDD algorithm still outperforms the RepairBySlack and InsertDelayed. On the other hand, RepairBySlack can be preferred when the schedule stability has a higher importance.

6.2.3 Effectiveness of the Repair Algorithms for Arrival of New Unexpected Flight

After the performance analysis of the repair algorithms for flight delays are conducted and the revised schedule is updated as the current schedule, and the performance of the repair algorithms for new unexpected flight arrivals are evaluated next. Owing to the sequential evaluation methodology developed to treat disruptions, the current schedule is not the very initial schedule but rather the schedule after treatment for cancelation and delay. The proposed algorithms are tested under various new unexpected arrivals and objective weight coefficient levels. Average errors for the reactive scheduling strategies and average CPU times are obtained by averaging the values for 55 instances for each aircraft-runway combination. The performance values of the RepairByTWST and InsertNew repair strategies are compared.

From Section 6.2.1, we concluded that when the weight coefficient value of the total weighted start time is equal to or greater than the total weighted start time deviation, the Left-Shift algorithm is the best strategy. On the other hand, Do-Nothing is the best strategy for a decision maker who gives more importance to the total weighted start time deviation than total weighted start time.

From Section 6.2.2, we concluded that when the weight coefficient value of the total weighted start time is equal to or greater than the total weighted start time deviation, and if the Left-Shift algorithm is applied for flight cancellations, then the RepairByEDD is the best strategy among the provided repair algorithms for flight delays.

We also concluded that when the weight coefficient value of the total weighted start time deviation is greater than the total weighted start time, and if the Do-Nothing algorithm is applied for flight cancellations, then the RepairBySlack is the best strategy among the provided repair algorithms for flight delays.

Our next step is dependent on our previous decision pattern. From the information that we gathered from Sections 6.2.1 and 6.2.2, the average error of the response strategies for different unique problem instances for unexpected flights arrivals are observed in two states. The first state presents the average error of the repair algorithms RepairByTWST and InsertNew for 55 problem instances for unexpected flight arrivals given that Left-Shift algorithm is applied to update the schedule after being disrupted by flight cancellation and repaired by RepairByEDD for flight delays. The second one provides the average error of the repair algorithms RepairByTWST and InsertNew for 55 problem instances for unexpected flight arrivals given that Do-Nothing algorithm is applied to update the schedule after being disrupted by flight cancellation and repaired by RepairBySlack for flight delays. The overall results of the first state and the second state are summarized in Table 31 and Table 36 respectively.

6.2.3.1 Effectiveness of the Repair Algorithms for Arrival of New Unexpected Flight Given that Left-Shift is Applied for Flight Cancellation and repaired by RepairByEDD

π_1	0	0	0	0	0	0.25	0.25	0.33	0.5	0.5	0.75	0.75	1
π_2	0	0.25	0.5	0.75	1	0	0.75	0.33	0	0.5	0	0.25	0
π_3	1	0.75	0.5	0.25	0	0.75	0.00	0.33	0.5	0	0.25	0	0
RepairByTWST	**0.023**	**0.139**	**0.254**	**0.369**	**0.485**	**0.075**	**0.421**	**0.246**	0.162	**0.357**	0.226	**0.293**	**0.229**
InsertNew	0.034	0.208	0.382	0.556	0.730	0.098	0.620	0.351	**0.126**	0.510	**0.177**	0.400	0.290

Table 31. Average error of the algorithms for arrival of new flights with Left-Shift for cancellations and RepairByEDD for delays

According to Table 31, for the scenarios where total weighted start time is more important than the total weighted start time deviation, RepairByTWST algorithm is the best strategy to apply. Although in few scenarios, the performance of the InsertNew algorithm is superior, the overall results imply that RepairByTWST would be a better strategy to treat the unexpected flight arrivals. Different from the previous analysis for flight cancellations and flight delays, it is worth noting that the average error values of the proposed algorithms are nonzero when $\pi_1 = 0, \pi_2 = 1, \pi_3 = 0$. Unlike the repair algorithms for flight cancellations and flight delays, the repair algorithms for new unexpected flight arrivals allow runway deviation; therefore, we observe such difference.

To further test the performance of the repair algorithms, they are analyzed statistically by t-test, again under two circumstances: schedule stability and solution quality. Unlike repair algorithms for the flight cancellations and delays, the runway deviation is allowed in the response strategies to new unexpected flight; therefore, π_1, π_2 and π_3 are all considered in hypothesis testing in this analysis.

When the Schedule Stability is More Important

i) Consider the different combinations of weight coefficients which satisfy $\pi_1 > \pi_3$.

Table 32 shows that there is a significant statistical difference (p-value=0.000<$\alpha = 0.05$) between the population mean values of performance in terms of average error for the algorithms where RepairByTWST has better mean performance than InsertNew when the stability is the more important, when the Left-Shift is applied for cancellation and RepairByEDD is applied for flight delays.

Paired T-Test and CI: LS-RepairEDD-RepairTWST-, LS-RepairEDD-INSERTNEW-S

```
Paired T for LS-RepairEDD-RepairTWST-STABILI - LS-RepairEDD-INSERTNEW-
STABILIT

                               N      Mean     StDev  SE Mean
LS-RepairEDD-RepairTWST-     275   0.30501   0.07538  0.00455
LS-RepairEDD-INSERTNEW-S     275   0.39962   0.15670  0.00945
Difference                   275  -0.09461   0.08535  0.00515

95% CI for mean difference: (-0.10474, -0.08448)
T-Test of mean difference = 0 (vs not = 0): T-Value = -18.38  P-Value =
0.000
```

Table 32. Paired T-Test on average error of the RepairByTWST and InsertNew with Left-Shift for cancellation and RepairByEDD for delays-Schedule Stability 1

ii) Observe the different combinations of weight coefficients which satisfy $\pi_2 > \pi_3$.

Table 33 shows that the overall result is same as the scenario (i), and when $\pi_2 > \pi_3$, RepairByTWST has better mean performance than InsertNew algorithm.

```
Paired T-Test and CI: LS-RepairEDD-RepairTWST-, LS-RepairEDD-INSERTNEW-S

Paired T for LS-RepairEDD-RepairTWST-STABI_1 - LS-RepairEDD-INSERTNEW-
STABIL_1

                             N      Mean    StDev  SE Mean
LS-RepairEDD-RepairTWST-   275   0.38493  0.06462  0.00390
LS-RepairEDD-INSERTNEW-S   275   0.56333  0.11015  0.00664
Difference                 275  -0.17841  0.04612  0.00278


95% CI for mean difference: (-0.18388, -0.17293)
T-Test of mean difference = 0 (vs not = 0): T-Value = -64.15  P-Value =
0.000
```

Table 33. Paired T-Test on average error of the RepairByTWST and InsertNew with Left-Shift for cancellation and RepairByEDD for flight delays-Schedule Stability 2

When the Solution Quality is more important

i) Consider the different combinations of weight coefficients which satisfy $\pi_1 \leq \pi_3$.

Table 34 shows that there is a significant statistical difference (p-value=0.000<α = 0.05) between the population mean values of performance in terms of average error for the algorithms where RepairByTWST has better mean performance than InsertNew if the solution quality is the more important, when the Left-Shift is applied for cancellation and RepairByEDD is applied for flight delays.

```
Paired T-Test and CI: LS-RepairEDD-RepairTWST-, LS-RepairEDD-INSERTNEW-Q

Paired T for LS-RepairEDD-RepairTWST-QUALITY - LS-RepairEDD-INSERTNEW-
QUALITY

                             N      Mean    StDev  SE Mean
LS-RepairEDD-RepairTWST-   440    0.2199   0.1445   0.0069
LS-RepairEDD-INSERTNEW-Q   440    0.3136   0.2272   0.0108
Difference                 440  -0.09367  0.08747  0.00417

95% CI for mean difference: (-0.10186, -0.08547)
T-Test of mean difference = 0 (vs not = 0): T-Value = -22.46  P-Value =
0.000
```

Table 34. Paired T-Test on average error of the RepairByTWST and InsertNew with Left-Shift for cancellation and RepairByEDD for flight delays -Solution Quality 1

ii) Observe the different combinations of weight coefficients which satisfy $\pi_2 \leq \pi_3$.

