การจัดสรรพนักงานแบบหลายวัตถุประสงค์ในสายการประกอบรูปตัวยู

นายรณชัย ศิโรเวฐนุกูล

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรดุษฎีบัณฑิต

สาขาวิชาวิศวกรรมอุตสาหการ ภาควิชาวิศวกรรมอุตสาหการ

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

ปีการศึกษา 2553

MULTI-OBJECTIVE WORKER ALLOCATION IN U-SHAPED ASSEMBLY LINES

Mr. Ronnachai Sirovetnukul

A Dissertation Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy Program in Industrial Engineering
Department of Industrial Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2010

Thesis Title	MULTI-OBJECTIVE WORKER ALLOCATION IN U-SHAPED ASSEMBLY LINES
By	Mr. Ronnachai Sirovetnukul
Field of Study	Industrial Engineering
Thesis Advisor	Associate Professor Parames Chutima, Ph.D.

Accepted by the Faculty of Engineering, Chulalongkorn University in Partial Fulfillment of the Requirements for the Doctoral Degree

.... Dean of the Faculty of Engineering
(Associate Professor Boonsom Lerdhirunwong, Dr.Ing.)

THESIS COMMITTEE

....................................Chairman
(Assistant Professor Rien Boondiskulchok, D.Eng.)

....................................Thesis Advisor
(Associate Professor Parames Chutima, Ph.D.)

....................................Examiner
(Assistant Professor Manop Reodecha, Ph.D.)

....................................Examiner
(Professor Prabhas Chongstitvattana, Ph.D.)

.... External Examiner
(Associate Professor Vanchai Rijiravanich, Ph.D.)

รณชัย ศิโรเวฐนุกูล : การจัดสรรพนักงานแบบหลายวัตถุประสงค์ในสายการประกอบรูปตัวยู. (MULTI-OBJECTIVE WORKER ALLOCATION IN U-SHAPED ASSEMBLY LINES) อ.ที่ปรึกษาวิทยานิพนธ์หลัก : รศ.ดร.ปารเมศ ชุติมา, 285 หน้า.

วิทยานิพนธ์ฉบับนี้ ศึกษาปัญหาการจัดสรรพนักงานแบบหลายวัตถุประสงค์ในสายการประกอบรูปตัวยูแบบสมมาตรและแบบผืนผ้าด้วยเครื่องจักรที่ปฏิบัติการแบบไม่อัตโนมัติ โดยมีวัตถุประสงค์ เพื่อมอบหมายงานเข้าสู่สายการประกอบรูปตัวยูและทำการจัดสรรงานให้พนักงานตามลำดับ ภายใต้ฟังก์ชันวัตถุประสงค์แรก คือ จำนวนพนักงานที่น้อยที่สุด หลังจากนั้นความเบี่ยงเบนของเวลาปฏิบัติงานของพนักงานที่น้อยที่สุดและเวลาเดินของพนักงานที่น้อยที่สุด จะถูกพิจารณาอย่างพร้อมเพรียงกัน ผลิตภัณฑ์ที่หลากหลายถูกประกอบขึ้นมาภายใต้ปัญหามาตรฐานตั้งแต่ 7-297 งาน ด้วยรอบเวลาการผลิตที่ถูกกำหนด งานวิจัยนี้ได้สร้างรูปแบบทางคณิตศาสตร์สำหรับปัญหาดังกล่าว ซึ่งเป็นปัญหาการหาคำตอบที่ดีที่สุดเชิงการจัด การแก้ปัญหาด้วยการพัฒนาอัลกอริทึมเชิงวิวัฒน์ถูกนำมาใช้สำหรับการหาคำตอบของปัญหาขนาดใหญ่ เพื่อคุณภาพคำตอบที่ดีและการใช้เวลาที่เหมาะสม ด้วยความสัมพันธ์ที่เหมาะสมที่สุดแบบพาเรโตอัลกอริทึมที่ถูกพัฒนาเพื่อนำมาแก้ปัญหาในการศึกษานี้มี 4 ชนิด ได้แก่ อัลกอริทึมเชิงพันธุกรรมเรียงลำดับแบบไม่ถูกข่ม (Non-dominated Sorting Genetic Algorithm-II: NSGA-II) อัลกอริทึมแบบเมมเมติก (Memetic Algorithm: MA) อัลกอริทึมแบบบรรจบ (COINcidence algorithm: COIN) และอัลกอริทึมการหาคำตอบที่เหมาะสมที่สุดแบบฝูงอนุภาคด้วยความรู้เชิงลบ (Particle Swarm Optimization with Negative Knowledge: PSONK) จากการเปรียบเทียบคุณภาพคำตอบที่เหมาะสมที่สุดแบบพาเรโตด้วยตัวชี้วัดสมรรถนะ 3 ตัว ได้แก่ การลู่เข้าสู่คำตอบที่แท้จริง (Convergence to the Pareto-optimal set) การกระจายของกลุ่มคำตอบที่ได้ (Spread) และอัตราส่วนของจำนวนกลุ่มคำตอบที่หาได้เทียบเท่ากลุ่มคำตอบที่แท้จริง (Ratio of non-dominated solutions) ผลการทดลองชี้ให้เห็นว่าวิธี MA ให้ผลดีที่สุดและอีกสามวิธีที่เหลือให้ผลที่ไม่แตกต่าง สำหรับตัวชี้วัดเวลาที่ใช้ในการประมวลผล (CPU time) ผลการทดลองพบว่าวิธี PSONK และ COIN ใช้เวลาใกล้เคียงกันแต่เร็วกว่าวิธี NSGA-II และ MA อย่างน้อย 10 เท่า สำหรับทุกปัญหา

ภาควิชา วิศวกรรมอุตสาหการ ลายมือชื่อนิสิต รณชัย ศิโรเวฐนุกูล

สาขาวิชา วิศวกรรมอุตสาหการ ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์

ปีการศึกษา 2553

4971859421 : MAJOR INDUSTRIAL ENGINEERING
KEYWORDS : WORKER ALLOCATION / U-SHAPED ASSEMBLY LINES / MULTIPLE OBJECTIVES / EVOLUTIONARY ALGORITHMS.

RONNACHAI SIROVETNUKUL : MULTI-OBJECTIVE WORKER ALLOCATION IN U-SHAPED ASSEMBLY LINES. / THESIS ADVISOR : ASSOC. PROF. PARAMES CHUTIMA, Ph.D., 285 pp.

This dissertation studies multi-objective U-shaped assembly line worker allocation problems with symmetrical and rectangular layouts having manually operated machines. The objective is to assign tasks into a U-line and allocate tasks to workers hierarchically. The primary objective function of a number of workers is minimized. Then, the deviation of operation times of workers and the walking time are minimized simultaneously. Several products are assembled in 7-task to 297-task benchmarked problems with given cycle times. This problem, a combinatorial optimization problem, is modeled with mathematical formulation initially. To produce a good quality solution and time, the development of evolutionary algorithms is used for a large-sized problem. With the Pareto dominance relationship that is used to solve a problem instead of relative preference of multiple objectives, four algorithms have been developed as follows: Non-dominated Sorting Genetic Algorithm-II (NSGA-II), Memetic Algorithm (MA), COINcidence algorithm (COIN), and Particle Swarm Optimization with Negative Knowledge (PSONK). The quality solutions of Convergence to the Pareto-optimal set, Spread, and Ratio of non-dominated solutions are compared in each problem for all multi-objective algorithms. The results indicate that MA is the best and the rest of algorithms are indifferent. For the performance of Central Processing Unit time (CPU time), the computational results show that PSONK is almost identical to COIN, but they are faster than NSGA-II and MA by at least 10 times for all problems.

Department : Industrial Engineering Student's Signature ....
Field of Study : Industrial Engineering Advisor's Signature ....
Academic Year : 2010

ACKNOWLEDGEMENTS

This dissertation could not be completed without the welcome help of certain people throughout the whole study period. In particular, I would like very much to acknowledge the guidance and support of my main supervisor – Assoc. Prof. Dr. Parames Chutima – during every stage of the work for his discussion of a number of interesting ideas and concepts, several excellent instructions, beneficial suggestions, and, equally importantly, his moral support. I would like to gratefully thank Asst. Prof. Dr. Rien Boondiskulchok, Asst. Prof. Dr. Manop Reodecha, Prof. Dr. Prabhas Chongstitvattana, and Assoc. Prof. Dr. Vanchai Rijiravanich for their valuable comments and contributions throughout this study. I appreciate their willingness to talk to me at any time I need them. Then, the author sincerely thanks Prof. Dr. Sirichan Thongprasert, who was my main master thesis advisor and gave me a hopeful saying – "Just do it". I am grateful to Assoc. Prof. Damrong Thawesaengskulthai, Assoc. Prof. Dr. Peerayuth Charnsethikul, Assoc. Prof. Dr. Prapaisri Sudasna Na Ayudthaya, and Assoc. Prof. Rachavarn Kanjanapanyakom for their recommendation letters. Additionally, I owe a special gratefulness to several Thai companies for providing me with the chance to get a better understanding of apparel business. Special thanks are also extended to Prof. Dr. Bart MacCarthy giving me a motto – "Research is never easy.", Lect. Dr. Thanakorn Uan-on, and Assoc. Prof. Dr. Duangpun Kritchanchai for their support and encouragement.

I am very grateful for partial funding provided by Mahidol University, Royal Thai Government, and my parents. I express my thanks to the IE department for allowing me to participate in its excellent program and for giving me facilities and assistance including all my teachers and educational schools. I wish to thank all my many great friends at the department of Industrial Engineering at both Chulalongkorn University and Mahidol University for their help and companionship. I am grateful to Graham Keith Rogers for his editing suggestions. I also acknowledge Warin Wattanapornprom, Penpak Pinkoompee, Panuwat Olanviwatchai, and Charat Jirakomate for their unconditional help in verification and validation of the program.

Finally, the greatest appreciation dedicate to my father (Charnchai), mother (Weerawan), brother (Chinnadeth), sister (Waranuch) and all the members of my families, Chuckpaiwong, Cholitkul, Kuansuwan, Vitayathanagorn, Kingsaingam, and Sirovetnukul, who have always been there for me. My special appreciation goes to my beloved wife – Donlaphon (Pook) – for her patience, continual encouragement, being there for me, and helping me to keep going when I felt that the task was impossible. I thank for your uncomplaining acceptance of the time I had had to spend away from you for one year in Nottingham, UK. Your help and understanding during this very long period of my research will not be forgotten. Last but not least, I dedicate this dissertation to my beloved daughters – Phanphak (Mook) and Koranit (Mai).

CONTENTS

Page

Page

Page

LIST OF TABLES

Tables **Page**

Tables **Page**

LIST OF FIGURES

Figures **Page**

Figures **Page**

Figures **Page**

Figures **Page**

CHAPTER I

INTRODUCTION

1.1 General Background

There is nothing wrong about mass production, but usually the process is a form of the straight-line assembly line system. The decision to transform straight-line assembly systems to U-shaped assembly line systems constitutes a major layout design change and investment for assembly operations. Proponents of the lean manufacturing and just-in-time (JIT) philosophies assert that U-shaped assembly systems offer several benefits over traditional straight-line layouts (Cheng *et al.*, 2000) including an improvement in labor productivity. U-lines have become popular in order to obtain the main benefits of smoothed workload, multi-skilled workforce and other principles of the JIT philosophy. Many researchers agree that U-lines are one of the most important components for a successful implementation of JIT production systems (Monden, 1993 and Miltenburg, 2001a). Approximately 75% of U-lines in the world are arranged to produce more than one product type or different models of a product on the same line (Miltenburg, 2001a). This type of production is called mixed-model production. The U-lines on which mixed-model production is performed are called mixed-model U-lines (MMULs). Although mixed-model straight lines are widely used in traditional production systems, MMULs have become a cornerstone of JIT systems as they are used to improve quality and productivity and to adapt demand changes quickly and cost effectively. A MMUL has several advantages over its equivalent straight line. Since workers work closed to each other in a MMUL, visibility, communications and interaction are improved. This also enables workers to help each other solve problems and to improve their skills. Such multi-skilled workers will then be more capable of responding to changes in cycle time or output rate of the MMUL. The number of workers required on a MMUL is never more than that required on a straight line.

In general, the traditional scheduling problems such as job shop, flexible flow shop or assembly line problems likely express the general forms which are more complicated beneficial than the specific forms. After surveying relevant literature papers, this research problem which is more complex and practical in the real situation fulfills rightsizing, worker-machine assignment, and worker's mobility reasonably. Filled in the gaps, the interesting issue for assembly line problems is worker allocation in U-shaped mixed-model assembly lines with manually operated machines under multiple objectives. In this dissertation proposal, the first section describes the general background. Secondly, the importance of related problems is addressed. In the third section, the literature review is presented. The statement of problem is identified in the fourth section. Fifthly, the objective of this dissertation is proposed. In the next sixth and seventh parts, the dissertation scope and contribution are given. In the following section, the processes of research methodology are presented in order. Finally, the plan of work and references are shown.

1.1.1 Definition of assembly line

An assembly line is a manufacturing process in which component parts are added to a product in a sequential manner to create a finished product. Assembly lines are special flow-line production systems which are of great importance in the industrial production of high quantity standardized commodities. Recently, assembly lines even gain importance in low volume production of customized products (mass-customization). Due to high capital requirements when installing or redesigning a line, its configuration planning is of great relevance for practitioners. Accordingly, this attracted attention of several researchers, who tried to support real-world configuration planning by suited optimization models.

1.1.1.1 Definition of line balancing

Balancing an assembly line means allocating the basic assembly tasks to be carried out to different stations, pursuing specific goals and all in compliance with given constraints. In other words, balancing a line means determining the number of stations to be used and the tasks allocated to each station.

1.1.1.2 Assembly line balancing objective

The assembly line balancing objective is to balance the task workload across workstations so that no workstation has an excessively high or low task workload.

1.1.1.3 Definition of worker allocation

The worker allocation problem consists of providing a simultaneous solution to a double assignment: (1) tasks to stations; and (2) available workers to stations. In manufacturing, the purpose of worker allocation is to minimize the labor costs, by telling a production facility what to make, when, with which staff, and on which equipment.

1.1.2 Classification of assembly line balancing and worker allocation problems

Research on assembly line balancing has focused primarily on the so called SALBP (Simple Assembly Line Balancing Problem) (Ghosh and Gagnon, 1989). In SALBP the complexity has been reduced considerably by introducing several simplifying assumptions. With regard to the objective function and restrictions considered, SALBP can further be divided into a range of sub-problems (SALBP-F, SALBP-1, SALBP-2, SALBP-E, see Table 1.1) which have been subject to extensive research later. In other words, two types of flow lines are distinguished. The first type is dedicated to the production of one single product (a single model line). The second type is dedicated to the assembly of more than one model (mixed and multi flow lines). With an increasing requirement for flexibility of production, motivated by fast changes in technology and by customer demand for greater product variety, mixed-model assembly lines are replacing the traditional mass production assembly lines. Mixed-model production is important to respond to diversified expectations of today customer perspective.

Table 1.1 Versions of SALBP

		Cycle time c	
		Given	Minimize
No. m of stations	Given	SALBP-F	SALBP-2
	Minimize	SALBP-1	SALBP-E

In spite of the enormous academic effort in assembly line balancing, there remains a considerable gap between requirements of real configuration problems and the capability of academic research development. Several issues to assembly line design and problems have been proposed in the section of literature review.

In SALBP-1 the aim is to minimize the number of stations given a target cycle time, and then the assembly line worker allocation problem of type I can be also constructed. However, the worker allocation problem of assembly line processes in this research is mainly based on the U-shaped Assembly Line Balancing Problem of type I (UALBP-I).

1.1.3 Doctoral framework

Figure 1.1 illustrates the doctoral framework in this study. All stages are proposed as follows: the research problem of the Non-deterministic Polynomial-time hard (NP-hard) class, mathematical model, approximation method, validation of all algorithms, and experimental results, conclusion and discussion.

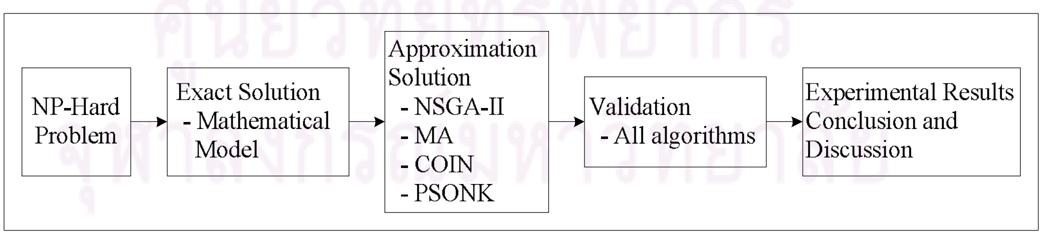


Figure 1.1 Framework of the doctoral dissertation

1.1.4 Brief expected outcomes

Two main expected outcomes of this research are developed in brief as follows:

1) This research first contributes the worker-machine assignment with minimum number of workers in U-shaped assembly lines that are no automated machines. Its expected outcomes are also loops of U-shaped machines assigned to each worker having no crossing path under two objective functions of smoothed workload in a sense of equity and minimum walking time to save the space needed for the actual size of a U-shaped line and shorten the distance for communication between workers.

2) The existing solution processes of multi-objective evolutionary algorithms are applied to search the Pareto-optimal frontier. Moreover, the comparisons of their computational results give us the performance of algorithms.

1.2 Importance of Related Problems

There are many imperative problems in this research. The importance of the worker allocation problems is given first in this section. Then, the physical importance of the mixed-model U-shaped assembly line problem is addressed in the following section.

1.2.1 Importance of the worker allocation problem

This research focuses on the worker allocation is because it is one of the most important decisions that can achieve productivity gains and rightsizing in a labor intensive manufacturing system. If one worker can only attend one machine, then the required number of workers is proportional to the number of machines in a workstation. However, one worker operating a few machines is more interesting in this research.

1.2.2 Importance of U-shaped Mixed-Model Assembly Line Problems (UMMALPs)

1.2.2.1 Benefits of the U-shaped cell

The Shingo Prize for excellence in manufacturing also encourages the use of U-lines (ANOM., 1994). Through the achievement of these goals the manufacturing ‘cell’ becomes an important weapon in the reduction of production cost. There are many papers that reveal advantages of the u-shaped layout over the linear layout (Ohno and Nakade, 1997; Urban, 1998; Cheng *et al.*, 2000; Aase *et al.*, 2004; Hwang *et al.*, 2008; Hwang and Katayama, 2009. Obviously, the reported benefits are impressive when a company changes from traditional production lines to U-shaped lines. Productivity improved by an average of 40-80%. Work in process (WIP) drops by 60-85%. Lead time reduces by 50-75%. Defective rate drops by 40-80% (Miltenburg, 2001b). Cheng *et al.* (2000) found collectively that the following benefits and factors favoring U-lines are better *volume flexibility, worker flexibility, number of workstations, material handling, visibility and teamwork, and rework*:

a) *volume flexibility:* The production rate of a line in a JIT environment changes frequently. In such an environment, a U-line is preferred to a straight line because of its volume flexibility. By increasing or decreasing the number of workers on the line, a company can adjust the production rate as required. This level of volume flexibility is harder to obtain with a straight line.

b) *worker flexibility:* Since walking distance is shorter on a U-shape than on a straight line, it is easier for one worker to oversee several work centers.

c) *number of workstations:* According to Figure 1.2 by Aase *et al.* (2004), without the issue of walking time, the number of workstations required on a U-line is never be more than, and sometimes less than, that required on a straight line. This is because there are more possibilities for grouping tasks into workstations on a U-line.

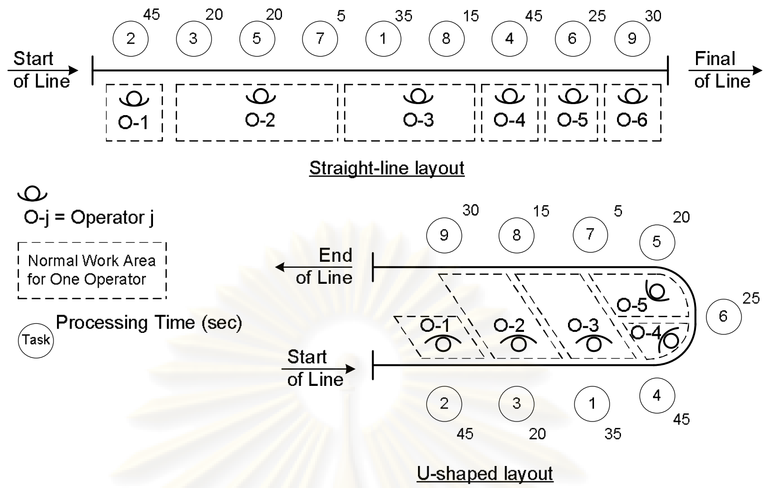


Figure 1.2 Balances for the SALB and SULB problems

d) *material handling:* A U-line eliminates the need for special material-handling equipment such as conveyors and special material-handling workers. Instead, production workers move products from machine to machine. It comforts for dropping raw materials and picking up finished goods because the entrance and the exit of the line are in the same position.

e) *visibility and teamwork:* The compact size of a U-line improves visibility and communication. This enhances teamwork, gives a greater sense of belonging, and increases responsibility and ownership compared to a straight line, where workers are spread out along a long line and may be separated by walls of inventory.

f) *rework:* A tenet of Total Quality Management (TQM) is quality at the source, which calls for correcting quality problems as soon as possible after they occur by returning a defective product to the station where it was produced. In a U-line, the distance to return the defective product is short, making it easier to follow this tenet. This is in contrast to the traditional policy of sending the defective product to a separate rework area.

In another viewpoint of U-line benefits by Clegg *et al.* (1999; p.131), the U-shaped cell is used to achieve three goals:

The first of these is *shojinka*, flexibility in the number of workers in the cell so that demand changes can readily be adapted to. Working inside the 'U', workers are required to operate more than one workstation simultaneously and must learn to perform all operations through job rotation.

The second goal is the reduction of unnecessary processes in the progressive system through the continuous improvement of work processes and machines for the number of workers required in a work cell.

The third is the introduction of 'one-piece flow' of work-in-progress units by replacing 'planned-center production' with JIT demand-pull, eliminating large-batch production (based on economic order quantities) and drastically reducing machine set-up times.

1.2.2.2 U-shaped mixed-model assembly line problems

The Mixed-Model Assembly Lines (MMALs) consist of finding a feasible line balance, i.e. an assignment of each task to a station such that the precedence constraints and some restrictions are fulfilled. The MMALs have become popular in recent years as an integral part of JIT production systems under increasing product variety. Among the many new production lines, they are being arranged as a 'U-shaped line' rather than a straight line as U-shaped mixed-model assembly lines (UMMALs). In any case, a U-shaped line provides more alternatives for assigning tasks to workers (or workstations) than comparable straight lines because workers can handle not only adjacent tasks, but also tasks on both sides of the U-shaped line. Another advantage is that a U-shaped line allows workers to work closely together, in turn both saving the space needed for the actual size of a U-shaped line and shortening the distance for communication between workers, creating a safer work environment. For the lack of a better team, a U-shaped line is set up to be 'user friendly' and possibly even to make work more satisfying since workers can easily be involved in team work communication. Therefore, it is not only scientifically effective because it is ergonomically fit, increases production output and saves on space, but it is also sociologically effective simply because it places the workers in close proximity. In addition, U-shaped lines could minimize workers; consequently, the workers at each workstation are required to possess more skills than on straight lines in some cases (Hwang and Katayama, 2009).

1.3 Problem Statement

This dissertation addresses the worker allocation problems in the assembly line having some special environments. First, the description of the problem development is described in this section. Secondly, the research gaps and research questions are presented in the following section.

1.3.1 Problem development from the real situation to the research problems

The reported benefits are impressive when a company changes from traditional production lines to U-shaped lines. Productivity improves by an average of 40-80%. Work in process (WIP) drops by 60-85%. Leadtime reduces by 50-75%. Defective rate drops by 40-80% (Miltenburg, 2001b). In a traditional production system, an order is generated in certain batches, that is, each order or job could have a quantity of more than one. Just-in-time (JIT) production system is adopted extensively in today's manufacturing industry such as apparel industry to meet the production demand (Guo *et al.*, 2006a). Mixed-model assembly line balancing is an approach employed to handle increasing product variety (Guo *et al.*, 2006b). Moreover, the JIT is one of interesting cases in U-shaped mixed-model production lines studied by Miltenburg (2002) and Kara *et al.* (2007). For the system, each order may be unique (a single product type and one piece). In other words, volume of each order generated is the batch size of one. Work-in-process in the system is always constant. In the recent decades, many apparel manufactures have installed several production systems on their apparel assembly line such as the traditional progressive bundle system and the automated Unit Production System (UPS) by Song *et al.* (2006). The assembly line to be studied in this paper is a modular production system (or a single U-line).

The U-shaped assembly line worker allocation problem is studied because the conditions of unbalanced straight line due to customized products and workers cut off. There are no automated processing machines in the production system. After each worker operates an item at a machine, a worker walks several patterns such as a circular loop, a rectangular loop or a straight-line loop and takes it to the next machine and at the end of each intra cell and generally a worker hands it over to the adjacent worker along the sequence of U-line. From some of sample companies, there is no equity of workload although line efficiency has been improved for a year. In practice, most of companies track on the assembly line problem of type F (by given number of workstations and given cycle time) and improve line efficiency by escaping the complexity of the problem. However, solving better methods, it is essential to study on the problem of type I (by minimum number of workstations at given cycle time).

In brief, the system dealt with is the U-shaped manual assembly line of type I. Each worker performs all assembly tasks allocated to a given workstation without crossing path. There are a single product and different product types (models) under a product family.

1.3.1.1 Major system features

U-shaped production line can be described as a special type of cellular manufacturing used in JIT production systems and Lean Manufacturing. The U-line arranges machines around a U-shaped line in the order in which production operations are serial. Workers work inside the U-line. One worker supervises both the entrance and the exit of the line. In apparel industry, the machine's efficiency is determined by the worker's performance. Machine-work is not separated from worker-work, that is, machines work dependently. Standard operation charts specify exactly how all work is done. U-lines are rebalanced periodically when production requirements change. The U-line satisfies the flow manufacturing principle. This requires workers to be multi-skilled to operate several different machines or processes and they also have same capability. A worker's efficiency varies in different operations, but they are determined with deterministic manual times. It also requires workers to work standing up and walking because they need to operate different machines. U-lines may be simple or complex, depending on the number of tasks to be performed, the production volume and setup times. From the study, setup times are negligible. Therefore, U-lines can be operated as single-model and mixed-model lines where each worker is able to produce any product in any cycle. *However, if setup times are larger, multiple U-lines that are scoped in this research are formed and dedicated to different products.* In the facet of the worker, all of the workers who are selected will be allocated for the new job. Each worker will only work on one machine at a time.

1.3.2 Research gaps

Even though U-shaped assembly line balancing problems have been studied by several previous researchers from the literature review, the multi-objective worker allocation problem is hardly studied in the U-shaped manual assembly line. The simultaneous optimization of worker sizes, cycle times, line balancing, job sequencing, and multi-function worker allocation is an extremely complex and difficult problem to solve (Heike *et al.*, 2001), but this research limits to study selected three issues of worker sizes, cycle times and line balancing. As a result, some gaps where this research should be fulfilled are as follows:

- This research problem is the single U-line that is one of several U-line types (Miltenburg, 2001b).

- The single U-line layout is mapped to several dimensions of the front, back and side of U-line, named the side U-line ratios.

- Worker allocation is designed how to assign grouped tasks unto workers (or workstations) in the single U-line with equity of workload and minimum walking path, which is necessary non-value added.

In the next section, some problem examples are presented to assist in understanding this problem.

1.4 Research Objective

This research objective is to develop a new evolutionary algorithm that allocates the minimum number of workers in a U-shaped manual assembly line to minimize both the deviation of operation times of workers and their walking time simultaneously.

1.5 Scope of the Study

Some broad issues are ignored in the scope of the study and can be developed further:

- when and how to rebalance the U-line;
- how to balance and sequence the U-line at the same time;
- how to study worker allocation problems in other types of complex U-lines.

There are three components of the worker allocation problems in the single U-shaped manual assembly line: (1) input parameters; (2) assembly line characteristics and a set of constraints; and (3) objective functions.

1.5.1 Input parameters

In any time period, the number of jobs is deterministic and job arrivals come from not only new customer orders but also remaining jobs from the previous planning period that is not achieved. Each of the n jobs is an entity worked on many operations. No job priority constraint is allowed, that is, each job is allowed to start its processing whenever it is ready. These jobs are sorted by the daily production order excluding the sequencing problem. Any job is part of UALWAP-type I. Twenty-five problems consisting of eleven precedence graphs and various given cycle times are representative of all jobs.

1.5.2 Assembly line characteristics and a set of constraints

This section provides the detailed limitations of the existing research as follows:

1. Although assembly line balancing from the literature survey has given us many standard problems such as SALBP-1, SALBP-2, MMALBP and so forth, only the standard assembly line problem, named UALBP. The UALBP-1 is narrowed down in this research.

2. Given precedence graphs for an assembly line are produced from the process of making intermediated parts to the final assembly line.

3. Nowadays shorter product life cycles and increased demands for customization make it difficult to produce some products on traditional production lines. The modern assembly line is necessary to fulfill customized orders, but not lacking of the high volumes of a continuous line (Mass customization). Just-in-time manufacturing is determined with elements of takt time, standard work, flow manufacturing on U-shaped lines, pull production, and jidoka not allowing defective parts to go from a machine to the next.

4. Processing times of all tasks in each precedence graphs are determined.

5. Each task is assigned to only one machine and one worker.

6. The precedence relationship among tasks must be satisfied.

7. Each station is manned by one worker (no crossing path).

8. Workers are assumed to be homogeneous and multi-functional skill (same efficiency on a same operation) although it is not stable even when working in

the same operation due to human factors such as worker's emotion, motivation, skill level and experience or other uncertainties like machine breakdowns.

9. The transportation time of parts between any two machines is negligible.

10. Jobs are available for processing at the next machine immediately after completing processing at the previous machine with one kanban tray.

11. Preemption is not permitted, that is, when an operation is started, it must be completed without interruption.

12. There is no buffer in every machine.

13. No machine breakdown is due to steadiness of scheduled maintenance.

14. Raw materials supply to all stages of the line is unlimited.

15. The system is under one piece flow manufacturing moving one workpiece at a time between operations within a U-line. It keeps work-in-process at the lowest possible level. It encourages work balance, better quality and a host of internal improvements.

16. Data used in this dissertation are gathered from the previous problems and real situation.

1.5.3 Objective functions for the research problems

The multiple objective functions are three minimum objectives: number of workstations (m), deviation of operation times of workers (DOW) and walking time (WT).

1.6 Research Contribution

The expected outcomes derive from this proposed research include:

1. The first contribution of this research develops the existing and new worker allocation problems of the single U-shaped assembly line having manually operated machines in several fixed layouts;

2. The second contribution gives us worker-machine assignment and walking path of each worker reducing the deviation of operation times of workers and walking time;

3. Thirdly, the ameliorated heuristic algorithm is applied from existing solution processes of Multi-Objective Evolutionary Algorithms (MOEAs), which are employed to search Pareto-optimal frontier.

4. Finally, the proposed methodology may be utilized to some other industries having the same circumstances.

1.7 Dissertation Structure

The outline of this dissertation is organized as follows. Chapter I states the general background, statement of problem, objective, scope of study, and contribution. The relevant literature of U-line problems, assembly line balancing problems, worker allocation problems, and solution techniques is reviewed in Chapter II. The research methodology is presented in chapter III. In chapter IV, the mathematical model is formulated under the single U-shaped assembly line worker allocation problems with the consideration of walking time. Then, an illustrative example is presented. Based on the complexity of the problem, Chapter V provides the multi-objective evolutionary algorithms of NSGA-II, MA, COIN, and PSONK. In Chapter VI, all of computational results are proposed, compared, and discussed. However, the initialized parameters input to four algorithms in this chapter are prepared. Finally, the conclusion of this research is presented and the future directions are also suggested in Chapter VII.

CHAPTER II

LITERATURE REVIEW

2.1 Introduction

To regain competitive edge, the just-in-time manufacturing is crucial to respond diversified expectations of customer perspective. The line balancing of modern assembly lines has been an interesting topic, especially in such areas as a U-line and worker allocation. Most of the previous line balancing approaches attempted to solve how to assign the tasks to an ordered sequence of stations so that the precedence relations should satisfied and some measures of effectiveness should be optimized. However, the worker factors are seldom considered in solving the assembly line balancing problem. They are also widely ignored in the real life situations of labor intensive industry such as apparel manufacturing. The line balancing problem can be replaced with the worker allocation problem, in which the goal is to determine which assigned machines are handled by each of workers. In the following sections in this chapter, previous works are reviewed on U-shaped assembly lines, line balancing problems, worker allocation problems, exact solutions, and multi-objective evolutionary algorithms.

2.2 History of Assembly Lines

Basic production layout formats by which departments are arranged in a facility are defined by the general pattern of work flow; there are three basic types and one hybrid type respectively (Chase *et al.*, 1998). In a fixed-position layout, the product remains at one location. Manufacturing equipment is moved to the product rather than vice versa. A process layout (also called a job-shop or functional layout) is a format which similar equipment or functions are grouped together, such as all lathes in one area and all stamping machines in another. A product layout (also called a flow-shop layout) is one which equipment or work processes are arranged according to the progressive steps by which the product is made. The part for each part is, in effect, a straight line. An assembly line is a special case of product layout. In a general sense, the term assembly line refers to progressive assembly linked by some material handling device. The usual assumption is that some form of pacing is present and the allowable processing time is equivalent for all workstations. Within this broad definition, there are important differences among line types. A few of these are material handling devices (belt or roller conveyor, overhead crane); line configuration (U-shape, straight, branching); pacing (mechanical, human); product mix (one product or multiple products); workstation characteristics (workers may sit, stand, or walk with the line); and length of the line (few or many workers). These characteristics may be classified clearly by Boysen *et al.* (2006).

A group technology (cellular) layout groups dissimilar machines into work centers (or cells) to work in products that have similar shapes and processing requirements. A group technology (GT) layout is similar to process layout in that cells are designed to perform a specific set of processes, and it is similar to product layout in that the cells are dedicated to a limited range of products. A cell involves multi-functional employees and arranges in a U-shaped way.

To ease the communication between researchers and practitioners, the development of assembly lines and a classification scheme of assembly line balancing problems are reviewed (Boysen *et al.*, 2007). This is a valuable step in identifying remaining research challenges which might contribute to closing the gap. Assembly line balancing problems (ALBP) arise whenever an assembly line is configured, redesigned or adjusted. The first published paper of the assembly line balancing problem (ALBP) was made by Salveson (1955) who suggested a linear programming solution. Since then, the topic of line balancing has been of great interest to researchers. There are exact methods to solve the ALB problems. (e.g. Jackson, 1956; Bowman, 1960; Van Assche and Herroelen, 1978; Mamoud, 1989; Hackman *et al.*, 1989; Sarin *et al.*, 1999). However, since the ALB problem falls into the NP hard class of combinatorial optimization problems (Gutjahr and Nemhauser, 1964), numerous research efforts have been developed consistently from the efficient algorithms for obtaining optimal solutions to computer-efficient approximation algorithms or heuristics (e.g. Kilbridge and Wester, 1961; Helgeson and Birnie, 1961; Hoffman, 1963; Mansoor, 1964; Arcus, 1966; Baybar, 1986a). In addition, with the growth of knowledge on the ALB problem, review articles are necessary to organize and summarize the finding for the researchers and practitioners. In fact, several articles (e.g. Kilbridge and Wester, 1962; Mastor, 1970; Johnson,1981; Talbot *et al.*,1986; Baybars, 1986b; Ghosh and Gagnon, 1989; Erel and Sarin, 1998) have reviewed the work published on this problem. Characteristics of balancing problems summarized into Kriengkorakot and Pianthong (2007) give some classification schemes (cf Ghosh and Gagnon, 1989; Becker and Scholl, 2006) as follows:

I. Ghosh and Gagnon (1989) classified the ALBP into four categories shown in Figure 2.1: (1) Single Model Deterministic (SMD); (2) Single Model stochastic (SMS); (3) Multi/Mixed Model Deterministic (MMD); (4) Multi/Mixed Model stochastic (MMS).

II. Becker and Scholl (2006) have classified the main characteristics of assembly line balancing problems considered in their several constraints and different objectives as shown in Figure 2.2. It illustrated the classification of assembly line balancing problems.

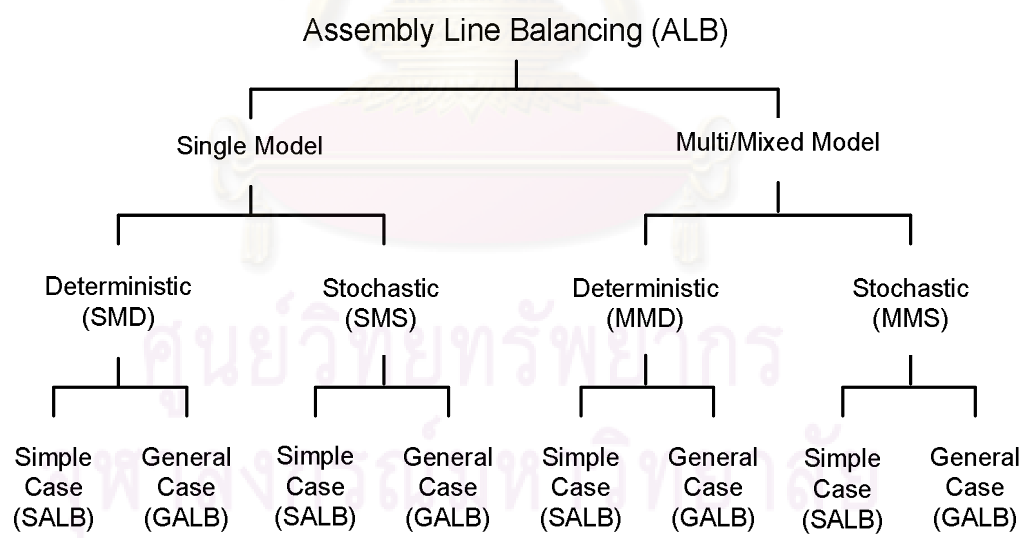


Figure 2.1 Classification of assembly line balancing literature (Ghosh and Gagnon, 1989)

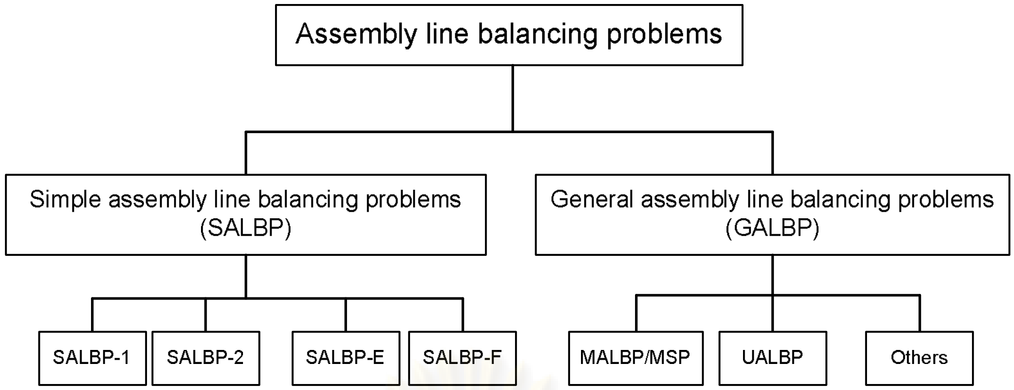


Figure 2.2 Classification of assembly line balancing problems (Becker and Scholl, 2006)

(1) SALBP: The simple assembly line balancing problem is relevant to straight single product assembly lines where only precedence constraints between tasks are considered:

- Type 1 (SALBP-1) of this problem consists of assigning tasks to work stations such that the number of stations *(m)* is minimized for a given production rate (fixed cycle time, *c*).

- Type 2 (SALBP-2) is to minimize cycle time (maximize the production rate) for a given number of stations *(m)*.

- Type E (SALBP-E) is the most general problem version maximizing the line efficiency *(E)* thereby simultaneously minimizing *c* and *m* considering their interrelationship.

- Type F (SALBP-F) is a feasibility problem which is to establish whether or not a feasible line balance exists for a given combination of *m* and *c*.

(2) GALBP: In the literature, all problem types which generalize or remove some assumptions of SALBP are called the generalized assembly line balancing problem (GALBP). This class of problems including UALBP and MALBP is very large and contains all problem extensions that might be relevant into practice including equipment selection, processing alternatives, assignment restrictions and so on.

- MALBP and MSP: The mixed-model assembly line balancing problem (MALBP) and Mixed-model sequencing problem (MSP) produce several models of a basic product in an intermixed sequence. Besides the MALBP, which has to assign tasks to stations considering the different task times for the different models and find a number of stations and a cycle time as well as a line balance such that a capacity- or even cost-oriented objective is optimized (Scholl, 1999, Chapter 3.2.2). However, the problem is more difficult than in the single-model case, because the station times of the different models have to be smoothed for each station (Merengo *et al.*, 1999). The better this balancing works, the better solutions are possible in the connected mixed-model sequencing problem. The MSP has to find a sequence of all model units to be produced such that inefficiencies (work overload, line stoppage, off-line repair and so forth) are minimized. (Bard *et al.*, 1992; Scholl *et al.*, 1998).

- UALBP: The U-line assembly balancing problem (UALBP) considers the case of U-shaped (single product) assembly lines, where stations are arranged within the shape of U. As a consequence, workers are allowed to work on either side of the U, that is, on early and late tasks in the production process simultaneously. Therefore, modified precedence constraints have to be observed. By analogy with SALBP, different problem types can be distinguished. (Miltenburg and Wijngaard, 1994; Urban, 1998; Scholl and Klein,1999; Erel *et al.*, 2001).

2.2.1 Level of automation

There are two kinds of automated level as follows:

1) *Manual lines:* In spite of the major advances in the automation of assembly processes, there are still many assembly systems which mainly or completely rely on manual labor. Manual lines are especially common where work pieces are fragile or if work pieces need to be seized frequently. In countries where wage costs are low, manual labor can also be a cost efficient alternative to expensive automated machinery.

2) *Automated lines:* Fully automated lines are mainly implement wherever the work environments is in some form hostile to human beings, as for instance in the body and paint shops of the automobile industry, or where industrial robots are able to perform tasks more economically and with a higher precision (e.g. metal processing tasks).

2.3 Mixed-Model Assembly Lines (MMALs) in Just-In-Time

Just-In-Time (JIT) has revolutionized the manufacturing world. In the late 1980s everyone was interested in implementing JIT to their manufacturing firm. JIT means producing what is needed when needed and no more. Anything over the minimum amount necessary is viewed as waste, because effort and material expended for something not needed and cannot be utilized now. This definition of JIT leaves no room for surplus or safety stock. No safety stocks are allowed because if you cannot use it at present, you do not also need to make it. The JIT principles relate to the four Ms that imposes additional conditions on the labor intensive line: Man (multiple skills); Method (flow production, manual or conveyor line and visual control); Material (immediate detection); and Machine (flow line layout and small and inexpensive machines). In addition, five elements that approach JIT are takt time, flow manufacturing on U-shaped production lines, standard work, pull production control, and jidoka (Miltenburg, 2001b). A goal of JIT production system is cycle time (C_i) = takt time ($\hat{C}_i$) for each *i*. Mixed-model production is crucial to respond diversified expectations of today's customer perspective. In such a production environment, more than one product with similar production characteristics or different models of a product are produced or assembled on the same line. The use of a U-line that often adopts the strategy of mixing product models is an important element in JIT production. It enables to easily adjust production facilities to demand changes, and increase labor productivity. Many benefits of U-lines utilized in JIT environment are reported in the literature (Monden, 1983; Miltenburg and Wijngaard, 1994; Cheng *et al.*, 2000; Miltenburg, 2001b) including increasing productivity, reduced work-in-process inventory, shorter throughput and improved quality. A successful utilization of mixed-model U-lines (MMULs) in a JIT environment requires effective solutions to two important problems (Kara *et al.*, 2007b and

Miltenburg and Sparling, 1998): (i) the mixed-model U-line balancing (MMU/LB) and (ii) the mixed-model U-line sequencing (MMU/S). The contribution of MMULs to JIT production can be increased by solving these two problems. However, according to Figure 2.3 how to accomplish the strategy of JIT is summarized by combining Kara *et al.* (2007b) and Davis *et al.* (2003).

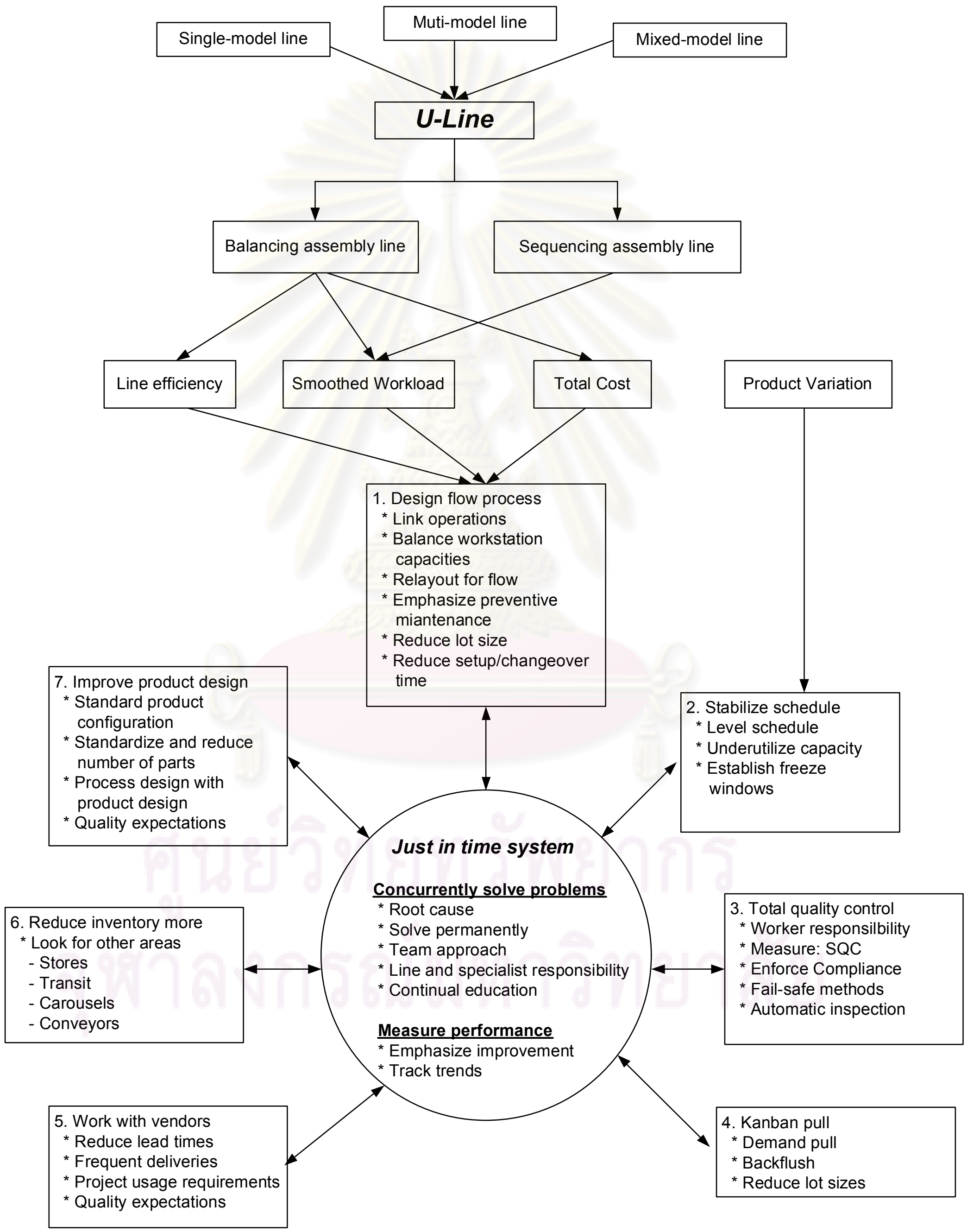


Figure 2.3 How to attain just-in-time manufacturing

2.4 Research on U-shaped Mixed-Model Assembly Lines (MMUALs)

2.4.1 U-shaped assembly lines

Since the focus of this study is the UALBP-1, the literature on U-shaped assembly lines is reviewed. Mitenburg and Wijngaard (1994) were the first to compare a U-shaped line with a straight line. They use a dynamic programming procedure and heuristic methods developed for the SALBP to solve the UALBP. Based on the work of Schrage and Baker (1978) they develop forward and backward "ideals" that are used to provide sets of feasible tasks. Workstations are assigned tasks by simultaneously moving backward and forward through the network. Their computational results show that the dynamic programming and modified heuristics worked well, though dynamic programming was used only for problem sets up to eleven tasks. An integer programming used a "phantom" network to move forward and backward through the network, and was able to optimally solve problems with up to forty-five tasks. Miltenburg (1998) analyzes the U-line facility problem where a multi-line station may include tasks from two adjacent U-lines. A dynamic programming approach is used. Sparling (1998) also investigates the multiple U-line problems and presents several heuristic approaches to solve the *N* U-line facility problem. More complex U-lines, which are not a single or simple U-line, are named *multi-lines in a single U*, *double-dependent U-lines*, *embedded U-lines, figure-eight-pattern U-lines*, and *multi-U-line facility*. Hardly are travel time between tasks considered; however, at present only both of Miltenburg (2001a) and Shewchuk (2008) considered walking time. Miltenburg (2001a)'s 10-task problem of a single U-line was studied hierarchically in USALBP-1. It gives us the optimal number of workstations with walking distance (one unit for adjacent machines at the same row and two units for opposite machines). Shewchuk (2008) studied the same problem of 5-20 machines with walking time (one second for adjacent machines at the same row and two units for opposite machines). They are the same constraint that is assumed for the following experiments in this research. However, Shewchuk (2008) did not refer to the input of precedence graph. As a result, its optimum number of workstations with walking time cannot be compared. This research relaxes Shewchuk's assumption that not guarantee minimum walking times in the paper (Shewchuk, 2008, p.3,489).

2.4.2 U-shaped mixed-model assembly line balancing

The characteristics of modern assembly lines found in many assembly operations today (Bukchin *et al.*, 2002). The first mixed-model U-line balancing problem (MUALBP) was addressed by Sparling and Miltenburg (1998). They adapt the four-step mixed-model straight-line procedure of Thomopoulos (1967, 1970) and set the initial balance using a branch and bound algorithm. A smoothing algorithm using a search procedure is then used to reduce the imbalance of the line for a given sequence of models. Characteristics of mixed-model U-shaped assembly lines were also described in Miltenburg (2002) and Kara *et al.* (2007). Kim *et al.* (2000) apply genetic algorithms to the mixed-model, U-shaped line balancing and sequencing problem. All three genetic algorithm representations developed by the authors generated better results than traditional hierarchical approaches. In conclusion, the single-model and mixed-model U-shaped assembly line balancing problems are interesting in both a practical and a theoretical viewpoint.

2.4.3 Data sets of U-shaped Assembly Line Balancing Problems-I (UALBP-I)

The well-known Talbot data set is based on 12 precedence networks with 8-111 tasks, each of which is combined with several cycle times to build a total of 64 instances (Talbot *et al.* 1986). Miltenburg (1998) noted that U-line problem sets with more than twenty-six tasks may be too difficult to solve in more restricted constraints. However, The Scholl data sets are composed of 168 instances with 25-297 tasks (Scholl, 1999). All instances form the combined data set with 269 instances. Complete descriptions of all data sets are given in Scholl (1999, chapter 7.2) and can be downloaded from the web at 'http://www.bwl.tu-darmstadt.de/bwl3/forsch/projekte/alb/index.htm' or 'http://www.assembly-line-balancing.de'. These sets are used for testing ULINO which is applied directly to the UALBP of type I.

2.5 Literature Survey on the Worker Allocation Problems

2.5.1 Early work on worker allocation problems

Assembly line balancing (ALB) problem has been widely studied and strongly reviewed by Song *et al.* (2006). Most of the previous line balancing approaches attempted to solve the same problem, which is defined as how to assign the tasks to an ordered sequence of stations so that the precedence relations should satisfied and some measures of effectiveness should be optimized. However, the worker factors are seldom considered in solving the ALB problem. It is widely ignored that in the real life situations of labor intensive industry, such as apparel manufacturing, even with the optimal task sequence employed, and minimized idle time (or cycle time) obtained, the production line still cannot be balanced in most cases because of the efficiency variance among workers and uncertain efficiency of the same worker in different situations. The worker efficiency is revealed to be greatly influenced by such factors as worker's emotion, motivation, health, skill level and experience of doing the similar operation previously (Kannan and Jensen, 2004)

Based on the fact that the variance of worker efficiency leads to production line imbalance in those industries that still heavily rely on labor skills, the problem to balance assembly production line optimally with the consideration of worker efficiency variance in thus raised. If a single worker can handle multiple machines, the line balancing problem can be replaced with the 'worker allocation' problem, where the goal is to determine which grouped machines are handled by each worker (Shewchuk, 2008). As right worker allocation, that is to allocate workers to operations so that each operation can have the same efficiency, is vital to keep line-balancing, this study proposes an optimization solution to solve the above problem by obtaining an optimal worker allocation before production based on predicted worker efficiency.

Most scheduling models assume that the number of workers available is equal to the number of machines in a production line, especially on the progressive bundle system. However, in practice on the modular production system there are fewer workers than machines. Chen (1991) addressed problems where a worker handles more than one machine. He formulated the problem of sequencing the operations performed by workers in a cell as a Mixed Integer Program (MIP) to find a cyclic worker walking pattern corresponding to minimum makespan.

Much of the existing literature solves the worker allocation problem by mathematical programming by assuming both deterministic data and single objective. Vembu and Srinivasan (1997) incorporated principles of JIT production to the combined problem of worker allocation and sequencing batches of jobs on the single goal of minimizing the makespan. Existing literature invariably solves the worker allocation problem using mathematical programming that uses deterministic data and a simplified objective. Bhaskar and Srinivasan (1997) developed a MIP formulation to solve a worker allocation problem for cellular manufacturing systems. Its objective was to balance the workload among cells, and to minimize the production make-span. They did not address the detailed worker allocation decision for the different stages within a cell. Ghinato *et al.* (1997) developed the Gray code transition sequences method based on the generation of all possible solutions that initially satisfy only the zone constraint to obtain the optimal assignment for problems with two workers and ten machines. The optimal solution satisfies three goals step by step. First, cycle time is minimized and then the reduction of the absolute deviation of workload and absolute deviation of routine time are subsequently realized. The solution method was applied to three groups of problems with different configurations. Optimal assignments were obtained and the results briefly discussed. It is likely to develop the present work (2 workers ×10 machines) to any dimension (n workers × m machines). Some traditional methods such as branch-and-bound and some metaheuristic approaches such as genetic algorithms in such a problem will be extended. Nakade and Ohno (1999) considered an optimization problem of finding an allocation of workers at a U-shaped production line with multi-function workers to minimize the cycle time under the deterministic processing and walking times assumptions. Ertay and Ruan (2005) proposed a decision-making approach based on data envelopment analysis (DEA) for determining the most efficient number of workers in U-shaped cellular manufacturing system. It evaluated the performances of all decision-making units (DMUs) by using simulation and only one line was considered. All of the above literature assumes a homogeneous skill in solving the worker allocation problem. In other words, the difference in skill set is not considered. This study identifies the minimum of workers required in the assembly line to obtain maximum output. Having seized maximum output, this number is different for different job varieties in the same line. Miralles *et al.* (2008) studied Assembly Line Worker Assignment and Balancing Problem (ALWABP) providing a simultaneous solution to a double assignment: (1) tasks to stations; and (2) available workers to stations. After defining the mathematical model that aims to minimize the cycle time for this problem, a basic Branch and Bound approach with three possible search strategies and different parameters is presented. They also propose the use of a Branch and Bound-based heuristic for large problems and analyze the behavior of both exact and heuristic methods through experimental studies. Finally the implementation of these procedures in a Shelters Work centre for Disabled – the real environment which has inspired this research – is described. At last, the study of Shewchuk (2008) differs from the widely-investigated U-line assembly line balancing problem in that the assignment of tasks to line locations is fixed. This paper address the worker allocation problem for lean U-shaped production lines where the objectives are to minimize the quantity of workers and maximize full work: such allocations provide the opportunity to eliminate the least-utilized worker by improving processes accordingly. A mathematical model is developed: the model allows for any allocation of machines to workers so long as workers do not cross paths. Walking times are considered, where workers follow circular paths and walk around other worker(s) on the line if necessary. A heuristic algorithm for tackling the problem is developed, along with a procedure representing the 'traditional' approach of constructing standard operations routines. Computational experiments considering three line sizes (up to 20 machines) and three takt time levels are performed. The results show that the proposed algorithm both improves upon the traditional approach and is more likely to provide optimal solutions.

2.5.2 Worker allocation in mixed-model assembly lines

Until now, rarely is the study of worker allocation in *mixed-model* or *U-shaped mixed-model* assembly lines found. The paper of Shewchuk (2008) is closest to the issue since lean manufacturing is imperatively relevant to mixed-model products. The gap of the approach that is not guarantee minimum walking times in the paper is fulfilled in this research.

2.5.3 Worker allocation objective functions

In the study of decision making, terms such as *multiple objectives*, *multiple attributes*, and *multiple criteria* are often used interchangeably. Multiple objectives decision making (MODM) consists of a set of conflicting goals that cannot be achieved simultaneously. The motivation to consider the problem of generating the efficient set of the worker allocation problem comes from the variety of industrial cases where the criteria related to minimum number of workstations, smoothed workload in a sense of equity and minimum walking time to save the space needed for the actual size of a U-shaped line and shorten distance for communication between workers.

Finally, their interesting study is likely to complete more than a single objective function and contribute the gaps of theoretical and practical U-shaped assembly line worker allocation problems.

2.5.3.1 Comparison of objective functions for the UALBP

Historical single and multiple objective functions have been studied by several researchers. Efficiency and balance performance measures that are two of three measures influencing to achieve just-in-time manufacturing (especially in UALBP) in the previous Figure are reviewed and shown in Figure 2.4.

2.6 Solution Approaches

2.6.1 Importance of exact algorithms

Exact searches have been known to be one of the most efficient approaches in solving optimization problems as they can guarantee finding the optimal solution satisfying all constraints and optimizing the objective value, if one exists. On the other hand, they can prove as well the non-existence of a feasible solution. The mathematical programming formulation of worker allocation problems can be provided in some papers (Miltenburg, 1998; Kuo and Yang, 2007; Miralles *et al.*, 2008). Generally they follow common approaches that are dynamic programming, mixed integer programming, and branch and bound approaches. However, it is well-known that optimal solutions can be found for only relatively small size problems.

