

**Crew Rostering in Thai Airways Case
Using Greedy Algorithm**

BY

Thanaphat Limgitnuwat

**A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF MASTER OF
ENGINEERING (LOGISTICS AND SUPPLY CHAIN SYSTEMS
ENGINEERING)
SIRINDHORN INTERNATIONAL INSTITUTE OF TECHNOLOGY
THAMMASAT UNIVERSITY
ACADEMIC YEAR 2014**

**Crew Rostering in Thai Airways Case
Using Greedy Algorithm**

BY

Thanaphat Limgitnuwat



**A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF MASTER OF
ENGINEERING (LOGISTICS AND SUPPLY CHAIN SYSTEMS
ENGINEERING)
SIRINDHORN INTERNATIONAL INSTITUTE OF TECH
THAMMASAT UNIVERSITY
ACADEMIC YEAR 2014**



**Crew Rostering in Thai Airways Case
Using Greedy Algorithm**

A Thesis Presented

By

Thanaphat Limgitnuwat

Submitted to

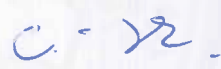
Sirindhorn International Institute of Technology

Thammasat University

In partial fulfillment of the requirements for the degree of
MASTER OF ENGINEERING (LOGISTICS AND SUPPLY CHAIN SYSTEMS
ENGINEERING)

Approved as to style and content by

Advisor and Chairperson of Thesis Committee



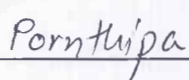
(Asst. Prof. Chawalit Jeenanunta, Ph.D.)

Committee Member and
Chairperson of Examination Committee



(Asst. Prof. Aussadavut Dumrongsiri, Ph.D.)

External Examiner



(Asst. Prof. Pornthipa Ongkunaruk, Ph.D.)

August 2015

Abstract

CREW ROSTERING IN THAI AIRWAYS CASE USING GREEDY ALGORITHM

by

THANAPHAT LIMGITNUWAT

Master of Engineering in Logistics and Supply Chain Systems Engineering (LSCSE),
Sirindhorn International Institute of Technology, Thammasat University, 2015

This thesis presents a crew rostering in Thai Airways case by using Greedy Algorithm. Crew rostering is the process that is used to assign suitable task to specific crew. The result of this experiment is formulated in form of crew timetable. The objective is to balance workload and perdiem simultaneously in order to increase fairness and reduce airline operation cost. Standard Deviation (SD) was declared in order to balance workload and perdiem. Thus, this experiment focus on SD of workload and perdiem minimization. This thesis can be divided into two main phases; construction phase and improvement phase. The construction phase is used for constructing the simple crew timetable. The improvement phase is used for reducing SD of workload and perdiem from construction phase. Moreover, improvement phase can be divided into four techniques including change pairing directly, change pairing descending, change pairing ascending, and high workload and perdiem distribution technique. These techniques were tested on five different methods which consist of workload minimization, perdiem minimization, workload and perdiem minimization simultaneously without bound, workload and perdiem minimization simultaneously with workload bound, and workload and perdiem minimization simultaneously with perdiem bound. In this thesis, C programming language and Microsoft Visual C++ 2010 program were applied to compile the solution. The result shows not too high SD of workload and perdiem reduction because of nature of greedy algorithm that can find only global optimal solution in some point. Even though, the compilation time is very short, approximately one to five seconds depending on the complexity of code, number of instances and constraints, and performance of compiler tool.

Keywords: Crew Rostering, Greedy Algorithm, Heuristic, Scheduling

Acknowledgements

First of all, I would like to take this opportunity to express my sincere thanks to my family, who always support and encourage me through good and bad times. I also truthfully express gratitude to my advisor, Assistant Professor Dr. Chawalit Jeenanunta, for everything throughout many years of my research. The next special thanks are for my friend; Mathus Tuachob, and Wittawat Jikrittum, who always help me with theory, algorithm, and programming. They recommended many important solutions that help me to complete the coding. Another special thanks to Nitipon Tansakul, and Pimnapa Pongsayaporn for their support and useful comments of my work. I also want to thanks Sophea Horng, Pongwat Poomchoompol, and Khiriphach Makarangkurn for their help with documents and programs. I also appreciate Tapanee Sunthornsaratul, Eksiri Lakswong, and Parichart Pornpisitchok for always being there when I need support.

I am greatly grateful to the examination committee Assistant Professor Dr. Aussadavut Dumrongsiri and the external examiner Assistant Professor Dr. Pornthipa Ongkunaruk for their useful comments and suggestions. I also would like to extend my gratitude to Associate Professor Dr. Navee Chiadamrong and Associate Professor Dr. Jirachai Buddhakulsomsiri for their advices to get through my hard time.

Finally, I would like to thank all staffs from Sirindhorn International Institute of Technology (SIIT), Thammasat University and Thai Airways Company for providing an opportunity and all the necessary information.

Table of Contents

Chapter	Title	Page
	Signature Page	i
	Abstract	ii
	Acknowledgements	iii
	Table of Contents	iv
	List of Tables	vi
	List of Figures	vii
1	Introduction	1
	1.1 Problem Statement	3
	1.2 Objective	4
	1.3 Overview of Research	4
2	Literature Review	5
	2.1 Greedy Algorithm	5
	2.2 Crew Rostering	6
	2.3 Airline Scheduling	9
3	Problem Formulation and Methodology	11
	3.1 ACRP of Thai Airways	11
	3.1.1 Test Instances	12
	3.2 Methodology	13
4	Greedy Algorithm for Solving ACRP Problem	17
	4.1 Greedy Algorithm	17
	4.2 Solution Model	18
	4.2.1 Construction Phase	20
	4.2.2 Improvement Phase	21
	4.2.2.1 Change pairing directly	21
	4.2.2.1 Change pairing descending	28
	4.2.2.1 Change pairing ascending	35
	4.2.2.1 High workload and perdiem distribution	42
5	Results and Discussions	49
	5.1 Result and Discussion	49
	5.2 Minimize SD of workload	49
	5.3 Minimize SD of perdiem	52
	5.4 Minimize SD of workload and perdiem simultaneously without bound	54
	5.5 Minimize SD of workload and perdiem simultaneously with workload bound	58
	5.6 Minimize SD of workload and perdiem simultaneously with perdiem bound	62
	5.7 Compare solution	66
6	Conclusion and Further Study	79
	6.1 Conclusion	79

6.2 Further Study	79
References	80
Appendices	82
Appendix A	83
Appendix B	87
Appendix C	98



List of Tables

Tables	Page
1.1 Weekly roster	3
1.1 Weekly roster	3
3.1 Thai Airways crew division	11
3.2 Thai Airways fleets	11
3.3 Examples of pairing data	12
3.4 Test instances	13
3.5 Rest period constraint	14
3.6 Crew time table format	14
5.1 SD of workload table using workload minimization	50
5.2 SD of perdiem table using workload minimization	51
5.3 SD of workload table using perdiem minimization	52
5.4 SD of perdiem table using perdiem minimization	53
5.5 SD of workload table using workload and perdiem minimization without bound	54
5.6 SD of perdiem table using workload and perdiem minimization without bound	55
5.7 Total normalized workload and perdiem SD (Twp) without bound table	57
5.8 SD of workload table using workload and perdiem minimization with workload bound	59
5.9 SD of perdiem table using workload and perdiem minimization with workload bound	60
5.10 Total normalized workload and perdiem SD (Twp) with workload bound table	61
5.11 SD of workload table using workload and perdiem minimization with perdiem bound	63
5.12 SD of perdiem table using workload and perdiem minimization with perdiem bound	64
5.13 Total normalized workload and perdiem SD (Twp) with perdiem bound table	65
5.14 Result comparison SD of workload	67
5.15 Result comparison SD of perdiem	68
5.16 Result comparison of percentage changed of total normalized workload and perdiem	70

List of Figures

Figures	Page
3.1 Example pairing for one day	12
3.1 Example pairing for two days	12
4.1 Example of good Greedy Algorithm	17
4.2 Example of problem that Greedy Algorithm cannot solve	18
4.3 Solution of exchange money	18
4.4 Flow chart of construction phase	19
4.5 Flow chart of change pairing directly with workload minimization	22
4.6 Flow chart of change pairing directly with perdiem minimization	23
4.7 Flow chart of change pairing directly with workload and perdiem minimization simultaneously without bound	24
4.8 Flow chart of change pairing directly with workload and perdiem minimization simultaneously with workload bound	25
4.9 Flow chart of change pairing directly with workload and perdiem minimization simultaneously with perdiem bound	26
4.10 Flow chart of change pairing descending with workload minimization	29
4.11 Flow chart of change pairing descending with perdiem minimization	30
4.12 Flow chart of change pairing descending with workload and perdiem minimization simultaneously without bound	31
4.13 Flow chart of change pairing descending with workload and perdiem minimization simultaneously with workload bound	32
4.14 Flow chart of change pairing descending with workload and perdiem minimization simultaneously with perdiem bound	33
4.15 Flow chart of change pairing ascending with workload minimization	36
4.16 Flow chart of change pairing ascending with perdiem minimization	37
4.17 Flow chart of change pairing ascending with workload and perdiem minimization simultaneously without bound	38
4.18 Flow chart of change pairing ascending with workload and perdiem minimization simultaneously with workload bound	39
4.19 Flow chart of change pairing ascending with workload and perdiem minimization simultaneously with perdiem bound	40
4.20 Flow chart of high workload and perdiem distribution with workload minimization	43
4.21 Flow chart of high workload and perdiem distribution with perdiem minimization	44
4.22 Flow chart of high workload and perdiem distribution with workload and perdiem minimization simultaneously without bound	45
4.23 Flow chart of high workload and perdiem distribution with workload and perdiem minimization simultaneously with workload bound	46
4.24 Flow chart of high workload and perdiem distribution with workload and perdiem minimization simultaneously with perdiem bound	47
5.1 Total normalized SD of all techniques without bound graph	58
5.2 Total normalized SD of all techniques with workload bound graph	62
5.3 Total normalized SD of all techniques with perdiem bound graph	66
5.4 Compare SD of workload by using change pairing directly technique	71

5.5 Compare SD of perdiem by using change pairing directly technique	72
5.6 Compare SD of workload by using change pairing descending technique	73
5.7 Compare SD of perdiem by using change pairing descending technique	74
5.8 Compare SD of workload by using change pairing ascending Technique	75
5.9 Compare SD of perdiem by using change pairing ascending technique	76
5.10 Compare SD of workload by using high workload and perdiem distribution technique	77
5.11 Compare SD of perdiem by using high workload and perdiem distribution technique	78



Chapter 1

Introduction

After the Wright brothers successfully flew using a powered aircraft in 1903, the first airline was established six years later. Today, there are hundreds of airlines in all areas of the world. The airline industry had a profit of about US\$369 billion in 2004 and profits doubled to US\$746 billion in 2014. Due to the popularity of air transportation, many airlines are planning to expand their routes and fleets. Thus, more crews and other employees are being hired. The approximate crew requirement in 2020 will be more than 550,000. Consequently, management of crew pairing, crew scheduling, and fleet assignment is a complex process. Thus, this thesis focuses on applying the Greedy Algorithm to solve crew rostering problems at Thai Airways. In addition, this experiment uses data from Thai Airways to find a solution of workload and per diem balancing optimization. The crew scheduling or crew rostering involves the process of assigning crew pairing or flight route, crews, and other information. Crew scheduling contains two major phases: crew pairing and crew rostering. Crew pairing is the management process of the flight legs within the same fleet that starts and ends at the same crew base. The meaning of crew base is the original airport, or hometown of the crew member. There are many constraints that crew pairing needs to consider, such as, airline union, government rules, and airline regulations. As a result, the objective of crew pairing is to manage a set of flights that start and end at home base station and also minimize the total crew cost. Table 1.1 shows the two-day pairing that has John F. Kennedy International Airport as a home base; thus, one pairing should include at least two flight trips: away, and departure.

Solution	Day 1			Day 2		
Pairing#1	FLT 125			FLT 105		
City Pairs	JFK-SFO			SFO-JSK		
Dept-Arr Times	7:25-9:55			9:55-18:20		
Pairing#21	FLT 131	FLT 111	FLT 136	FLT 113	FLT 138	FLT 118
City Pairs	JFK-ATL	ATL-JFK	JFK-MIA	MIA-JFK	JFK-BOS	BOS-JFK
Dept-Arr Times	9:30-12:00	13:10-15:40	18:10-21:10	9:10-12:10	12:30-14:00	15:00-16:30
Pairing#9	FLT 135			FLT 114		
City Pairs	JFK-MIA			MIA-JFK		
Dept-Arr Times	15:10-18:10			14:30-17:30		
Pairing#5	FLT 133			FLT 110		
City Pairs	JFK-ATL			ATL-JFK		
Dept-Arr Times	18:05-20:35			10:40-8:10		

Table 1.1 Example of crew pairing

Moreover, another phase of crew scheduling is crew rostering. The purpose of crew rostering is to assign individual crew members to a crew pairing which is usually on a monthly basis. There are three main methods to roster a crew, such as, assigning high priority employees to the high priority pairing, developing monthly rosters for individual crew members based on their requests, and developing monthly rosters for each day of the month without considering the crew request. Generally, there are different processes to assign the roster to cockpit aircrew members (captain, and first officer), and cabin aircrew members because cockpit aircrew members may require licenses to fly with specific type of aircraft. Moreover, crew rostering will be developed on a weekly basis instead of as a monthly roster because it is easier and less complex to solve. Table 1.2 shows the weekly roster of 14 captains and 4 pairings, where each pair needs to fly every day in a week. Then, the 4 weekly rosters are combined to be the monthly roster.

Table 1.2 weekly roster

Rosters	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Flight Hours
1	P1	0	0	P4	0	0	0	16
2	0	P1	0	0	P4	0	0	16
3	0	0	P1	0	0	P4	0	16
4	0	0	0	P1	0	0	P4	16
5	P2	0	0	0	0	0	P3	20
6	0	P2	0	0	P3	0	0	20
7	0	0	P2	0	0	P3	0	20
8	0	P3	0	0	P1	0	0	17
9	0	0	P3	0	0	P1	0	17
10	0	0	0	P3	0	0	P1	17
11	P3	0	0	0	0	0	P2	20
12	P4	0	0	P2	0	0	0	19
13	0	P4	0	0	P2	0	0	19
14	0	0	P4	0	0	P2	0	19

The Greedy Algorithm is a solution that attempts to find the global optimal solution by considering the local optimal solution as a priority. The method to solve problem of the Greedy Algorithm is choosing the stage that offers the most obvious and immediate benefit. The disadvantage of this algorithm is that it may not find the global optimal solution of the problem. After finishing this stage, it cannot reconsider the rest of direction or other phase again. Thus, it is very useful for some small instances problem.

1.1 Problem Statement

Due to the high operation cost of airlines and increasing crew hiring, the complexity of crew scheduling or crew rostering is significant. Thus, this thesis proposes the experiment on balancing workload and perdiem (salary) in order to increase fairness and reduce airline operation cost. The complexity of crew rostering comes from priority, perdiem, rest time, workload, different types of fleet, block time, and specific regulations in some countries. Thus, this thesis proposes a Greedy Algorithm technique to solve ACRP (Airline Crew Rostering Problem). The experiment consists of two phases: a construction phase, and an improvement phase. The construction phase aims to construct a simple crew table for revision to balance workload and perdiem in the improvement phase. The construction phase can be divided in to four techniques, such as change pairing directly, change pairing descending, change pairing ascending, and high crew distribution.

1.2 Objective

The objective is to balance workload and perdiem. To achieve the objective, Standard Deviation (SD) purposed to minimize. The Standard Deviation indicates the variation of all data and the best solution of SD is close to zero. This thesis proposes three different minimization experiments: minimize SD of workload, minimize SD of perdiem, and minimize both SD of workload and perdiem at the same time.

1.3 Overview of Research

This thesis report is organized as follows. Literature reviews are described in chapter 2. Chapter 3 explains about problem formulation and the method to solve crew rostering problem. Chapter 4 proposes a Greedy Algorithm with two phases, such as, construction phase, and improvement phase. Chapter 5 explains the results after applying the Greedy Algorithm. Finally, chapter 6 interprets the discussion and further study

Chapter 2

Literature Review

The review of literature that is related to the use of the Greedy Algorithm for crew rostering is presented in this chapter, which is organized into three parts as follows:

2.1 Greedy Algorithm

2.2 Crew Rostering

2.3 Airline Scheduling

2.1 Greedy Algorithm

The Greedy Algorithm is a popular method to find an optimal solution. It focuses on the best or highest benefit first. Thus, it is very useful and easy to apply to a small problem. So, there have been several researches that study the Greedy Algorithm, the most significant of which are mentioned below.

N. Lesh and M.Mitzenmacher (2005) introduces Bubble Search for applying with Greedy Heuristics. This program concerns the algorithm sequence for adding elements. These problems can be solved by using Bubble Search to select the closest element or path. Thus, the applied Greedy Heuristics is a consequence in effectiveness.

A.G. Logodimos and V. Leopoulos (2000) investigate Greedy heuristic algorithms for manpower shift planning. The instances of this experiment came from a food manufacturing company. The objective is to manage personal schedule tasks. The important thing to find is a minimum workforce in a working day period. They proposed Manpower Shift Planning (MSP) by creating the linear programming. They also proposed Greedy Heuristics Algorithm be used in the experiments. Consequently, the outcome solution was very satisfactory in terms of quality and computational time.

Kahraman et al. (2010) presented about multiprocessor task scheduling in multistage hybrid flow-shops using a parallel greedy algorithm approach. Their experiment was about the Hybrid Flow Shop Scheduling with Multiprocessor Task (HFSMT) problem. They also applied a parallel greedy algorithm (PGA) to get the solution. Two techniques are proposed to solve this problem: the Destruction technique, and Construction technique. In conclusion, they successfully applied this technique for making a span deduction.

Lin and Ying (2014) aimed to solve the personnel task scheduling problem by proposing an effectiveness and efficiency of three-phase algorithm for solving the shift minimization personnel task scheduling problem (SMPTSP). They tested the problem set that was inspired from employee scheduling application in order to demonstrate the increasing of efficiency. In conclusion, their algorithm was effective, efficient, and robust to the problem.

Korhan and Fatih (2014) experimented on greedy algorithm for solving the traveling sales man problem with time windows (TSPTW). The objective is to sum up travel cost or makespan minimization. The greedy algorithm is used for changing the neighborhood in Variable Neighborhood Search (VNS). In order of changing, the algorithm will remove previous neighborhood and construct the new best neighborhood solution. The result shows better performance compared to the literature.

2.2 Crew Rostering

Ernst et al. (2004) focused on staff scheduling and rostering, reviewing the applications, methods and models. The problem of this paper is the difficulty of satisfying the staff demand, such as, flexible workplace agreements, shift equity, staff preferences, and part-time work. The objective is to find the best technique for solving the staff scheduling and rostering problem. Thus, it can be categorized into 6 problems: demand modeling, days off scheduling, shift scheduling, line of work construction, task assignment, and staff assignment. This study also classifies the solutions and techniques into 5 methods: demand modeling, artificial intelligence approaches, constraint programming, metaheuristics, and mathematical programming approaches.

Yinghui et al. (2007) studied how to solve the problem of automatically generating crew rostering. The objective was to find the optimal crew rostering table while minimizing cost and maximizing profit. There are two techniques included in this study, a Genetic Algorithm (GA), and a simulated annealing algorithm. Moreover, the combination of simulated annealing and genetic algorithm proposed in this paper received satisfied results for multiple objectives.

Azadeh et al. (2013) examined Particle Swarm Optimization (PSO) which aimed to minimize total cost of crew assignment. The problem is in the form of an NP-hard problem. This method achieved a more effective result than other algorithms. Moreover, they studied and experimented about Genetic Algorithm (GA) and Ant Colony Optimization (ACO) to get the solution of a high number of instances.

El Moudanis et al. (2010) attempted to construct the assignment of the crew staff as a set of pairings covering all the scheduled flights. Also, this paper developed the new mathematical formulation of the crew scheduling problem. The objective is to minimize operation cost. The technique for solving this airline crew rostering problem is a bi-criterion method. The constraints are regulations of Civil Aviation, workload, number of crews, and cost. Thus, the mathematical programming and artificial intelligence techniques do not contribute an exact solution in pure mathematical terms

but appear to be quite adapted to give real support to the decision making process by providing, through a comprehensive process, an improved approximation of the set of non-inferior solutions attached to this bi-criterion decision problem.

Kharraziha et al. (1996) focused on large scale crew rostering to develop Carmen Systems' crew rostering product that currently is used by several major European airline and railway companies. In the largest problem, around 7,000 crew members are assigned roughly 22,000 tasks. The objective of this paper was to optimize the crew roster of European airlines and railways. This paper also implemented the technique used by Carmen Systems, a developer and vendor of resource optimization software.

Souai and Teghem (2009) developed a genetic algorithm based approach for the integrated airline crew-pairing and rostering problem that aimed to manage crew pairing and rostering by dividing the problem into two sub-problems; the airline crew pairing problem which consists of finding a set of trips that cover all the flights planned for a given period of time, and the airline crew rostering problem which consists of assigning the pairings found by solving the first sub-problem to the named airline crew members. The objective is to minimize the total cost.

Teodorovic Lucic (1998) attempted to solve the aircrew rostering problem consisting of the assignment of crew members to plan rotations. The basic algorithm for solving the aircrew rostering problem is a modification of the day-by-day heuristic method. The objective is to find a satisfactory solution that enables all crew members to have an approximately equal workload. In addition, the multi-criteria decision making problem has been solved using fuzzy control methods. The reason is that using fuzzy control methods makes it possible to accommodate qualitative criteria.

Bianco et al. (1992) focused on the problem of planning work schedules in a given time horizon which evenly distributes the workload among the drivers in a mass transit system. An integer programming formulation was applied in this experiment. The objective is to minimize operation cost. This paper implemented a heuristic algorithm that uses a lower bound derived from the mathematical formulation. The computational results show that problems involving 130 duties and a planning period of 7 days may be solved by the Heuristic Rostering Problem (HRP) algorithm in upmost 100 seconds on a personal computer.

Potthoff and Huisman (2010) developed some algorithms for crew re-scheduling. The main Dutch railway operator, NS, periodically changes the schedules in its Operational Control Centers. Currently, there is no decision support at all in these centers. So, this paper constructed a crew scheduling model to support the Dutch railway crew timetable. The objective was to minimize the total costs of the duties. So, this paper applied the very successful heuristic in the crew scheduling package to solve the remaining set covering problem. This heuristic is based on the ideas of Caprara et al. (1999) with some local improvement heuristics.

Dawid et al. (2001) developed a new algorithm that incorporates several strategies that exploit problem-specific knowledge in order to solve even large problems in very short runtimes. The data of this experiment came from real data from a medium-sized European airline. The objective was to minimize total costs and develop the optimal crew time table. Moreover, it uses a branch-and-bound technique to solve real world rostering problems for airline crews. In addition, this paper

implemented the enhanced model with downgrading method to solve European airline crew scheduling. The result outperforms standard optimizers and finds feasible solutions for large-scale crew rostering problems within a reasonable time.

Gamache et al. (2007) developed a graph coloring model for a feasibility problem in monthly crew scheduling with preferential bidding. It also proposed a new methodology based on a graph coloring model and a Tabu search algorithm for determining if the problem contains at least one feasible solution. The objective was to minimize operation cost. The column generation and branch-and-bound techniques were considered. In addition, the mathematical programming and artificial intelligence techniques do not produce an exact solution in pure mathematical terms but appear to be quite adapted to support decision making, by providing, through a comprehensive process, an improved approximation of the set of non-inferior solutions attached to this bi-criterion decision problem. Moreover, this algorithm only needs a few seconds to prove that no backtrack will be needed.

Panta and Dusan (2007) aimed to solve crew rostering problem that an airline usually hire many aircrew members to solve this monthly problem. This experiment emphasizes on managing the pilot schedule. Thus, they applied Simulated Annealing, Genetic Algorithm, and Tabu Searching techniques to solve the crew rostering problem. In conclusion, the experiment can be applied with large scale of problem with several criteria. The results are satisfied with acceptable CPU times.



2.3 Airline Scheduling

Deng and Lin (2011) formulated the airline crew scheduling problem as a Traveling Salesman Problem and introduced an ant colony optimization algorithm to solve it. In addition, the performance was evaluated by performing computational tests regarding real cases as the test problems. The objective was to minimize total crew costs. Traveling Salesman Problems (TSP) and Ant Colony Optimization (ACO) were applied to this paper. The results showed that an ACO-based algorithm can be a potential technique for airline crew scheduling.

Christodoulou and Stamatopoulos (2002) considered the crew assignment problem, which is a sub-problem of the airline crew scheduling problem. The objective of this method is to allocate the crew pairing. Moreover, this study proposed to optimize the allocation of a tested set of crew pairings to crew members in a way that a set of constraints is satisfied. Constraint Logic Programming (CLP) was applied in this paper to support a hybrid scheme combining the features of traditional logic programming and the efficiency of constraint solving. Also, this report proposed a formulation of the crew assignment problem as a constraint satisfaction problem and used a branch-and-bound technique combined with some heuristics in order to find quickly a solution identical, or at least very close to the optimal solution.

Zeghal and Minoux (2006) aimed to solve the Crew Assignment Problem (CAP) that is currently decomposed into two independent sub-problems which are modeled and solved sequentially: the well-known Crew Pairing Problem followed by the Working Schedules Construction Problem. The objective was to minimize the total cost. The implementation of heuristic method provided good solutions in reasonable computation times using CPLEX 6.0.2: guaranteed exact solutions are obtained for 60% of the test instances and solutions within 5% of the lower bound for the others.

Yan et al. (2008) developed a stochastic-demand scheduling model because of the daily passenger demands in actual operations. They employed arc-based and route-based strategies to develop two heuristic algorithms that can be used to solve the model. The objective was to minimize cost. Heuristic algorithms were applied for solving two models: a Stochastic-Demand Flight Scheduling Model (SDFSM), and a Deterministic Demand Flight Scheduling Model (DDFSM). In addition, to solve the SDFSM, based on arc-based and route-based strategies, they developed two solution algorithms. Also, DDFSMS can solve by fixing the selected flights and fleet routes

Weide, Ryan, and Ehrgott (2010) solved the two original problems, the integrated aircraft routing and crew pairing problem, to optimality. Starting from a minimal cost solution, they produced a series of solutions which are increasingly robust. Using data from domestic airline schedules they evaluated the benefits of the approach as well as the trade-off between cost and robustness. They extended their approach considering the aircraft routing problem together with two crew pairing problems, one for technical crew and one for flight attendants.

Weinert and Proksch (1999) tried to solve airline crew pairing by applying a simulated annealing algorithm to the model. They analyzed various ways to improve

the performance of their simulated annealing algorithm. Run time can be saved by using an initial solution which reflects characteristics of the problem and by storing move costs. The permutation based move sampling did not exhibit any advantage. Solution quality can be improved by setting the penalty term for relaxed constraints as low as possible in combination with a post processing routine, summarizing simulated annealing with a specific problem in local improvement heuristic that operates on a larger neighborhood, and the use of multiple independent runs.

Chu (2007) proposed GP models for an integrated problem of crew duties assignment for baggage services section staff at the Hong Kong International Airport. The problem is solved via decomposition into its duties generating phase-a GP planner, followed by its GP scheduling and rostering phase. The results can be adopted as a good crew schedule in the sense that it is feasible, satisfying various work conditions, and “optimal” in minimizing idle shifts.

Halatsis and et al. (2008) proposed the technique to solve the airline crew scheduling problem. They categorized the problem into four types depending on pairing selection, pairing construction, crew assignment, and duty construction. The Constraint Programming was proposed to construct the formulation and model. In conclusion, they received good solution results.

Mercier and Soumis (2005) developed an integrated aircraft routing, crew scheduling, and flight retiming model in order to minimize airline operation cost and construct crew scheduling. This experiment proposed a formulation and Benders decomposition for solving a problem. They experiment on two airline data with seven instances. The result can reduce crew operation cost, and number of operation aircrafts. It also provides a suitable aircraft for a maintenance procedure.

Claude and Nidhi (2007) try to solve crew scheduling problem by combining two main phases; planning, and operation. The planning phase can be divided into two sections; working patterns construction and individual crew assignment. The operation phase will be re-planned but in smaller scale than planning phase. Thus, this experiment shows pairing construction and pairing assignment in a single step method, and solution based on tree search, column generation, and shortest-path algorithm.

Chapter 3

Problem Formulation and Methodology

This section explains airline crew rostering (ACRP) in the case of Thai Airways. Rules and limitations of Thailand's department of civil aviation are included in the section. The methodology also is presented in this chapter.

3.1 ACRP of Thai Airways

Thai Airways is one of the biggest airlines in the world with flights to various destinations such as Asia, Africa, Australia and New Zealand, North America, Europe, and domestically. The Regional Asia routes comprise about 2,036 flights and Thai Airways also has a lot of employees (more than five thousand) which makes ACRP complicated and difficult to solve. Table 3.1 shows details of Thai Airways' crew divisions which are In-flight Manager (IM), Air Purser (AP), First class crew (F), E-business class crew (E), R-business class crew (R), and Economic class crew (Y). Table 3.2 shows various types of aircraft. These aircraft require a different number of crew.

Table 3.1 Thai Airways crew division

Position	Male	Female	Total
IM	171	56	227
AP	208	196	404
F	798	1,271	2,069
E	482	1,111	1,593
R	335	452	787
Y	337	526	963

Table 3.2 Thai Airways fleets

Aircraft Type	No. of Aircraft	No. of Seat
Boeing 747-400: 74R	6	375
Boeing 747-400: 74N	6	374
Boeing 777-300	6	364
Boeing 777-300ER	11	348
Boeing 777-200	8	309
Boeing 777-200ER	6	292
Boeing 787-8	4	264
Boeing 737-400	2	149
Airbus 380-800	6	507
Airbus 330-300: 333	7	305
Airbus 330-300: 330	8	299
Airbus 330-300: 33H	7	299
Airbus 320-200	5	168 / 174
Airbus 340-600	6	266

3.1.1 Test Instances

This section presents experiment data from Thai Airways. The instances data from table 3.4 is composed of start flight day (Day), total block time, adjusted arrival day and departure day (Adjusted ArrDay-DepDay), arrival day and departure day (ArrDay-DepDay), perdiem or salary (TTL THB), and workload score. Adjusted ArrDay-DepDay came from the calculation from Thai Airways. This thesis will convert adjusted arrival day and departure day, and arrival day and departure day into a variable named operation day which includes the number of days consumed in each pairing. Figure 3.1 and 3.2 show detail of flight pairings which consume one operation day.

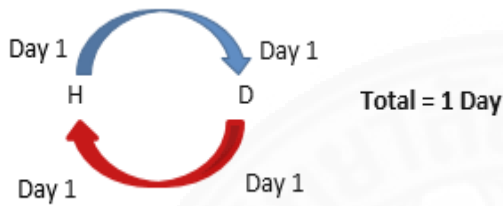


Figure 3.1 Example pairing for one day

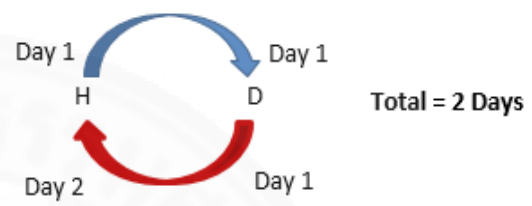


Figure 3.2 Example pairing for two days

Table 3.3 Examples of pairing data

Paring No.	DepFlt	ArrFlt	ArrDay-DepDay	Total Block Time	Adjusted ArrDay-DepDay	Flight Duty Period	Workload	Perdiem
1	315	316	1	8.92	1	10.42	45.92	4795.7616
2	403	404	0	4.58	0	6.08	32.58	1993.66
3	477	478	2	18.6	3	20.1	67.60	10698.1248
4	620	621	0	6.5	1	8	36.50	1998.234

Table 3.4 shows all different instances that are considered in this project. These instances can be divided into three categories and every category further divided into three sub-categories depending on size of pairing and distance of each flight. TA84S represents small pairing with eighty-four flight tasks and short haul which serve regional Asia countries, for example, South Korea, India, Indonesia, Philippines, and Singapore. TA84M represents small pairing and various distance routes of Asia, and Australia and New Zealand. TA84L represents small pairing with forty-five crews and long distance routes to Europe, Australia and New Zealand, and North America. TA150S represents medium pairing with regional Asia routes. TA140M represents medium pairing and various distance routes. TA146L represents medium pairing with long distance routes. TA330S represents the largest number of pairings (330) with short distance routes. TA334M represents the largest pairing with various distance routes. TA238L represents large pairing with long distance routes.

Table 3.4 Test instances

Instance	No. of Pairs	No. of Inflight Managers (IM)	Duration of Pairs	Duration of Schedule
TA84S	84	30	Short haul	14 Days
TA84M	84	30	Various haul	14 Days
TA84L	84	45	Long haul	14 Days
TA150S	150	60	Short haul	14 Days
TA140M	140	65	Various haul	14 Days
TA146L	146	65	Long haul	14 Days
TA330S	330	150	Short haul	14 Days
TA334M	334	150	Various haul	14 Days
TA238L	238	120	Long haul	14 Days

3.2 Methodology

In this thesis, the objective is to solve the crew rostering problem in Thai Airways by using the Greedy Algorithm. The objective is to manage the crew scheduling alignment with workload balancing and perdiem balancing, simultaneously with condition of flight time limitation. Workload means the service cost of crew duty for each flight leg. Perdiem is the value or money that consequently results from workload, while flight time means the time that an aircraft stands by at the departure airport to landing at the destination airport. Moreover, the flight time and rest period are crew pairing constraints, which are declared by the Department of Civil Aviation of Thailand. The flight time limitation is set as follows:

- Every 7 days must have flight time less than 34 hours.
- Every 28 days must have flight time less than 110 hours.
- Every 365 days must have flight time less than 1,000 hours.

Table 3.5 Rest period constraint

Flight Duty Period	Rest Period
0 - 8 hours	≥ 8 hours
8 - 10 hours	≥ 10 hours
10 - 12 hours	≥ 12 hours
12 - 14 hours	≥ 14 hours
14 - 16 hours	≥ 16 hours
16 - 20 hours	≥ 24 hours

We can construct the crew pairing by using table 3.5 for implementation of crew rostering. From the above information, we can conclude that the flight time per week must be less than 34 hours. Constructing the experiment using the Greedy Algorithm involves the following steps. This experiment is defined to test on fourteen days. The solution will result in the form of crew time table shown in table 3.6. In the table, a row represents working day $D = \{D_1, D_2, \dots, D_{14}\}$, flight duty, workload and per diem. Columns represent number of crew $C = \{C_1, C_2, \dots, C_n\}$ in each experiment. The number inside the table should be pairing id $P = \{P_1, P_2, \dots, P_n\}$;

Table 3.6 Crew time table format

	Day 1	Day 2	...	Day n	Workload	Per diem
Crew 1						
Crew 2						
...						
Crew n						

The objective is to align the crew roster with workload and perdiem balancing. The method is minimization of Standard Deviation (SD). SD is used to measure variation of value and a solution will be optimal when the value of SD is close to zero. Equation 3.1 shows the SD formulation.

$$\sigma = \sqrt{\frac{\sum(x-\bar{x})^2}{N-1}} \quad (3.1)$$

Where;

σ = Standard Deviation

x = Value of workload or perdiem

\bar{x} = Mean of workload or perdiem

N = Total number of workload or perdiem

This thesis primarily focuses on three scenarios: minimizing SD of workload and perdiem simultaneously without bound, minimizing SD of workload and perdiem simultaneously with workload bound, and minimizing SD of workload and perdiem simultaneously with perdiem bound. This thesis also examines minimizing only workload and perdiem. The experiment is constructed as two different phases for creating a crew schedule: a Construction Phase and an Improvement Phase. The Construction Phase involves creating a crew schedule or crew time table by assign pairing instances into a schedule under constraint conditions. The Improvement Phase aims to reduce target SD by applying a Greedy Algorithm. The improvement phase can be divided into four main techniques:

1. Change pairing directly
2. Change pairing descending
3. Change pairing ascending
4. High crew distribution

Due to the significantly different values of SD of workload and perdiem, the method to reduce both SD of workload and perdiem together is normalization. The instance values of workload and perdiem will normalize after being imported to the program. After normalization, SD of workload and perdiem will combine to be the target SD. The normalization and combination method are formulated in equation 3.2 as follows:

$$Twp_c = \frac{W_c}{M_w} + \frac{P_c}{M_p} \quad (3.2)$$

Where;

T = Total normalized workload and perdiem value

M_w = Maximum value of workload

M_p = Maximum value of perdiem

c = Index of workload and perdiem where $c = 1, 2, \dots, x$

W = Workload of pair c

P = Perdiem of pair c

The normalization is not used when minimizing only workload and perdiem. It is applied in the construction phase for minimizing workload and perdiem simultaneously in order to standardize both values. The reason that normalization is not applied to minimize only workload and perdiem is to prove the related significance between workload and perdiem.