Table 35 shows that there is a significant statistical difference (p-value=0.000<$\alpha = 0.05$) between the population mean values of performance in terms of average error for the algorithms. For $\pi_2 \leq \pi_3$, RepairByTWST has better mean performance than InsertNew when the solution quality is the more important, when the Left-Shift is applied for cancellation and RepairByEDD is applied for flight delays.

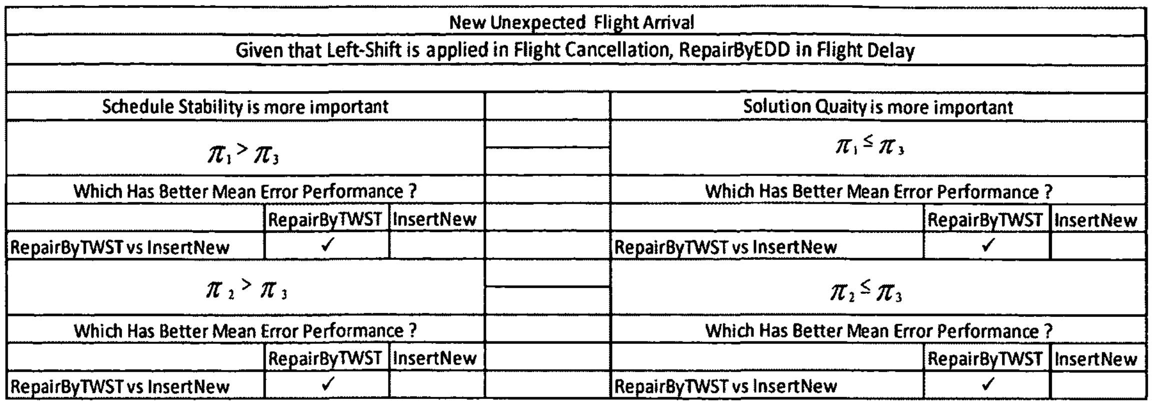
Paired T-Test and CI: LS-RepairEDD-RepairTWST-, LS-RepairEDD-INSERTNEW-Q

```
Paired T for LS-RepairEDD-RepairTWST-QUALI_1 - LS-RepairEDD-INSERTNEW-
QUALIT_1

                           N      Mean     StDev  SE Mean
LS-RepairEDD-RepairTWST-  440   0.16803  0.08028  0.00383
LS-RepairEDD-INSERTNEW-Q  440   0.20717  0.11673  0.00556
Difference                440  -0.03914  0.05980  0.00285

95% CI for mean difference: (-0.04474, -0.03354)
T-Test of mean difference = 0 (vs not = 0): T-Value = -13.73  P-Value =
0.000
```

Table 35. Paired T-Test on average error of the RepairByTWST and InsertNew with Left-Shift for cancellation and RepairByEDD for flight delays -Solution Quality 2

According to the deviation from the optimal information, the statistical analysis for the performance of the repair algorithms for flight delays given that Left-Shift Algorithm is applied for flight cancellation and RepairByEDD is applied for flight delays is summarized in Figure 50.

In conclusion, depending on the initially selected strategy in the flight cancellation (Left-Shift) and in the flight delay (RepairByEDD), RepairByTWST outperforms the InsertNew regardless whether the emphasis was on solution quality or schedule stability.



New Unexpected Flight Arrival						
Given that Left-Shift is applied in Flight Cancellation, RepairByEDD in Flight Delay						
Schedule Stability is more important				Solution Quaity is more important		
$\pi_1 > \pi_3$				$\pi_1 \leq \pi_3$		
Which Has Better Mean Error Performance ?				Which Has Better Mean Error Performance ?		
	RepairByTWST	InsertNew			RepairByTWST	InsertNew
RepairByTWST vs InsertNew	✓			RepairByTWST vs InsertNew	✓	
$\pi_2 > \pi_3$				$\pi_2 \leq \pi_3$		
Which Has Better Mean Error Performance ?				Which Has Better Mean Error Performance ?		
	RepairByTWST	InsertNew			RepairByTWST	InsertNew
RepairByTWST vs InsertNew	✓			RepairByTWST vs InsertNew	✓	

Figure 50. Summary of the statistical tests for new flight arrival – Part 1

6.2.3.2 Effectiveness of the Repair Algorithms for Arrival of New Unexpected Flight Given that Do-Nothing is Applied for Flight Cancellation then RepairBySlack is applied for flight delays

π_1	0	0	0	0	0	0.25	0.25	0.33	0.5	0.5	0.75	0.75	1
π_2	0	0.25	0.5	0.75	1	0	0.75	0.33	0	0.5	0	0.25	0
π_3	1	0.75	0.5	0.25	0	0.75	0.00	0.33	0.5	0	0.25	0	0
RepairByTWST	**0.141**	**0.258**	**0.367**	**0.426**	**0.485**	**0.157**	0.506	**0.283**	**0.172**	0.405	0.177	0.304	0.203
InsertNew	0.248	0.308	0.374	0.491	0.607	0.215	**0.393**	0.317	0.182	**0.300**	**0.149**	**0.208**	**0.116**

Table 36. Average error of the algorithms for arrival of new unexpected flights with the Do-Nothing algorithm for flight cancellations then the RepairBySlack for flight delays

According to Table 36, when the weight coefficient value of the total weighted start time is equal to or greater than the total weighted start time deviation, the RepairByTWST outperforms the InsertNew algorithm. Whereas when the stability is more important, InsertNew algorithm performs better than RepairByTWST when $(\pi_1 = 0.75, \pi_2 = 0, \pi_3 = 0.25)$, $(\pi_1 = 0.5, \pi_2 = 0.5, \pi_3 = 0)$, $(\pi_1 = 0.25, \pi_2 = 0.75, \pi_3 = 0)$, $(\pi_1 = 0.75, \pi_2 = 0.25, \pi_3 = 0)$, $(\pi_1 = 1, \pi_2 = 0, \pi_3 = 0)$.

When $\pi_1 < \pi_3$, the average error values of the RepairByTWST are smaller in Table 36 compared to Table 31. Whereas, when $\pi_1 \geq \pi_3$, the average error values of the InsertNew are smaller in Table 36 compared to Table 31.

Similar to Section 6.2.3.1, the detailed the statistical analysis are conducted for the proposed repair algorithms for new flight arrivals when Do-Nothing algorithm is applied for the flight cancellations and the RepairBySlack is applied for flight delays. The detailed results of the paired t-tests on average error of the RepairByTWST and InsertNew are provided in Appendix B. The analyses are categorized into two categories: schedule stability and solution quality and are summarized in Figure 51.

New Unexpected Flight Arrival						
Given that Do-Nothing is applied in Flight Cancellation, RepairSlack in Flight Delay						
Schedule Stability is more important				Solution Quaity is more important		
$\pi_1 > \pi_3$				$\pi_1 \leq \pi_3$		
Which Has Better Mean Error Performance ?				Which Has Better Mean Error Performance ?		
	RepairByTWST	InsertNew			RepairByTWST	InsertNew
RepairByTWST vs InsertNew		✓		RepairByTWST vs InsertNew	✓	
$\pi_2 > \pi_3$				$\pi_2 \leq \pi_3$		
Which Has Better Mean Error Performance ?				Which Has Better Mean Error Performance ?		
	RepairByTWST	InsertNew			RepairByTWST	InsertNew
RepairByTWST vs InsertNew		✓		RepairByTWST vs InsertNew	✓	

Figure 51. Summary of the statistical tests for new flight arrival – Part 2

Depending on the initially selected strategy in the flight cancellation (Do-Nothing) and in the flight delay (RepairBySlack), and the decision maker's interest in solution quality and stability, the performance of the repair algorithms was evaluated. When the solution quality has higher importance than schedule stability, RepairByTWST outperforms the InsertNew; otherwise, InsertNew provides lower average error.