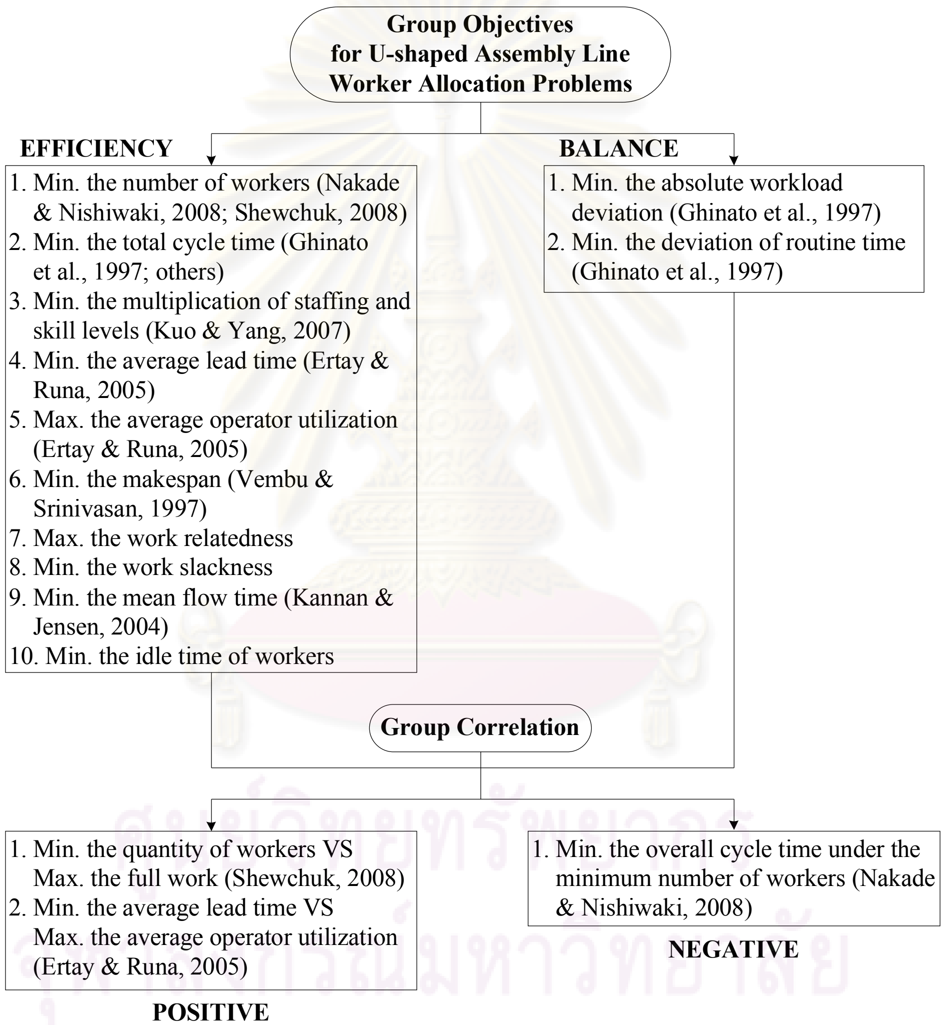


Figure 2.4 Comparison of objective functions for the UALBP

2.6.2 Importance of approximation algorithms

When facing NP-hard problems, metaheuristics (MH) and approximate searches are often proposed to quickly obtain near-optimal solutions in stead of seeking an optimal solution. This is particularly relevant in the context where the problem is subject to frequent disturbances (the best solution is less important because it will not remain optimal or even valid for a long time). MH methods include techniques such as simulated annealing, tabu search, guided local search, or else genetic algorithms and propose an approach where a heuristic criterion (typically the objective function) is used for guiding the search process through the search space (the set of the possible tasks' allocations). Their search paradigm is based on an iterative process where we start from an initial feasible solution and makes incremental changes by modifying the current tasks' allocation at each iteration using the objective function to guide this process towards better a set of solutions. In the case of population-based MH such as genetic algorithms, the search process maintains a population of solutions throughout the search process instead of a single solution and follows a similar iterative improvement process by applying genetic operators: crossover, mutation and selection. The necessary interface needed by MH to support solution modifications includes an insertion operation, which inserts a task before a specified activity (tour construction) and a swap operation, which exchanges two tasks (tour improvement). From these operations, more complex moves can be derived to help guiding the search more efficiently (Voudouris *et al.*, 2008).

2.6.2.1 Complexity

It is excerpted from Gen *et al.* (2008) that computational complexity theory is the study of the complexity of problems – that is, the difficulty of solving them. Problems can be classified by complexity class according to the time it takes for an algorithm to solve them as function of the problem size. For example, the traveling salesman problem can be solved in time $O(n^2 2^n)$ (where n is the size of the network to visit). Even though a problem may be solvable computationally in principle, in actual practice it may not be that simple. These problems might require large amounts of time or an inordinate amount of space. Computational complexity may be approached from many different aspects. Computational complexity can be investigated on the basis of time, memory or other resources used to solve the problem. Time and space are two of the most important and popular considerations when problems of complexity are analyzed. The time complexity of a problem is the number of steps that it takes to solve an instance of the problem as a function of the size of the input (usually measured in bits), using the most efficient algorithm. To understand this intuitively, consider the example of an instance that is n bits long that can be solved in n^2 steps. In this example we say the problem has a time complexity of n^2. Of course, the exact number of steps will depend on exactly what machine or language is being used. To avoid that problem, the Big O notation is generally used. If a problem has time complexity $O(n^2)$ on one typical computer, then it will also have complexity $O(n^2)$ on most other computers, so this notation allows us to generalize away from the details of a particular computer.

2.6.3 Common solutions for U-shaped assembly line balancing problems

Using exact solution for small size problems of U-shaped assembly line balancing are studied in some papers. It is well known that traditional assembly line balancing problems (ALBP) fall into the NP-hard class of combinatorial optimization problems (Hwang *et al.*, 2008). Since both the MALBP and the UALBP are subsets of the ALBP, they are also NP-hard. Therefore, methods that evaluate the entire solution space are not suitable for large sized problems and heuristics need to be employed in order to efficiently search the solution space. However, heuristics may become trapped at a local minimum as noted by Sparling and Miltenburg (1998).

2.6.3.1 Solution methods

Since the ALB model was first formulated by Helgeson *et al.* (1954), many solution approaches have been proposed. Several optimum seeking methods, such as *linear programming* (Salveson, 1955), *integer programming* (Bowman, 1960), *dynamic programming* (Held *et al.*, 1963) and *branch-and-bound approaches* (Jackson, 1956) have been employed to deal with ALB. However, none of these methods has proven to be of practical use for large problems due to their computational inefficiency. Since ALB models fall into the NP-hard class of combinatorial optimization problems (Karp, 1972), in recent years, to provide an alternative to traditional optimization techniques, numerous research efforts have been directed towards the development of heuristics Dar-El (1973) and meta-heuristics. While heuristic methods generating one or more feasible solutions were mostly developed until the mid 1990s, meta-heuristics such as tabu search (Scholl, 1966), simulated annealing (Suresh and Sahu, 1994), genetic algorithms (Falkenauer and Delchambre, 1992) and ant colony optimization (Bautista and Pereira, 2002) have been the focus of researchers in the last decade. For more information, the reader can refer to several review studies, *i.e.* Baybars (1986a) that survey the exact (optimal) methods, Talbot *et al.* (1986) that compare and evaluate the heuristic methods developed, Ghosh and Gagnon (1989) that present a comprehensive review and analysis of the different methods for design, balancing and scheduling of assembly systems, Erel and Sarin (1998) that present a comprehensive review of the procedures for SMALB, MALB and MUALB models, Rekiek *et al.* (2002) that focus on optimization methods for the line balancing and resource planning steps of assembly line design, Scholl and Becker (2006) that present a review and analysis of exact and heuristic solution procedures for SALB, Becker and Scholl (2006) that present a survey on problems and methods for GALB with features such as cost/profit oriented objectives, equipment selection/process alternatives, parallel stations/tasks, U-shaped line layout, assignment restrictions, stochastic task processing times and mixed model assembly lines, Rekiek and Delchambre (2006) that focus on solutions methods for solving SALB, and Ozmehmet Tasan and Tunali (2007) that present a comprehensive review of GAs approaches used for solving various ALB models. Among the meta-heuristics, the application of genetic algorithms (GAs) received considerable attention from the researchers, since it provides an alternative to traditional optimization techniques by using directed random searches to locate optimum solutions in complex landscapes and it is also proven to be effective in various combinatorial optimization problems. GAs are powerful and broadly applicable stochastic search and optimization techniques based on principles from evolutionary theory (Gen and Cheng, 2000). Falkenauer and Delchambre (1992) were the first to solve ALB with GAs. Following Falkenauer and Delchambre (1992), *application of GAs for solving ALB models was studied by many researchers*, *e.g.*, Kim *et al.* (1996b); (Leu *et al.*

(1994); Noorul Haq *et al.* (2006). However, most of the researchers focused on the simplest version of the problem, with single objective and ignored the recent trends, *i.e.* mixed-model production, U-shaped lines, and robotic lines, in the complex assembly environments, where ALB models are multi-objective in nature Ozmehmet Tasan and Tunali (2007). In the following section, multi-objective evolutionary algorithms are described.

<u>Another rational support of GAs acquisition</u>

A number of attempts have been made to apply Genetic Algorithms to other problems such as traveling salesman problem, production planning and scheduling, facility location problems, and cell design problems (Venugopal and Narendran, 1992; Gupta *et al.*, 1993).

Combinatorial optimization is the process of finding one or more best (optimal) solutions in a well defined discrete problem space. For a small size problem, a branch-and-bound approach is often the most efficient way to solve them. This research seems to use Ant Colony Optimization (ACO) for the solution of the travel path problem, but not taking good at the sub colony. This research is more suitable for the sub-structure problem making facilitate to change into bit strings.

2.6.3.2 Multi-objective evolutionary algorithms

Since the 1950's, some authors have been using concepts based on Darwin's evolution theory for the solution of optimization problems (Box, 1957; Friedberg, 1958; Bremermann, 1962). Numerous algorithms based on the same concepts have been developed over the last 30 years. They are usually described by the term "evolutionary computation methods." The most notable members of this group are simple genetic algorithms (GA's) (Holland, 1975; Goldberg, 1989), evolution strategies (Rechenberg, 1973), evolutionary programming (Fogel, 1996), classifier systems (Booker *et al.*, 1989), and genetic programming (Koza, 1992). Bäck *et al.* (1992) give an excellent review of evolutionary computation methods, and highlight some recent developments in the field.

Multi-objective optimization problem

In most cases, a multi-objective optimization problem (MOP) can be described, without loss of generality, by using the following formulation:

$$\underset{x\in\Omega}{Minimize} \quad f_1(x), f_2(x), \ldots, f_k(x) \tag{2.1}$$

where solution x is a vector of decision variable for the considered problem, Ω is the feasible solution space, and $f_i(\cdot)$ is the i^{th} objective function (for $i = 1,2,...,k$). Usually, there is no single optimal solution for equation 2.1, but rather a set of alternative solutions. These solutions are optimal in the wider sense such that no other solutions in the search space are superior to them when all objectives are considered. A decision vector x is said dominate a decision vector y (also written as $x \succ y$) if and only if:

$$f_i(x) \le f_i(y) \quad \text{for all } i \in \{1,2,\ldots,k\}; \tag{2.2}$$

$$f_i(x) < f_i(y) \quad \text{for at least one } i \in \{1,2,\ldots,k\} \tag{2.3}$$

All decision vectors which are not dominated by any other decision vector are called non-dominated or Pareto optimal. Figure 2.5 illustrates the non-dominated solutions for a two-objective minimization problem.

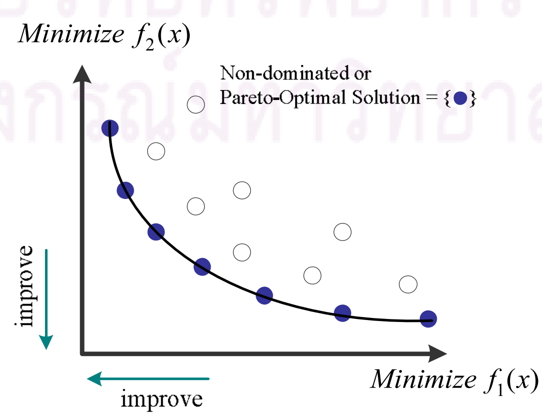


Figure 2.5 Non-dominated or Pareto-optimal solutions

Evolutionary algorithm in multi-objective optimization

Multi-objective evolutionary algorithms (MOEAs) have become popular and have been applied to a wide range of problems from social to engineering problems (Coello *et al.*, 2002). In general, MOEAs are ideally suited to the multi-objective problem because they are capable of searching multiple Pareto-optimal solutions in a single run. The approximation of Pareto-optimal set involves two conflicting objectives: (1) the distance to the true Pareto front is to be minimized; whereas (2) the diversity of the evolved solutions is to be maximized (Zitzler *et al.*, 2001). To achieve the first objective, a Pareto-based fitness assignment is normally designed to guide the search toward the true Pareto optimal front (Fonseca and Fleming, 1993). In the view of the second objective, some MOEAs successfully provide density estimation methods to preserve the population diversity. Since last two decades there are many MOEAs that were strongly reviewed by Chutima and Pinkoompee (2008) as follows: Vector Evaluated Genetic Algorithm (VEGA), Multi-Objective Algorithm (MOGA) (Fonseca and Fleming, 1993), Niched-Pareto Genetic Algorithm (NPGA) (Horn *et al.*, 1994), Non-dominated Sorting Genetic Algorithm (NSGA) (Srinivas and Deb, 1994), Pareto Stratum-Niche Cubicle Genetic Algorithm (Hyun *et al.*, 1998), Strength Pareto Evolutionary Strategy (SPEA), Pareto-Archived Evolutionary Strategy (PAES), Niched-Pareto Genetic Algorithm II (NPGA-II), Strength Pareto Evolutionary Algorithm 2 (SPEA 2), Non-dominated Sorting Genetic Algorithm II (NSGA-II) (Deb *et al.*, 2002), Rank Density Genetic Algorithm (RDGA) (Lu and Yen, 2003), and Memetic Algorithm (MA) (Kumar and Singh, 2007; Ishibushi, 2003).

This research studies and bases on four evolutionary algorithms, that is, Non-dominated Sorting Genetic Algorithm-II (NSGA-II), Memetic Algorithm (MA), COINcidence Algorithm (COIN), and Particle Swarm Optimization (PSO) as follows.

I. NSGA-II

Non-dominated sorting genetic algorithm II (NSGA-II) is one of the most popular genetic algorithms in recent years. It has the ability to find multiple Pareto-optimal solutions in one single run. In NSGA-II, the population is sorted according to the level of non-domination. The diversity among non-dominated solutions is maintained using a measure of density of solution in the neighborhood. NSGA-II is able to find much better widespread solutions and better convergence near the true pareto-optimal frontier in most problems. The steps and flowchart of NSGA-II is illustrated in the next section.

II. MA

Memetic Algorithm (MA) is a type of Evolutionary Algorithms (EAs) that applies a separate local search algorithm to refine individuals. These methods are inspired by models of adaptation in nature systems that combine evolutionary adaptation of populations of individuals with individual learning with a lifetime. Additionally, MA (hybrid EAs) uses EAs to perform exploration and use local search to exercise exploitation. Combining with local search is a strategy used by many successful global optimization approaches, and MA has been recognized as a powerful algorithmic paradigm for evolutionary computing. In particular, the relative advantage of MA over EAs is their ability to be more consistent on complex search spaces.

In the framework of this research, the basic concept of MA and MOEAs are used to enhance the performance of the original MOEAs as NSGA-II by combining them with local search. This study has not tried to specify an appropriate local search direction to all obtained solutions since it is time consuming. However, if the solutions of which the local search is applied are randomly selected, the improved quality of the new solutions may not be guaranteed. Therefore, an appropriate solution is selected by using binary tournament selection. Furthermore, we use the first improvement which is an execution of local search that is terminated when no better solution is found among k neighbors randomly generated from the current solution, where k is a user-definable parameter. This stopping criterion of local search, called early termination strategy, can help decease the computation time spent by local search. The other strategy for decreasing the computation time is the reduction in the number of neighboring solutions. The local search probability P_L indicates the opportunity that the local search is applied. In our MA, a suitable local search that can be used to improve the efficiency of NSGA_II is searched. Seven local search procedures are evaluated in the part of local search heuristics below. However, the steps and flowchart of MA are illustrated in the next section.

Initial population

A set of N chromosomes is generated randomly as an initial set of populations. The chromosome is represented by a sequence of genes (tasks). The position of gene in a sequence of the chromosome represents the task in the sequence.

Local search heuristics

Once a new chromosome is created in the initial population, it is improved by using local search. In fact, local search can be performed after obtaining initial solutions, new offspring and mutation. In this research, it is determined after obtaining initial solutions and mutation since our pilot runs indicated that these two points were enough for our MA to find significantly improved solutions, pull the solutions out of the local optimal, and reduce computational time. The local searches in this research are modified from Kumar and Singh (2007) that also focuses on TSP including the followings:

- Pairwise Interchange (PI): Select two arbitrary products located at positions i and j, $i \neq j$, and interchange them. It consists of all possible swaps of pairs of products in a given solution;

- Insertion Procedure or Shift Procedure (IP or SH): Remove a product from one position and insert it back to any position of the sequence;

- Adjacent Pairwise Interchange (API): This is a special case of the PI where two products located at positions i and i+1

$(1 \le i \le n-1)$ are interchanged to generate a neighboring solution;

- 2-opt: A neighboring solution is obtained by selecting two arbitrary products *i* and *j* and interchange them;

- 3-opt: In this case, 3 products are randomly selected and interchange them;

- Or-opt: It considers a smaller percentage of exchange that would be considered by a regular 3-opt by considering only those exchanges that would result in a string of one, two, or three currently adjacent products being insertion between two other products;

- Double-bridge: It cuts the chromosome into 4 segments by deleting four random genes and reinserts them in a different order to create a new chromosome.

Selection

In the proposed algorithm, selection is based on the binary tournament selection to obtain suitable parents to further perform local search and genetic operator. It chooses each parent by choosing two candidates at random and then choosing the best individual out of that set which has lower rank or lower fitness value to be a parent. If two individuals belong to the same rank or same fitness value, the tournament prefers the most isolated one, using a density mechanism.

Operator

a. Crossover

There are other crossover techniques available for general sequencing problems, e.g. partially-mapped crossover (PMX; Goldberg and

Lingle, 1985), cycle crossover (CX; Oliver *et al.*, 1987), order crossover (OX) (Michalewicz, 1996) and immediate successor relation crossover (ISRX; Hyun *et al.*, 1998). However, the two point-based weight mapping crossover (WMX) by Hwang *et al.* (2008) is used in this research.

b. Inversion

Inversion is an operator that generates offspring from a single parent. It first chooses two random cut points in a parent. The elements between the cut points are then reversed.

C. Mutation

Mutation is performed irregularly so that a solution could have an occasional trait that is unique from its parents. This is essentially done so that diversity remains in the population. Mutation for this research is the simple swapping of two unique elements in the sequence of interest.

III. COIN

Wattanapornprom *et al.* (2009) developed a new effective evolutionary algorithm called *combinatorial optimization with coincidence* (COIN) originally aiming to solve traveling salesman problems. Several benchmarks are compared to the experiment of Robles et al. (2002). The idea is that most well-known algorithms such as Genetic Algorithm (GA) search for good solutions by sampling through crossover and mutation tasks without much exploitation of the internal structure of good solution strings. This may not only generate large number of inefficient solutions dissipated over the solution space but also consume long CPU time. In contrast, COIN considers the internal structure of good solution strings and memorizes paths that could lead to good solutions. COIN replaces high computation time of crossover and mutation tasks of GA and employs a joint probability matrix as a means to generate neighborhood solutions. It prioritizes the selection of the paths with higher chances of moving towards good solutions.

Apart from traditional learning from good solutions, COIN allows learning from below average solutions as well. Any coincidence found in a situation can be statistically described whether the situation is good or bad. Most traditional algorithms always discard the bad solutions without utilizing any information associated with them. In contrast, COIN learns from the coincidence found in the bad solutions and uses this information to avoid such situations to be recurrent; meanwhile, experiences from good coincidences are also used to construct better solutions in Figure 2.6 (Sirovetnukul and Chutima, 2010b). Consequently, the chances that the paths being parts of the bad solutions are always used in the new generations are lessened. This lowers the number of solutions to be considered and hence increases the convergence speed.

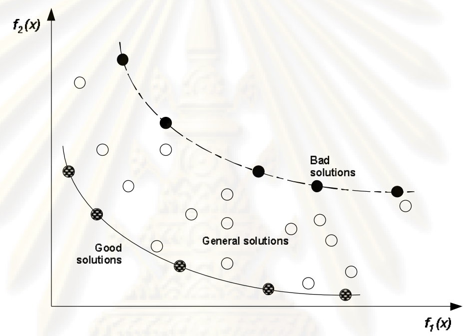


Figure 2.6 Good and bad solutions

COIN uses a joint probability matrix (generator) to create the population. The generator is initialized so that it can generate a random tree with equal probability for any configuration. The population is evaluated in the same way as traditional evolutionary algorithms. However, COIN uses both good and bad solutions to update the generator. Initially, COIN searches from a fully connected tree and then incrementally strengthening or weakening the connections. As generations pass by, the probabilities of selecting certain paths are increased or decreased depending on the incidences found in the good or bad solutions. The algorithm of COIN can be stated as follows.

1. To initialize the joint probability matrix (generator);

2. To generate the population using the generator;

3. To evaluate the population;

4. To make diversity preservation;

5. To select the candidates according to two options: (a) good solution selection (select the solutions in the first rank of the current Pareto frontier), and (b) bad solution selection (select the solutions in the last rank of the current Pareto frontier);

6. For each joint probability matrix $H(x_i, x_j)$, to adjust the generator according to the reward and punishment scheme as Eq. (2.4);

$$x_{i,j}(t+1) = x_{i,j}(t) + \frac{k}{(n-1-np_i)}\{r_{i,j}(t+1) - p_{i,j}(t+1)\} + \frac{k}{(n-1-np_i)^2}\{\textstyle\sum_{j=1}^{n} p_{i,j}(t+1) - \sum_{j=1}^{n} r_{i,j}(t+1)\} \quad (2.4)$$

where $x_{i,j}$ = the element (i, j) of joint probability matrix $H(x_i / x_j)$, k = the learning coefficient, $r_{i,j}$ = the number of coincidences (x_i, x_j) found in the good solutions, $p_{i,j}$ = the number of coincidences (x_i, x_j) found in the bad solutions, t = generation number, n = the size of the problem, and np_i = the number of the direct predecessors of task i;

7. To apply a strategy to maintain elitist solutions in the population, and then repeat step 2 until the terminating condition is met.

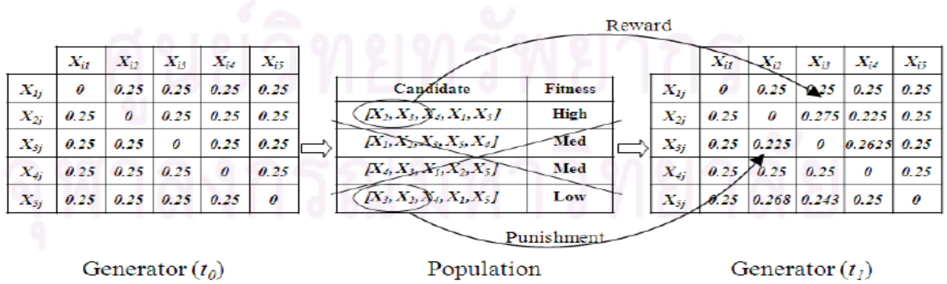


Figure 2.7 Updating the generator

According to Figure 2.7, it illustrates the process of initializing the generator, generating the first population, selection of good and not good candidates and finally updating the generator using the selected candidates. The generator is initialized so that each node of the dependency is equally to 0.25. The population is generated from the initiated generator. The candidates are sorted and classified into three classes: high fitness, medium fitness, and low fitness. The high fitness candidates are considered to be the good solutions while the low fitness candidates are taken into account to be the not-good solutions in the population.

The COIN, which is the very up-to-date algorithm, is not studied into the worker allocation problem of real world industrial application, however. It is interesting to modify the single-objective COIN algorithm to the multi-objective COIN algorithm (Sirovetnukul and Chutima, 2010b) in the following experiments. The flowchart of the modified COIN, named the multi-objective coincidence algorithm, is also shown in the next section.

IV. PSO

The renowned evolutionary combinatorial optimization, named particle swarm optimization (PSO), was developed by Kennedy and Eberhart in 1995. PSO is motivated by social behavior of birds flocking or fish schooling Solutions are represented by particles in the search space. Each of the particles keeps the path of the best solution as the *local* best (*l*best). A swarm of particles are identified to the best location named the *global* best (*g*best). The next move of particles is navigated by the *l*best and *g*best. To give an overview of directions and applications, a snapshot of the PSO technique is also reviewed and described extensively (Poli *et al.*, 2007). PSO can be used across a large number of applications such as combinatorial optimization problems (Salman *et al.*, 2002; Tseng and Liao, 2008).

Similar to NSGA-II, the decision of most optimization problems is relevant to conflicts between multiple criteria in practice. Thus, a set of solutions for multiple objectives are obtained as non-dominated solutions or a Pareto-optimal frontier. Although a comprehensive review of the various Multi-Objective PSO (MOPSO) papers is reported in Reyes-Sierria and Coello (2006), MOPSO is extended to the modified PSO, name Particle Swarm Optimization with Negative Knowledge (PSONK). The steps and the flowchart of PSONK are shown in the following section.

2.6.3.3 Heuristic rules

Mantazeri and Van Wassenhove (1990) found that no single heuristic is the best of all the possible performance measures. Often a combination of basic dispatching rules can perform significantly better. A lot of heuristic approaches can be found in the literature to solve the simple and U-shaped line balancing problems. However, there are a few papers for the U-shaped line balancing problem using heuristics until now. Martinez and Duff (2004) proposed ten heuristic rules used to find solutions to the U-shaped line balancing problem of type I. All these heuristic rules were previously used to solve the simple line balancing problem, but some modifications were made. The difference between the original versions and the modified versions is that tasks are available for assignment to a work station by having all successors or all predecessors previously assigned to a work station, and when solving for the simple LBP, tasks are available for assignment by having all successors previously assigned only. The first heuristic rule is the Modified Ranked Positional Weight procedure posted by Miltenburg and Wijngaard (1994). The other nine heuristics which are introduced in this research for solving the U-shaped LBP are: 2. Maximum Total Number of Follower Tasks or Precedence Tasks, 3. Minimum Total Number of Follower Tasks or Precedence Tasks, 4. Maximum Task Time, 5. Minimum Task Time, 6. Maximum Number of Immediate Followers or Immediate Precedence Tasks, 7. Minimum Number of Immediate Followers or Immediate precedence Tasks, 8. Minimum U-line Upper Bound, 9. Minimum U-line Lower Bound, 10. U-line Minimum Slack. The description of ten heuristic rules and formulations are explained in Martinez and Duff (2004, pp.288-289). The results showed that some very simple heuristic rules produced optimal or near optimal solutions.

In addition to the minimum number of immediate followers or immediate precedence tasks, nine of all rules in Martinez and Duff (2004) and other six task assignment rules are used in Baykasoglu (2006). They consists of Random Priority, Smallest Task Number, Greatest Average Ranked Positional Weight, Smallest (Upper Bound Divided by the Number of Successors) and Greatest (Processing Time Divided by the Upper Bound), and Greatest Number of Immediate Predecessors. In other words, Baykasoglu (2006)'s heuristic rules cover Mertinez and

Duff (2004)'s nine rules except for the minimum number of immediate followers or immediate precedence tasks.

Finally, the existing heuristic rules are used to approach optimal or near optimal solutions as much as possible. It is not essential to do experiments for all existing heuristic rules to all problem sets. After dealing with one problem set, the best heuristic rule will be representative for the rest of problem sets. It means that feasible experimental subsets are reduced from doing total complete enumeration.

2.6.3.4 Performance measures

In this study, three performance measures are used to achieve two goals of a multi-objective optimization: (1) convergence to the Pareto-optimal set, and (2) maintenance of diversity in the solutions of Pareto-optimal set. In Eq. (2.5), the convergence of the obtained Pareto-optimal solution towards a true Pareto-set (A^*) is the difference between the obtained solution set and true Pareto set. Mathematically, it is defined as follows.

$$convergence(A) = \frac{\sum_{i=1}^{|A^*|} d_i}{|A^*|} \tag{2.5}$$

$$d_i = \min_{j=1}^{|A^*|} \sqrt{\sum_{k=1}^{K} \left(\frac{f_k(x) - f_k(y)}{f_k^{\max} - f_k^{\min}} \right)^2} \tag{2.6}$$

In Eq. (2.6), $f_k^{\max}$ and $f_k^{\min}$ are maximum and minimum values of k^{th} objective function in the true-Pareto set respectively. For this measure, lower value indicates superiority of the solution set. When all solutions converge to Pareto-optimal front, this metric is zero indicating that the obtained solution set has all solutions in the true Pareto set. The true Pareto-optimal solution is obtained from Non-dominated solutions from the combination of all three algorithms (NSGA-II, MA, and COIN).

The second measure is a spread metric. This measure computes distribution of obtained Pareto-solution by calculating a relative distance between consecutive solutions as shown in Eq. (2.7)

$$spread(A) = \frac{d_f + d_l + \sum_{i=1}^{|A|-1} \left| d_i - \bar{d} \right|}{d_f + d_l + (|A|-1)\bar{d}} \tag{2.7}$$

where the parameters d_f and d_l are Euclidean distances between the extreme solutions and boundary solutions of the obtained Pareto-optimal. The value of this measure is zero for a uniform distribution, but it can be more than 1 when bad distribution is found.

Additionally, the third measure in Eq. (2.8) is the ratio of non-dominated solutions which indicates the coverage of one set over another. Let A_j be a solution sets $(j = 1, 2, ..., J)$. For comparing each J solution sets $(A = A_1 \cup A_2 ... \cup A_J)$, the ratio of non-dominated measure of the solution set A_J with respect to the J solution sets is the ratio of solutions in A_J that are not dominated by any other solutions in A, which is defined as follows.

$$R_{NDS}(A_j) = \frac{|A_j - \{x \in A_j \mid \exists y \in A : y \prec x\}|}{|A_j|} \tag{2.8}$$

where $y \prec x$ means the obtained solution x is dominated by the true-Pareto solution y. The higher ratio indicates superiority of one solution set over the other.

2.6.3.5 Comparison of objective functions

The correlation of objective functions is classified into positive and negative slopes. According to Figure 2.8, there are four types of correlation, that is, Min-Max, Max-Min, Min-Min, and Max-Max.

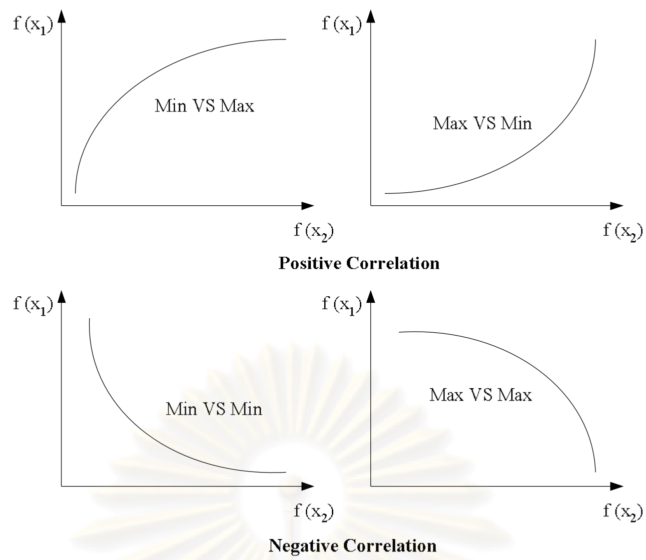


Figure 2.8 Correlation of objective functions

2.7 Comparison of the Related Research

According to the previous section of literature review, scheme of relevant works for worker allocation problems in U-shaped mixed-model assembly lines are summarized in Table 2.1. To gain more benefits and modification in addition to Erel and Sabuncuoglu (2001), summary of the work conducted on U-shaped assembly lines of type I is illustrated in Table 2.2. All of them study the single objective, but the problem decomposition is solved if multiple objectives are decided. No multi-objective solutions to this research problem are found at this point. Finally, this research study conducts the problem specification of single and mixed-model products in several single U-shaped assembly lines, problem sets of 7-297 tasks, multiple objective functions with the consideration of walking time, and multi-objective evolutionary algorithm approached.

2.8 Limitations of the Existing Research

The literature reviewed at this point was accomplished by searching some papers. Based on this search the following conclusions can be drawn as follows:

- There has been no prior documented work in worker allocation in U-shaped assembly lines of type I in the consideration of multiple objectives at present;

- Most of the published paper works for worker allocation problems do not focus on exact solutions, but evolutionary algorithms.

Table 2.1 Scheme of relevant works for worker allocation problems

Author(s)	**Year**	**Research Problem**	**Other Related Resource Constraints / Parameter(s)**	**Optimization Objective(s)**	**Solution Method(s)**
Hwang & Katayama	2009	Workload balancing in mixed-model U-shaped assembly line systems	1. Precedence constraints 2. Some restrictions	Min. the number of workstations & Min. the variation of workload	Genetic approach
Miralles *et al.*	2008	Double assignment: tasks to stations and available workers to stations	1. Assigned tasks for only one worker in every station 2. More than one task for each worker 3. Some assumptions e.g., deterministic processing times and precedence relationships, serial paced line, no buffers, specific disabled workers limitations, and so on	Min. the cycle time	Branch and bound procedures
Nakade & Nishiwaki	2008	Optimal allocation of heterogeneous workers in U-shaped production line	1. Multiple heterogeneous multi-function workers 2. All processing, operation and walking times are deterministic	Min. the overall cycle time under the minimum number of workers	Proposed algorithm
Shewchuk	2008	Worker allocation in lean U-shaped production lines	1. Several constraints 2. Not guarantee minimum walking time	Min. the quantity of workers & Max. full work	Developed heuristic algorithm
Kuo & Yang	2007	Mixed-skill multi-line worker allocation problem for cellular manufacturing systems in an anonymous TFT-LCD manufacturing company	1. the total number of allocated skill category 2. the maximum number of workers assigned to a workstation 3. the line throughput of each product 4. some other constraints	Min. the multiplication of staffing and skill levels	Mixed integer programming
Ertay & Ruan	2005	Labor assignment in U-shaped cellular manufacturing system	Different scenarios (Multi-criteria decision making) related to: - number of workers - transfer batch size - demand level	Min. the average lead time & Max. the average worker utilization	Simulation modelling
Miltenburg	2002	Assignment of tasks to stations and the selection of models sequencing simultaneously on mixed-model U-shaped production lines	1. Several constraints 2. Some assumptions	Min. the variation of work content in the stations & Min. the variation of production for models and parts	Mathematical model and Genetic algorithm
Miltenburg	2001b	U-shaped production lines: A review of the theory and practice	1. U-line layout and operations 2. Experiences of manufacturing companies with U-lines	-	-
Miltenburg	1998	Balancing (Task assignment) U-lines in multiple U-line facility	1. Cycle time constraints 2. Precedence constraints 3. Location constraints 4. Station-type constraints	Min. the number of regular, crossover, and multiline stations	Dynamic Programming
Vembu & Srinivasan	1997	Worker allocation and sequencing in product-line-cells with manually operated machines	1. Unconstraints 2. Some assumptions	Min. the makespan	Heuristic, Genetic Algorithm & Enumeration

48

Table 2.2 Summary of the papers conducted on U-shaped assembly lines

Authors	Problem description	Problem set	Objectives	Walking Time	Solutions
Miltenburg and Wijngaard (1994)	Single model	Up to 11 tasks Up to 111 tasks	Number of workstations	No	DP formulation RPWT-based heuristic
Miltenburg and Sparling (1995)	Single model	Up to 40 tasks	Number of workstations	No	DP-based exact algorithm Depth-first and breath-first B&B
Ajenblit and Wainwright (1998)	Single model	Up to 111 tasks	Number of workstations and workload balance	No	Genetic algorithm
Miltenburg (1998)	U-line facility with several individual U-lines	Individual U-lines with up to 22 tasks	Number of workstations and idle time in a single station	No	DP-based exact algorithm
Sparling and Miltenburg (1998)	Mixed model	Up to 25 tasks	Number of workstations	No	Heuristic
Urban (1998)	Single model	Up to 45 tasks	Number of workstations	No	IP formulation
Scholl and Klein (1999)	Single model	Up to 297 tasks	Number of workstations	No	B&B-based heuristic
Miltenburg (2001a)	Single model	10 tasks	Number of workstations and walking time	Yes	ILP and DP formulation*
Shewchuk (2008)	Lean single U-lines	Up to 20 machines	Number of workstations and maximize full work	Yes	Heuristic*
Hwang and Katayama (2009)	Mixed model	19, 61 and 111 tasks	Number of workstations and workload variation	No	Genetic algorithm
This research study	*Single & mixed model with several single U-lines*	*Up to 297 tasks*	*Number of workstations, workload smoothness, and walking time*	*Yes*	*Multi-objective evolutionary algorithms*

* Walking time is set at one unit for adjacent tasks at the same row and two units for opposite tasks.
** Shewchuk [16] did not use the standard problems of precedence constraints and given cycle time.

<u>P.S.</u> Aase and Olson noted in the paper of "Do U-shaped assembly lines really improve labor productivity" that the exact effect of the additional travel is unclear. Therefore, future research addressing the issue of worker travel is recommended.

49

CHAPTER III

RESEARCH METHODOLOGY

3.1 Introduction

The main purpose of this research is to study how assigned workers work equity and they do not overlap inside a U-line and how walking paths, which is necessary non-value added are best established. To achieve this purpose, the research methodology is addressed step by step in this chapter. It explains the problem environment that consists of the inputs of seven to two hundred and ninety-seven standard problems as well as a case study problem, decision variables, and their data sets and lower bounds. Then, the evolutionary optimization process of the U-shaped manual assembly line worker allocation problems – type I is also elaborated.

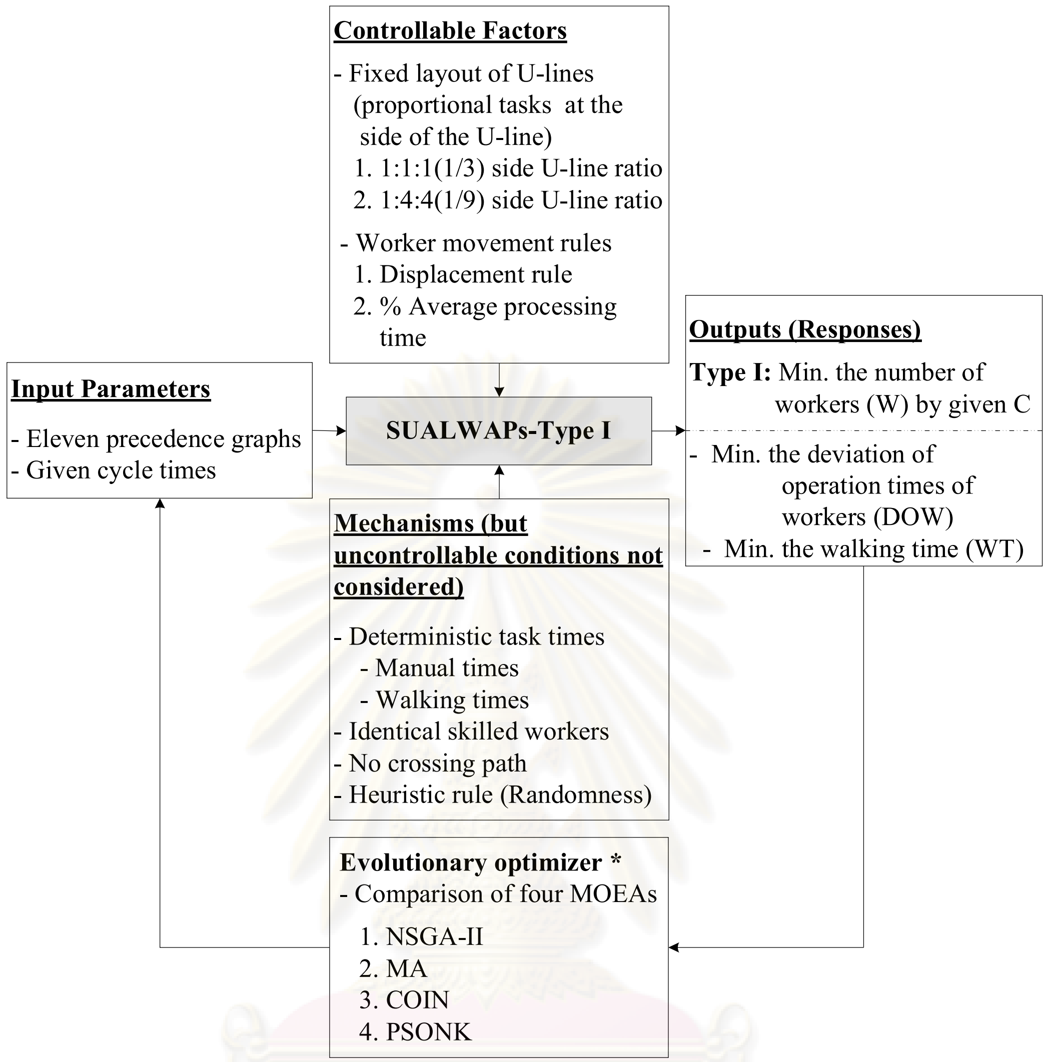
3.2 Research Methodology

After determining the context in which worker allocation is being defined, the methodology for worker-machine assignment is determined. This section addresses the research methodology in the following steps.

1. To study a problem in a setting;
2. To formulate a mathematical model of the problem;
3. To use data sets of existing optimum workers;
4. To develop evolutionary algorithms obtaining a good solution as the final optimum frontier;
5. To conduct computational experiments;
6. To make conclusion, discussion and future research directions.

3.3 Problem Environment

The Just-in-time (JIT) production system has been adopted extensively in today's manufacturing industries such as the apparel industry to meet production demands. A U-shaped production line can be described as a special type of cellular manufacturing used in JIT production systems. In recent decades, many apparel manufactures have installed several production systems on their apparel assembly lines such as the traditional progressive bundle system and the automated unit production system. The assembly line to be studied in this paper is a modular production system (or a single U-line). There are no automated processing machines in the production system. After each worker operates an item at a machine, a worker walks with several patterns such as a circular loop, a rectangular loop, or a straight-line loop and takes it to the next machine and at the end of each intra loop. Generally a worker hands it over to the adjacent worker along the sequence of U-line. From some of the sample companies, there is no equity of workload although line efficiency has been continuously improved. In practice, most companies manage the assembly line problem of type F (given number of workstations and given cycle time) and improve line efficiency by avoiding the complexity of the problem. However, this paper studies the problem of type I: the minimum number of workstations at given cycle times. The evolutionary combinatorial optimization process of Single U-shaped Assembly Line Worker Allocation Problems of type I (SUALWAPs-I) is illustrated in Figure 3.1. Input parameters, Controllable factors, Output responses, and Mechanisms, named ICOM, are detailed in the next section. Then, after the model is validated a multi-criteria optimization technique will be applied to find the set of non-dominated solutions. The criteria that can be considered are: minimum number of workers, minimum deviation of operation times of workers and minimum walking time. Mechanisms are identified into deterministic task times (manual time plus walking time), identical skilled workers, no crossing path (i.e., a worker does not work with any other station at the same time), and random priority rule. Finally, all of evolutionary combinatorial algorithms are computed in the next section.



* To modify the robust algorithms for the best worker allocation in the U-shaped manual assembly line worker allocation problem of type I (UALWAP-I)

Figure 3.1 Evolutionary optimization process for worker allocation problems in the situation of the single U-shaped manual assembly line of type I

3.3.1 Inputs of problem sets

The established benchmark data sets for SALBP are applicable to UALBP which have been used for testing and comparing solution procedures in almost all relevant studies since the early nineties. The well-known test sets of Talbot *et al.* (1986) and Hoffmann (1990) for UALBP-I minimizing the number of workstations for a given cycle time are required as data set. Ajenblit and Wainwright (1998) applied GA to the 12 well-known datasets from the literature of traditional assembly line balancing (Talbot *et al.*, 1986) for the Type I U-shaped assembly line balancing problem. When we considers all possible given cycle times used in their 12 datasets, there are 61 different problems. The 61 problems include six problems from Merten, one problem from Bowman, five problems from Jaeschke, and so forth ending with six problems from Arcus-111. Computational results comparing between ideal (optimal) workstations (N), dynamic programming workstations (N_{DP}) and GA workstations (N_{GA}) are shown in their Table (Ajenblit and Wainwright, 1998; p.100). Moreover, various cycle times for each problem are extended for same task problems in Chiang and Urban (2006, p.1776). Data sets of them are based on the RPWT of Helgeson and Birnie (1961), as modified by Miltenburg and Wijngaard (1994). However, this research uses data sets of Scholl and Klein (1999) because they found that the performance of ULINO was superior to RPWT (Erel *et al.*, 2001, p.3013).

Whenever the computation time is considered, to balance the U-lines it depends on the number of subsets, which depends, in part, on the density and width of the precedence graph. U-lines with dense, narrow precedence graphs were easier to solve than U-lines with sparse, wide precedence graphs (Miltenburg, 1998; p.16). Density is the equal to the number of arcs in precedence graph divided by NT(NT-1)/2 [Note: NT = number of tasks] (ibid).

In this research, the reason is why the problem set of seven tasks is started rather than 19-task, 61-task and 111-task problems by Hwang and Katayama (2009)'s reference because this research first looks down into the single U-shaped assembly line problem more than the reference of the mixed-model U-shaped assembly line problem. Thomopoulos's (1970) 19-task and Kim *et al.* (2006)'s 61-task problems were modified whereby tasks with the same number have the same operation time in each model as shown in the following Table. The original data from Arcus's 111-task problem are used and cited in 1963. Secondly, the number of machines and operators taken from the average Japanese and US U-line over 22 U-line implementations are determined into ten machines and three workers. The minimum and maximum machines are three and thirty-one (Miltenburg, 2001; p.210). One machine should work at least one task and our study assumes that one task is worked only one machine. The three-task problem is interesting, but the literature review of tradition assembly line balancing is originated by seven-task's Merten. As a result, this research focuses on it and ending with Arcus-111. This research does not input the problem of 11-task’s Dar-El because they are the same number of tasks as Jackson. The 9-task’s Bowman and Jaeschke problems are not input due to the closeness of 7-task’s Merten. The 21-task’s Michell problem is not studied due to the closeness of 19-task’s Thomopoulos. The 30-task’s Sawyer problem is not also studied due to the closeness of 28-task’s Heskiaoff. For the problem of Arcus, the 111-task problem inputs place of the 83-task problem. Largest number of tasks is regard as of the 297-task problem by Scholl and Klein (1999). Significantly, 10-task’s Miltenburg is necessary for the original location of a single U-shaped layout. Finally, the 36-task problem of a case study is also studied.

For given cycle times, this research scopes to study only three of each problem with the minimum, middle and maximum values of Scholl and Klein (1999) at the web ‘http://www.assembly-line-balancing.de’. Nevertheless, some data set problems give us only one cycle time.

In this study we take many well-known line balancing problems from the literature. Each problem consists of a number of tasks, task completion times and precedence constraints. The cycle times are also given from the literature. The combination of different values gives 25 problem instances. All precedence graphs are shown in Figure 3.2-3.12 and Table 3.1-3.2.

3.3.1.1 Precedence graphs

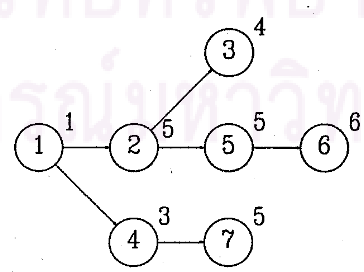


Figure 3.2 Precedence network for the Merten’s 7-task test example

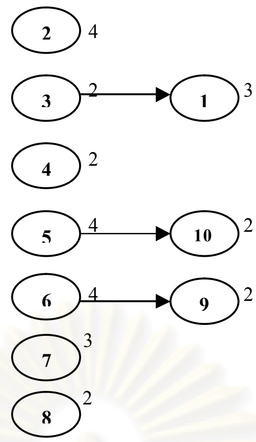


Figure 3.3 Precedence network for the Miltenburg's 10-task test example

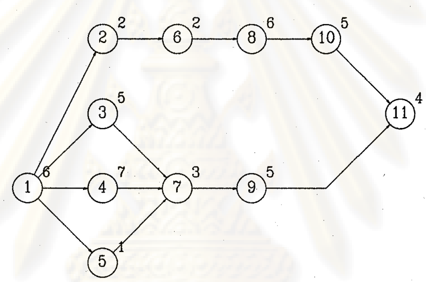


Figure 3.4 Precedence network for the Jackson's 11-task test example

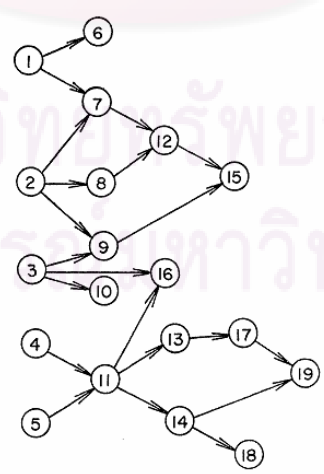


Figure 3.5 Precedence network for the Thomopoulos's 19-task test example

The elemental data pertaining to the assembly process are shown in Table 3.1 excerpted from Hwang and Katayama (2009). In column I it is seen that K = 19. Column II identifies the element times t_{jk} (j = 1, 2, 3; k = 1, 2,…, 19). Column III gives the average elemental time for all models (i.e. $\bar{t}_k$ k = 1, 2,…, 19). *Average element times* ($\bar{t}_k$) calculated from the real demand of each model in Thomopoulos (1970) are related to the precedence network of 19 tasks in Figure 3.5. The given cycle time is equal to 2 minutes.

Table 3.1 Element times (minute) for an example problem

(I) Element or Task *(k)*	(II) Element Times/Model			*(III) Average Times* ($\bar{t}_k$)
	1 (t_{1k})	2 (t_{2k})	3 (t_{3k})	
1	0.1	0	0.1	*0.37*
2	1.2	1.2	1.2	*1.20*
3	0	0.4	0.4	*0.27*
4	0.4	0	0	*0.13*
5	0.2	0.2	0.2	*0.20*
6	0.2	0	0	*0.07*
7	0.6	0.6	0.6	*0.60*
8	0	0.5	0.5	*0.33*
9	0.4	0.4	0.4	*0.40*
10	0	0	0.2	*0.07*
11	0.3	0.3	0.3	*0.30*
12	0.5	0.5	0.5	*0.50*
13	0.1	0	0.1	*0.07*
14	0.2	0.2	0.2	*0.20*
15	1.5	0	1.5	*1.00*
16	0	0.1	0	*0.03*
17	0.5	0.5	0	*0.33*
18	0.5	0.5	0.5	*0.50*
19	0.4	0.4	0	*0.27*
Total	8.0	5.8	7.6	*7.13*

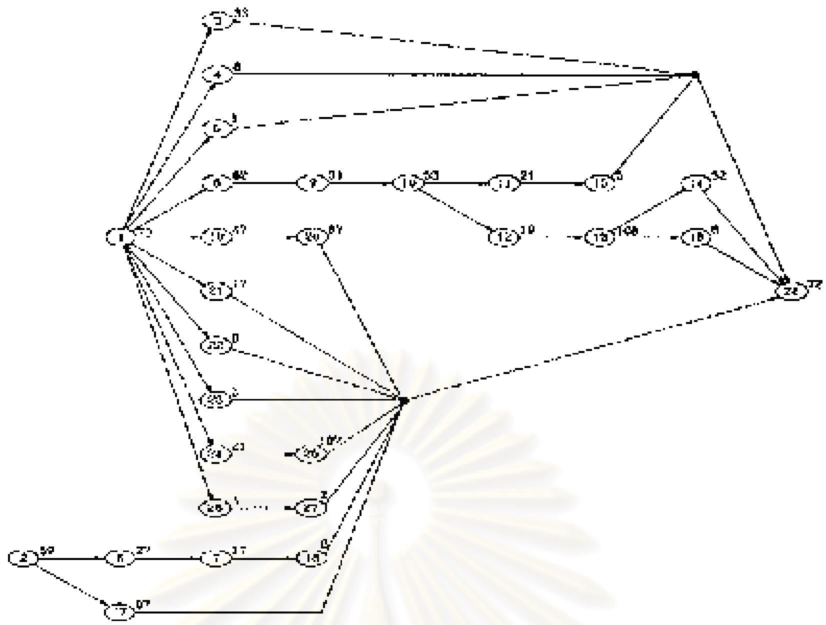


Figure 3.6 Precedence network for the Heskiaoff's 28-task test example

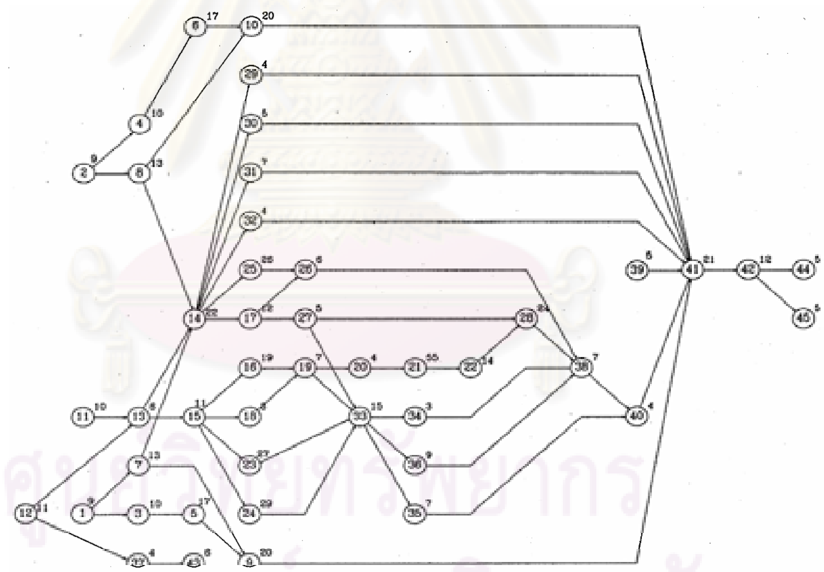


Figure 3.7 Precedence network for the Kilbridge&Wester's 45-task test example

Another problem composed of 61 tasks is a practical one obtained from an automobile company while Kim *et al.* are carrying out an industry project with the company. Their precedence graph is not shown, but precedence relation is presented at http://syslab.chonnam.ac.kr/links/data-mmulbs.doc and task times are excerpted from Hwang and Katayama (2009).

Figure 3.8 Precedence network for the Kim's 61-task test example

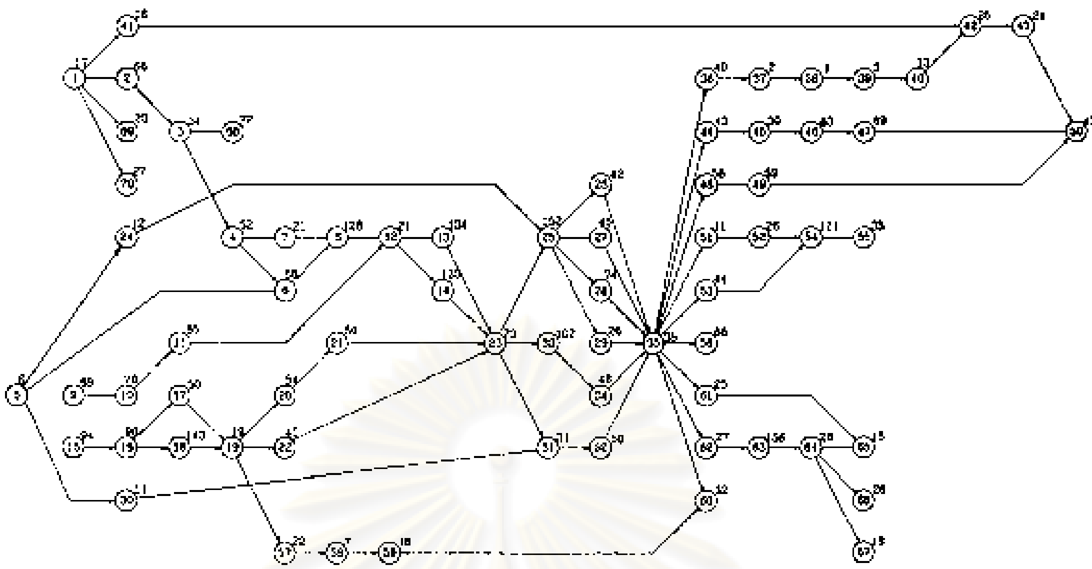


Figure 3.9 Precedence network for the Tongue's 70-task test example

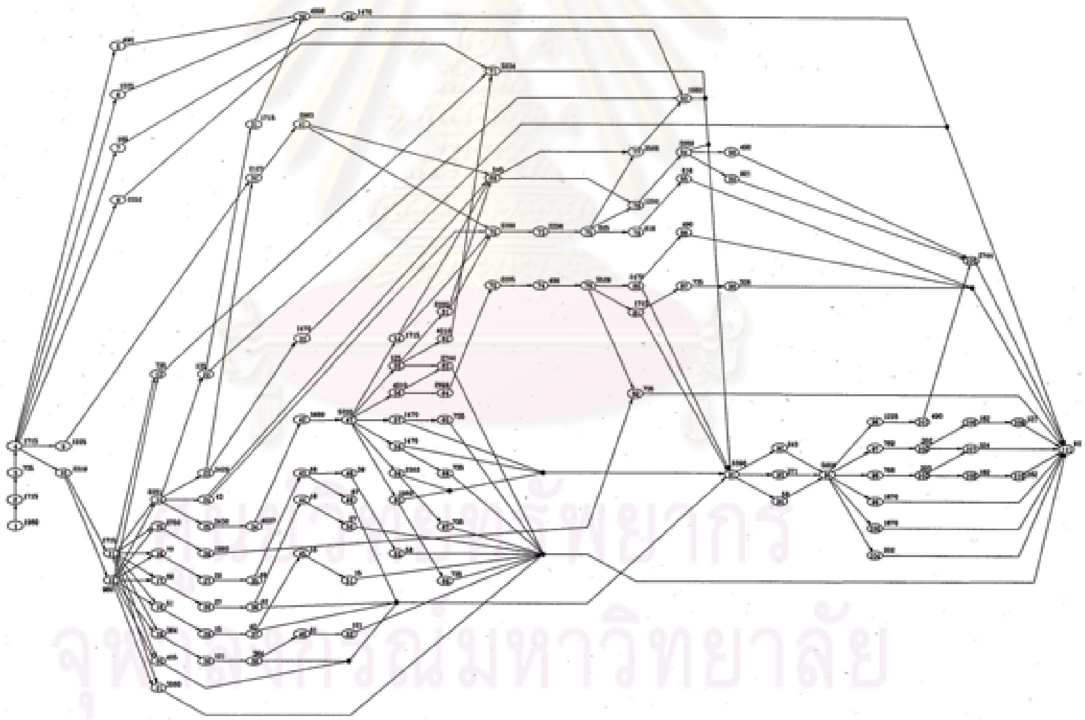


Figure 3.10 Precedence network for the Arcus's 111-task test example

Another problem composed of 297 tasks is obtained from data sets of Scholl and Klein (1999). Their precedence graph is not illustrated, but precedence relation and task times are presented numerically at http://www.bwl.tu-darmstadt.de/bwl3/forsch/projekte/alb/index.htm.

Figure 3.11 Precedence network for the Scholl and Klein's 297-task test example

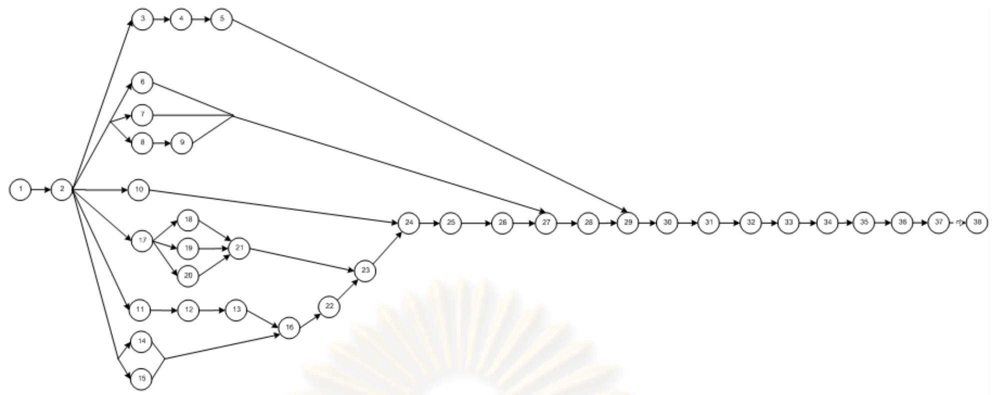


Figure 3.12 Precedence network for this case study's 36-task test example

From our case study, *deterministic mixed-model manual times* (MT_{Mi}) shown in Table 3.2 are related to the precedence network of 36 tasks in Figure 3.12. The given cycle time is equal to 23 minutes or 1,371 seconds (calculated by cycle time = work hour per day *60 *60/ daily demand = 8*60*60/21).

3.3.2 Decision variables

Controllable factors in this research are two fixed layout of U-lines and operator movement rules. Orthogonal distance is not practicable. Displacement rule is put into the model instead. The % average processing time is fixed in each problem after doing the experiments in the next chapter.

3.3.2.1 Fixed layout of U-lines

The task location of a U-line is placed in the front, back and side. The pattern of "U" is determined by the base line (side) that is not larger than the rest of lines (front and back). In other words, a worker does not walk across a line from a front line to a back line, or vice versa, if a side line is very wide. There are a few fixed U-line layouts that are assumed by Miltenburg (2001) and Cheng *et al.* (2000). A number of tasks and side ratios are 10 (Miltenburg) and 2:4, 28 (Heskiaoff) and 2:13, 30 (Sawyer) and 2:14, and 45 (Kilbridge&Wester) and 3:21.