Chapter 4

Greedy Algorithm for Solving ACRP Problem

4.1 Greedy Algorithm

A Greedy Algorithm is the method to solve the problem by choosing the obvious and immediate benefit. Usually, a Greedy Algorithm is proper for simple problems. Since a Greedy Algorithm does not support reconsideration of data, it has probability to find only a local optimal solution but not a global optimal solution. Normally, the components of a Greedy Algorithm are a candidate set, a selection function, a feasibility function, an objective function, and a solution function. Also, there are many methods to solve the optimal solution problem in form of Greedy Algorithm such as, Huffman encoding, Minimum spanning tree, Traveling Salesman Problem, Dijkstra's Algorithm, Kruskal's Algorithm and Prim's Algorithm.

The simple process of the Greedy Algorithm can be explained according to Figure 4.1. The objective of this problem is to exchange \$27 by using the least number of coins. The sequence of change is highest value of coins first. The variable coins for the change are \$10, \$5, and \$2 coins. Normally, a Greedy Algorithm must use the most beneficial way to solve the problem. Thus, the method in the solution is to change two \$10 coins, one \$5 coin, and one \$2 coin back. So, this simple problem can be solved very easily.

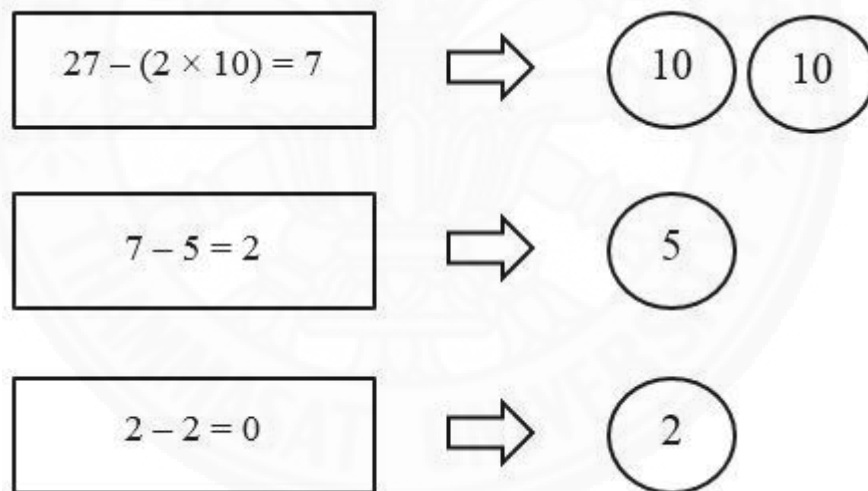


Figure 4.1 Example of good Greedy Algorithm

In Figure 4.2, the objective is to exchange \$21 with the same coins values from Figure 4.1, i.e., \$10, \$5, and \$2. The Greedy Algorithm cannot solve this problem because it will find only the way to exchange money with minimum coins, thus, there will be a remainder of \$1. If the solution used the human method or other algorithm, the solution will solve completely as shown in Figure 4.3. The method is to exchange one \$10, one \$5, and three \$2 coins.

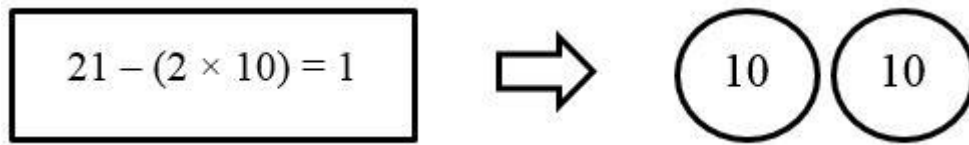


Figure 4.2 Example of problem that Greedy Algorithm cannot solve

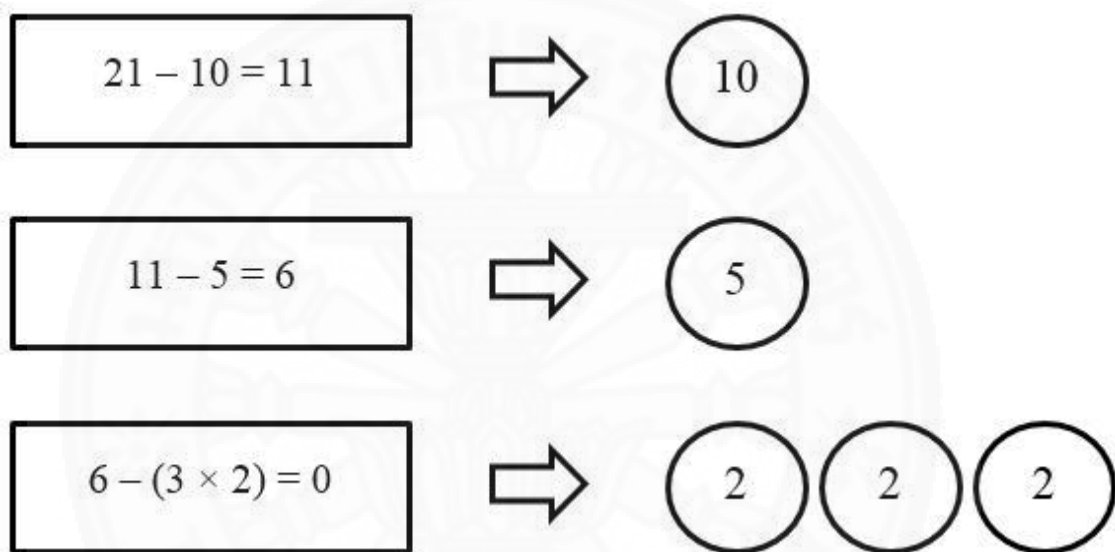


Figure 4.3 Solution of exchange money

4.2 Solution Model

This section presents a crew rostering approach that is implemented using a Greedy Algorithm. C language was introduced as a tool for formulating the solution. This language runs on Microsoft Visual C++ 2010 Express compiler. This experiment aims to minimize both workload and perdiem simultaneously. The experiment can be divided into two phases: construction and improvement. The purpose for the construction phase is to create a simple crew schedule without being concerned about SD. Thus, the improvement phase aims to reduce SD of workload and perdiem. The Improvement Phase can be divided into application of four techniques as follows:

- Change pairing directly
- Change pairing descending
- Change pairing ascending
- High crew distribution

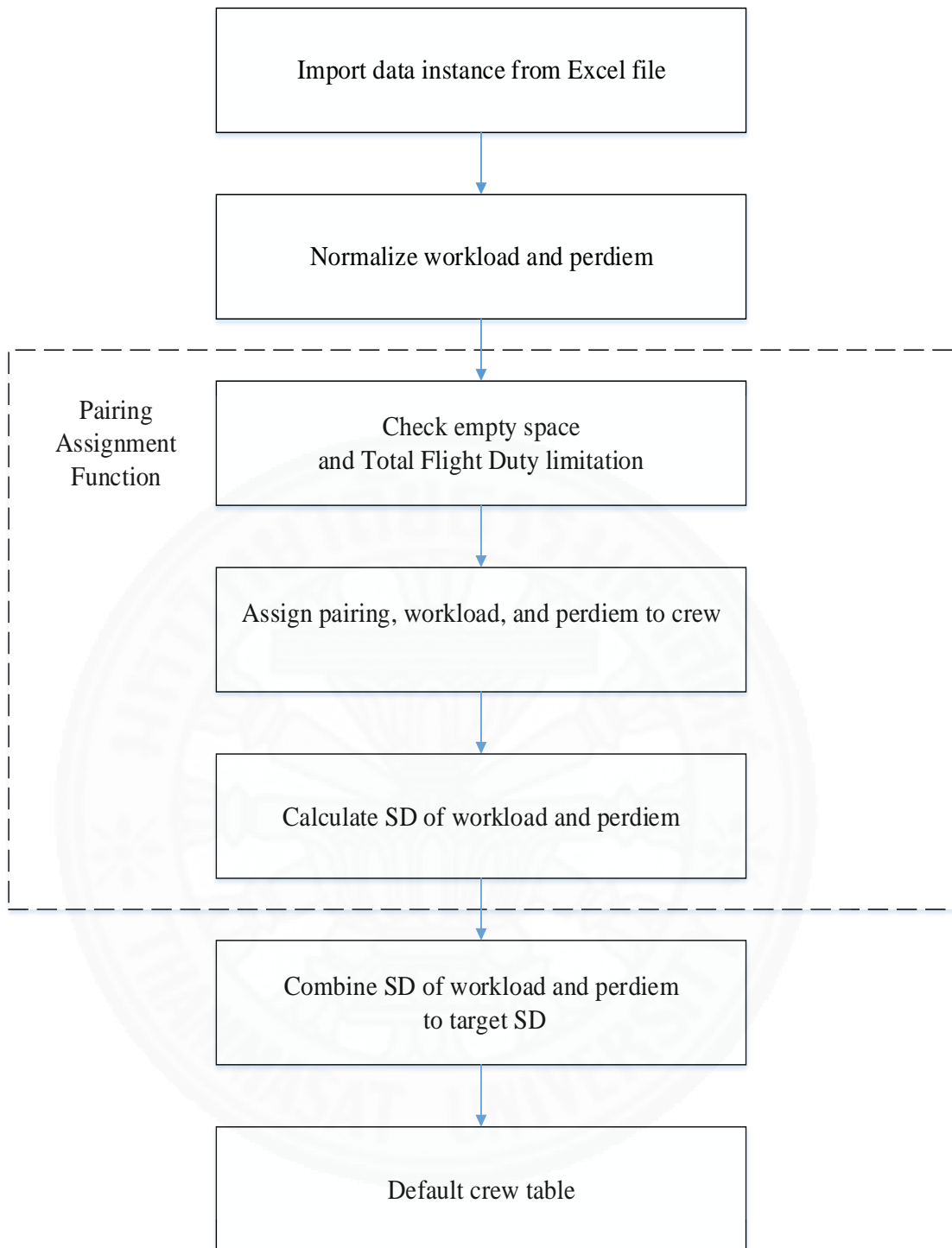


Figure 4.4 flow chart of construction phase

4.2.1 Construction Phase

This process constructs a crew schedule or crew time table while considering any limitations or constraints. One constraint is total flight duty hours per week. This process will create a fourteen-day crew table. The rows consist of day in week, SD of workload, and SD of perdiem. Columns consist of the number of crew which depends on each instance. The construction phase involves a set of crew $C = \{C_1, C_2, \dots, C_i\}$ where i is maximum of crews in each instance, day of work $D = \{D_1, D_2, \dots, D_i\}$, start working day $Std = \{Std_1, Std_2, \dots, Std_i\}$, and pairing $P = \{P_1, P_2, \dots, P_i\}$. In Figure 4.4, the steps of construction are presented as follow:

Step 1: Convert Thai Airways data from excel file to text file. The test data include start of day (Day), operation day, total flight duty or total block time, perdiem, and workload. All data is separated by “|” instead of “,” because commas may cause errors in later processes.

Step 2: Import data text file into the program. These data are collected in structure name “pair[]” which consists of variables; day, operate_day, fduty, perdiem, and workload.

Step 3: Normalizes value of workload and perdiem create total normalized workload and perdiem by using equation 3.2.

Step 4: Construct pairing that loops from P_1 to P_i . Then, check available day or empty task day of crew C_1 to C_i which have day D_x that matches with start working day Std_x of pair P_x .

Step 5: If a matched working day slot is found, assign id number of pairing P_x in to crew schedule. In addition the size of each pairing must be equal to size of the operation day. That means if P_x has operation day about three days and C_x has matched available working day at D_1 , program will book P_x from D_1 to D_3 . In pairing assignment process, value of total flight duty, workload, and perdiem are assigned to each crew. The program also checks total flight duty limitation every time before booking. After completing the assignment process, the program will focus on the next pairing, P_{x+1} , until it finishes P_i .

Step 6: If the pairing assignment process does not complete, the program will delete total flight duty, workload, and perdiem in the specific crew information. Then, it will processes to focus new available crew from C_{x+1} until C_i .

Step 7: After it completes assigning all pairings to crews, the program will calculate SD of both workload and perdiem to create a target SD. The target SD will be setup to be the default SD for comparison in the improvement phase. All values and information in the completed construction crew table are also setup to be the default table.

4.2.2 Improvement Phase

The purpose of this phase is to reduce Standard Deviation (SD) of workload and perdiem by using the target SD and applying a Greedy Algorithm. The main idea of this phase is compare each target SD with the default SD and choose lowest SD for setup to be initial or default SD. The table that has the selected target SD is also setup to be the initial or default table. The first default table came from construction phase. The default table always changes after finishing the improvement process.

This experiment also created workload and perdiem bound limitations. The process to limit workload bound occurred after a program sorted the workload at first iteration, the result was limited by the value of minimum and maximum total workload for all of the next iterations. The iterations after first iteration will be sorted by perdiem. The perdiem bound limitation is similar to the workload bound limitation. The process to limit perdiem bound occurred after a program sorted the perdiem at first iteration, the result was limited by the value of minimum and maximum total perdiem for all of the next iterations. The iterations after first iteration will be sorted by workload.

This thesis presents four methods of SD reduction. The techniques of improvement phase are presented as follow:

- Change pairing directly
- Change pairing descending
- Change pairing ascending
- High crew distribution

4.2.2.1 Change pairing directly

This technique presents simple SD reduction by changing pairing positions of every crew in the crew schedule. The idea is to find minimum SD for every one step changed. The solution will continuously decrease SD to meet the local optimal solution. In the worst case, this technique may end after a single iteration if there is no lower SD to select. The reason for this comes from the nature of a Greedy Algorithm to always select the smallest value.

This technique will change selected pairing P_x for every available crew $C_1 - C_i$. After one step of pairing change, the program will calculate the target SD of the whole crew table to find the minimum target SD. This process continues until it has finished moving all of pairing $P_1 - P_i$. The iteration can be assigned for more efficient target SD minimization. From figure 4.5, the target SD of workload minimization is SD of workload. Figure 4.6 shows the target SD of perdiem minimization is SD of perdiem. Figure 4.7, 4.8, and 4.9 shows the SD of workload and perdiem as a target SD.

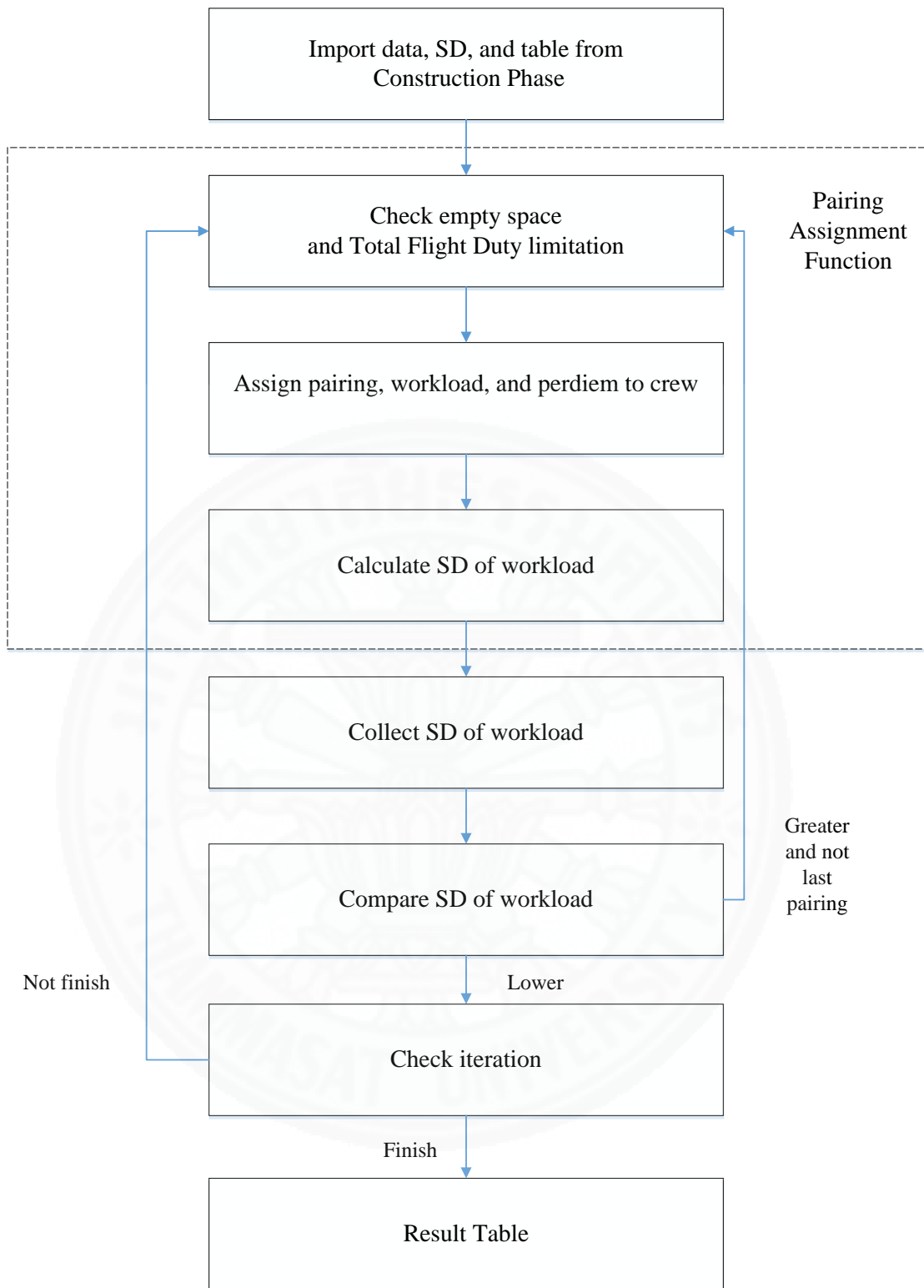


Figure 4.5 flow chart of change pairing directly with workload minimization

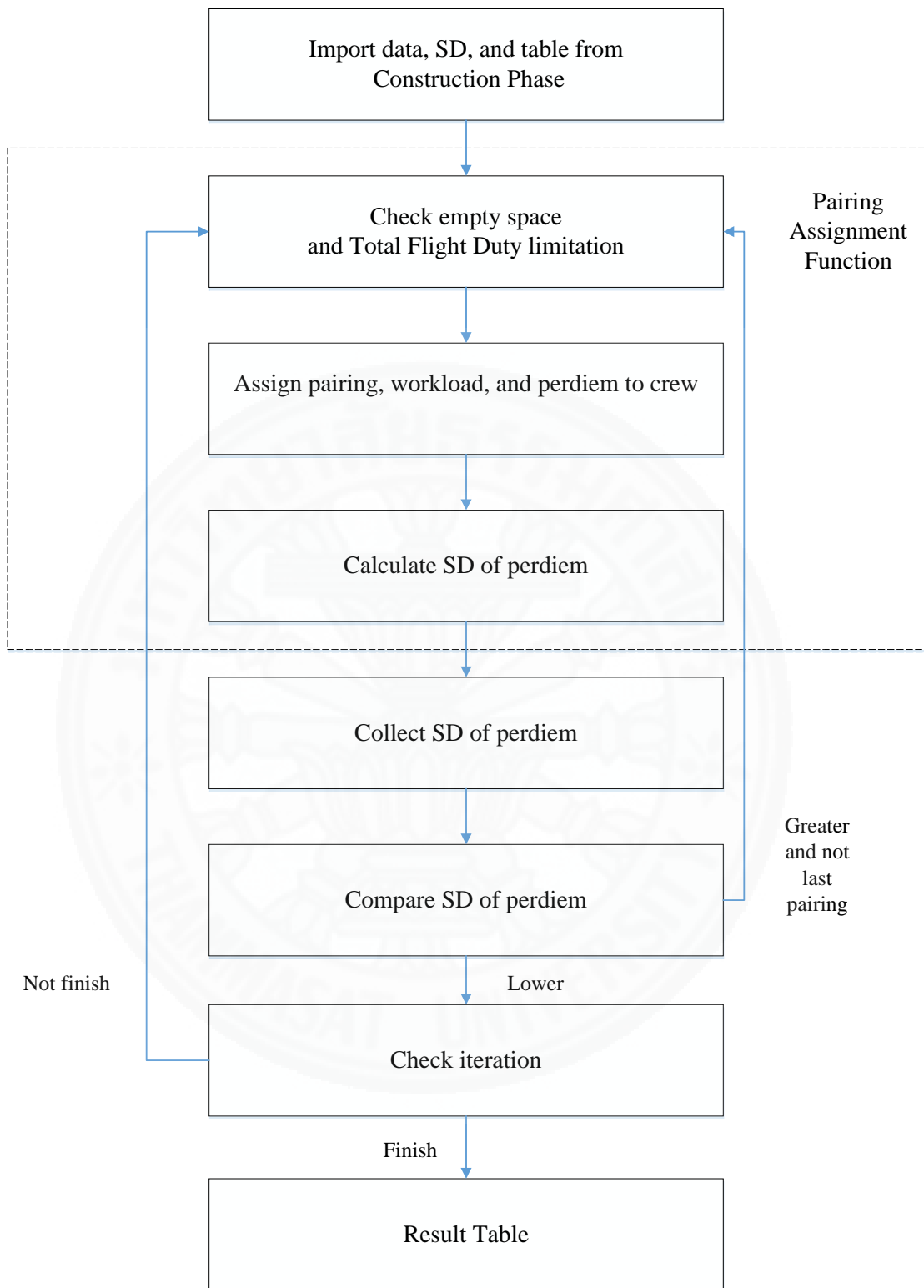


Figure 4.6 flow chart of change pairing directly with perdiem minimization

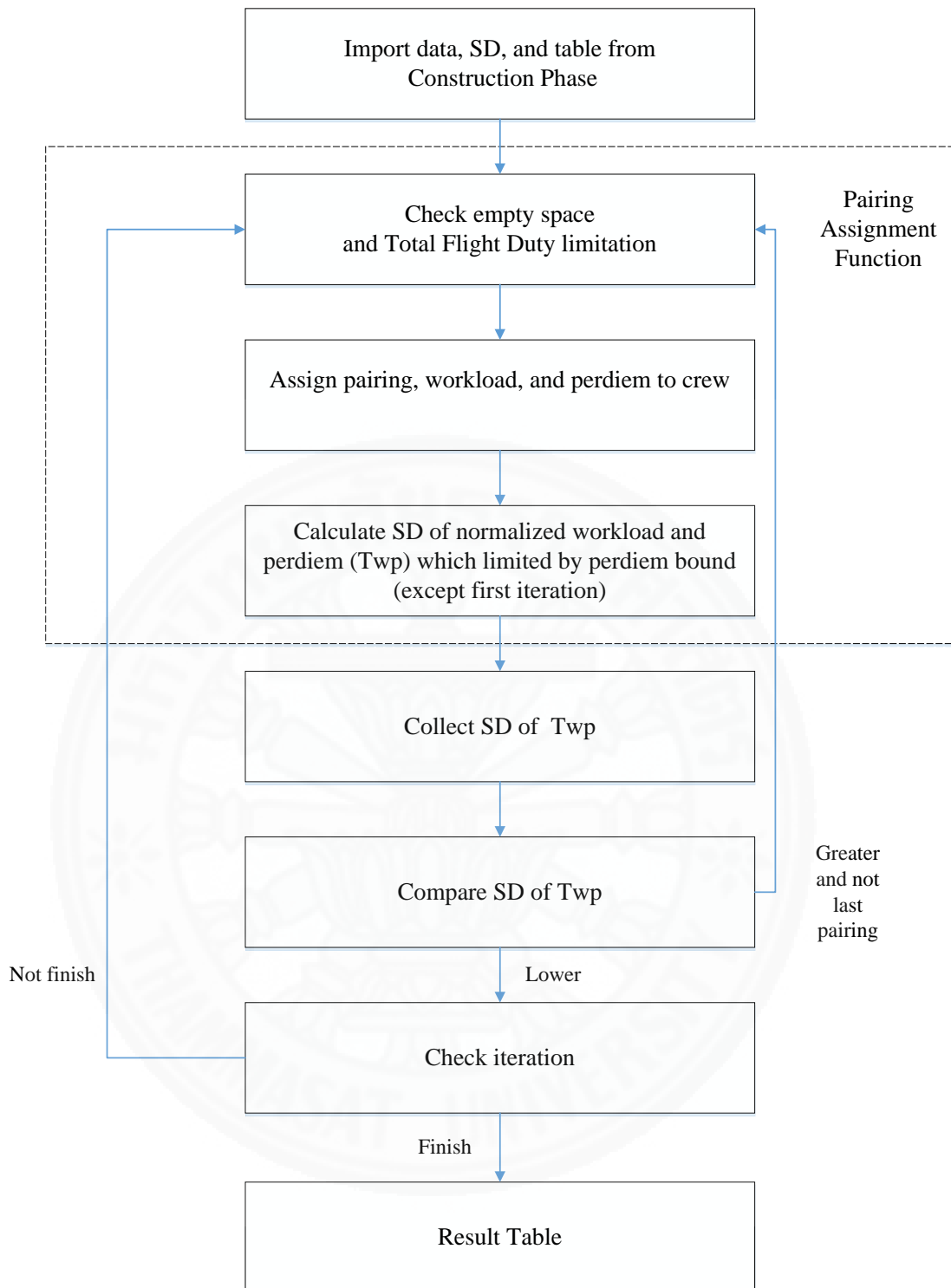


Figure 4.7 flow chart of change pairing directly with workload and perdiem minimization simultaneously without bound

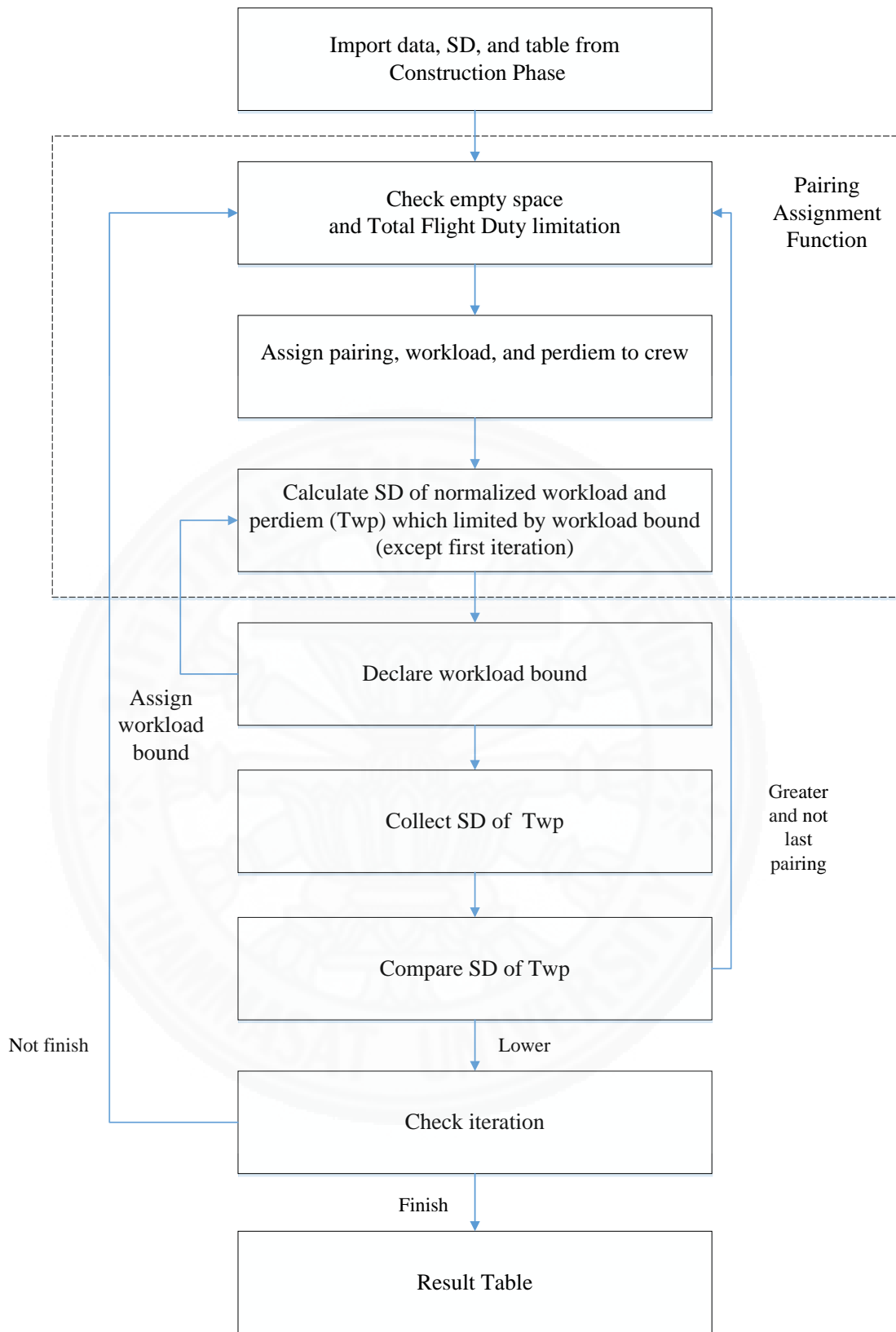


Figure 4.8 flow chart of change pairing directly with workload and perdiem minimization simultaneously with workload bound

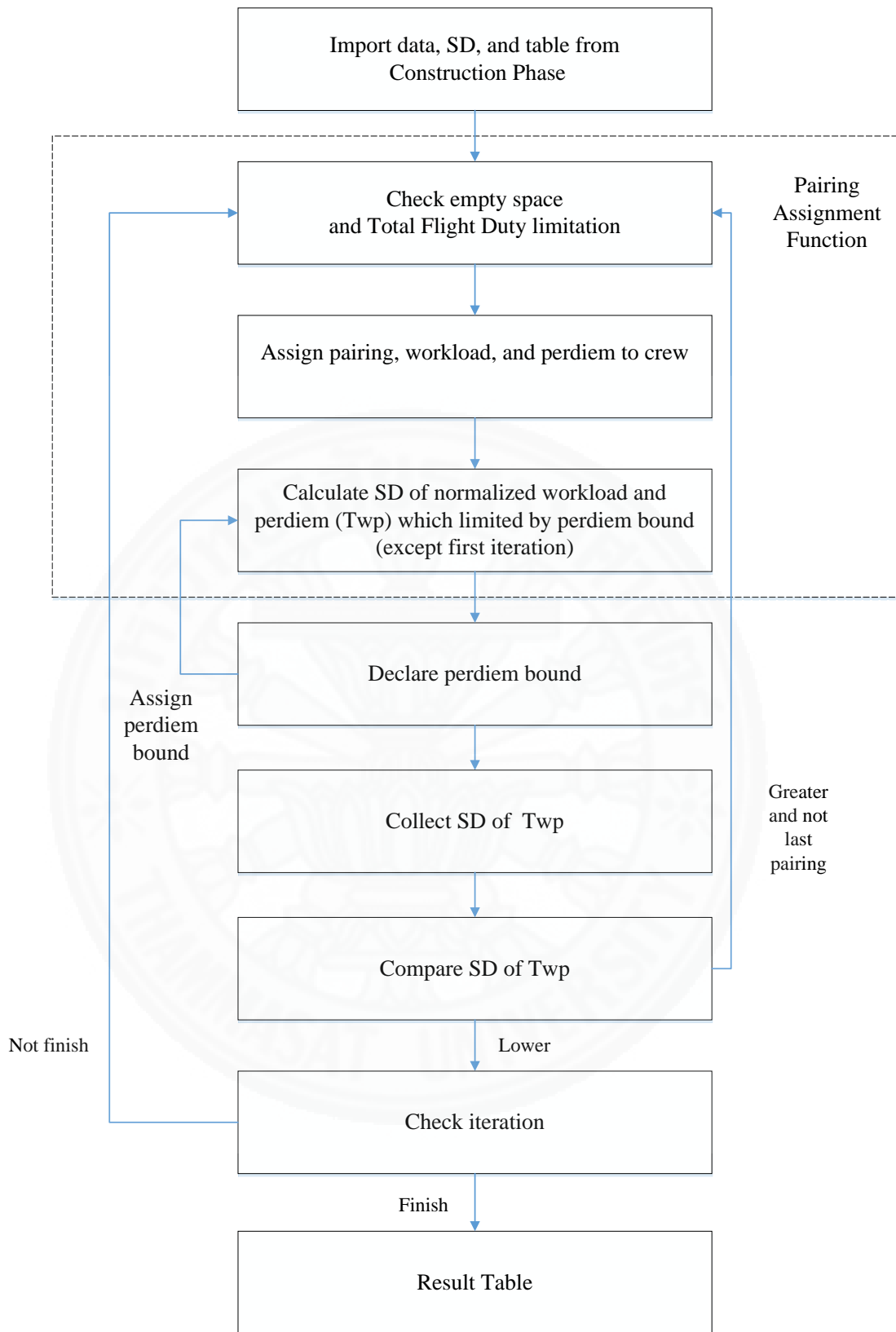


Figure 4.9 flow chart of change pairing directly with workload and perdiem minimization simultaneously with perdiem bound

Step 1: This step needs to import data from the construction phase, such as summation of workload value, per diem value, and total block time period for workload minimization, and per diem minimization, and summation of normalized workload value, normalized per diem value, and total flight duty period for workload and per diem minimization simultaneously without bound, workload and per diem minimization simultaneously with workload bound, and workload and per diem minimization simultaneously with per diem bound. The target SD from construction phase which is the SD of whole crew table is also included. The SD of construction phase set to be the default comparison SD with other new improvement phase. The construction phase crew table also is setup to be the default table for the pairing changing process.

Step 2: This process aims to reduce SD from construction phase. The method is to move pairing P_x to every available crew $C_1 - C_i$. In the first step, the program will check the crew C_{x+1} , where x is crew id that stores pairing P_x , to see if it matches the available size of P_x operation day or not. This process also checks total flight duty limitation. If matched, the program will remove the old P_x at crew C_x . Total flight duty, total workload, and total per diem also are removed from C_x information. Then the program will book the value of P_x and assign value of flight duty, workload, and per diem to C_x information.

After finishing booking, the program will proceed to step 3 to calculate SD of workload for workload minimization, SD of per diem for per diem minimization, and SD of total normalized workload and per diem (Twp) for workload and per diem minimization simultaneously with workload bound, per diem bound, and without bound. But if booking is not complete, the program will focus on the next crew C_{x+2} and execute step 2 again. This process will continue until the end of crew C_{x-1} .

Step 3: In this step, the program calculates a whole crew table to find the target SD. The target SD will change every time that a pairing is moved because the total workload and per diem value of each crew is changed. The calculated target SD after first iteration must less than bound limitation. The calculated target SD is stored in array parameter for comparison in the next step. From figure 4.8, the highest and lowest SD of workload will declare to be the limitation of workload bound. From figure 4.9, the highest and lowest SD of per diem will declare to be the limitation of per diem bound.