6.2.4 Effectiveness of the Complete Regeneration Algorithms

Complete regeneration algorithms reschedule all flights from scratch that treat flight cancellations, delays and unexpected arrivals simultaneously. The reason for proposing rescheduling algorithm is that one of the components of the total objective function is total weighted start time, and the percentage contribution of this metric to the overall objective function value is the most. Therefore, it would be a considerable approach to propose an algorithm to handle TWST minimization. The proposed algorithms are tested under various flight cancellations, delays and unexpected flight arrivals, with objective weight coefficient levels. Average errors for the TWST and SA-Re algorithms and average CPU times are obtained by averaging the values for 55 instances for each aircraft-runway combination. Table 37 provides the average error of the rescheduling strategies for different unique problem instances.

π_1	0	0	0	0	0	0.25	0.25	0.33	0.5	0.5	0.75	0.75	1
π_2	0	0.25	0.5	0.75	1	0	0.75	0.33	0	0.5	0	0.25	0
π_3	1	0.75	0.5	0.25	0	0.75	0.00	0.33	0.5	0	0.25	0	0
TWST	0.166	0.142	0.126	0.226	0.141	0.156	0.474	0.446	0.475	0.452	0.181	0.211	0.126
SA-Re	**0.051**	**0.072**	**0.084**	**0.095**	**0.071**	**0.073**	**0.089**	**0.072**	**0.089**	**0.086**	**0.081**	**0.089**	**0.083**

Table 37. Average error of the complete regeneration algorithms

The results show that regardless of the π_1, π_2, π_3combination, SA-Re performs better than the TWST algorithm in terms of average error. The reason for such result is that total weighted start time component is dominant in the objective function even though it's normalized.

To further test the performances of the complete regeneration algorithms, they are analyzed statistically by t-test, again under two circumstances: schedule stability and solution quality. Unlike repair algorithms for the flight cancellations and delays, the runway deviation is allowed in the response strategies for new unexpected flight; therefore, π_1, π_2 and π_3 are all important in the hypothesis testing.

Table 38 and Table 39 show that there is a significant statistical difference (p-value=0.000<$\alpha = 0.05$) between the population mean values in terms of average error for the algorithms with SA-Re performing better than TWST when the stability is more important.

When the Schedule Stability is More Important

i) Consider the different combinations of weight coefficients which satisfy $\pi_1 > \pi_3$.

```
Paired T-Test and CI: twst-stability, sa-re-stability

Paired T for twst-stability - sa-re-stability

                     N      Mean     StDev  SE Mean
twst-stability     275   0.28880   0.14525  0.00876
sa-re-stability    275   0.08560   0.00321  0.00019
Difference         275   0.20320   0.14341  0.00865

95% CI for mean difference: (0.18617, 0.22023)
T-Test of mean difference = 0 (vs not = 0): T-Value = 23.50  P-Value =
0.000
```

Table 38. Paired T-Test on average error of the TWST and SA-Re complete regeneration algorithms - Schedule Stability 1

ii) Consider the different combinations of weight coefficients which satisfy $\pi_2 > \pi_3$.

```
Paired T-Test and CI: twst-stability_1, sa-re-stability_1

Paired T for twst-stability_1 - sa-re-stability_1

                       N      Mean     StDev  SE Mean
twst-stability_1     275   0.30080   0.13593  0.00820
sa-re-stability_1    275   0.08600   0.00806  0.00049
Difference           275   0.21480   0.13323  0.00803

95% CI for mean difference: (0.19898, 0.23062)
T-Test of mean difference = 0 (vs not = 0): T-Value = 26.74  P-Value =
0.000
```

Table 39. Paired T-Test on average error of the TWST and SA-Re complete regeneration algorithms-Schedule Stability 2

When the Solution Quality is more important

i) Consider the different combinations of weight coefficients which satisfy $\pi_1 \leq \pi_3$.

Paired T-Test and CI: twst-quality, sa-re-quality

```
Paired T for twst-quality - sa-re-quality

                  N      Mean     StDev   SE Mean
twst-quality    440   0.23045   0.13094   0.00624
sa-re-quality   440   0.07567   0.01269   0.00061
Difference      440   0.15477   0.12797   0.00610

95% CI for mean difference: (0.14278, 0.16676)
T-Test of mean difference = 0 (vs not = 0): T-Value = 25.37  P-Value =
0.000
```

Table 40. Paired T-Test on average error of the TWST and SA-Re complete regeneration algorithms-Solution Quality 1

ii) Consider the different combinations of weight coefficients which satisfy $\pi_2 \leq \pi_3$.

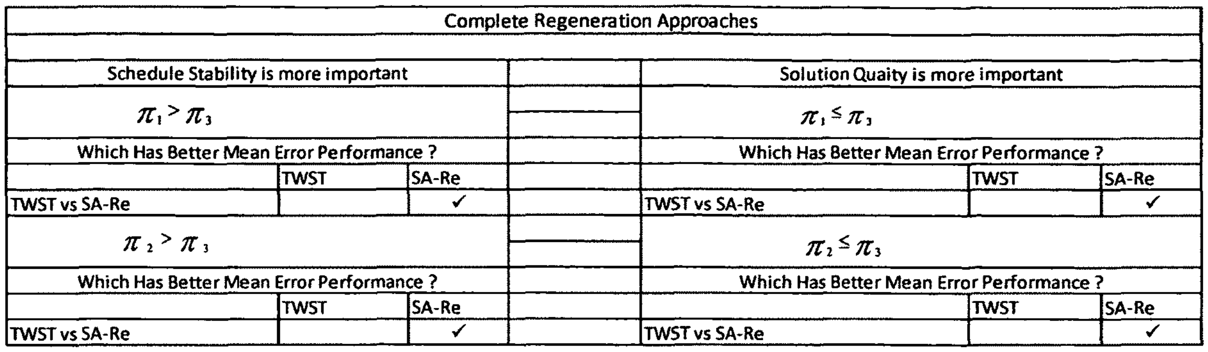
Paired T-Test and CI: twst-quality_1, sa-re-quality_1

```
Paired T for twst-quality_1 - sa-re-quality_1

                    N      Mean     StDev   SE Mean
twst-quality_1    440   0.22817   0.13634   0.00650
sa-re-quality_1   440   0.07548   0.01108   0.00053
Difference        440   0.15269   0.13424   0.00640

95% CI for mean difference: (0.14011, 0.16527)
T-Test of mean difference = 0 (vs not = 0): T-Value = 23.86  P-Value =
0.000
```

Table 41. Paired T-Test on average error of the TWST and SA-Re complete regeneration algorithms -Solution Quality 2

Table 40 and Table 41 show that there is a significant statistical difference (p-value=0.000<$\alpha = 0.05$) between the population mean values in terms of average error for the algorithms where SA-Re has better mean performance than TWST when the solution quality is the more important.

According to the deviation from the optimal information, the statistical analysis for the performance of the complete regeneration algorithms is summarized through Figure 52. It is can be concluded that either for solution quality or schedule stability, SA-Re performs better than TWST algorithm in terms of average error.