Table 3.2 Deterministic manual times (seconds) for all models

Task No.	Single Model			Mixed Model		
	A (MT_{Ai})	B (MT_{Bi})	C (MT_{Ci})	Average (MT_{Mi})	10:6:5 (21 pieces/ 8 hours)	T_{Mi} *per piece*
1	0.00	0.42	0.00	0.14	2.50	*7.14*
2	0.00	0.00	0.75	0.25	3.75	*10.71*
3	9.40	9.40	9.40	9.40	197.40	*564.00*
4	10.33	0.00	0.00	3.44	103.33	*295.23*
5	0.00	13.00	0.00	4.33	78.00	*222.86*
6	0.00	0.00	1.03	0.34	5.17	*14.77*
7	0.00	0.00	15.65	5.22	78.25	*223.57*
8	5.57	5.57	5.57	5.57	116.90	*334.00*
9	0.00	0.00	6.33	2.11	31.67	*90.49*
10	3.48	3.48	3.48	3.48	73.15	*209.00*
11	3.13	3.13	3.13	3.13	65.80	*188.00*
12	3.17	3.17	0.00	2.11	50.67	*144.77*
13	0.00	0.00	3.73	3.73	18.67	*53.34*
14	3.15	3.15	3.15	3.15	66.15	*189.00*
15	2.63	2.63	2.63	2.63	55.30	*158.00*
16	0.00	0.00	1.25	0.42	6.25	*17.86*
17	0.77	0.00	0.00	0.26	7.67	*21.91*
18	0.00	1.60	0.00	0.53	9.60	*27.43*
19	2.18	2.18	2.18	2.18	45.85	*131.00*
20	4.28	4.28	4.28	4.28	89.95	*257.00*
21	2.67	2.67	2.67	2.67	56.00	*160.00*
22	2.13	3.73	3.73	3.73	62.40	*178.29*
23	1.40	1.40	1.40	1.40	29.40	*84.00*
24	7.13	7.13	7.13	7.13	149.80	*428.00*
25	1.53	1.53	1.53	1.53	32.20	*92.00*
26	2.07	2.07	2.07	2.07	43.40	*124.00*
27	2.00	2.00	2.00	2.00	42.00	*120.00*
28	2.00	2.00	2.00	2.00	42.00	*120.00*
29	0.00	0.88	0.00	0.29	5.30	*15.14*
30	10.62	10.62	10.62	10.62	222.95	*637.00*
31	3.42	3.42	3.42	3.42	71.75	*205.00*
32	7.78	7.78	7.78	7.78	163.45	*467.00*
33	4.28	4.28	4.28	4.28	89.95	*257.00*
34	3.55	3.55	3.55	3.55	74.55	*213.00*
35	3.50	3.50	3.50	3.50	73.50	*210.00*
36	3.20	3.20	3.20	3.20	67.20	*192.00*
Total Time (s)	105.38	111.78	121.47	115.90	2331.87	*6662.51*

The shape of a single U-line layout effects on the results of number of workers, DOW and WT. Consequently, two U-shaped layouts are fixed in the symmetrical shape at the side ratio 1:1:1 (1/3) in Table 3.3 and in the rectangular shape at the side ratio 1:4:4 (1/9) in Table 3.4. Although the size of a layout depends on a facility location, the distance from one location to another location is determined by the % average processing time described in the next chapter. Likewise, Balakrishnan *et al.* (2009) used 5% and 10% average processing times in their paper.

Table 3.3 Task location for data sets of UALBPs at the side ratio of 1:1:1 (1/3)

Problem	Number of tasks	Side Tasks	Front Tasks	Back Tasks	Cycle Time	Number of product models
1. Merten (1967)	7	1	3	3	7	1
					10	
					18	
2. Miltenburg (2001)	10	2	4	4	10	1
					10	
3. Jackson (1956)	11	3	4	4	7	1
					13	
					21	
4. Thomopoulos (1970)	19	7	6	6	120	3
5. Heskiaoff (1968)	28	10	9	9	138	1
					256	
					342	
6. Kilbridge&Wester (1961)	45	15	15	15	57	1
					110	
					184	
7. Kim (2006)	61	21	20	20	600	4
8. Tongue (1961)	70	24	23	23	160	1
					251	
					527	
9. Arcus (1963)	111	37	37	37	6,837*	5
					7,916	
					17,067	
10. Scholl&Klein (1999)	297	99	99	99	1,394	1
					1,834	
					2,787	
11. This case study	36	12	12	12	1,371	3

* Minimum cycle time (5,755) is less than the operation time of 6,615. Thus, the feasible minimum cycle time from the data sets of UALBP-I is replaced.

Table 3.4 Task location for data sets of UALBPs at the side ratio of 1:4:4 (1/9)

Problem	Number of Tasks	Side Tasks	Front Tasks	Back Tasks	Cycle Time	Number of product models
1. Merten (1967)	7	1	3	3	7	1
					10	
					18	
2. Miltenburg (2001)	10	2	4	4	10	1
3. Jackson (1956)	11	1	5	5	7	
					13	1
					21	
4. Thomopoulos (1970)	19	3	8	8	120	3
5. Heskiaoff (1968)	28	4	12	12	138	1
					256	
					342	
6. Kilbridge&Wester (1961)	45	5	20	20	57	1
					110	
					184	
7. Kim (2006)	61	7	27	27	600	4
8. Tongue (1961)	70	8	31	31	160	1
					251	
					527	
9. Arcus (1963)	111	11	50	50	6,837*	5
					7,916	
					17,067	
10. Scholl&Klein (1999)	297	33	132	132	1,394	1
					1,834	
					2,787	
11. This case study	36	4	16	16	1,371	3

3.3.3 Data sets and lower bounds

From data sets of Scholl and Klein (1999), Miltenburg (2001a) for the 10-task problem and Hwang *et al.* (2008) for the 19-task and 61-task problems, UALBP-1 standard benchmarks in the section of literature review are established. The procedure of U-line optimizer yielded promising results especially for the objective of minimizing number of workers. To validate experimental results for our problems, according to Table 3.5, the lower bound of number of workstations of some selected problems extracting from the web at http://www.assembly-line-balancing.de, Miltenburg (2001a), and Hwang *et al.* (2008) can be compared. However, the lower bound of workstations is minimized globally without the consideration of walking path. In other words, the optimum solutions are the minimum number of workers with no the walking constraint.

Table 3.5 Optimal results of UALBP-I obtained with ULINO (U LINe Otpimizer)

Problem	Number of Tasks	Network Density (D)	Cycle Time	ULINO solutions (time limit 500 seconds, PC 486 DX-2 66MHz) Optimum workers [LB,UB]
1. Merten (1967)	7	0.2857	7	5
			10	3
			18	2
2. Miltenburg (2001)	10	0.0667	10	3
3. Jackson (1956)	11	0.2364	7	7
			13	4
			21	3
4. Thomopoulos (1970)	19	0.1228	120	*4 (From MATLAB 100 generations)*
5. Heskiaoff (1968)	28	0.1032	138	8
			256	4
			342	3
6. Kilbridge & Wester (1961)	45	0.0626	57	10
			110	6
			184	3
7. Kim (2006)	61	0.0361	600	9 *(From MATLAB 150 generations)*
8. Tongue (1961)	70	0.0356	160	[22,23]
			251	14
			527	7
9. Arcus (1963)	111	0.0283	6,837	[22,23]
			7,916	[19,20]
			17,067	9
10. Scholl & Klein (1999)	297	0.0096	1,394	[50,51]
			1,834	38
			2,787	25
11. This case study	36	0.0587	1,371	*5 (From MATLAB 100 generations)*

Network density, a characteristic which measures the strength of this relation, has been found to be an important factor in influencing heuristic performance in previous investigations of the line balancing problem (Talbot *et al.*, 1986). To define density, let W be an N x N 0-1 matrix that represents a precedence ordering relation P. For a given element ω_{ij} of W, let $\omega_{ij} = 1$ if $X_i \in P_j$ (i.e., if task i precedes task *j*) and $\omega_{ij} = 0$ if task *i* does not precede task *j*, or $X_i \notin P_j$. Let *d* be the total number of relations contained in *P*, or $d = \sum_{i=1}^{N}\sum_{j=1}^{N}\omega_{ij}$. The maximum number of relations that can be included in *P* is $N(N - 1)/2$. The density of the assembly network is defined to be the ratio: $D = 2d/[N(N - 1)]$, or the ratio of the number of relations that exist to the number which could exist. The measure 1 - *D* is the *F*-ratio used by Dar-El (1975), and the measure D is referred to as *order strength* by Mastor (1970). Values of *D* close to 1 indicate a highly interconnected network, and fewer alternatives available for assigning tasks to a work station. Values of *D* close to 0 indicate relatively fewer precedence relationships, and more opportunities for assigning tasks to a work station. In this evaluation, values of *D* shown in the column III from Table 3.5 are used for problem generation.

CHAPTER IV

MATHEMATICAL SOLUTION APPROACHES

4.1 Introduction

In this chapter, the U-shaped manual assembly line worker allocation problem is classified into the same nondeterministic polynomial time – hard (NP-hard) as the combinatorial optimization problem. Although it is essential to solve a large-sized problem, an exact solution that is generally accepted for solving a small-sized problem is unavoidably considered in the beginning stage. The mathematical formulation provides a better understanding of the problem in developing heuristic procedures. Thus, the problem characteristics, mathematical formulation, assumptions and an illustrative example are presented in this chapter. The collaborative with the case study company is extremely important in the process of model validation for the problem. The investigation of gathered data from reviewed papers and a case study helps to bridge some missing more completely.

A popular notation used in assembly line balancing problems is reviewed in the form of three basic elements $[\alpha|\beta|\gamma]$. The first parameter (α) describes the precedence graph characteristics. The second parameter (β) is the station and line characteristics. The last parameter (γ) contains the objectives. From the analytical study of the reviewed literature and Thai apparel companies, the viewpoint of the line balancing problem is developed to the single and mixed-model U-shaped manual assembly line balancing problem under the multiple objectives of the minimum number of workers, the minimum deviation of operation times of workers (*DOW*) and the minimum walking time (*WT*). In other words, this research problem focuses on the notation of $[mix|u|m, DOW, WT]$.

Finally, two minor research questions are also fulfilled at the end of this chapter. First, how much is the appropriate walking time between tasks considered in the U-line instead of the ignorance as the straight-line problem? Secondly, how do the U-shaped layouts effect on walking time and a number of workers?

4.2 Characteristics of a Single Assembly U-line

Although there are many types of U-lines, the configuration of this study is a single U-shaped assembly line only. The U-line arranges machines or tasks around a U-shaped line in the order in which production tasks are serial. The sequence of tasks on the U-line is not fixed, making it possible to reallocate tasks to different line locations. Thus, the assignment of tasks to line locations can be altered. The system is one-piece flow manufacturing moving one piece at a time between tasks within a U-line. One floating worker supervises both the entrance and the exit of the line. The task efficiency is proportional to the worker's performance. Machine-work is not separated from worker-work. Standard operation charts specify exactly how all work is done. Workers can be reallocated periodically when production requirements change (or cycle time changes). This requires workers to have multi-functional skills to operate several different machines or tasks. It also requires workers to work standing up and walking because they need to operate at different locations. Whenever a worker arrives at a task, one performs any needed tasks at the task location, and then walks to the next task. Following the last task of a path, the worker returns to the starting point and works or waits for the start of the next cycle. Any succeeded part is put in a bin at the location. A succeeded part is moved to an adjacent area in a single U-line in sequence from entrance to exit. However, taking a succeeded part in practice is conditioned as follows:

a. If next task is done by oneself and run in sequence along a single U-line, the part will be carried by oneself;

b. If next task is not done by oneself and any operation time of a worker is operated less than some units of cycle time, a worker may walk back to take a succeeded part by oneself;

c. If next task is not done by oneself and any operation time is operated equal to cycle time, taking a succeeded part will be prepared by a floating worker from signal of Andon light at that machine.

The characteristics of the single U-shaped assembly line worker allocation problem of type I are shown in Figure 4.1.

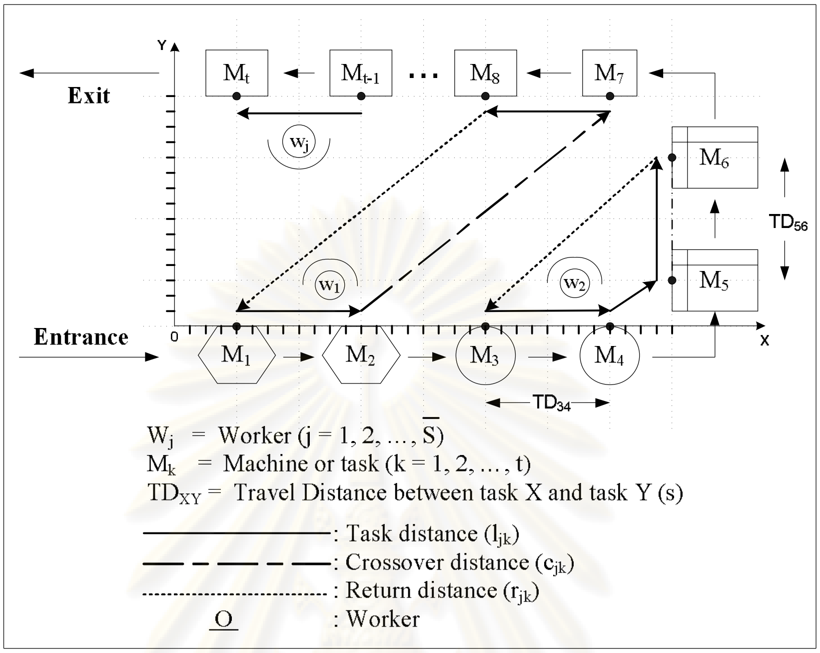


Figure 4.1 Mapping a diagram of a single U-shaped assembly line for *j* workers and *k* machines on grid arrangement

4.3 Exact Solution

First, this study focuses on obtaining a mathematical formulation for the worker-machine assignment problem on the single U-line of UALBP-type I (or ⊃) without the consideration of travel time, which appears to be a real case situation in several manufacturing systems.

Integer linear programming (Urban, 1998; Scholl and Klein, 1999) and dynamic programming (Miltenburg and Wijngaard, 1994; Miltenburg, 1998), linked by Miltenburg (2001) are two modeling approaches that have been used to assign tasks into stations (workers). To validate the results of the minimum of number of workers and the shortest walking time from the 10-task problem at the ratio of 2:4:4, this research refers to problem decomposition by Miltenburg (2001a). The former result is solved with Integer Linear Programming (ILP). The latter result is solved with Dynamic Programming (DP). Furthermore, the study of Miltenburg also gives us basic factors for this research.

Exact solution for worker allocation problem under the fixed workstation mode

There are two restrictions on the assignment of tasks to locations on the U-shaped line. The first, is that the assignment must satisfy whatever technological requirements exist for producing the product. These are specified as precedence constraints on the order in which tasks may be completed. The second restriction applies when the U-line works in fixed or overlapping workstation modes. In these cases the assignment must also permit tasks to be grouped into a minimum number of stations (workers).

The U-line may be described as shown in Figure 4.2. Denote by a the point where material enters the line, and a' the point where finished products leave. Let each task k = 1,2, ... require a distance $l_k \geq 0$ on the line. Define the *middle* of the line to be a point e located a distance greater than or equal to the integer part of $(\sum_k l_k)/2$ from a. Define the *front* of the line to be the locations from a to e, and the *back* of the line to be the locations from a' to e. Let $F = \|a-e\|$ be the length of the front of the line and $B = \|a'-e\|$ be the length of the back of the line, where $F + B \geq \sum_k l_k$.

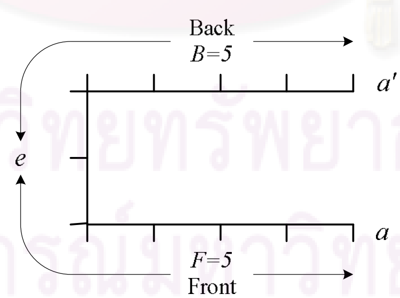


Figure 4.2 The single U-line

Miltenburg (2001a) assumed that ten tasks, each requiring one unit of distance, are to be placed around the U-shaped production line. The line begins at point a and ends 10 locations later at point a'. The middle of the line is 10/2 = 5 units of distance from *a*. Thus, five tasks should be placed on the front of the line and five tasks should be placed on the back.

Data set of Miltenburg (2001a, p.311)

Suppose C = 10; the manual task times are two time units for tasks 3, 4, 8, 9, 10, three time units for tasks 1, 7, and four time units for tasks 2, 5, 6; and the precedence constraints are (3, 1), (5, 10), (6, 9).

One feasible solution by hand calculation

According to Table 4.1 with no walking path, the minimum three workers from an example with a random priority rule is calculated, but the minimum number of workers with walking path are tabulated in the next section.

Table 4.1 One feasible solution without walking time

			Task assignment on a U-line			
Woker or Workstation	**Task considered Front**	**Task considered Back**	**Task**	**Task time (Time unit)**	**Total task time (Time unit)**	**Idle time (Time unit)**
1	7	1	1	3	3	7
1	7	10	7	3	6	4
1	2	10	2	4	10	0
2	6	10	10	2	2	8
2	6	9	6	4	6	4
2	5	9	5	4	10	0
3	8	9	8	2	2	8
3	4	9	9	2	4	6
3	4	-	4	2	6	4
3	3	-	3	2	8	2

* *All tasks are selected by a random heuristic rule.*

4.4 Model Formulation

4.4.1 Notations

The notation used in this section can be summarized as follows:

Indices

j = index on workers

k = index on tasks

Parameters

t = number of tasks in the U-line

C = given cycle time

P = $\{(k,l)$: task k must be completed before task l can begin$\}$ – precedence constraints

t_k = time to complete manual task at task k

α = coefficient of walking time

Variables

$\bar{S}$ = number of workers in the U-line

C_{jk} = cycle time at task k assigned to worker j

m = minimum number of workers

l_{jk} = task distance time at task k assigned to worker j

c_{jk} = crossover distance time at task k assigned to worker j

r_{jk} = return distance at task time k assigned to worker j

T_{jk} = task time at task k assigned to worker j

WT_{jk} = walking time at task k assigned to worker j

Decision variables

$$x_{jk} = \begin{cases} 1 & \text{if task } k \text{ is located on front of line and assigned to worker } j, \\ 0 & \text{otherwise.} \end{cases}$$

$$y_{jk} = \begin{cases} 1 & \text{if task } k \text{ is located on back of line and assigned to worker } j, \\ 0 & \text{otherwise.} \end{cases}$$

$$x_{jl} = \begin{cases} 1 & \text{if task } l \text{ is located on front of line and assigned to worker } j, \\ 0 & \text{otherwise.} \end{cases}$$

$$y_{jl} = \begin{cases} 1 & \text{if task } l \text{ is located on back of line and assigned to worker } j, \\ 0 & \text{otherwise.} \end{cases}$$

$$z_j = \begin{cases} 1 & \text{if worker } j \text{ is used,} \\ 0 & \text{otherwise.} \end{cases}$$

According to this notation, a mathematical model for solving the worker allocation problem is described in the following section.

4.4.2 Objective functions

This section identifies the minimum number of workers (workstations) in Eq. (4.1) required in the U-line to obtain the optimum of dual objectives. Besides aiming to increase productivity (minimizing the number of workers or the cycle time), some other goals are important for the addition of high productivity achievements, i.e., a sense of equity among workers and the shortest travel path. Hierarchically both objective functions are calculated accordingly in the same unit of time from Eq. (4.2), (4.3), and (4.4). An ineffective allocation of workers to tasks and machines would yield long idle times (imbalance workload) and long walking time.

Then select x_{jk}, y_{jk}, z_j to,

$$\text{(ILP) Minimize } \sum_{j=1}^{\bar{S}} z_j \qquad (4.1)$$

After computing the minimum number of workers in the first step, it is necessary to evaluate and minimize the deviation of operation times of workers (DOW) and the walking time (WT) with Pareto-optimal frontier.

Objective functions:

I. Min. the Deviation of Operation times of Workers (DOW)

$$DOW = \sqrt{\frac{\sum_{j=1}^{\bar{S}}\sum_{k=1}^{t}(C - C_{jk})^2}{m}} \tag{4.2}$$

$$\underset{\text{(U-Worker Balancing)}}{\text{Cycle Time } (C_{jk})} = \underset{\text{(U-line Balancing)}}{\text{Task Time } (T_{jk})} + \text{Walking Time } (WT_{jk}) \tag{4.3}$$

<u>P.S.</u> Both the root mean square error (RMSE) and the mean absolute error (MAE) result in the positive values (Konno and Yamazaki, 1992). The result of RMSE is measured in the same units as the data rather than in the square units of the mean square error (MSE). The MAE is also measured in the same units as the original data, but slightly smaller than the RMSE. To obviously clarify the spread of solutions, the RMSE, named DOW that is a model imbalance measure is evaluated in Eq. (4.2).

II. Min. the Walking Time (WT)

$$WT = \sum_{j=1}^{\bar{S}}\sum_{k=1}^{t}(l_{jk} + c_{jk} + r_{jk}) \tag{4.4}$$

<u>The foremost motivation to consider the criteria of DOW and WT</u>

- They are significant metrics of shop-floor performance for the labor-intensive industry.

- Lower idle time usually indicates more efficient utilization for a worker.

4.4.3 Constraints

Subject to:

$$\sum_j (x_{jk} + y_{jk}) = 1 \qquad \text{for each operation } k, \tag{4.5}$$

$$\sum_j (\bar{S} - j + 1)(x_{jk} - x_{jl}) \ge 0 \qquad \text{for all } (k,l) \in P, \tag{4.6}$$

$$\sum_j (\bar{S} - j + 1)(y_{jl} - y_{jk}) \ge 0 \qquad \text{for all } (k,l) \in P, \tag{4.7}$$

$$x_{jk}, y_{jk}, z_j \in \{0,1\} \qquad \text{for all } j,k. \tag{4.8}$$

The first constraint in Eq. (4.5) ensures that every task is located on the front or back of the line and is assigned to one worker. The next constraint in Eq. (4.6) ensures that the precedence constraints are satisfied for each task assigned to the front of the line. The following constraint in Eq. (4.7) does the same for the tasks of the equation (6) assigned to the back of the line. In other words, constraint (4.6) enforces task sequence assigned on the U-line by a set of ordered pairs of tasks reflecting the precedence relationships; for example, P = (3,1), (5,10) and (6,9) is the ordered pair of Miltenburg's 10-task problem indicating task k precedes task l. X_{jk} or/and X_{jl} is 1 when worker j does task k or/and task l. Otherwise, its value is 0 or their values are 0. Constraints (4.7) is the same, but is reversed because task on the U-line can be also assigned at the back line. The variables of x, y, and z are binary solution in Eq. (4.8). However, Miltenburg (2001a) does not take walking distance into account and may not find the best U-line design. Thus, the last constraint of walking time in Eq. (4.9) is essential to complete the worker allocation problem. The constraint proves that the sum of the manual task times for the tasks in each worker in the first term and the total walking distance in the second term does not exceed the cycle time, C. The coefficient of walking time (α) is varied by TD_{XY} in Figure 4.1 or the percentage of Average Processing Time (APT) from one task to another task. The average processing time is defined as APT = $\sum_{k=1}^{t} t_k / t$.

$$W_j = \sum_j T_{jk} + \alpha \sum_j (\mathrm{l}_{jk} + \mathrm{c}_{jk} + \mathrm{r}_{jk}) \le C \qquad \text{for all } j,k \tag{4.9}$$

4.5 Assumptions

In this study, the SUALWAPs-I is subjected to the following assumptions.

- A U-line comprises inexpensive and small non-automated machines. Several identical machines may be found and machines are enough to be allocated in a single U-line;

- Machines or tasks are located via a grid arrangement with the same distance of %APT between adjacent task locations in the same row. For other non-adjacent task locations, the walking distance is calculated by the displacement of Euclidean distance;

- Trained homogeneous skilled workers have the same efficiency and multi-functional skills and are able to operate any processes or machines. They walk in a circle inside the U-line (also called the zone constraint – machines allocated to each worker must be adjacently located within a loop);

- A worker is assigned to one station (or one loop) only;

- All parameters and variables such as processing times and walking times are deterministic (known and constant);

- The completion time of a machine or task summed with many subordinate tasks is known and a task cannot be split between two or more workers;

- Precedence relationships of the problem are consistent from model to model. That is, if task k precedes task l in any model there is no other model where task l must precede task k. Each unit of products is processed through all tasks in the same precedence order;

- Setup times (assumed to be less than 10% compared with processing time) are negligible. U-lines can be operated as single-model and mixed-model lines where each worker is able to produce any product in any cycle. Consequently, job sequence is regardless at any period;

- The mixed-model task times use the weight of composite demand to transform average task time into the task of a single model. However, a floating worker may be assisted unless task times in some model are feasible;

- Learning effect has no consideration since it is assumed that worker performance runs into steady state already;

The mathematical model of this research is not studied in depth because minimizing the number of workers, DOW and WT at the same time make the exact solution too complex to deal with.

4.6 An Illustrative Example

The feasible solutions are started with the Miltenburg's 10-task problem. Before solving computational problems, steps for getting the exemplified values of workers, DOW and WT are calculated by hand as follows:

1. The precedence graph of Miltenburg 's 10 tasks, i.e. (3,1), (5,10), (6,9) and deterministic manual times, i.e. $\text{Task}^{\text{Manual time}} = 1^3, 2^4, 3^2, 4^2, 5^4, 6^4, 7^3, 8^2, 9^2, 10^2$ are used. The given cycle time is ten time units.

2. It is assumed that the assembly line worked at the steady state for a while (because any machines have no jobs at the transient state).

3. The priority-rule based procedure randomly generated by an illustrative of string #12 in Table 4.2 is employed to represent the priority of the task node for constructing a task sequence among candidates.

Table 4.2 An example of the priority-based encoding procedure

Task	1	2	3	4	5	6	7	8	9	10
N	Priority Task									
1	10	8	1	2	5	6	9	4	3	7

P.S. All values are randomized by the function of randsample(1:10,10) uniformly by MATLAB.

4. Task Sequence (TS) shown in Table 4.3 is done by the precedence constraints and the priority-rule based procedure referred to in the previous step.

Table 4.3 Task sequence influenced by the front and back work

No.	**Task (Front)**	**Task (Back)**	**TS**
1	2,3,4,5,6,7,8	1,9,10	1
2	2,3,4,5,6,7,8	9,10	7
3	2,3,4,5,6,8	9,10	2
4	3,4,5,6,8	9,10	10
5	3,4,5,6,8	9	6
6	3,4,5,8,9	9	5
7	3,4,8,9	9	8
8	3,4,9	9	9*
9	3,4	-	4
10	3	-	3

P.S. (9*) After No.6, Task 9 can be located in either front or back U-line.

5. Area allocation U-line layout (grid arrangement) is shown in Figure 4.3.

Figure 4.3 U-line$^{task(10)}_{side(2)}$ Layout

6. Adjacent matrix (From-To chart) of walking time under orthogonal distance and displacement distance for U-line is shown in Table 4.4 and 4.5, respectively. However, it is assumed that one time unit is equal to one distance unit.

For example, the displacement gives us a travel distance of 2.24 distance units (or time units), calculated as the sum of task distance (equivalent to location distance) $\|(0,0,(-1,2)\| = \sqrt{(0-(-1)^2+(0-2)^2} = 2.24,$ *where* $\|\bullet\|$ is a Euclidean distance worker. *Note that*: Location 1 is assumed to be an origin (0,0).

Table 4.4 Orthogonal distance for U-line $^{task(10)}_{side(2)}$ at the ratio of 2:4:4

Walking Time		To									
		1	2	3	4	5	6	7	8	9	10
From	1	-	1	2	3	4	5	5	4	3	2
	2	1	-	1	2	3	4	4	3	2	3
	3	2	1	-	1	2	3	3	2	3	4
	4	3	2	1	-	1	2	2	3	4	5
	5	4	3	2	1	-	1	2	3	4	5
	6	5	4	3	2	1	-	1	2	3	4
	7	5	4	3	2	2	1	-	1	2	3
	8	4	3	2	3	3	2	1	-	1	2
	9	3	2	3	4	4	3	2	1	-	1
	10	2	3	4	5	5	4	3	2	1	-

Table 4.5 Displacement distance for U-line $^{task(10)}_{side(2)}$ at the ratio of 2:4:4

Walking Time		To									
		1	2	3	4	5	6	7	8	9	10
From	1	-	1	2	3	3.54	3.81	3.61	2.83	2.24	2
	2	1	-	1	2	2.55	2.92	2.83	2.24	2	2.24
	3	2	1	-	1	1.58	2.12	2.24	2	2.24	2.83
	4	3	2	1	-	0.71	1.58	2	2.24	2.83	3.61
	5	3.54	2.55	1.58	0.71	-	1	1.58	2.12	2.92	3.81
	6	3.81	2.92	2.12	1.58	1	-	0.71	1.58	2.55	3.54
	7	3.61	2.83	2.24	2	1.58	0.71	-	1	2	3
	8	2.83	2.24	2	2.24	2.12	1.58	1	-	1	2
	9	2.24	2	2.24	2.83	2.92	2.55	2	1	-	1
	10	2	2.24	2.83	3.61	3.81	3.54	3	2	1	-

7. In the next step, tasks are assigned to all workers (workstations) from the above task sequence. A feasible U-shaped line balance is obtained with the cycle time of a worker (C = 10 time units) and adjacent matrix by an example of orthogonal type. Hand calculated results are shown in Table 8.12. Five number of workers (W = 5

workers) are given by the worker loads W1 = {1,7}, W2 = { 2,10}, W3 = {6,5}, W4 = {8,9,4}, W5 = {3}. While workers 1, 2, 3, 4, and 5 show cycle times of workers of 10, 10, 10, 10, and 2 time units and idle times of workers of 0, 0, 0, 0, and 8 time units, consecutively.

In Figure 4.4, it is assumed that one worker works only one workstation under a machine, in other words, workers walk no crossing path. Exemplified results are shown in Table 4.6.

Table 4.6 An example of worker allocation in a single U-line

Worker or Workstation	Task considered Front graph	Task considered Back graph	Task assignment on a U-line			Total (1) + (2) Task time (time unit)	WT to Origin (time unit)	Idle time (time unit)
			Task	(1) WT (time unit)	(2) Manual (time unit)			
1	7	- 1	-1	-	3	3	-	7
1	7	- 10	7	2	3	5 [8]	2	0
2	2	- 10	2	-	4	4	-	6
2	6	- 10	-10	2	2	4 [8]	2	0
3	6	-9	6	-	4	4	-	6
3	5	-9	5	1	4	5 [9]	1	0
4	8	-9	8	-	2	2	-	8
4	4	-9	9	1	2	3 [5]	1	4
4	4	-	4	1	2	3 [8]	2	0
5	3	-	3	-	2	2	-	8

P.S. 1. Several tasks from Table 4.3 illustrate only one task in the front column (column 2) and back column (column 3) due to small space

2. Negative sign in task assignment on a U-line is located in the back of U-line.

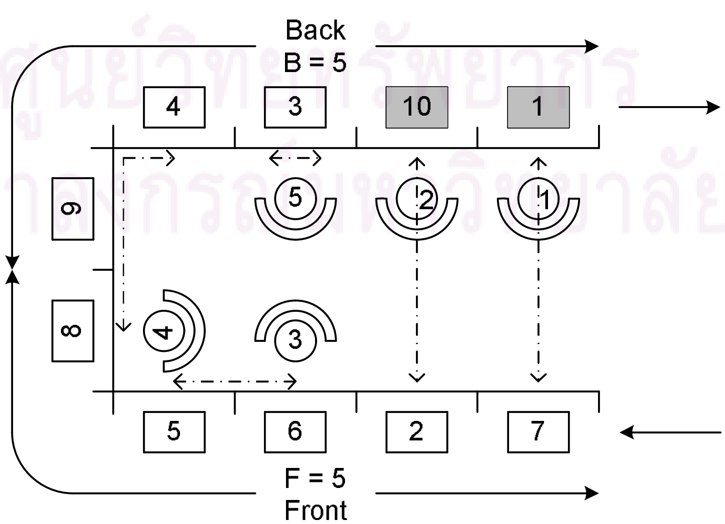


Figure 4.4 An example of worker allocation in a single U-line

Exemplified results are shown in Table 4.7.

Table 4.7 Final results of an example

String	Worker	WS	Manual Time	Travel Distance Time	Idle Time	Sorting Assigned Tasks
12	1	1	6	4	0	1, 7
	2	2	6	4	0	2, 10
	3	3	8	2	0	6, 5
	4	4	6	5	0	8, 9, 4
	5	5	2	0	8	3

8. The calculation of evaluation functions is composed of two objective functions:

Objective function I.

I. Min. the Deviation of Operation times of Workers (DOW)

$$\text{DOW} = \sqrt{\frac{\sum_{j=1}^{\bar{S}}\sum_{k=1}^{t}(C - C_{jk})^2}{m}}$$

$$\text{DOW} = \sqrt{\frac{(10-10)^2 + (10-10)^2 + (10-10)^2 + (10-10)^2 + (10-2)^2}{5}}$$

$$= \sqrt{\frac{64}{5}}$$

$$= 3.5777 \quad \text{time units}$$

Objective function II.

II. Min. the Walking Time (WT)

$$\text{WT} = \sum_{j=1}^{\bar{S}}\sum_{k=1}^{t}(l_{jk} + c_{jk} + r_{jk})$$

$$= (2+2)+(2+2)+(1+1)+(1+1+1+2)+(0)$$

$$= 15 \quad \text{time units (or distance unit by assumption)}$$

To evaluate the objective or fitness functions, heuristic algorithms for task allocation on the U-line have to be searched through both forward and backward directions randomly and assigned to a consecutive worker in each loop with the shortest path of all feasible task groups and the summation of task time and walking time that is less than and equal to given cycle time.

For example, suppose that task sequence in the 10-task U-line is [7,3,8,6,5,2,4,1,9,10], distance from task location to another task location is 0.14 s, and a given cycle time is 10 s. The tasks of the first worker are allocated with Eq. (4.9) in Figure 4.5 and Table 4.8. As a result, the task sequence [7,3,8,10] of the first worker gives the walking time minimum. After that, the cycle times of worker 2 to worker *j* are computed as the same.

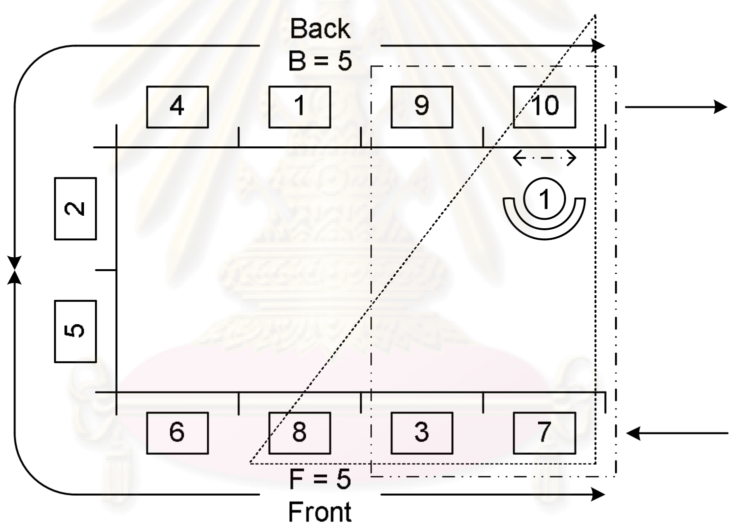


Figure 4.5 Task allocation for the first worker

Table 4.8 Task allocation of all feasible task groups for the first worker

Worker or Workstation	Task considered Front graph	Task considered Back graph	Task assignment on a U-line			Total (1) + (2) Task time (time unit)	WT to Origin (time unit)	Idle time (time unit)
			Task	(1) WT (time unit)	(2) Manual (time unit)			
1	7	- 10	7	-	3	3	-	7
(1-3	3	- 10	3	0.14	2	2.14[5.14]	0.14	4.72
tasks)	8	-10	-10	0.31	2	2.31[7.45]	0.28	2.27
1	7	- 10	7	-	3	3	-	7
(4 tasks)	3	- 10	3	0.14	2	2.14[5.14]	0.14	4.72
	8	-10	8	0.14	2	2.14[7.28]	0.28	2.44
	-	-10	10	0.40	2	2.40[9.68]	0.28	0.04
1	7	-10,-9	7	-	3	3	-	7
(4 tasks)	3	-10,-9	3	0.14	2	2.14[5.14]	0.14	4.72
	8	-10,-9	-9	0.28	2	2.28[7.42]	-	-
	8	-10	-10	0.14	2	2.14[9.56]	0.28	0.16

Except for the task of a line across another line and the selected tasks of the same line in sequence that are allocated as the procedure of Table 4.6, a number of tasks in each worker for all possible task groups must be selected by the shortest path of a worker as the procedure of Table 4.8.

e.g. Other patterns, that is, String #7 and String #11 as shown in Figure 4.6-4.7

String #7

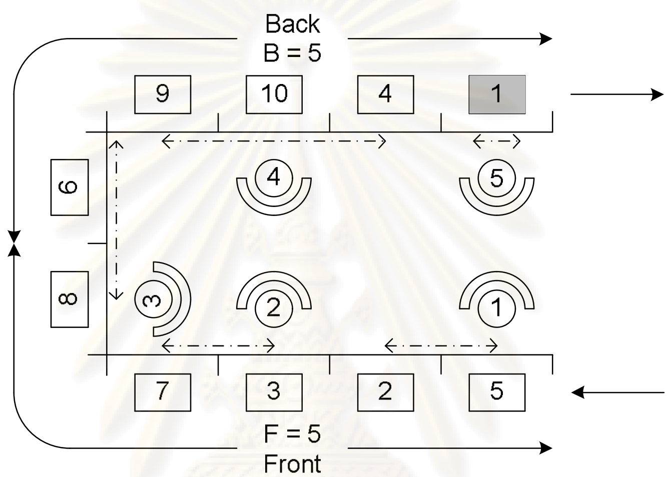


Figure 4.6 Another example of worker allocation in a single U-line

String #11

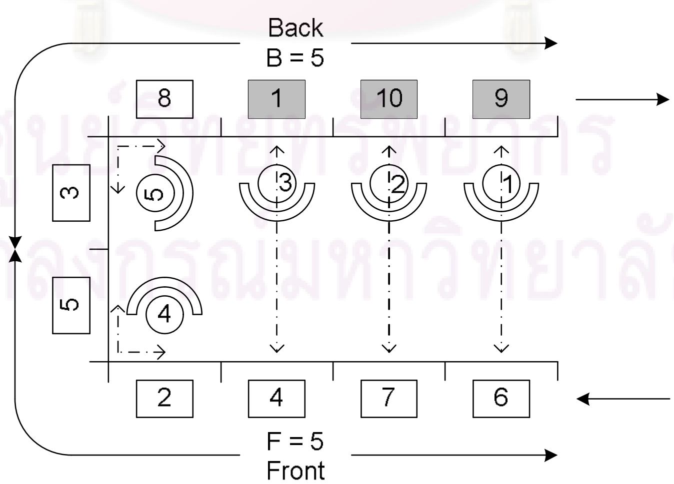


Figure 4.7 Another example of worker allocation in a single U-line (continued)

9. Several strings (examples) below are generated for getting several different results, i.e. number of workers, DOW and WT. Likewise, calculating from step 1 to step 8 is duplicated. The scatter plot including Pearson correlation -0.440 in Figure 4.8 makes confident preliminarily that two conflicting objectives are the Min.-Min. problem. Five, six and seven workers in Figure 4.9 are come up with DOW and WT.

TS_task =

String #0	3	8	9	10	4	2	6	1	7	5
String #1	3	8	1	4	7	9	5	6	2	10
String #2	7	3	8	10	2	1	5	9	6	4
String #3	5	7	9	4	6	3	1	10	8	2
String #4	7	2	6	9	3	5	10	4	8	1
String #5	2	3	5	8	6	9	4	7	1	10
String #6	6	9	10	4	8	1	5	3	7	2
String #7	5	2	1	3	7	8	6	9	10	4
String #8	6	7	3	4	8	2	5	10	1	9
String #9	5	7	2	3	10	1	8	4	9	6
String #10	10	2	7	5	8	6	4	9	1	3
String #11	9	6	10	7	1	4	2	5	3	8
String #12	1	7	2	10	6	5	8	9	4	3
String #13	5	4	7	10	9	6	1	8	2	3
String #14	9	5	2	8	4	7	10	3	6	1
String #15	5	4	8	6	2	1	7	10	3	9
String #16	9	3	5	10	7	1	6	2	4	8
String #17	9	10	8	5	3	6	4	2	1	7
String #18	7	6	10	9	8	2	3	4	5	1
String #19	10	7	2	8	3	4	9	6	5	1
String #20	10	3	4	9	5	8	2	1	7	6
String #21	3	1	8	4	7	9	10	6	2	5
String #22	2	7	5	9	6	10	1	4	3	8

[? = Front , ? = Back]

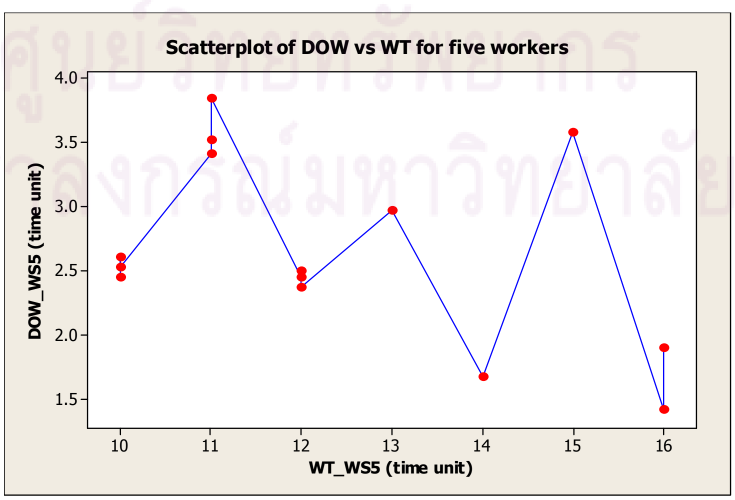


Figure 4.8 Scatter plot of DOW and WT for five workers from 14 strings

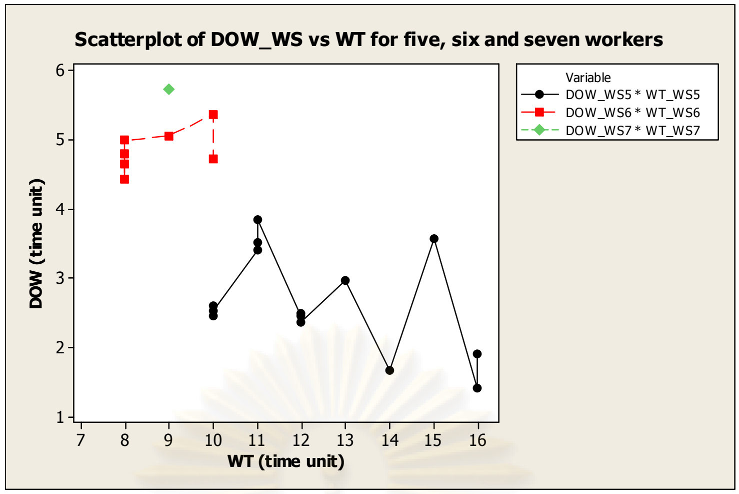


Figure 4.9 Scatter plot of DOW and WT for five, six and seven workers from 23 strings

4.7 Complexity of the Problem

It is well-known that the traditional assembly line balancing (ALB) problem is NP-hard. The ALB problem is a special case of the (single-model) U-line balancing problem, which, in turn, is a special case of the MMULB problem. Consequently, the MMULB problem is also NP-hard (Sparling and Miltenburg, 1998). In another viewpoint, minimizing walking time is equivalent to Traveling Salesman Problem (TSP) $O(n^2 2^n)$ that is definitely NP-hard (Lenstra and Rinnooy Kan, 1981). Therefore, from both significant substances it makes obviously strong that our research problem is NP-hard and looks forward to evolutionary algorithms in the next section.

4.8 Determination of Walking Time

In the previous papers (Miltenburg, 2001a; Miltenburg, 2001b; Miralles *et al.*, 2008), the coefficient of walking time (α) is required to travel a unit of distance or one time unit. Thus, in this study the adjacent matrix (From-To chart) of walking times under displacement distance for each problem of the symmetrical and rectangular shape is initially constructed at one time unit from one task to another task. Each of walking times between a pair of tasks is directly proportional to Euclidean distance between locations. The example of walking times for the 10-task problem is shown in Table 4.9. For example, the displacement gives us a travel distance of 0.7071 distance units (or time units), calculated as the sum of distance between task $\|(3,0),(3.5,0.5)\| = \sqrt{(3.5-3)^2+(0.5-0)^2} = 0.7071$, where $\|\bullet\|$ is a Euclidean distance operator. *Note that*: Location 1 is assumed to be an origin (0,0).

In practice, a worker keeps walking more than one second definitely. However, Balakrishnan *et al.* (2009) assumingly use two values of travel time for each problem instance, (i.e. walking time = the five or ten percentage of Average Processing Times (APT), where APT is the expected value of processing times which is defined as $APT = \sum_{k \in K} t_k / t$. In this study, the values of APT percentage are varied from 5% to 120% in Table 4.11 to find out the initial value that effects on the addition of a number of workers for each of all problems. For an example, the adjacent matrix of walking times for the 10-task problem at the 2:4:4 U-shaped layout is exemplified at 5% APT and shown in Table 4.10. Afterwards, the matrix that specifies the minimum APT percentage is input to the solution of minimum walking time.

Table 4.9 Exemplified displacement distance for U-line $^{task(10)}_{side(2)}$ at one time unit from one task to another task

Walking Time		To									
		1	2	3	4	5	6	7	8	9	10
From	1	0.0000	1.0000	2.0000	3.0000	3.5355	3.8079	3.6056	2.8284	2.2361	2.0000
	2	1.0000	0.0000	1.0000	2.0000	2.5495	2.9155	2.8284	2.2361	2.0000	2.2361
	3	2.0000	1.0000	0.0000	1.0000	1.5811	2.1213	2.2361	2.0000	2.2361	2.8284
	4	3.0000	2.0000	1.0000	0.0000	0.7071	1.5811	2.0000	2.2361	2.8284	3.6056
	5	3.5355	2.5495	1.5811	0.7071	0.0000	1.0000	1.5811	2.1213	2.9155	3.8079
	6	3.8079	2.9155	2.1213	1.5811	1.0000	0.0000	0.7071	1.5811	2.5495	3.5355
	7	3.6056	2.8284	2.2361	2.0000	1.5811	0.7071	0.0000	1.0000	2.0000	3.0000
	8	2.8284	2.2361	2.0000	2.2361	2.1213	1.5811	1.0000	0.0000	1.0000	2.0000
	9	2.2361	2.0000	2.2361	2.8284	2.9155	2.5495	2.0000	1.0000	0.0000	1.0000
	10	2.0000	2.2361	2.8284	3.6056	3.8079	3.5355	3.0000	2.0000	1.0000	0.0000

Table 4.10 Exemplified displacement distance for U-line $^{task(10)}_{side(2)}$ at 5% APT

Walking Time		To									
		1	2	3	4	5	6	7	8	9	10
From	1	0.0000	0.1400	0.2800	0.4200	0.4956	0.5334	0.5054	0.3962	0.3136	0.2800
	2	0.1400	0.0000	0.1400	0.2800	0.3570	0.4088	0.3962	0.3136	0.2800	0.3136
	3	0.2800	0.1400	0.0000	0.1400	0.2212	0.2968	0.3136	0.2800	0.3136	0.3962
	4	0.4200	0.2800	0.1400	0.0000	0.0994	0.2212	0.2800	0.3136	0.3962	0.5054
	5	0.4956	0.3570	0.2212	0.0994	0.0000	0.1400	0.2212	0.2968	0.4088	0.5334
	6	0.5334	0.4088	0.2968	0.2212	0.1400	0.0000	0.0994	0.2212	0.3570	0.4956
	7	0.5054	0.3962	0.3136	0.2800	0.2212	0.0994	0.0000	0.1400	0.2800	0.4200
	8	0.3962	0.3136	0.2800	0.3136	0.2968	0.2212	0.1400	0.0000	0.1400	0.2800
	9	0.3136	0.2800	0.3136	0.3962	0.4088	0.3570	0.2800	0.1400	0.0000	0.1400
	10	0.2800	0.3136	0.3962	0.5054	0.5334	0.4956	0.4200	0.2800	0.1400	0.0000

In this section, a variety of proposed average processing time percentage is tested in every problem. The computational results of a number of workers at the symmetrical and rectangular layouts are shown in Table 4.12 and 4.13, respectively. A number of tasks that are representative in each problem are shown in the first column. The second column displays the summation of processing times. The cycle time is determined from test-bed problems (Miltenburg, 2001a; Scholl and Klein, 1999; Hwang and Katayama, 2009; Miralles *et al.*, 2008) in the third column. The theoretical number of workers in the fourth column is calculated with the second column divided by the third column. However, no any paper displays the minimum number of workers for 7-task to 297-task U-shaped worker allocation problems with mathematical optimization technique. Thus, the straight-line ULINO for the line balancing problem of type I (Scholl and Klein, 1999) is benchmarked as lower bound on quantity of workers in the fifth column. The values of a number of workers in various %APT displays from the column six to the column twenty-one in every problem. The results show that the greater the walking time or %APT is, the larger a number of workers are. An example is shown in Figure 4.10.

From the experimental results of symmetrical and rectangular U-shaped layouts in Table 4.14, incrementing a number of workers in the first objective is sensitive to determining the walking time at only the five percentage of average processing time (or 0.14 to 65.61 seconds) in most problems. The 11-task problem is at the ten %APT (or 0.42 seconds.); the 19-task problem is at the 20 %APT (or 4.08 s.); and the 45-task problem is at the 15 %APT (or 1.84 seconds). It makes a conclusion that a decision to change a little walking time significantly effects the supplement of a larger number of workers in a single U-line. At last, the fixed average process time percentage is shown in the second and third columns and the differences of a number of workers between both layouts for all problems are shown in the fourth column. For an illustrative example, the experiment of 19-task problem in Figure 4.10 shows that walking time should be taken into account at the beginning of 20% average processing time (4.08 s.). After that, the difference of a number of workers between the symmetrical and rectangular layouts is cut off at the distinguished line at 60 %APT in the same Figure.

Table 4.11 Average processing time percentage of 5-120 for all problems

Problems / Number of tasks	Cycle time (s)	Average processing time (s)	Walking time from one task to another task (s)															
			5%	10%	15%	20%	25%	30%	35%	40%	50%	60%	70%	80%	90%	100%	110%	120%
1. Merten / 7	10	4.14	0.21	0.41	0.62	0.83	1.04	1.24	1.45	1.66	2.07	2.48	2.90	3.31	3.73	4.14	4.55	4.97
2. Miltenburg /10	10	2.8	0.14	0.28	0.42	0.56	0.70	0.84	0.98	1.12	1.40	1.68	1.96	2.24	2.52	2.80	3.08	3.36
3. Jackson / 11	13	4.18	0.21	0.42	0.63	0.84	1.05	1.25	1.46	1.67	2.09	2.51	2.93	3.34	3.76	4.18	4.60	5.02
4. Thomopoulos / 19	120	20.42	1.02	2.04	3.06	4.08	5.11	6.13	7.15	8.17	10.21	12.25	14.29	16.34	18.38	20.42	22.46	24.50
5. Heskiaoff / 28	256	36.57	1.83	3.66	5.49	7.31	9.14	10.97	12.80	14.63	18.29	21.94	25.60	29.26	32.91	36.57	40.23	43.88
6. Kilbridge&Wester / 45	110	12.27	0.61	1.23	1.84	2.45	3.07	3.68	4.29	4.91	6.14	7.36	8.59	9.82	11.04	12.27	13.50	14.72
7. Kim / 61	600	86.50	4.33	8.65	12.98	17.30	21.63	25.95	30.28	34.60	43.25	51.90	60.55	69.20	77.85	86.50	95.15	103.80
8. Tongue / 70	251	50.14	2.51	5.01	7.52	10.03	12.54	15.04	17.55	20.06	25.07	30.08	35.10	40.11	45.13	50.14	55.15	60.17
9. Arcus / 111	7,916	1,312.23	65.61	131.22	196.83	262.45	328.06	393.67	459.28	524.89	656.12	787.34	918.56	1,050	1,181	1,312	1,443	1,575
10. Scholl&Klein / 297	1,834	234.53	11.73	23.45	35.18	46.91	58.63	70.36	82.09	93.81	117.27	140.72	164.17	187.62	211.08	234.53	257.98	281.44
11. Case study / 36	1,371	185.08	9.25	18.51	27.76	37.02	46.27	55.53	64.78	74.03	92.54	111.05	129.56	148.07	166.58	185.08	203.59	222.10

86

Table 4.12 Theoretical, straight-line and U-line number of workers at the symmetrical layout

(1) No. of tasks	(2) Sum. of processing times (s)	(3) Cycle time (s)	(4) Number of workers at the symmetrical layout																	
			Theory (2) / (3)	Straight line ULINO	U-shaped line plus walking															
					5%	10%	15%	20%	25%	30%	35%	40%	50%	60%	70%	80%	90%	100%	110%	120%
7	29	10	2.9	3	4	4	4	5	5	5	5	5	6	6	6	7	7	7	7	7
10	28	10	2.8	3	4	4	4	4	4	4	5	5	5	5	5	6	6	6	8	8
11	46	13	3.54	4	4	5	5	5	5	6	6	6	6	8	8	8	9	9	9	9
19	388	120	3.23	4*	4	4	4	5	5	5	5	6	6	7	7	7	8	8	9	9
28	1,024	256	4	4	5	5	5	6	6	6	7	7	8	8	9	9	10	10	11	11
45	552	110	5.02	6	6	6	7	7	8	8	8	9	10	10	11	12	13	14	14	15
61	5,274	600	8.79	9*	10	11	12	13	13	14	15	15	17	18	20	20	22	23	25	27
70	3,510	251	13.98	14	17	18	19	20	22	22	23	25	26	28	31	33	34	36	38	39
111	145,657	7,916	18.4	19	22	24	25	26	29	30	30	30	40	40	40	40	50	50	50	50
297	69,655	1,834	37.98	38	44	49	52	55	59	62	65	69	74	81	87	93	100	106	109	115
36	6,663	1,371	4.86	5**	6	7	7	8	8	8	9	9	10	11	11	12	12	13	14	14

* Wattanapornprom *et al.*, 2009
** Olanviwatchai, 2009

Table 4.13 Theoretical, straight-line and U-line number of workers at the rectangular layout

(1) No. of tasks	(2) Sum. of processing times (s)	(3) Cycle time (s)	(4) Number of workers at the rectangular layout																	
			Theory (2) / (3)	Straight line ULINO	U-shaped line plus walking															
					5%	10%	15%	20%	25%	30%	35%	40%	50%	60%	70%	80%	90%	100%	110%	120%
7	29	10	2.9	3	4	4	4	5	5	5	5	5	6	6	6	7	7	7	7	7
10	28	10	2.8	3	4	4	4	4	4	4	5	5	5	5	5	6	6	6	8	8
11	46	13	3.54	4	4	5	5	5	5	5	6	6	6	8	8	8	9	9	9	10
19	388	120	3.23	4*	4	4	4	5	5	5	5	6	6	6	7	7	8	8	9	9
28	1,024	256	4	4	5	5	6	6	6	6	7	7	8	8	9	9	10	10	11	11
45	552	110	5.02	6	6	6	7	7	8	8	8	9	10	11	11	12	13	14	15	15
61	5,274	600	8.79	9*	10	11	12	12	13	14	15	15	17	18	19	20	22	23	25	27
70	3,510	251	13.98	14	16	18	19	20	21	22	23	25	27	28	30	32	34	36	38	40
111	145,657	7,916	18.4	19	22	24	25	27	28	30	31	33	36	39	41	44	47	50	52	54
297	69,655	1,834	37.98	38	44	48	52	55	58	62	65	69	75	81	87	93	100	106	109	114
36	6,663	1,371	4.86	5**	6	7	7	8	8	8	9	9	10	11	11	12	13	13	14	14

* Wattanapornprom *et al.*, 2009
** Olanviwatchai, 2009

87

Table 4.14 Fixed and different average processing time percentage for the 7-task to 297-task problems

Problem / Number of tasks	% Average processing time		
	Symmetry	Rectangle	Difference
1. Merten / 7	5% = 0.21 s	5% = 0.21 s	-
2. Miltenburg /10	5% = 0.14 s	5% = 0.14 s	-
3. Jackson / 11	10% = 0.42 s	10% = 0.42 s	30% = 1.25 s
4. Thomopoulos / 19	20% = 4.08 s	20% = 4.08 s	70% = 14.29 s
5. Heskiaoff / 28	5% = 1.83 s	5% = 1.83 s	15% = 5.49 s
6. Kilbridge&Wester / 45	15% = 1.84 s	15% = 1.84 s	60% = 7.36 s
7. Kim / 61	5% = 4.33 s	5% = 4.33 s	20% = 17.30 s
8. Tongue / 70	5% = 2.51 s	5% = 2.51 s	5% = 2.51 s
9. Arcus / 111	5% = 65.61 s	5% = 65.61 s	20% = 262.45 s
10. Scholl&Klein / 297	5% = 11.73 s	5% = 11.73 s	10% = 23.45 s
11. Case study /36	5% = 9.25 s	5% = 9.25 s	90% = 166.58 s

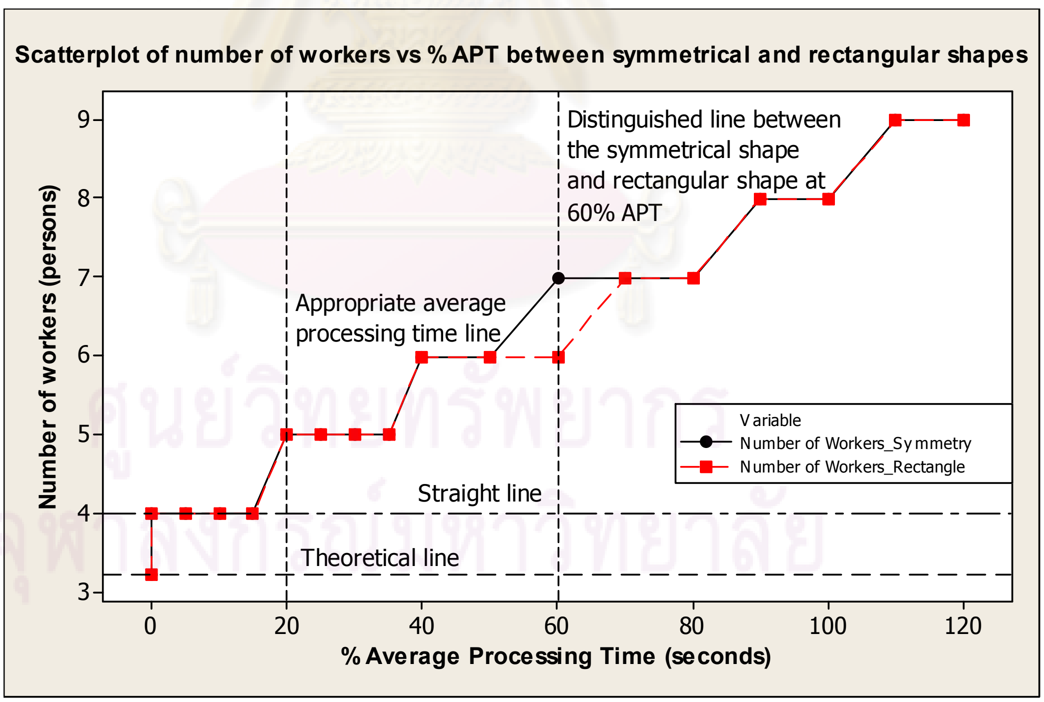


Figure 4.10 Appropriate average processing time line and distinguished line between symmetrical and rectangular layouts for the 19-task problem

CHAPTER V

EVOLUTIONARY ALGORITHMS

5.1 Introduction

This chapter describes the experimental approaches that use multi-objective evolutionary solution concepts, a case study and a comparative method. A multi-objective optimization is related to the problem in which two or more two objectives have to be optimized at the same time. The multi-objective evolutionary algorithms (MOEAs) have been applied to a wide range of problems from social to engineering problems over almost two decades. Although several versions of MOEAs have been developed to find multiple Pareto-optimal solutions in one single simulation run, the non-dominated sorting genetic algorithm (NSGA-II) is among the most favorable evolutionary algorithm in terms of convergence speed and distribution of the Pareto frontier. To perform more effective good Pareto-optimal solutions, memetic algorithm (MA) combines the exploration of population-based evolutionary adaptation and the exploitation of individual local search learning. However, most well-known algorithms such as NSGA-II and MA search only for good solutions by sampling through crossover and mutation operations without the useful exploitation of bad solutions and the internal structure of order pairs of permutation solutions. Consequently, the combinatorial optimization with coincidence (COIN) for solving permutation problems, e.g. traveling salesman problems and the particle swarm optimization with negative knowledge (PSONK) are developed to fill in the gaps.