Step 4: The objective of this step is to find a minimum target SD. This step occurs after the check available function result is true. The array of target SD will be compared with initial or default SD. After the comparison process, the minimum target SD will be set to be the default SD also with minimum target SD table.

4.2.2.2 Change pairing descending

This technique is similar to the previous Change Pairing Directly technique. The additional process is total workload and perdiem descending. The idea is to distribute or reduce high combined total workload and perdiem of crew C_x . The method is also similar to the previous technique by sorting descending target SD first and then continuously moving or changing position of pairing with high combined total workload and perdiem to others crews. The solution will continuously decrease target SD to meet the local optimal solution.

This technique will change selected pairing P_x for every available crew $C_1 - C_i$. After one step of pairing change, the program will calculate the target SD of the whole crew table and set the minimum SD to be the default SD for comparison. This process continues until it has finished moving all of pairing $P_1 - P_i$. The iteration can be added for more efficient target SD minimization. From figure 4.10, the target SD of workload minimization is SD of workload. Figure 4.11 shows the target SD of perdiem minimization is SD of perdiem. Figure 4.12, 4.13, and 4.14 shows the SD of workload and perdiem as a target SD.



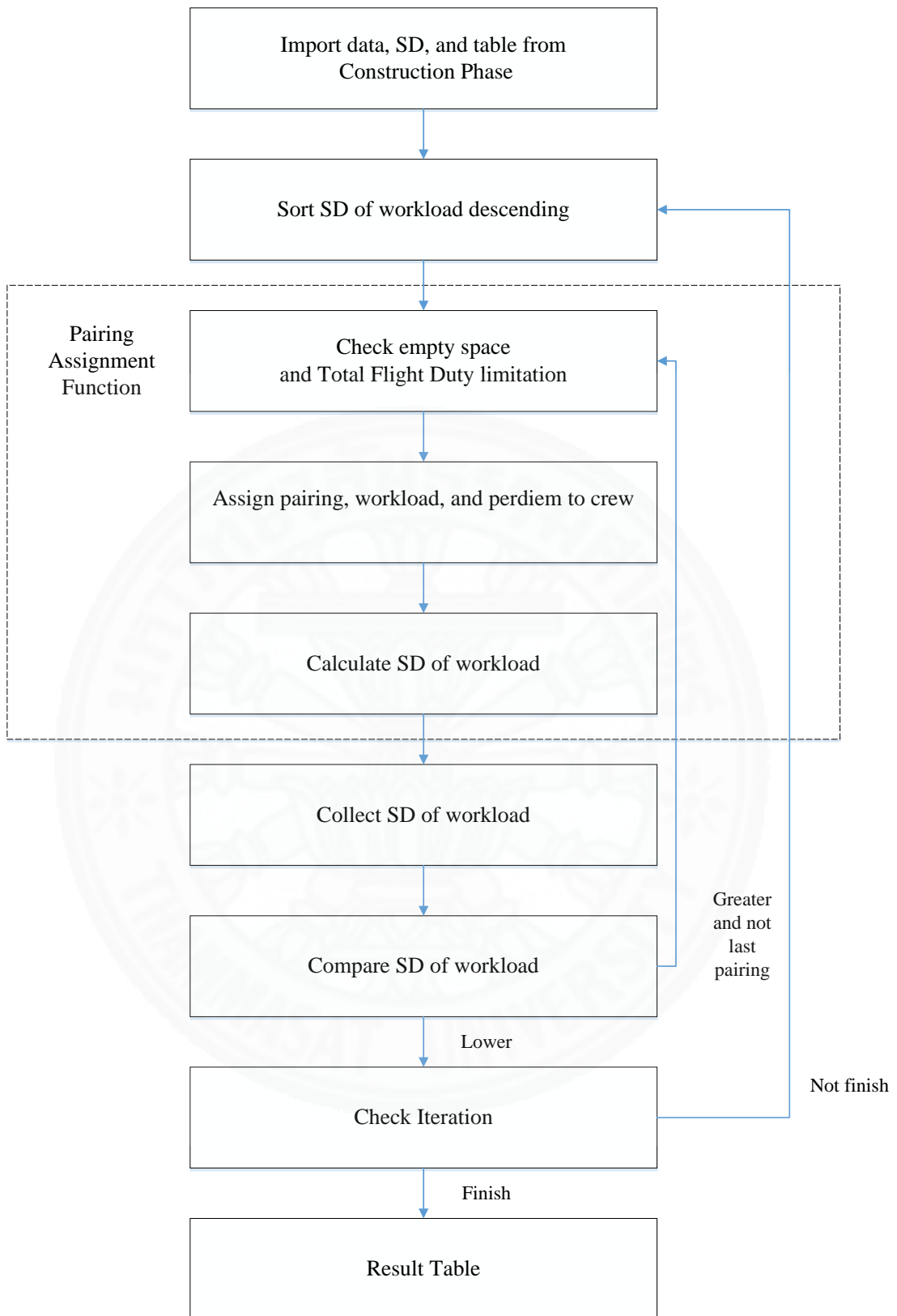


Figure 4.10 flow chart of change pairing descending with workload minimization

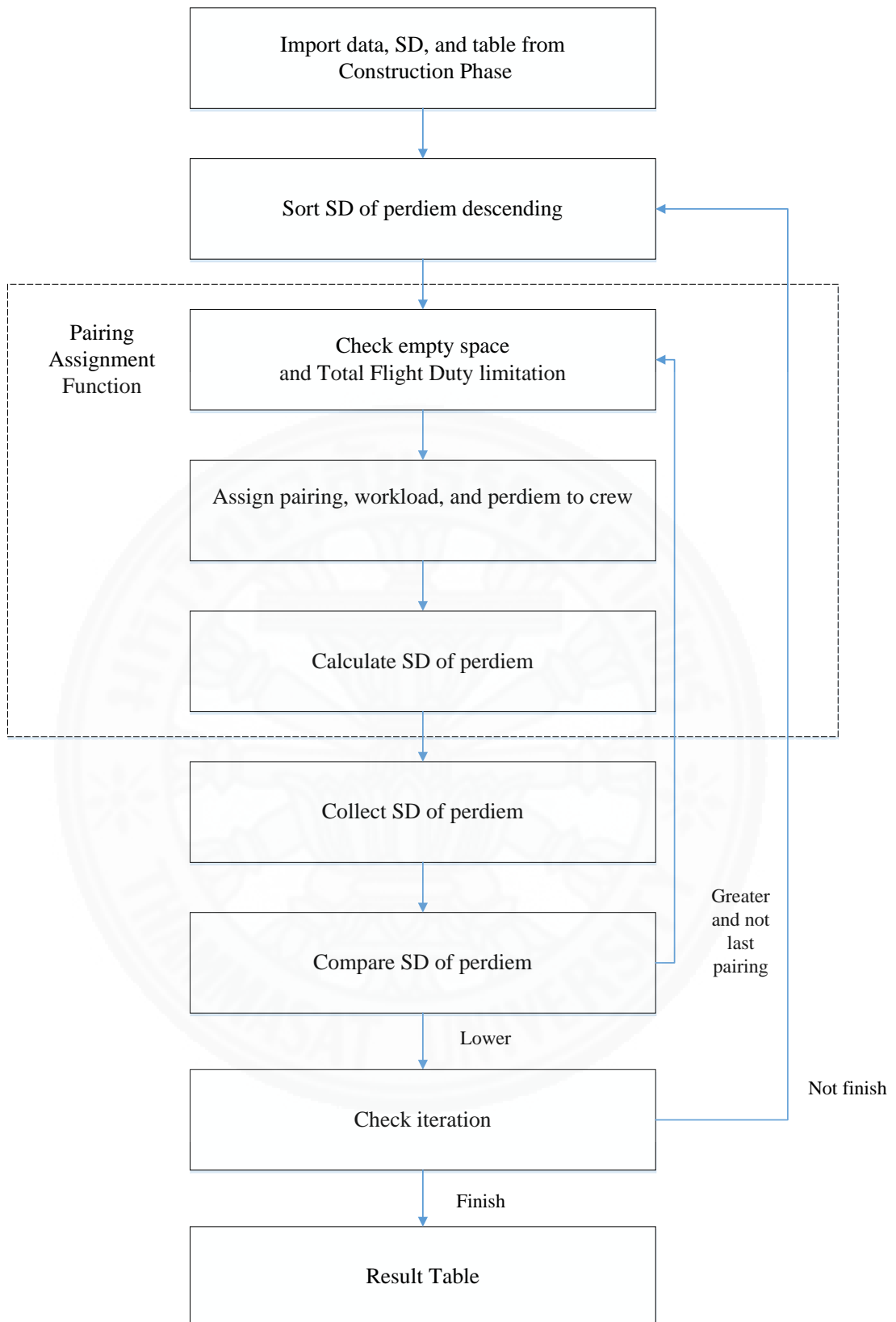


Figure 4.11 flow chart of change pairing descending with perdiem minimization

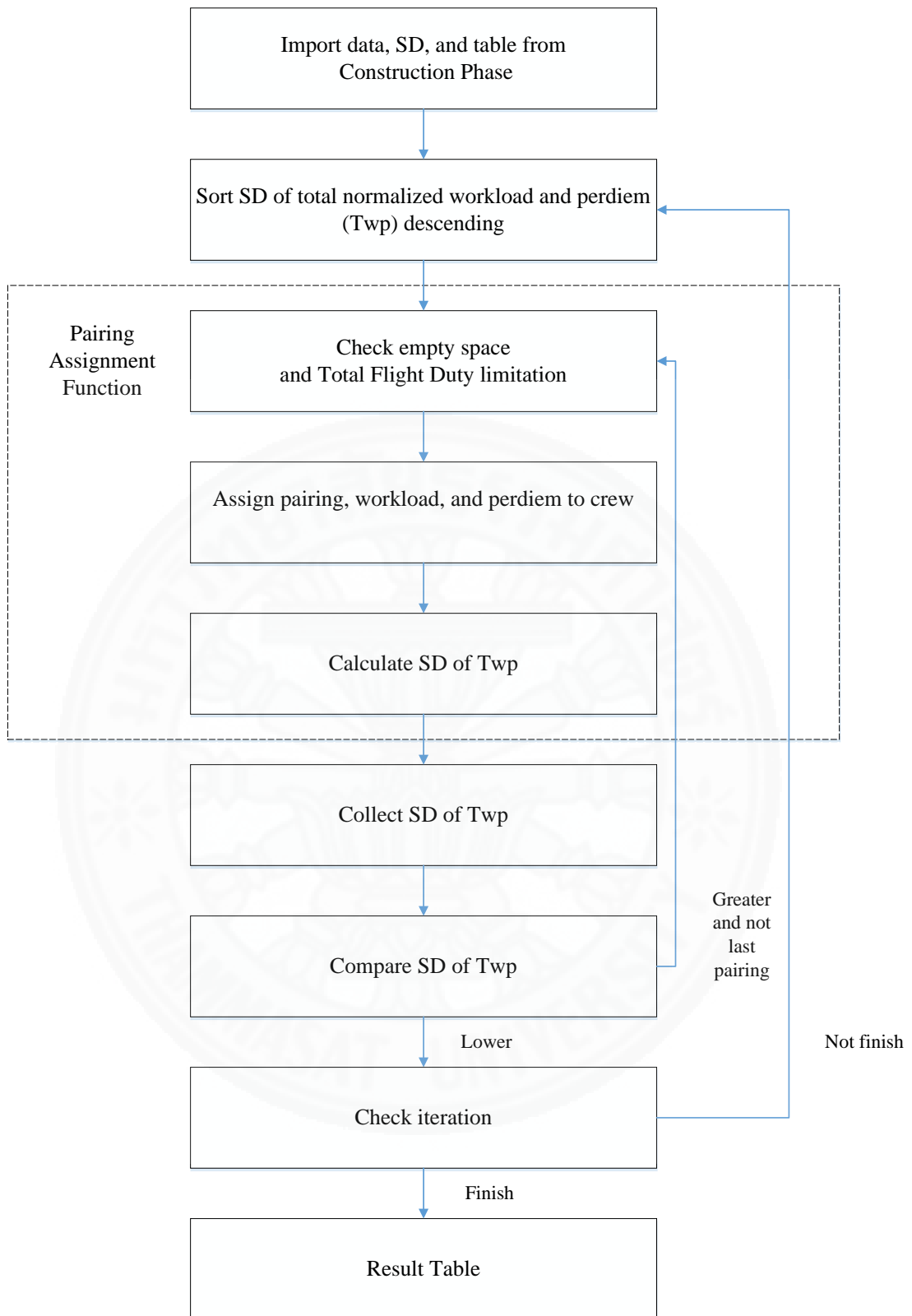


Figure 4.12 flow chart of change pairing descending with workload and perdiem minimization simultaneously without bound

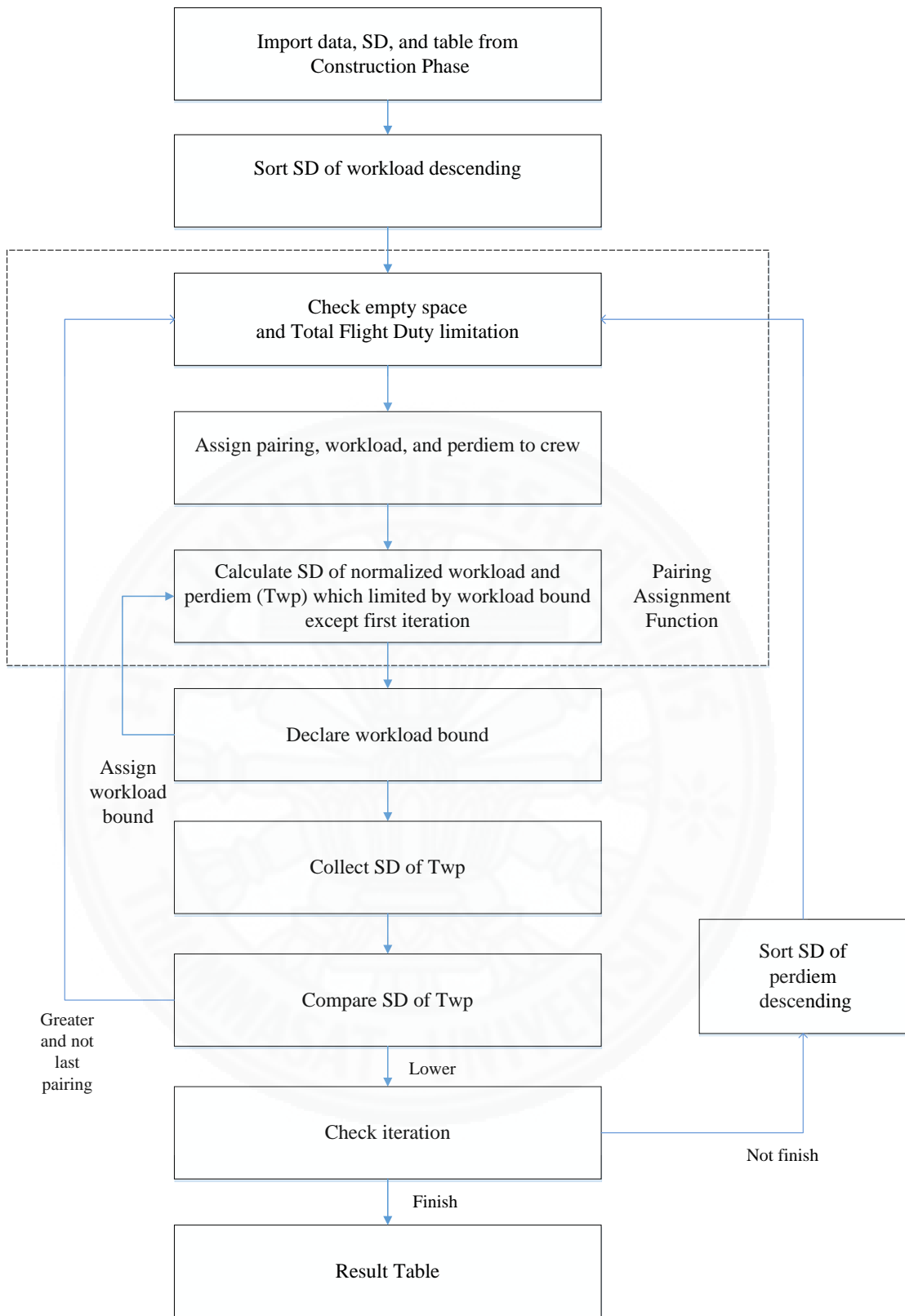


Figure 4.13 flow chart of change pairing descending with workload and perdiem minimization simultaneously with workload bound

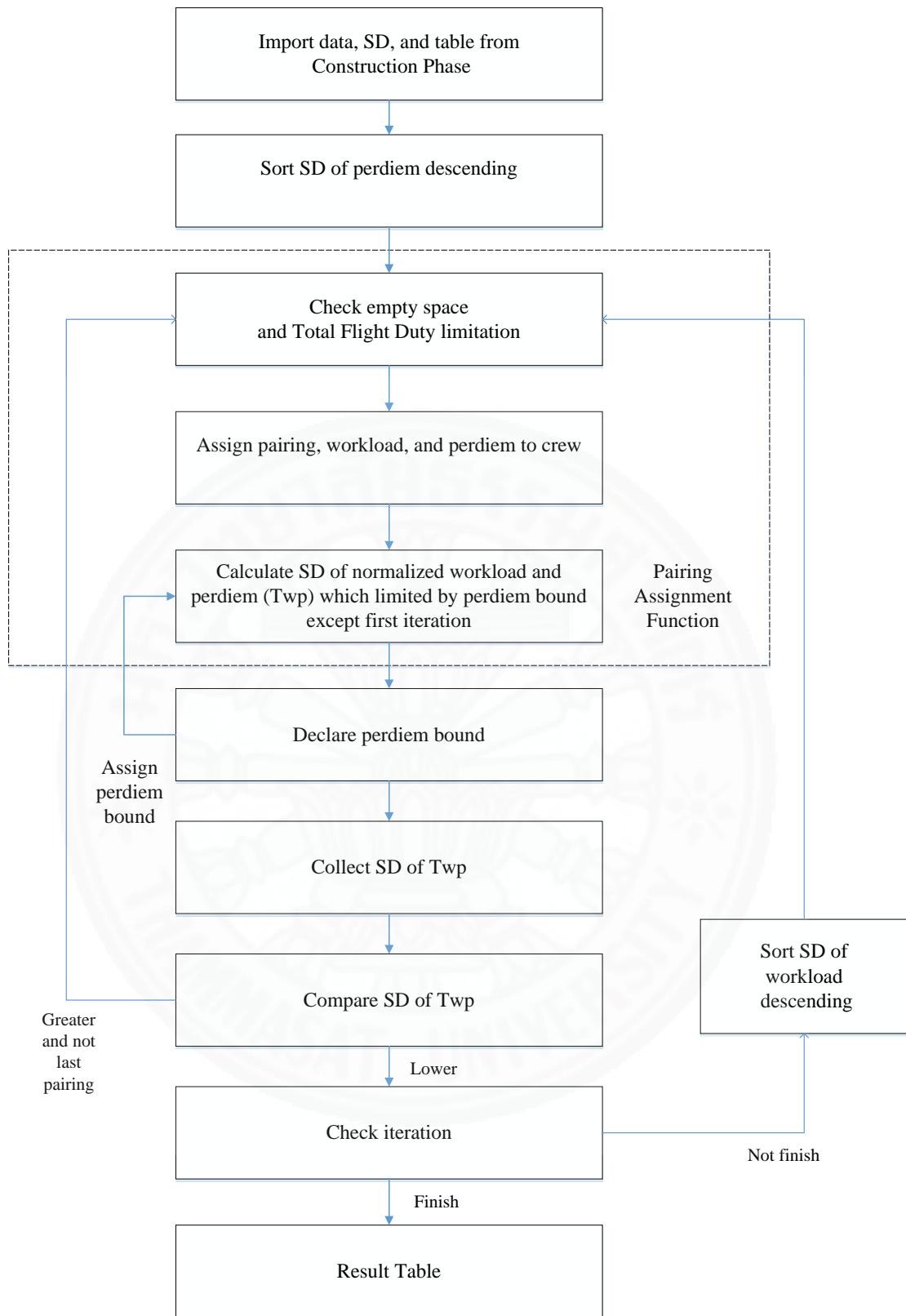


Figure 4.14 flow chart of change pairing descending with workload and perdiem minimization simultaneously with perdiem bound

Step 1: This step needs to import data from the construction phase, such as summation of normalized workload value, normalized perdiem value, and total flight duty period from all of crews. The target SD from the construction phase also included. The SD of the construction phase is used to be the default comparison SD with other new improvement phase creating. The construction phase crew table is also setup to be the default table for pairing changing process.

Step 2: This step will sort target SD in descending order. From figure 4.10, the workload minimization will sort by SD of workload. Figure 4.11 shows perdiem minimization that used SD of perdiem for sorting. The no bound part will use the summation normalized SD of workload and perdiem (Twp) according to figure 4.12. From figure 4.13, the workload bound will sort by SD of workload in the first iteration and perdiem in the remaining iterations. Similarly, for perdiem bound, it will sort by SD of perdiem in first iteration and workload in the remaining iterations according to figure 4.14. Thus, the sequence of crew will be changed but the information of each crew is still the same.

Step 3: This process aims to reduce target SD. The method is move pairing P_x to every available crew $C_1 - C_i$. In the first step, the program will check the crew C_{x+1} , where x is the crew id that stores pairing P_x , to see if it matches the available size with P_x operation day or not. This process also checks total flight duty limitation. If matched, the program will remove the old P_x at crew C_x . Total flight duty, total workload, and total perdiem also are removed from C_x information. Then the program will book the value of P_x and assign the value of flight duty, workload, and perdiem to C_x information. After finishing booking, the program will proceed to step 4 to calculate target SD. If not finished, the program will focus on the next crew C_{x+2} and execute step 3 again. This process will continue until the end of crew C_{x-1} .

Step 4: In this step, the program calculates a whole crew table to find the target SD. The target SD will change every time that a pairing is moved because the total workload and perdiem value of each crew is changed. The calculated target SD is stored in an array parameter for next step comparison.

From first iteration, the highest and lowest SD of workload will declare to be the limitation of workload bound. And the highest and lowest SD of perdiem will declare to be the limitation of perdiem bound. These bound will applied in step 3 in the rest iterations.

Step 5: The objective of this step is to find the minimum target SD. This step occurs after check available function result is true. The array of target SD will be compared with initial or default SD. After finishing the comparison process, the minimum target SD will be set to be default SD also with minimum target SD table.

4.2.2.3 Change workload ascending

This is method is similar to previous techniques by applying target SD ascending sorting. The idea is to find and analyze the differences between unsort, descending sort, and ascending sort techniques. This method requires the target SD from construction phase for sorting. This technique also moves all pairings $P_1 - P_i$ to all crews $C_1 - C_i$. The result should be different because the sequence of crews was changed and, thus, the total SD of workload and perdiem of each crew will changed.

From figure 4.15, the target SD of workload minimization is SD of workload. Figure 4.16 shows the target SD of perdiem minimization is SD of perdiem. Figure 4.17, 4.18, and 4.19 shows the SD of workload and perdiem as a target SD.



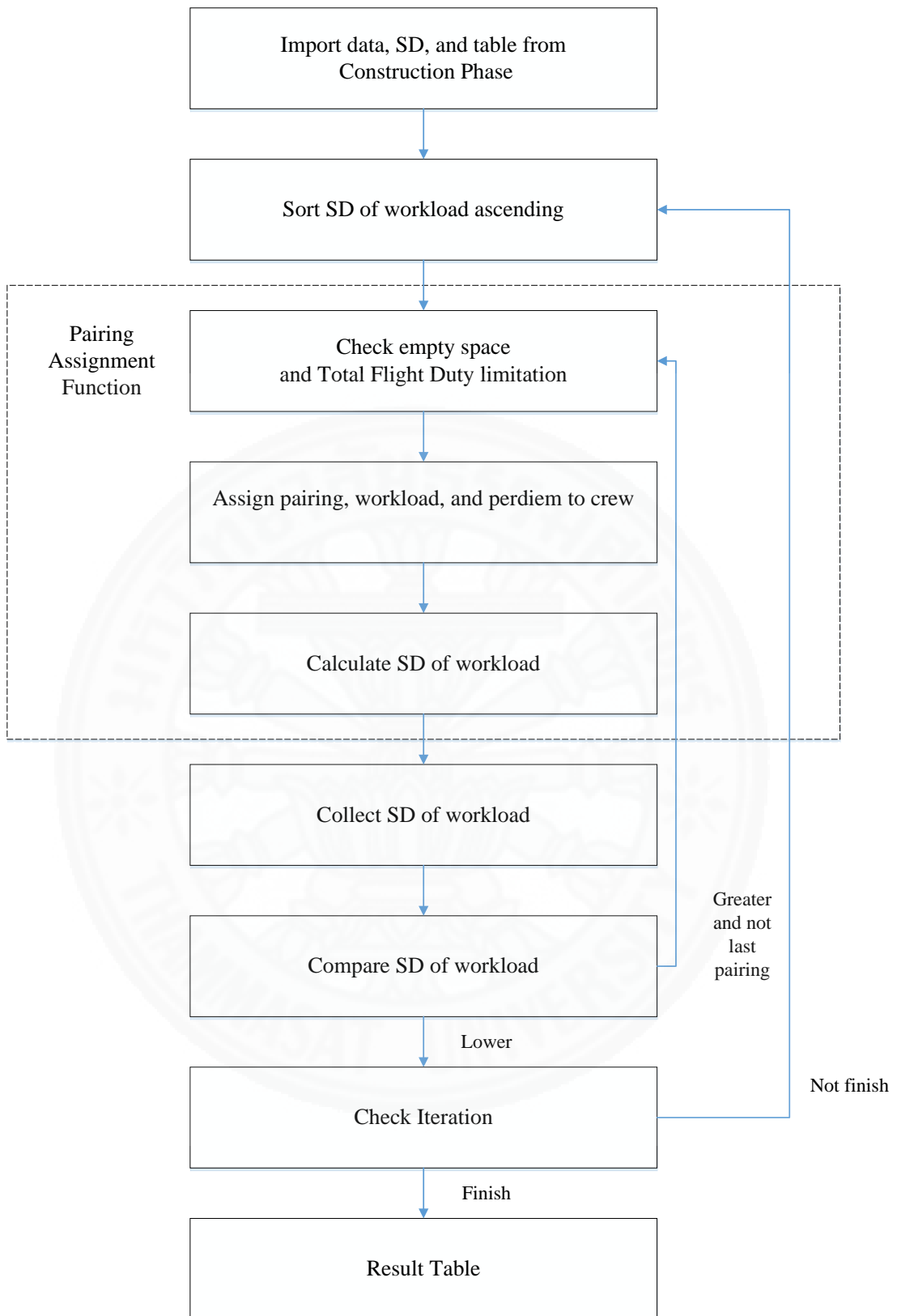


Figure 4.15 flow chart of change pairing ascending with workload minimization

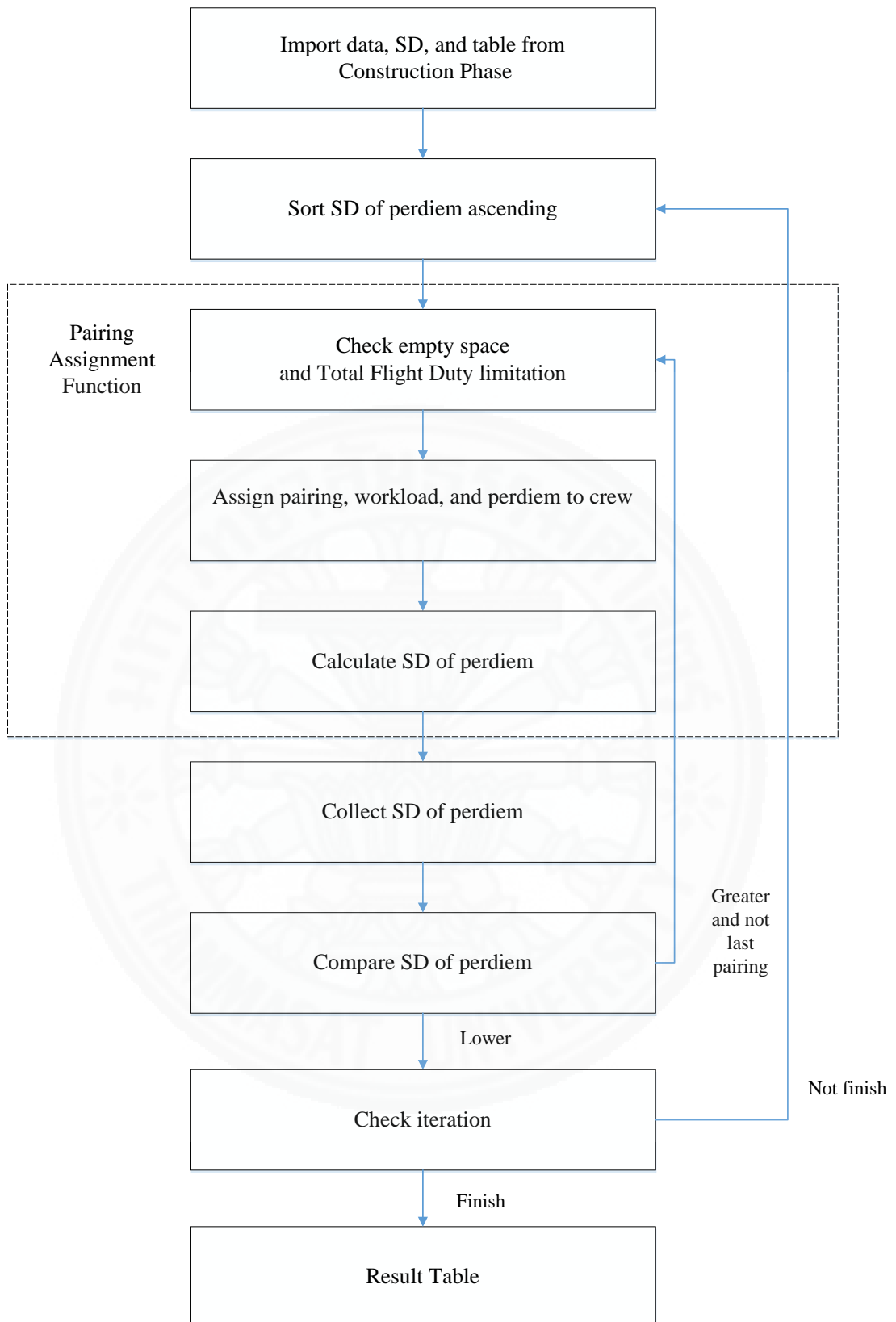


Figure 4.16 flow chart of change pairing ascending with perdiem minimization

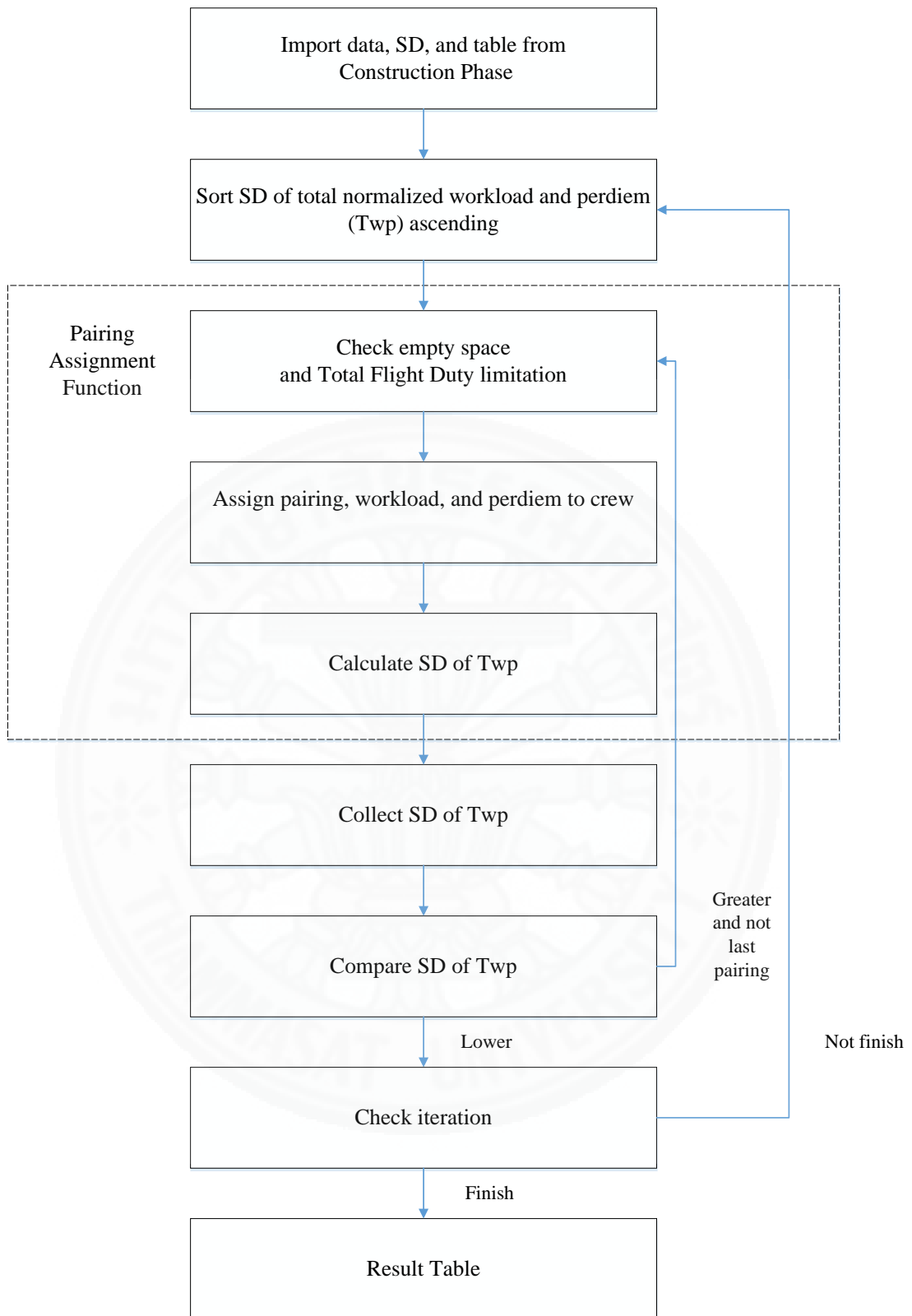


Figure 4.17 flow chart of change pairing ascending with workload and perdiem minimization simultaneously without bound