Complete Regeneration Approaches						
Schedule Stability is more important				Solution Quaity is more important		
$\pi_1 > \pi_3$				$\pi_1 \leq \pi_3$		
Which Has Better Mean Error Performance ?				Which Has Better Mean Error Performance ?		
	TWST	SA-Re			TWST	SA-Re
TWST vs SA-Re		✓		TWST vs SA-Re		✓
$\pi_2 > \pi_3$				$\pi_2 \leq \pi_3$		
Which Has Better Mean Error Performance ?				Which Has Better Mean Error Performance ?		
	TWST	SA-Re			TWST	SA-Re
TWST vs SA-Re		✓		TWST vs SA-Re		✓

Figure 52. Summary of the statistical tests for complete regeneration algorithms

Table 42 provides the average CPU times of the TWST and SA-Re algorithms for 55 different unique problem instances for flight cancellations. SA-Re requires longer CPU time than TWST. The average CPU time does not seem to change significantly with the change in π_1, π_2, and π_3.

π_1	0	0	0	0	0	0.25	0.25	0.33	0.5	0.5	0.75	0.75	1
π_2	0	0.25	0.5	0.75	1	0	0.75	0.33	0	0.5	0	0.25	0
π_3	1	0.75	0.5	0.25	0	0.75	0	0.33	0.5	0	0.25	0	0
TWST	0.048	0.046	0.047	0.047	0.047	0.047	0.047	0.047	0.048	0.047	0.048	0.047	0.047
SA-Re	21.3	22.82	21.93	21.44	21.67	21.76	21.88	21.56	22.12	22.87	22.35	21.98	21.89

Table 42. Average CPU time of the algorithms for flight cancelations

CHAPTER 7

CONCLUSIONS AND FUTURE RESEARCH

7.1 Conclusions

The dissertation addressed the Aircraft Sequencing Problem (ASP) and Aircraft Reactive Scheduling Problem (ARSP) in a realistic operational environment requiring competent solutions in a tolerable timeframe. The ASP was modeled as a parallel machine scheduling problem with unequal ready times, target times and deadlines to minimize the total weighted tardiness of aircraft landings and departures simultaneously. The problem concurrently determines the assignment of each aircraft (job) to a runway (machine), the appropriate sequence of aircraft on each runway, and their departing or landing times. The dissertation examines the ASP over multiple runways, under mixed mode operations with the sequence-dependent separation times to prevent the dangers associated with wake-vortex effects. Since ASP is NP-hard, it is necessary to develop qualified solution approaches to obtain solutions in reasonable computational times.

Three greedy algorithms, namely the Adapted Apparent Tardiness Cost with Separation and Ready Times (AATCSR), the Earliest Ready Time (ERT) and the Fast Priority Index (FPI) were developed to construct good initial solutions to the ASP. The AATCSR is an extension of the ATC rule but with considering unequal ready times, target times, deadlines and sequence-dependent separation times. The ERT is a version of the FCFS, and the FPI rule is a modification of AATCSR. Different from the AATCSR, in FPI the urgency of scheduling aircraft in FPI is treated in a linear manner rather than exponential, which makes it much faster in terms of computational time. Moreover, metaheuristics including Simulated Annealing (SA) and the Metaheuristic for Randomized Priority Search (Meta-RaPS) were introduced to improve solutions initially constructed by the proposed greedy algorithms.

The algorithms' solutions are compared to optimal solutions and their performances are evaluated in terms of solution quality and CPU time. The results show the performance of the proposed greedy algorithms is similar in terms of average CPU time. However, AATCSR outperforms both FPI and ERT with low relative deviation (error) the from optimal solutions. The performance analysis of the metaheuristics indicates that the SA algorithm is more efficient and more effective than Meta-RaPS algorithm when the same greedy algorithms are considered for their initial solution (i.e., SA_{AATCSR} is superior to Meta-$RaPS_{AATCSR}$, SA_{FPI} is superior to Meta-$RaPS_{FPI}$, SA_{ERT} is superior to Meta-$RaPS_{ERT}$). It is determined that the solution quality of SA and Meta-RaPS algorithms that use greedy algorithms as initial solutions are better than the greedy algorithms alone (i.e., SA_{AATCSR} is superior to AATCSR, SA_{FPI} is superior to FPI and Meta-$RaPS_{ERT}$ is superior to ERT). Statistically, it was shown that the CPU times of the proposed algorithms are quite similar, and they are considerably low compared to the optimal solution.

Throughout the course of daily operations, air traffic systems frequently encounter various disruptions because of the dynamic environment and unexpected events such as severe weather, aircraft failures or personnel shortages. Therefore, the initial plan may not be executed as designed. This dissertation addressed the Aircraft Reactive Scheduling Problem (ARSP) to update the existing aircraft schedule dynamically. The research considers disruptions including the arrival of new aircraft, flight cancellations and aircraft delays. ARSP is formulated as a multi-objective optimization problem in which both the schedule's quality and stability are of interest. The objectives consist of minimizing the total weighted start times (solution quality), total weighted start time deviation, and total weighted runway deviation (instability measures). Repair and complete regeneration approximate algorithms are developed for each type of disruptive events. Do-Nothing and Left-Shift are the repair strategies for the flight cancellations, RepairBySlack, RepairByEDD, InsertDelayed algorithms are proposed to repair the schedule for flight delays, and RepairByTWST and InsertNew are the repair algorithms for the arrival of new aircraft. Two complete regeneration algorithms, TWST Algorithm and SA-Re Algorithm are proposed to regenerate schedules from scratch to treat flight cancellations, delays and unexpected arrivals simultaneously.

All algorithms were tested against difficult benchmark problems and the solutions were compared to optimal solutions and to each other in terms of solution quality, schedule stability and computational time. A computational study was conducted for the three disruptive event types of ARSP with various values of objective weight coefficients π_1, π_2, π_3. Initially, response strategies to repair flight cancellation disruptions were evaluated. It was statistically illustrated that when the stability objective has a higher importance ($\pi_1 > \pi_3$), Do-Nothing algorithm is preferred. On the other hand, Left-Shift algorithm has significantly performed better when $\pi_1 \leq \pi_3$. Secondly, the repair algorithms for flight delays were tested. Depending on the initially selected strategy in the flight cancellation (i.e. Do-Nothing vs. Left-Shift) and the decision maker's interest in solution quality and stability, the performance of the repair algorithms was evaluated. When the solution quality is more important, RepairByEDD algorithm outperforms RepairBySlack and InsertDelayed. Conversely, InsertDelayed or RepairBySlack can be preferred depending on the repair algorithm used for cancellations when the schedule stability has a higher importance. Then, RepairByTWST and InsertNew repair algorithms were compared when Left-Shift strategy is used for flight cancellation, and RepairByEDD is used for delays. Depending on the importance of solution quality vs. stability, the performance of the repair algorithms was evaluated; when either the solution quality or schedule stability has higher importance, RepairByTWST outperforms the InsertNew. On the other hand, when the initially selected strategy in the flight cancellation is Do-Nothing, and in the flight delay is RepairBySlack, it was statistically shown that InsertNew is a better choice than RepairBySlack when the schedule stability is more important. Finally, the performances of the complete regeneration algorithms were tested. Although SA-Re requires longer CPU time than TWST, it is illustrated that either when the solution quality or schedule stability has higher importance, SA-Re performs better than TWST algorithm in terms of average error. Moreover, the average CPU time does not seem to change significantly with the change in π_1, π_2, and π_3.

7.2 Contributions

This dissertation research has the following contributions:

1. The ASP is modeled under a mixed mode of operations where both landing and departure flows are considered simultaneously, contrary to most existing studies that treat departures as separate from landings (i.e., segregated mode). Note that

the wake-vortex separation requirements for departures-only or arrivals-only operations usually satisfy the triangular inequality. However, the triangle inequality does not necessarily hold when both arrivals and departures are scheduled simultaneously increasing the complexity of the problem.

2. The problem of scheduling aircraft arrivals and departures over multiple runways is examined which much more dififcult than single runway problem. When a single runway is considered, although still NP-hard, one has merely to determine the sequence of the aircraft allocated to a runway. On the other hand, scheduling over multiple runways is a two-step process; first, one has to determine the assignment of aircraft to runways, then the sequence of the aircraft on each runway.