The outline of this chapter is in the following. After the components of experiments are presented, the main sections of evolutionary algorithms, i.e. NSGA-II, MA, COIN, and PSONK are proposed in details including their demonstrative examples. At the end of this chapter, the performance measures of multi-objective algorithms are explained and exemplified. It is remarkable that if today's computing devices cannot solve large-sized problems in an acceptable time, the development of faster computers would be able to solve a practical-sized problem in the future.

5.2 Evolutionary Algorithms Development

In this section, the solution methods of NSGA-II, MA, COIN, and PSONK are adopted, developed and exemplified to make final results and conclusion. The computation of study performs all experiments using MATLAB R2008a. The test environment is run on AMD Athlon™ 64 Processor 3500+ 2.21GHz with 960 MB DDR-SDRAM.

5.3 Components of Initial Sample Experiments

Initial experiments give us the exemplified results of optimal worker allocation for each algorithm with determined parameters.

Input parameters

There are a few problems are exemplified in this section. In any time period, the number of jobs is deterministic and job arrivals come from not only new customer orders but also remaining jobs from the previous planning period that were not completed. Each job is an entity worked on many tasks. No job priority (i.e. no preemption job) constraint is allowed: that is, each job is allowed to start its processing whenever it is ready. These jobs are sorted by the daily production order excluding the sequencing problem. The precedence graphs of ten test-based problems and a case study (7-task to 297-task assembly networks) and various cycle times (the time which is available at each station to perform all the tasks assigned to the station) are input in U-shaped assembly line worker allocation problems. They are referred to in the last chapter. Given precedence graphs for an assembly line are produced from the process of making intermediate parts in the final assembly line. The 10-task problem is exemplified into NSGA-II, MA, and COIN. The 11-task precedence graph with the given cycle time of 13 seconds is also exemplified into PSONK.

Decision variables

First, both of fixed U-line layouts at the side ratios of 1:1:1 (1/3) and 1:4:4 (1/9) are used at the same task location of front, back and side for 7-task and 10-task problems. Other problems are different between both layouts. Secondly, a random priority rule or a priority-based encoding method is used like Hwang *et al.* (2008). The position of a gene was used to represent a task node, and the value of the gene was used to represent the priority of the task node for constructing a task sequence among candidates. Finally, the worker movement rule of displacement is put into all experiments.

Performance measures

Each of task sequence distributed into a U-line is computed by a number of workers and the coordinate of DOW and WT. Finally, the Pareto frontier of the minimum number of workers (or the first rank) is illustrated in each of problems for all algorithms.

Optimizers

The algorithms of NSGA-II, MA, COIN, and PSONK are described in the following section.

5.4 Non-dominated Sorting Genetic Algorithms-II (NSGA-II)

Deb *et al.* (2002) suggested a nondominated sorting-based multiobjective evolutionary algorithm (MOEA), named Non-dominated Sorting Genetic Algorithm-II (NSGA-II). The algorithm of NSGA-II can be stated as follows.

1. To create an initial parent population of size N randomly;

2. To sort the population into several frontiers based on the fast non-dominated sorting algorithm;

3. To calculate a crowding distance measure for each solution;

4. To select the parent population into a mating pool based on the binary crowded tournament selection;

5. To apply crossover and mutation operators to create an offspring population of size N;

6. To combine the parent population with the offspring population and apply an elitist mechanism to the combined population of size $2N$ for a new population of size N;

7. To repeat the step 2 until the terminating condition is met.

The procedure of NSGA-II (ibid.) is shown in Figure 5.1.

5.4.1 Numerical example

The 10-task problem of the single product with 10 cycle time (time units) originated by Miltenburg (2001a) is used to elaborate the algorithm of NSGA-II. The manual task times are two time units for operations 3, 4, 8, 9, 10, three time units for operations 1, 7, and four time units for operations 2, 5, 6. The precedence constraints are (3,1), (5,10), and (6,9). The fixed U-shaped layout of the side, front, and back is 2, 4, and 4 respectively. The walking time from one task to another task is the five percentage of average processing time. Since the total operations time is 28, $0.05*(28/10) = 0.14$. As a result, the walking time matrix of 5% APT for each element $(x_{i,j})$ from one task x_i to another task x_j is shown in Table 5.1. Task assignment rule is randomized. Population size is ten chromosomes and two generations are described step by step as follows.

Table 5.1 The walking time matrix of 5% APT

x_i to x_j	1	2	3	4	5	6	7	8	9	10
1	0.00	0.14	0.28	0.42	0.50	0.53	0.50	0.40	0.31	0.28
2	0.14	0.00	0.14	0.28	0.36	0.41	0.40	0.31	0.28	0.31
3	0.28	0.14	0.00	0.14	0.22	0.30	0.31	0.28	0.31	0.40
4	0.42	0.28	0.14	0.00	0.10	0.22	0.28	0.31	0.40	0.50
5	0.50	0.36	0.22	0.10	0.00	0.14	0.22	0.30	0.41	0.53
6	0.53	0.41	0.30	0.22	0.14	0.00	0.10	0.22	0.36	0.50
7	0.50	0.40	0.31	0.28	0.22	0.10	0.00	0.14	0.28	0.42
8	0.40	0.31	0.28	0.31	0.30	0.22	0.14	0.00	0.14	0.28
9	0.31	0.28	0.31	0.40	0.41	0.36	0.28	0.14	0.00	0.14
10	0.28	0.31	0.40	0.50	0.53	0.50	0.42	0.28	0.14	0.00

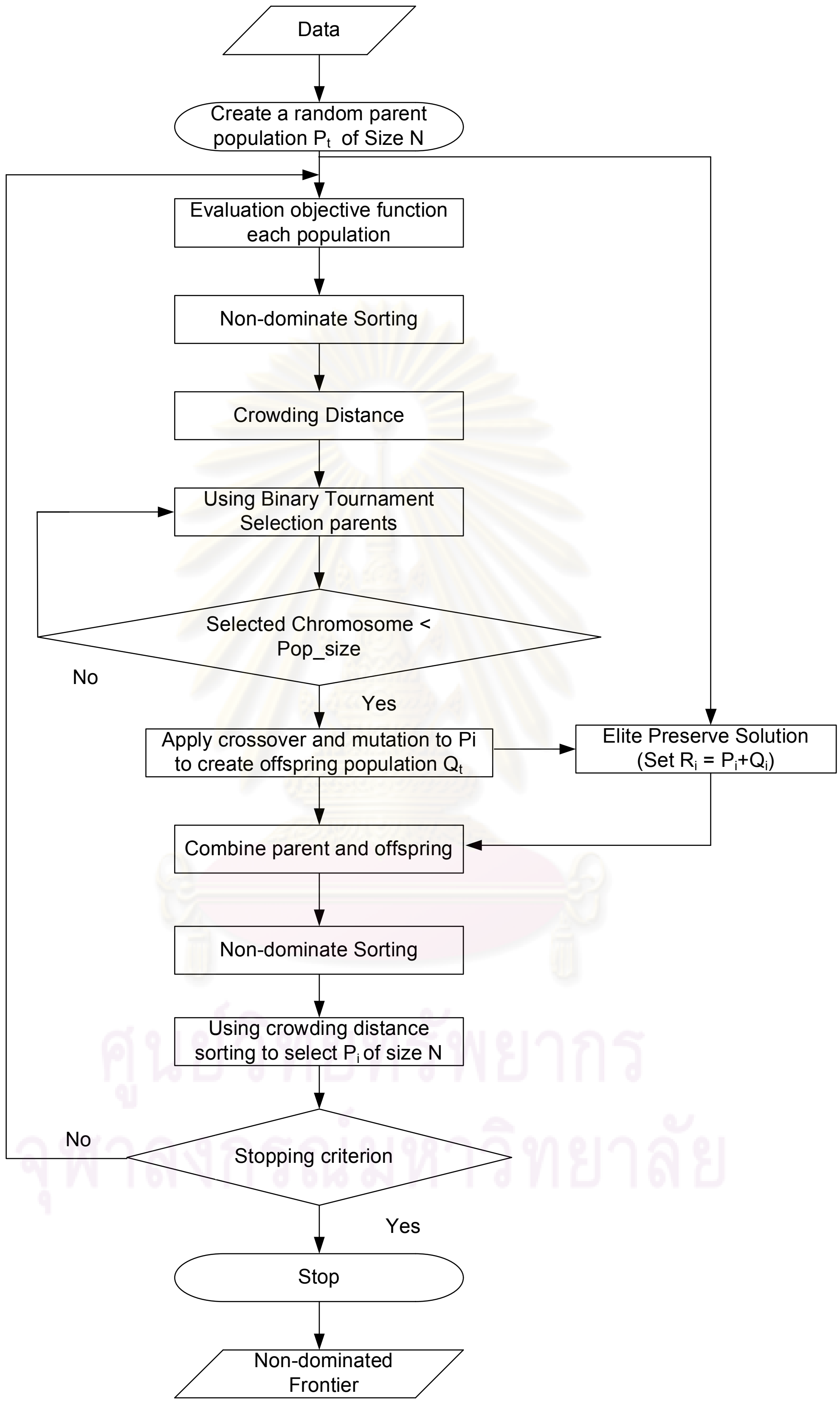


Figure 5.1 Procedure of Non-dominated Sorting Genetic Algorithm-II: NSGA-II

5.4.1.1 Population generation

The random parent population P_0 of size $N = 10$ chromosomes is created. To work effectively with precedence constraints, the priority-based encoding method (Gen and Cheng, 2000) is used. The position of a gene was used to represent a task node, and the value of the gene was used to represent the priority of the task node for constructing a task sequence among candidates. As is the proposed encoding method, first randomly generate the initial chromosome as shown in procedure 1 (Figure 5.2). Each chromosome position is called a gene. Each gene will use the priority of nodes in an assembly network. This encoding method easily verifies any permutation of the encoding to correspond to the sequences so that most existing genetic operators can easily be applied to the encoding. Consequently, the priority task of ten chromosomes is shown in Table 5.2.

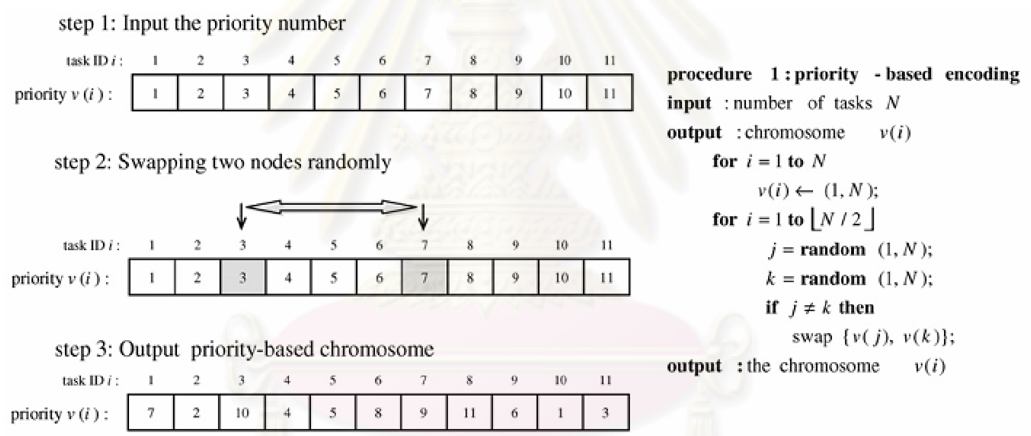


Figure 5.2 An example priority-based encoding procedure (Hwang *et al.*, 2008)

Table 5.2 Ten chromosomes by the priority-based encoding method

Task	1	2	3	4	5	6	7	8	9	10
N	Priority task									
1	1	7	9	10	2	8	6	4	3	5
2	5	3	4	6	7	8	1	2	10	9
3	7	8	4	3	9	6	10	1	5	2
4	10	2	8	4	3	1	7	9	6	5
5	3	1	10	9	8	2	7	6	4	5
6	7	10	6	2	3	1	9	8	4	5
7	4	10	9	5	1	8	3	6	2	7
8	10	8	5	1	3	9	7	2	4	6
9	1	5	8	10	3	2	9	7	6	4
10	4	6	8	9	1	5	7	2	3	10

From the priority task, a chromosome is input into the Task Sequence procedure. Task Sequence (TS) is constrained by the front precedence matrix in Table 5.3 and the back precedence matrix in Table 5.4. Task assignment into U-line is the same as the previous chapter in Table 4.3. Consequently, TS chromosomes are shown in Table 5.5.

Table 5.3 The front precedence matrix of the 10-task problem

N	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	1
6	0	0	0	0	0	0	0	0	1	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0

Table 5.4 The back precedence matrix of the 10-task problem

N	1	2	3	4	5	6	7	8	9	10
1	0	0	1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	1	0	0	0	0
10	0	0	0	0	1	0	0	0	0	0

Table 5.5 Ten TS chromosomes (*L*1-*L*10) influenced by the front and back work

N	Task Sequence									
	1	2	3	4	5	6	7	8	9	10
*L*1	4	8	5	3	6	9	2	10	1	7
*L*2	10	2	9	1	8	7	6	4	5	3
*L*3	2	5	10	1	6	9	7	3	8	4
*L*4	6	5	7	9	2	10	4	8	1	3
*L*5	4	6	7	5	8	2	9	10	1	3
*L*6	8	4	5	9	1	3	7	6	10	2
*L*7	7	2	3	8	6	1	10	4	9	5
*L*8	7	3	9	8	5	6	2	1	4	10
*L*9	4	6	7	2	9	3	1	8	5	10
*L*10	3	2	10	7	1	9	8	5	6	4

5.4.1.2 Population evaluation

As tabulating an example of worker allocation in a single U-line in the chapter IV and with the five %APT of displacement distance from the determination of walking time, three objective functions are shown in Table 5.6.

Table 5.6 Objective functions of ten TS chromosomes at the first generation

Chromosome Number	Number of workers	DOW	WT	Pareto Frontier	Crowding Distance
L8	4	2.6404	2.1336	1	Infinite
L4	4	3.0844	1.7206	1	Infinite
L10	4	2.7660	2.4234	2	Infinite
L3	4	2.8630	2.3324	2	2.0000
L5	4	3.1341	1.7388	2	Infinite
L2	4	2.8331	2.4584	3	Infinite
L1	4	3.6619	1.9600	3	Infinite
L6	4	3.2481	2.4822	4	Infinite
L9	4	4.0852	1.9600	4	Infinite
L7	4	3.4085	2.5200	5	Infinite

5.4.1.3 Non-dominated sorting and crowding distance

Non-dominated sorting is then used to classify the population into a number of Pareto fronts. The first front is the best in the combined population. The archive is created by selecting fronts based on their rankings. If the number of individuals in the archive is smaller than the population size, the next front will be selected and so on. If adding front would result in the number of individuals in the archive exceeding the initial population size, a truncation operator is applied to that front based on the crowded tournament selection by which the winner of two same rank solutions is the one that has the greater crowding distance.

The diversity mechanism is exercised when many individuals of the current generation do not dominate each other and only some of them have to be selected. It calculates density information of each individual. The one with lower density has a higher chance to be selected since less non-dominated solutions are clustering around (higher diversity). The density estimation technique for NSGA II and MAI uses the crowding distance method (Deb *et al*., 2002).

5.4.1.4 Binary tournament selection

In the process of binary tournament selection, after doing non-dominated sorting fitness values are transformed to dummy fitness values from the minimum value to maximum value. The probability pi of qi and the cumulative probability are calculated. The roulette wheel and binary tournament selection are shown in Table 5.7 and Table 5.8. The number of chromosome (string) from each row in binary tournament selection is selected with higher dummy fitness, but higher crowding distance is chosen if the dummy fitness is not different. However, if the crowding distance is also the same, one of two chromosomes in that row is randomized.

Table 5.7 Roulette wheel

String No.	DOW	WT	Fitness Value	Dummy Fitness	Crowding Distance	pi	qi
L1	3.6619	1.9600	3	3	Infinite	0.0909	0.0909
L2	2.8331	2.4584	3	3	Infinite	0.0909	0.1818
L3	2.8630	2.3324	2	4	2.0000	0.1212	0.3030
L4	3.0844	1.7206	1	5	Infinite	0.1515	0.4545
L5	3.1341	1.7388	2	4	Infinite	0.1212	0.5757
L6	3.2481	2.4822	4	2	Infinite	0.0606	0.6363
L7	3.4085	2.5200	5	1	Infinite	0.0303	0.6666
L8	2.6404	2.1336	1	5	Infinite	0.1515	0.8181
L9	4.0852	1.9600	4	2	Infinite	0.0606	0.8787
L10	2.7660	2.4234	2	4	Infinite	0.1212	1
			Total	33		1	

Table 5.8 Binary tournament selection

No.	Population 1				Population 2				No_String Selected
	r1	qi>r1	No_string	Dummy	r2	qi>r2	No_string	Dummy	
1	0.2272	0.3030	L3	4	0.5163	0.5757	L5	4	**L5**
2	0.4582	0.5757	L5	4	0.7032	0.8181	L8	5	**L8**
3	0.5825	0.6363	L6	2	0.5092	0.5757	L5	4	**L5**
4	0.0743	0.0909	L1	3	0.1932	0.6363	L6	2	**L1**
5	0.7709	0.8181	L8	5	0.3139	0.4545	L4	5	**L8**
6	0.6382	0.6666	L7	1	0.9866	1	L10	4	**L10**
7	0.5029	0.5757	L5	4	0.9477	1	L10	4	**L5**
8	0.1131	0.1818	L2	3	0.8121	0.8181	L8	5	**L8**
9	0.1221	0.1818	L2	3	0.7627	0.8181	L8	5	**L8**
10	0.7218	0.8181	L8	5	0.6516	0.6666	L7	1	**L8**

After that, the solutions of parents are shown in Table 5.9.

Table 5.9 Chromosomes of parents

String selected	N	Task Seq									
		1	2	3	4	5	6	7	8	9	10
L5	1	4	6	7	5	8	2	9	10	1	3
L8	2	7	3	9	8	5	6	2	1	4	10
L5	3	4	6	7	5	8	2	9	10	1	3
L1	4	4	8	5	3	6	9	2	10	1	7
L8	5	7	3	9	8	5	6	2	1	4	10
L10	6	3	2	10	7	1	9	8	5	6	4
L5	7	4	6	7	5	8	2	9	10	1	3
L8	8	7	3	9	8	5	6	2	1	4	10
L8	9	7	3	9	8	5	6	2	1	4	10
L8	10	7	3	9	8	5	6	2	1	4	10

5.4.1.5 Crossover

From the initial parameter setting, the crossover probability is assumed to be 0.7. The weight mapping crossover (WMX) is used by $0.7 \times 10 = 7$ chromosomes. They are randomized as shown in Table 5.10.

Table 5.10 WMX crossover chromosomes

String selected	N	Task Seq									
		1	2	3	4	5	6	7	8	9	10
L10	1	3	2	10	7	1	9	8	5	6	4
L5	2	4	6	7	5	8	2	9	10	1	3
L5	3	4	6	7	5	8	2	9	10	1	3
L8	4	7	3	9	8	5	6	2	1	4	10
L8	5	7	3	9	8	5	6	2	1	4	10
L1	6	4	8	5	3	6	9	2	10	1	7
L8	7	7	3	9	8	5	6	2	1	4	10

Hereby the position-based crossover operator of the weight mapping crossover (WMX) (Hwang *et al.*, 2008) can be randomly selected as a two-point crossover of a real number string and a remapping by weight order of different real number strings. This WMX operator is shown in Figure 5.3. However, there are only three pairs of seven chromosomes essential to use all for making various solutions. Thus, one of them is randomized to make a new pair of chromosomes.

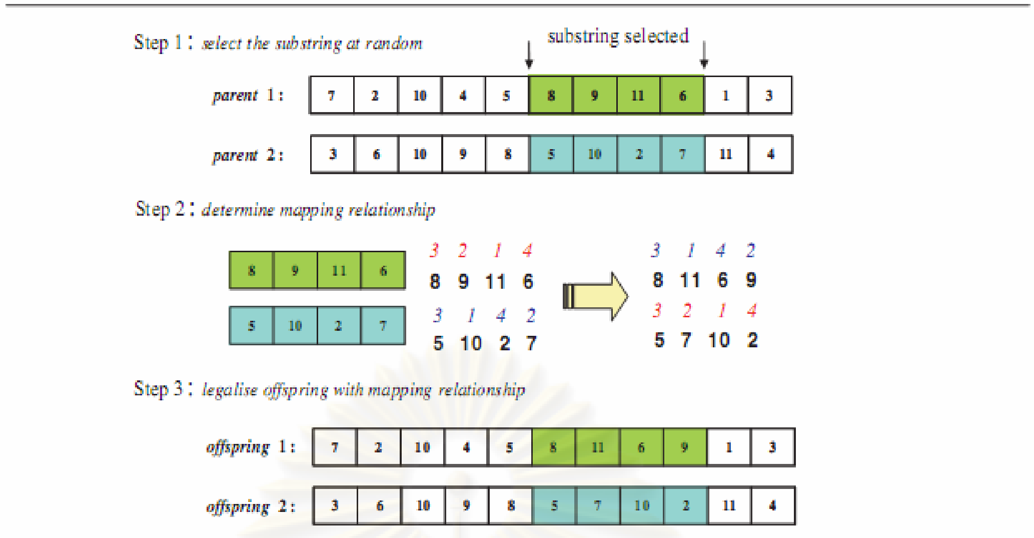


Figure 5.3 Weight mapping crossover (WMX)

After doing the procedure of weight mapping crossover, seven new chromosomes are shown in Table 5.11.

Table 5.11 Offspring after weight mapping crossover

String selected	N	Task Seq									
		1	2	3	4	5	6	7	8	9	10
L10*	1	3	2	10	7	8	1	9	5	6	4
L5*	2	4	6	7	5	2	9	8	10	1	3
L5*	3	4	5	8	7	6	2	9	10	1	3
L8*	4	7	5	8	3	9	6	2	1	4	10
L8*	5	7	3	9	8	5	2	4	1	6	10
L1*	6	4	4	8	5	3	10	6	2	9	1
L8*	7	3	7	9	8	5	6	2	1	4	10
L5**	8	4	6	7	5	8	2	9	10	1	3
L8**	9	7	3	9	8	5	6	2	1	4	10
L8**	10	7	3	9	8	5	6	2	1	4	10

* New chromosomes after doing WMX crossover

** Old chromosomes that is not selected to do WMX crossover

5.4.1.6 Mutation

From the initial parameter setting, the mutation probability is assumed to be 0.3. The mutation (inversion) is used by $0.3 \times 10 = 3$ chromosomes.

After doing the procedure of mutation crossover, one new randomized chromosome is shown in Table 5.12.

Table 5.12 Offspring after mutation

String selected	N	Task Seq									
		1	2	3	4	5	6	7	8	9	10
L10*	1	3	2	10	7	8	1	9	5	6	4
L5*	2	4	6	7	5	2	9	8	10	1	3
L5*	3	4	5	8	7	6	2	9	10	1	3
L8*	4	7	5	8	3	9	6	2	1	4	10
L8*	5	7	**2**	9	8	5	**3**	4	1	6	10
L1*	6	4	4	8	5	3	10	6	2	9	1
L8*	7	3	7	9	8	5	6	2	1	4	10
L5**	8	4	6	7	5	8	2	9	10	1	3
L8**	9	7	3	9	8	5	6	2	1	4	10
L8**	10	7	3	9	8	5	6	2	1	4	10

To combine ten chromosomes of parents and ten chromosomes of offspring, $Rt = Pt \cup Qt$ as shown in Table 5.13.

Table 5.13 Combination of parents and offspring chromosomes

$Rt = Pt \cup Qt$	N	Task Seq									
		1	2	3	4	5	6	7	8	9	10
Pt	1	4	6	7	5	8	2	9	10	1	3
	2	7	3	9	8	5	6	2	1	4	10
	3	4	6	7	5	8	2	9	10	1	3
	4	4	8	5	3	6	9	2	10	1	7
	5	7	3	9	8	5	6	2	1	4	10
	6	3	2	10	7	1	9	8	5	6	4
	7	4	6	7	5	8	2	9	10	1	3
	8	7	3	9	8	5	6	2	1	4	10
	9	7	3	9	8	5	6	2	1	4	10
	10	7	3	9	8	5	6	2	1	4	10
Qt	11	3	2	10	7	8	1	9	5	6	4
	12	4	6	7	5	2	9	8	10	1	3
	13	4	5	8	7	6	2	9	10	1	3
	14	7	5	8	3	9	6	2	1	4	10
	15	7	2	9	8	5	3	4	1	6	10
	16	4	4	8	5	3	10	6	2	9	1
	17	3	7	9	8	5	6	2	1	4	10
	18	4	6	7	5	8	2	9	10	1	3
	19	7	3	9	8	5	6	2	1	4	10
	20	7	3	9	8	5	6	2	1	4	10

Using the fast non-dominated sorting algorithm, the non-dominated fronts F1,F2,…,Fk in Rt are identified. Values of crowding distance for all chromosomes are identified. They are shown in Table 5.14 and Figure 5.4.

Table 5.14 Non-dominated sorting and crowing distance of parents and offspring at the first generation

Chromosome No.	DOW	WT	Fitness Value	Crowding Distance
2	2.6404	2.1336	1	Infinite
17	2.6404	2.1336	1	Infinite
5	2.6404	2.1336	1	Infinite
8	2.6404	2.1336	1	Infinite
9	2.6404	2.1336	1	Infinite
10	2.6404	2.1336	1	Infinite
19	2.6404	2.1336	1	Infinite
20	2.6404	2.1336	1	Infinite
13	2.9397	1.7388	1	Infinite
11	2.6744	2.4682	2	Infinite
6	2.7660	2.4234	2	2.0000
12	3.0921	1.7388	2	Infinite
16	3.1049	2.0930	3	Infinite
1	3.1341	1.7388	3	Infinite
18	3.1341	1.7388	3	Infinite
3	3.1341	1.7388	3	Infinite
7	3.1341	1.7388	3	Infinite
4	3.6619	1.9600	4	Infinite
14	4.0382	2.9036	5	Infinite
15	4.0845	2.0342	5	Infinite

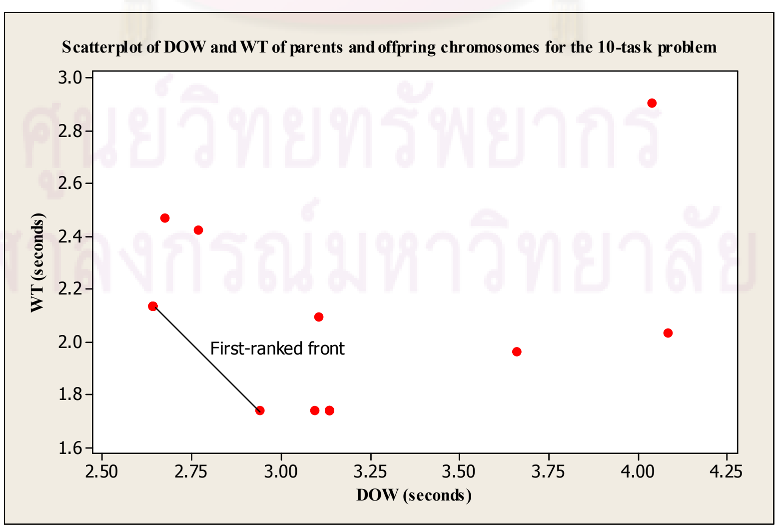


Figure 5.4 Scatterplot of DOW and WT of parents and offpring chromosomes for the 10-task problem

5.4.1.7 Next generation

In the second generation ($Pt+1$), first ten chromosomes with solving non-dominated sorting and crowding distance shown in Table 5.15 are input instead of the initial population from the first generation. If fitness value and crowding distance are the same, a chromosome is randomly selected. For example, at the tenth position of population, the chromosome 12 is selected and the chromosome 11 is discarded.

Table 5.15 Ten chromosomes ($Pt+1$) used in the second generation

No.	Chromosome No.	Task Seq									
		1	2	3	4	5	6	7	8	9	10
1	2	7	3	9	8	5	6	2	1	4	10
2	17	3	7	9	8	5	6	2	1	4	10
3	5	7	3	9	8	5	6	2	1	4	10
4	8	7	3	9	8	5	6	2	1	4	10
5	9	7	3	9	8	5	6	2	1	4	10
6	10	7	3	9	8	5	6	2	1	4	10
7	19	7	3	9	8	5	6	2	1	4	10
8	20	7	3	9	8	5	6	2	1	4	10
9	13	4	5	8	7	6	2	9	10	1	3
10	12	4	6	7	5	2	9	8	10	1	3

5.4.1.8 Elitism strategy

Elitism is the mechanism of constantly updating and keeping the best solutions found so far. An archive with a fixed number of elitists is established. NSGA II sets the archive size equal to the initial population size. The current archive is determined by combining the current archive and the previous archive. The non-dominated individuals generated by the combined elitists are considered as a set of tentative elitists. These individuals are added to the original archive. The non-dominated solutions residing in the archive are updated and the dominated ones are discarded.

Only the different solutions in the first non-dominated frontier are filled in the current elitist list. If the number of solutions in the first non-dominated frontier is less than or equal to the size of the elitist list, the new elitist list will contain all solutions of the first non-dominated frontier. For example, three different solutions at the first-ranked fitness of the first generation are shown in Table 5.16. Otherwise, two solutions from the first non-dominated solutions are randomly selected and then the solution with larger crowding distance measure and not being selected before is added to the new elitist list. This approach not only ensures that all solutions in the elitist list are non-dominated solutions but also promoting diversity of the solutions.

Table 5.16 Elitist solutions at the first generation

No.	Chromosome No.	Task Seq									
		1	2	3	4	5	6	7	8	9	10
1	13	4	5	8	7	6	2	9	10	1	3
2	17	3	7	9	8	5	6	2	1	4	10
3	20	7	3	9	8	5	6	2	1	4	10

After that, the steps of population evaluation to elitism strategy are repeated in every generation until the terminating condition is met.

5.4.2 Exemplified results

An example of 100 strings at Max.100 generations for the displacement rule is shown in Figure 5.5. Data of 100 strings at the first generation and the Pareto-optimal frontier of 30 strings at Max. 100 generations (the final frontier) are shown in Figure 5.6, in which the scatter plot including Pearson Correlation -0.965 and P-value = 0 that make confident that two conflicting objectives are the Min.-Min. problem. Experimental results of final task sequence; front and back task position; DOW, WT and number of workers are the section of answers.

Example

Solution: Run NSGAII algorithm at (10 tasks, 30 strings, 10 cycle time, 100 gen., 0.7 Pc, 0.3 Pm, 35% APT = 1 second)

Answers:

WT_DOW =

1.0981 17.3200 5.0000

1.3736 15.6600 5.0000

1.5384	14.8400	5.0000
1.5688	14.5200	5.0000
1.6733	14.0000	5.0000
2.0702	12.5000	5.0000
2.2500	10.8400	5.0000
2.4495	10.0000	5.0000
3.1988	9.7100	5.0000
3.2249	9.4200	5.0000
4.1710	9.1600	6.0000
4.2238	8.5800	6.0000
4.3874	7.4200	6.0000
4.4059	6.8400	6.0000
5.5291	6.0000	7.0000
5.5634	5.4200	7.0000
5.6121	4.8400	7.0000
6.2249	4.0000	8.0000

task_seq =

1	8	2	4	9	3	7	10	5	6
10	7	1	3	8	2	9	4	6	5
4	1	3	9	10	8	5	6	2	7
3	10	8	2	1	5	4	9	6	7
10	7	9	4	3	5	8	2	6	1
3	5	10	9	2	8	4	6	7	1
10	4	8	2	3	5	7	1	9	6
4	5	6	9	7	3	8	2	10	1
5	3	8	10	4	6	9	1	7	2
4	2	5	9	1	7	6	3	10	8

task_pos =

2	1	1	1	2	1	1	2	1	1
2	1	2	1	1	1	2	1	1	1
1	2	1	2	2	1	1	1	1	1
1	2	1	1	1	1	1	2	1	1
2	1	2	1	1	1	1	1	1	1

1	1	1	2	1	1	1	1	1	1
2	1	1	1	1	1	1	1	2	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	2	2	1	1	1	1	1

define_station =

1	1	2	2	3	3	4	4	5	5
1	1	2	2	3	3	4	4	5	5
1	1	2	2	3	3	4	4	5	5
1	1	2	2	3	3	4	4	5	5
1	1	2	2	3	3	4	4	5	5
1	1	2	2	3	3	4	4	5	5
1	1	2	2	3	3	4	4	5	5
1	1	2	2	3	3	4	4	5	5
1	1	2	2	2	3	3	4	4	5
1	1	2	3	3	4	4	5	5	5

WT_DOW_J =

1.0981	17.3200	5.0000
1.3736	15.6600	5.0000
1.5384	14.8400	5.0000
1.5688	14.5200	5.0000
1.6733	14.0000	5.0000
2.0702	12.5000	5.0000
2.2500	10.8400	5.0000
2.4495	10.0000	5.0000
3.1988	9.7100	5.0000
3.2249	9.4200	5.0000

Elapsed time is 2231.032257 seconds.

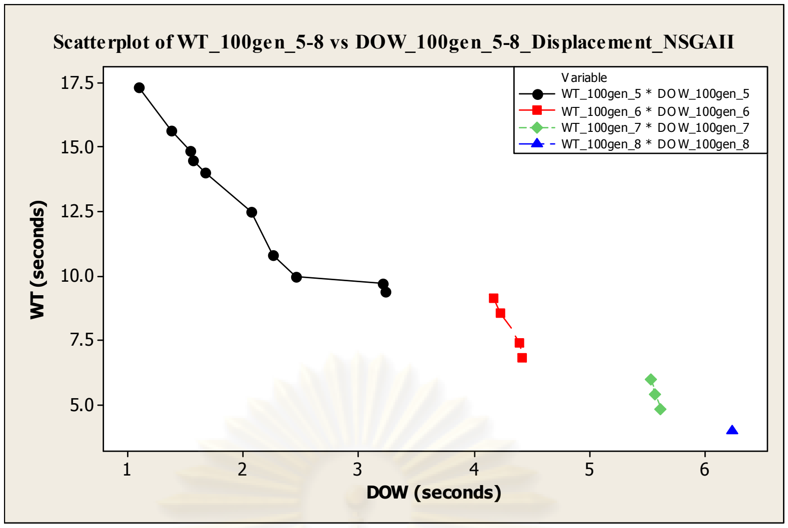


Figure 5.5 DOW vs. WT for 5, 6, 7 and 8 workers at 30 strings and 100 gen. (Pc=0.7, Pm=0.3)

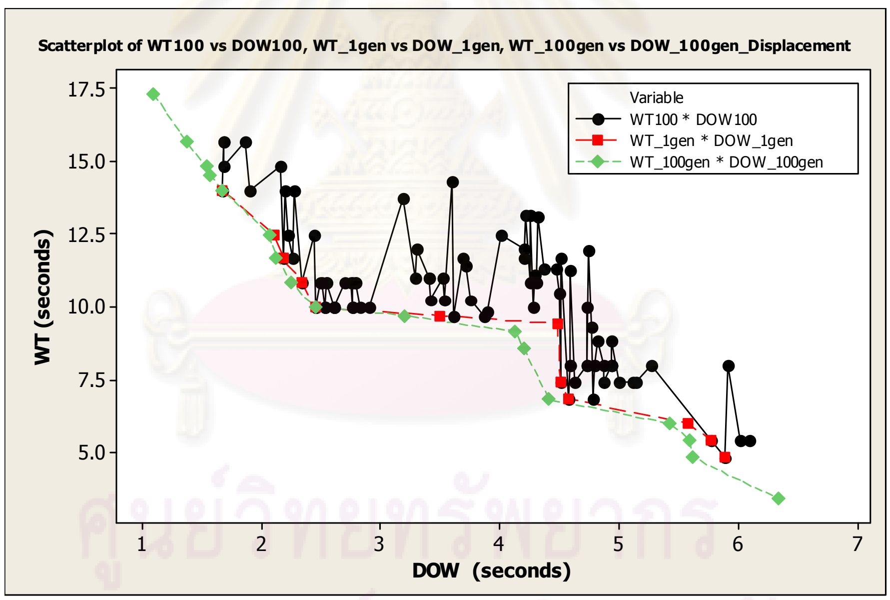


Figure 5.6 DOW vs. WT for 5, 6 and 7 workers at 100 strings and 1 gen. (Pc=0.7, Pm=0.3) 'compared with' DOW vs. WT for 5, 6, 7 and 8 workers at 30 strings and 100 gen. (Pc=0.7, Pm=0.3)

5.5 Memetic Algorithms (MA)

MA or MNSGA-II is a memetic version of NSGA-II. Appropriate local searches can additionally embed into several positions of the NSGA-II's algorithm, i.e. after initial population, after crossover, and after mutation (Lacomme *et al*., 2004). The number of places to apply local search has a direct effect on the quality of solution and computation time. Hence, if computation time needs to be saved, local search should be taken only at some specific steps in the algorithm of MA rather than at all possible steps. In this research, local search is chosen after obtaining initial solution and after mutation since pilot experiments and our previous research (Chutima and Pinkoompee, 2008) indicate that these two points are enough to find significantly improved solutions, pull the solutions out of the local optimal, and reduce computational time. The algorithm of MA can be stated as follows.

1. To create an initial parent population of size N randomly;

2. To apply a local search to the initial parent population;

3. To sort the population into several frontiers based on the fast non-dominated sorting algorithm;

4. To calculate a crowding distance measure for each solution;

5. To select the parent population into a mating pool based on the binary crowded tournament selection;

6. To apply crossover and mutation operators to create an offspring population of size N;

7. To apply a local search to the offspring population;

8. To combine the parent population with the offspring population and apply an elitist mechanism to the combined population of size $2N$ for a new population of size N;

9. To repeat the step 3 until the terminating condition is met.

The procedure of MA (Chutima and Pinkoompee, 2008) is shown in Figure 5.7.

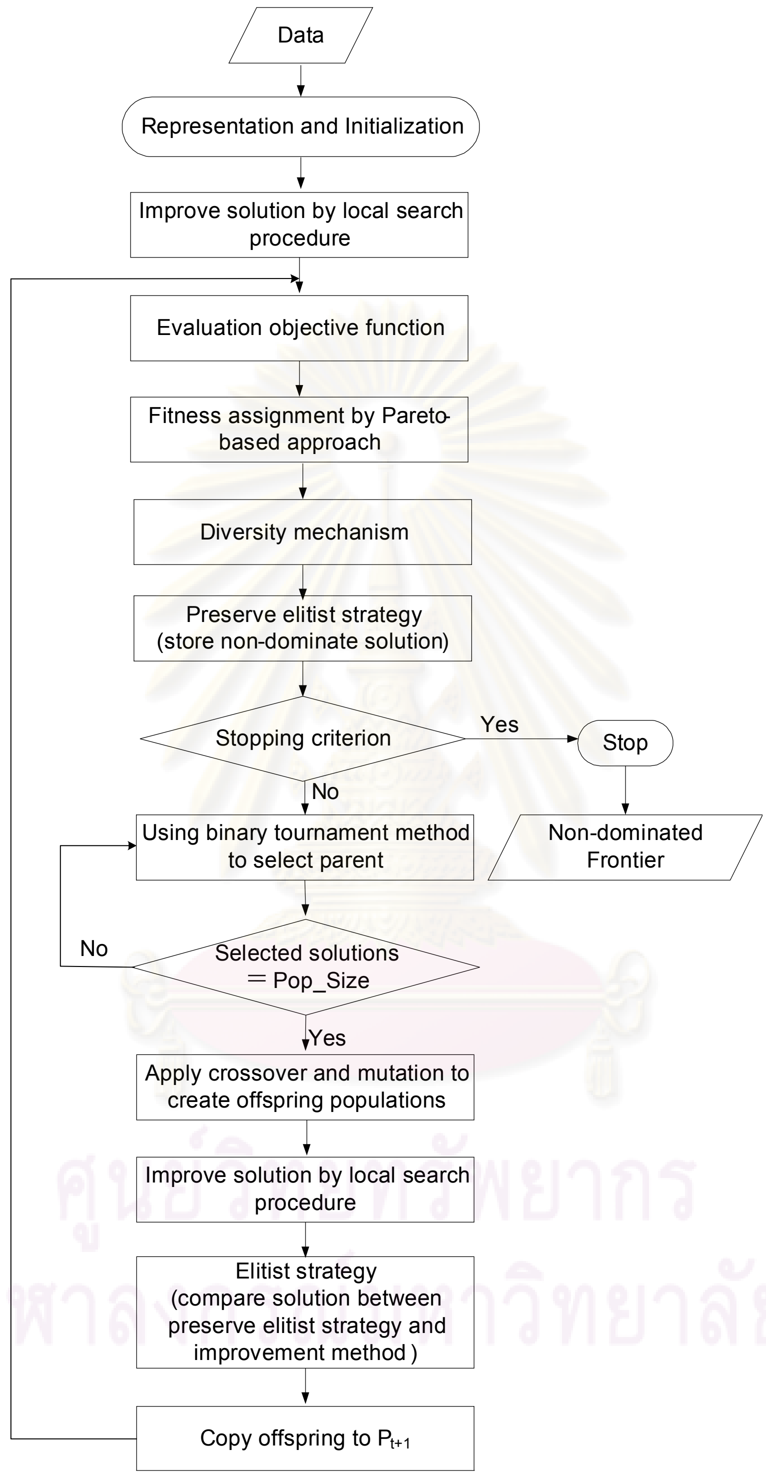


Figure 5.7 Procedure of Memetic Algorithms: MA

Four local searches modified from Kumar and Singh (2007) originally developed to solve traveling salesman problems by repeatedly exchanging edges of the tour until no improvement is attained are examined including Pairwise Interchange (PI), Insertion Procedures (IP), 2-Opt, and 3-Opt. Three criteria are used to test whether to accept a move that a local search heuristic creates a neighbor solution from the current solution as follows: (1) to accept the new solution if $f_1(x)$ is descendent, (2) to accept the new solution if $f_2(x)$ is descendent, and (3) to accept the new solution if $f_1(x)$ is the same or descendent and $f_2(x)$ is descendent; or to accept the new solution if $f_2(x)$ is the same or descendent and $f_1(x)$ is descendent.

The local search in the initial experiments is the 2-opt method that is one of several local searches from Kumar and Singh (2007). A neighboring solution (2-opt) is obtained by selecting two arbitrary products *i* and *j* and interchange them as shown in Figure 5.8.

Individual A

Position	1	2	3	4	5	6	7	8	9	10	11	12
	A	B	E	C	A	A	D	E	D	B	C	B

Exchange position 3, 12

Individual A'

A	B	B	C	A	A	D	E	D	B	C	E

Figure 5.8 Procedure of 2-opt local search

5.5.1 Numerical example

The core procedure and illustrative examples are not different from NSGA-II. Only the step of Local Search is put in two positions after doing the initial parent population and offspring population. Assumed that probability is equal to 0.2 that means 20% of all solutions will be subjected to local search. In addition, if the solutions on which the local search is applied are randomly selected, the improved quality of the new solutions may not be guaranteed. Hence, to select an appropriate solution to apply the local search, binary tournament selection is used.

5.5.1.1 Local search after initial population

Initial population generation is the first step in the proposed MA. A set of ten exemplified chromosomes as shown in Table 5.17 is generated randomly as an initial set of populations. First, local search after the initial population is exemplified.

Table 5.17 Initial parent population

N	Task Sequence									
	1	2	3	4	5	6	7	8	9	10
1	4	8	5	3	6	9	2	10	1	7
2	10	2	9	1	8	7	6	4	5	3
3	2	5	10	1	6	9	7	3	8	4
4	6	5	7	9	2	10	4	8	1	3
5	4	6	7	5	8	2	9	10	1	3
6	8	4	5	9	1	3	7	6	10	2
7	7	2	3	8	6	1	10	4	9	5
8	7	3	9	8	5	6	2	1	4	10
9	4	6	7	2	9	3	1	8	5	10
10	3	2	10	7	1	9	8	5	6	4

Local search after initial population by the method of Pairwise Interchange (PI) is done only once. PI selects two arbitrary tasks located at positions i and j, $i \neq j$, and interchange them to generate a neighboring solution. All possible swaps of pairs of tasks in a given solution are feasible. e.g.:

Parent 4 8 5 3 6 9 2 10 1 7

Neighbor 4 8 5 2 6 9 3 10 1 7

For example, eight chromosomes are selected with the procedure of binary tournament as shown in Table 5.18. Then, two neighboring solutions from local search with PI are shown in Table 5.19.

Table 5.18 Two chromosomes from binary tournament selection

N	Task Sequence									
	1	2	3	4	5	6	7	8	9	10
2	*10*	*2*	*9*	*1*	*8*	*7*	*6*	*4*	*5*	*3*
5	*4*	*6*	*7*	*5*	*8*	*2*	*9*	*10*	*1*	*3*
1	4	8	5	3	6	9	2	10	1	7
3	2	5	10	1	6	9	7	3	8	4
4	6	5	7	9	2	10	4	8	1	3
5	4	6	7	5	8	2	9	10	1	3
6	8	4	5	9	1	3	7	6	10	2
7	7	2	3	8	6	1	10	4	9	5
8	7	3	9	8	5	6	2	1	4	10
9	4	6	7	2	9	3	1	8	5	10
10	3	2	10	7	1	9	8	5	6	4

Table 5.19 Two neighboring solutions from local search with PI

N	Task Sequence									
	1	2	3	4	5	6	7	8	9	10
L2	*10*	*7*	*9*	*1*	*8*	*2*	*6*	*4*	*5*	*3*
L5	*4*	*6*	*9*	*5*	*8*	*2*	*7*	*10*	*1*	*3*
1	4	8	5	3	6	9	2	10	1	7
3	2	5	10	1	6	9	7	3	8	4
4	6	5	7	9	2	10	4	8	1	3
5	4	6	7	5	8	2	9	10	1	3
6	8	4	5	9	1	3	7	6	10	2
7	7	2	3	8	6	1	10	4	9	5
8	7	3	9	8	5	6	2	1	4	10
9	4	6	7	2	9	3	1	8	5	10
10	3	2	10	7	1	9	8	5	6	4

After that, each of two selected chromosomes is compared between a chromosome and a neighboring solution with two objective functions by previous acceptance rules. If chromosome L2 and L5 are accepted rather than chromosome 2 and 5 by those acceptance rules, Table 5.19 is input to the crossover operator.

5.5.1.2 Local search after offspring

Secondly, local search after the mutation population is exemplified. Ten chromosomes of offspring population obtained after doing mutation are exemplified and shown in Table 5.20.

Table 5.20 Offspring population

String selected	N	Task Seq									
		1	2	3	4	5	6	7	8	9	10
10	1	3	2	10	7	8	1	9	5	6	4
5	2	4	6	7	5	2	9	8	10	1	3
5	3	4	5	8	7	6	2	9	10	1	3
8	4	7	5	8	3	9	6	2	1	4	10
8	5	7	2	9	8	5	3	4	1	6	10
1	6	4	4	8	5	3	10	6	2	9	1
8	7	3	7	9	8	5	6	2	1	4	10
5	8	4	6	7	5	8	2	9	10	1	3
8	9	7	3	9	8	5	6	2	1	4	10
8	10	7	3	9	8	5	6	2	1	4	10

Local search after mutation population by the method of Insertion Procedure (IP) is done in every generation. IP removes a task from one position i and then insert it back to any position j where $i \neq j$ of a given position. e.g.:

Parent 3 2 |10| 7 8 1 9 5 6 4

Neighbor 3 2 7 8 1 9 |10| 5 6 4

For example, two chromosomes are selected with the procedure of binary tournament as shown in Table 5.21. Then, two neighboring solutions from local search with PI are shown in Table 5.22.

Table 5.21 Two chromosomes from binary tournament selection

N	Task Sequence									
	1	2	3	4	5	6	7	8	9	10
1	3	2	10	7	8	1	9	5	6	4
2	4	6	7	5	2	9	8	10	1	3
3	4	5	8	7	6	2	9	10	1	3
4	7	5	8	3	9	6	2	1	4	10
5	7	2	9	8	5	3	4	1	6	10
6	4	4	8	5	3	10	6	2	9	1
7	3	7	9	8	5	6	2	1	4	10
8	4	6	7	5	8	2	9	10	1	3
9	7	3	9	8	5	6	2	1	4	10
10	7	3	9	8	5	6	2	1	4	10

Table 5.22 Two neighboring solutions from local search with PI

N	Task Sequence									
	1	2	3	4	5	6	7	8	9	10
L2	4	6	7	5	2	9	8	1	3	10
L3	4	5	7	6	2	8	9	10	1	3
1	3	2	10	7	8	1	9	5	6	4
4	7	5	8	3	9	6	2	1	4	10
5	7	2	9	8	5	3	4	1	6	10
6	4	4	8	5	3	10	6	2	9	1
7	3	7	9	8	5	6	2	1	4	10
8	4	6	7	5	8	2	9	10	1	3
9	7	3	9	8	5	6	2	1	4	10
10	7	3	9	8	5	6	2	1	4	10

After that, each of two selected chromosomes is compared between a chromosome and a neighboring solution with two objective functions by previous acceptance rules. If chromosome L2 and L3 are accepted rather than chromosome 2 and 3 by those acceptance rules, Table 5.22 is input to the procedure of evaluation population.

5.5.2 Exemplified results

Data of 100 strings at the first generation and the Pareto-optimal frontier of 100 strings at Max. 100 generations for the displacement rule are shown in Figure 5.9. Figure 5.10 illustrates 100 strings at the first generation for only five workers on the final Pareto-optimal frontier. Experimental results of final task sequence; front and back task position; DOW, WT and number of workers are the section of answers below.

<u>Example</u>

Solution: Run Memetic Algorithms at (10 tasks, 100 strings, 10 cycle time, 100 gen., 0.7 P_C, 0.3 P_M, P_L = 0.8, 35% APT = 1 second)

<u>Answers:</u>

WT_DOW =

1.0535 17.3200 5.0000

1.2978 16.1800 5.0000

1.3736 15.6600 5.0000

1.5384	14.8400	5.0000
1.5688	14.5200	5.0000
1.6733	14.0000	5.0000
2.0702	12.5000	5.0000
2.1257	11.6600	5.0000
2.2500	10.8400	5.0000
2.4495	10.0000	5.0000
3.2249	9.4200	5.0000
4.1244	9.1600	6.0000
4.1778	8.5800	6.0000
4.3331	7.4200	6.0000
4.4059	6.8400	6.0000
5.3184	6.0000	7.0000
5.5782	5.4200	7.0000
5.6121	4.8400	7.0000
6.3836	4.0000	8.0000

task_seq =

1	3	7	2	8	9	4	10	5	6
8	2	3	9	6	1	4	10	5	7
1	3	9	4	7	2	10	8	5	6
1	3	4	10	9	8	2	6	5	7
1	4	5	10	7	2	9	8	3	6
9	8	3	10	6	7	1	2	4	5
8	5	3	9	2	4	1	7	6	10
1	9	8	2	6	3	4	10	5	7
9	8	3	6	7	1	2	4	5	10
2	4	5	10	9	1	7	3	8	6
5	8	2	1	9	6	7	3	10	4

task_pos =

2	1	1	1	1	2	1	2	1	1
1	1	1	2	1	1	1	2	1	1
2	1	2	1	1	1	2	1	1	1
2	1	1	2	2	1	1	1	1	1

```
   2   1   1   1   1   1   2   1   1   1
   2   1   1   2   1   1   1   1   1   1
   1   1   1   2   1   1   1   1   1   1
   2   2   1   1   1   1   1   2   1   1
   2   1   1   1   1   1   1   1   1   1
   1   1   1   1   2   2   1   1   1   1
   1   1   1   2   2   1   1   1   1   1

define_station =

   1   1   2   2   3   3   4   4   5   5
   1   1   2   2   3   3   4   4   5   5
   1   1   2   2   3   3   4   4   5   5
   1   1   2   2   3   3   4   4   5   5
   1   1   2   2   3   3   4   4   5   5
   1   1   2   2   3   3   4   4   5   5
   1   1   2   2   3   3   4   4   5   5
   1   1   2   2   3   3   4   4   5   5
   1   1   2   2   3   3   4   4   5   5
   1   1   2   2   3   3   4   4   5   5
   1   1   2   3   3   4   4   5   5   5

WT_DOW_J =

   1.0535   17.3200    5.0000
   1.2978   16.1800    5.0000
   1.3736   15.6600    5.0000
   1.5384   14.8400    5.0000
   1.5688   14.5200    5.0000
   1.6733   14.0000    5.0000
   2.0702   12.5000    5.0000
   2.1257   11.6600    5.0000
   2.2500   10.8400    5.0000
   2.4495   10.0000    5.0000
   3.2249    9.4200    5.0000

Elapsed time is 1770.545626 seconds.
```

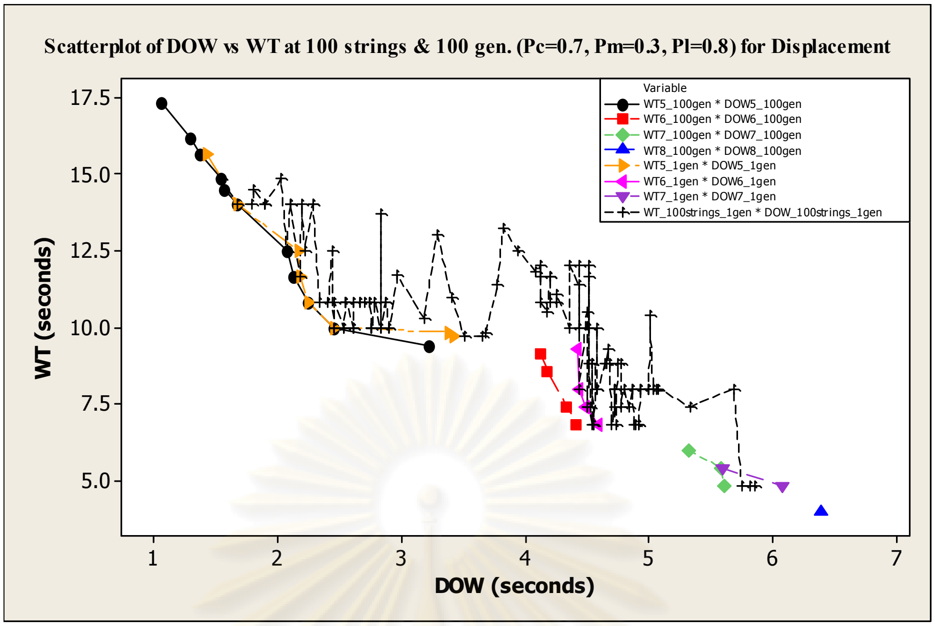


Figure 5.9 DOW vs. WT for 5, 6 and 7 workers at 100 strings and 1 gen. (Pc=0.7,Pm=0.3,Pl=0.8) 'compared with' DOW vs. WT for 5, 6, 7 and 8 workers at 100 strings and 100 gen. (Pc=0.7,Pm=0.3,Pl=0.8)

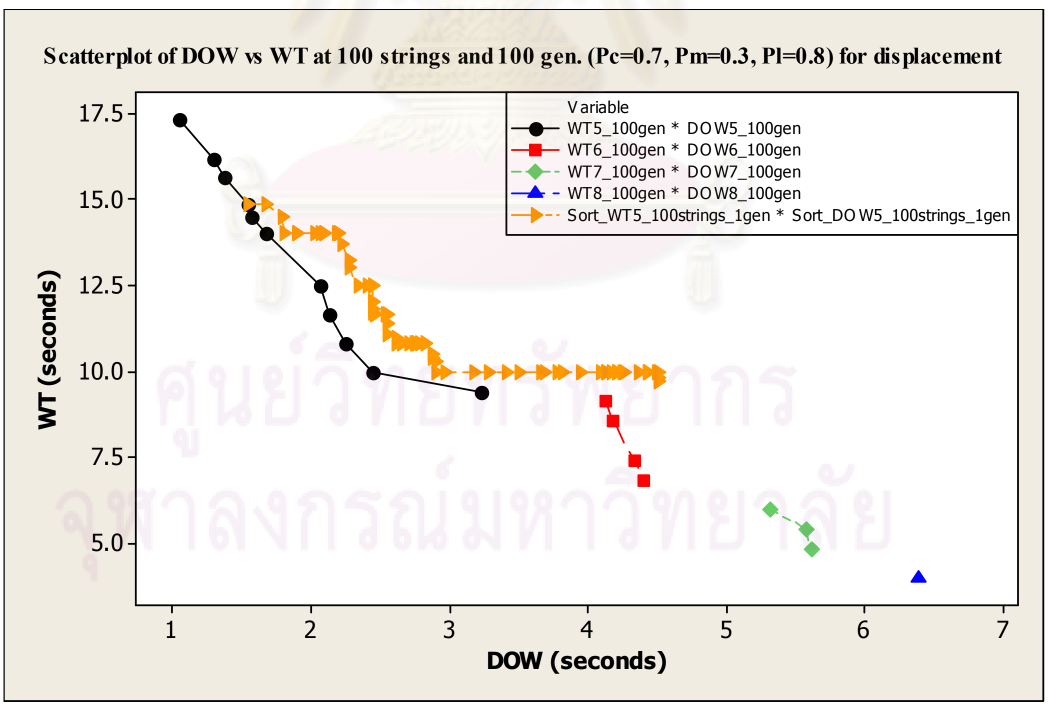


Figure 5.10 DOW vs. WT for 5 workers at 51 selected strings from first 100 strings and 1 gen. 'compared with' DOW vs. WT for 5, 6, 7 and 8 workers at 100 strings and 100 gen. (Pc=0.7,Pm=0.3,Pl=0.8)

5.6 COINcidence Algorithm (COIN)

Wattanapornprom *et al.* (2009) developed a new effective evolutionary algorithm called *combinatorial optimization with coincidence* (COIN) originally aiming for solving traveling salesman problems. The idea is that most well-known algorithms such as Genetic Algorithm (GA) searches for good solutions by sampling through crossover and mutation operations without much exploitation of the internal structure of good solution strings. This may not only generate large number of inefficient solutions dissipated over the solution space but also consume long CPU time. In contrast, COIN considers the internal structure of good solution strings and memorizes paths that could lead to good solutions. COIN replaces high computation time of crossover and mutation operations of GA and employs joint probability matrix as a means to generate neighborhood solutions. It prioritizes the selection of the paths with higher chances of moving towards good solutions.

Apart from traditional learning from good solutions, COIN allows learning from below average solutions as well. Any coincidence found in a situation can be statistically described whether the situation is good or bad. Most traditional algorithms always discard the bad solutions without utilizing any information associated with them. In contrast, COIN learns from the coincidence found in the bad solutions and uses this information to avoid such situations to recurrent; meanwhile, experiences from good coincidences are also used to construct better solutions in Figure 5.11. Consequently, the chances that the paths being parts of the bad solutions are always used in the new generations are lessened. This lowers the number of solutions to be considered and hence increases the convergence speed.

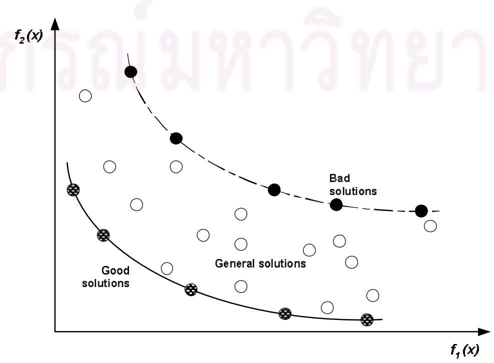


Figure 5.11 Good and bad solutions

COIN uses a joint probability matrix (generator) to create the population. The generator is initialized so that it can generate a random tree with equal probability for any configuration. The population is evaluated in the same way as traditional evolutionary algorithms. However, COIN uses both good and bad solutions to update the generator. Initially, COIN searches from a fully connected tree and then incrementally strengthening or weakening the connections. As generations pass by, the probabilities of selecting certain paths are increased or decreased depending on the incidences found in the good or bad solutions. The procedure of COINcidence algorithm is described in Figure 5.12 and can be stated as follows.