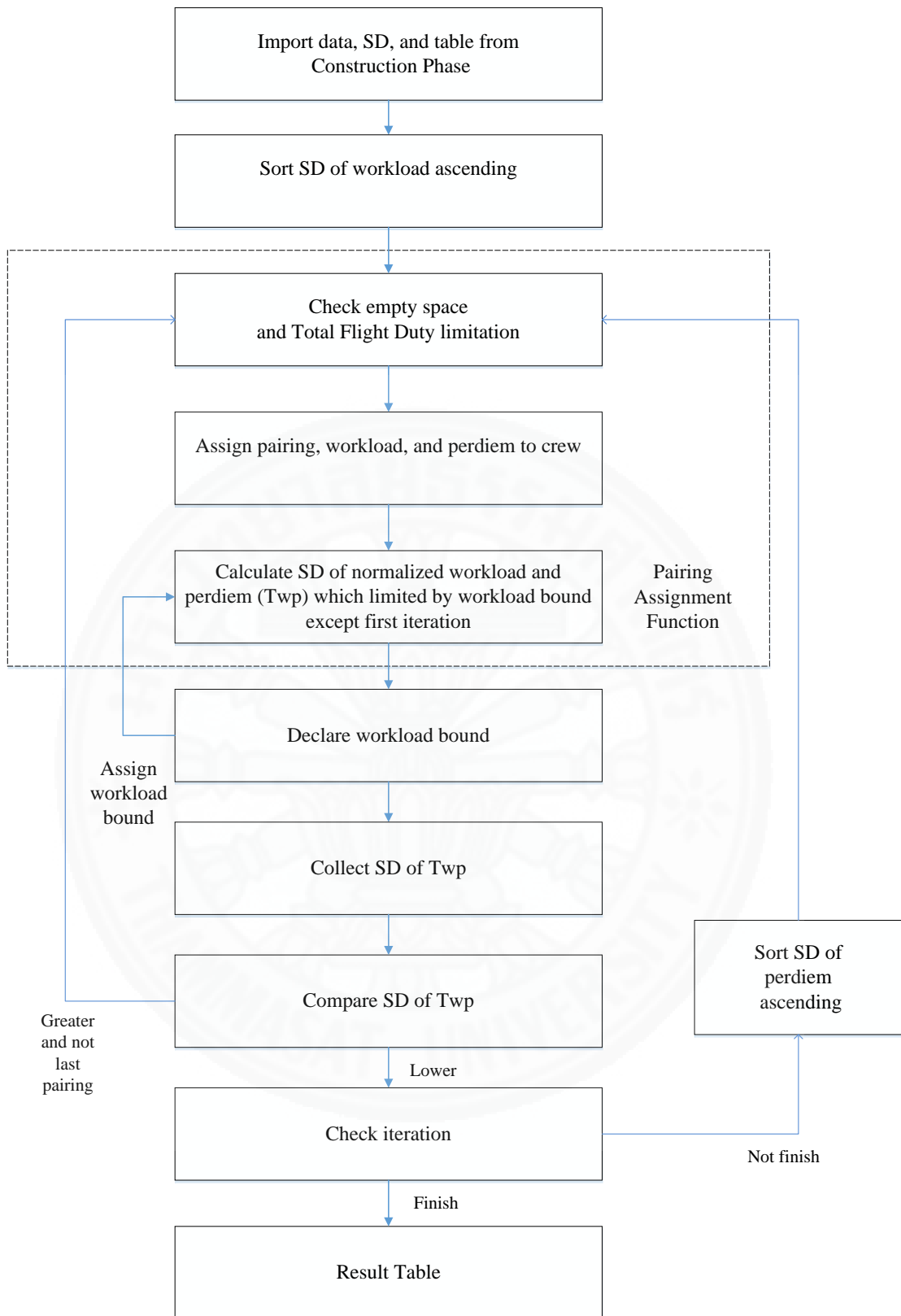


Figure 4.18 flow chart of change pairing ascending with workload and perdiem minimization simultaneously with workload bound

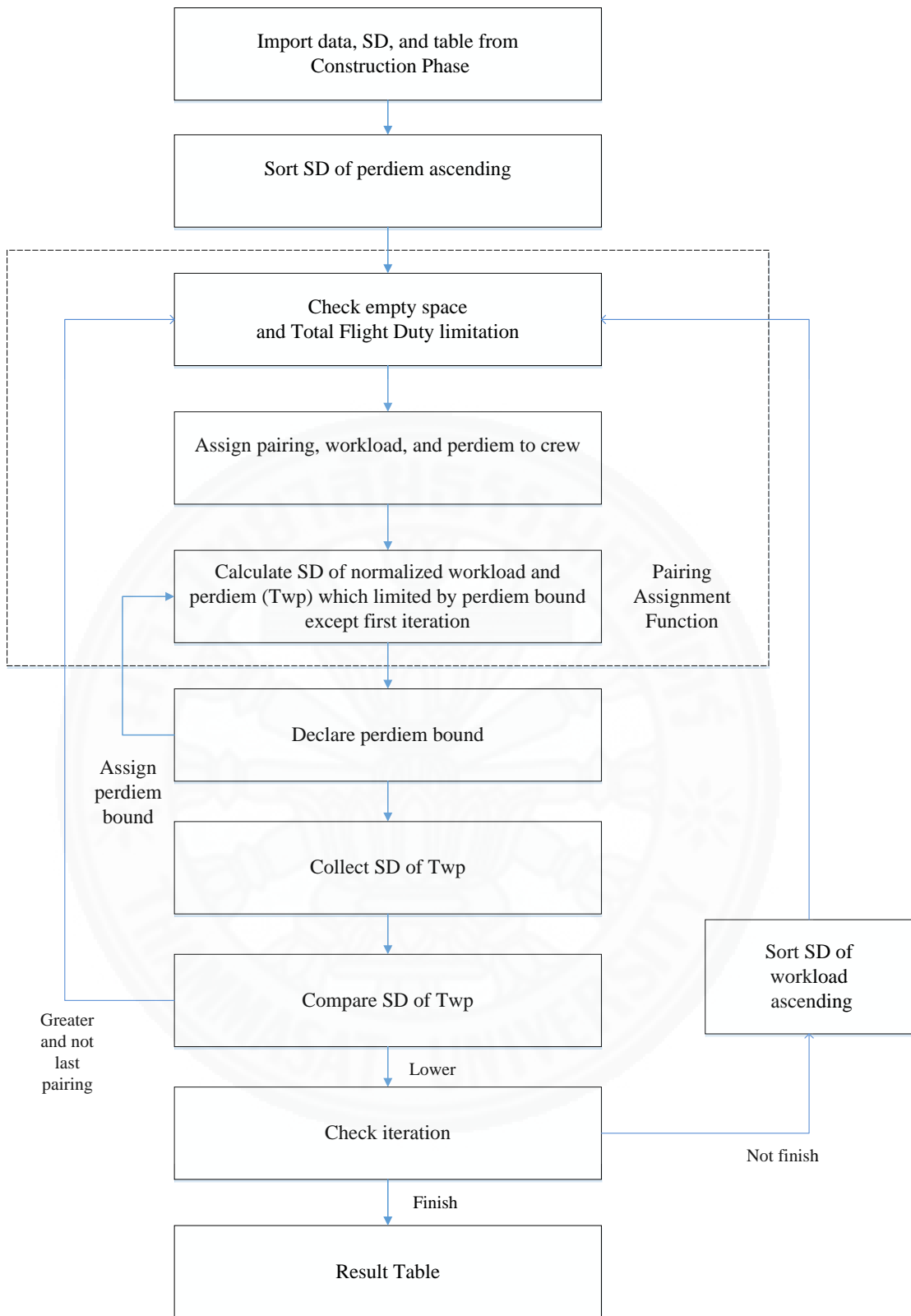


Figure 4.19 flow chart of change pairing ascending with workload and perdiem minimization simultaneously with perdiem bound

Step 1: This step needs to import data from the construction phase, such as summation of normalized workload value, normalized perdiem value, and total flight duty period from all crews. The target SD from construction phase is also included. The SD of construction phase is used as the default comparison SD with other new improvement phase creating. The construction phase crew table is also setup to be the default table for the pairing change process.

Step 2: This step will sort target SD in ascending order. From figure 4.15, the workload minimization will sort by SD of workload. Figure 4.16 shows perdiem minimization that used SD of perdiem for sorting. The no bound part will use the summation normalized SD of workload and perdiem (Twp) according to figure 4.17. From figure 4.18, the workload bound will sort by SD of workload in the first iteration and perdiem in the remaining iterations. Similarly, for perdiem bound, it will sort by SD of perdiem in first iteration and workload in the remaining iterations according to figure 4.19. Thus, the sequence of crew will be changed but the information of each crew is still the same.

Step 3: This process aims to reduce target SD. The method is to move pairing P_x to every available crew $C_1 - C_i$. In the first step, the program will check the crew C_{x+1} , where x is crew id that stores pairing P_x , to see if it matches the available size of P_x operation day or not. This process also checks total flight duty limitation. If matched, the program will remove the old P_x at crew C_x . Total flight duty, total workload, and total perdiem also are removed from C_x information. Then the program will book value of P_x and assign the value of flight duty, workload, and perdiem to C_x information. After finishing booking, the program will proceed to step 4 to calculate target SD. If not finished, the program will focus on the next crew C_{x+2} and execute step 3 again. This process will continue until the end of crew C_{x-1} .

Step 4: In this step, the program will calculate a whole crew table to find the target SD. The target SD will change every time that a pairing is moved because the total workload and perdiem value of each crew is changed. The calculated target SD is stored in an array parameter for next step comparison.

From first iteration, the highest and lowest SD of workload will declare to be the limitation of workload bound. And the highest and lowest SD of perdiem will declare to be the limitation of perdiem bound. These bound will applied in step 3 in the rest iterations.

Step 5: The objective of this step is to find the minimum target SD. This step occurs after check available function result is true. The array of target SD will be compared with initial or default SD. After finishing the comparison process, the minimum target SD will setup to be the default SD also with minimum target SD table.

4.2.2.4 High workload and perdiem distribution

This techniques aims to minimize target SD by dispersing the pairing that made the combined workload and perdiem greater to other crews in the table. The solution will continuously decrease Target SD to meet the local optimal solution. In addition, the data for selected distribution depends on the objective of SD minimization. In addition, figure 4.20 shows target SD of workload minimization is SD of workload. Figure 4.21 shows the target SD of perdiem minimization is SD of perdiem. Figure 4.22, 4.23, and 4.24 shows the SD of workload and perdiem as a target SD.

This technique sorts the target SD descending first. Then, the crews will be divided into two groups, upper class and lower class, for disperse pairing from upper class to lower class. The upper class is the sequence of crews from the crew with highest workload for workload minimization method, perdiem for perdiem minimization method, and total normalized of workload and perdiem for workload bound, perdiem bound, and without bound method to the crew before mean or average of total workload and perdiem. The lower class represents crews from average workload for workload minimization method, perdiem for perdiem minimization method, and total normalized of workload and perdiem for workload bound, perdiem bound, and without bound method to the end of crew id. This classification method can be formulated as follows:

$$m = \frac{\sum_{c=1}^i Twp_c}{N} \quad (4.1)$$

Where;

m = Mean of workload

N = total number of workload or perdiem

i = Max workload id

c = Index of crew where $c = 1, 2, \dots, i$

Twp = Total normalize of workload and perdiem of crew c

Thus,

bT = Set of upper class $\{bT_1, bT_2, \dots, bT_{m-1}\}$

sT = Set of lower class $\{sT_m, sT_{m-1}, \dots, sT_i\}$

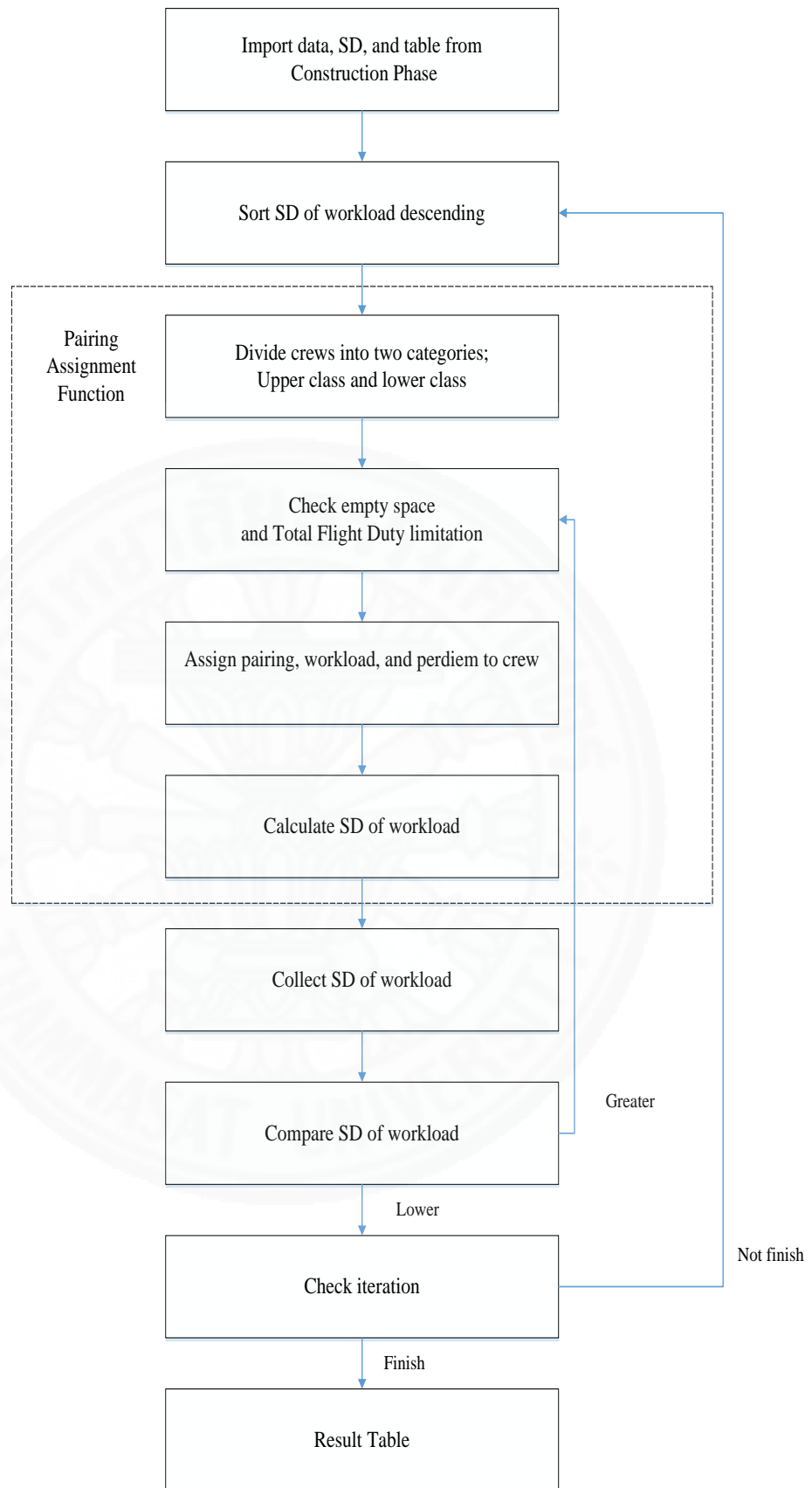


Figure 4.20 flow chart of high workload and perdiem distribution with workload minimization

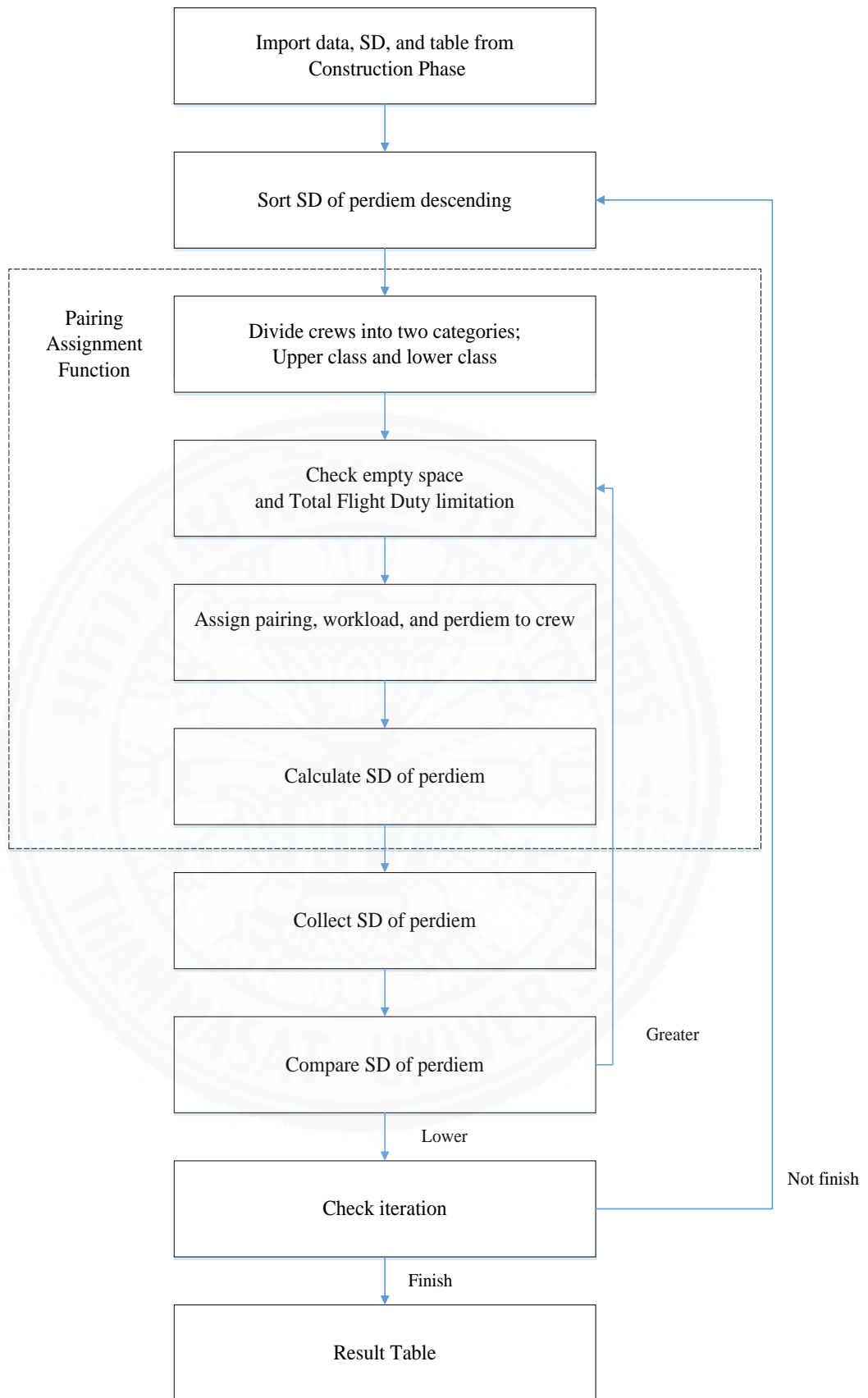


Figure 4.21 flow chart of high workload and per diem distribution with per diem minimization

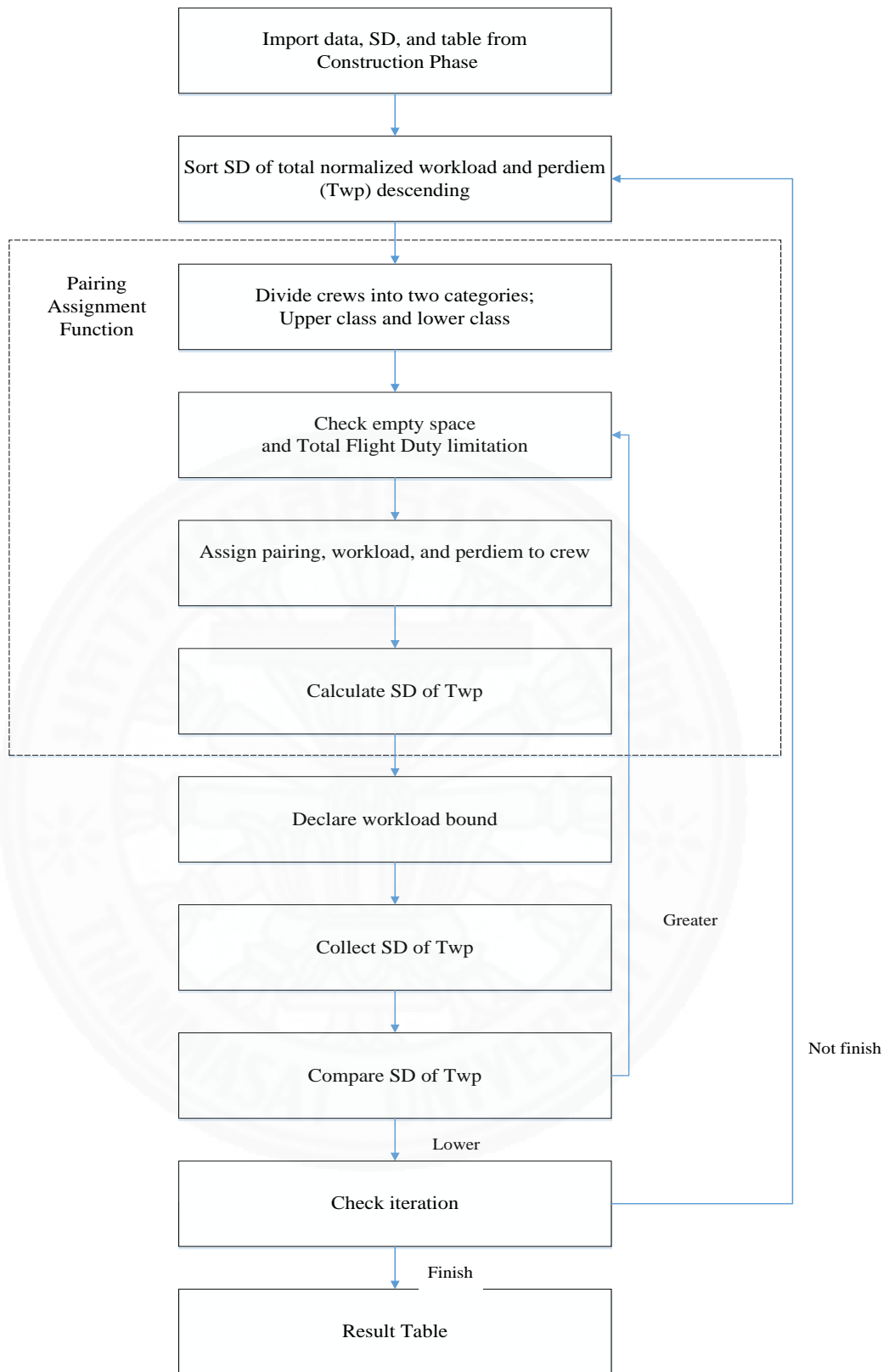


Figure 4.22 flow chart of high workload and perdiem distribution with workload and perdiem minimization simultaneously without bound

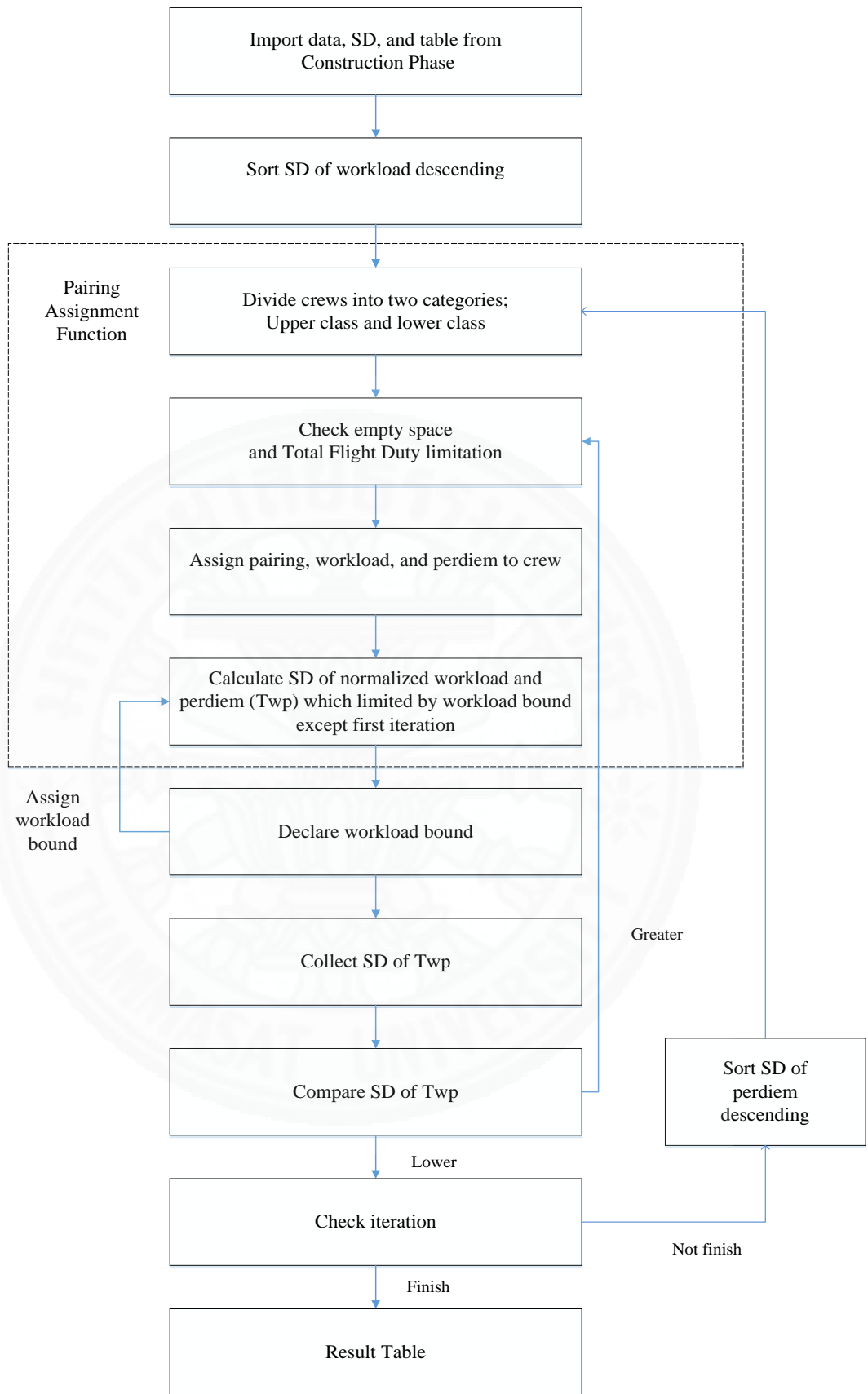


Figure 4.23 flow chart of high workload and perdiem distribution with workload and perdiem minimization simultaneously with workload bound

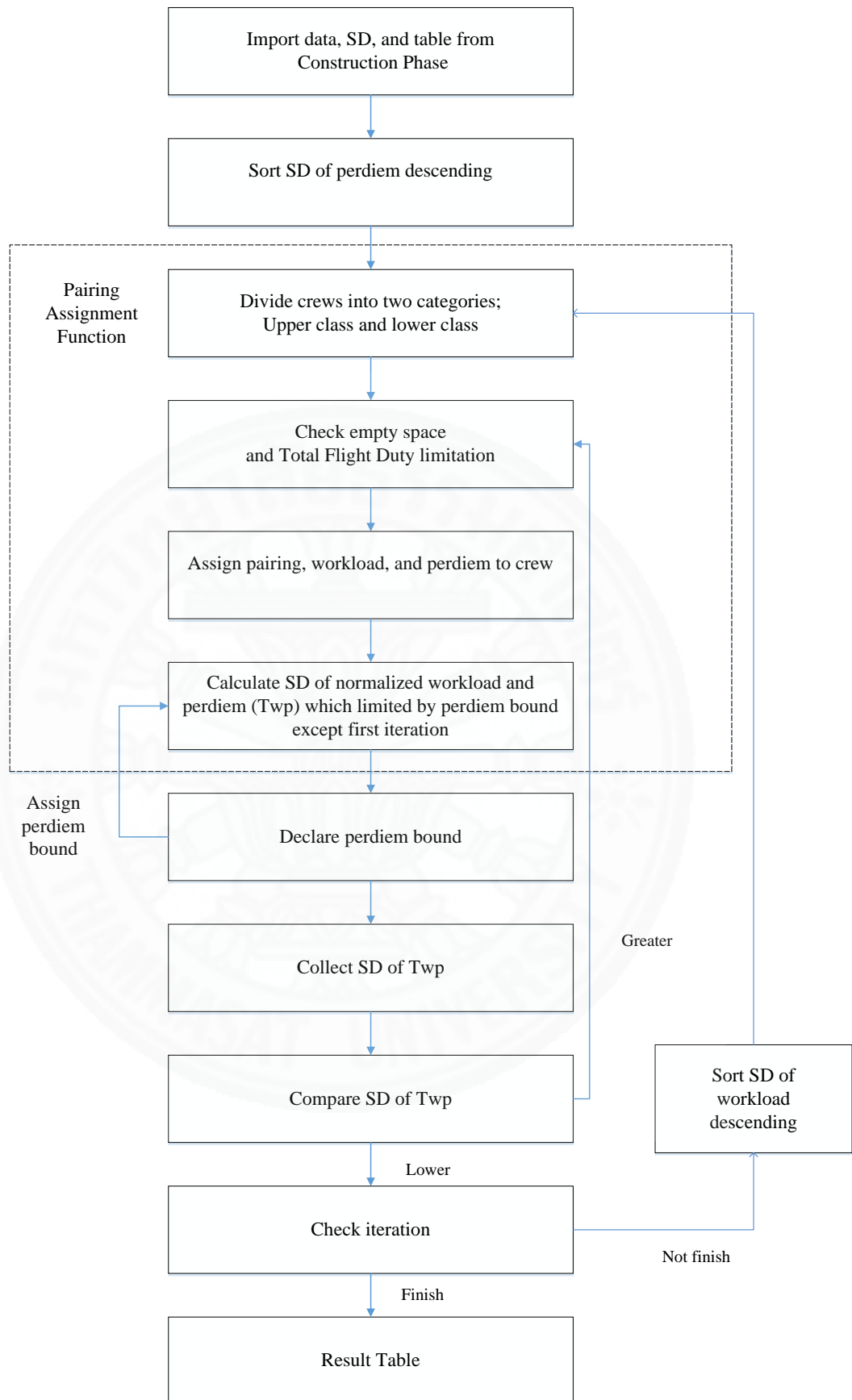


Figure 4.24 flow chart of high workload and perdiem distribution with workload and perdiem minimization simultaneously with perdiem bound

Step 1: Program will import data from the construction phase, such as crew table, target SD, total workload, total perdiem, and total flight duty of each crew.

Step 2: This step will sort target SD in descending order. From figure 4.20, the workload minimization will sort by SD of workload. Figure 4.21 shows perdiem minimization that used SD of perdiem for sorting. The no bound part will use the summation normalized SD of workload and perdiem (Twp) according to figure 4.22. From figure 4.23, the workload bound will sort by SD of workload in the first iteration and perdiem in the remaining iterations. Similarly, for perdiem bound, it will sort by SD of perdiem in first iteration and workload in the remaining iterations according to figure 4.24. Thus, the sequence of crew will be changed but the information of each crew is still the same.

Step 3: This step will divide set of crews $C = (C_1, C_2, \dots, C_i)$ into two categories: upper class $bT = (bT_1, bT_2, \dots, bT_{m-1})$ and lower class $sT = (sT_m, sT_{m-1}, \dots, sT_i)$ by using equation 4.1. This process aims to distribute crew pairing from high crews with workload for workload minimization method, perdiem for perdiem minimization method, and combined workload and perdiem for workload bound, perdiem bound, and non-bound method to other lower class crew.

Step 4: Given pairing P_x , where x is target for moving, the program selects the highest pairing that causes each crew in upper class bT_x to have high combined workload and perdiem. P_x is setup to be moving pairing. Then, the program checks total flight duty limitation and available day of sT_m with value of operation size and start day of P_x . If it is successful, the value of P_x will added to crew table and crew information. The value of total flight duty, workload, and perdiem also are added to sT_m . Pairing P_x that originally was stored at bT_x will be deleted. After the completed booking process, this program continues to execute step 4 again until the end of upper class bT_{m-1} .

If there are no available spaces, the program continue to the next sT_{m-1} until sT_{it} . After that, the program will focus on the next upper class bT_x and execute step 4 again.

Step 5: After moving a P_x to some lower class, the target SD will be calculated. This SD will be collected every time a P_x is moved. The comparison will start after P_x finished moving to all possible crews in lower class $sT = (sT_m, sT_{m-1}, \dots, sT_i)$. The collected target SDs are compared to find the minimum SD. The crew table with minimum target SD will be set to be the initial table or default table and, then, step 4 is executed again.

This step also creates the limitation bound. From first iteration, the highest and lowest SD of workload will declare to be the limitation of workload bound. And the highest and lowest SD of perdiem will declare to be the limitation of perdiem bound. These bound will applied in step 4 in the rest iterations.

Step 6: This technique will end after complete bT_{m-1} at step 4. The expected result should show greater balance for both workload and perdiem.

Chapter 5

Results and Discussion

This chapter explains the result from solving ACRP (Airline Crew Rostering Problem) problem by applying the Greedy Algorithm. This project focused on three main objectives: to minimize workload, minimize perdiem, and minimize both workload and perdiem. The solution techniques can be divided into two phases: a construction phase and an improvement phase. The expected result of the construction phase is a simple crew time table. The expected outcome of the improvement phase is to reduce the SD of target objective as much as possible. This thesis focuses on minimizing workload and perdiem simultaneously, but also experiments on minimizing only workload and perdiem. The results have five main parts which are, minimize SD of workload only, minimize SD of perdiem only, minimize SD of workload and perdiem simultaneously without any bound, minimize SD of workload and perdiem simultaneously with workload bound, and minimize SD of workload and perdiem simultaneously with perdiem bound. In addition, only the minimization of workload and perdiem did not use normalization method. The results in this chapter will be shown in the form of tables that consist of iteration, SD of workload, SD of perdiem, and percentage SD changed or reduction from construction phase.

5.1 Result and Discussion

This section discusses the combined results of SD of workload and perdiem from all techniques. The result of all improvement phases is very different and interesting. The idea is to compare the differences and find the best improvement technique. This experiment used nine instances and much iteration. Iterations indicate the number of compile cycles and can adjust to change to any value. Various iterations can make the resulting SD of workload and perdiem different. This experiment tested three different methods: minimize SD of workload alone, minimize SD of perdiem alone, and minimize SD of both workload and perdiem at the same time. All of the result will be comparing and discussed at the end of this section.

5.2 Minimize SD of workload

This section shows the result of SD of workload minimization only while ignoring perdiem. The result shows good solutions in the SD of workload. The solutions were obtained by four techniques: change pairing directly, change pairing descending, change pairing ascending, and high workload and perdiem distribution.

Table 5.1: SD of workload table using workload minimization

Instance	Iteration	SD of Construction Phase	SD of each Improvement Phase			
			Change pairing directly	Change pairing descending	Change pairing ascending	High workload distribution
TA84S	1	25.8306	9.0750	9.0750	9.1122	13.8453
	10	25.8306	9.0750	9.0750	9.1122	13.2422
	100	25.8306	9.0750	9.0750	9.1122	13.2422
TA84M	1	41.4780	9.2450	9.2450	8.0025	24.3254
	10	41.4780	9.2450	9.2450	8.0025	21.2779
	100	41.4780	9.2450	9.2450	8.0025	21.2779
TA84L	1	28.2833	20.0357	20.0357	20.0598	16.0043
	10	28.2833	20.0357	20.0357	20.0598	14.5692
	100	28.2833	20.0357	20.0357	20.0598	14.5692
TA150S	1	23.3733	8.1383	8.1383	7.7622	14.4219
	10	23.3733	7.5526	7.5526	7.7502	11.7359
	100	23.3733	7.5526	7.5526	7.7502	11.7359
TA140M	1	26.6441	16.9032	16.9032	17.1479	16.4588
	10	26.6441	15.8973	15.8973	15.8951	13.4063
	100	26.6441	15.8973	15.8973	15.8951	13.4063
TA146L	1	19.0579	14.2010	14.2010	14.2010	12.7461
	10	19.0579	14.1544	14.1544	14.1544	12.0618
	100	19.0579	14.1544	14.1544	14.1544	12.0618
TA330S	1	21.8071	13.8690	13.8690	13.8157	15.3869
	10	21.8071	13.4339	13.4339	13.5513	14.2736
	100	21.8071	13.4339	13.4339	13.5513	14.2736
TA334M	1	24.4522	15.8078	15.8078	15.8078	17.5612
	10	24.4522	14.5659	14.5659	14.5659	16.6937
	100	24.4522	14.5659	14.5659	14.5659	16.6937
TA238L	1	19.8712	17.0812	17.0812	17.0812	13.2343
	10	19.8712	17.0812	17.0812	17.0812	12.3839
	100	19.8712	17.0812	17.0812	17.0812	12.3839

From table 5.1, all of the improvement phase can produce satisfactory results. The result of improvement phases can reduce high value of SD of workload from the construction phase. Most of the improvement techniques produced a different optimal SD of workload in every instances test. The change pairing directly technique resulted in the best solutions in test instances TA84S, TA150S, TA330S, and TA334M, with minimal SD of workloads of 9.0750, 7.5526, 13.4339, and 14.5659, respectively. Interestingly, the most successful minimized instances are short distance routes. The reason may be the delicate changing pairing method that moves every possible crew to produces minimum SD of workload.