3. The features of the problem; unequal ready time, target time, deadline, sequence-dependent separation time, multi-resourced, single and multi-objective structure of the problem is unique which makes the dissertation remarkable as a new application of scheduling theory.

4. To our knowledge, more aspects to the problem are considered than any other previous work where we propose greedy algorithms (AATCSR, ERT and FPI) for the combined arrival-departure ASP with unequal ready-time, target-time, deadline, and sequence-dependent separation time. Finally, two metaheuristics (SA and Meta-RaPS) are introduced for the problem for the first time to improve initially constructed solutions by the proposed greedy algorithms.

5. The research that address multi-objective optimization problem in aircraft reactive scheduling problem that are liable to flight related disruptions is very limited. To fill this research gap, this dissertation updated mixed integer linear programming with normalized objective function to find optimal solutions, and proposed approximate algorithms to obtain near optimal schedules efficiently. The trade-off between the objectives are evaluated; the components of the multi-objective function are the total weighted start time which represents solution quality, and

the total weighted start time deviation and total weighted runway deviation which represent solution stability.

6. Unlike most studies in the literature which focus on one disruption type at a time, in this dissertation, different types of disruptions with multiple disruptive events are considered simultaneously. Therefore, the sequential evaluation methodology is developed to treat the disruptions and revise the schedules periodically. Alternative reactive scheduling approaches (Do-Nothing, Left-Shift, RepairByEDD, RepairBySlack, InsertDelayed, RepairByTWST, InsertNew, TWST and SA-Re) for different disruptions are proposed in which the model itself dynamically select the most appropriate from several candidate solution methods with respect to (conflicting) objectives of quality and stability.

7.3 Future Research

The problem can be extended in the future to address the following aspects:

1. The disruption of the schedules affects the capacity of the airport, causes passenger dissatisfaction and imposes substantial costs. In addition to maintaining conformity to the initial schedule, the convenience of the passengers who have connecting flights can be taken into account.

2. Case studies and real data analyses based on airport data from around the world would be interesting.

3. The work can be generalized for more complex aviation regulations such as stochastic time-windows (i.e. uncertain ready time, target time, deadline, etc.).

4. Reactive scheduling problem can be extended by considering more disruptive events such as runway closure.

5. The problem can be revised in such a way that the impacts of the operational settings (i.e., using the proposed algorithms) on the fuel cost savings and greenhouse gas emission be examined empirically.

BIBLIOGRAPHY

Abela, J., Abramson, D., Krishnamoorthy, M., De Silva, A., Mills, G. (1993). Computing optimal schedules for landing aircraft. *Proceedings of the 12th National ASOR Conference*, 71-90.

Akturk, M.S., & Gorgulu, E. (1999). Match-up scheduling under a machine breakdown. *Journal of Operational Research,* 112, 81-97.

Alagoz, O., & Azizoglu, M. (2003). Rescheduling of identical parallel machines under machine eligibility constraints. *European Journal of Operational Research,* 149, 523-532.

Al-Salem, A., Farhadi, F., Kharbeche, M., Ghoniem, A. (2012). Multiple-runway aircraft sequencing problems using mixed-integer programming. *Proceedings of the 2012 Industrial and Systems Engineering Research Conference.*

Arcus, A.L. (1966). COMSOAL: a computer method of sequencing operations for assembly lines. *International Journal of Production Research,* 4, 259-277.

Arguello, M.F., Bard,J.F., Yu, G. (1997). A GRASP for Aircraft Routing in Response to Groundings and Delays. *Journal of Combinatorial Optimization*, 5, 211–228.

Arnaout, J.P., & Rabadi G. (2008). Rescheduling of unrelated parallel machines under machine breakdowns. *International Journal of Applied Management Science, 1(1).*

Arnaout, J-P, Rabadi, G. & Musa, R. (2009). A two-stage ant colony optimization to minimize the makespan on unrelated parallel machines with sequence-dependent setup times, *Journal of Intelligent Manufacturing*, in press and available online.

Atkin, J.A.D., Burke, E.K., Greenwood, J.S. and Reeson, D., (2007). Hybrid metaheuristics to aid runway scheduling at London Heathrow Airport. *Transportation Science*, 41 (1), 90-106.

Atkin, J.A.D., Burke, E.K., Greenwood, J.S., Reeson, D., (2008). A meta-heuristic approach to departure scheduling at London Heathrow Airport. *Computer Aided Systems of Public Transport*, 235-252.

Atkin, J.A.D. (2008). On-line decision support for take-off runway scheduling at London Heathrow airport. Ph.D. thesis, The University of Nottingham

Atkin, J.A.D., Burke, E.K., Greenwood, J.S. (2010). TSAT Allocation at London Heathrow: the relationship between slot compliance, throughput and equity. Public Transport, 2(3), 173-198.

Azizoglu, M., & Alagoz, O. (2005). Parallel-machine rescheduling with machine disruptions. *HE Transactions,* 37, 1113-1118.

Balakrishnan, H. & Chandran, B. (2006) Scheduling aircraft landings under constrained position shifting. *AIAA guidance, navigation and control conference and exhibit*, Keystone, CO, USA.

Balakrishnan, H. and Chandran, B. (2010). Algorithms for scheduling runway operations under constrained position shifting, *Operations Research*, 58 (6), 1650-1665.

Bäuerle, N., Engelhardt-Funke, O., Kolonko, M. (2007), On the waiting time of arriving aircrafts and the capacity of airports with one or two runways, European Journal of Operational Research, 177 2 (1), 1180-1196.

Bazargan, M., Fleming, K. & Subramanian, P. (2002) A simulation study to investigate runway capacity using TAAM. *Proceeding of the 34th winter simulation conference*, San Diego, CA, USA.

Beasley, J. E., Krishnamoorthy, M., Sharaiha, Y. M., Abramson, D. (2000). Scheduling aircraft landings-the static case. *Transportation Science,* 34, 180-197.

Beasley, J., J. Sonander and P. Havelock (2001). Scheduling aircraft landings at London Heathrow using a population heuristic. *Journal of the Operational Research Society*, 52, 483-493.

Beasley, J. E., M. Krishnamoorthy, Y.M. Sharaiha, and D. Abramson (2004). Displacement problem and dynamically scheduling aircraft landings. *Journal of the Operational Research Society*, 55, 54–64.

Bean, J. C, Birge, J. R., Mittenthal, J., & Noon, C. E. (1991). Match-up scheduling with multiple resources, release dates and disruptions. *OperationsResearch, 39(3),* 470-483.

Bencheikh G., Boukachour J., Alaoui, A. E. H., and El Khoukhi F. (2009), Hybrid method for aircraft landing scheduling based on a job shop formulation, *International Journal of Computer Science and Network Security*, 9(8).

Bennell, J. A., Mesgarpour, M., Potts, C. N. (2011). Airport runway scheduling. *OR:Quartterly Journal of Operations Research*, 9(2), 115-138.

Bianco, L., Rinaldi, G. & Sassano, A. (1987). A combinatorial optimization approach to aircraft sequencing problem. *Flow Control of Congested Networks*. NATO AS I series, 38, 323-339.

Bianco, L., Dell'Olmo, P. & Giordani, S. (1997). Scheduling models and algorithms for TMA traffic management. *Modeling and Simulation in Air Traffic Management*. Springer, 139-167.

Blazewicz, J. , Ecker, K. H., Pesch, E., Schmidt, G. & Weglarz, J.(2007). *Handbook on Scheduling: From Theory to Applications*. Springer, New York.

Boysen, N., M. Fliedner. (2011). Scheduling aircraft landings to balance workload of ground staff. *Computers & Industrial Engineering*. 60(2) 206-217.

Brinton, C.R. (1992). An implicit enumeration algorithm for arrival aircraft scheduling. *Proceedings of the IEEE/AIAA 11th digital avionics systems conference*, Seattle, WA, USA.