1. To initialize the joint probability matrix (generator);

2. To generate the population using the generator;

3. To evaluate the population;

4. To rank the population (Goldberg's Pareto ranking) and make diversity preservation;

5. To select the candidates according to two options: (a) good solution selection (select the solutions in the first rank of the current Pareto frontier), and (b) bad solution selection (select the solutions in the last rank of the current Pareto frontier);

6. For each joint probability matrix $H(x_i, x_j)$, to adjust the generator according to the reward and punishment scheme as Eq.(5.1);

$$x_{i,j}(t+1) = x_{i,j}(t) + \frac{k}{(n-1-np_i)}\{r_{i,j}(t+1) - p_{i,j}(t+1)\} + \frac{k}{(n-1-np_i)^2}\{\textstyle\sum_{j=1}^{n} p_{i,j}(t+1) - \sum_{j=1}^{n} r_{i,j}(t+1)\} \quad (5.1)$$

where $x_{i,j}$ = the element (i,j) of joint probability matrix $H(x_i/x_j)$, k = the learning coefficient, $r_{i,j}$ = the number of coincidences (x_i, x_j) found in the good solutions, $p_{i,j}$ = the number of coincidences (x_i, x_j) found in the bad solutions, t = generation number, n = the size of the problem, and np_i = the number of the direct predecessors of task i;

7. To apply a strategy to maintain elitist solutions in the population, and then repeat the step 2 until the terminating condition is met.

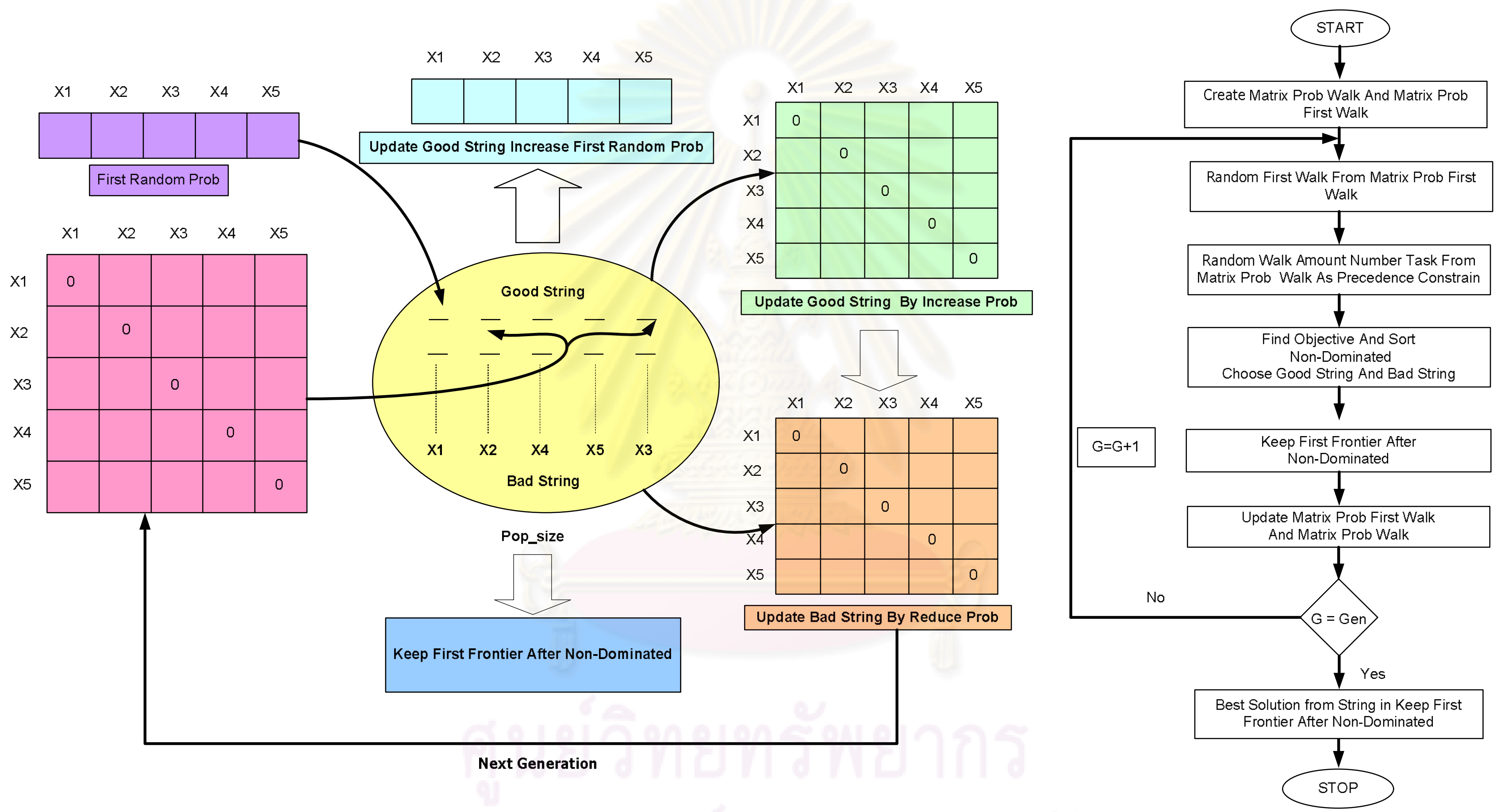


Figure 5.12 Flowchart of combinatorial optimization with coincidence algorithm

119

5.6.1 Numerical example

The 10-task problem of the single product with 10-minute cycle time originated by Miltenburg (2001a) is used to elaborate the algorithm of COIN. The manual task times are two time units for tasks 3, 4, 8, 9, 10, three time units for tasks 1, 7, and four time units for tasks 2, 5, 6. The precedence constraints are (3,1), (5,10), and (6,9). The fixed U-shaped layout of the side, front, and back is 2, 4, and 4 respectively. The walking time from one task to another task is five percent of average processing time. Since the total tasks time is $28,\ 0.05*(28/10) = 0.14$. As a result, the walking time matrix of 5% APT for each element $(x_{i,j})$ is shown in Table 5.23. The task assignment rule is randomized. Learning probability (k) and reward or punishment values are assumed to be 0.1. Population size is ten chromosomes and two generations are described step by step as follows.

Table 5.23 The walking time matrix of 5% APT

x_i to x_j	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	0.00	0.14	0.28	0.42	0.50	0.53	0.50	0.40	0.31	0.28
2	0.14	0.00	0.14	0.28	0.36	0.41	0.40	0.31	0.28	0.31
3	0.28	0.14	0.00	0.14	0.22	0.30	0.31	0.28	0.31	0.40
4	0.42	0.28	0.14	0.00	0.10	0.22	0.28	0.31	0.40	0.50
5	0.50	0.36	0.22	0.10	0.00	0.14	0.22	0.30	0.41	0.53
6	0.53	0.41	0.30	0.22	0.14	0.00	0.10	0.22	0.36	0.50
7	0.50	0.40	0.31	0.28	0.22	0.10	0.00	0.14	0.28	0.42
8	0.40	0.31	0.28	0.31	0.30	0.22	0.14	0.00	0.14	0.28
9	0.31	0.28	0.31	0.40	0.41	0.36	0.28	0.14	0.00	0.14
10	0.28	0.31	0.40	0.50	0.53	0.50	0.42	0.28	0.14	0.00

5.6.1.1 Joint probability matrix initialization

The number of tasks to be considered is 10. Therefore, the dimension of From-To joint probability matrix $H(x_i, x_j)$ is the matrix 10x10. The value of each element ($x_{i,j}$) in the matrix is the probability of selecting task j after task i. In order to incorporate some precedence relationship into the matrix, in each row the element which belongs to the direct predecessor of the task is set to 0 to prohibit producing such a task before its direct predecessor. For example, the direct predecessor of task 1 is task 3; hence, $x_{1,3}$= 0. Also, $x_{1,1}$= 0, since it cannot move within itself. Initially, the value of the remaining elements in the first row of the matrix are equal to $1/(n-1-np_1)=1/(10-1-1)=0.125$. Continuing this computation for all the remaining tasks (rows), the initial joint probability matrix is shown in Table 5.24.

Table 5.24 Initial joint probability matrix

Task	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	0.0000	0.1250	0.0000	0.1250	0.1250	0.1250	0.1250	0.1250	0.1250	0.1250
2	0.1111	0.0000	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111
3	0.1111	0.1111	0.0000	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111
4	0.1111	0.1111	0.1111	0.0000	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111
5	0.1111	0.1111	0.1111	0.1111	0.0000	0.1111	0.1111	0.1111	0.1111	0.1111
6	0.1111	0.1111	0.1111	0.1111	0.1111	0.0000	0.1111	0.1111	0.1111	0.1111
7	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111	0.0000	0.1111	0.1111	0.1111
8	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111	0.0000	0.1111	0.1111
9	0.1250	0.1250	0.1250	0.1250	0.1250	0.0000	0.1250	0.1250	0.0000	0.1250
10	0.1250	0.1250	0.1250	0.1250	0.0000	0.1250	0.1250	0.1250	0.1250	0.0000

5.6.1.2 Population generation

The order representation scheme is used to create chromosomes. The task order list in a chromosome is created by moving forward from one task to another task. If more than one possible task can be selected, the probability of selecting any task will depend on its value on the joint probability matrix. In each generation, the first task (or the first order pair) is selected from the current elitist of the first Pareto-ranked chromosome(s). The same probability of selection will be randomized. For example, task 7 is randomly selected for the first position. After selecting the row of task 7, the set of eligible tasks comprises tasks 1 to 6 and 8 to 10. From row 7 of the joint probability matrix, a task is randomly selected according to its probability of selection ($p_{7,j} = 0.1111, \text{ for } j = 1,2,3,4,5,6,8,9,10$). Supposing that task 6 is selected, the new set of eligible tasks becomes tasks 1 to 5 and 8 to 10. This mechanism is continued as long as all positions in the task order list are filled in and the task order list of L_1={7,6,2,3,1,4,8,9,5,10} is obtained. As the population size is assumed to be 10, the nine remaining initial population consists of chromosomes L_2={8,6,5,3,1,9,10,2,4,7}, L_3={5,4,6,8,9,3,2,1,7,10}, L_4={8,5,7,2,10,4,6,3,9,1}, L_5={3, 1,7,6,8,5,4,9,10,2}, L_6={3,7,4,8,5,6,1,10,2,9}, L_7={5,3,2,6,4,10,9,1,8,7}, L_8={4,8,7,5, 10,6,3,2,1,9}, L_9={2,4,7, 8,6,3, 9,5,10,1}, and L_{10}={2,3,6,8,4,1,7,5,10,9}.

5.6.1.3 Population evaluation

To find tentative tasks to be allocated on the U-line, all tasks have to be searched through the task order list in both forward and backward directions. The tentative task on forward or backward searching is found first. The task has its task time and walking time to the next task less than or equal to the remaining worker cycle time. If both forward and backward tentative tasks are found, either one is selected randomly. If any task from the task order list has not yet being allocated, a new workstation is opened. This procedure is repeated for the remaining task order list to obtain the number of workers (or workstations), walking time and worker load distribution for each of them. An example chromosome (L_1) is shown in Table 5.25. The deviation of operation times of workers is calculated from Eq. (4.2) with C_{jk} = 7.28, 9.46, 6.46, and 6.28 respectively. Thus, it is equal to 2.918 time units. The walking times of 0.14, 0.14, 0.14, 0.14, 0.10, 0.22, 0.10, 0.10, 0.14, 0.22, 0.14, and 0.14 are summed and the total walking time is equal to 1.72 time units. Having obtained feasible worker allocations three objectives have to be evaluated for each chromosome. Table 5.26 indicates that all chromosomes give the same number of workstations; therefore, all of them are eligible for Pareto ranking based on Deviation of Operation times of Workers (DOW) and Walking Time (WT) objectives. The Pareto ranking technique proposed by Goldberg (Deb *et al.*, 2002) is used to classify the population into non-dominated frontiers with a dummy fitness value that lower value is better. They are assigned to each chromosome in Figure 5.13.

Table 5.25 An example of worker allocation in a single U-line

Worker	Task considered Front graph	Task considered Back graph	Task assignment on a U-line			(3)	(4)	(5)	(5)-(3)-(4)
			Task	(1) WT (time unit)	(2) Task time (time unit)	Total (1)+ (2) Cycle time (time unit)	WT to Origin (time unit)	Given cycle time	Idle time (time unit)
1	7	- 10	7	-	3	3	-	10	7
1	6	- 10	6	0.14	4	4.14 [7.14]	0.14		2.72
2	2	- 10	2	-	4	4	-	10	6
2	3	- 10	3	0.14	2	2.14 [6.14]	0.14		3.72
2	1	- 10	1	0.10	3	3.10 [9.24]	0.22		0.54
3	4	- 10	4	-	2	2	-	10	8
3	8	- 10	8	0.10	2	2.10 [4.10]	0.10		5.80
3	9	- 10	9	0.14	2	2.14 [6.24]	0.22		3.54
4	5	- 10	5	-	4	4	-	10	6
4	-	- 10	10	0.14	2	2.14 [6.14]	0.14		3.72

Table 5.26 Objective functions of each chromosome from the first generation

Chromosome Number	Number of workers	DOW	WT	Pareto Frontier	Crowding Distance
L5	4	2.8608	1.7373	1	Infinite
L1	4	2.9179	1.7197	1	Infinite
L10	4	2.8608	1.7783	2	Infinite
L2*	4	3.1136	1.7197	2	Infinite
L3*	4	3.1136	1.7197	2	Infinite
L8	4	2.9605	1.7960	3	Infinite
L7	4	3.1297	1.8773	4	Infinite
L4**	4	3.6619	1.9593	5	Infinite
L9**	4	3.6619	1.9593	5	Infinite
L6	4	4.0981	2.2400	6	Infinite

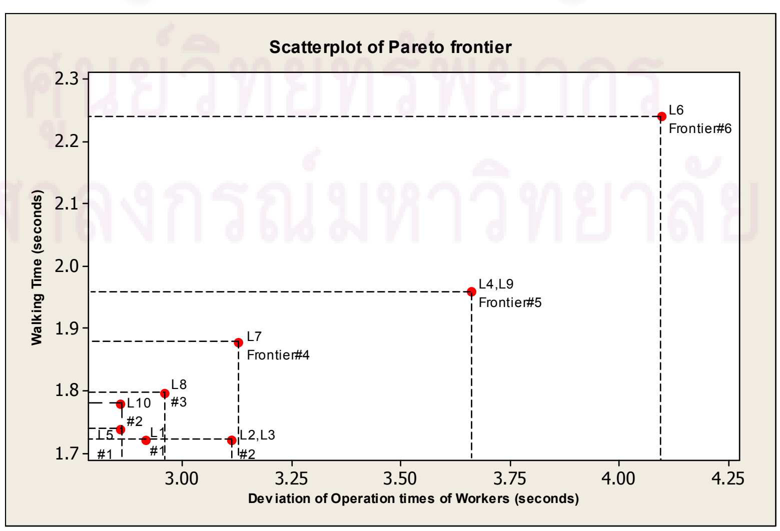


Figure 5.13 Pareto frontier of each chromosome

5.6.1.4 Diversity preservation

COIN employs a crowding distance approach (Deb *et al.*, 2002) to generate a diversified population uniformly spread over the Pareto frontier and avoids a genetic drift phenomenon (a few clusters of populations being formed in the solution space). The salient characteristic of this approach is that there is no need to define any parameter in calculating a measure of population density around a solution. The crowding distances computed for all solutions are infinite since at least two solutions are found for each frontier. Although both objectives of the chromosome L_2 are the same as L_3, and L_4 are the same as L_9, task sequence of each chromosome is different.

5.6.1.5 Solution selection

Having defined the Pareto frontier, the good solution is the chromosome located on the first Pareto frontier (dummy fitness = 1) and there is only one chromosome by the multiplication of reward value and population sizes, i.e. 0.1*10 = 1 solution. However, there are two solutions (L_1 and L_5) in the first rank. One of both is randomized, i.e. L_5 = {3,1,7,6,8,5,4,9,10,2}. In contrast, the bad solution is one located on the last Pareto frontier (dummy fitness = 6) and there is only one chromosome by the multiplication of punishment value and population sizes, i.e. 0.1*10 = 1, L_6={3,7,4,8,5,6,1,10,2,9}.

5.6.1.6 Joint probability matrix adjustment

The adjustment of the joint probability matrix is crucial to the performance of COIN. Reward will be given to $x_{i,j}$ if the order pair (i, j) is in the good solution to increase the chance of selection in the next round. For example, the first order pair (3,1) is the good solution of the chromosome L_5 = {3,1,7,6,8, 5,4,9,10,2}. Assumed that $k = 0.1$; therefore, the value of $x_{i,j}$ where $i = 3$ and $j = 1$ is increased by $k/(n-1-np_3) =$ $0.1/(10-1-0) = 0.0111$. The updated value of $x_{i,j}$ of the order pair (3,1) becomes $0.1111 + 0.0111 = 0.1222$. The values of the other order pairs located in the same row of the order pair (3,1) is reduced by

$k/(n-1-np_3)^2 = 0.1/(10-1-0)^2 = 0.1/81 = 0.0012$ For example, the value $x_{i,j}$ where $i=3$ and $j=1$ is 0.1222 - 0.0012 = 0.121. For the other positions of j in the third row, each adjusted joint probability is 0.1111 - 0.0012 = 0.1099. Previously, the summation of probability of $x_{3,j}$ is equal to one. Continuing this procedure to all order pairs located in the good solution; the revised joint probability matrix is obtained in Table 5.27.

Table 5.27 Revised joint probability matrix (good solution)

Task	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	0.0000	0.1238	0.0000	0.1238	0.1238	0.1238	0.1349	0.1238	0.1238	0.1238
2	0.1111	0.0000	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111
3	0.1210	0.1099	0.0000	0.1099	0.1099	0.1099	0.1099	0.1099	0.1099	0.1099
4	0.1099	0.1099	0.1099	0.0000	0.1099	0.1099	0.1099	0.1099	0.1210	0.1099
5	0.1099	0.1099	0.1099	0.1210	0.0000	0.1099	0.1099	0.1099	0.1099	0.1099
6	0.1099	0.1099	0.1099	0.1099	0.1099	0.0000	0.1099	0.1210	0.1099	0.1099
7	0.1099	0.1099	0.1099	0.1099	0.1099	0.1210	0.0000	0.1099	0.1099	0.1099
8	0.1099	0.1099	0.1099	0.1099	0.1210	0.1099	0.1099	0.0000	0.1099	0.1099
9	0.1238	0.1238	0.1238	0.1238	0.1238	0.0000	0.1238	0.1238	0.0000	0.1349
10	0.1238	0.1349	0.1238	0.1238	0.0000	0.1238	0.1238	0.1238	0.1238	0.0000

On the contrary, if the order pair (i,j) is in the bad solution, $x_{i,j}$ will be penalized to reduce the chance of selection in the next round. For example, the first order pair (3,7) is in the bad solution of the chromosome L_6={3,7,4,8,5,6,1,10,2,9}. Assuming that $k=0.1$; hence, the value of $x_{i,j}$ where $i=3$ and $j=7$ is decreased by $k/(n-1-np_3) = 0.1/(10-1-0) = 0.0111$. The updated value of $x_{i,j}$ of the order pair (3,7), which is later adjusted from Table 5.27 becomes 0.1099 – 0.0111 = 0.0988. The values of the other order pairs located in the same row of the order pair (3,7) is increased by $k/(n-1-np_3)^2 = 0.1/(10-1-0)^2 =$ $0.1/81 = 0.0012$. For example, the value $x_{i,j}$ where $i=3$ and $j=7$ is $0.0988 +$ $0.0012 = 0.1000$. For the position of $j=1$ in the 3rd row, the adjusted joint probability is 0.1210 + 0.0012 = 0.1222. For the other positions of j in the third row, each adjusted joint probability is $0.1099 + 0.0012 = 0.1111$. Continuing this procedure to all order pairs located in the bad solution, the revised joint probability matrix is obtained in Table 5.28.

Table 5.28 Revised joint probability matrix (bad solution)

Task	1	2	3	4	5	6	7	8	9	10
1	0.0000	0.1250	0.0000	0.1250	0.1250	0.1250	0.1361	0.1250	0.1250	0.1139
2	0.1123	0.0000	0.1123	0.1123	0.1123	0.1123	0.1123	0.1123	0.1012	0.1123
3	0.1222	0.1111	0.0000	0.1111	0.1111	0.1111	0.1000	0.1111	0.1111	0.1111
4	0.1111	0.1111	0.1111	0.0000	0.1111	0.1111	0.1111	0.1000	0.1222	0.1111
5	0.1111	0.1111	0.1111	0.1222	0.0000	0.1000	0.1111	0.1111	0.1111	0.1111
6	0.1000	0.1111	0.1111	0.1111	0.1111	0.0000	0.1111	0.1222	0.1111	0.1111
7	0.1111	0.1111	0.1111	0.1000	0.1111	0.1222	0.0000	0.1111	0.1111	0.1111
8	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111	0.0000	0.1111	0.1111
9	0.1238	0.1238	0.1238	0.1238	0.1238	0.0000	0.1238	0.1238	0.0000	0.1349
10	0.1250	0.1250	0.1250	0.1250	0.0000	0.1250	0.1250	0.1250	0.1250	0.0000

5.6.1.7 Elitism

To keep the best solutions found and to survive in the next generation, COIN uses an external list with the same size as the population size to store elitist solutions. All non-dominated solutions created in the previous population are combined with the current elitist solutions. Goldberg's Pareto ranking technique is used to classify the combined population into several non-dominated frontiers. Only the solutions in the first non-dominated frontier are filled in the new elitist list. If the number of solutions in the first non-dominated frontier is less than or equal to the size of the elitist list, the new elitist list will contain all solutions of the first non-dominated frontier. Otherwise, tournament selection for Pareto domination (Horn *et al.*, 1994) is exercised. Two solutions from the first non-dominated solutions are randomly selected and then the solution with larger crowding distance measure and not being selected before is added to the new elitist list. This approach not only ensures that all solutions in the elitist list are non-dominated solutions but also promoting diversity of the solutions. According to our example, the first-ranked elitist list from the first generation is L_1={7,6,2,3,1,4,8,9,5,10} and L_5={3,1,7,6,8,5,4,9,10, 2}. From Table 5.29, the task order list of the population size is L_{11}={7,6,2,3,1,4,8,9,5,10}, L_{12}={8,2,7,5,10,4,3,6,1,9}, L_{13}={8,4,7,2,5,6,3,9,1,10}, L_{14}={3,5,7,4,2,10,1,6,8,9}, L_{15}={4,3,5,7,1,10,6, 9,2,8}, L_{16}={5,4,3,2,10,7,6,9,1,8}, L_{17}={8,3,2,1,5,4,7,10,6,9}, L_{18}={6,5,4,2,9,7,8,10,3,1}, L_{19}={7,5,8,3, 6,10,9,1,2,4}, and L_{20}={2,5,3,1,8,6,9,10,4,7}. The solutions in the current first non-dominated frontier is L_{11}={7,6,2,3,1,4,8,9,5,10}, L_{13}={8,4,7,2,5,6,3,9,1,10}, L_{15}={4,3,5,7,1,10,6,9,2,8} and L_{20}={2,5,3,1,8,6,9,10,4,7}. When the number of the combined solutions is less than the size of the elitist list, both solutions are added to the new elitist. Hence, five good solutions of current elitist list are L_1 or L_{11}={7,6,2,3,1,4,8,9,5,10}, L_5={3,1,7,6,8,5,4,9,10,2}, L_{13}={8,4,7,2,5,6,3,9,1,10}, L_{15}={4,3,5,7,1,10,6,9,2,8} and L_{20}={2,5,3,1,8,6,9,10,4,7}.

Table 5.29 Objective functions of each chromosome from the second generation

Chromosome Number	**Number of workers**	**DOW**	**WT**	**Pareto Frontier**	**Crowding Distance**
L15	4	2.8338	1.8193	1	Infinite
L13	4	2.9090	1.7373	1	2.0000
L11*	4	2.9179	1.7197	1	Infinite
L20*	4	2.9179	1.7197	1	Infinite
L18	4	3.6878	1.9366	2	Infinite
L16**	4	4.0792	1.9593	3	Infinite
L17**	4	4.0792	1.9593	3	Infinite
L12***	4	4.0853	1.9593	4	Infinite
L14***	4	4.0853	1.9593	4	Infinite
L19***	4	4.0853	1.9593	4	Infinite

5.6.1.8 Worker allocation

Finally, the results of 10-task worker allocation of a chromosome L_1 or L_{11} in a single U-shaped assembly line are exemplified in Table 5.30 and Figure 5.14 and 5.15. Previously, its detailed calculation is clearly described in Table 5.25.

Table 5.30 Final exemplified results of Miltenburg's 10-task worker allocation problem for a chromosome L_1 or L_{11}

Chromosome	Worker	Manual time	Travel distance time	Idle time	Allocated tasks (t_k)
1 or 11	1	7	0.14	2.72	7 (3), 6 (4)
	2	9	0.24	0.54	2 (4), 3 (2), 1 (3)
	3	6	0.24	3.54	4 (2), 8 (2), 9 (2)
	4	6	0.14	3.72	5 (4), 10 (2)

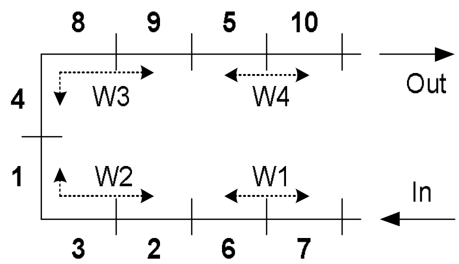


Figure 5.14 Final exemplified 10-task worker allocation results of a chromosome L_1 or L_{11} on a single U-line

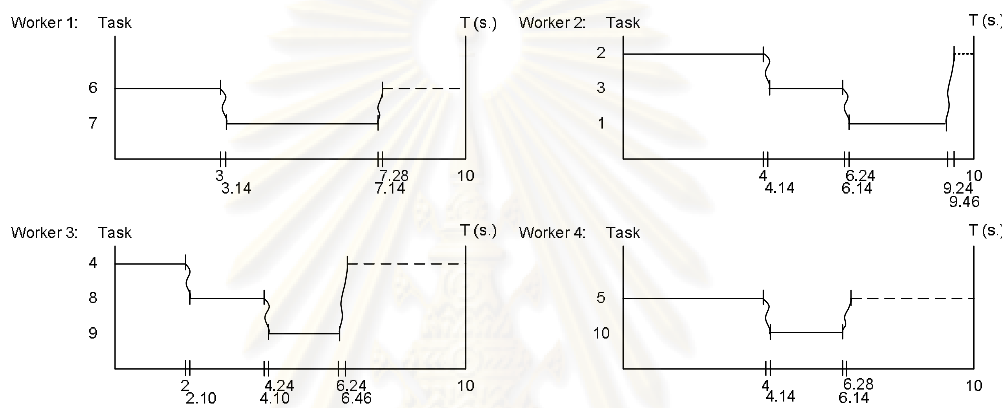


Figure 5.15 Work load routines, showing allocation of four workers: solid line = manual time; wavy line = walking time; dashed line = idle time

5.6.2 Exemplified results

Data of 100 strings at the first generation and the Pareto-optimal frontier of 100 strings at Max. 100 generations for the displacement rule are shown in Figure 5.16. Figure 5.17 illustrates 100 strings at the first generation for only five workers on the final Pareto-optimal frontier. Experimental results of final task sequence; front and back task position; DOW, WT and number of workers are the section of answers below.

Example

Solution: Run COINcidence Algorithm at (10 tasks, 100 strings, 10 cycle time, 100 gen., k = 0.1, 35% APT = 1 second)

Answers:

WT_DOW =

1.0535	17.3200	5.0000
1.2978	16.1800	5.0000
1.3736	15.6600	5.0000
1.5384	14.8400	5.0000
1.5688	14.5200	5.0000
1.6733	14.0000	5.0000
2.0702	12.5000	5.0000
2.1257	11.6600	5.0000
2.2500	10.8400	5.0000
2.4495	10.0000	5.0000
3.6878	9.7100	5.0000
4.2970	9.3200	6.0000
4.5334	6.8400	6.0000

task_seq =

1	4	7	2	10	8	9	3	5	6
3	2	9	4	7	6	8	10	1	5
8	10	1	4	7	5	3	9	2	6
4	10	1	8	9	3	7	6	2	5
1	4	8	2	7	5	10	9	6	3
8	9	10	7	2	3	4	6	5	1
2	3	9	8	5	10	7	1	4	6
9	1	2	4	6	3	10	8	7	5
8	10	5	3	2	4	6	9	7	1
7	8	5	3	1	4	2	10	6	9
5	6	9	8	4	2	7	10	3	1

task_pos =

2	1	1	1	2	1	2	1	1	1
1	1	2	1	1	1	1	2	1	1
1	2	2	1	1	1	1	2	1	1
1	2	2	1	2	1	1	1	1	1

2	1	1	1	1	1	1	2	1	1
1	2	2	1	1	1	1	1	1	1
1	1	2	1	1	1	1	1	1	1
2	2	1	1	1	1	2	1	1	1
1	2	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1

define_station =

1	1	2	2	3	3	4	4	5	5
1	1	2	2	3	3	4	4	5	5
1	1	2	2	3	3	4	4	5	5
1	1	2	2	3	3	4	4	5	5
1	1	2	2	3	3	4	4	5	5
1	1	2	2	3	3	4	4	5	5
1	1	2	2	3	3	4	4	5	5
1	1	2	2	3	3	4	4	5	5
1	1	2	2	3	3	4	4	5	5
1	1	2	2	3	3	4	4	5	5
1	1	2	2	2	3	3	4	4	5

WT_DOW_J =

1.0535	17.3200	5.0000
1.2978	16.1800	5.0000
1.3736	15.6600	5.0000
1.5384	14.8400	5.0000
1.5688	14.5200	5.0000
1.6733	14.0000	5.0000
2.0702	12.5000	5.0000
2.1257	11.6600	5.0000
2.2500	10.8400	5.0000
2.4495	10.0000	5.0000
3.6878	9.7100	5.0000

Elapsed time is 506.407990 seconds.

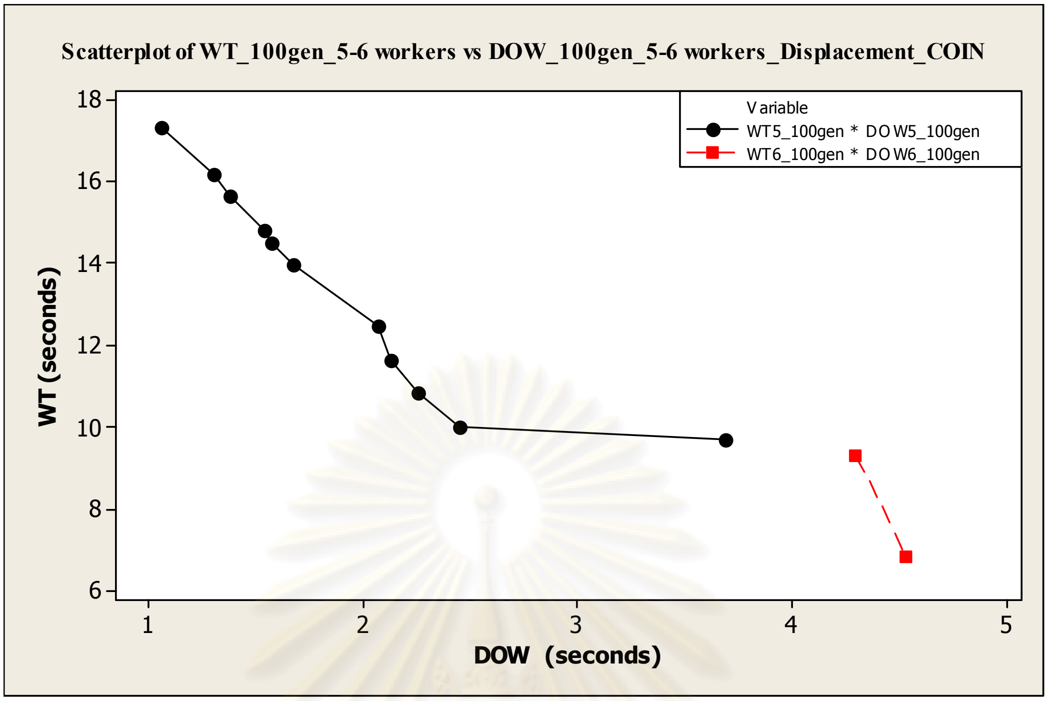


Figure 5.16 DOW vs. WT for 5 and 6 workers at 100 strings and 100 gen. (k = 0.1)

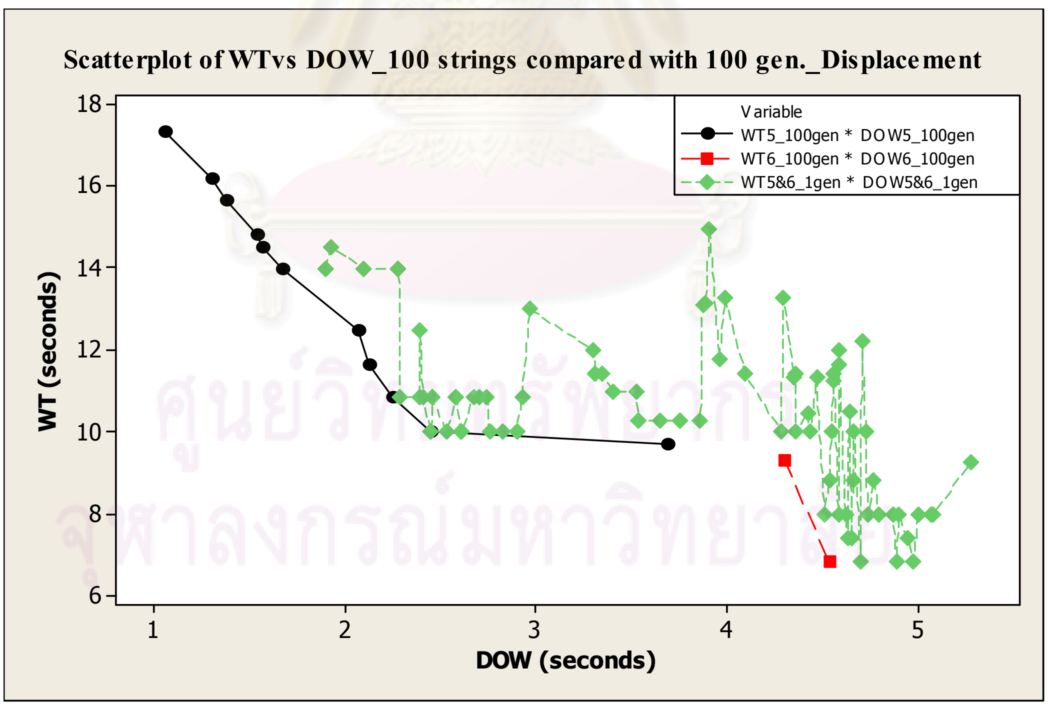


Figure 5.17 DOW vs. WT for 5 and 6 workers at 100 strings and 1 gen. (k = 0.1) 'compared with' DOW vs. WT for 5 and 6 workers at 100 strings and 100 gen. (k = 0.1)

5.7 Particle Swarm Optimization with Negative Knowledge (PSONK)

In this section, particle swarm optimization with negative knowledge for solving multi-objective optimization is proposed. It contributes mainly to avoid producing bad solutions. The mechanism of the PSONK algorithm (Sirovetnukul and Chutima, 2010a) is illustrated in Figure 5.18 and also described by making an illustrative example in Section 5.6.1. It begins by initializing the joint probability matrix and first walk matrix. Then, velocity is normalized to be the probability value into the velocity matrix. Good and bad solutions are rewarded and punished in the updated joint probability matrix with *k*, which is the coefficient of the learning step. Good solutions from the first rank of Pareto frontier are kept in the elitist list of each generation. The generating evaluation and updating steps are repeated until a terminated condition is met.

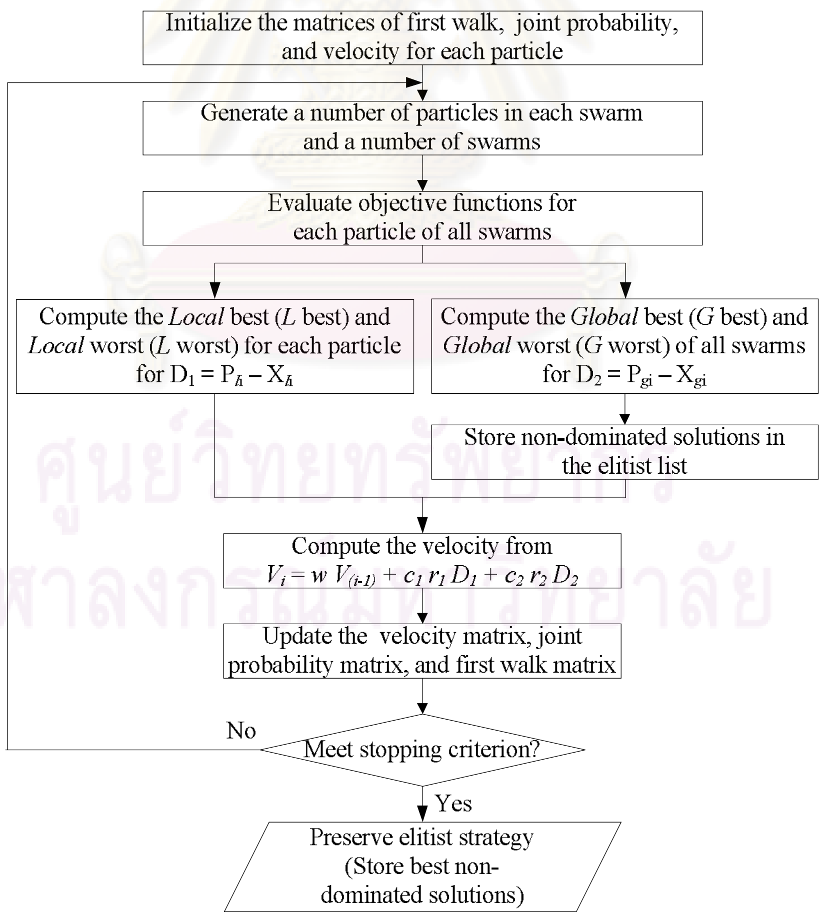


Figure 5.18 Structure of PSONK algorithm

Before starting with PSONK, the representative knowledge of multiobjective PSO that searches only for good solutions is proved by the experimental pilot runs. For instance by 45 tasks at the cycle time of 110 seconds, the results of solutions as shown in Figure 5.19 concludes that the multiobjective PSO is less efficiency than PSONK or PSONK increases the convergence speed than PSO. As a result, the novel DPSO algorithm namely PSONK is operated in the following section.

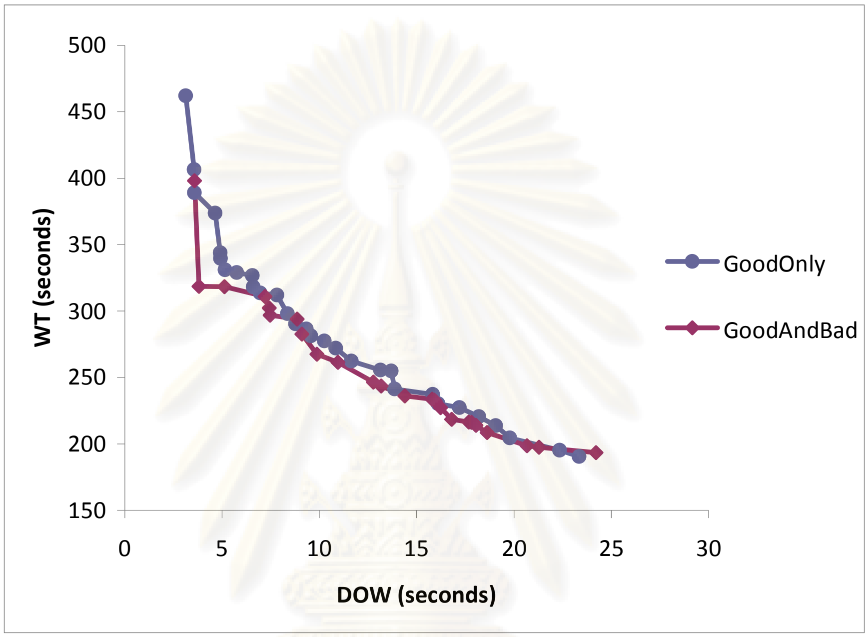


Figure 5.19 PSO with reward only vs. PSONK with both reward and punishment

5.7.1 Numerical example

To present a simpler way, the 11-task renowned problem originated by Jackson (1956) with 13-second cycle time is obviously exemplified to elaborate the algorithm of PSONK at the first generation step by step. The precedence diagram of 11 tasks is shown in Figure 5.20.

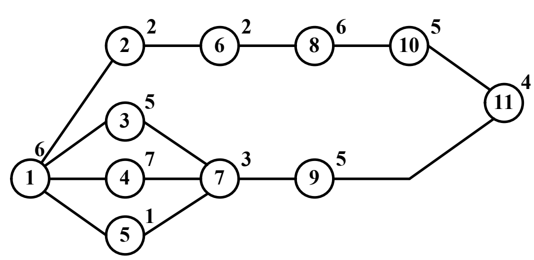


Figure 5.20 Jackson's 11-task precedence diagram

5.7.1.1 First walk and joint probability matrices

First, the dimension of the From-To joint probability matrix is initiated at 11×11. The value of each element from row to column in the matrix is the probability of selecting task at the column after task at the row. Except for the zero diagonal values that cannot move within themselves, the rest of From-To joint probability values are equal to 1/(11-1)=0.1. To randomize the first task in each particle, the dimension of the first walk probability matrix is set off at 1×11. Its values are equally distributed at 1/11=0.0909. The initial velocity matrix is launched at a zero matrix of 11×11.

5.7.1.2 Task sequence

Secondly, two swarms that are that entire collection of particles and four particles of each swarm are exemplified. Each of eight particles are randomized in the first task from the first walk probability matrix and the rest of the tasks (10 tasks) are selected from the From-To joint probability matrix by incorporating the precedence relationship to prohibit producing a task before its direct predecessor. After that, an eight feasible task sequence, named eight particles, is bent in the shape of U, i.e. from the first swarm: P_{11} = {1,5,4,2,6,8,3,7,10,9,11}, P_{12} = {1,5,2,4,3,7,9,6,8,10,11}, P_{13} = {1,5,3,2,4,7,6,8,10,9,11}, and P_{14} = {1,2,6,3,8,5,10,4,7,9,11}; and from the second swarm: P_{21} = {1,2,3,5,4,7,9,6,8,10,11}, P_{22} = {1,4,2,5,6,8,3,10,7,9,11}, P_{23} = {1,5,3,2,6,8,4,7,9,10,11}, and P_{24} = {1,5,2,6,8,10,3,4,7,9,11}.

5.7.1.3 Fitness evaluation

Thirdly, to evaluate the objective or fitness functions, tasks allocated on the U-line have to be searched through both forward and backward directions randomly and assigned to a worker in each loop with the shortest path and the summation of task time and walking time that is less than and equal to given cycle time. From Table 4.14, reasonable walking time from a location to another location is determined by 0.42 (Sirovetnukul and Chutima, 2010b).

5.7.1.4 Non-dominated sorting

Fourthly, non-dominated sorting in Table IV is computed to sort the first rank for the *l*best and the last rank for the *l*worst in each swarm.

Table 5.31 Non-dominated sorting for local particles

Particle	DOW	WT	Rank	Crowding distance	*Local*
14	2.7580	8.0033	1	Infinity	*l*best
13	3.1125	7.5605	1	1.1804	*lbest*
12	3.2284	4.7940	1	1.6822	*l*best
11	4.6274	4.5479	1	Infinity	*l*best
14	2.7580	8.0033	1	Infinity	*l*worst
13	3.1125	7.5605	1	1.1804	*l*worst
12	3.2284	4.7940	1	1.6822	*l*worst
11	4.6274	4.5479	1	Infinity	*l*worst
21	3.3016	4.5479	1	Infinity	*l*best
24	3.6755	4.3019	1	2.0000	*l*best
22	4.2002	3.9540	1	Infinite	*l*best
23	3.6067	4.8959	2	Infinity	*l*worst

In MOPSO, the global vector controls the entire swarm to the global Pareto frontier. For the selection of the *global* best and worst, four particles of each of two swarms are combined to sort non-dominated solutions in Table 5.32. The first and last ranks are representative for the *g*best and *g*worst.

Table 5.32 Non-dominated sorting for global particles

Particle	DOW	WT	Rank	Crowding distance	*Global*
4 (14)	2.7580	8.0033	1	Infinity	gbest
3 (13)	3.1125	7.5605	1	1.1187	gbest
2 (12)	3.2284	4.7940	1	0.8751	gbest
5 (21)	3.3016	4.5479	1	0.4316	gbest
8 (24)	3.6755	4.3019	1	0.7697	gbest
6 (22)	4.2002	3.9540	1	Infinity	gbest
7 (23)	3.6067	4.8959	2	Infinity	gworst
1 (11)	4.6274	4.5479	2	Infinity	gworst

Only the solutions at the first rank of *g*best are stored in the new elitist list to be survived in the next generation.

5.7.1.5 Velocity matrix

In the fifth step, the velocity matrix is computed from Eq. (5.2).

$$V(i,j) = w \times V(i\text{-}1,j) + c_1 \times r_1 \times D_1 + c_2 \times r_2 \times D_2 \qquad (5.2)$$

where i is the generation, j is the swarm, D_1 is the updated local matrix, and D_2 is the updated global matrix. For the first swarm, the velocity matrix of zero is improved with each of the particles of *l*best, *l*worst, *g*best, and *g*worst. Reward and punishment are given to the order pair of tasks with learning step probability (r) assumed to be 0.1. An order pair of tasks in each of all particles is added by the ratio of learning probability (r) divided by (number of tasks - 2). The remaining tasks of the particle are reduced by $r/(t\text{-}2)^2$. Each of the two swarms is updated and exemplified in the first swarm in Table 5.33. Later, another swarm is done on the same procedure.

Table 5.33 The velocity matrix of a sample swarm

From/To	1	2	3	4	5	6	7	8	9	10	11
1	0.000	0.020	-0.005	0.007	0.007	-0.005	-0.005	-0.005	-0.005	-0.005	-0.005
2	-0.005	0.000	0.007	0.020	0.007	-0.005	-0.005	-0.005	-0.005	-0.005	-0.005
3	-0.005	-0.005	0.000	0.007	0.007	-0.005	-0.005	0.007	-0.005	0.007	-0.005
4	-0.005	-0.005	0.007	0.000	-0.005	-0.005	0.032	-0.005	-0.005	-0.005	-0.005
5	-0.005	0.020	-0.005	-0.005	0.000	0.007	-0.005	-0.005	-0.005	0.007	-0.005
6	-0.005	-0.005	0.007	-0.005	-0.005	0.000	-0.005	0.032	-0.005	-0.005	-0.005
7	-0.005	-0.005	-0.005	-0.005	-0.005	0.007	0.000	-0.005	0.044	-0.017	-0.005
8	-0.005	-0.005	-0.005	-0.017	0.007	-0.005	-0.005	0.000	-0.005	0.044	-0.005
9	-0.005	-0.005	-0.005	-0.005	-0.005	0.020	-0.005	-0.005	0.000	-0.017	0.032
10	-0.005	-0.005	0.007	0.007	-0.005	-0.005	0.007	-0.005	-0.005	0.000	0.007
11	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Then, the adjustment of the joint probability matrix $X(i,j)$ is improved with the mixture of the latest velocity matrix $V(i, j)$ normalized by the min-max normalized procedure and the previous joint probability matrix $X(i\text{-}1, j)$. Joint probability matrix is also improved for all swarms.

To update and enter the first task into U-line, the first walk matrix is improved to each of the particles of *l*best, *l*worst, *g*best, and *g*worst. Similar to adjusting reward and punishment before, an example of the first walk matrix of the first swarm is updated and shown in Table 5.34.

Table 5.34 The first walk matrix of a sample swarm

1	2	3	4	5	6	7	8	9	10	11
0.135	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09

5.7.1.6 Worker allocation

Finally, the updated matrices of velocity, joint probability, and first walk are used in the next iteration until the terminating condition is met. The 11-task U-shaped worker allocation of a feasible particle (P_{14}) is illustrated in Figure 5.21.

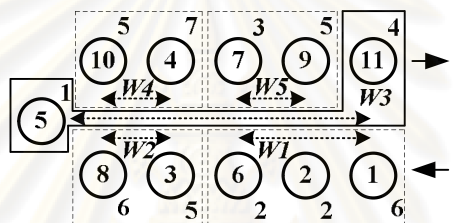


Figure 5.21 An example for the 11-task worker allocation of 13 cycle time

5.7.2 Exemplified results

Data of 100 strings at the first generation and the Pareto-optimal frontier of 100 strings at Max. 100 generations for the displacement rule are shown in Figure 5.22. Experimental results of final task sequence; front and back task position; DOW, WT and number of workers are the section of answers below.

Example

Solution: Run PSONK Algorithm at (11 tasks, 10 swarms&10 particles, C_1, C_2, ω =1, 13 cycle time, 100 gen., r1, r2 = 0.1, 10% APT = 0.42 second, symmetrical layout)

Answers:

WT_DOW =

1.7381 11.1636 5.0000

1.8821 10.7394 5.0000

1.8859	10.4622	5.0000
1.9357	10.1178	5.0000
1.9582	9.9078	5.0000
2.0348	9.5508	5.0000
2.0846	9.2106	5.0000
2.1013	9.1266	5.0000
2.1887	9.0636	5.0000
2.2423	8.8536	5.0000
2.3017	8.5008	5.0000
2.3231	8.4252	5.0000
2.3803	8.0262	5.0000
2.4668	7.6020	5.0000
2.4950	7.1736	5.0000
2.5622	6.6108	5.0000
2.6491	6.4764	5.0000
2.6803	6.1782	5.0000
2.7443	5.4936	5.0000
2.9666	5.0400	5.0000
2.9695	4.7964	5.0000
3.0159	4.7418	5.0000
3.3605	4.7292	5.0000
3.3631	4.4982	5.0000
3.3706	4.4436	5.0000
4.7231	4.2000	6.0000
4.7683	3.9564	6.0000
4.8766	3.9018	6.0000
5.0256	3.6582	6.0000

TS_task_minWS =

1	5	2	6	3	4	7	9	8	10	11
1	5	3	2	6	8	10	4	7	9	11
1	5	2	6	8	10	3	4	7	9	11
1	2	3	6	8	10	4	5	7	9	11
1	3	2	5	6	4	8	10	7	9	11

1	2	6	8	10	4	5	3	7	9	11
1	3	5	2	6	8	10	4	7	9	11
1	5	2	6	8	10	3	4	7	9	11
1	5	2	6	8	10	3	4	7	9	11
1	2	4	6	8	10	5	3	7	9	11
1	5	2	4	6	8	10	3	7	9	11
1	3	5	2	4	7	6	8	10	9	11
1	5	2	4	6	8	10	3	7	9	11
1	5	2	4	6	8	10	3	7	9	11
1	3	5	2	6	8	10	4	7	9	11
1	2	5	3	4	6	8	10	7	9	11
1	5	2	4	6	8	3	7	9	10	11
1	3	2	4	6	5	7	9	8	10	11
1	5	2	4	6	8	10	3	7	9	11
1	3	2	4	5	6	8	10	7	9	11
1	5	2	6	8	10	3	4	7	9	11
1	3	2	4	6	8	10	5	7	9	11
1	3	4	5	7	2	6	8	9	10	11
1	2	6	8	10	4	5	3	7	9	11
1	3	4	2	6	8	10	5	7	9	11

Station =

1	2	2	3	4	5	5	4	3	2	1
2	3	4	5	5	5	4	3	2	1	1
2	3	4	4	5	5	4	3	2	1	1
2	3	3	4	5	5	4	3	2	1	1
1	1	2	2	3	4	5	5	4	3	2
2	3	4	5	5	4	3	3	2	1	1
1	1	2	2	3	4	4	5	5	3	2
2	3	5	5	5	4	4	3	2	1	1
1	1	1	2	3	3	4	5	5	4	2
2	3	5	5	4	4	3	3	2	1	1
1	2	2	3	3	4	5	5	4	2	1
3	4	4	5	5	4	3	2	2	1	1

2	4	5	5	4	4	3	3	2	1	1
2	4	4	5	5	4	3	3	2	1	1
4	5	5	5	4	3	3	2	2	1	1
1	1	1	3	5	5	4	4	3	2	2
1	1	1	2	2	3	3	4	5	5	4
2	2	3	5	5	4	4	4	3	1	1
1	1	1	3	3	4	5	5	4	2	2
4	4	5	5	3	3	3	2	2	1	1
1	1	1	4	4	5	5	3	3	2	2
5	5	4	4	3	3	2	2	2	1	1
1	1	5	4	4	4	4	3	3	2	2
1	1	1	4	4	5	3	3	3	2	2
3	3	5	4	4	4	2	2	2	1	1

WT_DOW_J =

1.7381	11.1636	5.0000
1.8821	10.7394	5.0000
1.8859	10.4622	5.0000
1.9357	10.1178	5.0000
1.9582	9.9078	5.0000
2.0348	9.5508	5.0000
2.0846	9.2106	5.0000
2.1013	9.1266	5.0000
2.1887	9.0636	5.0000
2.2423	8.8536	5.0000
2.3017	8.5008	5.0000
2.3231	8.4252	5.0000
2.3803	8.0262	5.0000
2.4668	7.6020	5.0000
2.4950	7.1736	5.0000
2.5622	6.6108	5.0000
2.6491	6.4764	5.0000
2.6803	6.1782	5.0000
2.7443	5.4936	5.0000

2.9666	5.0400	5.0000
2.9695	4.7964	5.0000
3.0159	4.7418	5.0000
3.3605	4.7292	5.0000
3.3631	4.4982	5.0000
3.3706	4.4436	5.0000

Elapsed time is 115.373901 seconds.

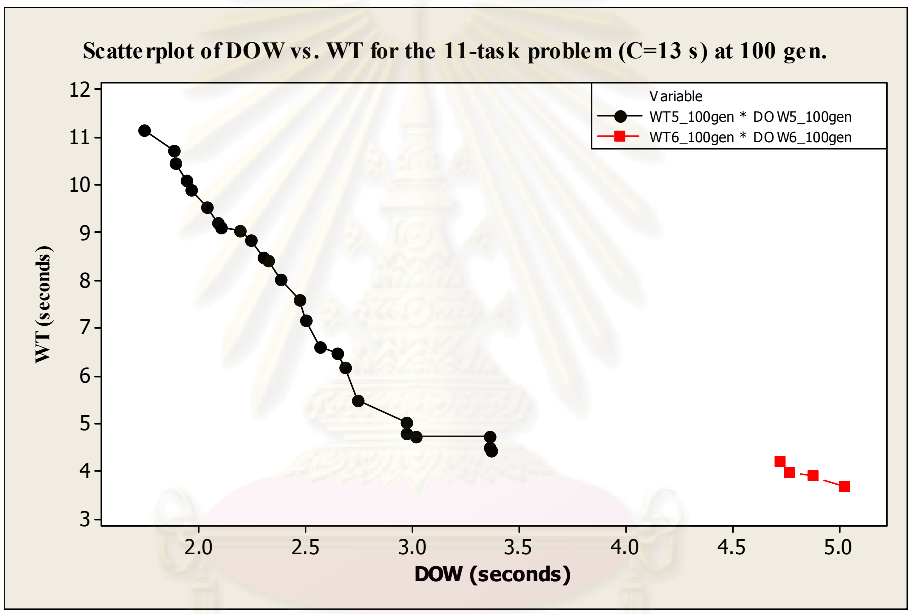


Figure 5.22 DOW vs. WT for 5 and 6 workers at 100 gen.

5.8 Comparisons of Performance Measures

If all objective functions are minimized, a feasible solution x is said to dominate another feasible solution y $(x \succ y)$ as shown in Figure 5.23. A solution is said to be Pareto optimal if it is not improved by any other solution. The set of all feasible non-dominated solutions is referred to as the Pareto optimal set.

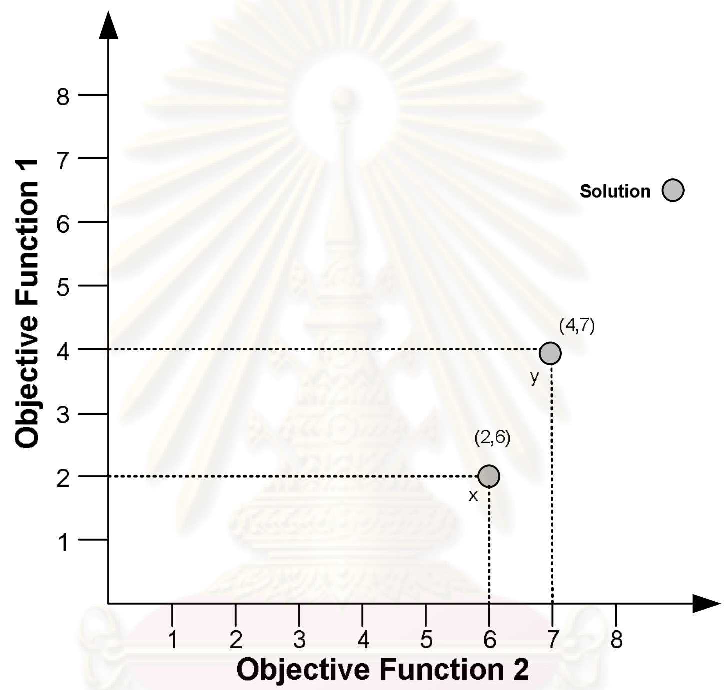


Figure 5.23 Non-dominated solution

The Pareto optimal set shown in Figure 5.24 is a solution frontier from only one solution set which is compared with the other Pareto optimal sets from the other solution sets. It is necessary to identify the reference solution frontier, which refers to the optimal solution, named the true Pareto optimal solution set. However, the true Pareto optimal solution set is not able to compute from existing solution sets, but it is estimated by the approximation of obtained sets. Thus, the approximate true Pareto optimal solution set is employed to compare with the others. To identify the approximate true Pareto set, obtained Pareto optimal sets have to be plotted in the same graph to find the local solution which is unable to be dominated as shown in Figure 5.25.