Another best technique is high workload distribution. The results were satisfactory on test instances TA84L, TA140M, TA146L, and TA238L which had minimum SDs of 14.5692, 13.4063, 12.0618, and 12.3839, respectively. The interesting finding is that the most successful minimized instances are long distance routes. The reason is that this technique is designed to reduce SD of workload by dividing crews into two classes. The class of high total workload will disperse high

workload pairing to crews in another class. Consequently, total workloads of all crews were satisfactorily balanced.

The remaining techniques, change pairing descending and change pairing ascending, produced a few satisfactory solutions. The change pairing descending technique produced at test instance TA334M the same solution as the change pairing directly technique. Another, change pairing ascending can produce two satisfactory solutions: TA84M, and TA334M. The interesting thing is this method mostly produced optimal solution in various distance route instances. Moreover, the most results of the three change pairing methods; directly, descending, and ascending are similar. From this we can conclude that the unsort and sort methods before improvement have little effect on the result

Table 5.2: SD of perdiem table using workload minimization

Instance	Iteration	SD of Construction Phase	SD of each Improvement Phase			
			Change pairing directly	Change pairing descending	Change pairing ascending	High workload distribution
TA84S	1	4952.6465	4932.8921	4932.8398	4940.8657	5249.7915
	10	4952.6465	4932.8921	4932.8398	4940.8657	5303.6177
	100	4952.6465	4932.8921	4932.8398	4940.8657	5303.6177
TA84M	1	11208.2412	5418.5239	5754.2432	5355.6890	8761.6152
	10	11208.2412	5418.5239	5754.2432	5355.6890	8289.5449
	100	11208.2412	5418.5239	5754.2432	5355.6890	8289.5449
TA84L	1	6143.6514	4533.1421	4745.6099	4516.3062	3629.8899
	10	6143.6514	4533.1421	4745.6094	4516.3062	3382.1333
	100	6143.6514	4533.1421	4745.6094	4516.3062	3382.1333
TA150S	1	4638.0747	5324.5317	5324.5308	5395.1660	4529.4263
	10	4638.0747	5158.8940	5158.8936	5371.4683	4446.7505
	100	4638.0747	5158.8940	5158.8936	5371.4683	4446.7505
TA140M	1	7169.1719	4964.2559	4937.7622	4986.1797	4980.9150
	10	7169.1719	4971.8882	4948.4253	4971.8882	4517.5200
	100	7169.1719	4971.8882	4948.4253	4971.8882	4517.5200
TA146L	1	4999.6504	4206.8076	4206.8071	4206.8081	3907.7407
	10	4999.6504	4058.6511	4058.6506	4058.6511	3790.8203
	100	4999.6504	4058.6511	4058.6506	4058.6511	3790.8203
TA330S	1	6022.7490	5955.3721	5963.4233	5958.6055	5661.6704
	10	6022.7490	5890.8311	5946.7412	5941.9111	5622.9077
	100	6022.7490	5890.8311	5946.7412	5941.9111	5622.9077
TA334M	1	7148.8438	6474.2012	6475.4458	6470.8467	6269.1968
	10	7148.8438	6329.3433	6328.3052	6324.6396	6239.8481
	100	7148.8438	6329.3433	6328.3052	6324.6396	6239.8481
TA238L	1	6331.2549	5972.3550	5972.3555	5972.3535	5385.1401
	10	6331.2549	5972.3535	5972.3555	5972.3535	5291.8179
	100	6331.2549	5972.3545	5972.3555	5972.3535	5291.8179

From table 5.2, most of improvement techniques result in a satisfactory SD of perdiem. There are some techniques that produced higher SDs than the SDs of the construction phase such as change pairing directly, change pairing descending, and change pairing ascending in instance TA150S. The change pairing ascending in this instance produced a less satisfactory result, at 5371.4683, than other techniques. However, the high workload distribution still produced an acceptable SD of perdiem, in this case about 4446.7505. We can conclude that there is no clear relation between workload and perdiem in this instance.

The high workload distribution can produce most of the satisfied solutions, such as TA84L, TA150S, TA140M, TA146L, TA330S, TA334M, and TA238L, whose SDs of perdiem are 3382.1333, 4446.7505, 4517.5200, 3790.8203, 5622.9077, 6239.8481, and 5291.8179, respectively. So, this obviously shows that the workload distribution can reduce SD of perdiem and also SD of long distance route workload.

Another change pairing method can produce two optimal solutions, such as TA84S for change pairing descending technique, and TA84M for change pairing ascending technique.

5.3 Minimize SD of perdiem

This section shows the result of SD of perdiem minimization only, by ignoring workload. The result shows good solutions in SD of perdiem.

Table 5.3 SD of workload table using perdiem minimization

Instance	Iteration	SD of Construction Phase	SD of each Improvement Phase			
			Change pairing directly	Change pairing descending	Change pairing ascending	High perdiem distribution
TA84S	1	25.8306	40.9871	40.7937	40.7949	20.7698
	10	25.8306	40.9871	40.7937	41.8933	20.1775
	100	25.8306	40.9871	40.7937	41.8933	20.1775
TA84M	1	41.4780	34.7609	33.8553	35.8218	25.3657
	10	41.4780	30.1491	30.2490	35.1450	21.2231
	100	41.4780	30.1491	30.2490	35.1450	21.2231
TA84L	1	28.2833	20.7667	20.0373	20.8992	16.2396
	10	28.2833	20.7667	20.0373	20.8992	15.6292
	100	28.2833	20.7667	20.0373	20.8992	15.6292
TA150S	1	23.3733	37.4462	36.2416	36.4622	19.3914
	10	23.3733	35.6293	34.0828	34.6036	17.8067
	100	23.3733	35.6293	34.0828	34.6036	17.8067
TA140M	1	26.6441	32.4200	32.3181	31.4286	16.9465
	10	26.6441	26.4240	24.2467	23.2702	14.5087
	100	26.6441	26.4240	24.2467	23.2702	14.5087
TA146L	1	19.0579	21.4943	21.8900	21.9899	14.0848
	10	19.0579	19.7913	21.0767	21.0767	13.3307
	100	19.0579	19.7913	21.0767	21.0767	13.3307
TA330S	1	21.8071	29.3076	29.5491	29.0551	17.6498
	10	21.8071	26.3895	27.0843	25.9532	17.0778
	100	21.8071	26.3895	27.0843	25.9532	17.0778

TA334M	1	24.4522	32.5122	32.7639	32.5239	20.2152
	10	24.4522	30.5178	30.5211	30.2585	19.4801
	100	24.4522	30.5178	30.5211	30.2585	19.4801
TA238L	1	19.8712	19.5419	20.2365	19.5461	15.3938
	10	19.8712	19.5412	20.1902	19.5453	14.9090
	100	19.8712	19.5412	20.1902	19.5453	14.9090

Table 5.3 shows the incredible result from high perdiem distribution technique. All of results from this technique are better solutions than other techniques. The reason is it can disperse many with high total workload from one crew to other crews. The remaining techniques produce very high SD of workload especially change pairing descending and ascending techniques that produce four high SD each.

Table 5.4 SD of perdiem table using perdiem minimization

Instance	Iteration	SD of Construction Phase	SD of each Improvement Phase			
			Change pairing directly	Change pairing descending	Change pairing ascending	High perdiem distribution
TA84S	1	4952.6460	1557.1240	1477.4646	1565.8047	4443.5234
	10	4952.6460	1557.1240	1477.4646	1481.8721	4376.6812
	100	4952.6460	1557.1240	1477.4646	1481.8721	4376.6812
TA84M	1	11208.2402	2926.1685	2944.9614	2791.6023	8809.3418
	10	11208.2402	2741.8958	2741.9299	2516.7654	8228.6641
	100	11208.2402	2741.8958	2741.9299	2516.7654	8228.6641
TA84L	1	6143.6519	4644.1855	4561.5308	4647.9175	3525.0706
	10	6143.6519	4644.1855	4561.5303	4647.9175	3422.9509
	100	6143.6519	4644.1855	4561.5303	4647.9175	3422.9509
TA150S	1	4683.0742	1547.6014	1560.1687	1510.0592	3673.5864
	10	4683.0742	1474.6520	1474.8788	1342.0878	3538.0178
	100	4683.0742	1474.6520	1474.8788	1342.0878	3538.0178
TA140M	1	7169.1729	3148.3926	3277.9463	3192.0593	5021.6162
	10	7169.1729	2848.7959	2965.1982	2848.2712	4615.4595
	100	7169.1729	2848.7959	2965.1982	2848.2712	4615.4595
TA146L	1	4999.6499	3113.5918	3168.1221	3168.1221	3469.8267
	10	4999.6499	2981.0764	3017.8857	3017.8860	3235.1953
	100	4999.6499	2981.0764	3017.8857	3017.8860	3235.1953
TA330S	1	6022.7480	3224.1492	3391.2781	3360.9858	5176.2012
	10	6022.7480	3075.2869	3249.9578	3207.0588	5071.7715
	100	6022.7480	3075.2869	3249.9578	3207.0588	5071.7715
TA334M	1	7148.8447	4038.0454	3950.5808	3958.8188	5641.5400
	10	7148.8447	3881.4170	3811.6323	3811.0105	5514.3535
	100	7148.8447	3881.4170	3811.6323	3811.0105	5514.3535
TA238L	1	6331.2559	4672.3589	4653.7144	4672.3574	5037.9883
	10	6331.2559	4672.3589	4653.7144	4672.3574	4966.5522
	100	6331.2559	4672.3589	4653.7144	4672.3574	4966.5522

In contrast with previous result, table 5.4 shows that most high perdiem results came from the high perdiem distribution technique. There is only one satisfactory result, TA84L, which is a small test instance and long distance route. The best results in this experiment are from the change pairing ascending technique with 2516.7654, 1342.0878, 2848.2712, and 3811.0105 from instances TA84M, TA150S, TA140M, and TA334M respectively. Most of the good solutions are for various range flights. The other two techniques also produce good solutions. The change pairing directly techniques produced the best solutions in instances TA146L and TA330S. The change pairing descending techniques produced the best solutions in instances TA84S and TA238L.

5.4 Minimize SD of workload and perdiem simultaneously without bound

This section aims to minimize SD of both workload and perdiem simultaneously. Due to the highly different values of SD between workload and perdiem, normalization was applied to solve this problem. This solution did not apply any workload or perdiem bound.

Table 5.5 SD of workload table using workload and perdiem minimization without bound

Instance	Iteration	SD of Construction Phase	SD of each Improvement Phase			
			Change pairing directly	Change pairing descending	Change pairing ascending	High WL and PD distribution
TA84S	1	25.8306	12.8448	10.4426	13.0062	26.1736
	10	25.8306	12.2737	10.1846	11.3851	17.7565
	100	25.8306	12.2737	10.1846	11.3852	17.7565
TA84M	1	41.4780	15.9578	15.4209	15.9578	21.7941
	10	41.4780	11.1579	12.3778	11.1570	21.4041
	100	41.4780	11.1570	12.3778	11.1570	21.4041
TA84L	1	28.2833	20.1555	20.9574	20.1555	26.1290
	10	28.2833	20.1555	20.9574	20.1555	26.1290
	100	28.2833	20.1555	20.9574	20.1555	26.1290
TA150S	1	23.3733	10.2592	9.1113	9.9664	14.8895
	10	23.3733	9.4683	9.2989	9.1962	14.8895
	100	23.3733	9.4683	9.2989	9.1962	14.8895
TA140M	1	26.6441	15.8752	15.4932	15.8698	25.2845
	10	26.6441	15.1985	15.4537	15.4448	25.2282
	100	26.6441	15.1985	15.4537	15.4448	25.2282
TA146L	1	19.0579	15.4303	15.4303	15.4303	18.3158
	10	19.0579	15.4303	15.4303	15.4303	18.3158
	100	19.0579	15.4303	15.4303	15.4303	18.3158
TA330S	1	21.8071	14.3296	14.5260	14.5260	20.3814
	10	21.8071	13.2523	13.4412	13.4412	20.4866
	100	21.8071	13.2523	13.4412	13.4412	20.4866
TA334M	1	24.4522	16.5527	16.5527	16.5527	23.5936
	10	24.4522	15.1432	15.1432	15.1432	22.9227

	100	24.4522	15.1432	15.1432	15.1432	22.9227
TA238L	1	19.8712	18.0734	18.0734	18.0734	20.0006
	10	19.8712	18.0734	18.0734	18.0734	20.0006
	100	19.8712	18.0734	18.0734	18.0734	20.0006

From Table 5.5, this experiment gives a satisfactory solution with reduction of all SD of workload. However, the result did not complete with a good optimal solution. The best technique to minimize SD of workload is the change pairing directly technique which gave best results in instances TA84M, TA84L, TA140M, TA146L, TA330S, TA334M, and TA238L. The result can shows that this technique is proper for the various mix route and long distance route. Another interesting technique is change pairing ascending technique which gave the best result in all long distance routes and most various distance routes, such as TA84M, TA84L, TA150S, TA146L, TA334M, and TA238L. The change pairing descending technique yielded satisfactory results in four instances: TA84S, TA146L, TA334M, and TA238L. The high workload and perdiem distribution technique cannot produce any best result at all and gives the worst result in every test instance.

Table 5.6 SD of perdiem table using workload and perdiem minimization without bound

Instance	Iteration	SD of Construction Phase	SD of each Improvement Phase			
			Change pairing directly	Change pairing descending	Change pairing ascending	High WL and PD distribution
TA84S	1	4952.6460	2712.8418	2493.4668	2673.1040	3014.9819
	10	4952.6460	2400.2322	2423.3892	2336.9780	2470.4265
	100	4952.6460	2400.2322	2423.3892	2336.9778	2470.4265
TA84M	1	11208.2402	3678.1956	3865.2637	3678.1950	4571.7173
	10	11208.2402	3959.2625	3967.3340	3959.2622	4207.0645
	100	11208.2402	3959.2625	3967.3340	3959.2622	4207.0645
TA84L	1	6143.6519	4557.7705	4873.2905	4557.7710	5463.3770
	10	6143.6519	4557.7705	4873.2905	4557.7700	5463.3765
	100	6143.6519	4557.7705	4873.2905	4557.7700	5463.3765
TA150S	1	4683.0742	3165.5696	2884.9604	3100.9329	5480.1914
	10	4683.0742	2537.6790	2439.6409	2861.1836	5480.1914
	100	4683.0742	2537.6790	2439.6409	2861.1836	5480.1914
TA140M	1	7169.1729	4519.6846	4660.0034	4591.3013	5085.0986
	10	7169.1729	4498.1973	4625.7358	4578.6187	5085.0986
	100	7169.1729	4498.1973	4625.7358	4578.6187	5085.0986
TA146L	1	4999.6499	3133.8350	3133.8350	3133.8352	4725.9521
	10	4999.6499	3133.8350	3133.8350	3133.8352	4725.9521
	100	4999.6499	3133.8350	3133.8350	3133.8352	4725.9521
TA330S	1	6022.7480	4289.7637	4367.2773	4367.2773	5297.5420
	10	6022.7480	4127.5513	4156.7324	4156.7319	5210.0557
	100	6022.7480	4127.5513	4156.7324	4156.7319	5210.0557
TA334M	1	7148.8447	5199.1387	5199.1396	5199.1387	6045.2754
	10	7148.8447	5090.5605	5090.5601	5090.5615	6079.0552

	100	7148.8447	5090.5605	5090.5601	5090.5615	6079.0552
TA238L	1	6331.2559	5653.2837	5653.2827	5653.2842	5846.9331
	10	6331.2559	5653.2837	5653.2827	5653.2842	5846.9331
	100	6331.2559	5653.2837	5653.2827	5653.2842	5846.9331

From table 5.6, most results show highly significant reductions. The lower value of SD of perdiem means a reduction of unbalanced salary. The change pairing directly technique, change pairing descending technique, and change pairing ascending technique produce similar numbers of best results, which are 3, 4, and 3, respectively. The change directly pairing technique reduces SD of workload in instances TA140M, TA146L, and TA330S. The change pairing descending technique can significantly reduce SD in instances TA150S, TA146L, TA334M, and TA238L. Another interesting technique is change pairing ascending which has the best results in instances TA84S, TA84M, and TA84L. Interestingly, this technique gives satisfactory solutions in all short distance route instances. In contrast, the high workload and perdiem distribution produces the worst solution for all instances that is the same with SD of workload in table 5.5.

Table 5.7 shows total normalized SD of workload and perdiem and percentage of total workload and perdiem SD reduction from construction phase. The high value of percentage indicates the efficiency of the improvement algorithm. From this table, the change pairing directly technique can produce SD of workload and perdiem satisfaction in most instances, except TA84S, and TA150S. It also gives the best average percentage reduction which is 38.6216% with average total normalized SD of workload and perdiem about 0.4442. On the other hand, the high workload and perdiem distribution produces the worst total normalized SD of workload and perdiem in all case instances. It produces considerably less percentage reduction of about 18.6357%. The other two techniques also produce five best reductions with non-worst reduction but in different instances. The change pairing descending produced best total normalized SD of workload and perdiem reduction in instances TA84S, TA150S, TA146L, TA334M, and TA238L. The change pairing ascending produced best total normalized SD of workload and perdiem reduction in instances TA84M, TA84L, TA146L, TA334M, and TA238L.

Table 5.7 Total normalized workload and perdiem SD (Twp) without bound table

Instance	SD of Construction phase	SD and percent changed of each Improvement Phase							
		SD Tech 1	SD Changed Tech 1 (%)	SD Tech 2	SD Changed Tech 2 (%)	SD Tech 3	SD Changed Tech 3 (%)	SD Tech 4	SD Changed Tech 4 (%)
TA84S	1.3080	0.6285	51.9454	0.5865	55.1571	0.5997	54.1526	0.7585	42.0091
TA84M	1.6257	0.5226	67.8555	0.5414	66.6999	0.5226	67.8555	0.6965	57.1551
TA84L	0.6710	0.4883	27.2245	0.5153	23.1982	0.4883	27.2245	0.6078	9.4080
TA150S	0.7183	0.3353	53.3130	0.3257	54.6558	0.3531	50.8323	0.6305	12.2197
TA140M	0.7169	0.4297	40.0579	0.4396	38.6823	0.4371	39.0331	0.5921	17.4033
TA146L	0.4713	0.3353	28.8630	0.3353	28.8630	0.3353	28.8630	0.4490	4.7450
TA330S	0.5998	0.3882	35.2807	0.3922	34.6086	0.3922	34.6086	0.5408	9.8394
TA334M	0.6059	0.4055	33.0763	0.4055	33.0763	0.4055	33.0763	0.5396	10.9454
TA238L	0.5162	0.4647	9.9784	0.4647	9.9784	0.4647	9.9784	0.4956	3.9960
AVG	0.8037	0.4442	38.6216	0.4451	38.3244	0.4443	38.4027	0.5900	18.6357

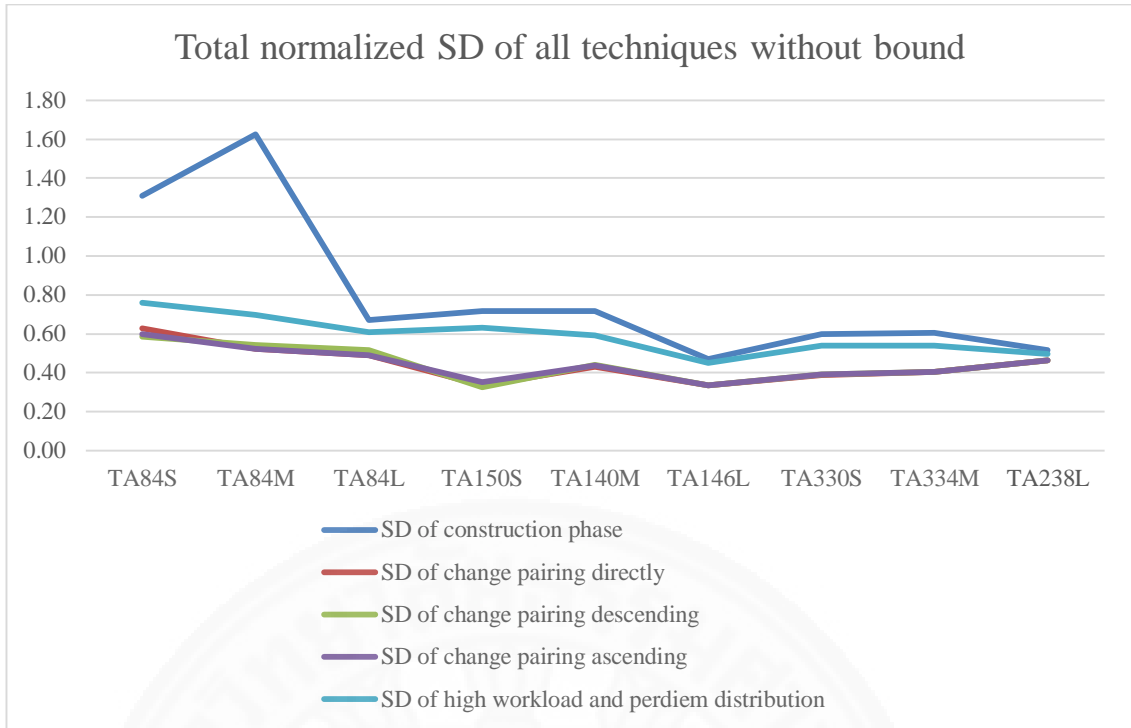


Figure 5.1 Total normalized SD of all techniques without bound graph

Figure 5.1 shows SD reduction from the construction phase (dark blue line). The three change pairing directly, descending, and ascending produced similar results. The average total normalized SD of workload and perdiem of these methods are very close; about 0.4442 for directly, 0.4451 for descending, and 0.4443 for ascending. The high workload and perdiem distribution produced higher total normalized SD of workload and perdiem than others methods (light blue line).

5.5 Minimize SD of workload and perdiem simultaneously with workload bound

This section shows the result of SD of workload and perdiem minimization with workload bound. The workload limitation aims to reduce the variation of total workload and perdiem value. This method needs normalization to minimize SD of both workload and perdiem simultaneously. Thus, the table of total normalization SD of workload and perdiem also shows in this section.

Table 5.8 SD of workload table using workload and perdiem minimization with workload bound

Instance	Iteration	SD of Construction Phase	SD of each Improvement Phase			
			Change pairing directly	Change pairing descending	Change pairing ascending	High WL and PD distribution
TA84S	1	25.8306	12.8448	10.4426	13.0062	13.4145
	10	25.8306	11.3168	10.1846	12.7126	12.8267
	100	25.8306	11.3168	10.1846	12.7126	12.8267
TA84M	1	41.4780	15.9578	15.9578	14.3837	21.7941
	10	41.4780	13.0765	11.5588	11.4641	18.6937
	100	41.4780	13.0765	11.5588	11.4641	18.6937
TA84L	1	28.2833	20.1555	20.9574	20.1555	26.1290
	10	28.2833	20.1555	20.9574	20.1555	17.5597
	100	28.2833	20.1555	20.9574	20.1555	17.5597
TA150S	1	23.3733	10.2592	10.3194	9.9664	15.0243
	10	23.3733	8.8061	9.0043	10.0552	14.4971
	100	23.3733	8.8061	9.0043	10.0552	14.4971
TA140M	1	26.6441	15.9752	15.4932	15.8698	25.3626
	10	26.6441	15.8752	15.2929	15.2889	19.4284
	100	26.6441	15.8752	15.2929	15.2889	19.4282
TA146L	1	19.0579	15.4303	15.4303	15.4303	18.3116
	10	19.0579	15.4303	15.4303	15.4303	18.3415
	100	19.0579	15.4303	15.4303	15.4303	18.3415
TA330S	1	21.8071	14.3296	14.5260	14.5260	20.2821
	10	21.8071	14.0231	14.4956	13.8558	20.4675
	100	21.8071	14.0231	14.4956	13.8558	20.4675
TA334M	1	24.4522	16.5527	16.5527	16.5527	23.0720
	10	24.4522	16.4198	15.8150	16.2938	23.3990
	100	24.4522	16.4198	15.8150	16.2938	23.3990
TA238L	1	19.8712	18.0734	18.0734	18.0734	20.4340
	10	19.8712	18.0734	18.0734	18.0734	14.6170
	100	19.8712	18.0734	18.0734	18.0734	14.6170

From table 5.8, there are two techniques which produce four best solutions and the other two techniques produce two best solutions with different number of worst solutions. The change pairing descending technique produces four best solutions, such as TA84S, TA140M, TA146L, and TA334M, with two worst solutions; TA84L and TA238L. The change pairing ascending technique also produces four best solutions which are TA84M, TA140M, TA146L, and TA330S. Most of the best solutions come from various route instances. The change pairing directly produces only two best solutions, TA150S and TA146L. This is similar to high workload and perdiem distribution techniques which produce the best solutions in TA84L, and TA238L but the rest of its instances are the worst solutions. The most interesting thing is all of its best solutions come from long distance route instances. The reason is that this technique disperses high pairings that cause high workload and perdiem away to less total workload and perdiem crew.

Table 5.9 SD of perdiem table using workload and perdiem minimization with workload bound

Instance	Iteration	SD of Construction Phase	SD of each Improvement Phase			
			Change pairing directly	Change pairing descending	Change pairing ascending	High WL and PD distribution
TA84S	1	4952.6460	2712.8418	2493.4666	2673.1040	5186.4365
	10	4952.6460	2717.9968	2423.3889	2463.5803	4761.7637
	100	4952.6460	2717.9968	2423.3889	2463.5803	4761.7637
TA84M	1	11208.2402	3678.1956	3678.1956	3872.7554	4571.7173
	10	11208.2402	3872.2119	4040.2937	4055.3579	4090.7754
	100	11208.2402	3872.2119	4040.2937	4055.3579	4090.7754
TA84L	1	6143.6519	4557.7705	4873.2905	4557.7710	5463.3770
	10	6143.6519	4557.7705	4873.2905	4557.7705	4616.1416
	100	6143.6519	4557.7705	4873.2905	4557.7705	4616.1416
TA150S	1	4683.0742	3165.5696	3066.2192	3100.9326	5256.8228
	10	4683.0742	3300.2295	3304.9180	2383.4795	3931.7058
	100	4683.0742	3300.2295	3304.9180	2383.4795	3931.7058
TA140M	1	7169.1729	4519.6846	4660.0044	4591.3022	5104.9624
	10	7169.1729	4519.6846	4683.1929	4652.2412	4147.2964
	100	7169.1729	4519.6846	4683.1929	4652.2412	4147.2964
TA146L	1	4999.6499	3133.8350	3133.8350	3133.8350	4726.2998
	10	4999.6499	3133.8350	3133.8350	3133.8350	4589.9175
	100	4999.6499	3133.8350	3133.8350	3133.8350	4589.9175
TA330S	1	6022.7480	4289.7637	4367.2788	4367.2769	5366.4932
	10	6022.7480	4218.7290	4372.8999	4278.3794	4808.0405
	100	6022.7480	4218.7290	4372.8999	4278.3794	4808.0405
TA334M	1	7148.8447	5199.1387	5199.1396	5199.1387	6094.9028
	10	7148.8447	5184.9219	5247.7534	5219.4507	5346.5132
	100	7148.8447	5184.9219	5247.7534	5219.4507	5346.5132
TA238L	1	6331.2559	5653.2837	5653.2837	5653.2842	5943.8994
	10	6331.2559	5653.2837	5653.2837	5653.2842	5863.3257
	100	6331.2559	5653.2837	5653.2837	5653.2842	5863.3257

From table 5.9, all techniques can produce satisfactory solutions especially the change pairing directly technique that can produce many best results, such as TA84M, TA84L, TA146L, TA330S, TA334M, and TA238L. This technique mostly produced good solutions in high number of pairing cases and long distance route instances. The change pairing descending produced the best solutions in instances TA84S, TA146L, and TA338L. The change pairing ascending also produced the three best solutions in TA84L, TA150S TA146L. Finally, the high workload and perdiem distribution technique can produce only one best solution, TA140M, while producing many worst solutions.

Table 5.10 Total normalized workload and perdiem SD (Twp) with workload bound table

Instance	SD of Construction phase	SD and percent changed of each Improvement Phase							
		SD Tech 1	SD Changed Tech 1 (%)	SD Tech 2	SD Changed Tech 2 (%)	SD Tech 3	SD Changed Tech 3 (%)	SD Tech 4	SD Changed Tech 4 (%)
TA84S	1.3080	0.6555	49.8820	0.5865	55.1572	0.6476	50.4857	0.9960	23.8504
TA84M	1.6257	0.5431	66.5925	0.5358	67.0398	0.5358	67.0423	0.6460	60.2672
TA84L	0.6710	0.4883	27.2245	0.5153	23.1982	0.4883	27.2245	0.4619	31.1582
TA150S	0.7183	0.3770	47.5144	0.3806	47.0041	0.3345	53.4225	0.5166	28.0709
TA140M	0.7169	0.4397	38.6579	0.4404	38.5709	0.4387	38.7980	0.4678	34.7511
TA146L	0.4713	0.3353	28.8630	0.3353	28.8630	0.3353	28.8630	0.4424	6.1450
TA330S	0.5998	0.4033	32.7740	0.4174	30.4074	0.4040	32.6471	0.5202	13.2781
TA334M	0.6059	0.4244	29.9556	0.4204	30.6242	0.4245	29.9335	0.5116	15.5627
TA238L	0.5162	0.4647	9.9784	0.4647	9.9784	0.4647	9.9784	0.4347	15.7811
AVG	0.8037	0.4590	36.8269	0.4552	36.7604	0.4526	37.5994	0.5552	25.4294

Table 5.10 presents total normalized SD of workload and perdiem, and percentage reduction from SD of construction phase. The best average solution technique is the change pairing ascending technique which produced average total normalized workload and perdiem SD about 0.4526, a reduction of SD from the construction phase of about 37.5994%. The values for change pairing directly and change pairing descending are close to the change pairing ascending technique at 0.4590, and 0.4552, respectively. The average worst solution comes from the high workload and perdiem technique which produced the worst solutions for most instances, except TA84L, and TA238L. The reason is it can reduce a lot of high working day pairing.

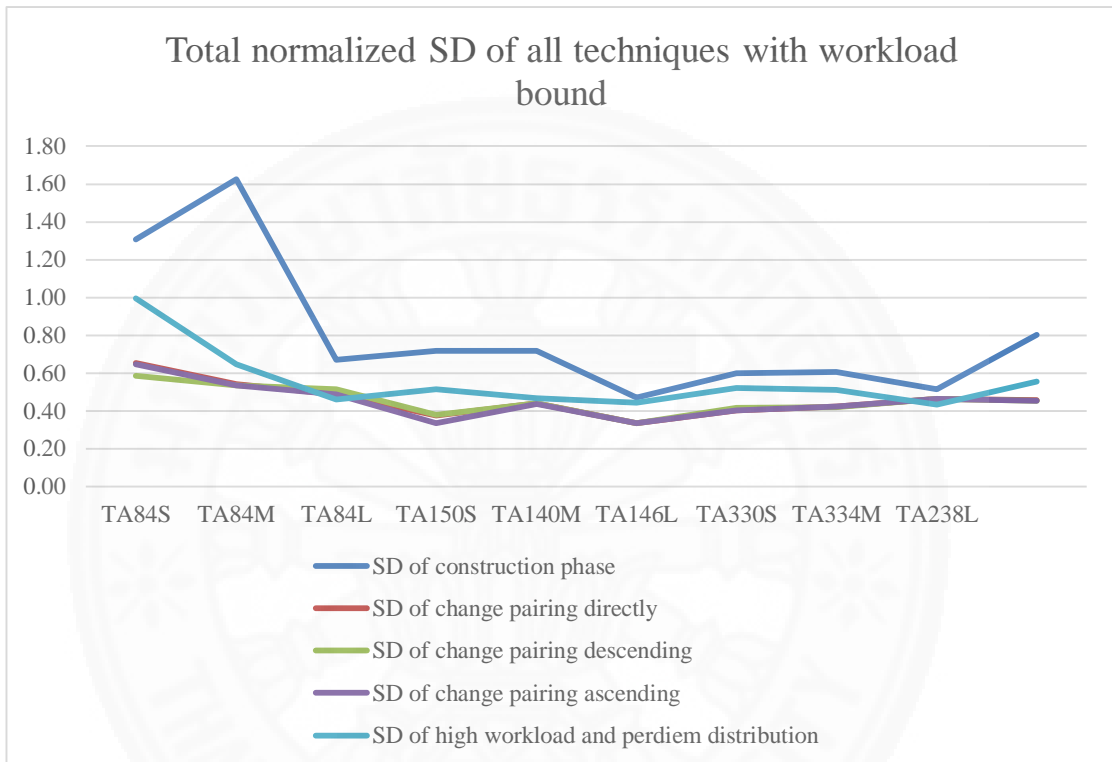


Figure 5.2 Total normalized SD of all techniques with workload bound graph

Figure 5.2 represents comparison of all improvement techniques. This graph shows that all techniques can reduce total normalization SD of workload and perdiem, but not too much compared with non-bound method. The high workload and perdiem distribution produced the worst results, especially in TA146L, and TA238L. The other methods also produced results similar to high workload and distribution method.

5.6 Minimize SD of workload and perdiem simultaneously with perdiem bound

This section shows the result of SD of workload and perdiem minimization with perdiem bound. The perdiem limitation also construct for reduce the variation of total workload and perdiem value. The method also applied normalization algorithm in order to standardize value of workload and perdiem.

Table 5.11 SD of workload table using workload and perdiem minimization with perdiem bound

Instance	Iteration	SD of Construction Phase	SD of each Improvement Phase			
			Change pairing directly	Change pairing descending	Change pairing ascending	High WL and PD distribution
TA84S	1	25.8306	12.8448	10.4426	13.0062	30.6717
	10	25.8306	11.0263	10.4426	10.1427	28.8094
	100	25.8306	11.0263	10.4426	10.1427	28.8094
TA84M	1	41.4780	15.9578	15.9578	15.9578	25.0943
	10	41.4780	15.9578	15.9578	13.1517	19.8105
	100	41.4780	15.9578	15.9578	13.1517	19.8105
TA84L	1	28.2833	20.1555	21.1011	20.1555	26.1290
	10	28.2833	20.1555	21.1011	20.1555	17.1138
	100	28.2833	20.1555	21.1011	20.1555	17.1138
TA150S	1	23.3733	10.2592	10.0024	10.0270	17.7764
	10	23.3733	10.8423	11.4715	9.4126	15.4658
	100	23.3733	10.8423	11.4715	9.4126	15.4658
TA140M	1	26.6441	15.8752	15.4932	15.8676	25.3031
	10	26.6441	15.8752	15.4932	15.2847	19.6962
	100	26.6441	15.8752	15.4932	15.2847	19.6962
TA146L	1	19.0579	15.4303	15.4303	15.4303	18.6793
	10	19.0579	15.4303	15.4303	15.4303	18.4734
	100	19.0579	15.4303	15.4303	15.4303	18.4734
TA330S	1	21.8071	14.3296	14.5260	14.5268	19.2230
	10	21.8071	14.0333	14.5260	13.8594	16.6858
	100	21.8071	14.0333	14.5260	13.8594	16.6858
TA334M	1	24.4522	16.5527	16.5527	16.5527	23.1487
	10	24.4522	16.5527	16.7255	16.2434	21.9446
	100	24.4522	16.5527	16.7255	16.2434	21.9446
TA238L	1	19.8712	18.0734	18.0734	18.0734	20.0679
	10	19.8712	18.0734	18.0734	18.0734	15.5680
	100	19.8712	18.0734	18.0734	18.0734	15.5680

The interesting thing in table 5.11 is there are two techniques that produce opposite results to each other: the change pairing ascending technique and high workload and perdiem distribution technique. The change pairing ascending techniques produced good solutions in TA84S, TA84M, TA150S, TA140M, TA146L, TA330S, and TA334M. The other two instances are the worst compared with the high workload and perdiem distribution which produced best solution in these two instances. The two best solutions come from long destination routes similar to other experiments.