Capri S., Ignaccolo, M. (2004). Genetic algorithms for solving the aircraft-sequencing problem:the introduction of departures into the dynamic model. *Journal of Air Transport Management*, 10, 345-351.

Carr, G.C., Erzberger, H. & Neuman, F. (2000). Fast-time study of airline-influenced arrival sequencing and scheduling. *Journal of Guidance,Control, and Dynamics*, 23,526–531.

Cheng, S-C, Shiau, D.-F., Huang, Y.-M., & Lin, Y.-T. (2009). Dynamic hardreal-time scheduling using genetic algorithm for multiprocessor task with resource and timing constraints. *Expert Systems with Applications,* 36, 852-860.

Church, L.K., & Uzsoy, R. (1992). Analysis of periodic and event driven rescheduling policies in dynamic shops. *International Journal of Computer Integrated Manufacturing,* 5, 153-163.

Ciesielski, V. and Scerri, P. (1998). Real time genetic scheduling of aircraft landing times, *Proceedings of the 1998 IEEE International conference on Evolutionary Computation (ICEC98)*, 360-364.

Curry, J., & Peters, B. (2005). Rescheduling parallel machines with stepwise increasing tardiness and machine assignment stability objectives. *International Journal of Production Research,* 43(15), 3231 - 3246.

Damodaran, P., Gallego, M. C. (2010). Heuristics for makespan minimization on parallel batch processing machines with unequal job ready times. *International Journal of Advanced Manufacturing Technologies,* 49, 1119-1128.

Dear, R. G., Sherif, Y. S. (1989). The dynamic scheduling of aircraft in high density terminal areas. *Microelectrons Reliability*, 29, 743–749.

Dear, R. G., Sherif, Y. S. (1991). An algorithm for computer assisted sequencing and scheduling of terminal area operations. *Transportation Research A,* 25, 29-139.

DePuy G.W., Whitehouse, G.E., Moraga, R.J. (2001). MetaRaPS: A simple and efficient approach for solving combinatorial problems. *29th International Conference on Computers and Industrial Engineering,* November 1-3, Montreal, Canada, 644-649.

Dorigo M., & Gambardella, L.M. (1997). Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions in Evolutionary Computation,* 1, 53-66.

Driessel, R., & Mönch, L. (2009). Scheduling jobs on parallel machines with sequence-dependent setup times, precedence constraints, and ready times using variable neighborhood search. *Proceedings of the 2009 IEEE International Conference on Industrial Engineering and Engineering Management.*

Du, J. & Leung, J. Y. (1990). Minimizing total tardiness on one machine is NP-Hard. *Mathematics of Operations Research,* 15, 483-494.

Duenas, A., & Petrovic, D. (2008). An approach to predictive-reactive scheduling of parallel machines subject to disruptions. *Annual Operations Research,* 159, 65-82.

Ernst, A.T., Krishnamoorthy, M. & Storer, R.H. (1999). Heuristic and exact algorithms for scheduling aircraft landings. *Networks,* 34, 229–241.

Federal Aviation Administration. (2003). *Aeronautical Information Manual/Federal Aviation Regulation.* McGraw-Hill, New York.

Gharehgozli, A. H., Tavakkoli, R., & Zaerpour, N. (2009). A fuzzy-mixed-integer goal programming model for a parallel-machine scheduling problem with sequence-dependent setup times and ready dates. *Robotics and Computer-Integrated Manufacturing,* 25, 853–859.

Ghoniem, A., Farhadi, F. (2012). Multiple-runway aircraft sequencing problems: Enhanced formulations and accelerated column generation Approach. *Manuscript Isenberg School of Management,* University of Massachusetts, Amherst, USA.

Grodzevich, O., & Romanko, O. (2006). Normalization and other topics in multiobjective optimization. *Proceedings of the Fields-MITACS Industrial Problems Workshop.*

Gupta, G., Malik, W., Jung, Y.C. (2009). A mixed integer linear program for airport departure scheduling. *9th AIAA Aviation Technology, Integration, and Operations Conference (ATIO).* AIAA, Hilton Head, South Carolina.

Hall, N.G., & Potts, C.N. (2004). Rescheduling for new orders. *OperationsResearch,* 52, 440-453.

Hancerliogullari, G., Rabadi, G., Al-Salem A., Kharbeche, M. (2013). Greedy Algorithms and Metaheuristics for a Multiple Runway Combined Arrival-Departure Aircraft Sequencing Problem, *Journal of Air Transport Management*, 32, 39-48.

Hansen, J. V. (2004). Genetic search methods in air traffic control. *Computers & Operations Research*, 31, 445-459.

Hazir, O., Giinalay, Y., & Erel, E. (2008). Customer order scheduling problem: a comparative metaheuristics study. *International Journal of Advanced Manufacturing Technology,* 37, 589-598.

Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, Ann Arbor, MI.

Hu, X.B., Chen, W.H. (2005). Genetic algorithm based on receding horizon control for arrival sequencing and scheduling. *Eng. Appl. Artif. Intell.* 18, 633–642.

Hu, X.B. and Paolo, E.D. (2008). Binary-representation-based genetic algorithm for aircraft arrival sequencing and scheduling. IEEE *Transactions on Intelligent Transportation System*, 9(2), 301-310.

Isaacson, D. R., Davis, T. J., & Robinson, J. E. (1997). Knowledge-based runway assignment for arrival aircraft in the terminal area. *AIAA Guidance, Navigation, and Control Conference,* New Orleans, Louisiana.

Itayef, A. B., Loukil, T., & Teghem, J. (2009). Rescheduling a permutation flowshop problems under the arrival a new set of jobs. *IEEE Transactions.*

Jain, A.K., & ElMaraghy, H. (1997). Production scheduling/rescheduling in flexible manufacturing. *International Journal of Production Research,* 35, 281-309.

Jarrah, A.I.Z., Yu G. N., Krishnamurthy, and Rakshit, A. (1993). A decision support framework for airline flight cancellations and delays, *Transportation Science*, 27 (3). 266–280.

Jeong S.J., Kim, K.S. (2008). Parallel machine scheduling with earliness-tardiness penalties and space limits. *The International Journal of Advanced Manufacturing Technology*, 37, 793-802.

Jungwattanakit, J., Reodechaa, M., Chaovalitwongsea, P., & Werner, F. (2009). A comparison of scheduling algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria, *Computers & Operations Research,* 36,358-378.

Kim,Y.D., Lim, H.G., & Park, M.W. (1996). Search heuristics for a flowshop scheduling problem in a printed circuit board assembly process. *European Journal of Operational Research,* 91, 124-43.

Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220:671.

Koulamas, C. (2010). The single-machine total tardiness scheduling problem: Review and extensions. *European Journal of Operational Research,* 202, 1-7.

Larson, R.E., Dessouky, M.I. (1978). Heuristic procedure for the single machine problem to minimize maximum lateness. *AIIE Transactions,* 10, 176-183.

Lawler, E. L., Lenstra, J.K. & Rinnooy A.H.G. (1982). Recent developments in deterministic sequencing and scheduling. *Deterministic and Stochastic Scheduling*, Reidel, Dordrecht, 35-73.

Lee, C.-Y., Leung, J. Y-T., & Yu, G. (2006). Two machine scheduling under disruptions with transportation considerations. *Journal of Scheduling,* 9, 35^-8.

Lee, Y.H. & Sherali, H. D. (1994). Unrelated machine scheduling with time-window and machine- downtime constraints: An application to a naval battle-group problem. *Annuals of Operations Research, special issue on Applications of Combinatorial Optimization*, 50, 339-365.

Lee, Y. H., Pinedo, M.(1997). Theory and methodology: Scheduling jobs on parallel machines with sequence-dependent setup times. *European Journal of Operational Research*, 100, 464-474.