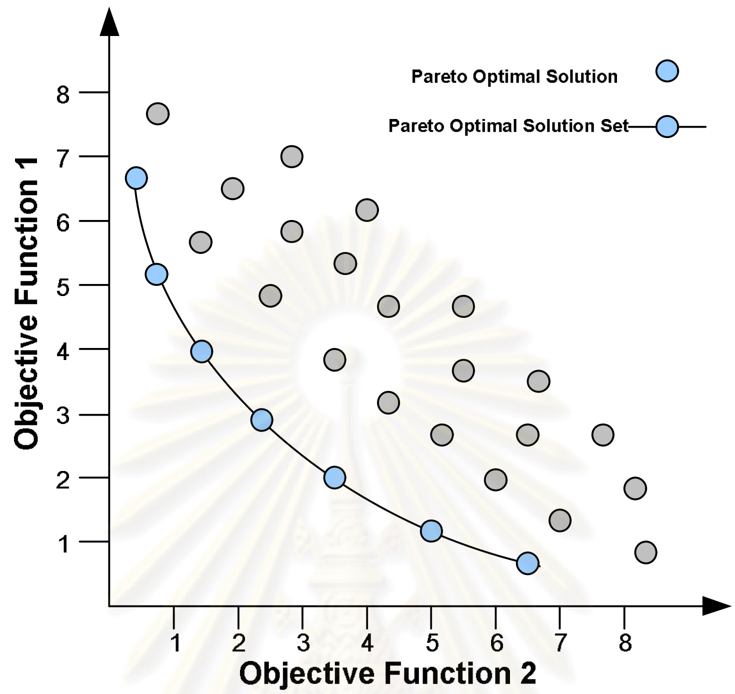


Figure 5.24 Pareto optimal solution set

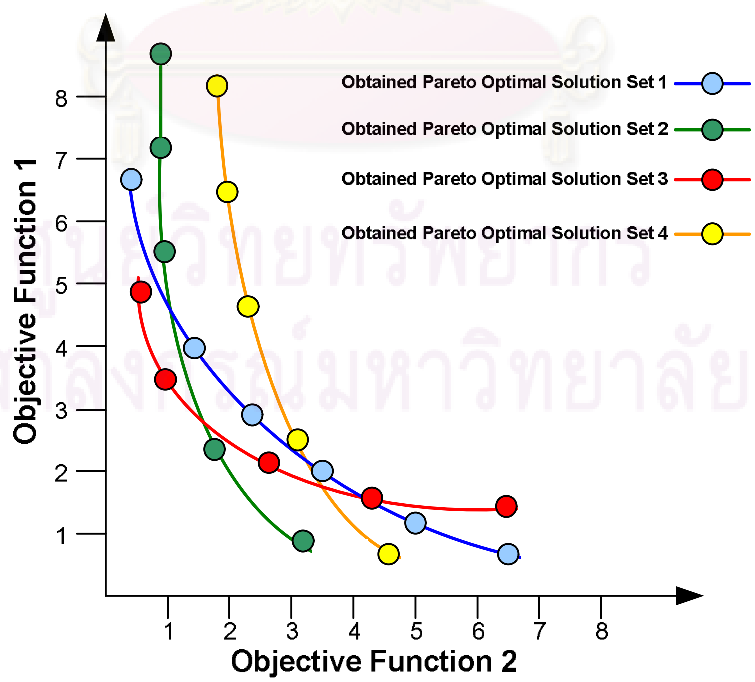


Figure 5.25 Obtained Pareto optimal solution set

In Figure 5.26, the approximate true Pareto optimal solution set may not be computed from all solutions. Only dominated solutions are selected as obtained values for calculating the approximate true Pareto optimal solution set. The approximate true Pareto optimal solution set is employed to compare the performance measures of each solution frontier, that is, convergence to the Pareto-optimal set and Ratio of non-dominated solution in the following section.

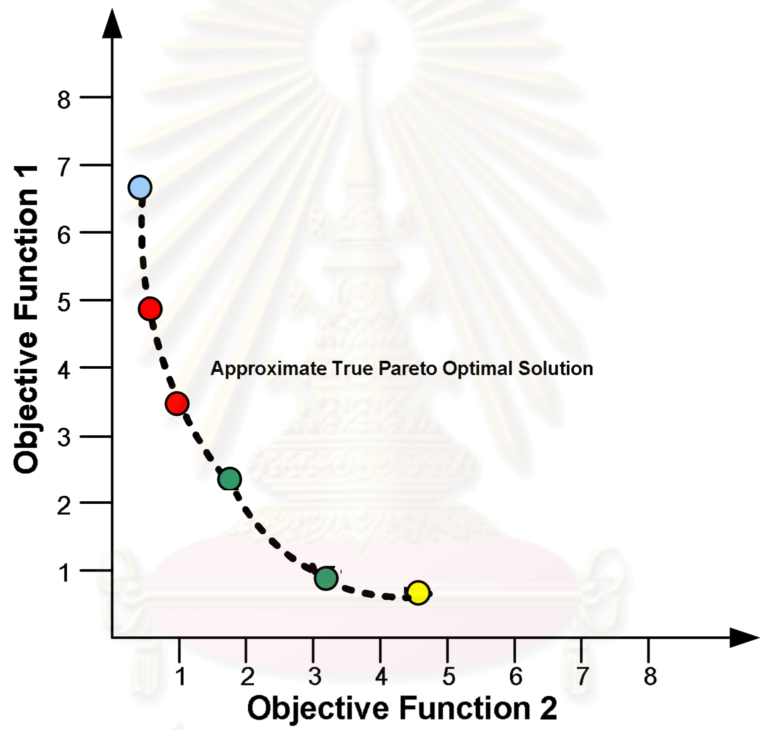


Figure 5.26 Approximate true Pareto optimal solution set

5.8.1 Convergence to the Pareto-optimal set

The solutions of an obtained Pareto optimal set (NSGA-II) and an approximate true Pareto optimal set are exemplified and shown in Table 5.35 and Figure 5.27.

Table 5.35 Obtained Pareto optimal set of NSGA-II and approximate true Pareto optimal set

Solutions	**No.**	f_1 **(x) =** ***DOW***	f_2 **(x) =** ***WT***
Obtained Pareto optimal set	1	2.6179	1.9108
	2	2.7419	1.4849
	3	2.7655	1.3109
	4	2.8199	1.1370
Approximate true Pareto optimal set	1*	2.3756	2.9008
	2*	2.3921	2.6922
	3*	2.4730	2.3940
	4*	2.4928	2.3114
	5*	2.5253	2.1770
	6*	2.5309	2.1616
	7*	2.5654	2.1434
	8*	2.5681	2.0860
	9*	2.6012	2.0314
	10*	2.6105	1.9600
	11*	2.6179	1.9108
	12*	2.6208	1.7542
	13*	2.6922	1.6730
	14*	2.6983	1.4994
	15*	2.7419	1.4849
	16*	2.7655	1.3109
	17*	2.8199	1.1370

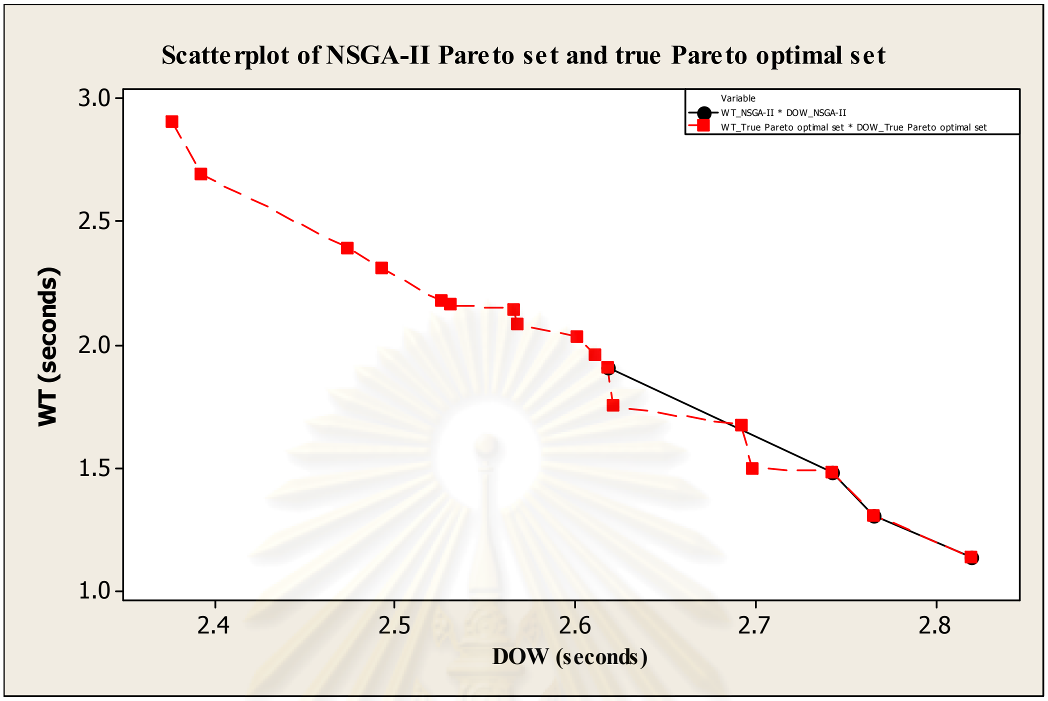


Figure 5.27 Obtained Pareto optimal solutions of NSGA-II and approximate true Pareto optimal solutions

From the chapter II, $d_i = \min_{j=1}^{|A^*|} \sqrt{\sum_{k=1}^{K} \left(\frac{f_k(x) - f_k(y)}{f_k^{\max} - f_k^{\min}} \right)^2}$ in Eq. (2.6) and $convergence(A) = \frac{\sum_{i=1}^{|A^*|} d_i}{|A^*|}$ in Eq. (2.5) are computed step by step, where A = obtained solutions set, A^* = approximate true solutions set, $/A^*/$ = number of approximate true solutions, k = number of objective functions, x = obtained Pareto optimal solutions, y = approximate true Pareto optimal solutions, d_i = normalized Euclidean distance between x and y.

Step 1. From Table 5.35, the normalized Euclidean distance of f_1 *(x)*, named *DOW*, is computed with $f_1^{\min}$ = 2.5756, $f_1^{\max}$ = 2.8199, and the matrix of f_1

(x) and f_1 *(y)*. For example, $\left(\frac{f_1(2.6179)-f_1(2.3756)}{f_1^{\max}-f_1^{\min}}\right)^2 = \left(\frac{2.6179-2.3756}{2.8199-2.3756}\right)^2 =$ 0.2974. The rest of results are computed as the same way and shown in Table 5.36.

Table 5.36 Normalized Euclidean distance of f_1 *(x)*, DOW

NSGA-II Pareto solutions		2.6179	2.7419	2.7655	2.8199
Approximate true Pareto optimal solutions	2.3756	0.2974	0.6797	0.7701	1.0000
	2.3921	0.2583	0.6199	0.7063	0.9271
	2.4730	0.1064	0.3663	0.4334	0.6096
	2.4928	0.0793	0.3143	0.3767	0.5420
	2.5253	0.0434	0.2377	0.2923	0.4397
	2.5309	0.0383	0.2255	0.2788	0.4231
	2.5654	0.0140	0.1578	0.2028	0.3281
	2.5681	0.0126	0.1530	0.1974	0.3212
	2.6012	0.0014	0.1003	0.1367	0.2423
	2.6105	0.0003	0.0875	0.1217	0.2221
	2.6179	0.0000	0.0779	0.1104	0.2067
	2.6208	0.0000	0.0743	0.1061	0.2008
	2.6922	0.0280	0.0125	0.0272	0.0826
	2.6983	0.0327	0.0096	0.0229	0.0749
	2.7419	0.0779	0.0000	0.0028	0.0308
	2.7655	0.1104	0.0028	0.0000	0.0150
	2.8199	0.2067	0.0308	0.0150	0.0000

Step 2. From Table 5.35, the normalized Euclidean distance of f_2 *(x)*, named *WT*, is computed with $f_2^{\min}$ = 1.137, $f_2^{\max}$ = 2.9008, and the matrix of f_2 *(x)* and f_2 *(y)*. For example, $\left(\frac{f_2(1.4849)-f_2(2.6922)}{f_2^{\max}-f_2^{\min}}\right)^2 = \left(\frac{1.4819-2.6922}{2.9008-1.137}\right)^2$ = 0.4685. The rest of results are shown in Table 5.37.

Table 5.37 Normalized Euclidean distance of $f_2(x)$, WT

NSGA-II Pareto solutions		1.9108	1.4849	1.3109	1.1370
Approximate true Pareto optimal solutions	2.9008	0.3150	0.6444	0.8125	1.0000
	2.6922	0.1963	0.4685	0.6133	0.7775
	2.3940	0.0751	0.2657	0.3771	0.5079
	2.3114	0.0516	0.2196	0.3218	0.4433
	2.1770	0.0228	0.1540	0.2411	0.3477
	2.1616	0.0202	0.1472	0.2326	0.3375
	2.1434	0.0174	0.1394	0.2228	0.3256
	2.0860	0.0099	0.1161	0.1931	0.2895
	2.0314	0.0047	0.0960	0.1669	0.2571
	1.9600	0.0008	0.0726	0.1354	0.2177
	1.9108	0.0000	0.0583	0.1157	0.1925
	1.7542	0.0079	0.0233	0.0632	0.1224
	1.6730	0.0182	0.0114	0.0421	0.0923
	1.4994	0.0544	0.0001	0.0114	0.0422
	1.4849	0.0583	0.0000	0.0097	0.0389
	1.3109	0.1157	0.0097	0.0000	0.0097
	1.1370	0.1925	0.0389	0.0097	0.0000

Step 3. The square root of summation of *DOW* and *WT* in Table 5.36 and 5.37 is computed, e.g. $\sqrt{(0.2974+0.3150)}$=0.7826. The rest of results are calculated and shown in Table 5.38.

Step 4. The minimum distance of each normalized Euclidean distance (d_i) of *DOW* and *WT* is computed, i.e., $d_i = \min_{j=1}^{|A^*|}\sqrt{\sum_{k=1}^{K}\left(\frac{f_k(x)-f_k(y)}{f_k^{\max}-f_k^{\min}}\right)^2}$ = $\min_{j=1}^{17}\sqrt{\sum_{k=1}^{K}\left(\frac{f_k(x)-f_k(y)}{f_k^{\max}-f_k^{\min}}\right)^2}$ = min(0.7826,1.1507,1.2580,1.4142)$_1$, min(0.6712, 1.0433,1.1487,1.3056)$_2$,…, min(0.6318,0.2641,0.1572,0)$_{17}$. After that, the average minimum distance or convergence is the total minimum distance divided by the number of true Pareto optimal solutions, i.e., $convergence(NSGA-II) = \frac{\sum_{i=1}^{|A^*|} d_i}{|A^*|} = \frac{\sum_{i=1}^{17} d_i}{17} = \frac{(0.7286+0.6742+...+0)}{17} = 0.2072$. The results are shown in Table 5.39.

Table 5.38 Normalized Euclidean distance (d_i) of *DOW* and *WT*

NSGA-II Pareto solutions		**1**	**2**	**3**	**4**
Approximate true Pareto optimal solutions	**1**	0.7826	1.1507	1.2580	1.4142
	2	0.6742	1.0433	1.1487	1.3056
	3	0.4259	0.7950	0.9003	1.0571
	4	0.3618	0.7307	0.8358	0.9926
	5	0.2573	0.6258	0.7303	0.8873
	6	0.2420	0.6105	0.7151	0.8721
	7	0.1771	0.5452	0.6524	0.8085
	8	0.1498	0.5188	0.6249	0.7815
	9	0.0780	0.4430	0.5510	0.7067
	10	0.0325	0.4000	0.5071	0.6632
	11	0.0000	0.3691	0.4754	0.6318
	12	0.0890	0.3124	0.4114	0.5686
	13	0.2148	0.1546	0.2634	0.4183
	14	0.2952	0.0985	0.1852	0.3422
	15	0.3691	0.0000	0.1120	0.2641
	16	0.4754	0.1120	0.0000	0.1572
	17	0.6318	0.2641	0.1572	0.0000

Table 5.39 Average minimum distance of *DOW* and *WT*

NSGA-II Pareto solutions		1	2	3	4	Min (d_i)
Approximate true Pareto optimal solutions	1	0.7826	1.1507	1.4142	1.2580	0.7826
	2	0.6742	1.0433	1.3056	1.1487	0.6742
	3	0.4259	0.7950	1.0571	0.9003	0.4259
	4	0.3618	0.7307	0.9926	0.8358	0.3618
	5	0.2573	0.6258	0.8873	0.7303	0.2573
	6	0.2420	0.6105	0.8721	0.7151	0.2420
	7	0.1771	0.5452	0.8085	0.6524	0.1771
	8	0.1498	0.5188	0.7815	0.6249	0.1498
	9	0.0780	0.4430	0.7067	0.5510	0.0780
	10	0.0325	0.4000	0.6632	0.5071	0.0325
	11	0.0000	0.3691	0.6318	0.4754	0.0000
	12	0.0890	0.3124	0.5686	0.4114	0.0890
	13	0.2148	0.1546	0.4183	0.2634	0.1546
	14	0.2952	0.0985	0.3422	0.1852	0.0985
	15	0.3691	0.0000	0.2641	0.1120	0.0000
	16	0.4754	0.1120	0.1572	0.0000	0.0000
	17	0.6318	0.2641	0.0000	0.1572	0.0000
					Total minimum distance	3.5232
					Average	<u>0.2072</u>

5.8.2 Spread of non-dominated solutions

From the chapter II, $spread(A) = \dfrac{d_f + d_l + \sum_{i=1}^{|A|-1} \left| d_i - \bar{d} \right|}{d_f + d_l + (|A|-1)\bar{d}}$ in Eq. (2.7) is computed step by step, where d_f and d_l are Euclidean distances between extreme solutions and boundary of the obtained non-dominated set, $|A|$ is the number of obtained solutions, $\bar{d}$ is the average of all distances d_i, $i = 1, 2, ..., |A|-1$, assuming that there are solutions on the best non-dominated front. With A solutions, there are *(A-1)* consecutive distances.

Step 1. The obtained Pareto set of NSGA-II, i.e., {(2.6179, 1.9108), (2.7419, 1.4849), (2.7655, 1.3109), (2.8199, 1.1370)} is exemplified. The consecutive distances (d_i) are computed by the normalized equation of $\sqrt{\sum_{k=1}^{2}\left(\frac{f_k(x)-f_k(x+1)}{f_k^{\max}-f_k^{\min}}\right)^2}$ and shown in Table 5.40. For example, at the objectives of DOW and WT, d_1 or d_f is

$$\sqrt{\left(\frac{f_1(1)-f_1(2)}{f_1^{\max}-f_1^{\min}}\right)^2+\left(\frac{f_2(1)-f_2(2)}{f_2^{\max}-f_2^{\min}}\right)^2} = \sqrt{\left(\frac{2.6179-2.7419}{2.8199-2.6179}\right)^2+\left(\frac{1.9108-1.4849}{1.9108-1.137}\right)^2} =$$

0.8245. After that, the average distance is calculated by $\frac{\Sigma_{i=1}^{4-1} d_i}{3}$ = 0.4762.

Table 5.40 Consecutive distances (d_i)

No.	NSGA-II		Normalized		Euclidean distance	
	$f_1(x)$	$f_2(x)$	$f_1(x)$	$f_2(x)$		
1.	2.6179	1.9108	0.3768	0.3029	d_1 or d_f	0.8245
2.	2.7419	1.4849	0.0136	0.0506	d_2	0.2534
3.	2.7655	1.3109	0.0725	0.0505	d_3 or d_l	0.3508
4.	2.8199	1.137	Average distance ($\bar{d}$)			0.4762

Step 2. The spread measure is computed by *spread (NSGA-II)* =

$$\frac{0.8425+0.3508+[(0.8245-0.4762)+(0.2534-0.4762)+(0.3508-0.4762)]}{0.8245+0.3508+[(4-1)*0.4762]}=\frac{1.8718}{2.6039}$$

= 0.7188.

5.8.3 Ratio of non-dominated solutions

This measure simply counts the number of solutions which are members of the Pareto optimal set. The measure of ratio of non-dominated solution can be written in Eq. (2.8) from the chapter II and computed step by step as shown in

Table 5.41. $R_{NDS}(A_j) = \dfrac{|A_j - \{x \in A_j \mid \exists y \in A : y \prec x\}|}{|A_j|}$ means that the ratio of solutions in A_j are not dominated by any other solutions in A ,where A_j is a solution set $(j = 1, 2, ..., J)$, $A = A_1 \cup A_2 ... \cup A_j$, $y \prec x$ means the obtained solution x is dominated by the true Pareto solution *y*.

Table 5.41 Ratio of non-dominated solutions (NSGA-II)

True Pareto optimal solutions	No.	Obtained Pareto solutions of NSGA-II				
		1	2	3	4	Value
		(2.6179,1.9108)	(2.7419,1.4849)	(2.7655,1.3109)	(2.8199,1.137)	
(2.3756,2.9008)	1	0	0	0	0	0
(2.3921,2.6922)	2	0	0	0	0	0
(2.4730,2.3940)	3	0	0	0	0	0
(2.4928,2.3114)	4	0	0	0	0	0
(2.5253,2.1770)	5	0	0	0	0	0
(2.5309,2.1616)	6	0	0	0	0	0
(2.5654,2.1434)	7	0	0	0	0	0
(2.5681,2.0860)	8	0	0	0	0	0
(2.6012,2.0314)	9	0	0	0	0	0
(2.6105,1.9600)	10	0	0	0	0	0
(2.6179,1.9108)	11	1	0	0	0	1
(2.6208,1.7542)	12	0	0	0	0	0
(2.6922,1.6730)	13	0	0	0	0	0
(2.6983,1.4994)	14	0	0	0	0	0
(2.7419,1.4849)	15	0	1	0	0	1
(2.7655,1.3109)	16	0	0	1	0	1
(2.8199,1.1370)	17	0	0	0	1	1
Total value						4
Ratio of non-dominated solutions (NSGA-II)						<u>1</u>

Step 1. The sizes of matrix are determined by the number of obtained Pareto solutions (x) and the number of true Pareto optimal solutions (y).

Step 2. The Pareto dominance is compared between x and y. If an obtained Pareto solution (x) is equal to a true Pareto optimal solution (y), then the value of x and y is the value of 1. If not, the value is 0.

Step 3. The values in each row of true Pareto optimal solutions are summed into the last column in Table 5.41. Finally, the ratio of non-dominated solutions is calculated by the total value divided by the number of obtained Pareto solutions (Total value / $|A_j|$). For example, the ratio of NSGA-II is equal to 4/4 = 1.

5.8.4 Central processing unit (CPU time)

The Central Processing Unit (CPU) or processor is the portion of a computer system that carries out the instructions of a computer program, and is the primary element carrying out the computer's functions. This term has been in use in the computer industry at least since the early 1960s. It is used for the purpose of comparing algorithms and compared at the same iterations for each other. It is shown in the elapsed time after the terminating condition is met, e.g. Elapsed time is 115.373901 seconds.

5.8.5 Exemplified results

Solutions of DOW and WT of NSGA-II and PSONK obtained from the previous algorithms are exemplified and input to find out true-Pareto optimal solutions. After that, each of solutions of NSGA-II and PSONK is compared to true-Pareto optimal solutions.

Example:

<u>Inputs</u>

DOW_NSGA-II	WT_NSGA-II	DOW_PSONK	WT_PSONK
1. 2.6179	1. 1.9108	1. 2.3756	1. 2.9008

DOW_NSGA-II	WT_NSGA-II	DOW_PSONK	WT_PSONK
2. 2.7419	2. 1.4849	2. 2.3921	2. 2.6922
3. 2.7655	3. 1.3109	3. 2.4730	3. 2.3940
4. 2.8199	4. 1.1370	4. 2.4928	4. 2.3114
		5. 2.5253	5. 2.1770
		6. 2.5309	6. 2.1616
		7. 2.5654	7. 2.1434
		8. 2.5681	8. 2.0860
		9. 2.6012	9. 2.0314
		10. 2.6105	10. 1.9600
		11. 2.6208	11. 1.7542
		12. 2.6922	12. 1.6730
		13. 2.6983	13. 1.4994
		14. 2.9178	14. 1.4812
		15. 3.5418	15. 1.4588

<u>Outputs</u>

True-Pareto optimal solutions =

2.3756	2.9008
2.3921	2.6922
2.4730	2.3940
2.4928	2.3114
2.5253	2.1770
2.5309	2.1616
2.5654	2.1434
2.5681	2.0860
2.6012	2.0314
2.6105	1.9600
2.6179	1.9108
2.6208	1.7542
2.6922	1.6730
2.6983	1.4994
2.7419	1.4849
2.7655	1.3109
2.8199	1.1370

ans = Convergence to the Pareto optimal set
Spread of non-dominated solutions
Ratio of non-dominated solutions

NSGA-II		PSONK	
ans =	0.2072	ans =	0.0236
	0.7188		0.7646
	1.0000		0.8667

The convergence of the obtained Pareto-optimal solutions towards a true Pareto-set is the difference between the obtained solution set and the true-Pareto set. The lower the value is, the better the convergence metric is. The second measure is a spread metric. This measure computes the distribution of the obtained Pareto-solutions by calculating a relative distance between consecutive solutions. The value of this measure is zero for a uniform distribution, but it can be more than zero when bad distribution is found. The third measure is the ratio of non-dominated solutions which indicates the coverage of one set over another. The higher ratio indicates superiority of one solution set over another.

CHAPTER VI

EXPERIMENTS AND COMPUTATIONAL RESULTS

6.1 Introduction

There are two fundamental performances that are quality solutions and running time to decide what a good heuristic is selected. This chapter is organized in the following. First, it provides the findings of experimental parameter settings such as number of generations, population size, reward and punishment probability, and so on. Then, experimental results of NSGA-II, MA, COIN, and PSONK are proposed and input to evaluate the performances of algorithms in terms of the former performances of convergence to the Pareto-optimal set, spread of non-dominated solutions and ratio of non-dominated solutions; and the latter performance of processing time. Finally, the discussion of all algorithms and given cycle times is taken into account.

6.2 Findings of Experimental Parameter Settings

In this section, there are many factors related to parameter settings in the SUALWAPs. Their reference values are described in the following section.

6.2.1 Number of generations

Olanviwatchai (2009) was used as a starting point for parameter settings. The selection of number of generations was based on quality solutions by extensive pilot runs. Having done that, the number of generations of 19-task, 36-task, 61-task, and 111-task problems are 100, 100, 150, and 300, respectively. For example, after increasing the number of generations from one, four, fifty, and one hundred better solutions are improved as the Pareto-optimal frontier from the 36-task problem with the COIN algorithm shown in Figure 6.1. In this study, the number of generation of 100, 150, 300, and 50 are set to tiny task at 7-45 tasks, small task at 61-40 tasks, medium task at 111 tasks, and large task at 297 tasks, respectively.

6.2.2 Population size

Small population size results in the local optimum due to a genetic drift phenomenon (a few clusters of populations being formed in the solution space), but overpopulation may lose computational time. Thus, Hwang and Katayama (2009) claim that the population size should be set to 100 in the U-line balancing problem.

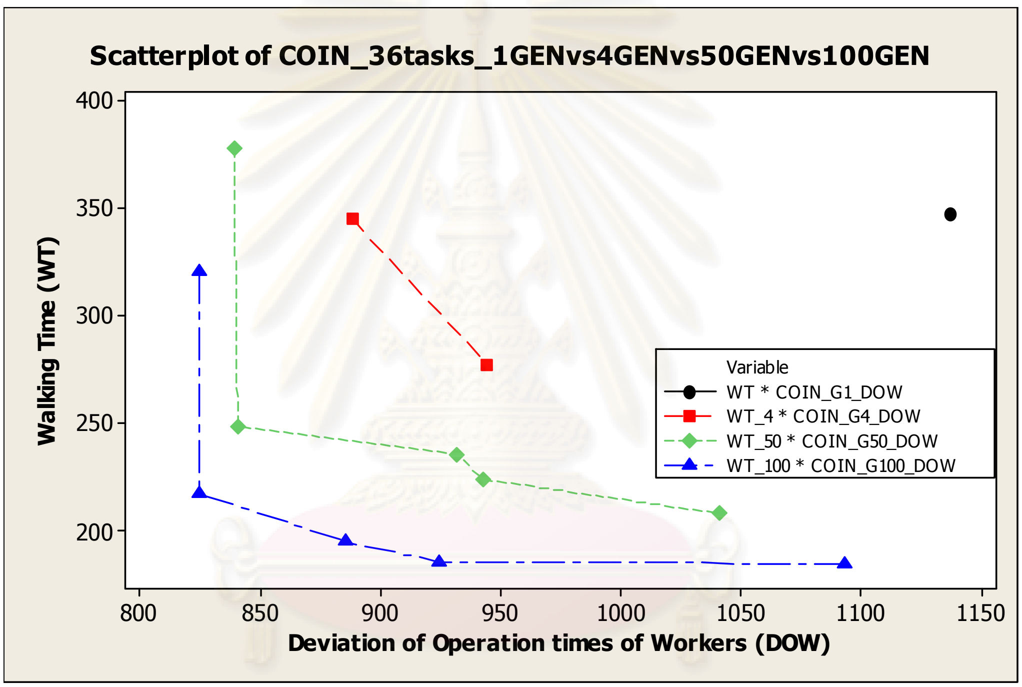


Figure 6.1 Comparison of generation 1, 4, 50, and 100 with COIN for the 36-task problem (C = 1,371 seconds)

6.2.3 Selection method

Binary tournament selection method is used to choose strings or population size. Binary tournament is run to determine a relative fitness ranking. Initially the entire population is in the tournament. Two members are selected at random to compete against each other with only the winner of the competition progressing to the next level of the tournament. Binary tournament selection implies that two individuals directly compete for selection.

6.2.4 Pareto-based approach

There are a few Pareto based ranking methods, that is, Belegundu, Goldberg and Fonseca and Fleming. Goldberg's ranking or non-dominated sorting (Deb *et al., 2002*) is applied in this study. It assigns equal probability of reproduction to all non-dominated individuals in the population. The method consisted of assigning rank 1 to the non dominated individuals and removing them from contention, then finding a new set of non dominated individuals, ranked 2, and so forth.

6.2.5 Density information

The use of the crowed comparison operator, which basically is a computation of the crowding distance of each solution, as a diversity operator by NSGA-II wa able to produce a better distribution of the generated nondominated solutions. Thus, it is likely applied into other multi-objectieve algorithms. It is obviously supported that Raquel and Naval (2005) claim that crowding distance is effective in multiobjective particle swarm optimization. They present an approach that extends the Particle Swarm Optimization (PSO) algorithm to handle multiobjective optimization problems by incorporating the mechanism of crowding distance computation into the algorithm of PSO, specifically on global best selection and in the deletion method of an external archive of nondominated solutions. The crowding distance mechanism together with a mutation operator maintains the diversity of nondominated solutions in the external archive. The performance of this approach is evaluated on test functions and metrics from literature. The results show that the proposed approach is highly competitive in converging towards the Pareto front and generates a well distributed set of nondominated solutions.

6.2.6 Crossover method

In line balancing problems and others, several crossover operators have been proposed to create an offspring such as partially-mapped crossover (PMX), order crossover (OX), and modified order crossover (modOX). However, the two point-based weight mapping crossover (WMX) by Hwang *et al.* (2008) is used in this study. Crossover probability (P_C) is set to 0.7. In addition, Zhang and Gen (2009) also use weight mapped crossover (WMX) for worker allocation in an assembly line balancing problem.

6.2.7 Mutation method

In genetic algorithms of computing, mutation is a genetic operator used to maintain genetic diversity from one generation of a population of algorithm chromosomes to the next. It is analogous to biological mutation. The purpose of mutation in GAs is preserving and introducing diversity. Mutation should allow the algorithm to avoid local minima by preventing the population of chromosomes from becoming too similar to each other. Reciprocal exchange mutation probability (P_M) is set to 0.3 by Hwang *et al.* (2008) and referred to in this study.

6.2.8 Local search

Four local searches modified from Kumar and Singh (2007) originally developed to solved traveling salesman problems by repeatedly exchanging edges of tour until no improvement is attained are examined including Pair wise Interchange (PI), Insertion Procedures (IP), 2-Opt, and 3-Opt. The number of places to apply local search has a direct effect on the quality of solution and computation time. Hence, if computation time needs to be saved, local search should be taken only at some specific steps in the algorithm of MA rather than at all possible steps. In this research, local searches are chosen to take after obtaining initial solution and after mutation due to previous research (Chutima and Pinkoompee, 2008). Consequently, PI is set to local search after initial stage and IP is set to local search after mutation in this study. Local search probability (P_L) is set to 0.8 as the same Ishibuchi *et al.* (2003).

6.2.9 Heuristic

The proposed approach employs a randomly task assignment heuristic rule in this study.

6.2.10 Reward and punishment probability

To study the effect of reward and punishment of good and bad instances in multi-objective problems, the multi-objective COIN is tested in a multi-objective TSP problem. Wattanapornprom *et al.* (2009) set up an experiment using kroa100 and krob100 as a dual-objective 100 tours TSP problem obtained from the TSPLIB. The population size used in the experiment is 500 and the learning step k or the reward and punishment probability is equal to 0.1. Furthermore, it is also set to 0.1 after testing pilot runs in the U-line balancing problems by Olanviwatchai (2009).

6.2.11 Cognitive, social and inertia weights

In the experiments of Salman *et al.* (2002)'s task assignment problem, the following values for the weights of cognitive component (C_1) and social component (C_2) is set to 1 in Eq. (5.2). The inertia weight (ω) is also set to 1 approximately in the same equation.

6.3 Experimental Results of NSGA-II, MA, COIN, and PSONK

Metaheuristics can be used to fine-tune parameters. When there are several parameters, it is quite tedious to fine-tune these parameters using an experimental design. After doing the pilot run in the last chapter, the algorithm applies the following parameters throughout the simulations.

6.3.1 Initialization of all algorithms

<u>Initialization of NSGA-II</u>

The algorithm applies the following parameters throughout the simulations.

Parameters of NSGA-II

Fixed layout:	*Side ratio 1:1 (1/3) and Side ratio 1:4 (1/9)*	*Assumption*
Task assignment rule:	*Random*	Hwang and Katayama (2009)
Crossover:	*Weight mapping crossover (WMX)*	Hwang and Katayama (2009)
Crossover probability:	$P_C = 0.7$	Hwang *et al.* (2008)
Mutation:	*Swap or reciprocal mutation*	Hwang and Katayama (2009)
Mutation probability:	$P_M = 0.3$	Hwang *et al.* (2008)
Population size:	*100*	Hwang *et al.* (2008)
Walking time in each problem:	*% APT at the end of Chapter IV*	Pilot run
Generation:	*100 for 7 to 45 tasks*	Olanviwatchai (2009)
	150 for 61 and 70 tasks	Olanviwatchai (2009)
	300 for 111 tasks	Olanviwatchai (2009)
	1 day (the same as 111 tasks) for 297 tasks	*Assumption*

<u>Initialization of MA</u>

The algorithm applies the following parameters throughout the simulations.

Parameters of MA

Fixed layout:	*Side ratio 1:1 (1/3) and Side ratio 1:4 (1/9)*	*Assumption*
Task assignment rule:	*Random*	Hwang and Katayama (2009)
Crossover:	*Weight mapping crossover (WMX)*	Hwang and Katayama (2009)
Crossover	$P_C = 0.7$	Hwang *et al.* (2008)

probability:		
Mutation:	*Swap or reciprocal mutation*	Hwang and Katayama (2009)
Mutation probability:	$P_M = 0.3$	Hwang *et al.* (2008)
Local search after initial population:	*Pairwise Interchange (PI)*	Olanviwatchai (2009)
Local search after mutation:	*Insertion Procedure (IP)*	Olanviwatchai (2009)
Local search probability:	$P_L = 0.8$	Chutima and Pinkoompee (2008)
Population size:	*100*	Hwang *et al.* (2008)
Walking time in each problem:	*% APT at the end of Chapter IV*	Pilot run
Generation:	*100 for 7 to 45 tasks*	Olanviwatchai (2009)
	150 for 61 and 70 tasks	Olanviwatchai (2009)
	300 for 111 tasks	Olanviwatchai (2009)
	1 day (the same as 111 tasks) for 297 tasks	*Assumption*

Initialization of COIN

The algorithm applies the following parameters throughout the simulations.

Parameters of COIN

Fixed layout:	*Side ratio 1:1 (1/3) and Side ratio 1:4 (1/9)*	*Assumption*
Task assignment rule:	*Random*	Hwang and Katayama (2009)
Learning probability:	$k = 0.1$	Wattanapornprom *et al.* (2009)
Population size:	*100*	Hwang *et al.* (2008)
Walking time in each problem:	*% APT at the end of Chapter IV*	Pilot run

Generation:	*100 for 7 to 45 tasks*	Olanviwatchai (2009)
	150 for 61 and 70 tasks	Olanviwatchai (2009)
	300 for 111 tasks	Olanviwatchai (2009)
	1 day (the same as 111 tasks) for 297 tasks	*Assumption*

<u>Initialization of PSONK</u>

The algorithm applies the following parameters throughout the simulations.

Parameters of PSONK

Fixed layout:	*Side ratio 1:1 (1/3) and Side ratio 1:4 (1/9)*	*Assumption*
Task assignment rule:	*Random*	Hwang and Katayama (2009)
Number of particles in each swarm	*10**	*Assumption*
Number of swarms	*10**	*Assumption*
Cognitive component	$C_1 = 1$	Salman *et al.* (2002)
Social component	$C_2 = 1$	Salman *et al.* (2002)
Inertia weight	$\omega = 1$	*Assumption*
Learning coefficient:	$r_1, r_2 = 0.1$	Wattanapornprom *et al.* (2009)
Walking time in each problem:	*% APT at the end of Chapter IV*	Pilot run
Generation:	*100 for 7 to 45 tasks*	Olanviwatchai (2009)
	150 for 61 and 70 tasks	Olanviwatchai (2009)
	300 for 111 tasks	Olanviwatchai (2009)
	1 day (the same as 111 tasks) for 297 tasks	*Assumption*

* Number of swarms × number of particles in each swarm 10 × 10 = *100* [Population size's Hwang *et al.* (2008)]

6.3.2 Comparison of the computational results and analysis

All objective values of DOW and WT at the minimum number of workers in each problem of NSGA-II, MA, COIN, and PSONK are shown thoroughly in Appendix A and supplementary CD-ROM. To access the achievement of DOW and WT goals, these experiments are compared among all algorithms by the Pareto-optimal frontier in four aspects:

1. Convergence to the Pareto-optimal set;

2. Spread to the Pareto-optimal set;

3. Ratio of non-dominated solution;

4. Processing time *compared with the same iterations*.

In order to demonstrate the effectiveness of four approaches, computational results are obtained on a set of single U-shaped assembly line worker allocation problems with multiple objectives. For given cycle times, this research aims to study at most three values of each problem with the minimum, middle and maximum values. Walking distance is calculated with displacement. It is assumed that one walking unit (second) is equivalent to one walking distance unit (pace). To validate the feasibility of workers (workstations), experimental results are compared with ULINO data sets of lower bounds available from http://www.assembly-line-balancing.de. All algorithms are programmed by using MATLAB R2008a, and the set of test problems are solved on an AMD Athlon^{TM} 64 Processor 3500+ 2.21 GHz PC with 960 MB DDR-SDRAM. All numbers of workers are feasible and most are the same. All results are shown in Table 6.1-6.4 at the side ratio of 1:1:1 (1/3) and Table 6.5-6.8 at the side ratio of 1:4:4 (1/9).

The best algorithm should provide the convergence and spread of the solution to zero and its ratio to one. COIN and PSONK seems to perform better than NSGA-II for most problem sets between Columns IV-VI. Furthermore, in terms of CPU time in the last column, the multi-objective PSONK is faster than COIN and much faster than NSGA-II and MA. However, comparing NSGAII, MA, COIN, and PSONK, MA takes maximize CPU time for all problems.

According to an example at the side ratio of 1:4:4 (1/9) cited in Sirovetnukul and Chutima (2010a), the results of NSGA-II and PSONK shown in Fig. 6.2 are also compared at the 11-task problem. At the minimum number of five workers, the best Pareto-optimum frontier gives the same good solutions as pointed out in Fig. 6.2 for both algorithms. In terms of convergence and ratio of non-dominated solutions, PSONK is more potential, but spread is quite similar. The performances of three measures of at least 45 tasks are the same as the previous results of the 11-task problem. In contrast to most small-sized problems between 7-28 tasks, NSGA-II is preferable to PSONK. However, the 70-task problem at the cycle time of 160 seconds and the 111-task problem at 6,837 seconds are not relevant to these measures since PSONK provides fewer workers than the minimum with NSGA-II. For the 61-task problem and the large problem of 297 tasks, NSGA-II provides fewer workers than the minimum with PSONK. In CPU time, not only PSONK can reach Pareto-optimum solutions faster than NSGA-II for a sample problem, but also the rest of problems are definitely fast convergence rapidity.

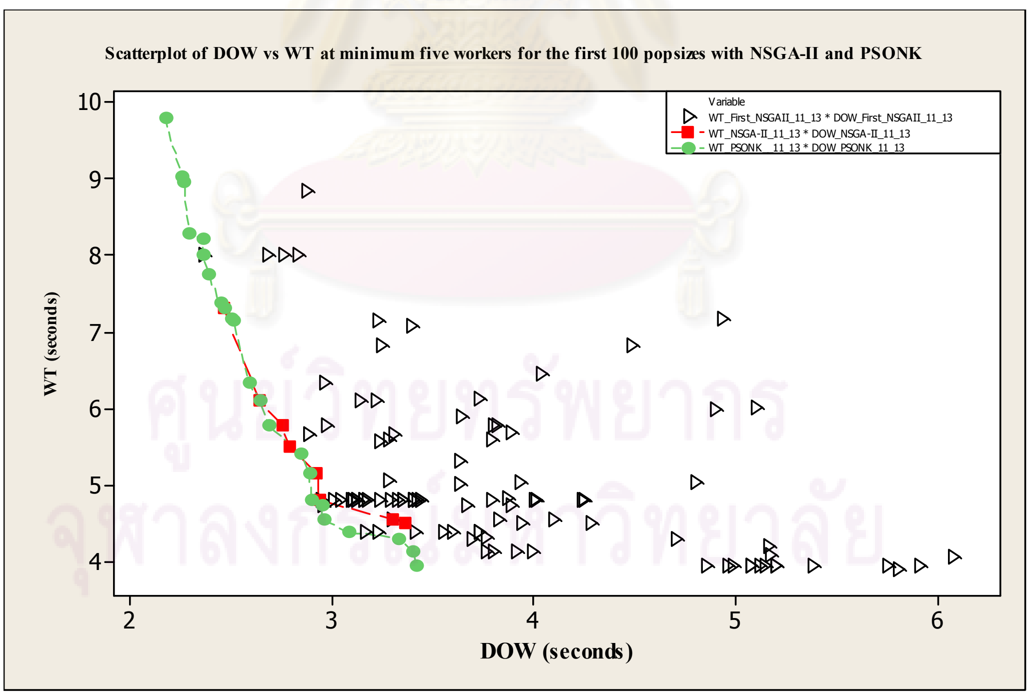


Figure 6.2 NSGA-II vs. PSONK for the 11-task problem of 13 cycle time

The three-dimensional scatter plotting of all algorithms are compared and shown in Figure 6.3-6.6. The tiny, small, medium, and large problems as seen in the following pictures are exemplified with the problems of 11 tasks given 21 seconds, 70 tasks given 527 seconds, 111 tasks given 17,067 seconds, and 297 tasks given 2,787 seconds.

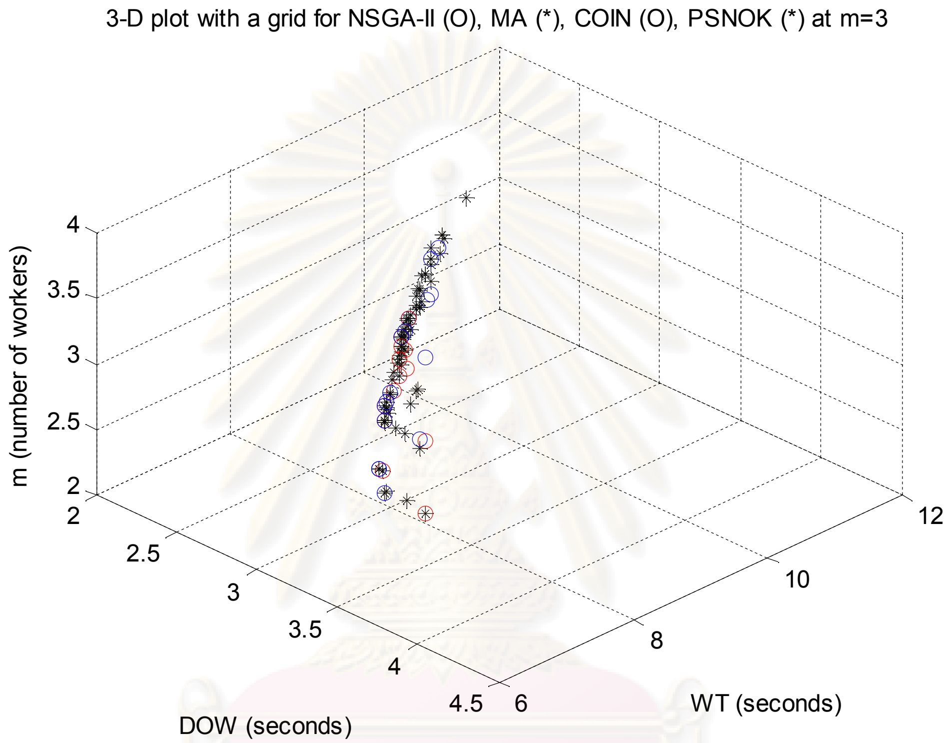


Figure 6.3 3-D graph at the side ratio 1:1:1 (1/3) for 11-task problem of 21 seconds

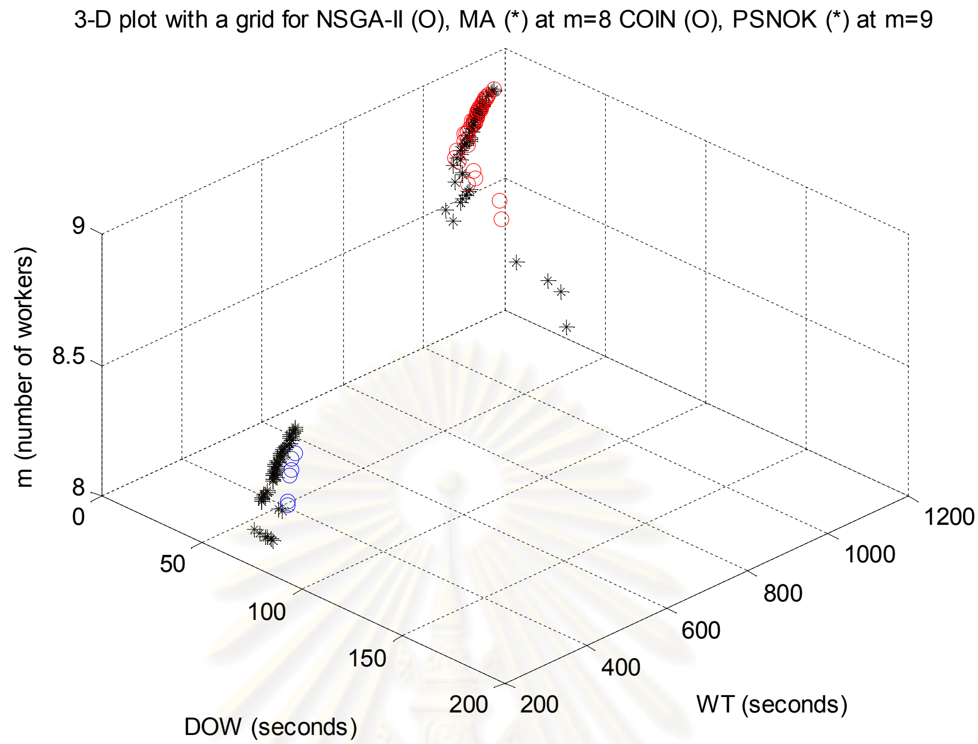


Figure 6.4 3-D graph at the side ratio 1:1:1 (1/3) for 70-task problem of 527 seconds

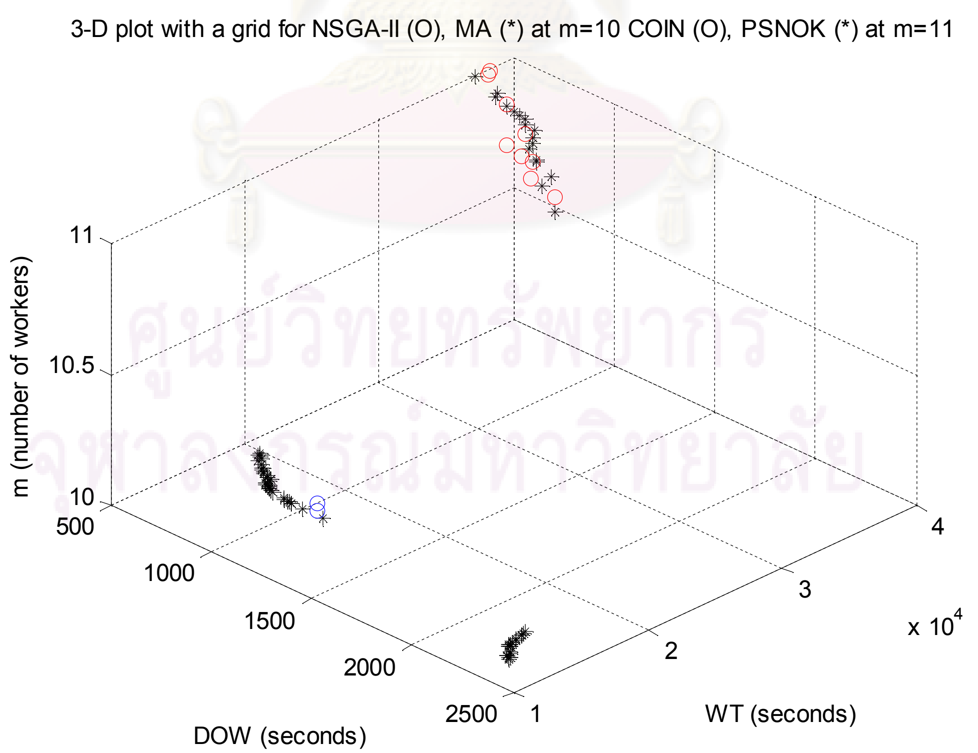


Figure 6.5 3-D graph at the side ratio 1:1:1 (1/3) for 111-task problem of 17,067 seconds

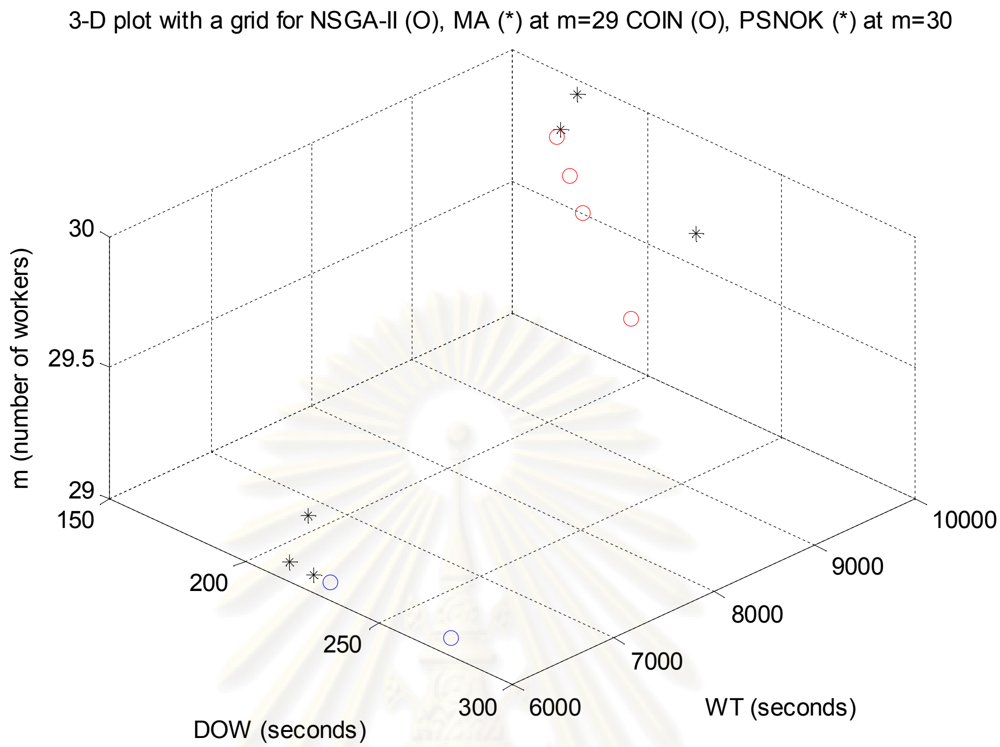


Figure 6.6 3-D graph at the side ratio 1:1:1 (1/3) for 297-task problem of 2,787 seconds

6.4 Discussion of NSGA-II, MA, COIN, and PSONK

This study is mainly about the worker allocation for U-shaped assembly line with four multi-objective algorithms. Since COIN originated only one generator with negative knowledge, a new PSONK has been developed with the addition of negative knowledge to renowned PSO. In particular, local best and global best recognize the positive knowledge appearing in the order pairs of the good solution by giving an increased reward to the updated joint probability matrix. In contrast, the negative knowledge, which is often remiss in NSGA-II and MA algorithms, is found in the order pairs of the bad solution. To prevent undesired solutions, it is utilized for local worst and global worst to reduce the updated joint probability. The comparative study shows that in most problems the proposed PSONK produces solution sets that are preferable to NSGA-II, MA, and COIN in terms of convergence and CPU time.

Table 6.1 NSGA-II with displacement for worker allocation at the side ratio of 1:1:1 (1/3)

Problem / Task	Cycle Time	No. of workers	Convergence	Spread	Ratio	Time (seconds)
1.Merten / 7	7	6	0	0.5364	1	1,084
	10	4	0.1295	0.7188	1	1,569
	18	2	0.0082	0.5077	1	968
2.Miltenburg / 10	10	4	0.0502	0.6582	0.6667	2,712
3.Jackson / 11	7	9	0.0067	0.4849	1	2,325
	13	5	0.0393	0.4894	0.7895	3,103
	21	3	0.0322	0.6846	0.6471	2,606
4.Thomopoulos / 19	120	5	0.9368	0.6088	0	4,557
5.Heskiaoff / 28	138	9	0.0259	0.6337	0.3415	4,737
	256	5	0.0349	0.5136	0.1905	4,977
	342	4	0.0290	0.5634	0.2759	6,836
6.Kilbridge & Wester / 45	57	*14*	-	-	-	11,809
	110	7	0.0820	0.7459	0.2000	11,418
	184	*4*	None**	None**	None**	11,735
7.Kim / 61	600	*11*	-	-	-	27,388
8.Tongue / 70	160	*29*	-	-	-	35,672
	251	17	None**	None**	None**	30,714
	527	*8*	0.1576	0.7131	0	27,301
9.Arcus / 111	6,837*	*27*	-	-	-	159,981
	7,916	*23*	-	-	-	133,348
	17,067	*10*	0.4523	0.7500	0	114,256
10. Scholl & Klein / 297	1,394	*62*	-	-	-	144,817
	1,834	*46*	-	-	-	144,147
	2,787	*29*	0.3414	0.7500	0.5000	166,070
11.Case study / 36	1,371	6	0.0329	0.7636	0.2951	3,438

* Minimum cycle time (5,755) is less than the operation time of 6,615. Thus, the feasible minimum cycle time from the data sets of UALBP-I is replaced.

** One local optimal solution (or one coordinate) on the DOW and WT

Table 6.2 MA (PI) with displacement for worker allocation at the side ratio of 1:1:1 (1/3)

Problem / Task	Cycle Time	No. of workers	Convergence	Spread	Ratio	Time (seconds)
1.Merten / 7	7	6	0	0.5364	1	2,034
	10	4	0.0161	0.7068	1	1,981
	18	2	0	0.6884	1	1,301
2.Miltenburg / 10	10	4	0.0068	0.8109	0.8571	1,782
3.Jackson / 11	7	9	0.0007	0.5343	0.9474	2,068
	13	5	0.0183	0.6141	0.7308	1,620
	21	3	0.0001	0.7806	1	1,504
4.Thomopoulos / 19	120	5	0.8045	0.7650	0	2,941
5.Heskiaoff / 28	138	9	0.1874	1	0.8333	4,322
	256	5	0.0155	0.7675	0.7531	3,869
	342	4	0.0013	0.6517	0.9596	3,727
6.Kilbridge & Wester / 45	57	*13*	0.1714	1	0	4,928
	110	7	0.0035	0.5763	1	5,069
	184	*4*	0.0030	0.6911	1	5,952
7.Kim / 61	600	*10*	-	-	-	13,079
8.Tongue / 70	160	*28*	-	-	-	22,499
	251	17	0.0123	0.8065	0.9333	16,361
	527	*8*	0	0.7841	1	16,118
9.Arcus / 111	6,837*	*27*	-	-	-	122,022
	7,916	*22*	0.3897	0.7798	1	113,651
	17,067	*10*	0	1	0.8571	114,134
10. Scholl & Klein / 297	1,394	*61*	-	-	-	485,946
	1,834	*45*	None**	None**	None**	506,220
	2,787	*29*	0.1678	0.7518	1	488,792
11.Case study / 36	1,371	6	0.0100	0.7007	0.6900	3,681

* Minimum cycle time (5,755) is less than the operation time of 6,615. Thus, the feasible minimum cycle time from the data sets of UALBP-I is replaced.

** One local optimal solution (or one coordinate) on the DOW and WT

Table 6.3 COIN with displacement for worker allocation at the side ratio of 1:1:1 (1/3)

Problem / Task	Cycle Time	No. of workers	Convergence	Spread	Ratio	Time (seconds)
1.Merten / 7	7	6	None**	None**	None**	246
	10	4	0.0743	0.8252	0.5000	276
	18	2	0.1425	0.7559	1	255
2.Miltenburg / 10	10	4	0.0340	0.8447	0.3333	354
3.Jackson / 11	7	9	0.0147	0.5072	0.9333	278
	13	5	0.0412	0.5399	0.5000	350
	21	3	0.0322	0.6846	0.3846	423
4.Thomopoulos / 19	120	5	0.8513	0.6563	0	485
5.Heskiaoff / 28	138	9	0.0259	0.6337	0.3415	638
	256	5	0.0795	0.7036	0.5781	591
	342	4	0.1404	0.6953	0.3077	742
6.Kilbridge & Wester / 45	57	*13*	0.0672	0.7726	0.6364	965
	110	7	0.0958	0.4768	0.1000	1,029
	184	*5*	-	-	-	1,134
7.Kim / 61	600	*11*	-	-	-	1,819
8.Tongue / 70	160	*26*	-	-	-	2,298
	251	17	0.1118	0.6091	0.2500	2,562
	527	*9*	-	-	-	2,122
9.Arcus / 111	6,837*	*26*	-	-	-	7,397
	7,916	*22*	0.0765	0.7818	0.6667	7,491
	17,067	*11*	-	-	-	7,631
10. Scholl & Klein / 297	1,394	*60*	0.0597	0.7667	1	5,852
	1,834	*45*	0.1616	0.6761	1	5,716
	2,787	*30*	-	-	-	5,744
11.Case study / 36	1,371	6	0.5187	0.5560	0	801

* Minimum cycle time (5,755) is less than the operation time of 6,615. Thus, the feasible minimum cycle time from the data sets of UALBP-I is replaced.

** One local optimal solution (or one coordinate) on the DOW and WT

Table 6.4 PSONK with displacement for worker allocation at the side ratio of 1:1:1 (1/3)

Problem / Task	Cycle Time	No. of workers	Convergence	Spread	Ratio	Time (seconds)
1.Merten / 7	7	6	None**	None**	None**	109
	10	4	0.1023	0.7438	0.6000	121
	18	2	0.1425	0.7559	1	103
2.Miltenburg / 10	10	4	0.0418	0.7646	0.1333	155
3.Jackson / 11	7	9	0.0094	0.5258	0.8824	168
	13	5	0.0154	0.5988	0.7083	153
	21	3	0.0915	0.6051	0.2667	129
4.Thomopoulos / 19	120	5	0.8289	0.6109	0	233
5.Heskiaoff / 28	138	9	0.0248	0.6470	0.4167	437
	256	5	0.0220	0.6559	0.3611	315
	342	4	0.1396	0.7903	0.3529	293
6.Kilbridge & Wester / 45	57	*13*	0.0189	0.5921	0.8824	533
	110	7	0.1612	0.7323	0	510
	184	*5*	-	-	-	532
7.Kim / 61	600	*11*	-	-	-	1,230
8.Tongue / 70	160	*27*	-	-	-	1,578
	251	17	0.1253	0.6545	0	1,633
	527	*9*	-	-	-	1,462
9.Arcus / 111	6,837*	*25*	-	-	-	5,100
	7,916	*22*	0.1744	0.7500	1	5,712
	17,067	*11*	-	-	-	5,114
10. Scholl & Klein / 297	1,394	*60*	0.0607	0.7277	0.5714	5,522
	1,834	*45*	0.0880	0.6612	0.8000	5,239
	2,787	*30*	-	-	-	5,503
11.Case study / 36	1,371	6	0.0523	0.8859	0.6267	374

* Minimum cycle time (5,755) is less than the operation time of 6,615. Thus, the feasible minimum cycle time from the data sets of UALBP-I is replaced.