Table 5.12 SD of perdiem table using workload and perdiem minimization with perdiem bound

Instance	Iteration	SD of Construction Phase	SD of each Improvement Phase			
			Change pairing directly	Change pairing descending	Change pairing ascending	High WL and PD distribution
TA84S	1	4952.6460	2712.8418	2493.4668	2673.1042	2508.3445
	10	4952.6460	2609.4407	2493.4668	2824.9751	2408.4204
	100	4952.6460	2609.4407	2493.4668	2824.9751	2408.4204
TA84M	1	11208.2402	3678.1956	3678.1956	3678.1958	4395.8457
	10	11208.2402	3678.1956	3678.1956	3766.6455	3728.0747
	100	11208.2402	3678.1956	3678.1956	3766.6455	3728.0747
TA84L	1	6143.6519	4557.7705	4765.2573	4557.7705	5463.3770
	10	6143.6519	4557.7705	4765.2568	4557.7705	3485.3069
	100	6143.6519	4557.7705	4765.2568	4557.7705	3485.3069
TA150S	1	4683.0742	3165.5696	3016.8967	2999.4424	3873.4058
	10	4683.0742	2997.5266	2624.4651	2965.9031	3518.6279
	100	4683.0742	2997.5266	2624.4651	2965.9031	3518.6279
TA140M	1	7169.1729	4519.6846	4660.0034	4591.3013	5026.7549
	10	7169.1729	4519.6846	4660.0034	4636.9380	4050.0896
	100	7169.1729	4519.6846	4660.0034	4636.9380	4050.0896
TA146L	1	4999.6499	3133.8350	3133.8350	3133.8350	4452.2598
	10	4999.6499	3133.8350	3133.8350	3133.8350	4371.8535
	100	4999.6499	3133.8350	3133.8350	3133.8350	4371.8535
TA330S	1	6022.7480	4289.7637	4367.2778	4367.2788	5077.1250
	10	6022.7480	4238.6699	4367.2778	4378.7500	5255.8032
	100	6022.7480	4238.6699	4367.2778	4378.7500	5255.8032
TA334M	1	7148.8447	5199.1387	5199.1382	5199.1382	5989.0928
	10	7148.8447	5199.1387	5122.2197	5213.6665	5618.5679
	100	7148.8447	5199.1387	5122.2197	5213.6665	5618.5679
TA238L	1	6331.2559	5653.2837	5653.2842	5653.2827	5813.6294
	10	6331.2559	5653.2837	5653.2827	5653.2827	5291.3511
	100	6331.2559	5653.2837	5653.2827	5653.2827	5291.3511

Table 5.12 shows that the high workload and perdiem distribution can produce a lot of good solutions, such as TA84S, TA84L, TA140M, and TA238L. The interesting thing is this technique will reduce a lot of SD in workload or perdiem in order to minimize total SD of workload and perdiem. The change pairing descending techniques also produced four best results in TA84M, TA150S, TA146L, and TA334M. The change pairing directly produced best solutions in instances TA84M, TA146L, and TA330S. The change pairing ascending can produce only one best result, TA146L, in contrast with SD of workload result where it produced a lot of best solutions.

Table 5.13 Total normalized workload and perdiem SD (Twp) with perdiem bound table

Instance	SD of Construction phase	SD and percent changed of each Improvement Phase							
		SD Tech 1	SD Changed Tech 1 (%)	SD Tech 2	SD Changed Tech 2 (%)	SD Tech 3	SD Changed Tech 3 (%)	SD Tech 4	SD Changed Tech 4 (%)
TA84S	1.3080	0.6329	51.6149	0.6027	53.9212	0.6461	50.6058	0.9899	24.3201
TA84M	1.6257	0.5682	65.0484	0.5682	65.0484	0.5347	67.1104	0.6297	61.2657
TA84L	0.6710	0.4883	27.2245	0.5108	23.8637	0.4883	27.2245	0.3928	41.4489
TA150S	0.7183	0.3903	45.6548	0.3751	47.7754	0.3640	49.3147	0.5044	29.7806
TA140M	0.7169	0.4397	38.6579	0.4418	38.3663	0.4379	38.9143	0.4664	34.9469
TA146L	0.4713	0.3353	28.8630	0.3353	28.8630	0.3353	28.8630	0.4328	8.1690
TA330S	0.5998	0.4044	32.5826	0.4176	30.3864	0.4091	31.7912	0.4918	18.0156
TA334M	0.6059	0.4266	29.5976	0.4250	29.8507	0.4237	30.0722	0.5074	16.2597
TA238L	0.5162	0.4647	9.9784	0.4647	9.9784	0.4647	9.9784	0.4195	18.7296
AVG	0.8037	0.4612	36.5802	0.4601	36.4504	0.4560	37.0972	0.5372	28.1040

Table 5.13 shows similar result with table 5.10. The best average solution is the change pairing ascending technique and the worst solution is the high workload and perdiem distribution technique. The values also are similar, with the average total normalized SD of workload and perdiem, and percentage changed of the change pairing ascending technique of 0.4560 and 37.0972%, respectively. The average total normalized SD of workload and perdiem, and percentage changed of high workload and perdiem distribution are 0.5372 and 28.1040%. The change pairing descending can produce only two best solutions, less than the workload bound result.

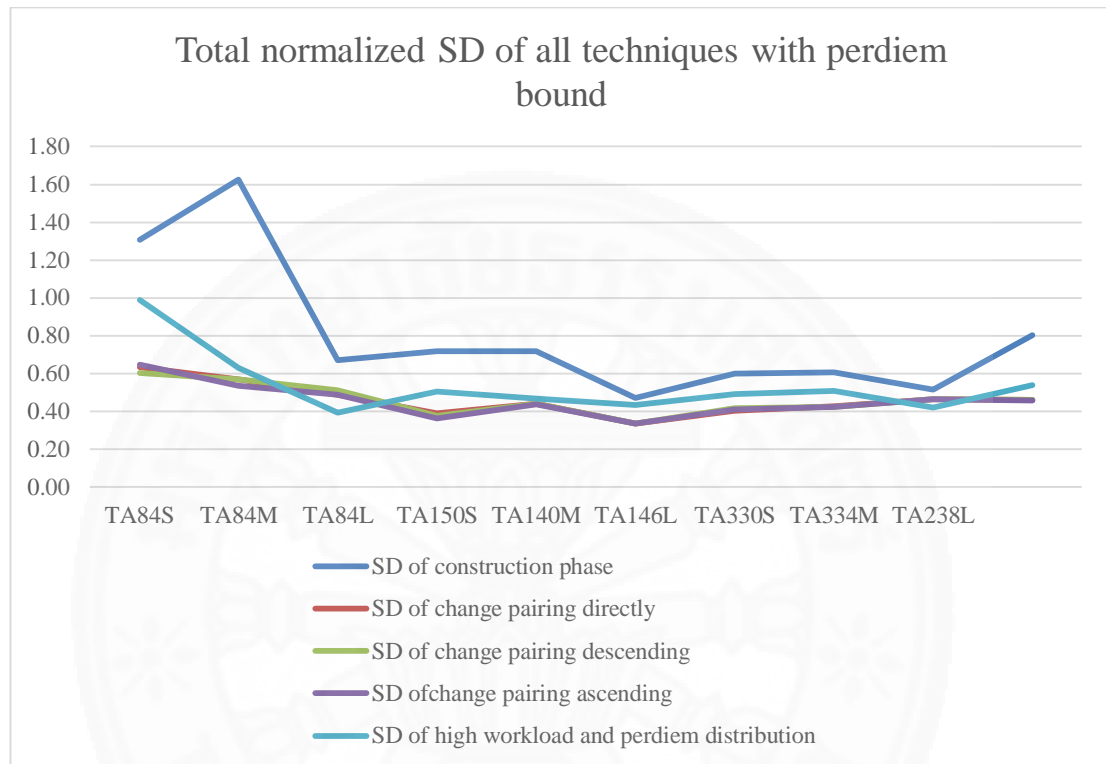


Figure 5.3 Total normalized SD of all techniques with perdiem bound graph

Figure 5.3 shows the best SD reduction in instances TA84S, and TA84M. But similar to the workload bound method, all techniques reduced SD only slightly from the construction phase. The graph shows very little reduction in instances TA146L and TA238L. However, all three change pairing techniques can reduce higher total normalized SD of workload and perdiem than the high workload and perdiem distribution technique.

5.7 Compare solution

This section shows the comparison of SD minimization in each instance between all three methods: minimize both workload and perdiem, minimize workload, and minimize paerdiem. These tables also show the best and worst SD of each instance. The results are divided into two tables, workload and perdiem.

Table 5.14 Result comparison SD of workload

Instance	Method	Best WL	Best Tech	Worst WL	Worst Tech
TA84S	min only workload	9.075	1	13.2422	4
	min only perdiem	20.1775	4	41.8933	3
	min both without bound	10.1846	2	17.7565	4
	min both with workload bound	10.1846	2	12.8267	4
	min both with perdiem bound	10.1427	3	28.8094	4
TA84M	min only workload	8.0025	3	21.2779	4
	min only perdiem	21.2231	4	35.145	3
	min both without bound	11.1570	1,3	21.4041	4
	min both with workload bound	11.4641	3	18.6937	4
	min both with perdiem bound	13.1517	3	19.8105	4
TA84L	min only workload	14.5692	4	21.1109	2
	min only perdiem	15.6292	4	20.8992	3
	min both without bound	20.1555	1,3	26.1290	4
	min both with workload bound	17.5597	4	20.9574	3
	min both with perdiem bound	17.1138	4	20.1555	1,3
TA150S	min only workload	7.5526	1	11.7359	4
	min only perdiem	17.8067	4	35.6293	1
	min both without bound	9.1962	3	14.8895	4
	min both with workload bound	8.8061	1	14.4971	4
	min both with perdiem bound	9.4126	3	15.4658	4
TA140M	min only workload	13.4063	4	15.8951	3
	min only perdiem	14.5087	4	26.424	1
	min both without bound	15.1985	1	25.2282	4
	min both with workload bound	15.2929	2,3	19.4282	4
	min both with perdiem bound	15.2847	3	19.6962	4
TA146L	min only workload	12.0618	4	14.1544	1,2,3
	min only perdiem	13.3307	4	21.0767	2,3
	min both without bound	15.4303	1,2,3	18.3158	4
	min both with workload bound	15.4303	1,2,3	18.3415	4
	min both with perdiem bound	15.4303	1,2,3	18.4734	4
TA330S	min only workload	13.4339	1	14.2736	4
	min only perdiem	17.0778	4	27.0843	2
	min both without bound	13.2523	1	20.4866	4
	min both with workload bound	13.8558	3	20.4675	4
	min both with perdiem bound	13.8594	3	16.6858	4
TA334M	min only workload	14.5659	1,2,3	16.6937	4
	min only perdiem	19.4801	4	30.5211	2
	min both without bound	15.1432	1	22.9227	4
	min both with workload bound	15.8150	2	23.3990	4
	min both with perdiem bound	16.2434	3	21.9446	4
TA238L	min only workload	12.3839	4	17.0812	1,2,3
	min only perdiem	14.909	4	20.1902	2
	min both without bound	18.0734	1	20.0006	4
	min both with workload bound	14.6170	4	18.0734	1,2,3
	min both with perdiem bound	15.5680	4	18.0734	1,2,3

From table 5.14, the high workload or perdiem distribution technique (Tech 4) produced mostly the best solution in minimize only workload and perdiem methods. This is in contrast with minimize workload and perdiem simultaneously method which produced most of the worst solutions. The high workload and perdiem

distribution technique produces best SD of workload solution in most of long range flight instances. In TA84L and TA238L, it produced the best solution in most methods except minimize both workload and perdiem simultaneously without bound method.

The change pairing ascending mostly produced results in minimize workload and perdiem simultaneously method for all of without bound, workload bound, and perdiem bound. This technique yielded satisfactory results in most of small route and various mixed distance routes. However, this technique produced the worst solution for long distance routes.

Table 5.15 Result comparison SD of perdiem

Instance	Method	Best PD	Best Tech	Worst PD	Worst Tech
TA84S	min only workload	4932.8398	2	5303.6177	4
	min only perdiem	1477.4646	2	4376.6812	4
	min both without bound	2336.9778	3	2470.4265	4
	min both with workload bound	2423.3889	2	4761.7637	4
	min both with perdiem bound	2408.4204	4	2824.9751	3
TA84M	min only workload	5355.6890	3	8289.5449	4
	min only perdiem	2516.7654	3	8228.6641	4
	min both without bound	3959.2622	3	4207.0645	4
	min both with workload bound	3872.2119	1	4090.7754	4
	min both with perdiem bound	3678.1956	1,2	3766.6455	3
TA84L	min only workload	3382.1333	4	4745.6094	2
	min only perdiem	3422.9509	4	4647.9175	3
	min both without bound	4557.7700	3	5463.3765	4
	min both with workload bound	4557.7705	1,3	4873.2905	2
	min both with perdiem bound	3485.3069	4	4765.2568	2
TA150S	min only workload	4446.7505	4	5371.4683	3
	min only perdiem	1342.0878	3	3538.0178	4
	min both without bound	2439.6409	2	5480.1914	4
	min both with workload bound	2383.4795	3	3931.7058	4
	min both with perdiem bound	2624.4651	3	3518.6279	4
TA140M	min only workload	4517.5200	4	4971.8882	1,3
	min only perdiem	2848.2712	3	4615.4595	4
	min both without bound	4498.1973	1	5085.0986	4
	min both with workload bound	4147.2964	4	4683.1929	2
	min both with perdiem bound	4050.0896	4	4660.0034	2
TA146L	min only workload	3790.8203	4	4058.6511	1,3
	min only perdiem	2981.0764	1	3235.1953	4
	min both without bound	3133.8350	1,2	4725.9521	4
	min both with workload bound	3133.8350	1,2,3	4589.9175	4
	min both with perdiem bound	3133.8350	1,2,3	4371.8535	4
TA330S	min only workload	5622.9077	4	5946.7412	2
	min only perdiem	3075.2869	1	5071.7715	4
	min both without bound	4127.5513	1	5210.0557	4
	min both with workload	4218.7290	1	4808.0405	4

	bound min both with perdiem bound	4238.6699	1	5255.8032	4
TA334M	min only workload	6239.8481	4	6329.3433	1
	min only perdiem	3811.0105	3	5514.3535	4
	min both without bound	5090.5601	2	6079.0552	4
	min both with workload bound	5184.9219	1	5346.5132	4
	min both with perdiem bound	5122.2197	2	5618.5679	4
TA238L	min only workload	5291.8179	4	5872.3545	1
	min only perdiem	4653.7144	3	4966.5522	4
	min both without bound	5653.2827	2	5846.9331	4
	min both with workload bound	5653.2837	1,2	5863.3257	4
	min both with perdiem bound	5291.3511	4	5653.2837	1

From table 5.15, the result shows various techniques that are suitable for minimization in different scenarios. For example, the change pairing directly technique is suitable for instances TA146L and TA330S for most methods, except minimization only workload. In contrast, the high workload and perdiem distribution technique is suitable for long distance routes in workload minimization. Moreover, this technique produced a lot of worst solutions in many instances, for example; TA84S, TA83M, TA150S, TA146L, TA330S, and TA334M.

Table 5.16 Result comparison of percentage changed of total normalized workload and perdiem

Instance	Workload and perdiem minimization simultaneously without bound				Workload and perdiem minimization simultaneously with workload bound				Workload and perdiem minimization simultaneously with perdiem bound			
	SD Changed Tech 1 (%)	SD Changed Tech 2 (%)	SD Changed Tech 3 (%)	SD Changed Tech 4 (%)	SD Changed Tech 1 (%)	SD Changed Tech 2 (%)	SD Changed Tech 3 (%)	SD Changed Tech 4 (%)	SD Changed Tech 1 (%)	SD Changed Tech 2 (%)	SD Changed Tech 3 (%)	SD Changed Tech 4 (%)
TA84S	51.9454	55.1571	54.1526	42.0091	49.8820	55.1572	50.4857	23.8504	51.6149	53.9212	50.6058	24.3201
TA84M	67.8555	66.6999	67.8555	57.1551	66.5925	67.0398	67.0423	60.2672	65.0484	65.0484	67.1104	61.2657
TA84L	27.2245	23.1982	27.2245	9.4080	27.2245	23.1982	27.2245	31.1582	27.2245	23.8637	27.2245	41.4489
TA150S	53.3130	54.6558	50.8323	12.2197	47.5144	47.0041	53.4225	28.0709	45.6548	47.7754	49.3147	29.7806
TA140M	40.0579	38.6823	39.0331	17.4033	38.6579	38.5709	38.7980	34.7511	38.6579	38.3663	38.9143	34.9469
TA146L	28.8630	28.8630	28.8630	4.7450	28.8630	28.8630	28.8630	6.1450	28.8630	28.8630	28.8630	8.1690
TA330S	35.2807	34.6086	34.6086	9.8394	32.7740	30.4074	32.6471	13.2781	32.5826	30.3864	31.7912	18.0156
TA334M	33.0763	33.0763	33.0763	10.9454	29.9556	30.6242	29.9335	15.5627	29.5976	29.8507	30.0722	16.2597
TA238L	9.9784	9.9784	9.9784	3.9960	9.9784	9.9784	9.9784	15.7811	9.9784	9.9784	9.9784	18.7296
AVG	38.6216	38.3244	38.4027	18.6357	36.8269	36.7604	37.5994	25.4294	36.5802	36.4504	37.0972	28.1040

From table 5.16, the change pairing directly technique can produce high percentage change for many instances, especially in minimize workload and perdiem simultaneously without bound such as in instances TA84M, TA140M, TA146L, TA330S, and TA334M. It also produced the best average percentage change of 38.6216%. Most cases in which it produced an optimal solution are various mixed distance route instances. The change pairing descending produced the best solutions in instances TA84S, TA150S, TA146L, and TA334M that also occurs in the no bound method. The change pairing ascending produced three best solutions that also happen in the no bound method as in TA84M, TA146L, and TA334M. The interesting thing is all three change pairing techniques give best results for instance TA146L in all situations. The high workload and perdiem distribution can produce only two best solutions, in long distance route instances where all of them occur in minimization of workload and perdiem simultaneously with perdiem bound. The average also shows that the high workload and perdiem technique produce the worst percentage changes which are 18.6357% for non-bound, 25.4294% for workload bound, and 28.1040% for perdiem bound.

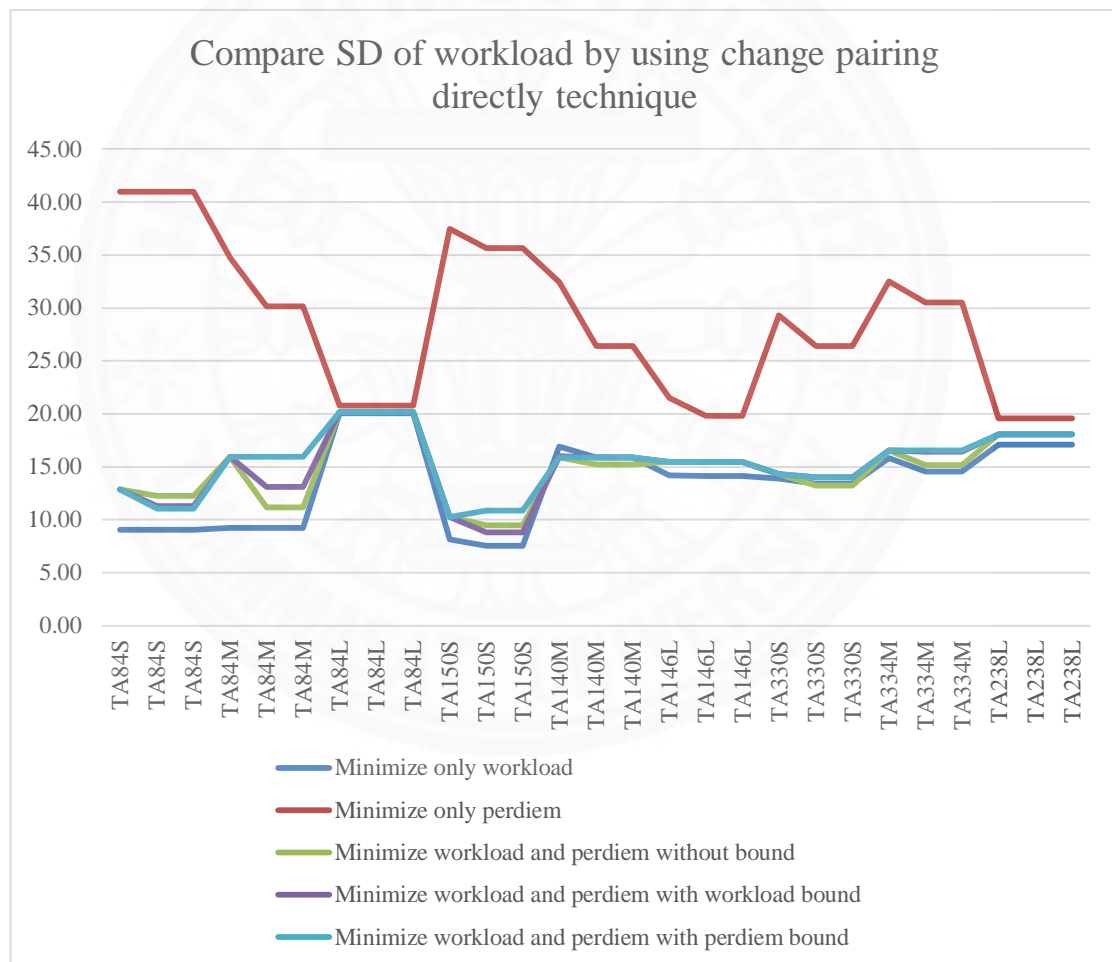


Figure 5.4 Compare SD of workload by using change pairing directly technique

From figure 5.4, the SD of workload of perdiem minimization is very high because the method focuses on reducing only the SD of perdiem. This figure shows that there is no relation between workload and perdiem. The other methods are similar but the workload minimization method yields a high reduction. The

minimization of both workload and perdiem with perdiem bound produced higher values than other methods, except the perdiem minimization method.

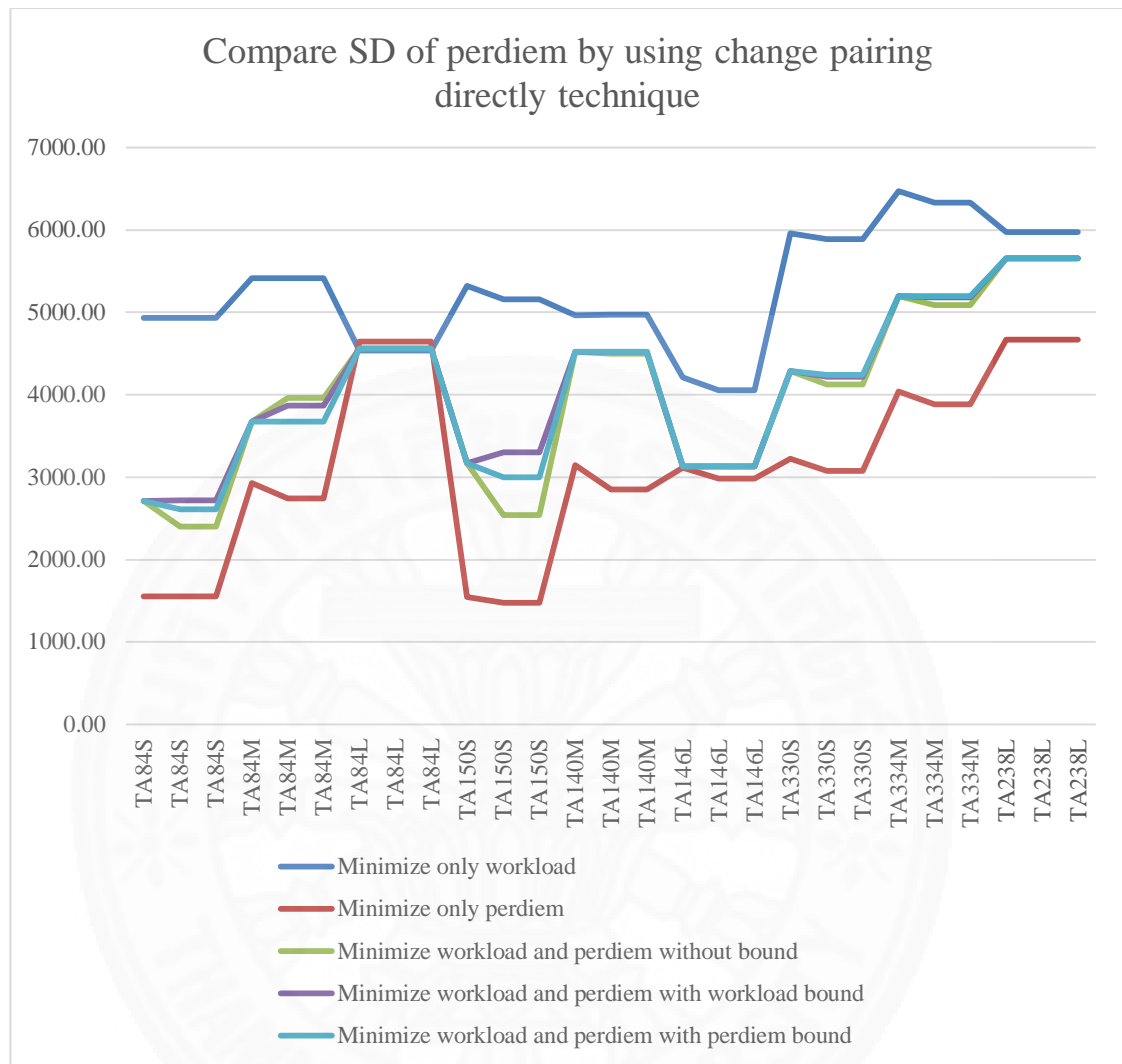


Figure 5.5 Compare SD of perdiem by using change pairing directly technique

Figure 5.5 shows various results of minimization. The best method is perdiem minimization but it also has the worst solution in instance TA84L. The workload minimization method produced the worst solution for most instances, except in TA84L. The three workload and perdiem simultaneous minimization produced similar results. But the non-bound method mostly produced worse solutions than others in short distance route instances.

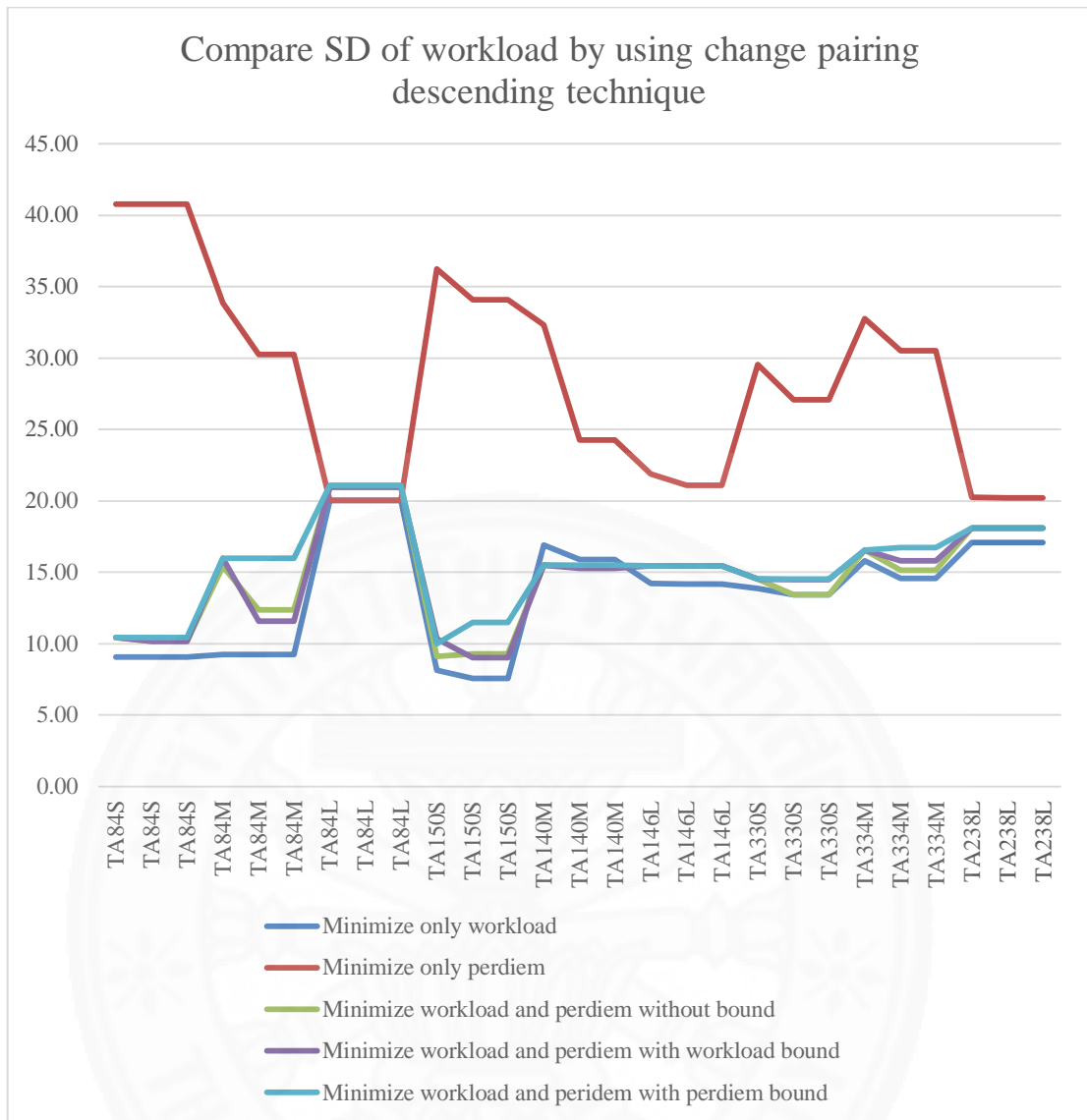


Figure 5.6 Compare SD of workload by using change pairing descending technique

From Figure 5.6, the perdiem minimization method, and minimize both with perdiem bound method did not produce good solutions. The perdiem minimization method yield satisfaction in minimal solution in long distance route instance. The perdiem bound method produced the worst solution in almost every instance except for the perdiem minimization method that aims to reduce only SD of perdiem. The other methods produced satisfactory minimization results.

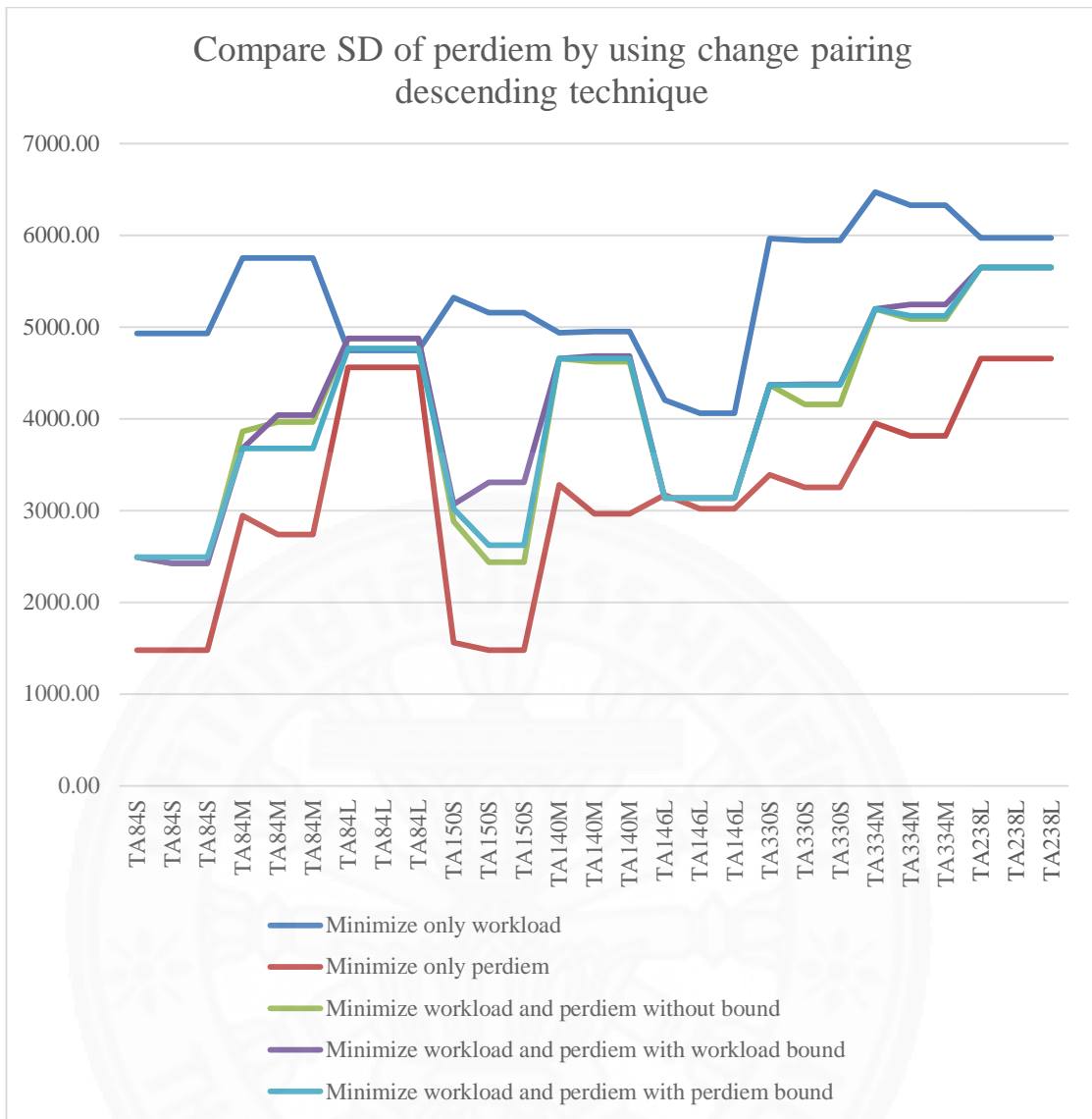


Figure 5.7 Compare SD of perdiem by using change pairing descending technique

Figure 5.7 shows that the change pairing descending technique produced closer results for all methods than the change pairing directly technique. The perdiem minimization method also produced the best solution. But there has worst result in long distance route instance. The three that minimize both SD of workload and perdiem simultaneously produced similar results. Even though the workload bound method yielded the worst result in TA150S instance and most of short distance route instances.