Liu, Y.H. (2010). A genetic local search algorithm with a threshold accepting mechanism for solving with a threshold accepting mechanism for solving the runway dependent aircraft landing problem. *Optimization Letters,* 5, 229-245.

Marler, R.T., & Arora, J.S. (2004). Survey of multi-objective optimization methods for engineering. Structural Multidisciplinary Optimization, 26, 369-395.

Marler, R.T., & Arora, J.S. (2005). Function-transformation methods for multiobjective optimization. *Engineering Optimization, 37(6),* 551-570.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., Teller, E. (1956). Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21, 1087-1092.

Moraga, R. J. (2002). Meta-RAPS: An Effective Approach for Combinatorial Problems. *Ph.D. Dissertation.* University of Central Florida, Orlando.

Mönch, L., Balasubramanian, H., Fowler, J. W. & Pfund, M.E. (2005). Heuristic scheduling of jobs on parallel batch machines with incompatible job families and unequal ready times. *Computers & Operations Research,* 32, 2731–2750.

Neuman, F., Erzberger, H. (1990). Analysis of sequencing and scheduling methods for arrival traffic. *NASA Technical Memorandum*, April 1990.

Neuman, F. & Erzberger, H. (1991). Analysis of delay reducing and fuel saving sequencing and spacing algorithms for arrival traffic. *Technical report, TM-103880, Ames Research Center*, NASA, USA

Newell, G.F. (1979). Airport capacity and delays. *Transport Science*, 13, 201–241.

Parthasarathy, S., & Rajendran, C. (1998). Scheduling to minimize mean tardiness and weighted mean tardiness in flowshop and flowline-based manufacturing cell. *Computers and Industrial Engineering,* 34, 531-546.

Pfund, M., Fowler, J.W., Gadkari, A., Chen, Y. (2008). Scheduling jobs on parallel machines with setup times and ready times. *Computers and Industrial Engineering,* 54, 764–782.

Pinedo, M. (2008). *Scheduling Theory, Algorithms, and Systems.* 3rd Ed.,Springer, New York.

Pinol, H. and Beasley, J. E. (2006). Scatter search and bionomic algorithms for the aircraft landing problem. *European Journal of Operational Research,* 171, 439-462.

Psaraftis, H. N. (1980). Dynamic programming approach for sequencing group of identified jobs. *Operations Research*, 28, 1347-1359.

Rabadi, G., Mollaghasemi, M., & Anagnostopoulos, G.C. (2004). A branch-and-bound algorithm for the early/tardy machine scheduling problem with a common due-date and sequence-dependent setup time. *Computers & Operations Research*, 31(10), 1727-1751.

Rabadi, G., Moraga, R., & Al-Salem, A. (2006). Heuristics for the unrelated parallel machine scheduling problem with setup times. *Journal of Intelligent Manufacturing*, 17, 85-97.

Rabadi, G., Hancerliogullari, G., Kharbeche, M., Al-Salem, A. (2012). Metaheuristics for aircraft arrival and departure scheduling on multiple runways. *Proceedings of the 2012 Industrial and Systems Engineering Research Conference.*

Rangaiah, G. P. (2008). *Multi-Objective Optimization: Techniques and Applications in Chemical Engineering.* World Scientific Publishing.

Reichelt, D., Monch, L., Gottlieb, J. & Raidl, G.R. (2006). Multi-objective scheduling of jobs with incompatible families on parallel batch machines. *EvoCOP 2006,* LNCS 3906, Berlin Springer-Verlag Heidelberg, 209-221.

Sherali, H.D., Hobeika, A.G., Trani, A.A., Kim, B.J. (1992). An integrated simulation and dynamic programming approach for determining optimal runway exit locations. *Management Science,* 38, 1049-1062.

Sherali, H.D., Ghoniem, A., Baik, H. and Trani, A.A. (2010). A combined arrival-departure aircraft sequencing problem. *Manuscript, Grado Department of Industrial and Systems Engineering.* Virginia Polytechnic Institute and State University, 250 Durham Hall, Blacksburg, VA 24061.

Shi-jin, W., Li-feng, X. & Bing-hai, Z. (2007). Filtered-beam-search-based algorithm for dynamic rescheduling in FMS. *Robotics and Computer-Integrated Manufacturing,* 23, 457-468.

Soomer, M.J. & Franx, G.J. (2008). Scheduling aircraft landings using airlines' preferences. *European Journal of Operational Research*, 190, 277-291.

Subramaniam, V., Raheja, A.S. & Reddy, K.R.B. (2005). Reactive repair tool for job shop schedules. *International Journal of Production Research,* 43, 1-23.

Tsai, W.J., Lee, S.Y. (1996). Real-time scheduling of multimedia data retrieval to minimize buffer requirement. *ACM SIGOPS Operating Systems Review,* 30, 67-80.

Unal, A.T., Uzsoy, R., & Kiran, A.S. (1997). Rescheduling on a single machine with part-type dependant setup times and deadlines. *Annals of Operations Research,* 70, 93-113.

U.S. Department of Transportation (2008). Air Travel Consumer Report October 2008.URL:http://airconsumer.ost.dot.gov/reports/2008/October/200810ATCR.pdf

Vallada, E. Ruiz, R, & Minella, G. (2008). Minimising total tardiness in the mmachine flowshop problem: A review and evaluation of heuristics and metaheuristics. *Computers and Operations Research,* 35, 1350- 1373.

Venkatakrishnan, C.S., Barnett, A.I. & Odoni, A.R. (1993). Landings at Logan Airport: Describing and increasing airport capacity. *Transportation Science*, 27, 211-227.

Vieira, G.E., Herrmann, J. W., & Lin E. (2000). Predicting the performance of rescheduling strategies for parallel machine systems. *Journal of Manufacturing Systems,* 9(4), 256-266.

Vieira, G.E., Herrmann, J. W., & Lin E. (2003). Rescheduling manufacturing systems: a framework of strategies, policies and methods. *Journal of Scheduling, 6(39),* 62.

Wang, S., (2009). Solving aircraft-sequencing problem based on bee evolutionary genetic algorithm and clustering method, *8th IEEE International Conference on Dependable,* Autonomic and Secure Computing, 157-161.

Wen, M., Larsen, J., Clausen, J. (2005). An exact algorithm for Aircraft Landing Problem, Technical Paper.

Wu, S. D., R. H. Storer, P.-C. Chang. (1993). One-machine rescheduling heuristics with efficiency and stability as criteria. *Computers and Operations Research,* 20, 1-14.

Yang, M.-H., Chung, S.-H., & Kao, C.-K. (2006). A comparative study to minimize the makespan of parallel-machine problem with job arrival in uncertainty. *HE Conference Proceedings.*

Yu, S. P., Cao, X.B., Zhang, J. (2011). A real-time schedule method for aircraft landing scheduling problem based on cellular automation, *Applied Soft Computing Journal.*

Zobolas, G.I., Tarantilis, CD., & Ioannou, G. (2008). Exact, heuristic and metaheuristic algorithms for solving shop scheduling problems. *Studies in Computational Intelligence,* 128, 1-40.