** One local optimal solution (or one coordinate) on the DOW and WT

Table 6.5 NSGA-II with displacement for worker allocation at the side ratio of 1:4:4 (1/9)

Problem / Task	Cycle Time	No. of workers	Convergence	Spread	Ratio	Time (seconds)
1.Merten / 7	7	6	0	0.5364	1	1,084
	10	4	0.1295	0.7188	1	1,459
	18	2	0.0082	0.5077	1	1,503
2.Miltenburg / 10	10	4	0.0355	0.4861	0.3889	3,384
3.Jackson / 11	7	9	0.0522	0.5744	0.7778	3,417
	13	5	0.1220	0.6738	0.1250	4,081
	21	3	0.0445	0.7597	0.2857	3,053
4.Thomopoulos / 19	120	5	0.1360	0.9145	0.1429	5,101
5.Heskiaoff / 28	138	9	0.1195	0.6077	0	8,097
	256	5	0.4578	0.5073	0	6,782
	342	4	0.0271	0.6507	0.0952	6,498
6.Kilbridge & Wester / 45	57	13	0.3158	0.7500	0.5000	13,300
	110	7	0.0625	0.5194	0	11,354
	184	4	0.0555	0.5465	0.2000	10,358
7.Kim / 61	600	*10*	None**	None**	None**	21,015
8.Tongue / 70	160	*29*	-	-	-	34,533
	251	17	0.1655	0.8016	0.2500	33,043
	527	8	0.0553	0.7918	0.2778	20,188
9.Arcus / 111	6,837*	*27*	-	-	-	103,151
	7,916	*23*	-	-	-	95,679
	17,067	10	0.0881	0.6559	0	95,257
10. Scholl & Klein / 297	1,394	*60*	None**	None**	None**	123,051
	1,834	*46*	-	-	-	119,709
	2,787	*29*	0.4261	0.7500	0	120,788
11.Case study / 36	1,371	6	0.0201	0.6221	0.1957	3,489

* Minimum cycle time (5,755) is less than the operation time of 6,615. Thus, the feasible minimum cycle time from the data sets of UALBP-I is replaced.

** One local optimal solution (or one coordinate) on the DOW and WT

Table 6.6 MA (PI) with displacement for worker allocation at the side ratio of 1:4:4 (1/9)

Problem / Task	Cycle Time	No. of workers	Convergence	Spread	Ratio	Time (seconds)
1.Merten / 7	7	6	0	0.5364	1	1,058
	10	4	0.0161	0.7068	1	1,389
	18	2	0	0.6884	1	1,070
2.Miltenburg / 10	10	4	0.0076	0.7949	0.8800	1,756
3.Jackson / 11	7	9	0.0062	0.7350	1	2,114
	13	5	0.0074	0.5363	0.7600	1,835
	21	3	0	0.8117	1	1,618
4.Thomopoulos / 19	120	5	0.0108	0.6445	0.3731	2,901
5.Heskiaoff / 28	138	9	0.3147	0.8562	0.8750	4,163
	256	5	0.3337	0.7024	0	2,808
	342	4	0.0001	0.7258	0.9796	3,590
6.Kilbridge & Wester / 45	57	13	0.2130	0.6652	0.6000	6,263
	110	7	0.0112	0.5992	0.7273	6,092
	184	4	0.0181	0.6213	0.7065	6,159
7.Kim / 61	600	*10*	0	0.7651	1	12,552
8.Tongue / 70	160	*27*	None**	None**	None**	16,583
	251	17	0.0808	0.6058	1	16,657
	527	8	0.0053	0.6743	0.7604	16,388
9.Arcus / 111	6,837*	*26*	0.2059	0.6734	1	107,847
	7,916	*22*	-	-	-	105,640
	17,067	10	0.0124	0.7559	0.8046	102,541
10. Scholl & Klein / 297	1,394	*60*	0	0.7500	1	483,324
	1,834	*44*	-	-	-	483,971
	2,787	*29*	0	0.8645	1	483,642
11.Case study / 36	1,371	6	0.0044	0.7267	0.8415	3,707

* Minimum cycle time (5,755) is less than the operation time of 6,615. Thus, the feasible minimum cycle time from the data sets of UALBP-I is replaced.

** One local optimal solution (or one coordinate) on the DOW and WT

Table 6.7 COIN with displacement for worker allocation at the side ratio of 1:4:4 (1/9)

Problem / Task	Cycle Time	No. of workers	Convergence	Spread	Ratio	Time (seconds)
1.Merten / 7	7	6	None**	None**	None**	221
	10	4	0.0743	0.8252	0.5000	234
	18	2	0.1425	0.7559	1	231
2.Miltenburg / 10	10	4	0.0391	0.8108	0.1765	353
3.Jackson / 11	7	9	0.0250	0.6902	0.7500	315
	13	5	0.0316	0.6328	0.3636	489
	21	3	0.0284	0.5706	0.2632	427
4.Thomopoulos / 19	120	5	0.0132	0.5269	0.7083	510
5.Heskiaoff / 28	138	9	0.0175	0.5294	0.5128	834
	256	5	0.3847	0.7277	0	787
	342	4	0.1258	0.7517	0.1053	869
6.Kilbridge & Wester / 45	57	13	0.2581	0.5961	1	1,072
	110	7	0.0389	0.7058	0.2969	926
	184	4	0.0105	0.5256	0.3673	920
7.Kim / 61	600	*11*	-	-	-	1,859
8.Tongue / 70	160	*27*	0	0.5444	1	2,271
	251	17	0.1162	0.4348	0.3571	2,323
	527	8	0.0593	0.5745	0.3750	2,097
9.Arcus / 111	6,837*	*26*	0.1380	0.6210	0.8889	7,468
	7,916	*23*	-	-	-	7,157
	17,067	10	0.1070	0.6521	0.3673	7,196
10. Scholl & Klein / 297	1,394	*60*	None**	None**	None**	5,548
	1,834	*46*	-	-	-	5,621
	2,787	*35*	-	-	-	5,589
11.Case study / 36	1,371	6	0.3393	0.6739	0.1071	825

* Minimum cycle time (5,755) is less than the operation time of 6,615. Thus, the feasible minimum cycle time from the data sets of UALBP-I is replaced.

** One local optimal solution (or one coordinate) on the DOW and WT

Table 6.8 PSONK with displacement for worker allocation at the side ratio of 1:4:4 (1/9)

Problem / Task	Cycle Time	No. of workers	Convergence	Spread	Ratio	Time (seconds)
1.Merten / 7	7	6	None**	None**	None**	117
	10	4	0.1023	0.7558	0.7500	139
	18	2	0.1425	0.7559	1	115
2.Miltenburg / 10	10	4	0.0401	0.8390	0.2105	158
3.Jackson / 11	7	9	0.0354	0.8404	0.9231	164
	13	5	0.0278	0.5992	0.3478	161
	21	3	0.0196	0.5928	0.5417	143
4.Thomopoulos / 19	120	5	0.0069	0.6418	0.3229	308
5.Heskiaoff / 28	138	9	0.0228	0.6505	0.2857	451
	256	5	0.3754	0.9054	0	430
	342	4	0.1426	0.7935	0	430
6.Kilbridge & Wester / 45	57	13	0.0979	0.6456	0.3333	742
	110	7	0.0283	0.8041	0.5455	730
	184	4	0.0036	0.7459	0.6058	722
7.Kim / 61	600	*11*	-	-	-	1,420
8.Tongue / 70	160	*27*	0.1227	0.4320	0	1,853
	251	17	0.1287	0.6513	0.3333	1,880
	527	8	0.0482	0.6105	0.2667	2,113
9.Arcus / 111	6,837*	*26*	None**	None**	None**	6,194
	7,916	*23*	-	-	-	6,455
	17,067	10	0.0916	0.7579	0.3721	6,101
10. Scholl & Klein / 297	1,394	*61*	-	-	-	5,419
	1,834	*47*	-	-	-	5,203
	2,787	*31*	-	-	-	5,305
11.Case study / 36	1,371	6	0.0137	0.8548	0.2642	356

* Minimum cycle time (5,755) is less than the operation time of 6,615. Thus, the feasible minimum cycle time from the data sets of UALBP-I is replaced.

** One local optimal solution (or one coordinate) on the DOW and WT

6.5 Discussion of Given Cycle Times

For the cycle time ratio that influences on the task allocation of a problem by given cycle time from 7-task to 297-task problems, all the frequency distributions are shown in Table 6.9. It is noticed that if the high values of a number of tasks in the third column are skewed positively (asymmetric right skew, with a long tail to one side), then the findings of selecting a number of tasks into an assigned-task position in a U-line will be computed with long CPU time. On the contrary, the frequency distribution is a negative (or left) skewness; consequently, there are only one or two tasks allocated into the given cycle time. From Table 6.9, there are 22 cases, i.e. the cases of 3, 4, and 6-25 for right skew distributions and the rest of cases are three, i.e. the cases of 1, 2, and 5 for left skew distributions. By the graphical expression, the histogram of task time for the 11-task problem of the 13 cycle time is exemplified and shown in Figure 6.7. Finally, the tabulated results are shown obviously that most of cases are right skew and take long CPU time to allocate candidate tasks to a position in a U-line.

Table 6.9 Frequency distribution for the cycle time ratio data of 7-10 tasks

Case /cycle time (time units)	Interval class of cycle time ratio	Interval class of given cycle time	Frequency (No. of tasks)	Relative frequency	Cumulative relative frequency
1. 7 tasks /7	$0 \le x < 0.25$	$0 \le x < 1.75$	1	0.1429	0.1429
	$0.25 \le x < 0.50$	$1.75 \le x < 3.50$	1	0.1429	0.2858
	$0.50 \le x < 0.75$	$3.50 \le x < 5.25$	4	0.5714	0.8572
	$0.75 \le x < 1$	$5.25 \le x < 7$	1	0.1429	1
2. 7 tasks /10	$0 \le x < 0.25$	$0 \le x < 2.50$	1	0.1429	0.1429
	$0.25 \le x < 0.50$	$2.50 \le x < 5.00$	2	0.2857	0.4286
	$0.50 \le x < 0.75$	$5.00 \le x < 7.50$	4	0.5714	1
	$0.75 \le x < 1$	$7.50 \le x < 10$	0	0	1
3. 7 tasks /18	$0 \le x < 0.25$	$0 \le x < 4.50$	3	0.4286	0.4286
	$0.25 \le x < 0.50$	$4.50 \le x < 9.00$	4	0.5714	1
	$0.50 \le x < 0.75$	$9.00 \le x < 13.50$	0	0	1
	$0.75 \le x < 1$	$13.50 \le x < 18$	0	0	1
4. 10 tasks /10	$0 \le x < 0.25$	$0 \le x < 2.50$	5	0.5000	0.5000
	$0.25 \le x < 0.50$	$2.50 \le x < 5.00$	5	0.5000	1
	$0.50 \le x < 0.75$	$5.00 \le x < 7.50$	0	0	1
	$0.75 \le x < 1$	$7.50 \le x < 10$	0	0	1

Table 6.10 Frequency distribution for the cycle time ratio data of 11-61 tasks

Case /cycle time (time units)	Interval class of cycle time ratio	Interval class of given cycle time	Frequency (No. of tasks)	Relative frequency	Cumulative relative frequency
5. 11 tasks /7	$0 \le x < 0.25$	$0 \le x < 1.75$	1	0.0909	0.0909
	$0.25 \le x < 0.50$	$1.75 \le x < 3.50$	3	0.2727	0.3636
	$0.50 \le x < 0.75$	$3.50 \le x < 5.25$	4	0.3636	0.7273
	$0.75 \le x < 1$	$5.25 \le x < 7$	3	0.2727	1
6. 11 tasks /13	$0 \le x < 0.25$	$0 \le x < 3.25$	4	0.3636	0.3636
	$0.25 \le x < 0.50$	$3.25 \le x < 6.50$	6	0.5455	0.9091
	$0.50 \le x < 0.75$	$6.50 \le x < 9.75$	1	0.0909	1
	$0.75 \le x < 1$	$9.75 \le x < 13$	0	0	1
7. 11 tasks /21	$0 \le x < 0.25$	$0 \le x < 5.25$	5	0.4545	0.4545
	$0.25 \le x < 0.50$	$5.25 \le x < 10.50$	6	0.5455	1
	$0.50 \le x < 0.75$	$10.50 \le x < 15.75$	0	0	1
	$0.75 \le x < 1$	$15.75 \le x < 21$	0	0	1
8. 19 tasks /120	$0 \le x < 0.25$	$0 \le x < 30$	16	0.8421	0.8421
	$0.25 \le x < 0.50$	$30 \le x < 60$	1	0.0526	0.8947
	$0.50 \le x < 0.75$	$60 \le x < 90$	2	0.1053	1
	$0.75 \le x < 1$	$90 \le x < 120$	0	0	1
9. 28 tasks /138	$0 \le x < 0.25$	$0 \le x < 35$	17	0.6071	0.6071
	$0.25 \le x < 0.50$	$35 \le x < 69$	6	0.2143	0.8214
	$0.50 \le x < 0.75$	$69 \le x < 104$	3	0.1071	0.9286
	$0.75 \le x < 1$	$104 \le x < 138$	2	0.0714	1
10. 28 tasks /256	$0 \le x < 0.25$	$0 \le x < 64$	22	0.7857	0.7857
	$0.25 \le x < 0.50$	$64 \le x < 128$	6	0.2143	1
	$0.50 \le x < 0.75$	$128 \le x < 192$	0	0	1
	$0.75 \le x < 1$	$192 \le x < 256$	0	0	1
11. 28 tasks /342	$0 \le x < 0.25$	$0 \le x < 86$	25	0.8929	0.8929
	$0.25 \le x < 0.50$	$86 \le x < 171$	3	0.1071	1
	$0.50 \le x < 0.75$	$171 \le x < 257$	0	0	1
	$0.75 \le x < 1$	$257 \le x < 342$	0	0	1
12. 45 tasks /57	$0 \le x < 0.25$	$0 \le x < 14$	32	0.7111	0.7111
	$0.25 \le x < 0.50$	$14 \le x < 29$	11	0.2444	0.9556
	$0.50 \le x < 0.75$	$29 \le x < 43$	1	0.0222	0.9778
	$0.75 \le x < 1$	$43 \le x < 57$	1	0.0222	1
13. 45 tasks /110	$0 \le x < 0.25$	$0 \le x < 28$	43	0.9556	0.9556
	$0.25 \le x < 0.50$	$28 \le x < 55$	2	0.0444	1
	$0.50 \le x < 0.75$	$55 \le x < 83$	0	0	1
	$0.75 \le x < 1$	$83 \le x < 110$	0	0	1
14. 45 tasks /184	$0 \le x < 0.25$	$0 \le x < 46$	44	0.9778	0.9778
	$0.25 \le x < 0.50$	$46 \le x < 92$	1	0.0222	1
	$0.50 \le x < 0.75$	$92 \le x < 138$	0	0	1
	$0.75 \le x < 1$	$138 \le x < 184$	0	0	1
15. 61 tasks /600	$0 \le x < 0.25$	$0 \le x < 150$	58	0.9508	0.9508
	$0.25 \le x < 0.50$	$150 \le x < 300$	3	0.0492	1
	$0.50 \le x < 0.75$	$300 \le x < 450$	0	0	1
	$0.75 \le x < 1$	$450 \le x < 600$	0	0	1

Table 6.11 Frequency distribution for the cycle time ratio data of 70-297 and 36 tasks

Case /cycle time (time units)	Interval class of cycle time ratio	Interval class of given cycle time	Frequency (No. of tasks)	Relative frequency	Cumulative relative frequency
16. 70 tasks /160	$0 \le x < 0.25$	$0 \le x < 40$	34	0.4857	0.4857
	$0.25 \le x < 0.50$	$40 \le x < 80$	22	0.3143	0.8000
	$0.50 \le x < 0.75$	$80 \le x < 120$	7	0.1000	0.9000
	$0.75 \le x < 1$	$120 \le x < 160$	7	0.1000	1
17. 70 tasks /251	$0 \le x < 0.25$	$0 \le x < 63$	49	0.7000	0.7000
	$0.25 \le x < 0.50$	$63 \le x < 126$	15	0.2143	0.9143
	$0.50 \le x < 0.75$	$126 \le x < 188$	6	0.0857	1
	$0.75 \le x < 1$	$188 \le x < 251$	0	0	1
18. 70 tasks /527	$0 \le x < 0.25$	$0 \le x < 132$	65	0.9286	0.9286
	$0.25 \le x < 0.50$	$132 \le x < 264$	5	0.0714	1
	$0.50 \le x < 0.75$	$264 \le x < 395$	0	0	1
	$0.75 \le x < 1$	$395 \le x < 527$	0	0	1
19. 111tasks /6,837	$0 \le x < 0.25$	$0 \le x < 1709$	74	0.6667	0.6667
	$0.25 \le x < 0.50$	$1709 \le x < 3419$	25	0.2252	0.8919
	$0.50 \le x < 0.75$	$3419 \le x < 5128$	8	0.0721	0.9640
	$0.75 \le x < 1$	$5128 \le x < 6837$	4	0.0360	1
20. 111tasks /7,916	$0 \le x < 0.25$	$0 \le x < 1979$	83	0.7477	0.7477
	$0.25 \le x < 0.50$	$1979 \le x < 3958$	20	0.1802	0.9279
	$0.50 \le x < 0.75$	$3958 \le x < 5937$	7	0.0631	0.9910
	$0.75 \le x < 1$	$5937 \le x < 7916$	1	0.0090	1
21. 111tasks /17,067	$0 \le x < 0.25$	$0 \le x < 4402$	106	0.9550	0.9550
	$0.25 \le x < 0.50$	$4402 \le x < 8534$	5	0.0450	1
	$0.50 \le x < 0.75$	$8534 \le x < 12800$	0	0	1
	$0.75 \le x < 1$	$12800 \le x < 17067$	0	0	1
22. 297tasks /1,394	$0 \le x < 0.25$	$0 \le x < 349$	251	0.8451	0.8451
	$0.25 \le x < 0.50$	$349 \le x < 697$	33	0.1111	0.9562
	$0.50 \le x < 0.75$	$697 \le x < 1046$	10	0.0337	0.9899
	$0.75 \le x < 1$	$1046 \le x < 1394$	3	0.0101	1
23. 297tasks /1,834	$0 \le x < 0.25$	$0 \le x < 459$	273	0.9192	0.9192
	$0.25 \le x < 0.50$	$459 \le x < 917$	19	0.0640	0.9832
	$0.50 \le x < 0.75$	$917 \le x < 1376$	4	0.0135	0.9966
	$0.75 \le x < 1$	$1376 \le x < 1834$	1	0.0034	1
24. 297tasks /2,787	$0 \le x < 0.25$	$0 \le x < 697$	285	0.9596	0.9596
	$0.25 \le x < 0.50$	$697 \le x < 1394$	12	0.0404	1
	$0.50 \le x < 0.75$	$1394 \le x < 2090$	0	0	1
	$0.75 \le x < 1$	$2090 \le x < 2787$	0	0	1
25. 36 tasks /1,371	$0 \le x < 0.25$	$0 \le x < 343$	32	0.8889	0.8889
	$0.25 \le x < 0.50$	$343 \le x < 686$	4	0.1111	1
	$0.50 \le x < 0.75$	$686 \le x < 1028$	0	0	1
	$0.75 \le x < 1$	$1028 \le x < 1371$	0	0	1

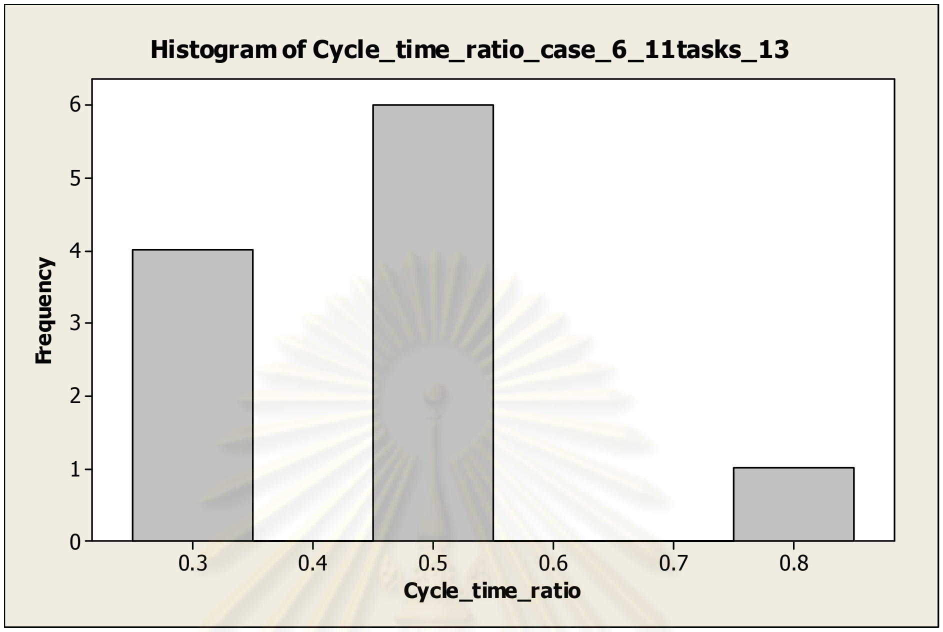


Figure 6.7 Histogram of cycle time ratio for the 11-task problem of the 13 cycle time

CHAPTER VII

CONCLUSION AND RECOMMENDATION FOR FUTURE RESEARCH

7.1 Introduction

This final chapter summarizes the findings and conclusion of the research. The exact algorithm was first studied to discover relevant factors on a small ten-task problem. Four multi-objective evolutionary algorithms are then developed to solve larger problems and a practical problem. The summary of experimental results closes the gap with the research objective and research questions posed at the beginning of the dissertation. Recommendation for future research is presented in the issues of bounds, heuristics, relaxation of some restrictions, and extension of the problem into other line configurations.

7.2 Conclusion

In conclusion, the importance of a U-shaped assembly line has increased to respond more product variety. Many manufactures use mixed-model production to produce different products on the same line. It helps them provide their customers with a variety of products in a timely and cost effective manner. Much research work has been done on the traditional line balancing problems. Recently, the U-shaped line balancing problems have been researched for almost two decades. A U-line is widely used in just-in-time production systems and well-suited to a mixed-model production. Previous research has compared a U-line to be more efficient than a straight line. Workers must be trained to complete many tasks. It is more reasonable undertaking in the fixed location layouts of all problems where only a limited number of tasks are feasible for any worker. In general the location of task in many manufacturing settings is fixed to a specific position of a production line due to machine and material handling constraints. However, the model in this study can be modified because of the existing industry conditions of manually small and inexpensive machines including mobility. The single assembly line balancing problem differs from the single U-shaped assembly line balancing problem. In the assembly line balancing problem, tasks may be assigned while moving through the graph in one direction (forward or backward) only, whereas in the single U-shaped line balancing problem tasks may be assigned to stations while moving through the precedence graph in two directions (forward and backward) at the same time. For a single U-line as one of other U-shaped types, there has been no prior documented work in the U-shaped worker allocation with 7-task to 297-task standard problems under one-piece flow production environment using the development of evolutionary algorithms. Most of the published work in optimum U-shaped worker allocation problem did not take into account the impact of walking time, medium-sized and large-sized benchmarked problems, and multi-criteria optimization. The performance index of all algorithms in this study is the minimum number of workers. Hierarchically, the deviation of operations of workers and walking time are evaluated with multi-objective performance at the same time as the Pareto-optimal frontier. The ULINO solutions of benchmarked data sets without walking time are used as lower bounds to come up with optimal solutions. Subsequently, the impact of walking time on the SUALWAPs of type I is conducted and subjected to a constraint. From the experimental results of symmetrical and rectangular U-shaped layouts, incrementing a number of workers in the former objective is sensitive to determine the walking time at only five percent of average processing time (equivalent to time units) in most problems. Just a few problems are at the ten and twenty percentage of average processing time (equivalent to time units). It gives the conclusion that a decision to change a little walking time significantly effects the supplement of a larger number of workers in a single U-line. After getting the fixed % average processing time from one task to another task of all problems, every worker is assigned to do task(s) by the consideration of the major objective and minor dual objectives respectively. Since the complexity of the problems falls into the NP-hard class of combinatorial optimization problems, the four multi-objective evolutionary algorithms of NSGA-II, MA, COIN, and PSONK are applied to SUALWAPs and conducted experiments to evaluate the application. Non-dominated Sorting Genetic Algorithm (NSGA-II) as a former multi-objective method is first solved in these problems. Secondly, Memetic Algorithm (MA) is extended by adding the concept of local search. Thirdly, combinatorial optimization with multi-objective COINcidence algorithm (COIN) as a novel evolutionary algorithm at combinatorial optimization has been applied successfully to this dissertation and many industrial engineering problems (Chongstitvatana *et al.*, 2010). It recognizes the positive knowledge appearing in the order pairs of the good solution by giving a marginal reward (increased probability) to its related element of the joint probability matrix. In contrast, the negative knowledge found in the order pairs of the bad solution, which is often remiss in most algorithms, is also utilized in COIN (reduced probability) to prevent undesirable solutions coincidentally found in this generation to be less recurrent in the next generation. Then, the negative knowledge of the coincidence algorithm is developed into Particle Swarm Optimization (PSO) as a renowned evolutionary technique. The fourth algorithm of this study is called Particle Swarm Optimization with Negative Knowledge (PSONK). From the results, it is quite evident that PSONK provides the objective functions optimal comparing with NSGA-II, MA, and COIN in most cases. Besides, the CPU time is quite short as compared to the others. In executing every algorithm, the algorithm takes significantly longer times for larger problem sets because the time is directly proportional to the number of tasks.

Capacity planning is the process of identifying necessary resources to meet fluctuating demands. Inadequate capacity planning can lead to the loss of customer demands. There are three phases for capacity planning as follows. First, long-term capacity planning is the plan for future plant capacity made by an executive manager. Secondly, medium-term capacity planning that is related to employment, layoffs, overtime, etc. is the planning based on the assumption that the capacity of the plant does not change. Finally, short-term capacity planning that is related to material availability, absenteeism rate, etc. involves the day to day issues and decisions in operations planning. Thus, assembly line balancing and worker allocation problems are important tasks in medium-term production planning that concern the installation of the line and the division of work among stations. From the horizontal time, these problems should be planned for making decisions before a few months. From doing our experiments with four algorithms previously, the best case of PSONK in the 297-task problem took a few hours only and even the worst case of MA taking the longest time spent about six days in the same problem. It is no doubt to make a conclusion that a decision maker from a plant can use the ameliorated algorithm of PSONK to achieve the U-shaped assembly line worker allocation problem by time schedule.

7.3 Recommendation for Future Research

Numerous research opportunities remain in the search for greater assembly line efficiency and more accurate depiction of manufacturing situations and difficulties although many aspects of U-shaped worker allocation were studied in this dissertation. The objective of the dissertation was to model and investigate the characteristics of U-shaped worker allocation problems and to propose a method of solving them. Other directions and the effectiveness of the algorithms may be improved through further research into the following topics.

7.3.1 Bounds

Owing to the feasibility of bound associated with the U-shaped worker allocation problem, processing and walking times are significant factors in algorithm performance. Although several bounds were introduced in the U-shaped line balancing algorithms, the tightness of the bounds should be also improved particularly relative to U-shaped worker allocation algorithms and extended from the only Miltenburg's 10-task problem to other standard problems. The tradeoff between computational requirements for bound calculation and potential improvements in algorithm performance must be evaluated. However, difficult worker allocation problems such 111-task and 297-task problems cannot be solved optimally and dictates the use of heuristic solution procedures.

7.3.2 Heuristics

Even though the PSONK used in this study proved to be the best effective evolutionary algorithm for solving SUALWAPs, the development of PSONK procedures could be improved in the quality of solutions and the computation time. The further improvements of PSONK should be focused on the memory of the wrong first walk losing opportunity to make bad solutions and the addition of the beneficial local search in MA to make better solutions.

7.3.3 Relaxation of some restrictions for SUALWAPs

Some conditions may be relaxed in SUALWAPs as follows:

- A mixed-model version of the simple U-line can be developed. Although some problems in this study input many products, task times from different precedence graphs are averaged as a single product. The benefits for using mixed-model U-lines having stochastic processing times remain untested. These issues may involve more elaborate simulation methodology.

- In this study it is assumed that all workers have the same processing time at each machine. If skills of workers are different, then the worker allocation to machines according to their skills will be taken into account. In practice, the skilled worker has to wait for the completion of operations of a new worker in the U-line. The problem for the stochastic model, in which processing and walking times are stochastic, is also important. As a result, other computation algorithms should be developed.

- The relevance of the learning curve on task assignment rules should be considered as a significant topic.

- The model developed here is three of multiple criteria for the decision making approach. A decision maker may be interested in measuring the success of these problems from other groups of criteria.

- In addition, to achieve a “satisfactory” rather than “optimal” solution other goals, i.e. max-min, min-max, and max-max curves in addition to a conflicting goal (min-min curve) may be developed.

- Extending the proposed approaches by considering the worker’s parallel workstations, fixed task locations, zoning constraints, etc. should be also considered in the future study.

- To allocate tasks in other real assembly U-lines, the distance that is equivalent to %APT may be adjusted.

- Other possibilities may be taken into account for other important aspects of real environments, e.g. the period time of rebalancing a U-line should be studied; the balancing and sequencing problem should be fulfilled at the same time; and the effect of setup times should be also considered on a new mixed-model sequencing U-line problem.

7.3.4 Extension of the single U-line worker allocation into other line configurations

More complex U-shaped lines can be developed for practical problems such as these where stations share tasks or where balances span more than one line. It is possible to extend the proposed methods for more complex U-lines such as multi-lines in a single U-line, double dependent U-lines, embedded U-lines, and multi U-line facilities as well as automated U-lines. It is interesting for setting a research question whether the shaped of the line has affected on the other performance measures such as cost, speed and flexibility. Many of these issues will be dealt with and extended in the future research.

REFERENCES

Aase, G. R., Olson, J. R., and Chniederjans, M. J. 2004. U-shaped assembly line layouts and their impact on labour productivity: An experimental study. **European Journal of Operational Research** 56: 698-711.

Ajenblit, D. A. and Wainwright, R. L. 1998. APPLYING GENETIC ALGORITHMS TO THE U-SHAPED ASSEMBLY LINE BALANCING PROBLEM. **Proceedings of the 1998 IEEE International Conference on Evolutionary Computation**, Anchorage, Alaska.

Allahverdi, A., Ng, C. T., Cheng, T. C. E., and Kovalyov, M. Y. 2008. A survey of scheduling problems with setup times or costs. **European Journal of Operational Research** 187: 985-1032.

Andres, C., Miralles, C., and Pastor, R. 2006. Balancing and scheduling tasks in assembly lines with sequence-dependent setup times. **European Journal of Operational Research** 187(3): 1212-1223.

ANOM. 1994. Shingo Prize for excellence in Manufacturing: 1994-95 Application Guidelines. Logan, UT, College of Business, Utah State University.

Arcus, A. L. 1966. COMSOAL: A Computer Method of Sequencing Operations for Assembly Lines. **International Journal of Production Research** 4(4): 259-277.

Baker, K. R. 1974. **Introduction to Sequencing and Scheduling**. Canada, John Wiley&Sons.

Balakrishnan, J., Cheng, C. H., Ho, K. C., and Yang, K. K. 2009. The application of single-pass heuristics for U-lines. **Journal of Manufacturing Systems** 28(1): 28-40.

Bard, J. 1989. Assembly line balancing with parallel workstations and dead time. **International Journal of Production Research** 27(6): 1005-1018.

Bard, J. F., Dar-El, E., and Shtub, A. 1992. An analytic framework for sequencing mixed model assembly lines. **International Journal of Production Research** 30: 35-48.

Bautista, J. and Pereira, J. 2002. Ant algorithms for assembly line balancing. **Lecture Notes in Computer Science**: 2463, 65-75.

Baybars, I. 1986. An Efficient Heuristic Method for the Simple Assembly Line Balancing Problem. **International Journal of Production Research** 24(1): 149-166.

Baybars, I. 1986. A survey of exact algorithms for the simple assembly line balancing problem. **Management Science** 32(8): 909-932.

Baybars, I. and Frieze, A. 1986. Expected behavior of line balancing heuristics. **IMA Journal of Mathematics in Management** 10: 304-335.

Baykasoglu, A. 2006. Multi-rule multi-objective simulated annealing algorithm for straight and U type assembly line balancing problems. **Journal of Intelligent Manufacturing** 17: 217-232.

Baykasoglu, A. and Dereli, T. 2009. Simple and U-type assembly line balancing by using an ant colony based algorithm. **Mathematical and Computational Applications** 14(1): 1-12.

Becker, C. and Scholl, A. 2006. A survey on problems and methods in generalized assembly line balancing. **European Journal of Operational Research** 168: 694-715.

Beenhakker, H. L. 1963. **Development of alternate criteria for optimality in the machine sequencing problem**, Purdue University.

Berto, T. P. and Ferreira, J. C. E. **LINE BALANCING WITH GENETIC ALGORITHMS**. em 18th Congresso Brasileiro de Engenharia Mecânica - COBEM'2005, Ouro Preto, MG, v. CD-ROM, novembro de.

Bhaskar, K. and Srinivasan, G. 1997. Static and dynamic operator allocation problems in cellular manufacturing systems. **International Journal of Production Research** 35: 3467-3481.

Booker, L. B., Goldberg, D. E., and Holland, J. H. 1989. **Classifier systems and genetic algorithms in Machine Learning: Paradigms and Methods**. Cambridge, MA: MIT Press/Elsevier, 235-282.

Bowman, E. H. 1960. Assembly Line Balancing by Linear Programming. **Operations Research** 8(3): 385-389.

Box, G. E. P. 1957. Evolutionary operation: A method for increasing industrial productivity. **J. Roy. Statist. Soc.** 6(2): 81-101.

Boysen, N., Fliedner, M., and Scholl, A. 2006. A classification of assembly line balancing problems. **European Journal of Operational Research** 168: 694-715.

Boysen, N., Fliedner, M., and Scholl, A. 2007. Sequencing mixed-model assembly lines: Survey, classification and model critique. **European Journal of Operational Research** In press, Corrected Proof.

Bremermann, H. J. 1962. **Optimization through evolution and recombination in Self-Organizing Systems**. Washington, DC: Spartan.

Bukchin, J., Dar-El, E. M., and Rubinovitz, J. 2002. Mixed model assembly line design in a make-to-order environment. **Computers & Industrial Engineering** 41: 405-421.

Campbell, G. M. and Diaby, M. 2002. Development and evaluation of an assignment heuristic for allocating cross-trained workers. **European Journal of Operational Research** 138: 9-20.

Celano, G., Ficher, S., Grasso, V., La Commare, U., and Perrone, G. 1999. An evolutionary approach to multi-objective scheduling of mixed model assembly lines. **Computers and Industrial Engineering** 37(1-2): 69-73.

Cesani, V. I. and. Steudel, H. J. 2005. A study of labor assignment flexibility in cellular manufacturing systems. **Computers and Industrial Engineering** 48: 571-591.

Chase, R. B., Aquilano, N. J., and Jacobs, F. R. 1998. **Production and Operations Management: Manufacturing and Services**. (Eight ed.): McGraw-Hill companies.

Chen, H. G. 1991. A mixed integer programming model for operator cyclic walking pattern development in GT cells. **Computers and Industrial Engineering** 20: 77-88.

Cheng, C. H., Miltenburg, J., and Motwani, J. 2000. The Effect of Straight- and U-Shaped Lines on Quality. **IEEE Transactions on Engineering Management** 47(3): 321-334.

Chiang, W.-C. and Urban, T. L. 2006. The stochastic U-line balancing problem: A heuristic procedure. **European Journal of Operational Research** 175: 1767-1781.

Chongstitvatana, P., Wattanapornprom, W., Olanviwitchai, P., Sirovetnukul, R., Kampirom, N., and Chutima, P. 2010. Coincidence algorithm for combinatorial optimisation and its applications, **Proceedings of Electrical Engineering Conference (33th)**, Chiangmai, Thailand, December 1-3, invited paper, IP-53-58.

Chutima, P. and Pinkoompee, P. 2008. An investigation of local searches in Memetic Algorithms for multi-objective sequencing problems on mixed-model assembly lines. **Proceedings of Computers and Industrial Engineering**, Beijing, China.

Chutima, P. and Pinkoompee, P. 2009. Multi-objective sequencing problems of mixed-model assembly systems using memetic algorithms. **ScienceAsia** 35: 295-305.

Clegg, S., Ibarra, E., and. Bueno-Rodriquez, L. 1999. **Global management: Universal theories and local realities**. Sage Publications Ltd.

Cochran, J. and Horng, H. 1999. Dynamic dispatching rule-pairs for multitasking workers in JIT production systems. **International Journal of Production Research** 37(10): 2175-2190.

Coello, C. A., Coello, D. A., and Veldhuizen, G. B. 2002. **Evolutionary algorithms for solving multi-objective problems**. Kluwer Academic Publishers.

Copaceanu, C. 2006. Mixed-model assembly line balancing problem: variants and solving techniques. **Proceedings of ICMI 2006**, Bacau, Romania.

Dar-El, E. M. 1973. MALB-A heuristic technique for balancing large single-model assembly lines. **AIIE Transactions** 5(4): 343-356.

Davis, M. M., Aquilano, N. J., and Richard, B. C. 2003. **Fundamentals of operations management**. New York: McGraw-Hill.

Deb, K. 2001. **Multi-objective optimization using evolutionary algorithms**. Chichester: John Wiley & Sons.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. 2002. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. **IEEE Transactions on Evolutionary Computation** 6(2): 182-197.

Deb, K., Sundar, J., Bhaskara, U. R. N., and Chaudhuri, S. 2006. Reference Point Based Multi-Objective Optimization Using Evolutionary Algorithms. **International Journal of Computational Intelligence Research** 2(3): 273-286.

Ebeling, A. and Lee, C. 1994. Cross-training effectiveness and profitability. **International Journal of Production Research** 32(12): 2843-2859.

Erel, E., Sabuncuoglu, I., and Aksu, B. A. 2001. Balancing of U-type assembly systems using simulated annealing. **International Journal of Production Research** 39: 3003-3015.

Erel, E. and Sarin, S. C. 1998. A Survey of the Assembly Line Balancing Procedures. **Production Planning and Control** 9(5): 414-434.

Ertay, T. and Ruan, D. 2005. Data envelopment analysis based decision model for optimal operator allocation in CMS. **European Journal of Operational Research** 164: 800-810.

Falkenauer, E. and Delchambre, A. 1992. A genetic algorithm for bin packing and line balancing. **Proceedings of the 1992 IEEE International Conference on Robotics and Automation**, Nice, France.

Fogel, L. J., Owens, A. J., and Walsch, M. J. 1996. **Artificial Intelligence Through Simulated Evolution**. New York: John Wiley & Sons.

Fonseca, C. M. and Fleming, P. J. 1993. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. **Proceedings of the 5th International Conference on Genetic Algorithm**, University of Illinois at Urbana-Champaign.

Friedberg, R. M. 1958. A learning machine: Part I. **IBM J. Res. Develop**. 2(1): 2-13.

Gen, M. and Cheng, R. 2000. **Genetic Algorithms and Engineering Optimization**. NewYork: John Wiley & Sons.

Gen, M., Cheng, R., and Lin, L. 2008. **Network models and optimization: multiobjective genetic algorithm approach**. London: Springer-Verlag Limited.

Ghinato, P., Fujii, S., and Morita, H. 1997. A BASIC APPROACH TO THE MULTIFUNCTION WORKERS ASSIGNMENT PROBLEM IN U-SHAPED PRODUCTION LINES. **Proceedings of the 3rd International Congress of Industrial Engineering,** Gramado, Brazil, CD-ROM October.

Ghinato, P., Fujii, S., and Morita, H. 1998. THE ALLOCATION OF MULTIFUNCTION WORKERS IN U-SHAPED PRODUCTION LINES: A MULTIOBJECTIVE OPTIMIZATION APPROACH. **Proceedings of the 1998 Pacific Conference on Manufacturing**, Agosto, Brisbane, Australia.

Ghinato, P., Fujii, S., and Morita, H. 1998. A BASIC STUDY ON THE MULTIFUNCTION WORKER ASSIGNMENT PROBLEM IN U-SHAPED PRODUCTION LINES. **Memoirs of the Graduate School of Science & Technology of Kobe University**, Kobe, Marco, Japan.

Ghinato, P., Fujii, S., and Motwani, J. 1998. A HEURISTIC APPROACH TO SOLVE THE MULTIFUNCTION WORKER ASSIGNMENT PROBLEM IN U-SHAPED PRODUCTION LINES. **Proceedings of the 5th International Conference on Automation Technology**, Julho, Taipei, Taiwan, CD ROM.

Ghosh, S. and Gagnon, R. J. 1989. A comprehensive literature review and analysis of the design, balancing and scheduling of assembly systems. **International Journal of Production Research** 27: 637-670.

Gilkinson, J. C., Rabelo, L. C., and Bush, B. O. 1995. A Real-World Scheduling Problem Using Genetic Algorithms. **Proceedings of the 17th International Conference on Computers and Industrial Engineering**.

Goldberg, D. E. 1989. **Genetic Algorithms in Search, Optimization and Machine Learning**. Reading, MA: Addison-Wesley.

Goldberg, D. E. and Lingle, R. 1985. Alleles, loci, and the TSP. **Proceedings of the First International Conference on Genetic Algorithms**.

Guo, Z. X., Wong, W. K., Leung, S. Y. S., Fan, J. T., and Chan, S. F. 2006. Mathematical model and genetic optimization for the job shop scheduling problem in a mixed- and multi-product assembly environment: A case study based on the apparel industry. **Computers & Industrial Engineering** 50: 202-219.

Guo, Z. X., Wong, W. K., Leung, S. Y. S., Fan, J. T., and Chan, S. F. 2006. A Bi-level Genetic Algorithm for Multi-objective Scheduling of Multi- and Mixed-Model Apparel Assembly Lines. **AI 2006: Advances in Artificial Intelligence (Lecture Notes in Computer Science)** 4304: 934-941.

Gupta, M., Gupta, Y., and Kumar, A. 1993. Minimizing flow time variance in single machine system using genetic algorithms. **European Journal of Operational Research** 70: 289-303.

Gutjahr, A. L. and Nemhauser, G. L. 1964. An algorithm for the line balancing problem. **Management Science** 11(2): 308-315.

Hackman, S. T., Magazine, M. J. and Wee, T. S. 1989. Fast, Effective Algorithms for Simple Assembly Line Balancing Problems. **Operations Research** 37(6): 916-924.

Heike, G., Ramulu, M., Sorenson, E., Shanahan, P., and Moinzadeh., K. 2001. Mixed model assembly alternatives for low-volume manufacturing: The case of aerospace industry. **International Journal of Production Economics** 72: 103-120.

Held, M., Karp, R. M., and Shareshian, R. 1963. Assembly line balancing-Dynamic programming with precedence constraints. **Operations Research** 11: 442-459.

Helgeson, W. B., Salveson, M. E., and Smith, W. W. 1954. **How to balance an assembly line**. Technical Report. Carr Press, New Caraan, Conn.

Helgeson, W. P. and Birnie, D. P. 1961. Assembly Line Balancing Using the Ranked Positional Weight Technique. **Journal of Industrial Engineering** 12(6): 394-398.

Hoffmann, T. R. 1963. Assembly Line Balancing with a Precedence Matrix. **Management Science** 9(4): 551-562.

Holland, J. H. 1975. **Adaptation in Natural and Artificial Systems**. Ann Arbor, MI: Univ. of Michigan Press.

Hoogeveen, H. 2005. Multicriteria scheduling. **European Journal of Operational Research** 167(3): 592-623.

Horn, J., Nafpliotis, N., and Goldberg, D. E. 1994. A niched Pareto genetic for multi-objective optimization. **Proceedings of the 1st IEEE International Conference on Evolutionary Computation**, Orlando, FL.

Hwang, R. K. and Katayama, H. 2009. A multi-decision genetic approach for workload balancing of mixed-model U-shaped assembly line systems. **International Journal of Production Research** First published on 31 March 2008.

Hwang, R. K., Katayama, H., and Gen, M. 2008. U-shaped assembly line balancing problem with genetic algorithm. **International Journal of Production Research** 46(16): 4637-4649.

Hyun, C. J., Kim, Y., and Kim, Y. K. 1998. A GENETIC ALGORITHM FOR MULTIPLE OBJECTIVE SEQUENCING PROBLEMS IN MIXED MODEL ASSEMBLY LINES. **Computers & Operations Research** 25(7/8): 675-690.

Ishibuchi, H., Yoshida, T., and Murata, T. 2003. Balance Between Genetic Search and Local Search in Mematic Algorithms for Multiobjective Permutation Flowshop Scheduling. **IEEE Transactions on Evolutionary Computation** 7(2): 204-223.

Jackson, J. R. 1956. A Computing Procedure for a Line Balancing Problem. **Management Science** 2(3): 261-271.

Johnson, R. V. 1981. Assembly Line Balancing Algorithms: Computation Comparisons. **International Journal of Production Research** 19(3): 277-287.

Kannan, V. R. and Jensen, J. B. 2004. Learning and labor assignment in a dual resource constrained cellular shop. **International Journal of Production Research** 42 (7): 1455-1470.

Kara, Y. 2008. Line balancing and model sequencing to reduce work overload in mixed-model U-line production environments. **Engineering Optimization** 40(7): 669-684.

Kara, Y., Ozcan, U., and Peker, A. 2007. An approach for balancing and sequencing mixed-model JIT U-lines. **International Journal of Advanced Manufacturing Technology** 32: 1218-1231.

Kara, Y., Ozcan, U., and Peker, A. 2007. Balancing and sequencing mixed-model just-in-time U-lines with multiple objectives. **Applied Mathematics and Computation** 184: 566-588.

Kara, Y. and Tekin, M. A. 2008. A mixed integer linear programming formulation for optimal balancing of mixed-model U-lines. **International Journal of Production Research**: 1-33, iFirst.

Karp, R. M. 1972. **Reducibility among combinatorial problems**. New York: Plenum Press.

Kelner, V., Capitanescu, F., Leonard, O., and Wehenkel, L. 2008. A hybrid optimization technique coupling an evolutionary and a local search algorithm. **Journal of Computational and Applied Mathematics** 215: 448-456.

Kilbridge, M. D. and Wester, L. 1961. A Heuristic Method of Assembly Line Balancing. **Journal of Industrial Engineering** 12(4): 292-298.

Kilbridge, M. D. and Wester, L. 1962. A review of analytical systems of line balancing. **Operations Research** 10(5): 626-638.

Kim, Y. K., Hyun, C. J., and Kim, Y. 1996. SEQUENCING IN MIXED MODEL ASSEMBLY LINES: A GENETIC ALGORITHM APPROACH. **Computers & Operations Research** 23(12): 1131-1145.

Kim, Y. K., Kim, S. J., and Kim, J. Y. 2000. Balancing and sequencing mixed-model U-lines with a co-evolutionary algorithm. **Production Planning and Control** 11(8): 754-764.

Kim, Y. K., Kim, Y. J., and Kim, Y. 1996. GENETIC ALGORITHMS FOR ASSEMBLY LINE BALANCING WITH VARIOUS OBJECTIVES. **Computers & Industrial Engineering** 30(3): 397-409.

Kim, Y. K., Kim, Y. J., and Kim, Y. 2006. An endosymbiotic evolutionary algorithm for the integration of balancing and sequencing in mixed-model U-lines. **European Journal of Operational Research** 168: 838-852.

Konno, H. and Yamazaki, H. 1992. Mean-Absolute Deviation Portfolio Optimization Model and Its Applications to Tokyo Stock Market. **Management Science** 39: 519-531.

Konak, A., Coit, D. W., and Smith, A. E. 2006. Multi-objective optimization using genetic algorithms: A tutorial. **Reliability Engineering and System Safety** 91: 992-1007.

Koza, J. R. 1992. **Genetic Programming: On the Programming of Computers by Means of Natural Selection**. Cambridge, MA: MIT Press.

Kriengkorakot, N. and Pianthong, N. 2007. The Assembly Line Balancing Problem: Review articles. **KKU Engineering Journal** 34(2): 133-140.

Kumar, R. and Singh, P. K. 2007. Pareto Evolutionary Algorithm Hybridized with Local Search for Biobjective TSP. **Computational Intelligence (SCI)** 75: 361-398.

Kuo, Y. and Yang, T. 2007. Optimization of mixed-skill multi-line operator allocation problem. **Computers and Industrial Engineering** 53: 386-393.

Lalsare, P. and Sen, S. 1995. Evaluating backward scheduling and sequencing rules for an assembly shop environment. **Production and Inventory Management** 36(4): 72-78.

Lee, C. and Vairaktarakis, G. 1997. Workforce planning in mixed model assembly systems. **Operations Research** 45(4): 553-567.

Lenstra, J. K. and Rinnooy Kan, A. H. G. 1981. Complexity of Vehicle Routing and Scheduling Problems. **Networks** 11: 221-227.

Leu, Y. Y., Matheson, L. A., and Rees, L. P. 1994. Assembly line balancing using genetic algorithms with heuristic generated initial populations and multiple criteria. **Decision Sciences** 15: 581-606.

Liu, S. B., Ong, H. L., and Huang, H. C. 2003. Two bi-directional heuristics for the assembly line type II problem. **International Journal of Advanced Manufacturing Technology** 22: 656-661.

Liu, S. B., Ong, H. L., and Huang, H. C. 2005. A bidirectional heuristic for stochastic assembly line balancing Type II problem. **International Journal of Advanced Manufacturing Technology** 25: 71-77.

Loukil, T., Teghem, J., and Tuyttens, D. 2005. Solving multi-objective production scheduling problems using metaheuristics. **European Journal of Operational Research** 161(1): 42-61.

Lu, H. and Yen, G. 2003. Rank-Density-Based Multiobjective Genetic Algorithm and Benchmark Test Function Study. **IEEE Transactions on Evolutionary Computation** 7(4): 325-343.

Luthi, H. and Polymeris, A. 1985. Scheduling to Minimize Maximum Workload. **Management Science** 31(11): 1409-1415.

Mamoud, K. I. 1989. **A Generalised Assembly Line Balancing Algorithm**. Ph.D. Dissertation, University of Bradford, UK.

Mansoor, E. M. 1964. Assembly Line Balancing -An Improvement on the Ranked Positional Weight Technique. **Journal of Industrial Engineering** 15(2): 73-77.

Mansouri, S. A. 2005. A Multi-Objective Genetic Algorithm for mixed-model sequencing on JIT assembly lines. **European Journal of Operational Research** 167: 696-716.

Mantazeri, M. and Van Wassenhove, L. N. 1990. Analysis of scheduling rules for an FMS. **International Journal of Production Research** 28(4): 785-802.

Martinez, U. and Duff, W. S. 2004. HEURISTIC APPROACHES TO SOLVE THE U-SHAPED LINE BALANCING PROBLEM AUGMENTED BY GENETIC ALGORITHMS. **Proceedings of the 2004 Systems and Information Engineering Design Symposium**, Char-lottesville, 16 April 2004.

Mastor, A. A. 1970. An Experimental Investigation and Comparative Evaluation of Production Line Balancing Techniques. **Management Science** 16(11): 728-746.

McMullen, P. R. 1998. JIT sequencing for mixed-model assembly lines with setups using Tabu search. **Production Planning and Control** 9(5): 504-510.

Merengo, C., Nava, F., and Pozzetti, A. 1999. Balancing and sequencing manual mixed-model assembly lines. **International Journal of Production Research** 37(12): 2835-2860.

Michalewicz, Z. 1996. **Genetic Algorithms + Data Structures = Evolution Programs**. New York: Springer.

Miltenburg, J. 1998. Balancing U-lines in a multiple U-line facility. **European Journal of Operational Research** 109: 1-23.

Miltenburg, J. 2001. One-piece flow manufacturing on U-shaped production lines: a tutorial. **IIE Transactions** 33: 303-321.

Miltenburg, J. 2001. U-shaped production lines: A review of theory and practice. **International Journal of Production Economics** 70: 201-214.

Miltenburg, J. 2002. Balancing and scheduling mixed-model U-shaped production lines. **International Journal of Flexible Manufacturing Systems** 14(2): 119-151.

Miltenburg, J. and Sparling, D. 1995. **Optimal solution algorithms for the U-line balancing problem**. Working Paper. McMaster University, Hamilton.

Miltenburg, J. and Wijngaard, J. 1994. The U-line balancing problem. **Management Science** 40(10): 1378-1388.

Miralles, C., Garcia-Sabater, J. P., Andres, C., and Cardos, M. 2008. Branch and bound procedures for solving the Assembly Line Worker Assignment and Balancing Problem: Application to Sheltered Work centres for Disabled. **Discrete Applied Mathematics** 156: 352-367.

Monden, Y. 1993. **Toyota Production System**. Norcross, GA: Engineering and Management Press.

Montazeri, M. and Van Wassenhove, L. N. 1990. Analysis of Scheduling Rules for a FMS. **International Journal of Production Research** 28(4): 785-802.

Nakade, K. and Nishiwaki, R. 2008. Optimal allocation of heterogeneous workers in a U-shaped production line. **Computers & Industrial Engineering** 54: 432-440.

Nakade, K. and Ohno, K. 1995. Reversibility and dependence in a U-shaped production line. **Queueing Systems** 21: 183-197.

Nakade, K. and Ohno, K. 1997. Stochastic Analysis of a U-shaped Production Line with Multiple Workers. **Computers and Industrial Engineering** 3-4: 809-812.

Nakade, K. and Ohno, K. 1999. An optimal worker allocation problem for a U-shaped production line. **International Journal of Production Economics** 60-61: 353-358.

Nakade, K. and Ohno, K. 2003. Separate and carousel type allocations of workers in a U-shaped production line. **European Journal of Operational Research** 145: 403-424.

Nakade, K., Ohno, K. and Shanthikumar, G. 1997. Bounds and approximations for cycle times of a U-shaped production line. **Operations Research Letters** 21: 191-200.

Noorul Haq, A., Jayaprakash, J., and Rengarajan, K. 2006. A hybrid genetic algorithm approach to mixed-model assembly line balancing. **International Journal of Advanced Manufacturing Technology** 28: 337-341.

Nussbaum, M., Sepulveda, M., and Laval, E. 1998. An architecture for solving sequencing and resource allocation problems using approximation methods. **Journal of the Operational Research Society** 49: 52-65.

Ohno, K. and Nakade, K. 1997. ANALYSIS AND OPTIMIZATION OF A U-SHAPED PRODUCTION LINE. **Journal of the Operations Research Society of Japan** 40(1): 90-104.

Olanviwatchai, P. 2009. **APPLICATION OF MEMETIC ALGORITHMS FOR MULTI-OBJECTIVE BALANCING PROBLEM ON MIXED-MODEL U-SHAPED ASSEMBLY LINE IN JIT PRODUCTION SYSTEMS**. Master Thesis, Department of Industrial Engineering, Chulalongkorn University.

Oliver, I. M., Smith, D. J., and Holland, J. R. C. 1987. A study of permutation crossover operators on the traveling salesman problem. **Proceedings of the Second International Conference on Genetic Algorithms**, October, Cambridge, Massachusetts, United States

Ozcan, U. and Toklu, B. 2008. A new hybrid improvement heuristic approach to simple straight and U-type assembly line balancing problems. **Journal of Intelligent Manufacturing** Online First.

Ozmehmet Tasan, S. and Tunali, S. 2007. A review of the current applications of genetic algorithms in assembly line balancing. **Journal of Intelligent Manufacturing**. Available from DOI: 10.1007/s10845-007-0045-5.

Pinedo, M. 2001. **Scheduling: Theory, algorithms and systems**. New York: Prentice Hall.

Pinto, J. and Grossmann, I. E. 1998. Assignment and sequencing models for the scheduling of process systems. **Annals of Operations Research** 81: 433-466.

Poli, R., Kennedy, J., and Blackwell, T. 2007. Particle swarm optimization: an overview. **Swarm Intelligence** 1(1): 33-57.

Ponnambalam, S. G., Aravindan P., and Rao, M. S. 2003. Genetic algorithms for sequencing problems in mixed model assembly lines. **Computers & Industrial Engineering** 45: 669-690.

Prajogo, N. H. and Johnston, R. B. 2001. A Barriers Framework for Understanding Just-In-Time Implementation in Small Manufacturing Enterprises. **Asia Pacific Management Journal** 6(2): 175-195.

Prasad, P. and Maravelias, C. T. 2008. Batch selection, assignment and sequencing in multi-stage multi-product processes. **Computers and Chemical Engineering** 32: 1106-1119.

Rahimi-Vahed, A. R. 2007. A hybrid multi-objective shuffled frog-leaping algorithm for a mixed-model assembly line sequencing problem. **Computers & Industrial Engineering** 53: 642-666.

Raquel, C. R. and Naval, Jr., P.C. 2005. An effective use of crowding distance in multiobjective particle swarm optimization. **Proceedings of the 2005 conference on Genetic and evolutionary computation**, Washington DC, USA, June 25-29.

Rechenberg, I. 1973. **Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution**. Stuttgart, Germany: Frommann-Holzboog.

Rekiek, B. and Delchambre, A. 2006. **Assembly line design: The balancing of mixed-model hybrid assembly lines with genetic algorithms**. Springer Series in Advanced Manufacturing. London.

Rekiek, B., Dolgui, A., Delchambre, A. and Bratcu, A. 2002. State of art of optimization methods for assembly line design. **Annual Reviews in Control** 26: 163-174.

Reyes-Sierra, M. and Coello, C. A. C. 2006. Multi-objective particle swarm optimizers: a survey of the state-of-art. **International Journal of Computational Intelligence Research** 2(3): 287-308.

Robles, V., Miguel, P. D., and Larrañaga, P. 2002. Solving the Traveling Salesman Problem with EDAs. **Estimation of Distribution Algorithm: A New Tool for Evolutionary Computation**.

Salman, A., Ahmad, I., and Al-Madani, S. 2002. Particle swarm optimization for task assignment problem. **Microprocessors and Microsystems** 26: 363-371.

Salveson, M. E. 1955. The Assembly Line Balancing Problem. **Journal of Industrial Engineering** 6(3): 18-25.

Sarin, S. C., Erel, E. and Dar-El, E. M. 1999. A Methodology for Solving Single-Model, Stochastic Assembly Line Balancing Problem. **Omega** 27: 525-535.

Sarker, R., Liang, K.-H., and Newton, C. 2002. A new multiobjective evolutionary algorithm. **European Journal of Operational Research** 140: 12-23.

Sbalzarini, I. F., Muller, S., and Koumoutsakos, P. 2000. Multiobjective optimization using evolutionary algorithms. **Proceedings of the Summer Program 2000**, Center for Turbulence Research, Stanford University.

Scholl, A. 1999. **Balancing and sequencing of assembly lines**. Germany: Physica-Verlag Heidelberg Company.

Scholl, A. and Becker, C. 2006. State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. **European Journal of Operational Research** 168: 666-693.

Scholl, A. and Klein, R. 1999. ULINO: Optimally balancing U-shaped JIT assembly lines. **International Journal of Production Research** 37: 721-736.

Scholl, A., Klein, R., and Domschke, W. 1998. Pattern based vocabulary building for effectively sequencing mixed model assembly lines. **Journal of Heuristics** 4(4): 359-381.

Scholl, A. and Voss, S. 1996. Simple assembly line balancing-heuristic approaches. **Journal of Heuristics** 2: 217-244.

Schrage, L. E. and Baker, K. R. 1978. Dynamic programming solution of sequencing problems with precedence constraints. **Operations Research** 26: 444-449.

Shewchuk, J. P. 2008. Worker allocation in lean U-shaped production lines. **International Journal of Production Research** 46(13): 3485-3502.

Simaria, A. S. and Vilarinho, P. M. 2004. A genetic algorithm based approach to the mixed-model assembly line balancing problem of type II. **Computers and Industrial Engineering** 47: 391-407.

Sirovetnukul, R. and Chutima, P. 2009. Worker allocation in U-shaped assembly lines with multiple objectives. **Proceedings of the 2009 IEEE International Conference on Industrial Engineering and Engineering Management**, Hong Kong, China, December 8-11.

Sirovetnukul, R. and Chutima, P. 2010. Multi-objective particle swarm optimization with negative knowledge for U-shaped assembly line worker allocation problems, **Proceedings of the 2010 IEEE International Conference on Industrial Engineering and Engineering Management**, Macao, China, December 7-10.

Sirovetnukul, R. and Chutima, P. 2010. The Impact of Walking Time on U-shaped Assembly Line Worker Allocation Problems. **Engineering Jounal** 14(2): 53-78.

Solimanpur, M., Vrat, P., and Shankar, R. 2004. A multi-objective genetic algorithm approach to the design of cellular manufacturing systems. **International Journal of Production Research** 7: 1419-1441.