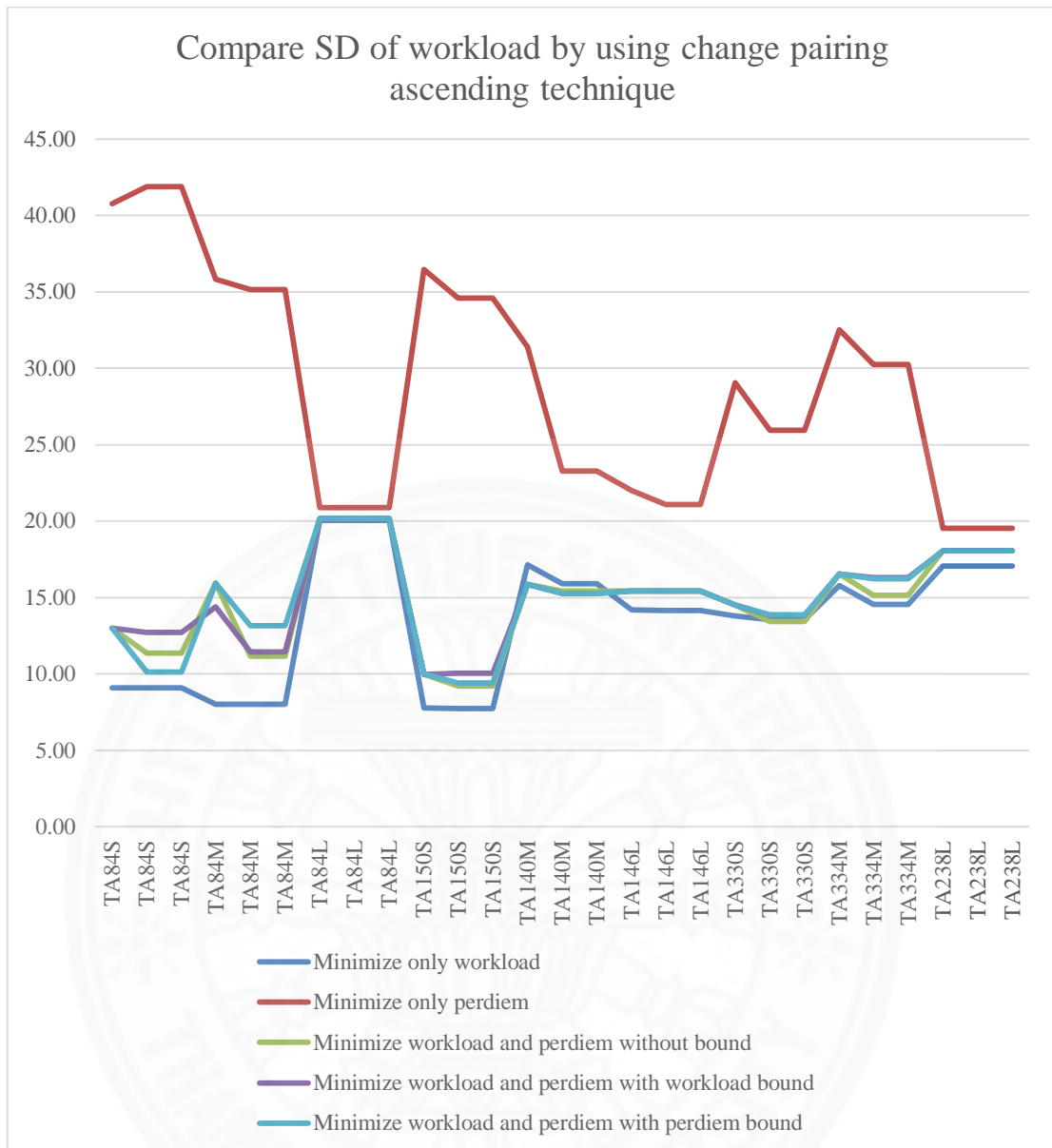


Figure 5.8 Compare SD of workload by using change pairing ascending technique

Figure 5.8 shows the results of workload minimization, without bound minimization, workload bound minimization, and perdiem bound minimization. The figure shows that the change pairing ascending technique can produce best result of SD of workload minimization for all methods except perdiem minimization method. From the graph, the minimization of workload and perdiem simultaneously with workload bound can produce average satisfactory results. However, it still produced the worst result in instance TA84S.

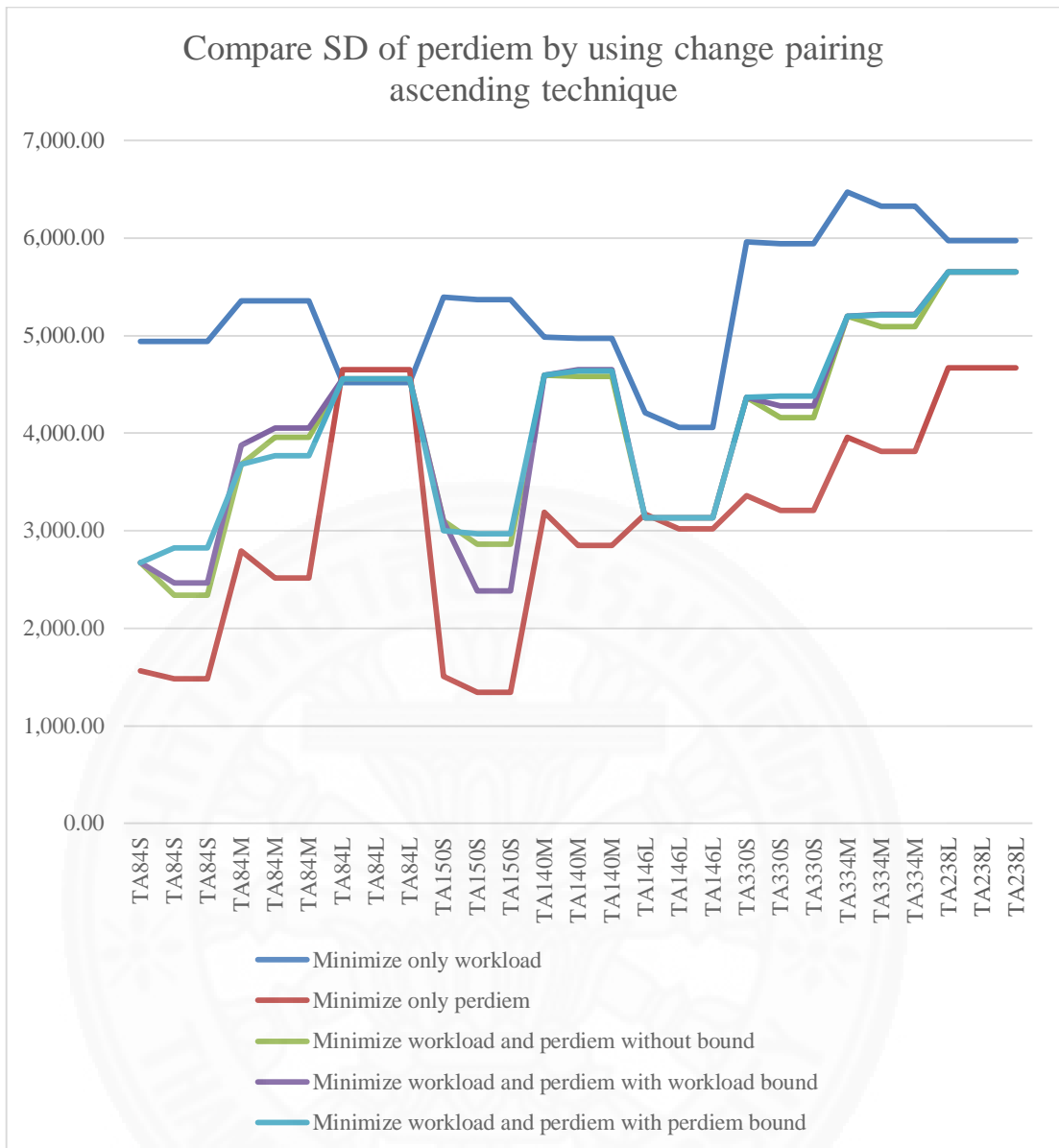


Figure 5.9 Compare SD of perdiem by using change pairing ascending technique

From Figure 5.9, the result of the change pairing ascending can be seen as fluctuating. The workload minimization method produced most of the worst solutions, except in instance TA84L. The three non-bound, workload bound, and perdiem bound methods produced similar results. The best method still is the perdiem minimization method but it produced very high SD of workload. The workload bound method produced average good solution especially in instance TA150S.

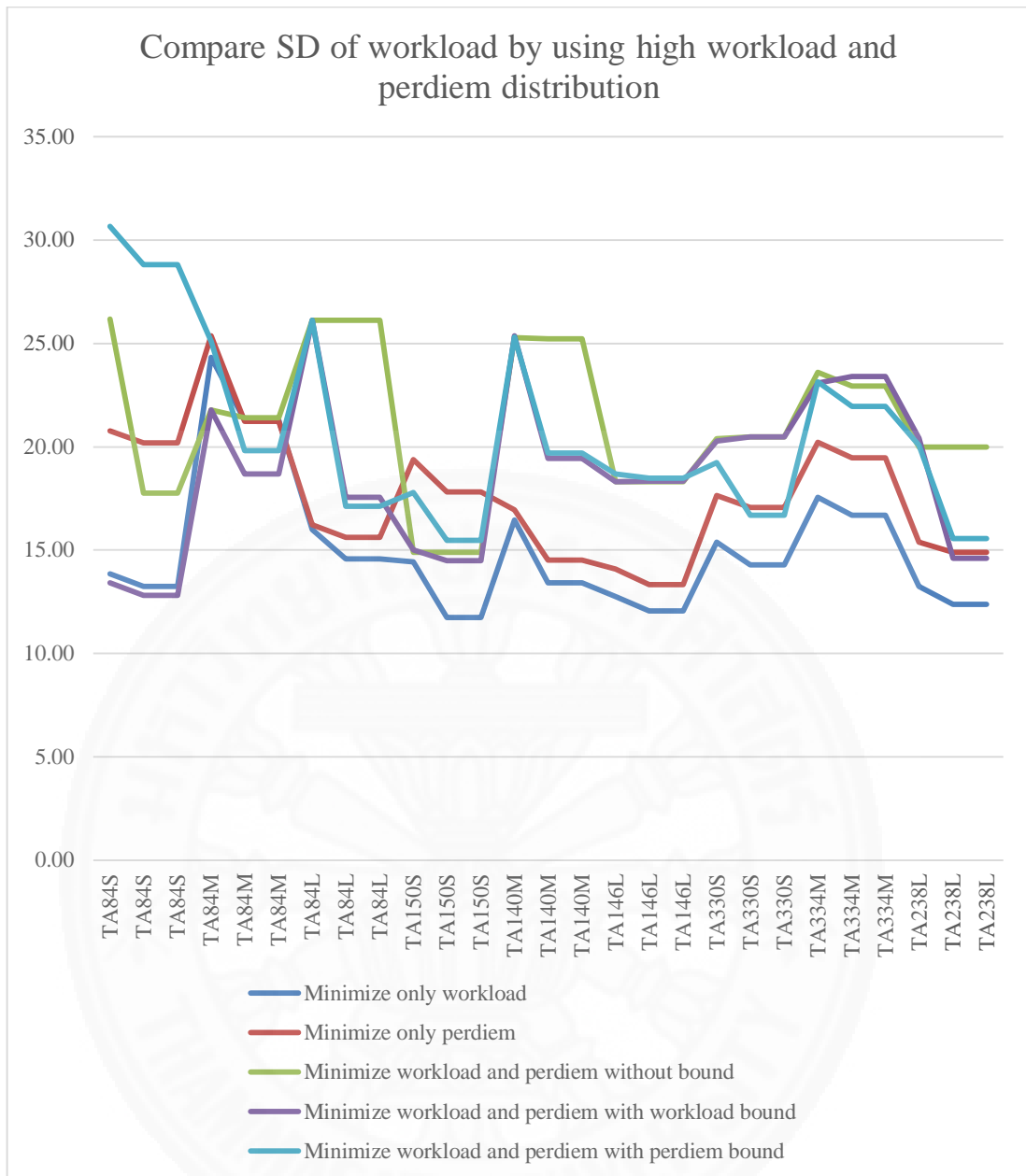


Figure 5.10 Compare SD of workload by using high workload and perdiem distribution technique

Figure 5.10 shows the much fluctuated result especially in short, and various distance route instances. The reason is there are more pairings for dispersal which may cause many unexpected results. Thus, this technique is suitable for long distance route instances. The three non-bound, workload bound method, and perdiem bound methods produced high SD of workload in many test instances. The non-bound method is the worst method in this technique. It results in high SD of workload in many instances, for example, TA84L and TA146L.

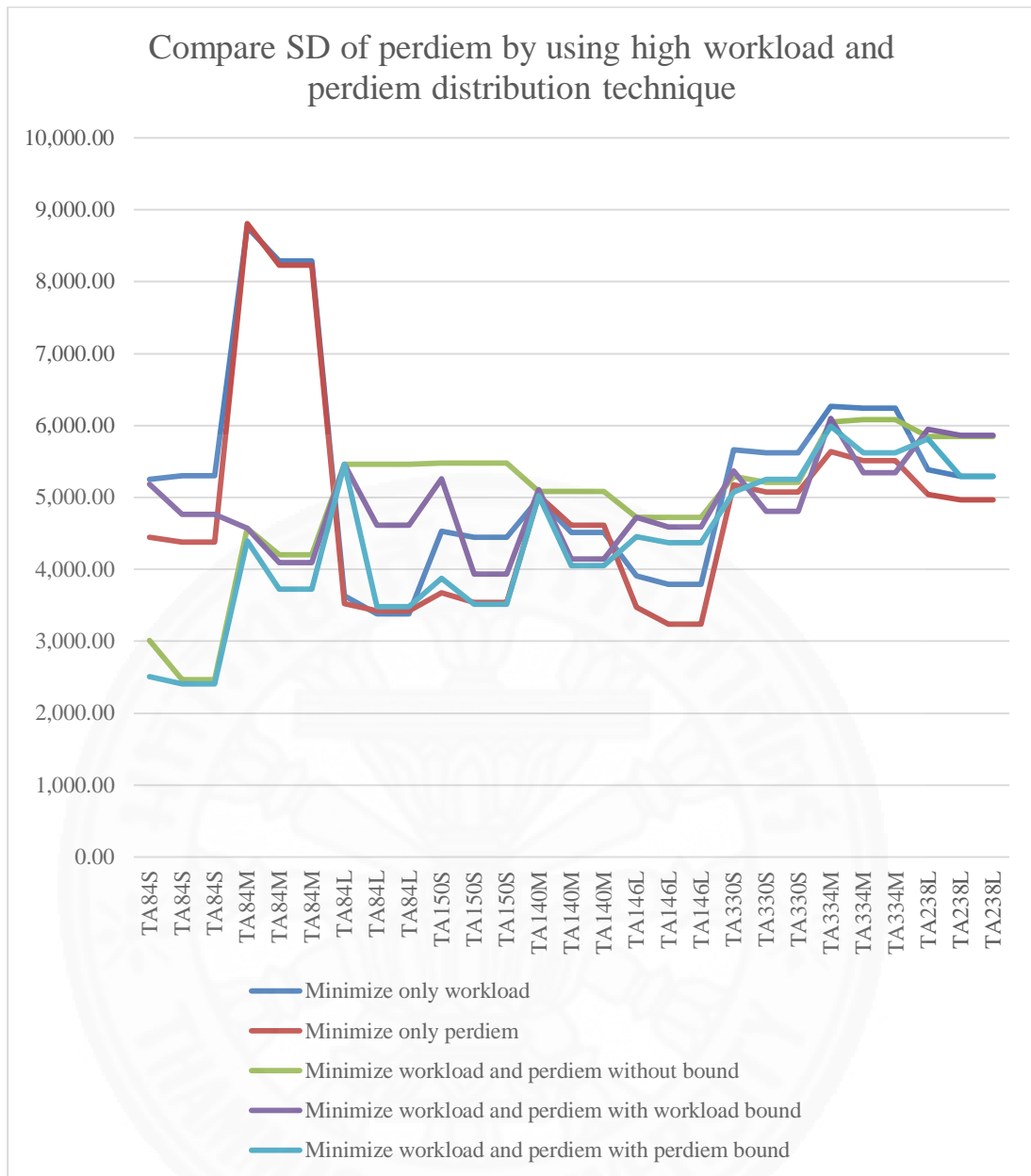


Figure 5.11 Compare SD of perdiem by using high workload and perdiem distribution technique

Figure 5.11 also shows the fluctuated result, with every method going up and down in every test instance. The interesting thing is that workload minimization and perdiem minimization methods produced very high SD of perdiem in instance TA84M. The reason may be that the instance has only a few pairings for moving. Another interesting thing is that all methods produced similar results in large pairing size instances TA330S, TA334M, and TA238L. The reason may be there are many pairing choices for moving.

Chapter 6

Conclusion and Further Study

6.1 Conclusion

This chapter proposed the method to solve Thai Airways Crew Rostering Problem by using the Greedy Algorithm. The objective is to minimize SD of both workload and perdiem simultaneously. The bound also was set to limit the variation of results. Thus, this experiment includes minimization of workload and perdiem simultaneously with non-bound, workload bound, and perdiem bound. This thesis also experimented on minimizing workload and perdiem separately for comparison with the objective result. A Greedy Algorithm was applied in this thesis because of its simplicity and low computation time. The Greedy Algorithm in this thesis can be divided into two phases: the construction phase and the improvement phase. First, the construction phase aims to construct a simple crew table for the improvement phase. Second, the improvement phase aims balance the SD of both workload and perdiem. The improvement phase is divided into four techniques as follows:

- Change pairing directly
- Change pairing descending
- Change pairing ascending
- High combined workload and perdiem distribution

From the experiment, the result of minimization of both workload and perdiem simultaneously for non-bound, workload bound, and perdiem bound mostly yielded average SD minimization. Thus, this means it reduces both workload and perdiem, and has no abnormally high SD. This differs from minimizing SD of workload and perdiem separately, which produces some values of SD which are increased, not reduced.

The different improvement techniques can produce different best solutions. The change pairing ascending can produce a lot of best solutions in minimization of workload and perdiem simultaneously with workload bound and perdiem bound, which are 4, and 5 respectively. But after comparison to all situations, the change pairing directly technique produced a greater number of best solutions than other techniques. Most of the best solutions occur in non-bound situations. Moreover, the high workload and perdiem distribution can produce a few best solutions, but mostly produced the worst solutions in all situations.

In conclusion, these techniques produced minimal SD of workload and perdiem but not a global optimal solution. The reason for this comes from the nature of the Greedy Algorithm to find only the optimal solution at some point of the problem. Although the program compilation time is quite small, about one to five seconds, it depends on the complexity of code, number of instances and constraints, and performance of compiler tool.

6.2 Further Study

In further study, the researcher will continue to construct new implemented solutions with various algorithms, such as Hill Climbing, in order to find a global optimal solution in order to reduce more SD.

References

- Christian, Artigues, Michel, Gendreau, Louis-Martin, Rousseau, and et al., (2009). Solving an integrated employee timetabling and job-shop scheduling problem via hybrid branch-and-bound. *Computers & Operations Research*, 36, 2330 – 2340.
- A.,T., Ernst, H., Jiang, M., Krishnamoorthy, and D., Sier., (2004). Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research*, 153, 3–27.
- Zhang, Yinghui, Rao, Yunbo, and Zhou, Mingtian., (2007). GASA Hybrid Algorithm Applied in Airline Crew Rostering System. *TSINGHUA SCIENCE AND TECHNOLOGY*, ISSN 1007-0214, 46/49, pp255-259.
- Walid, El, Moudanis, Carlos, Albert, Nunes, Cosenza, and Felix, Mora-Camino., (2001). An Intelligent Approach for Solving the Airlines Crew Rostering Problem. *IEEE*.
- Nadia, Souai, and Jacques, Teghem., (2009). Genetic algorithm based approach for the integrated airline crew-pairing and rostering problem. *European Journal of Operational Research*, 199, 674–683.
- Dusan, Teodorovic, and Panta, Lucic., (1998). A fuzzy set theory approach to the aircrew rostering problem. *Fuzzy Sets and Systems*, 95. 261-271.
- Lucio, Bianco, Maurizio, Bielli, Salvatore, Ricciardelli, and et al., (1992). A heuristic procedure for the crew rostering problem. *European Journal of Operational Research*, 58, 272-283.
- Herbert, Dawid, Johannes, Konig, and Christine, Strauss., (2001). An enhanced rostering model for airline crews. *Computers & Operations Research*, 28, 671-688.
- Michel, Gamache, Alain, Hertz, and Jerome, Olivier, Ouellet., (2007). A graph coloring model for a feasibility problem in monthly crew scheduling with preferential bidding. *Computers & Operations Research*, 34, 2384 – 2395.
- Guang-Feng, Deng, and Woo-Tsong, Lin., (2011). Ant colony optimization-based algorithm for airline crew scheduling problem. *Expert Systems with Applications*, 38, 5787–5793.
- F., M., Zeghal, and M., Minoux., (2006). Modeling and solving a Crew Assignment Problem in air transportation. *European Journal of Operational Research*, 175, 187–209.

- Shangyao, Yan, Ching-Hui, Tang, and Tseng-Chih, Fu., (2008). An airline scheduling model and solution algorithms under stochastic demands. *European Journal of Operational Research*, 190, 22–39.
- Oliver, Weidea, David, Ryan, and Matthias, Ehrgott., (2010). An iterative approach to robust and integrated aircraft routing and crew scheduling. *Computers & Operations Research*, 37, 833 – 844.
- Thomas, Emden-Weinert, and Mark, Proksch., (1999). Best Practice Simulated Annealing for the Airline Crew Scheduling Problem. *Journal of Heuristics*, 5, 419–436.
- Sydney, C.,K., Chu., (2007). Generating, scheduling and rostering of shift crew-duties: Applications at the Hong Kong International Airport. *European Journal of Operational Research*, 177, 1764–1778.
- Azadeh, Hosseinabagi, Eivazy, and et al., (2013). A hybrid meta-heuristics algorithm for optimization of crew scheduling. *Applied Soft Computing*, 13, 158-164.
- Anne, Mercier, and Francois, Soumis., (2005). An integrated aircraft routing, crew scheduling and flight retiming model. *Computer & Operations Research*, 34, 2251-2265.
- Claude, P., Medard, and Nidhi, Sawhney., (2007). Airline crew scheduling from planning to operations. *European Journal of Operation Research*, 183, 1013-1027.
- Panta, Lucic, and Dusan, Teodorovic., (2007). Metaheuristics approach to the aircrew rostering problem. *Ann Oper Res*, 155, 311-338.
- Shih-Wei, Lin, and Kuo-Ching, Ying., (2014). Minimizing shifts for personnel task scheduling problems: A three-phase algorithm. *European Journal of Operation Research*, 237, 323-334.
- Korhan, Karabulut, and M., Fatih, Tasgetiren., (2014). A variable iterated greedy algorithm for the travelling salesman problem with time windows. *Information Sciences*, 279, 383-395.



Appendices

Appendix A

Test Instance

The test instance of small pairing with various mix distance route of Thai Airways, TA84M.

Pairing No.	Day	DepFlt	ArrFlt	Total Block Time	Adjusted ArrDay-DepDay	ArrDay-DepDay	TTL THB	Workload Score	Zone	Station
1	1	315	316	8.92	1	1	4795.76	45.92	Regional	Delhi
2	1	403	404	4.58	0	0	1993.66	32.58	Regional	Singapore
3	1	465	466	18.33	2	1	10698.12	67.33	Australia NZ	Melbourne
4	1	477	478	18.60	3	2	10698.12	67.60	Australia NZ	Sydney
5	1	620	621	6.50	1	0	1998.23	36.50	Regional	Manila
6	1	656	629	11.67	3	2	11073.55	56.67	Regional	Seoul
7	2	315	316	8.92	1	1	4795.76	45.92	Regional	Delhi
8	2	403	404	4.58	0	0	1993.66	32.58	Regional	Singapore
9	2	465	466	18.33	2	1	10698.12	67.33	Australia NZ	Melbourne
10	2	477	478	18.60	3	2	10698.12	67.60	Australia NZ	Sydney
11	2	620	621	6.50	1	0	1998.23	36.50	Regional	Manila
12	2	656	629	11.67	3	2	11073.55	56.67	Regional	Seoul
13	3	315	316	8.92	1	1	4795.76	45.92	Regional	Delhi
14	3	403	404	4.58	0	0	1993.66	32.58	Regional	Singapore
15	3	465	466	18.33	2	1	10698.12	67.33	Australia NZ	Melbourne
16	3	477	478	18.60	3	2	10698.12	67.60	Australia NZ	Sydney
17	3	620	621	6.50	1	0	1998.23	36.50	Regional	Manila

18	3	656	629	11.67	3	2	11073.55	56.67	Regional	Seoul
19	4	315	316	8.92	1	1	4795.76	45.92	Regional	Delhi
20	4	403	404	4.58	0	0	1993.66	32.58	Regional	Singapore
21	4	465	466	18.33	2	1	10698.12	67.33	Australia NZ	Melbourne
22	4	477	478	18.60	3	2	10698.12	67.60	Australia NZ	Sydney
23	4	620	621	6.50	1	0	1998.23	36.50	Regional	Manila
24	4	656	629	11.67	3	2	11073.55	56.67	Regional	Seoul
25	5	315	316	8.92	1	1	4795.76	45.92	Regional	Delhi
26	5	403	404	4.58	0	0	1993.66	32.58	Regional	Singapore
27	5	465	466	18.33	2	1	10698.12	67.33	Australia NZ	Melbourne
28	5	477	478	18.60	3	2	10698.12	67.60	Australia NZ	Sydney
29	5	620	621	6.50	1	0	1998.23	36.50	Regional	Manila
30	5	656	629	11.67	3	2	11073.55	56.67	Regional	Seoul
31	6	315	316	8.92	1	1	4795.76	45.92	Regional	Delhi
32	6	403	404	4.58	0	0	1993.66	32.58	Regional	Singapore
33	6	465	466	18.33	2	1	10698.12	67.33	Australia NZ	Melbourne
34	6	477	478	18.60	3	2	10698.12	67.60	Australia NZ	Sydney
35	6	620	621	6.50	1	0	1998.23	36.50	Regional	Manila
36	6	656	629	11.67	3	2	11073.55	56.67	Regional	Seoul
37	7	315	316	8.92	1	1	4795.76	45.92	Regional	Delhi
38	7	403	404	4.58	0	0	1993.66	32.58	Regional	Singapore
39	7	465	466	18.33	2	1	10698.12	67.33	Australia NZ	Melbourne
40	7	477	478	18.60	3	2	10698.12	67.60	Australia NZ	Sydney
41	7	620	621	6.50	1	0	1998.23	36.50	Regional	Manila
42	7	656	629	11.67	3	2	11073.55	56.67	Regional	Seoul
43	8	315	316	8.92	1	1	4795.76	45.92	Regional	Delhi
44	8	403	404	4.58	0	0	1993.66	32.58	Regional	Singapore

45	8	465	466	18.33	2	1	10698.12	67.33	Australia NZ	Melbourne
46	8	477	478	18.60	3	2	10698.12	67.60	Australia NZ	Sydney
47	8	620	621	6.50	1	0	1998.23	36.50	Regional	Manila
48	8	656	629	11.67	3	2	11073.55	56.67	Regional	Seoul
49	9	315	316	8.92	1	1	4795.76	45.92	Regional	Delhi
50	9	403	404	4.58	0	0	1993.66	32.58	Regional	Singapore
51	9	465	466	18.33	2	1	10698.12	67.33	Australia NZ	Melbourne
52	9	477	478	18.60	3	2	10698.12	67.60	Australia NZ	Sydney
53	9	620	621	6.50	1	0	1998.23	36.50	Regional	Manila
54	9	656	629	11.67	3	2	11073.55	56.67	Regional	Seoul
55	10	315	316	8.92	1	1	4795.76	45.92	Regional	Delhi
56	10	403	404	4.58	0	0	1993.66	32.58	Regional	Singapore
57	10	465	466	18.33	2	1	10698.12	67.33	Australia NZ	Melbourne
58	10	477	478	18.60	3	2	10698.12	67.60	Australia NZ	Sydney
59	10	620	621	6.50	1	0	1998.23	36.50	Regional	Manila
60	10	656	629	11.67	3	2	11073.55	56.67	Regional	Seoul
61	11	315	316	8.92	1	1	4795.76	45.92	Regional	Delhi
62	11	403	404	4.58	0	0	1993.66	32.58	Regional	Singapore
63	11	465	466	18.33	2	1	10698.12	67.33	Australia NZ	Melbourne
64	11	477	478	18.60	3	2	10698.12	67.60	Australia NZ	Sydney
65	11	620	621	6.50	1	0	1998.23	36.50	Regional	Manila
66	11	656	629	11.67	3	2	11073.55	56.67	Regional	Seoul
67	12	315	316	8.92	1	1	4795.76	45.92	Regional	Delhi
68	12	403	404	4.58	0	0	1993.66	32.58	Regional	Singapore
69	12	465	466	18.33	2	1	10698.12	67.33	Australia NZ	Melbourne
70	12	477	478	18.60	3	2	10698.12	67.60	Australia NZ	Sydney
71	12	620	621	6.50	1	0	1998.23	36.50	Regional	Manila

72	12	656	629	11.67	3	2	11073.55	56.67	Regional	Seoul
73	13	315	316	8.92	1	1	4795.76	45.92	Regional	Delhi
74	13	403	404	4.58	0	0	1993.66	32.58	Regional	Singapore
75	13	465	466	18.33	2	1	10698.12	67.33	Australia NZ	Melbourne
76	13	477	478	18.60	3	2	10698.12	67.60	Australia NZ	Sydney
77	13	620	621	6.50	1	0	1998.23	36.50	Regional	Manila
78	13	656	629	11.67	3	2	11073.55	56.67	Regional	Seoul
79	14	315	316	8.92	1	1	4795.76	45.92	Regional	Delhi
80	14	403	404	4.58	0	0	1993.66	32.58	Regional	Singapore
81	14	465	466	18.33	2	1	10698.12	67.33	Australia NZ	Melbourne
82	14	477	478	18.60	3	2	10698.12	67.60	Australia NZ	Sydney
83	14	620	621	6.50	1	0	1998.23	36.50	Regional	Manila
84	14	656	629	11.67	3	2	11073.55	56.67	Regional	Seoul

Appendix B

Change pairing descending code

This section shows the change pairing descending technique in order to minimize workload and perdiem simultaneously with workload bound that applied to using with TA84M instance. In addition, code of change pairing directly, change pairing descending, and change pairing ascending are similarly.

```
#include "stdio.h"
#include "string.h"
#include "math.h"

struct crew_pair {
    int day;
    int operate_day;
    float fduty;
    float perdiem;
    float workload;
}; struct crew_pair pair[84] = {0}; // number of pair

struct crew_sch {
    int icrew;
    int jday;
    int remain;
    int operate_day;
    float fduty;
    float workload;
    float fduty_unit;
    float workload_unit;
    float perdiem;
    float perdiem_unit;
}; struct crew_sch impair[84] = {0}; // number of pair

//define task
int task_size = 84; // number of pair

//schedule
int crew = 30;
int day_in_week = 14;
int schedule[30][30][14];

//Constraint
float max_fduty = 68.00;

void init_schedule(){
    int ii,jj,kk;
    for(ii=0; ii<crew; ii++){
        for(jj=0; jj<day_in_week; jj++){
            schedule[0][ii][jj]=0;
        }
    }
}
```

```

void data_import(){
    FILE *fp1;
    int i = 0;

    if((fp1 = fopen("C:/Users/YURIWON/Documents/Visual Studio 2010/Projects/Thesis 2015
test 2/small mix.txt","r")) == NULL){
        printf("Cannot open");
        getch();

    }else{

        while((fscanf(fp1, "%d%d%f%f%f\n",&pair[i].day,&pair[i].operate_day,&pair[i].fduty,&pai
r[i].perdiem,&pair[i].workload)) != EOF){

            i++;

        } }
}

int main(){

    int ii,jj,kk,zz,cc,dd,mc,nd,xx,s_count;
    int icw = 0;
    int _id = 0;
    int remain = 0;
    int day = 0;
    int max_op = 0;
    float sum_fduty[30][30] = {0};
    float fduty_unit = 0;
    int remove_over = 0;
    float workload_unit = 0;
    float sum_workload[30][30] = {0};
    float temp_workload = 0;
    float avg_workload = 0;
    float sum_avg_workload = 0;
    float sum_perdiem[30][30] = {0};
    float perdiem_unit = 0;
    float temp_perdiem = 0;
    float avg_perdiem = 0;
    float sum_avg_perdiem = 0;
    int _going=0;
    int id_table = 0;
    int new_crew = 0;
    int new_day = 0;
    int new_remain = 0;
    int new_max_op = 0;
    int state = 1;
    int x_index = 0;
    int new_id = 0;
    float sd_workload[30] = {0};
    float sd_perdiem[30] = {0};
    int checkTF;
    float new_temp_workload;
    float new_temp_perdiem;
    float new_sum_avg_workload;
    float new_sum_avg_perdiem;
    int max_state=0;
    int icrew[84] = {0};
    int oo = 0;
    int pp =0;

```

```

float swap_temp_workload = 0;
float swap_temp_perdiem = 0;
float ftemp = 0;
int swap_temp = 0;
float max_temp_wl = 0;
float max_temp_pd = 0;
float stand_workload[84] = {0};
float stand_perdiem[84] = {0};
float stand_sd[30] = {0};
float real_con_wl = 0;
float real_con_pd = 0;
float max_sum_wl[30] = {0};
float min_sum_wl[30] = {0};
float test_temp = 0;
int iFlag = 1;

FILE *fp2, *fp3;

data_import();

init_schedule();

printf("\n\n ----- Construction Phase ----- \n\n");

/*-----Start Normalize-----*/

for(xx = 0;xx < task_size;xx++){
    if(pair[xx].workload > max_temp_wl){

        max_temp_wl = pair[xx].workload;

    }

    if(pair[xx].perdiem > max_temp_pd){
        max_temp_pd = pair[xx].perdiem;
    }
}

for(xx=0;xx<task_size;xx++){
    stand_workload[xx] = pair[xx].workload / max_temp_wl;
    stand_perdiem[xx] = pair[xx].perdiem / max_temp_pd;
}

/*-----End Normalize-----*/

while(_id < task_size){

    day = pair[_id].day - 1;
    remain = pair[_id].operate_day;
    max_op = day + (pair[_id].operate_day-1);
    fduty_unit = pair[_id].fduty / pair[_id].operate_day;
    workload_unit = stand_workload[_id] / pair[_id].operate_day;
    perdiem_unit = stand_perdiem[_id] / pair[_id].operate_day;

    for(dd = day; dd <= max_op; dd++){ // check empty
        if(schedule[0][icw][dd] == 0){ // Have available space?

            while(remain != 0){ // Book table

                sum_fduty[0][icw] += fduty_unit;
                sum_workload[0][icw] += workload_unit;
            }
        }
    }
}

```

```

sum_perdiem[0][icw] += perdiem_unit;

if(sum_fduty[0][icw] <= max_fduty){

    schedule[0][icw][day] = _id+1;
    remain--;
    day++;

}else{

remove_over = pair[_id].operate_day - remain;
sum_fduty[0][icw] -= (fduty_unit + (fduty_unit * remove_over));
sum_workload[0][icw] -= (workload_unit + (workload_unit * remove_over));
sum_perdiem[0][icw] -= (perdiem_unit + (perdiem_unit * remove_over));

for(mc=0;mc<remove_over;mc++){
    day--;
    remain++;
    schedule[0][icw][day] = 0;
}

icw++;
}

if(day == day_in_week){
    remain = 0;
}

} // 1 means bookable , End while loop

icw++;
_id++;
break;

}else{

icw++;
break;

}
} // End of check empty

if(icw == crew){ // check crew not over max_crew
icw = 0;
}

} // End of while pairing loop

for(ii=0;ii<crew;ii++){
    for(jj=0;jj<day_in_week;jj++){
        printf("%d ", schedule[0][ii][jj]);
    }

    printf("\t FD: %.2f",sum_fduty[0][ii]);
    printf("\t WL: %.2f",sum_workload[0][ii]);
    printf("\n");

}

// Find SD
for(kk=0;kk<crew;kk++){