APPENDICES

APPENDIX A. T-TEST RESULTS OF THE FLIGHT DELAY ALGORITHMS

Paired T-Test and CI: do nothing-RepairEDD-sta, do nothing-RepairBySl-st

```
Paired T for do nothing-RepairEDD-stabilit - do nothing-
RepairBySl-stabili

                            N      Mean     StDev   SE Mean
do nothing-RepairEDD-sta  275   0.18240   0.09886   0.00596
do nothing-RepairBySl-st  275   0.08554   0.09611   0.00580
Difference                275   0.09685   0.04393   0.00265


95% CI for mean difference: (0.09164, 0.10207)
T-Test of mean difference = 0 (vs not = 0): T-Value = 36.56
P-Value = 0.000
```

Paired T-Test and CI: do nothing-RepairEDD-sta, do nothing-InsertDel-sta

```
Paired T for do nothing-RepairEDD-stabilit - do nothing-
InsertDel-stabilit

                            N      Mean     StDev   SE Mean
do nothing-RepairEDD-sta  275   0.18240   0.09886   0.00596
do nothing-InsertDel-sta  275   0.09483   0.08554   0.00516
Difference                275   0.08757   0.03555   0.00214


95% CI for mean difference: (0.08335, 0.09179)
T-Test of mean difference = 0 (vs not = 0): T-Value = 40.86
P-Value = 0.000
```

Paired T-Test and CI: do nothing-RepairBySl-st, do nothing-InsertDel-sta

```
Paired T for do nothing-RepairBySl-stabili - do nothing-
InsertDel-stabilit

                           N       Mean      StDev    SE
Mean
do nothing-RepairBySl-st  275    0.08554   0.09611
0.00580
do nothing-InsertDel-sta  275    0.09483   0.08554
0.00516
Difference                275  -0.009282  0.013540
0.000817


95% CI for mean difference: (-0.010889, -0.007674)
T-Test of mean difference = 0 (vs not = 0): T-Value = -
11.37  P-Value = 0.000
```

Paired T-Test and CI: do nothing-RepairEDD-qua, do nothing-RepairBySl-qu

```
Paired T for do nothing-RepairEDD-quality - do nothing-
RepairBySl-quality

                           N      Mean     StDev  SE Mean
do nothing-RepairEDD-qua  440    0.3835    0.2129   0.0101
do nothing-RepairBySl-qu  440    0.4762    0.2767   0.0132
Difference                440  -0.09268   0.07408  0.00353


95% CI for mean difference: (-0.09963, -0.08574)
T-Test of mean difference = 0 (vs not = 0): T-Value = -
26.24  P-Value = 0.000
```

Paired T-Test and CI: do nothing-RepairEDD-qua, do nothing-InsertDel-qua

```
Paired T for do nothing-RepairEDD-quality - do nothing-
InsertDel-quality

                               N       Mean     StDev  SE Mean
do nothing-RepairEDD-qua     440     0.3835    0.2129   0.0101
do nothing-InsertDel-qua     440     0.4116    0.2369   0.0113
Difference                   440   -0.02802   0.03772  0.00180


95% CI for mean difference: (-0.03155, -0.02448)
T-Test of mean difference = 0 (vs not = 0): T-Value = -
15.58  P-Value = 0.000
```

Paired T-Test and CI: do nothing-RepairBySl-qu, do nothing-InsertDel-qua

```
Paired T for do nothing-RepairBySl-quality - do nothing-
InsertDel-quality

                               N      Mean     StDev  SE Mean
do nothing-RepairBySl-qu     440    0.4762    0.2767   0.0132
do nothing-InsertDel-qua     440    0.4116    0.2369   0.0113
Difference                   440   0.06466   0.04034  0.00192


95% CI for mean difference: (0.06089, 0.06844)
T-Test of mean difference = 0 (vs not = 0): T-Value = 33.63
P-Value = 0.000
```

APPENDIX B. T-TEST RESULTS OF THE NEW FLIGHT ALGORITHMS

Paired T-Test and CI: DN-RepairSLACK-RepairTWS, DN-RepairSLACK-INSERTNEW

```
Paired T for DN-RepairSLACK-RepairTWST-STABI - DN-
RepairSLACK-INSERTNEW-STABIL

                             N      Mean     StDev  SE Mean
DN-RepairSLACK-INSERTNEW   275   0.23888   0.09743  0.00588
DN-RepairSLACK-RepairTWS   275   0.31365   0.13057  0.00787
Difference                 275  -0.07477   0.05223  0.00315


95% CI for mean difference: (-0.08097, -0.06857)
T-Test of mean difference = 0 (vs not = 0): T-Value = -
23.74  P-Value = 0.000
```

Paired T-Test and CI: DN-RepairSLACK-RepairTWS, DN-RepairSLACK-INSERTNEW

```
Paired T for DN-RepairSLACK-RepairTWST-QUALI - DN-
RepairSLACK-INSERTNEW-QUALIT

                             N      Mean     StDev  SE Mean
DN-RepairSLACK-RepairTWS   440   0.28900   0.12026  0.00573
DN-RepairSLACK-INSERTNEW   440   0.34440   0.13626  0.00650
Difference                 440  -0.05540   0.04190  0.00200


95% CI for mean difference: (-0.05932, -0.05147)
T-Test of mean difference = 0 (vs not = 0): T-Value = -
27.73  P-Value = 0.000
```

Paired T-Test and CI: DN-RepairSLACK-RepairTWS, DN-RepairSLACK-INSERTNEW

```
Paired T for DN-RepairSLACK-RepairTWST-STA_1 - DN-
RepairSLACK-INSERTNEW-STAB_1

                            N      Mean     StDev  SE Mean
DN-RepairSLACK-INSERTNEW  275   0.36239   0.09766  0.00589
DN-RepairSLACK-RepairTWS  275   0.46286   0.10220  0.00616
Difference                275  -0.10047   0.01979  0.00119


95% CI for mean difference: (-0.10282, -0.09812)
T-Test of mean difference = 0 (vs not = 0): T-Value = -
84.17  P-Value = 0.000
```

Paired T-Test and CI: DN-RepairSLACK-RepairTWS, DN-RepairSLACK-INSERTNEW

```
Paired T for DN-RepairSLACK-RepairTWST-QUA_1 - DN-
RepairSLACK-INSERTNEW-QUAL_1

                            N      Mean     StDev  SE Mean
DN-RepairSLACK-RepairTWS  440   0.21170   0.07929  0.00378
DN-RepairSLACK-INSERTNEW  440   0.24933   0.07368  0.00351
Difference                440  -0.03763   0.04365  0.00208


95% CI for mean difference: (-0.04172, -0.03354)
T-Test of mean difference = 0 (vs not = 0): T-Value = -
18.09  P-Value = 0.000
```

VITA

Gulsah HANCERLIOGULLARI

Engineering Management and Systems Engineering
Old Dominion University, Norfolk, VA
e-mail: ghancerl@odu.edu, gulsah_2207@hotmail.com

EDUCATION

Old Dominion University, Norfolk, VA	
Ph.D., Engineering Management and Systems Engineering	June 2013
Bilkent University, Ankara, Turkey	
M.S., Industrial Engineering	July 2010
Bilkent University, Ankara, Turkey	
B.S., Industrial Engineering	June 2008

RESEARCH INTERESTS

Algorithm Development for Optimization Problems, Modeling and Analysis of Scheduling and Rescheduling Systems, Decision Support Systems, Empirical Research in Inventory Management, Supply Chain Management for Retailers, Quality Engineering and Experimental Design

PUBLICATIONS

Hancerliogullari, G., Rabadi, G., Al-Salem A., Kharbeche, M. , 2013. Greedy Algorithms and Metaheuristics for a Multiple Runway Combined Arrival-Departure Aircraft Sequencing Problem, *Journal of Air Transport Management*, 32, 39-48.

Hancerliogullari, G., Rabadi, G., Kharbeche, M., Al-Salem, A. 2012. Heuristic algorithms for aircraft sequencing problem, *Proceeding of the 2012 International Annual Conference of the American Society for Engineering Management.*

Rabadi, G., Hancerliogullari, G., Kharbeche, M., Al-Salem, A. 2012. Meta-heuristics for aircraft arrival and departure scheduling on multiple runways, *Proceedings of the 2012 Industrial and Systems Engineering Research Conference.*

Rabadi, G., Ghoniem, A., Al-Salem, A., Kharbeche, M., Hancerliogullari, G., Shahrestanaki, F. 2012. Aircraft Scheduling on Multiple Runways, *Qatar Foundation Annual Research Forum Proceedings.*