```

```

temp_workload += sum_workload[0][kk];
temp_perdiem += sum_perdiem[0][kk];
sum_fduty[0][kk] = sum_fduty[0][kk]; // 0 is default state 0
sum_workload[0][kk] = sum_workload[0][kk]; // 0 is default state 0
}

avg_workload = temp_workload / crew;
avg_perdiem = temp_perdiem / crew;

for (zz=0;zz<crew;zz++){

sum_avg_workload += pow((sum_workload[0][zz] - avg_workload),2);
sum_avg_perdiem += pow((sum_perdiem[0][zz] - avg_perdiem),2);
}

sd_workload[0] = sqrt(sum_avg_workload/(crew-1));
sd_perdiem[0] = sqrt(sum_avg_perdiem/(crew-1));

stand_sd[0] = sd_workload[0] + sd_perdiem[0];

printf("\n Standard Deviation of workload is %.4f \n", sd_workload[0]);
printf("\n Standard Deviation of perdiem is %.4f \n", sd_perdiem[0]);
printf("\n Sum Standard Deviation %.4f \n", stand_sd[0]);

//=====printf real SD=====
real_con_wl = sd_workload[0] * max_temp_wl;
real_con_pd = sd_perdiem[0] * max_temp_pd;

printf("\n ----- Real sd of workload is %.4f ----- \n",real_con_wl);
printf("\n ----- Real sd of perdiem is %.4f ----- \n",real_con_pd);

printf("\n WL mean: %.4f \n", avg_workload * max_temp_wl);
printf("\n PD mean: %.4f \n", avg_perdiem * max_temp_pd);

if((fp2 = fopen("C:/Users/YURIWON/Documents/Visual Studio 2010/Projects/Test PaRn/wl pd
improve 2/sort wl pd/result con small mix.xls", "w"))==NULL){

printf("\n Cannot create file to write \n");
getch();
}

for(ii=0;ii<crew;ii++){
for(jj=0;jj<day_in_week;jj++){
fprintf(fp2,"%d ", schedule[0][ii][jj]);
}

fprintf(fp2,"\t fduty is %.2f",sum_fduty[0][ii]);
fprintf(fp2,"\t workload is %.2f",sum_workload[0][ii]);
fprintf(fp2,"\t perdiem is %.2f",sum_perdiem[0][ii]);
fprintf(fp2,"\n");

}

fprintf(fp2,"\n\n SD of sd_workload[0] is %.4f \n",sd_workload[0]);
fprintf(fp2,"\n\n SD of sd_perdiem[0] is %.4f \n",sd_perdiem[0]);

fclose(fp2);

printf("\n\n ----- Improvement Phase ----- \n\n");

```

```

for(ii=0;ii<crew;ii++){
    for(jj=0;jj<day_in_week;jj++){

        if(schedule[0][ii][jj] != 0){
            id_table = schedule[0][ii][jj];

            impair[id_table-1].remain++;

        }
    } // End table loop
}

for(zz=0;zz<task_size;zz++){
    impair[zz].jday = pair[zz].day;
    impair[zz].fduty = pair[zz].fduty;
    impair[zz].workload = stand_workload[zz];
    impair[zz].perdiem = stand_perdiem[zz];
    impair[zz].operate_day = pair[zz].operate_day;
    impair[zz].fduty_unit = impair[zz].fduty / impair[zz].operate_day;
    impair[zz].workload_unit = stand_workload[zz] / impair[zz].operate_day;
    impair[zz].perdiem_unit = stand_perdiem[zz] / impair[zz].operate_day;
}

for(ii=0;ii<crew;ii++){
    for(jj=0;jj<day_in_week;jj++){
        schedule[1][ii][jj] = schedule[0][ii][jj];
    }
}

for(ii=0;ii<crew;ii++){
    sum_fduty[1][ii] = sum_fduty[0][ii];
    sum_workload[1][ii] = sum_workload[0][ii];
    sum_perdiem[1][ii] = sum_perdiem[0][ii];
}

stand_sd[1] = stand_sd[0];

for(_going = 0; _going < 100; _going++){

    for(new_id = 0; new_id < task_size; new_id++){

        if(_going == 0){ // iteration

// Swap
for(ii=(crew-1);ii>0;ii--){ // sort by workload from least to greater
    for(jj=1;jj<=ii;jj++){
        if(sum_workload[state][jj-1] < sum_workload[state][jj]){

            swap_temp_workload = sum_workload[state][jj-1];
            sum_workload[state][jj-1] = sum_workload[state][jj];
            sum_workload[state][jj] = swap_temp_workload;
            swap_temp_perdiem = sum_perdiem[state][jj-1];
            sum_perdiem[state][jj-1] = sum_perdiem[state][jj];
            sum_perdiem[state][jj] = swap_temp_perdiem;

            ftemp = sum_fduty[state][jj-1];
            sum_fduty[state][jj-1] = sum_fduty[state][jj];
            sum_fduty[state][jj] = ftemp;

        }
    }
}

for(dd=0;dd<day_in_week;dd++){ // copy new sorted schedule, 1- 30

```

```

swap_temp = schedule[state][jj-1][dd];

schedule[state][jj-1][dd] = schedule[state][jj][dd];
schedule[state][jj][dd] = swap_temp;
    }
    }
}

}else{ // End if going == 0 & start if going == 1 // other iteration

for(ii=(crew-1);ii>0;ii--){ // sort by workload from least to greater
    for(jj=1;jj<=ii;jj++){
        if(sum_perdiem[state][jj-1] < sum_perdiem[state][jj]){

            swap_temp_workload = sum_workload[state][jj-1];
            sum_workload[state][jj-1] = sum_workload[state][jj];
            sum_workload[state][jj] = swap_temp_workload;
            swap_temp_perdiem = sum_perdiem[state][jj-1];
            sum_perdiem[state][jj-1] = sum_perdiem[state][jj];
            sum_perdiem[state][jj] = swap_temp_perdiem;

            ftemp = sum_fduty[state][jj-1];
            sum_fduty[state][jj-1] = sum_fduty[state][jj];
            sum_fduty[state][jj] = ftemp;

for(dd=0;dd<day_in_week;dd++){ // copy new sorted schedule, 1- 30
            swap_temp = schedule[state][jj-1][dd];
            schedule[state][jj-1][dd] = schedule[state][jj][dd];
            schedule[state][jj][dd] = swap_temp;
                }
            }
        }
    } // End else going != 0

    for(ii=0;ii<crew;ii++){
        for(jj=0;jj<day_in_week;jj++){

            if(schedule[1][ii][jj] != 0){
                id_table = schedule[1][ii][jj];

                icrew[id_table-1] = ii;
            }
        }
    } // End table loop

    new_crew = icrew[new_id] + 1;

    if(new_crew == crew){
        new_crew = 0;
    }

    new_day = impair[new_id].jday - 1;
    new_max_op = new_day + (impair[new_id].remain - 1);
    new_remain = impair[new_id].remain;

```

```

while(new_crew != icrew[new_id]){

    checkTF = 1;

    for(nd = new_day; nd <= new_max_op; nd++){

        if(schedule[state][new_crew][nd] == 0){
            checkTF *= 1;
        }else{
            checkTF *= 0;
        }
    } // End for nd loop

    if(!_going != 0){

test_temp = (impair[new_id].workload_unit * impair[new_id].remain) +
sum_workload[state][new_crew];

if(test_temp >= min_sum_wl[state] && test_temp <= max_sum_wl[state]){

            iFlag = 1;

        }else{

            iFlag = 0;

        }

        } // End of limited bound

if(checkTF == 1 && ((impair[new_id].fduty_unit * impair[new_id].remain) +
sum_fduty[state][new_crew]) <= max_fduty && iFlag == 1){

            state++;

            // Copy code
            for(ii=0;ii<crew;ii++){
                for(jj=0;jj<day_in_week;jj++){
                    schedule[state][ii][jj] = schedule[1][ii][jj];
                }
            }

for(ii=0;ii<crew;ii++){
    sum_fduty[state][ii] = sum_fduty[1][ii];
    sum_workload[state][ii] = sum_workload[1][ii];
    sum_perdiem[state][ii] = sum_perdiem[1][ii];
} // End of copy code

// Remove old pair
for(cc = 0; cc < crew; cc++){
    for(dd = 0; dd < day_in_week; dd++){
        if(schedule[state][cc][dd] == new_id+1){

            schedule[state][cc][dd] = 0;
            sum_fduty[state][cc] -= impair[new_id].fduty_unit;

            sum_workload[state][cc] -= impair[new_id].workload_unit;

            sum_perdiem[state][cc] -= impair[new_id].perdiem_unit;
        }
    } // End for dd loop
}

```



```

    } // End for cc loop

// Booking new pair
for(xx = new_day; xx <= new_max_op; xx++){
    schedule[state][new_crew][xx] = new_id + 1;
    sum_fduty[state][new_crew] += impair[new_id].fduty_unit;
    sum_workload[state][new_crew] += impair[new_id].workload_unit;
    sum_perdiem[state][new_crew] += impair[new_id].perdiem_unit;
}

// Cal SD
new_temp_workload = 0;
new_sum_avg_workload = 0;
new_temp_perdiem = 0;
new_sum_avg_perdiem = 0;

for(kk=0;kk<crew;kk++){
    new_temp_workload += sum_workload[state][kk];
    new_temp_perdiem += sum_perdiem[state][kk];
}

avg_workload = new_temp_workload / crew;
avg_perdiem = new_temp_perdiem / crew;

for (zz=0;zz<crew;zz++){
    new_sum_avg_workload += pow((sum_workload[state][zz] - avg_workload),2);
    new_sum_avg_perdiem += pow((sum_perdiem[state][zz] - avg_perdiem),2);
}

sd_workload[state] = sqrt(new_sum_avg_workload/(crew-1));
sd_perdiem[state] = sqrt(new_sum_avg_perdiem/(crew-1));

stand_sd[state] = sd_workload[state] + sd_perdiem[state];

new_crew++;
max_state++;

} else { // else of check not over 34 hrs

    new_crew++;

}

if(new_crew == crew){ // Use to continue crew loop
    new_crew = 0;
}

} // End while crew loop

// compare state
for(s_count = 0; s_count < max_state; s_count++){
    if(stand_sd[1] >= stand_sd[s_count+2]){
        sd_workload[1] = sd_workload[s_count+2];
        sd_perdiem[1] = sd_perdiem[s_count+2];

        stand_sd[1] = sd_workload[1] + sd_perdiem[1];

        for(ii=0;ii<crew;ii++){

            for(jj=0;jj<day_in_week;jj++){

```

```

schedule[1][ii][jj] = schedule[s_count+2][ii][jj];

        }
    }

for(ii=0;ii<crew;ii++){
    sum_fduty[1][ii] = sum_fduty[s_count+2][ii];
    sum_workload[1][ii] = sum_workload[s_count+2][ii];
    sum_perdiem[1][ii] = sum_perdiem[s_count+2][ii];
    }

} // End of for(s_count) , compare state

    max_state = 0;
    state = 1;

} // End for new_id loop

// find min & max workload
if(_going == 0){
    min_sum_wl[state] = 888;

    for(cc = 0; cc < crew; cc++){

        if(sum_workload[state][cc] > max_sum_wl[state]){
            max_sum_wl[state] = sum_workload[state][cc];
        }

        if(sum_workload[state][cc] < min_sum_wl[state]){
            min_sum_wl[state] = sum_workload[state][cc];
        }

    }

} // End find max min, do only first iteration

} // End _going loop

printf("\n !!!!!!!!!!!!!!! Pass End new_id loop !!!!!!!!!!!!!!! \n");

// Print improvement table
printf("\n\n");
for(ii=0;ii<crew;ii++){
    for(jj=0;jj<day_in_week;jj++){
        printf("%d ",schedule[1][ii][jj]);
    }
    printf("\t FD: %.2f",sum_fduty[1][ii]);
    printf("\t WL: %.2f",sum_workload[1][ii]);
    printf("\t PD: %.2f",sum_perdiem[1][ii]);
    printf("\n");
}

printf("\n #### Max WL %.2f Min WL %.2f #####\n",max_sum_wl[state],min_sum_wl[state]);

    printf("\n\n SD of workload is %.4f \n",sd_workload[1]);
    printf("\n\n SD of perdiem is %.4f \n",sd_perdiem[1]);
    printf("\n\n SD of Sum Stand is %.4f \n",stand_sd[1]);

    real_con_wl = sd_workload[1] * max_temp_wl;

```

```

    real_con_pd = sd_perdiem[1] * max_temp_pd;

    printf("\n ----- Real sd of workload is %.4f ----- \n",real_con_wl);
    printf("\n ----- Real sd of perdiem is %.4f ----- \n",real_con_pd);

    printf("\n ----- Normalize sd of workload is %.4f ----- \n",sd_workload[1]);
    printf("\n ----- Normalize sd of perdiem is %.4f ----- \n",sd_perdiem[1]);
    printf("\n ----- Total normalize SD %.4f ----- \n",stand_sd[1]);

    printf("\n Max workload: %.2f\n",max_temp_wl);
    printf("\n Max perdiem: %.2f\n",max_temp_pd);

    printf("\n WL mean: %.4f \n", avg_workload * max_temp_wl);
    printf("\n PD mean: %.4f \n", avg_perdiem * max_temp_pd);

    printf("\n WL mean: %.4f \n", avg_workload);
    printf("\n PD mean: %.4f \n", avg_perdiem);

// write to file
if((fp3 = fopen("C:/Users/YURIWON/Documents/Visual Studio 2010/Projects/Test PaRn/wl pd
improve 2/sort wl pd/result improve small mix.xls", "w"))==NULL){

    printf("\n Cannot create file to write \n");
    getch();
}

for(ii=0;ii<crew;ii++){
    for(jj=0;jj<day_in_week;jj++){
        fprintf(fp3,"t %d", schedule[0][ii][jj]);
    }

    fprintf(fp3,"t %.2f",sum_f-duty[0][ii]);
    fprintf(fp3,"t %.2f",sum_workload[0][ii] * max_temp_wl);
    fprintf(fp3,"t %.2f",sum_perdiem[0][ii] * max_temp_pd);
    fprintf(fp3,"t %.2f",sum_workload[0][ii]);
    fprintf(fp3,"t %.2f",sum_perdiem[0][ii]);
    fprintf(fp3,"n");
}

fprintf(fp3,"n\n SD of sd_workload[1] is %.4f \n",sd_workload[1] * max_temp_wl);
fprintf(fp3,"n\n SD of sd_perdiem[1] is %.4f \n",sd_perdiem[1] * max_temp_pd);

fclose(fp3);

getch();
return 0;
} // End main

```

Appendix C

High workload and perdiem distribution code

This section shows code of the high workload and perdiem distribution technique in order to minimize workload and perdiem simultaneously with workload bound. The code use to modify for TA84M test instance.

```
#include "stdio.h"
#include "string.h"
#include "math.h"

struct crew_pair {
    int day;
    int operate_day;
    float fduty;
    float perdiem;
    float workload;
}; struct crew_pair pair[84] = {0}; // number of pair

//define task
int task_size = 84; // number of pair

//schedule
int crew = 30;
int day_in_week = 14;
int schedule[30][30][14];
int con_table[30][14];

//Constraint
float max_fduty = 34.00;
float new_wl_unit[30][84] = {0};
float new_pd_unit[30][30] = {0};
float new_fd_unit[30][84] = {0};
float new_tt_unit[30][84] = {0};
int new_day = 0;
int new_remain[30][84] = {0}; // no. of pair

void init_schedule(){
    int ii,jj,kk;
    for(ii=0; ii<crew; ii++){
        for(jj=0; jj<day_in_week; jj++){
            schedule[0][ii][jj]=0;
        }
    }
}

void data_import(){
    FILE *fp1;
    int i = 0;

    if((fp1 = fopen("C:/Users/YURIWON/Documents/Visual Studio 2010/Projects/Thesis 2015 test
2/small mix.txt", "r")) == NULL){
        printf("Cannot open");
        getch();
    }else{
```

```

while((fscanf(fp1, "%d%d%f%f%f\n", &pair[i].day, &pair[i].operate_day, &pair[i].fduty, &pair[i].perdi
em, &pair[i].workload)) != EOF){

    i++;

    }

    }

fclose(fp1);

}

```

```

int find_max_wl(int ii, int state){ // ii is number of crew

```

```

int x_index = 0;
float each_pair_wl[84] = {0};
float temp = 0;
int res_index = 0;
int jj, check_max;

for(jj = 0; jj < day_in_week; jj++){
    if(schedule[1][ii][jj] != 0){
        x_index = schedule[1][ii][jj];
        each_pair_wl[x_index-1] += new_wl_unit[state][x_index-1];
    }
}

for(check_max = 0; check_max < task_size; check_max++){
    if(each_pair_wl[check_max-1] > temp){
        temp = each_pair_wl[check_max-1];
        res_index = check_max;
    }
}

return res_index;

```

```

} // End find_max_wl function

```

```

int find_max_pd(int ii, int state){ // ii is number of crew

```

```

int x_index = 0;
float each_pair_wl[84] = {0};
float temp = 0;
int res_index = 0;
int jj, check_max;

for(jj = 0; jj < day_in_week; jj++){
    if(schedule[1][ii][jj] != 0){
        x_index = schedule[1][ii][jj];
        each_pair_wl[x_index-1] += new_pd_unit[state][x_index-1];
    }
}

for(check_max = 0; check_max < task_size; check_max++){
    if(each_pair_wl[check_max-1] > temp){
        temp = each_pair_wl[check_max-1];
        res_index = check_max;
    }
}

```

```

    }

    return res_index;

} // End find_max_wl function

int main(){

int ii,jj,kk,zz,cc,dd,mc,nd,ismall,ibig,xx;
float icw = 0;
int _id = 0;
int remain = 0;
int day = 0;
int max_op = 0;
float sum_fduty[30][30] = {0};
float fduty_unit = 0;
int remove_over = 0;
float workload_unit = 0;
float sum_workload[30][30] = {0};
float temp = 0;
float avg = 0;
float sum_avg = 0;
float result = 0;
int swap_temp;
int _going=0;
int id_table = 0;
int new_crew = 0;
int new_max_op = 0;
int state = 0;
int x_index = 0;
int id_index = 0;
int new_id = 0;
float sd[30] = {0};
float new_workload[30][30] = {0};
int checkTF;
float new_wl_temp = 0;
float new_pd_temp = 0;
int mean_crew = 0;
int max_state = 0;
int icrew[84] = {0};
int oo = 0;
int pp =0;
float sd_temp = 0;
float ftemp = 0;
int small_id = 0;
int big_id = 0;
int max_sd = 0;
float sub_sd = 0;
int s_count = 1;
int max_count = 0;
float temp_workload = 0;
float temp_perdiem = 0;
float new_sum_avg_workload;
float new_sum_avg_perdiem;
float sd_workload[30] = {0};
float sd_perdiem[30] = {0};
float perdiem_unit = 0;
float sum_perdiem[30][30] = {0};
float sum_avg_workload = 0;
float avg_perdiem = 0;

```

```

float sum_avg_perdiem = 0;
float avg_workload = 0;
float avg_wl = 0;
float avg_pd = 0;
float sum_avg_wl = 0;
float sum_avg_pd = 0;
float max_temp_wl = 0;
float max_temp_pd = 0;
float stand_workload[84] = {0};
float stand_perdiem[84] = {0};
float stand_sd[30] = {0};
float real_con_wl = 0;
float real_con_pd = 0;
float total_stand[30][30] = {0};
float new_tt_temp = 0;
float max_sum_wl[30] = {0};
float min_sum_wl[30] = {0};
float test_temp = 0;
int iFlag = 1;

FILE *fp2, *fp3;

data_import();

init_schedule();

printf("\n\n ----- Construction Phase ----- \n\n");

/*-----Start Normalize-----*/

for(xx = 0;xx < task_size;xx++){
    if(pair[xx].workload > max_temp_wl){
        max_temp_wl = pair[xx].workload;
    }

    if(pair[xx].perdiem > max_temp_pd){
        max_temp_pd = pair[xx].perdiem;
    }
}

for(xx=0;xx<task_size;xx++){

    stand_workload[xx] = pair[xx].workload / max_temp_wl;    // represent pair[ ].crew
    stand_perdiem[xx] = pair[xx].perdiem / max_temp_pd;

}

/*-----End Normalize-----*/

while(_id < task_size){

    day = pair[_id].day - 1;
    remain = pair[_id].operate_day;
    max_op = day + (pair[_id].operate_day-1);
    fduty_unit = pair[_id].fduty / pair[_id].operate_day;
    workload_unit = stand_workload[_id] / pair[_id].operate_day;
    perdiem_unit = stand_perdiem[_id] / pair[_id].operate_day;

    for(dd = day; dd <= max_op; dd++){ // check empty

```

```

if(schedule[0][icw][dd] == 0){ // Have available space?

    while(remain != 0){ // Book table

        sum_fduty[0][icw] += fduty_unit;
        sum_workload[0][icw] += workload_unit;
        sum_perdiem[0][icw] += perdiem_unit;

        if(sum_fduty[0][icw] <= max_fduty * 2){

            schedule[0][icw][day] = _id+1;
            remain--;
            day++;

        }else{

remove_over = pair[_id].operate_day - remain;
sum_fduty[0][icw] -= (fduty_unit + (fduty_unit * remove_over));
sum_workload[0][icw] -= (workload_unit + (workload_unit * remove_over));
sum_perdiem[0][icw] -= (perdiem_unit + (perdiem_unit * remove_over));

            for(mc=0;mc<remove_over;mc++){
                day--;
                remain++;
                schedule[0][icw][day] = 0;
            }

            icw++;

        }

        if(day == day_in_week){
            remain = 0;
        }

    } // 1 means bookable , End while loop

        icw++;
        _id++;
        break;

    }else{

        icw++;
        break;

    }

} // End of check empty

if(icw == crew){ // check crew not over max_crew
    icw = 0;
}

} // End of while pairing loop

for(ii=0;ii<crew;ii++){
    for(jj=0;jj<day_in_week;jj++){

```



```

        printf("%d ", schedule[0][ii][jj]);
    }

    printf("\t WL: %.2f",sum_workload[0][ii]);
    printf("\t PD: %.2f",sum_perdiem[0][ii]);
    printf("\n");
}

for(kk=0;kk<crew;kk++){

    temp_workload += sum_workload[0][kk];
    temp_perdiem += sum_perdiem[0][kk];

}

    avg_workload = temp_workload / crew;
    avg_perdiem = temp_perdiem / crew;

for (zz=0;zz<crew;zz++){

    sum_avg_workload += pow((sum_workload[0][zz] - avg_workload),2);
    sum_avg_perdiem += pow((sum_perdiem[0][zz] - avg_perdiem),2);

}

    sd_workload[0] = sqrt(sum_avg_workload/(crew-1));
    sd_perdiem[0] = sqrt(sum_avg_perdiem/(crew-1));

    stand_sd[0] = sd_workload[0] + sd_perdiem[0];

    printf("\n Standard Deviation of workload is %.4f \n", sd_workload[0]);
    printf("\n Standard Deviation of perdiem is %.4f \n", sd_perdiem[0]);
    printf("\n Sum Standard Deviation %.4f \n", stand_sd[0]);

    //=====printf real SD=====
    real_con_wl = sd_workload[0] * max_temp_wl;
    real_con_pd = sd_perdiem[0] * max_temp_pd;

    printf("\n ----- Real sd of workload is %.4f ----- \n",real_con_wl);
    printf("\n ----- Real sd of perdiem is %.4f ----- \n",real_con_pd);

if((fp2 = fopen("C:/Users/YURIWON/Documents/Visual Studio 2010/Projects/Test PaRn/wl pd
improve 4/sort wl pd/result con small mix.xls", "w"))==NULL){

    printf("\n Cannot create file to write \n");
    getch();
}

for(ii=0;ii<crew;ii++){
    for(jj=0;jj<day_in_week;jj++){
        fprintf(fp2,"%d ", schedule[0][ii][jj]);

    }

    fprintf(fp2,"\t %.2f",sum_fduty[0][ii]);
    fprintf(fp2,"\t %.2f",sum_workload[0][ii]* max_temp_wl);
    fprintf(fp2,"\t %.2f",sum_perdiem[0][ii]* max_temp_pd);
    fprintf(fp2,"\n");

}

fprintf(fp2,"\n\n SD of sd_workload[0] is %.4f \n",sd_workload[0]* max_temp_wl);

```

```

fprintf(fp2, "\n\n SD of sd_perdiem[0] is %.4f \n", sd_perdiem[0]* max_temp_pd);

fclose(fp2);

printf("\n\n ----- Improvement Phase ----- \n\n");

state = 1;

for(ii=0;ii<crew;ii++){ // copy table
    for(jj=0;jj<day_in_week;jj++){
        schedule[state][ii][jj] = schedule[0][ii][jj];
    }
}

for(ii=0;ii<crew;ii++){ // copy workload
    sum_fduty[state][ii] = sum_fduty[0][ii];
    sum_workload[state][ii] = sum_workload[0][ii];
    sum_perdiem[state][ii] = sum_perdiem[0][ii];
}

sd_workload[state] = sd_workload[0]; // copy sd[0] to sd[state]
sd_perdiem[state] = sd_perdiem[0];

stand_sd[state] = sd_workload[state] + sd_perdiem[state];

for(_going = 0; _going < 100; _going++){ // iteration

    if(_going == 0){

// swap table
        for(ii=(crew-1);ii>0;ii--){
            for(jj=1;jj<=ii;jj++){
                if(sum_workload[state][jj-1] < sum_workload[state][jj]){

                    new_wl_temp = sum_workload[state][jj-1];
                    sum_workload[state][jj-1] = sum_workload[state][jj];
                    sum_workload[state][jj] = new_wl_temp;

                    new_pd_temp = sum_perdiem[state][jj-1];
                    sum_perdiem[state][jj-1] = sum_perdiem[state][jj];
                    sum_perdiem[state][jj] = new_pd_temp;

                    ftemp = sum_fduty[state][jj-1];
                    sum_fduty[state][jj-1] = sum_fduty[state][jj];
                    sum_fduty[state][jj] = ftemp;

                    for(dd=0;dd<day_in_week;dd++){
                        swap_temp = schedule[state][jj-1][dd];
                        schedule[state][jj-1][dd] = schedule[state][jj][dd];
                        schedule[state][jj][dd] = swap_temp;
                    }
                }
            }
        } // End swap

    } else{
        for(ii=(crew-1);ii>0;ii--){
            for(jj=1;jj<=ii;jj++){
                if(sum_perdiem[state][jj-1] < sum_perdiem[state][jj]){

                    new_wl_temp = sum_workload[state][jj-1];

```

```

sum_workload[state][jj-1] = sum_workload[state][jj];
sum_workload[state][jj] = new_wl_temp;
new_pd_temp = sum_perdiem[state][jj-1];
sum_perdiem[state][jj-1] = sum_perdiem[state][jj];
sum_perdiem[state][jj] = new_pd_temp;

ftemp = sum_fduty[state][jj-1];
sum_fduty[state][jj-1] = sum_fduty[state][jj];
sum_fduty[state][jj] = ftemp;

for(dd=0;dd<day_in_week;dd++){
    swap_temp = schedule[state][jj-1][dd];
    schedule[state][jj-1][dd] = schedule[state][jj][dd];
    schedule[state][jj][dd] = swap_temp;
}
}
} // End swap
} // End swap wl & pd

for(xx = 0; xx < task_size; xx++){
    new_remain[state][xx] = 0;
}

for(ii = 0; ii < crew; ii++){
    for(jj = 0; jj < day_in_week; jj++){

        if(schedule[state][ii][jj] != 0){
            id_index = schedule[state][ii][jj];
            new_remain[state][id_index-1]++;
        }
    }
}

for(cc = 0; cc < task_size; cc++){

    new_wl_unit[state][cc] = stand_workload[cc] / pair[cc].operate_day;
    new_pd_unit[state][cc] = stand_perdiem[cc] / pair[cc].operate_day;
    new_fd_unit[state][cc] = pair[cc].fduty / pair[cc].operate_day;
}

for(ibig = 0; ibig < (crew/2) - 1; ibig++){

    if(_going == 0){
        new_id = find_max_wl(ibig,state) - 1 ;

    }else{
        new_id = find_max_pd(ibig,state) - 1 ;
    }
    new_day = pair[new_id].day - 1;
    new_max_op = new_day + (new_remain[state][new_id] - 1);

for(ismall = crew-1; ismall > (crew/2); ismall--){

    checkTF = 1;

    for(nd = new_day; nd <= new_max_op; nd++){

```

```

        if(schedule[state][ismall][nd] == 0){
            checkTF *= 1;
        }else{
            checkTF *= 0;
        }
    }

if(_going != 0){

    test_temp = (new_wl_unit[state][new_id] *new_remain[state][new_id]) +
    sum_workload[state][ismall]; // test_temp should be array?

    if(test_temp >= min_sum_wl[state] && test_temp <= max_sum_wl[state]){
        iFlag = 1;
    }else{
        iFlag = 0;
    }

    } // End of limited bound

if(checkTF == 1 && ((new_fd_unit[state][new_id] * new_remain[state][new_id]) +
sum_fduty[state][ismall]) < max_fduty * 2 && iFlag == 1){ // Booking

    state++;
    max_state++;

    // copy table from state 1
    for(ii=0;ii<crew;ii++){
        for(jj=0;jj<day_in_week;jj++){

            schedule[state][ii][jj] = schedule[1][ii][jj];
        }
    }

    // copy fduty workload perdiem
    for(ii=0;ii<crew;ii++){
        sum_fduty[state][ii] = sum_fduty[1][ii];
        sum_workload[state][ii] = sum_workload[1][ii];
        sum_perdiem[state][ii] = sum_perdiem[1][ii];
    }

    // find new remain
    for(xx = 0; xx < task_size; xx++){
        new_remain[state][xx] = 0;
    }

    for(ii = 0; ii < crew; ii++){
        for(jj = 0; jj < day_in_week; jj++){
            if(schedule[state][ii][jj] != 0){
                x_index = schedule[state][ii][jj];
                new_remain[state][x_index-1]++;
            }
        }
    }
}

if(_going != 0){
test_temp = (new_wl_unit[state][new_id] *new_remain[state][new_id]) +
sum_workload[state][ismall]; // test_temp should be array?
if(test_temp >= min_sum_wl[state] && test_temp <= max_sum_wl[state]){

```

```

        iFlag = 1;
    }else{
        iFlag = 0;
    }
} // End of limited bound

if(iFlag == 1){
    // declare value per unit of fduty workload and perdiem
    for(cc = 0; cc < task_size; cc++){

        new_wl_unit[state][cc] = stand_workload[cc] / pair[cc].operate_day;
        new_pd_unit[state][cc] = stand_perdiem[cc] / pair[cc].operate_day;
        new_fd_unit[state][cc] = pair[cc].fduty / pair[cc].operate_day;

    }

// remove old pair
for(xx = new_day; xx <= new_max_op; xx++){

    schedule[state][ibig][xx] = 0;
    sum_fduty[state][ibig] -= new_fd_unit[state][new_id];
    sum_workload[state][ibig] -= new_wl_unit[state][new_id];
    sum_perdiem[state][ibig] -= new_pd_unit[state][new_id];
}

} // End of inside if iFlag == 1

// assign new pair
for(zz = new_day; zz <= new_max_op; zz++){

    schedule[state][ismall][zz] = new_id + 1;
    sum_fduty[state][ismall] += new_fd_unit[state][new_id];
    sum_workload[state][ismall] += new_wl_unit[state][new_id];
    sum_perdiem[state][ismall] += new_pd_unit[state][new_id];
}

// Cal SD
temp_workload = 0;
temp_perdiem = 0;

for(kk=0;kk<crew;kk++){
    temp_workload += sum_workload[state][kk];
    temp_perdiem += sum_perdiem[state][kk];
}

avg_wl = 0;
avg_pd = 0;
sum_avg_wl = 0;
sum_avg_pd = 0;
avg_wl = temp_workload / crew;
avg_pd = temp_perdiem / crew;

for (zz=0;zz<crew;zz++){
    sum_avg_wl += pow((sum_workload[state][zz] - avg_wl),2);
    sum_avg_pd += pow((sum_perdiem[state][zz] - avg_pd),2);
}

sd_workload[state] = sqrt(sum_avg_wl/(crew-1));
sd_perdiem[state] = sqrt(sum_avg_pd/(crew-1));

```

```

        stand_sd[state] = sd_workload[state] + sd_perdiem[state];

    } // End of check available

} // End of ismall loop

// Compare function
for(s_count = 0; s_count < max_state; s_count++){

    if(stand_sd[1] >= stand_sd[s_count+2]){

        sd_workload[1] = sd_workload[s_count+2];
        sd_perdiem[1] = sd_perdiem[s_count+2];
        stand_sd[1] = stand_sd[s_count+2];

    }

    for(ii=0;ii<crew;ii++){

        for(jj=0;jj<day_in_week;jj++){

            schedule[1][ii][jj] = schedule[s_count+2][ii][jj];

        }

    }

    for(ii=0;ii<crew;ii++){

        sum_fduty[1][ii] = sum_fduty[s_count+2][ii];
        sum_workload[1][ii] = sum_workload[s_count+2][ii];
        sum_perdiem[1][ii] = sum_perdiem[s_count+2][ii];

    }

}

state = 1;
max_state = 0;

} // End of ibig loop

// find min & max workload
if(_going == 0){
    min_sum_wl[state] = 888;

    for(cc = 0; cc < crew; cc++){

        if(sum_workload[state][cc] > max_sum_wl[state]){
            max_sum_wl[state] = sum_workload[state][cc];
        }
        if(sum_workload[state][cc] < min_sum_wl[state]){
            min_sum_wl[state] = sum_workload[state][cc];
        }

    }

} // End find max min, do only first iteration
} // End _going Iteration

printf("\n\n >>>>> Improvement <<<<<< \n");
for(ii=0;ii<crew;ii++){

    for(jj=0;jj<day_in_week;jj++){
        printf("%d ",schedule[state][ii][jj]);
    }
}

```

```

        printf("\t FD: %.2f",sum_fduty[state][ii]);
        printf("\t WL: %.2f",sum_workload[state][ii]);
        printf("\t PD: %.2f",sum_perdiem[state][ii]);
        printf("\n");
    }

printf("\n #### Max WL %.2f Min WL %.2f #####\n",max_sum_wl[state],min_sum_wl[state]);

printf("\n ---- SD of workload is %.4f ---- \n",sd_workload[1]);
printf("\n ---- SD of perdiem is %.4f ---- \n",sd_perdiem[1]);
printf("\n ---- Stand SD is %.4f ---- \n",stand_sd[1]);

real_con_wl = sd_workload[1] * max_temp_wl;
real_con_pd = sd_perdiem[1] * max_temp_pd;

printf("\n ---- Real sd of workload is %.4f ---- \n",real_con_wl);
printf("\n ---- Real sd of perdiem is %.4f ---- \n",real_con_pd);

// write to file
if((fp3 = fopen("C:/Users/YURIWON/Documents/Visual Studio 2010/Projects/Test PaRn/wl pd
improve 4/sort wl pd/result improve small mix.xls", "w"))==NULL){
    printf("\n Cannot create file to write \n");
    getch();
}

for(ii=0;ii<crew;ii++){
    for(jj=0;jj<day_in_week;jj++){
        fprintf(fp3,"%d ", schedule[state][ii][jj]);
    }

    fprintf(fp3,"\t %.2f",sum_fduty[state][ii]);
    fprintf(fp3,"\t %.2f",sum_workload[state][ii] * max_temp_wl);
    fprintf(fp3,"\t %.2f",sum_perdiem[state][ii] * max_temp_pd);
    fprintf(fp3,"\n");
}

fprintf(fp3,"\n\n SD workload is %.4f \n",sd_workload[1] * max_temp_wl);
fprintf(fp3,"\n\n SD perdiem is %.4f \n",sd_perdiem[1] * max_temp_pd);

fclose(fp3);

getch();
return 0;

} // End main

```