Song, B. L., Wong, W. K., Fan, J. T., and Chan, S. F. 2006. A recursive operator allocation approach for assembly line-balancing optimization problem with the consideration of operator efficiency. **Computers and Industrial Engineering** 51: 585-608.

Sparling, D. and Miltenburg, J. 1998. The mixed-model U-line balancing problem. **International Journal of Production Research** 36: 485-501.

Sprecher, A. 1999. A competitive branch-and-bound algorithm for the simple assembly line balancing problem. **International Journal of Production Research** 37(8): 1787-1816.

Srinivas, N. and Deb, K. 1994. Multiobjective optimization using nondominated sorting in genetic algorithms. **Evolutionary Computation** 2(3): 221-248.

Steiner, G. and Yeomans, S. 1993. Level schedules for mixed-model just-in-time processes. **Management Science** 39(6): 728-735.

Stephen, D. R. and Caros, D. A. 1970. On a multi product assembly line balancing problem. **AIIE Transactions** December.

Suresh, G. and Sahu, S. 1994. Stochastic assembly line balancing using simulated annealing. **International Journal of Production Research** 32(8): 1801-1810.

Taboada, H. A. and Coit, D. W. 2008. MOEA-DAP: A New Multiple Objective Evolutionary Algorithm for Solving Design Allocation Problems. **IEEE Transactions on Reliability** 57(1): 182-191.

Taboada, H. A. and Coit, D. W. 2008. Multiple Objective Scheduling Problems: Determination of Pruned Pareto Sets. **IIE Transactions** 40(5): (in print).

Talbot, F. B., Patterson, J. H., and Gehrlein, W. V. 1986. A comparative evaluation of heuristic line balancing techniques. **Management Science** 32(4): 430-454.

Tambe, P. Y. 2006. **BALANCING MIXED-MODEL ASSEMBLY LINE TO REDUCE WORK OVERLOAD IN A MULTI-LEVEL PRODUCTION SYSTEM**. Master Thesis, Department of Industrial Engineering, Louisiana State University and Agricultural and Mechanical College.

Tavakkoli-Moghaddam, R. and Rahimi-Vahed, A. R. 2006. Multi-criteria sequencing problem for a mixed-model assembly line in a JIT production system. **Applied Mathematics and Computation** 181: 1471-1481.

Thomopoulos, N. T. 1967. Line balancing-sequencing for mixed-model assembly. **Management Science** 14(2): B59-B75.

Thomopoulos, N. T. 1970. Mixed Model Line Balancing with Smoothed Station Assignments. **Management Science** 16(9): 593-603.

Tseng, C. T. and Liao, C. J. 2008. A discrete particle swarm optimization for lot-streaming flowshop scheduling problem. **Europeon Journal of Operational Research** 191: 360-373.

Urban, T. L. 1998. Note. Optimal Balancing of U-shaped Assembly Lines. **Management Science** 44(5): 738-741.

Van Assche, F. and Herroelen, W. S. 1978. An Optimal Procedure for the Single-Model Deterministic Assembly Line Balancing Problem. **European Journal of Operations Research** 3: 142-149.

Vembu, S. and Srinivasan, G. 1995. HEURISTIC FOR OPERATOR ALLOCATION AND SEQUENCING IN JUST-IN-TIME FLOW LINE MANUFACTURING CELL. **Computers and Industrial Engineering** 29(1-4): 309-313.

Vembu, S. and Srinivasan, G. 1997. HEURISTICS FOR OPERATOR ALLOCATION AND SEQUENCING IN PRODUCT-LINE-CELLS WITH MANUALLY OPERATED MACHINES. **Computers & Industrial Engineering** 32(2): 265-279.

Venugopal, V. and Narendran, T. T. 1992. A genetic algorithm approach to the machine-component grouping problem with multiple objectives. **Computers and Industrial Engineering** 22(4): 469-480.

Voudouris, C., Owusu, G., Dorne, R., and Lesaint, D. 2008. **Service Chain Management: Chapter 10 Work Allocation and Scheduling**. Springer Berlin Heidelberg, 139-152.

Wattanapornprom, W., Olanviwitchai, P., Chutima, P., and Chongstitvatana, P. 2009. Multi-objective Combinatorial Optimisation with Coincidence Algorithm. **IEEE Congress on Evolutionary Computation**, Norway, May 18-21.

Yang, C. and Simon, D. 2005. A New Particle Swarm Optimization Technique. **Proceedings of the 18th International Conference on Systems Engineering**, IEEE Computer Society Washington, DC, USA.

Yano, C. and Rachamadugu, R. 1991. Sequencing to minimize work overload in assembly lines with product options. **Management Science** 37(5): 572-586.

Zhang, W. and Gen, M. 2009. An efficient multiobjective genetic algorithm for mixed-model assembly line balancing problem considering demand ratio-based cycle time. **Journal of Intelligent Manufacturing** DOI: 10.1007/s10845-009-0295-5, Online First.

Zitzler, E., Laumanns, M., and Thiele, L. 2001. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. **Technical Report 103, Computer Engineering and Networks Laboratory (TIK)**. Zurich, Switzerland, Swiss Federal Institute of Technology (ETH).

APPENDIX

<u>An example of results of NSGA-II at the side ratio 1:1:1 (1/3)</u>

7 tasks

Merten_7task_cycle7_3:3:1

TS_task_minWS =

6	3	7	4	1	2	5
3	7	4	1	2	5	6
3	6	5	7	4	1	2
7	4	1	2	5	6	3
7	6	3	5	2	4	1

position =

2	2	2	2	1	1	1
2	2	2	1	1	1	1
2	2	2	2	2	1	1
2	2	1	1	1	1	1
2	2	2	2	2	2	1

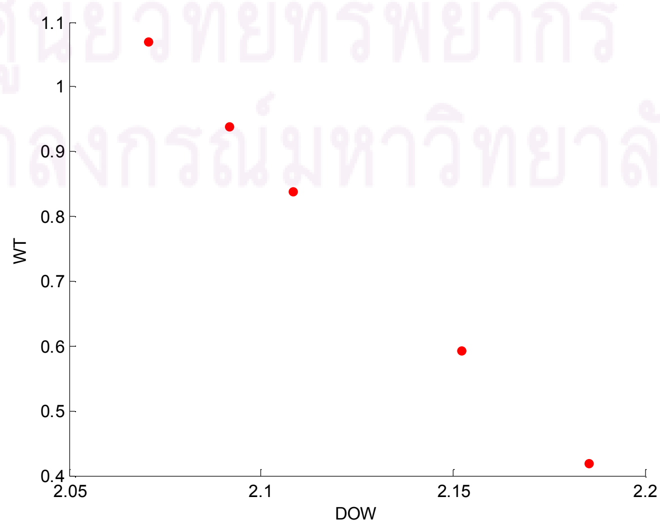
WT_DOW_J =

2.0705	1.0708	6.0000
2.0915	0.9392	6.0000
2.1081	0.8400	6.0000
2.1521	0.5940	6.0000
2.1854	0.4200	6.0000

Define_Station =

1	2	3	4	4	5	6
1	2	3	3	4	5	6
1	2	3	4	5	5	6
1	2	2	3	4	5	6
1	2	3	4	5	6	6

Elapsed time is 1083.630858 seconds.



10 tasks

Miltenburg_10task_cycle10_4:4:2

TS_task_minWS =

5	8	4	9	2	1	3	10	6	7
3	8	1	7	4	10	9	2	5	6
2	6	9	4	10	1	3	8	5	7
6	2	3	4	7	10	5	1	8	9
9	3	1	7	4	8	2	10	5	6
3	9	1	4	10	8	2	5	7	6
2	6	3	9	10	7	8	4	5	1
3	1	4	6	7	10	9	8	2	5
2	6	3	4	7	10	9	8	5	1
10	9	4	2	1	5	8	3	7	6
2	1	7	3	4	5	6	8	9	10
9	10	8	5	2	4	1	3	6	7
3	10	4	5	1	6	2	8	7	9
10	3	4	5	6	9	8	1	2	7
8	9	10	6	7	5	2	1	3	4
2	7	6	5	4	10	9	1	3	8
3	4	8	5	6	1	2	7	10	9
4	3	8	5	6	10	9	1	2	7
5	7	6	10	8	3	9	4	2	1
6	5	7	3	1	4	10	9	8	2
5	3	7	8	1	10	4	6	9	2

position =

1	1	1	2	1	2	1	1	1	1
1	1	1	1	1	2	2	1	1	1
1	1	1	1	2	2	1	1	1	1
1	1	1	1	1	2	1	1	1	1
2	1	1	1	1	1	1	2	1	1
1	2	1	1	2	1	1	1	1	1
1	1	1	1	2	1	1	1	1	1
1	1	1	1	1	2	1	1	1	1
1	1	1	1	1	2	1	1	1	1
2	2	1	1	2	1	1	1	1	1
1	2	1	1	1	1	1	1	1	1
2	2	1	1	1	1	2	1	1	1
1	2	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1
1	2	2	1	1	1	1	2	1	1
1	1	1	1	1	1	1	2	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1

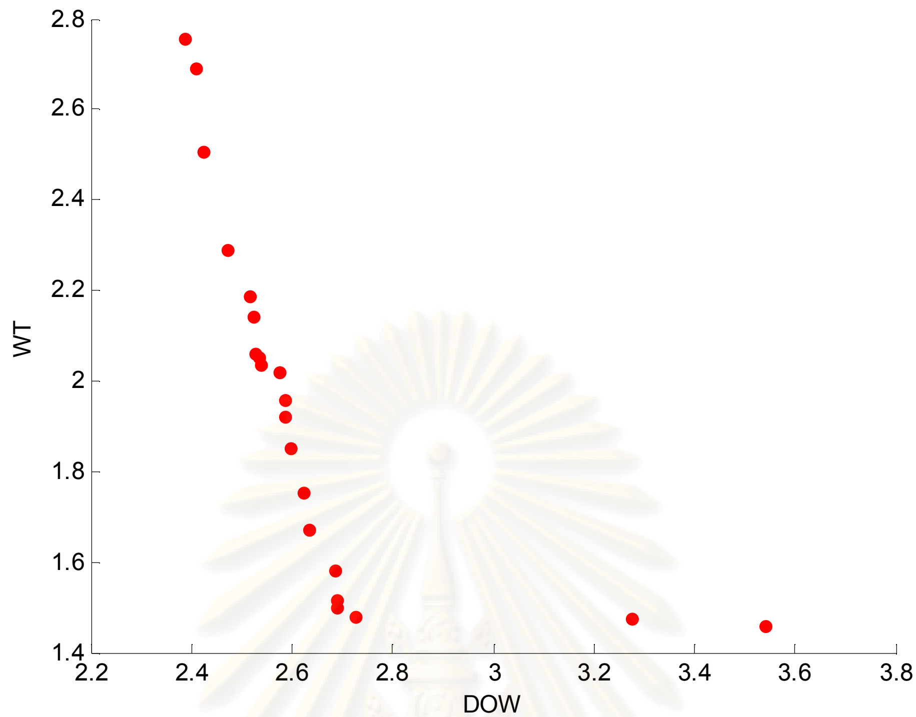
WT_DOW_J =

2.3871	2.7566	4.0000
2.4071	2.6922	4.0000
2.4236	2.5074	4.0000
2.4705	2.2918	4.0000
2.5170	2.1882	4.0000
2.5208	2.1434	4.0000
2.5277	2.0622	4.0000
2.5332	2.0538	4.0000
2.5355	2.0356	4.0000
2.5747	2.0202	4.0000
2.5836	1.9600	4.0000
2.5845	1.9222	4.0000
2.5971	1.8536	4.0000
2.6208	1.7542	4.0000
2.6344	1.6730	4.0000
2.6850	1.5806	4.0000
2.6869	1.5176	4.0000
2.6891	1.4994	4.0000
2.7246	1.4812	4.0000
3.2759	1.4770	4.0000
3.5418	1.4588	4.0000

Define_Station =

1	1	1	2	2	3	3	3	4	4
1	1	1	2	2	2	3	3	4	4
1	1	2	2	2	3	3	3	4	4
1	1	2	2	2	3	3	4	4	4
1	1	1	2	2	2	3	3	4	4
1	1	1	2	2	2	3	3	4	4
1	1	2	2	2	3	3	3	4	4
1	1	1	2	2	3	3	3	4	4
1	1	2	2	2	3	3	3	4	4
1	1	1	2	2	3	3	3	4	4
1	1	2	2	2	3	3	4	4	4
1	1	1	2	2	3	3	3	4	4
1	1	1	2	2	3	3	4	4	4
1	1	1	2	2	3	3	3	4	4
1	1	1	2	2	3	3	4	4	4
1	1	2	2	3	3	3	4	4	4
1	1	1	2	2	3	3	4	4	4
1	1	1	2	2	3	3	3	4	4
1	1	2	2	2	3	3	3	4	4
1	1	2	2	2	3	3	3	3	4
1	1	1	2	2	2	2	3	3	4

Elapsed time is 2711.902717 seconds.



An example of results of NSGA-II at the side ratio 1:4:4 (1/9)

11 tasks

Jackson_11task_cycle13_4:4:3

TS_task_minWS =

11	9	7	4	10	8	6	1	3	5	2
1	11	10	9	2	8	7	5	3	4	6
1	3	2	4	11	10	8	6	9	7	5
1	2	5	11	10	8	6	3	9	7	4
1	2	5	11	9	7	3	10	8	6	4
11	9	7	4	3	10	8	6	5	2	1
11	1	3	10	2	8	6	5	9	7	4
1	3	11	10	8	2	6	4	9	7	5

position =

2	2	2	2	2	2	2	1	1	1	1
1	2	2	2	1	2	2	1	1	1	1
1	1	1	1	2	2	2	1	2	2	1
1	1	1	2	2	2	1	1	2	2	1
1	1	1	2	2	2	1	2	2	1	1
2	2	2	2	2	2	2	2	2	2	1
2	1	1	2	1	2	1	1	2	2	1
1	1	2	2	2	1	1	1	2	2	1

WT_DOW_J =

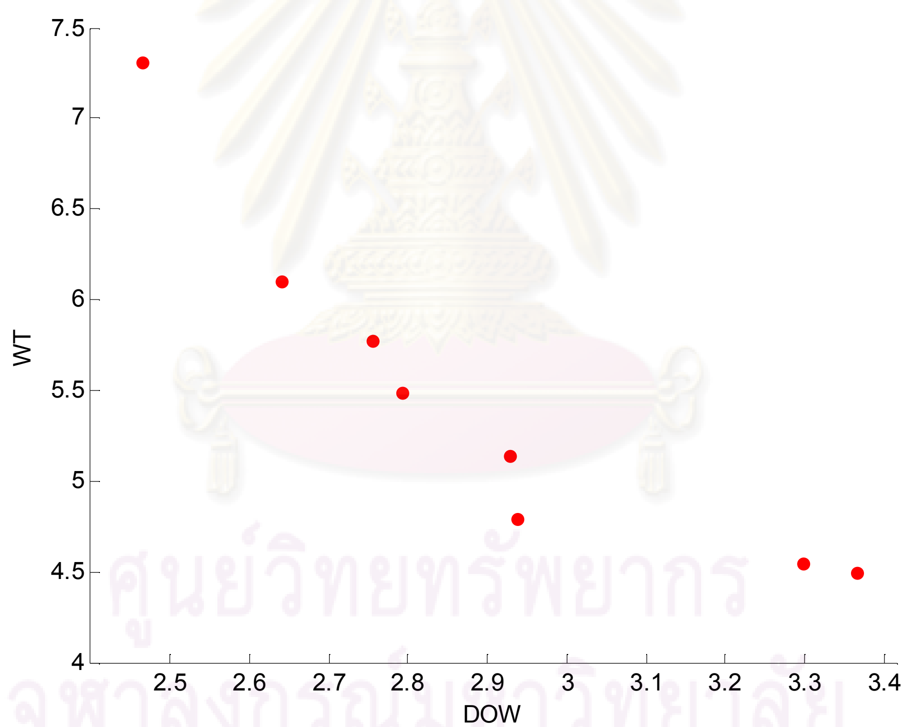
2.4633	7.3140	5.0000
2.6407	6.1054	5.0000

2.7539	5.7794	5.0000
2.7919	5.4898	5.0000
2.9273	5.1419	5.0000
2.9369	4.7940	5.0000
3.2975	4.5479	5.0000
3.3658	4.4950	5.0000

Define_Station =

1	1	2	2	3	3	4	4	5	5	5
1	1	2	2	3	3	4	4	4	5	5
1	1	2	2	3	3	4	4	5	5	5
1	1	1	2	2	3	3	4	4	5	5
1	1	1	2	2	3	3	4	4	5	5
1	1	2	2	3	3	4	4	4	5	5
1	1	2	2	3	3	3	4	4	4	5
1	1	2	2	3	3	3	4	5	5	5

Elapsed time is 4081.092483 seconds.



19 tasks

Thomopoulos_19task_cycle120_8:8:3

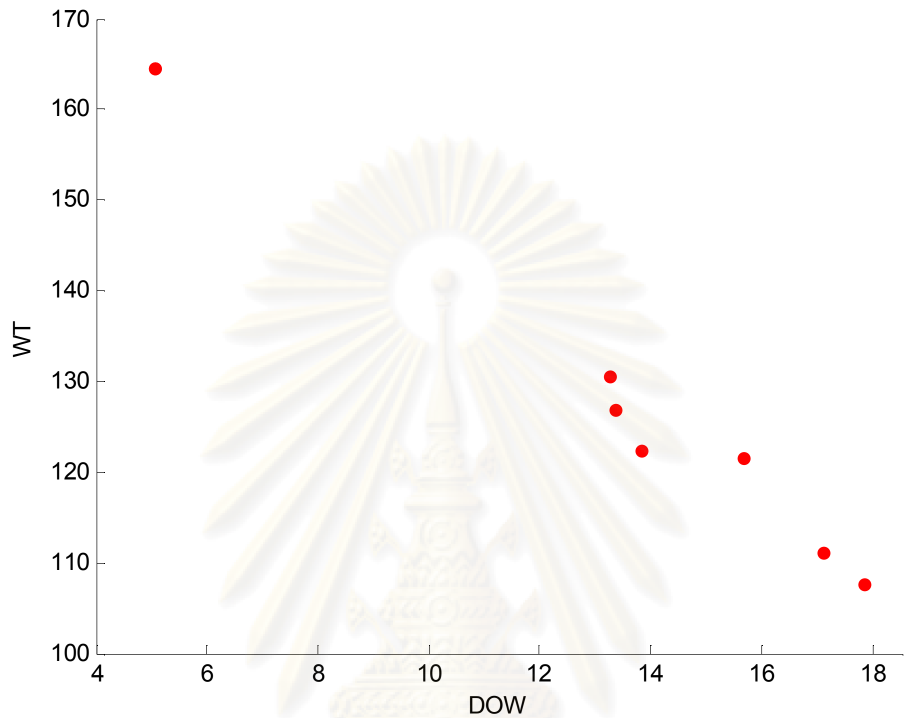
TS_task_minWS =

Columns 1 through 17

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

```
  10   16   19   17   18    2    8    6   14   15   12    9    7    4    1   13   11
   3    6    4    5   11   14    2    9    8   10   18    1    7   13   17   12   15
   6   19   15    3    5    4   11   14   13    2    8   17    9   18   16   12    7
   3    2    9    8    1    5    4   11   13    7   14   10   17    6   16   18   12
 Columns 18 through 19
  17   13
   1    3
  12   17
   5    3
  19   16
   1   10
  15   19
position =
 Columns 1 through 17
   2    2    2    2    2    2    2    2    1    1    1    1    1    1    1    1    2
   2    2    2    2    2    1    1    2    2    2    2    2    2    1    2    2    1
   2    2    2    2    2    2    1    1    1    1    1    1    1    1    2    1    1
   2    2    2    2    2    1    1    2    2    2    2    2    1    1    2    2    2
   1    2    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
   2    2    2    1    1    1    1    1    1    1    1    1    1    1    1    2    2
   1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
 Columns 18 through 19
   2    1
   1    1
   1    1
   1    1
   1    1
   1    1
   1    1
WT_DOW_J =
   5.0597  164.5488    5.0000
  13.2397  130.5600    5.0000
  13.3435  126.9663    5.0000
  13.8376  122.4000    5.0000
  15.6794  121.5224    5.0000
  17.1088  111.2069    5.0000
  17.8277  107.7699    5.0000
Define_Station =
 Columns 1 through 17
   1    1    2    2    2    2    2    3    3    4    4    4    4    5    5    5    5
   1    1    1    1    1    2    2    3    3    3    4    4    4    5    5    5    5
   1    1    1    1    1    1    2    2    2    3    3    3    3    3    4    4    5
   1    1    1    1    1    2    2    3    3    3    4    4    4    5    5    5    5
   1    1    1    1    1    2    2    3    3    3    3    3    4    4    4    4    5
   1    1    1    2    2    2    2    2    2    3    3    4    4    4    4    5    5
   1    1    2    2    2    2    2    3    3    3    3    3    4    4    4    4    4
 Columns 18 through 19
   5    5
   5    5
   5    5
```

5 5
5 5
5 5
5 5
Elapsed time is 5101.182266 seconds.



An example of results of MA at the side ratio 1:1:1 (1/3)

28 tasks

Heskiaoff_28task_cycle256_9:9:10

TS_task_minWS =
Columns 1 through 18
1 3 26 22 28 17 20 18 14 16 8 9 4 7 6 21 15 11
28 22 20 2 6 7 27 26 4 15 11 18 3 21 14 19 17 5
1 21 3 22 28 20 18 16 14 26 8 9 4 7 6 17 15 11
2 28 18 20 19 15 22 1 26 8 9 21 10 5 27 14 7 16
28 5 15 4 18 27 26 2 21 25 11 3 20 19 17 14 6
16
2 6 7 1 19 20 26 27 22 21 28 15 5 16 17 11 3 18
28 15 11 5 18 7 2 25 6 4 27 26 24 16 17 20 19
23
28 17 20 19 18 7 16 25 24 2 14 6 4 15 1 8 9 10
28 14 21 2 6 17 4 27 26 3 18 25 20 19 16 13 12
7
1 23 21 2 6 7 18 4 19 20 28 15 27 14 17 3 16 13
28 4 21 18 14 27 15 3 16 13 12 1 24 8 9 10 19 7

1 21 3 22 28 20 18 17 14 26 8 9 4 7 6 16 15 11
28 14 20 18 7 27 2 25 6 4 26 19 24 17 16 5 13
12
28 14 20 18 7 2 25 6 4 27 26 19 24 17 16 5 13
12
28 14 21 2 6 7 17 4 27 26 3 20 19 25 18 16 13
12
28 14 20 18 7 2 25 6 4 27 26 19 24 16 17 5 13
12
1 21 19 20 24 25 4 28 14 15 11 18 17 16 7 6 27
13
28 17 3 16 2 6 7 4 5 27 26 1 19 22 20 18 15 11
28 5 21 3 27 17 2 14 26 4 20 19 25 24 18 7 16
13
28 20 18 19 3 2 4 5 27 26 1 6 15 11 22 7 8 9
2 17 6 7 18 28 21 22 20 19 4 14 3 15 11 1 8 26
1 21 19 20 4 3 8 5 28 15 16 18 7 17 25 27 14 13
2 17 6 7 18 28 21 22 20 19 3 14 4 15 11 1 8 26
2 17 6 7 18 28 21 4 23 15 27 16 3 14 13 12 1 8
2 17 1 19 20 26 27 8 22 21 3 5 28 14 6 18 16 13
2 17 6 7 18 28 15 14 11 20 19 16 3 4 1 24 26 13
2 17 6 7 18 28 15 11 20 19 14 16 3 27 4 1 24 13
2 6 7 18 17 28 22 23 14 27 16 5 15 11 3 13 12 1
2 17 1 19 20 26 27 8 22 21 5 28 14 6 7 18 16 13
2 17 1 19 20 26 27 8 22 21 5 28 14 6 7 18 16 13
28 18 15 4 14 5 27 26 21 7 16 13 1 8 24 9 10 19
2 6 17 7 18 1 8 26 23 4 22 3 27 24 25 5 21 28
2 1 26 23 8 22 17 5 27 24 25 6 7 3 21 18 4 28
2 17 1 19 26 27 8 21 20 5 28 25 14 6 7 18 16 13
2 17 6 7 18 28 16 27 26 25 24 1 19 15 8 9 10 12
2 6 7 18 17 28 15 21 22 16 25 3 4 14 13 12 1 8
2 6 17 7 18 28 16 23 25 24 27 1 19 8 15 9 10 12
1 24 4 21 23 5 8 9 2 6 7 17 18 10 11 15 25 22
2 17 6 7 18 1 5 24 25 4 23 19 21 3 26 27 28 14
2 17 6 7 18 28 16 27 26 25 24 1 19 8 9 15 10 12
2 17 6 7 18 1 21 5 19 26 27 23 3 4 24 25 22 28
2 17 6 7 18 28 20 16 21 19 5 3 14 13 12 1 26 15
2 17 6 7 18 28 20 16 21 14 19 3 4 13 12 1 26 15
1 8 19 3 20 4 5 9 10 11 15 12 13 14 22 26 27 16
2 17 6 7 18 28 20 16 21 19 3 14 13 12 1 26 15
11
2 6 7 18 17 28 22 20 16 4 23 27 3 19 15 14 13 1
1 8 3 4 5 21 24 19 20 9 10 12 13 14 16 11 15 22
2 17 6 7 18 28 20 16 21 19 3 27 14 13 12 1 15
11
28 3 27 17 15 14 16 13 18 11 5 22 1 19 8 9 10 7
2 6 7 18 17 1 26 8 9 4 24 21 3 27 22 5 19 23
2 17 6 7 18 1 19 4 8 21 23 26 9 10 5 12 27 13
2 17 6 7 18 1 19 4 8 21 23 26 5 9 27 10 12 13
2 6 7 18 17 1 19 4 21 8 23 26 27 9 10 12 13 14

2 6 17 7 18 1 24 4 25 19 20 5 26 8 9 10 11 12
2 6 7 17 18 1 4 5 26 24 22 8 9 10 27 23 11 15
2 17 6 7 18 1 24 4 25 5 26 19 20 8 9 10 12 11
2 17 6 7 18 1 8 5 4 26 27 24 9 10 11 15 12 13
2 17 6 7 18 1 8 9 4 26 27 24 10 11 12 19 20 13
2 6 7 18 17 28 14 3 4 23 22 5 15 16 13 27 26 12
1 8 4 26 23 27 21 9 10 11 15 5 12 13 14 16 3 19
2 17 6 7 18 1 8 5 4 26 27 24 9 10 12 13 11 15
2 17 6 7 18 1 22 4 8 9 23 24 10 11 12 15 13 14
2 17 6 7 18 1 8 5 19 26 27 24 3 9 10 12 20 13
1 8 22 4 23 24 9 10 11 15 5 12 13 14 16 3 19 20
1 24 4 5 8 23 9 10 12 11 15 13 2 6 7 17 18 14
1 24 4 5 2 6 7 8 22 9 10 11 15 12 13 18 17 14
2 17 6 7 18 1 22 8 9 23 24 4 10 11 12 15 5 13
2 17 6 7 18 1 21 4 8 9 22 23 26 10 11 12 15 13
2 17 6 7 18 28 15 25 11 20 24 23 4 16 27 22 21
5
2 17 6 7 18 1 22 4 8 9 23 24 10 11 12 15 13 14
2 6 7 17 1 8 26 24 4 9 27 22 10 12 13 11 15 14
2 6 7 17 28 3 25 15 14 24 4 5 20 21 11 16 27 13
2 17 6 7 1 8 26 23 4 27 24 9 10 11 12 13 14 16
2 17 6 7 28 3 25 14 24 4 5 23 20 16 15 11 27 13
2 6 7 17 1 19 5 22 24 26 8 27 9 10 12 13 14 11
1 8 3 19 20 4 5 21 24 26 27 22 9 10 11 15 12 13
1 8 3 19 20 21 4 5 22 24 26 27 23 9 10 12 13 14
2 6 17 28 14 25 21 24 23 15 11 5 16 13 27 18 22
26
2 17 6 28 3 25 20 5 24 4 14 18 27 26 22 23 21
15
2 17 6 28 25 20 5 18 4 14 3 22 27 26 24 21 23
15
2 17 28 14 25 21 24 23 15 5 4 16 13 27 18 7 22
26

Columns 19 through 28

2 24 13 12 25 5 10 19 27 23
16 13 12 25 24 10 9 8 1 23
2 24 13 12 25 5 10 19 27 23
13 12 3 6 11 24 17 23 4 25
13 12 10 9 8 7 24 23 1 22
14 13 12 10 9 24 8 23 4 25
14 13 12 10 9 8 22 3 1 21
12 13 22 23 5 27 3 21 26 11
5 23 15 24 11 10 9 8 1 22
12 8 9 10 24 26 25 5 22 11
26 11 6 2 20 25 17 23 22 5
2 24 13 12 25 5 10 19 27 23
22 15 11 10 9 8 23 3 1 21
22 15 11 10 9 8 23 3 1 21
5 23 15 24 11 10 9 8 1 22
22 15 11 10 9 8 23 3 1 21

12	23	3	8	9	2	5	22	26	10
24	14	13	12	10	9	23	21	8	25
12	6	23	15	11	10	9	8	1	22
10	12	13	14	24	16	23	17	21	25
27	16	13	12	10	23	5	25	24	9
12	23	24	9	10	2	6	22	26	11
27	16	13	12	10	23	5	25	24	9
9	24	20	11	19	25	26	5	22	10
12	7	15	11	10	24	9	23	4	25
12	22	23	8	9	27	5	21	25	10
12	22	23	8	9	26	5	21	25	10
8	24	20	10	19	26	25	4	21	9
3	12	15	11	10	24	9	23	4	25
12	3	15	11	10	24	9	23	4	25
20	25	3	12	6	2	22	11	23	17
14	15	11	20	19	16	13	12	10	9
14	20	15	11	19	16	13	12	10	9
12	3	15	11	10	23	9	22	24	4
13	14	5	4	20	11	23	3	22	21
9	19	20	11	24	27	26	5	23	10
13	14	5	4	20	11	21	3	26	22
12	13	16	26	28	14	20	27	3	19
15	16	13	12	11	10	9	22	8	20
13	14	5	4	20	11	22	3	21	23
14	15	16	13	12	11	10	9	8	20
11	27	24	8	9	23	4	22	25	10
11	27	24	8	9	23	5	22	25	10
24	25	21	28	17	18	7	23	2	6
10	9	27	25	8	23	5	22	24	4
8	9	21	12	25	11	26	5	24	10
25	23	28	27	18	17	7	26	2	6
10	9	26	25	8	23	5	22	24	4
25	12	6	2	21	26	20	24	23	4
28	15	16	14	11	13	12	20	25	10
14	16	11	15	3	28	25	22	20	24
14	16	11	15	3	28	25	22	20	24
5	16	11	15	3	28	25	22	20	24
13	16	14	21	3	23	28	27	22	15
12	13	19	21	14	28	16	20	25	3
15	13	14	22	3	23	28	27	16	21
19	20	14	16	3	25	28	23	22	21
14	16	5	15	3	25	28	23	22	21
11	10	9	20	19	8	25	24	1	21
20	2	6	17	7	18	28	25	24	22
19	20	14	16	3	25	23	28	22	21
16	25	5	19	20	3	28	27	26	21
14	16	11	15	4	25	23	28	22	21
2	17	6	7	18	28	27	26	25	21
25	22	19	28	27	16	21	26	3	20
25	23	19	28	27	16	21	26	3	20

14	16	3	19	20	28	27	26	25	21
14	19	5	27	20	3	28	25	24	16
26	14	13	12	10	9	8	3	1	19
16	5	19	20	3	28	27	26	25	21
16	3	19	20	5	28	21	23	25	18
12	10	9	8	19	23	22	26	1	18
15	3	19	20	5	28	22	21	25	18
12	10	9	8	19	22	21	26	1	18
15	16	21	3	20	4	28	23	18	25
14	16	23	28	18	17	7	25	2	6
16	11	15	28	18	17	7	25	2	6
12	3	10	9	8	4	20	19	1	7
11	16	13	12	10	9	8	19	1	7
11	16	13	12	10	9	8	19	1	7
11	3	12	10	9	8	20	19	1	6

position =

Columns 1 through 18

1	1	1	1	2	2	2	2	2	2	1	1	1	2	2	1	2	2
2	2	2	1	1	1	2	2	2	2	2	1	2	2	2	2	1	2
1	1	1	1	2	2	2	2	2	1	1	1	1	2	2	2	2	2
1	2	2	2	2	2	2	1	1	1	1	1	1	1	1	2	2	2
2	2	2	2	2	2	2	1	2	2	2	2	2	2	1	2	1	2
1	1	1	1	1	1	1	1	1	1	2	2	1	2	1	2	1	1
2	2	2	2	2	2	1	2	1	2	2	2	2	2	1	2	2	2
2	2	2	2	2	2	2	2	2	1	2	1	2	2	1	1	1	1
2	2	2	1	1	1	2	2	2	2	2	2	2	2	2	2	2	1
1	1	1	1	1	1	1	1	1	1	2	2	2	2	1	1	2	2
2	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	2
1	1	1	1	2	2	2	2	2	1	1	1	1	2	2	2	2	2
2	2	2	2	2	2	1	2	1	2	2	2	2	1	2	2	2	2
2	2	2	2	2	1	2	1	2	2	2	2	2	1	2	2	2	2
2	2	2	1	1	1	1	2	2	2	2	2	2	2	1	2	2	2
2	2	2	2	2	1	2	1	2	2	2	2	2	2	1	2	2	2
1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	2
2	2	2	2	1	1	1	2	2	2	2	1	1	1	1	1	2	2
2	2	2	2	2	2	1	2	2	2	2	2	2	2	2	2	2	2
2	2	2	2	2	1	2	2	2	2	1	1	2	2	1	1	1	1
1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	1	1	1
1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2
1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	1	1	1
1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	2	1	1
1	1	1	1	1	1	1	1	1	1	1	1	2	2	1	2	2	2
1	1	1	1	1	2	2	2	2	2	2	2	2	2	1	1	1	2
1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	1	1	2
1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	2	2	1
1	1	1	1	1	1	1	1	1	1	1	2	2	1	1	1	2	2
1	1	1	1	1	1	1	1	1	1	1	2	2	1	1	1	2	2
2	2	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2

1	1	1	1	1	1	1	1	1	1	2	2	2	1	1	1	2	2
1	1	1	1	1	2	2	2	2	2	2	1	1	2	1	1	1	1
1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	2	1	1
1	1	1	1	1	2	2	2	2	2	2	1	1	1	2	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2
1	1	1	1	1	2	2	2	2	2	2	1	1	1	1	2	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	1	1	2
1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	1	1	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	2	2	2	2	2	2	2	2	2	1	1	2	2
1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	2	2	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	1	2	2
2	2	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
1	1	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
1	1	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
1	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2

Columns 19 through 28

1	1	2	2	1	1	1	1	1	1

2	2	2	2	2	2	2	2	1	1
1	1	2	2	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1
2	2	2	2	2	1	2	2	1	1
2	2	2	2	2	1	1	1	1	1
2	2	2	2	2	2	2	2	1	1
1	1	1	1	1	2	1	1	1	1
2	2	2	2	2	2	2	2	1	1
2	1	1	1	1	1	1	1	1	1
1	1	2	1	1	1	1	1	1	1
1	1	2	2	1	1	1	1	1	1
2	2	2	2	2	2	2	2	1	1
2	2	2	2	2	2	2	2	1	1
2	2	2	2	2	2	2	2	1	1
2	2	2	2	2	2	2	2	1	1
2	1	1	1	1	1	1	1	1	1
1	2	2	2	2	2	1	1	1	1
2	1	2	2	2	2	2	2	1	1
1	1	1	1	1	1	1	1	1	1
1	2	2	2	2	1	1	2	1	1
2	1	1	1	1	1	1	1	1	1
1	2	2	2	2	1	1	2	1	1
1	1	2	2	1	1	1	1	1	1
2	1	2	2	2	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1
1	1	2	2	1	1	1	1	1	1
1	2	2	2	2	1	1	1	1	1
2	1	2	2	2	1	1	1	1	1
1	1	1	1	2	1	1	1	1	1
2	2	2	2	1	2	2	2	2	1
2	2	2	2	1	2	2	2	2	1
2	1	2	2	2	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	2	1	2	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	2	1	2	1	1	1
2	2	2	2	2	2	2	1	1	1
1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	1	1
2	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1
1	1	1	2	2	2	2	1	1	1
2	2	1	2	1	1	1	1	1	1
1	1	1	2	2	2	1	1	1	1
1	1	2	2	2	2	2	1	1	1
2	2	1	2	1	1	1	1	1	1
2	1	2	1	1	1	1	1	1	1
2	2	2	2	2	2	2	1	1	1
1	1	1	1	1	2	2	1	1	1

1	1	1	1	1	2	2	1	1	1
1	1	1	1	1	2	2	1	1	1
1	1	1	1	1	1	2	1	1	1
1	1	1	1	1	2	1	1	1	1
1	1	1	1	1	1	2	1	1	1
1	1	1	1	1	1	2	1	1	1
1	1	1	1	1	1	2	1	1	1
2	2	2	2	2	2	2	2	1	1
1	1	1	1	1	1	2	2	1	1
1	1	1	1	1	1	1	2	1	1
1	1	1	1	1	1	2	2	1	1
1	1	1	1	1	1	1	2	1	1
1	1	1	1	1	2	2	1	1	1
1	1	1	2	2	1	1	1	1	1
1	1	1	2	2	1	1	1	1	1
1	1	1	1	1	2	2	1	1	1
1	1	1	1	1	1	2	2	1	1
2	2	2	2	2	2	2	2	1	1
1	1	1	1	1	2	2	1	1	1
1	1	1	1	1	2	1	1	1	1
2	2	2	2	2	2	2	2	1	1
1	1	1	1	1	2	1	1	1	1
2	2	2	2	2	2	2	2	1	1
1	1	1	1	1	1	2	1	1	1
1	1	1	2	2	2	2	1	1	1
1	1	1	2	2	2	2	1	1	1
2	2	2	2	2	2	2	2	1	1
2	2	2	2	2	2	2	2	1	1
2	2	2	2	2	2	2	2	1	1
2	2	2	2	2	2	2	2	1	1

WT_DOW_J =

14.7182	194.8597	5.0000
14.8385	194.4997	5.0000
15.7909	183.2979	5.0000
15.9518	182.8840	5.0000
16.3734	177.8200	5.0000
17.8765	170.3505	5.0000
19.2544	167.1947	5.0000
19.6360	166.4034	5.0000
19.9701	161.1331	5.0000
20.4002	160.8954	5.0000
20.7378	160.2551	5.0000
20.9583	153.8682	5.0000
21.2519	150.0278	5.0000
21.8997	147.4689	5.0000
22.2930	146.9201	5.0000
22.3525	144.7091	5.0000
23.2949	141.2720	5.0000
24.5858	139.5163	5.0000
24.6069	138.9047	5.0000

24.9859	137.9154	5.0000
25.2287	136.0790	5.0000
25.3734	135.1393	5.0000
25.3884	134.1577	5.0000
25.4873	131.3892	5.0000
25.5183	130.4530	5.0000
25.6799	129.6740	5.0000
26.3765	126.7033	5.0000
26.8226	123.7772	5.0000
27.0896	123.5517	5.0000
27.7922	120.9798	5.0000
28.2871	117.6079	5.0000
28.7648	114.7293	5.0000
29.0699	114.6061	5.0000
29.2400	114.3862	5.0000
29.4425	111.0525	5.0000
29.4740	110.9394	5.0000
29.8027	109.3138	5.0000
30.2011	108.4140	5.0000
30.4405	106.4581	5.0000
30.4548	106.3358	5.0000
30.4819	105.1168	5.0000
30.5017	104.3241	5.0000
30.5261	104.2985	5.0000
30.8335	102.2145	5.0000
30.8823	101.9749	5.0000
31.5092	99.1095	5.0000
31.9410	98.5435	5.0000
32.0117	96.8564	5.0000
32.1063	95.6872	5.0000
32.9581	95.3800	5.0000
32.9928	93.7511	5.0000
33.0542	93.1078	5.0000
33.3799	91.4912	5.0000
33.5021	90.1140	5.0000
33.7552	89.4625	5.0000
33.9026	87.3964	5.0000
34.2560	85.8259	5.0000
34.5352	85.1376	5.0000
34.5362	84.1800	5.0000
34.6927	82.7797	5.0000
34.8675	82.1659	5.0000
35.0608	81.6745	5.0000
35.1123	81.4776	5.0000
35.2851	80.0764	5.0000
35.4329	79.5153	5.0000
35.4466	79.2004	5.0000
35.5186	79.1491	5.0000
35.5303	78.4916	5.0000
35.6324	78.3485	5.0000

```
   35.7458   77.8176    5.0000
   36.0241   77.5369    5.0000
   36.2697   76.7191    5.0000
   36.7532   76.6097    5.0000
   36.8196   76.3866    5.0000
   38.1157   76.3289    5.0000
   39.0040   75.9845    5.0000
   39.2948   75.6310    5.0000
   40.4606   75.0305    5.0000
   41.7282   74.7453    5.0000
   41.9537   74.7299    5.0000
   47.6980   74.0499    5.0000
Define_Station =
  Columns 1 through 18
     1     1     1     1     1     2     2     2     2     2     3     3     3     3     3     3     3     4
     1     1     1     1     2     2     2     2     2     2     2     2     2     2     3     3     3     3
     1     1     1     1     1     2     2     2     2     2     2     3     3     3     3     3     3     3
     1     1     1     1     2     2     2     2     2     2     3     3     3     3     3     3     3     3
     1     1     1     1     1     1     1     1     1     2     2     2     2     3     3     3     4     4
     1     1     1     1     1     2     2     2     2     2     2     2     2     3     3     3     3     3
     1     1     1     1     1     1     1     2     2     2     2     2     2     2     3     3     3     3
     1     1     1     2     2     2     2     2     2     3     3     3     3     3     4     4     4     4
     1     1     1     1     2     2     2     2     2     2     2     3     3     3     3     4     4     4
     1     1     1     1     1     1     1     1     2     2     2     2     2     3     3     3     3     4
     1     1     1     1     1     1     1     1     1     2     2     2     3     3     3     3     3     4
     1     1     1     1     1     2     2     2     2     3     3     3     3     3     3     3     3     3
     1     1     1     1     1     1     2     2     2     3     3     3     3     3     3     3     4     4
     1     1     1     1     1     2     2     2     3     3     3     3     3     3     3     3     4     4
     1     1     1     1     2     2     2     2     2     2     2     3     3     3     4     4     4     4
     1     1     1     1     1     2     2     2     3     3     3     3     3     3     3     3     4     4
     1     1     1     1     1     2     2     2     3     3     3     3     3     3     3     4     4     4
     1     1     1     1     2     2     2     2     2     2     2     2     3     3     3     3     3     3
     1     1     1     1     1     1     2     2     2     2     2     3     3     3     3     3     3     4
     1     1     1     1     1     2     2     2     2     2     2     2     3     3     3     3     3     3
     1     1     1     1     1     2     2     2     2     2     2     3     3     3     3     3     4     4
     1     1     1     1     1     2     2     2     2     2     2     2     3     3     3     3     4     4
     1     1     1     1     1     2     2     2     2     2     3     3     3     3     3     3     4     4
     1     1     1     1     1     2     2     2     2     2     2     2     2     2     3     3     3     4
     1     1     1     2     2     2     2     2     2     2     3     3     3     3     3     4     4     4
     1     1     1     1     1     2     2     2     2     2     3     3     3     3     3     3     3     4
     1     1     1     1     1     2     2     2     2     2     3     3     3     3     3     3     3     4
     1     1     1     1     1     2     2     2     2     2     2     2     2     2     2     3     3     3
     1     1     1     2     2     2     2     2     2     2     2     3     3     3     3     3     3     4
     1     1     1     2     2     2     2     2     2     2     2     3     3     3     3     3     3     4
     1     1     1     1     1     1     1     1     1     1     1     2     2     3     3     3     3     3
     1     1     1     1     1     2     2     2     2     2     2     2     2     2     3     3     3     3
     1     1     1     1     1     1     2     2     2     2     2     3     3     3     3     3     3     3
     1     1     1     2     2     2     2     2     2     2     3     3     3     4     4     4     4     4
     1     1     1     1     1     2     2     2     2     2     2     3     3     3     3     4     4     4
     1     1     1     1     1     2     2     2     2     2     2     3     3     3     3     3     4     4
```

1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	4	4	4
1	1	1	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3
1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3	3	4
1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	4	4	4
1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	3	3	3
1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	4	4	4
1	1	1	1	1	2	2	2	2	2	3	3	3	3	3	4	4	4
1	1	1	1	2	2	2	2	2	2	2	2	3	3	3	3	3	3
1	1	1	1	1	2	2	2	2	2	3	3	3	3	4	4	4	4
1	1	1	1	1	2	2	2	2	2	2	2	2	3	3	3	3	4
1	1	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3	3
1	1	1	1	1	2	2	2	2	2	3	3	3	3	3	4	4	4
1	1	1	1	1	2	2	2	2	2	2	2	3	3	3	3	4	4
1	1	1	1	1	2	2	2	2	2	2	2	3	3	3	3	3	3
1	1	1	1	1	2	2	2	2	2	2	2	3	3	3	3	3	3
1	1	1	1	1	2	2	2	2	2	2	2	2	3	3	3	3	3
1	1	1	1	1	2	2	2	2	2	2	2	2	3	3	3	3	4
1	1	1	1	1	2	2	2	2	3	3	3	3	3	3	4	4	4
1	1	1	1	1	2	2	2	2	2	2	2	2	3	3	3	3	3
1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	4	4	4
1	1	1	1	1	2	2	2	2	2	2	2	2	3	3	3	3	3
1	1	1	1	1	2	2	2	2	2	2	2	3	3	3	3	3	4
1	1	1	1	1	2	2	2	2	2	2	2	2	2	3	3	3	3
1	1	1	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3
1	1	1	1	1	2	2	2	2	2	2	2	2	3	3	3	3	3
1	1	1	1	1	2	2	2	2	2	2	2	3	3	3	3	3	4
1	1	1	1	1	2	2	2	2	2	2	2	3	3	3	3	3	4
1	1	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3
1	1	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3	4
1	1	1	1	1	1	1	2	2	2	2	2	2	2	3	3	3	4
1	1	1	1	1	2	2	2	2	2	2	2	3	3	3	3	3	3
1	1	1	1	1	2	2	2	2	2	2	2	2	3	3	3	3	3
1	1	1	1	1	2	2	2	2	3	3	3	3	3	3	3	3	3
1	1	1	1	1	2	2	2	2	2	2	2	3	3	3	3	3	4
1	1	1	1	2	2	2	2	2	2	2	2	3	3	3	3	3	4
1	1	1	1	2	2	2	2	3	3	3	3	3	3	3	3	3	4
1	1	1	1	2	2	2	2	2	2	2	2	3	3	3	3	4	4
1	1	1	1	2	2	2	3	3	3	3	3	3	3	3	3	3	4
1	1	1	1	2	2	2	2	2	2	2	2	3	3	3	3	4	4
1	1	1	1	2	2	2	2	2	2	2	2	2	2	3	3	3	3
1	1	1	1	2	2	2	2	2	2	2	2	2	2	2	3	3	3
1	1	1	2	2	2	3	3	3	3	3	3	3	3	3	3	3	3
1	1	1	2	2	2	3	3	3	3	3	3	3	3	3	3	3	3
1	1	1	2	2	2	3	3	3	3	3	3	3	3	3	3	3	3
1	1	2	2	2	3	3	3	3	3	3	3	3	3	3	3	3	3

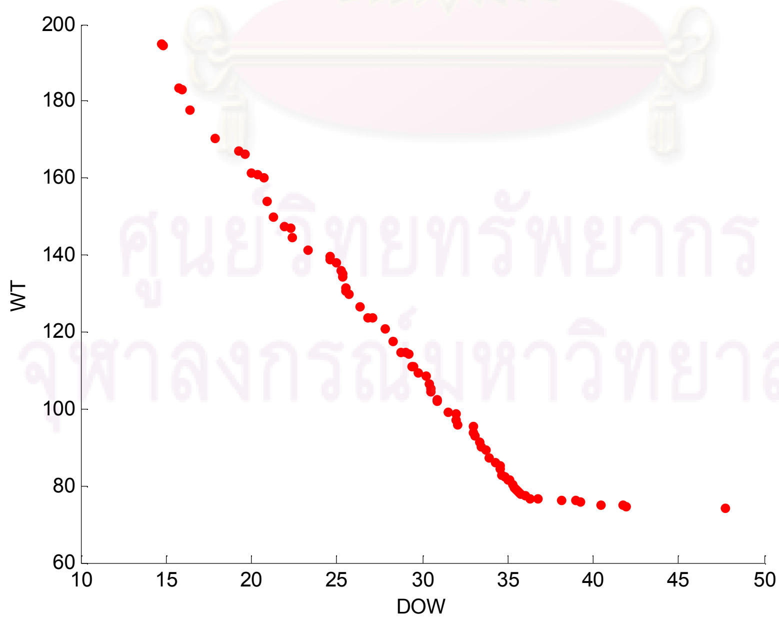
Columns 19 through 28

4	4	4	5	5	5	5	5	5	5
3	4	4	4	5	5	5	5	5	5
4	4	4	4	5	5	5	5	5	5
4	4	4	4	4	5	5	5	5	5

4	4	4	5	5	5	5	5	5	5
4	4	4	4	5	5	5	5	5	5
4	4	4	4	5	5	5	5	5	5
4	5	5	5	5	5	5	5	5	5
4	4	4	4	4	5	5	5	5	5
4	4	4	5	5	5	5	5	5	5
4	4	4	4	4	5	5	5	5	5
4	4	4	4	5	5	5	5	5	5
4	4	4	4	5	5	5	5	5	5
4	4	4	4	5	5	5	5	5	5
4	4	4	4	4	5	5	5	5	5
4	4	4	4	5	5	5	5	5	5
4	4	4	5	5	5	5	5	5	5
3	4	4	4	4	5	5	5	5	5
4	4	4	4	4	5	5	5	5	5
3	4	4	4	4	4	4	5	5	5
4	4	4	4	5	5	5	5	5	5
4	4	4	5	5	5	5	5	5	5
4	4	4	4	5	5	5	5	5	5
4	4	4	4	5	5	5	5	5	5
4	4	4	4	5	5	5	5	5	5
4	4	4	4	5	5	5	5	5	5
4	4	4	4	5	5	5	5	5	5
4	4	4	4	5	5	5	5	5	5
4	4	4	4	5	5	5	5	5	5
4	4	4	4	5	5	5	5	5	5
4	4	4	4	5	5	5	5	5	5
4	4	4	4	4	4	5	5	5	5
4	4	4	4	4	4	5	5	5	5
4	5	5	5	5	5	5	5	5	5
4	5	5	5	5	5	5	5	5	5
4	4	5	5	5	5	5	5	5	5
4	5	5	5	5	5	5	5	5	5
3	4	4	4	4	5	5	5	5	5
4	4	4	4	4	5	5	5	5	5
4	5	5	5	5	5	5	5	5	5
4	4	4	4	4	4	5	5	5	5
4	4	4	4	5	5	5	5	5	5
4	4	4	4	5	5	5	5	5	5
3	4	4	4	5	5	5	5	5	5
4	4	4	5	5	5	5	5	5	5
4	4	4	4	5	5	5	5	5	5
4	4	4	4	4	5	5	5	5	5
4	4	4	5	5	5	5	5	5	5
4	4	5	5	5	5	5	5	5	5
3	3	3	4	4	4	4	5	5	5
4	4	4	4	4	4	5	5	5	5
4	4	4	4	4	4	5	5	5	5
4	4	4	4	4	4	5	5	5	5
4	4	5	5	5	5	5	5	5	5

3	3	4	4	4	4	4	5	5	5
4	4	5	5	5	5	5	5	5	5
4	4	4	4	4	5	5	5	5	5
4	4	4	4	4	5	5	5	5	5
3	3	4	4	4	4	5	5	5	5
3	4	4	4	4	4	5	5	5	5
4	4	4	4	4	5	5	5	5	5
4	4	4	4	5	5	5	5	5	5
4	4	4	4	4	5	5	5	5	5
4	4	4	4	4	5	5	5	5	5
4	4	4	5	5	5	5	5	5	5
4	4	4	5	5	5	5	5	5	5
4	4	4	4	4	5	5	5	5	5
4	4	4	4	4	4	5	5	5	5
3	3	4	4	4	4	5	5	5	5
4	4	4	4	4	5	5	5	5	5
4	4	4	4	4	5	5	5	5	5
4	4	4	5	5	5	5	5	5	5
4	4	4	4	4	5	5	5	5	5
4	4	4	5	5	5	5	5	5	5
4	4	4	4	4	4	5	5	5	5
3	3	3	4	4	4	4	5	5	5
3	3	3	4	4	4	4	5	5	5
4	4	4	4	4	4	5	5	5	5
3	4	4	4	4	4	5	5	5	5
3	3	4	4	4	4	5	5	5	5
4	4	4	4	4	4	5	5	5	5

Elapsed time is 3869.078000 seconds.



<u>An example of results of MA at the side ratio 1:4:4 (1/9)</u>

45 tasks

Kilbridge&Wester_45task_cycle110_20:20:5

```
TS_task_minWS =
  Columns 1 through 21
    2    8    1   39   11    7    4   43   37    3    5    9    6   10   45   12   13   15
18   16   23
    2    8   39    1   11    7    4   43   37    3    5    9    6   10   45   12   13   15
18   16   24
    1    7   39   43   11   37    3    2    8    4    6   10    5    9   45   12   13   15
18   16   23
    1   39   43    7   11   37    3    2    8    4    6   10    5    9   45   12   13   15
18   23   16
    2   39    8    1   11    7    3    4   43   37    5    9    6   10   45   12   13   15
18   16   23
    2   39    8    1    7   11    3    4    6   43   37    5    9   10   45   12   13   15
18   16   23
    2   39    8    1   11    7    3    4   43   37    5    9    6   10   45   12   13   15
18   16   24
    2    1   11    3    7    4   39    8   43   37    5    9    6   10   45   12   13   15
24   16   18
    2   39    8    1   11    3    5    4    6   43   37    7    9   10   45   12   13   15
18   16   19
    2    8    1   11    7    3    5    4   43   37   39    9    6   10   45   12   13   15
16   18   19
    1    3   11    7    5    9   39   43   37    2    4    8    6   10   45   12   13   15
18   16   14
    2   39    1   12    7   37    3    5    9    4   43    8    6   10   45   11   13   14
29   15   18
    2   39   12   37    1    8    3    5    4   43    6   10    7    9   45   11   13   14
17   32   30
    2   39    1    3    7   12   37    4   43    8    5    9    6   10   45   11   13   14
29   15   23
    2   39    1   12    7   37    3    5    9    4   43    6    8   10   45   11   13   14
29   15   18
    2    8   39   43   44   45   42   41    9   29   32   30   31    5   11    1   10    6
12   13   15
    1    7    3    5    9   43   45   44   39   11   42   12   13   15   16   18   23
24   19   20   21
    1    3   11    7   12   43   45   44   39    5    9   13   15   24   23   16   42
18   19   20   21
   12    2    1   37   39    4    8    6   10   44   43   11   13   15   18   16   24
23   19   20    3
    2    8   39   43   44   45   42   41    9   30   29   32   31    5   11    1   10    6
12   13   15
```

1 3 11 12 7 43 45 44 39 5 9 13 15 24 23 18 16
42 19 20 21
11 12 43 45 39 44 13 2 8 42 4 6 10 15 24 16 23
18 19 20 21
12 43 45 39 44 2 8 42 4 6 10 11 13 15 16 23 24
18 19 20 21
12 43 45 39 44 2 8 42 4 6 10 11 13 15 18 23 24
16 19 20 21
12 43 45 39 44 2 8 42 4 6 10 11 13 15 18 16 23
24 19 20 21
44 12 43 45 39 2 8 42 4 6 10 11 13 15 16 23 24
18 19 20 21
44 12 43 45 39 2 8 42 4 6 10 11 13 15 18 16 23
24 19 20 21
12 43 45 37 39 1 7 3 5 9 2 4 6 11 13 15 23 24
16 8 14
1 44 12 3 43 45 42 41 9 32 31 29 30 2 8 7 39 4
6 37 5
1 2 8 43 45 11 44 42 39 3 5 7 9 12 13 14 32 17
25 15 18
39 44 12 37 11 43 1 7 3 5 9 2 4 6 8 10 13 14
29 17 30
12 43 45 44 39 11 2 8 4 6 10 42 13 15 16 24 23
18 19 20 21
1 7 11 43 45 44 39 12 3 5 9 42 13 15 18 16 24
23 19 20 21
11 43 45 44 39 12 2 8 4 6 10 42 13 15 18 16 23
24 19 20 21
39 11 44 12 37 43 1 7 3 5 9 2 4 6 8 10 13 14
25 17 30
39 11 44 12 37 43 1 7 3 5 9 2 4 6 8 10 13 14
17 30 31
39 44 12 37 11 43 1 7 3 5 9 2 4 6 8 10 13 14
25 17 30
39 12 37 44 43 11 1 7 3 5 9 2 4 6 8 10 13 14
25 17 30
39 11 44 12 37 43 1 7 3 5 9 2 4 6 8 10 13 14
25 17 26
11 44 39 12 37 43 1 7 3 5 9 2 4 6 8 10 13 14
25 17 26
11 43 45 39 44 2 8 4 6 10 12 13 15 18 24 23 16
19 20 21 22
43 45 44 39 37 12 11 13 15 18 23 16 24 19 20 21 22
1 7 3 5
39 11 43 45 44 2 37 8 4 6 10 12 13 15 18 23 24
16 19 20 21
43 11 39 45 1 7 3 12 13 15 24 23 16 18 19 20 21
44 42 41 9
12 39 45 1 7 3 43 11 13 15 24 23 16 18 19 20 21
44 42 41 9

44 39 11 45 43 37 2 8 4 6 10 12 13 15 18 24 16
19 20 23 21
39 11 43 45 44 2 8 37 4 6 10 12 13 15 18 23 24
16 19 20 21
39 11 44 43 37 1 3 5 7 9 2 8 4 6 10 12 13 14
25 17 30
12 2 1 3 7 43 44 45 42 39 8 5 9 37 11 13 14 29
31 32 25
11 1 7 39 43 37 45 44 2 4 8 6 10 42 12 13 14 32
17 25 15
12 39 1 11 13 15 23 16 3 43 44 45 42 37 2 8 24
7 14 17 25
1 12 37 7 2 44 39 3 8 11 4 6 10 5 9 13 15 18
24 23 14
1 7 39 43 37 45 44 2 4 42 8 6 10 11 12 13 14 32
30 31 17
12 37 39 11 43 45 44 1 7 3 5 9 13 15 18 16 24
23 19 20 21
12 37 39 43 45 44 2 8 4 6 10 11 13 15 18 23 16
19 24 20 21
12 37 39 43 45 44 2 8 4 6 10 11 13 15 18 24 16
19 23 20 21
39 12 37 43 45 44 2 8 4 6 10 11 13 15 18 16 23
24 19 20 21
39 12 37 44 45 2 8 4 6 10 43 11 13 15 18 23 16
24 19 20 21
12 37 2 11 4 1 3 44 5 6 13 7 9 39 8 10 14 32
31 17 25
12 37 44 45 39 2 8 4 6 10 43 11 13 15 18 16 19
23 24 20 21
12 2 43 44 45 42 8 39 1 7 11 13 14 29 31 32 25
17 27 30 37
11 2 8 43 44 45 42 39 12 13 15 23 24 16 1 7 14
29 31 32 25
11 2 8 43 44 45 42 39 12 13 15 24 23 16 1 7 14
30 31 29 17
12 2 8 43 45 44 42 39 1 7 11 13 15 24 23 16 14
30 17 31 29
11 2 43 44 45 42 39 8 12 13 15 24 23 16 1 7 14
30 31 29 17
39 11 43 37 44 45 42 1 3 12 13 15 24 16 18 23 2
8 7 14 17
39 1 11 3 43 37 44 45 42 12 13 15 24 18 16 23 2
8 7 14 17
2 11 4 12 37 1 3 13 5 6 39 15 23 18 24 8 43 10
7 9 14
12 37 2 1 3 7 39 8 5 9 43 11 13 14 15 16 25 29
30 18 19
2 11 1 7 12 8 37 13 15 23 16 14 32 25 30 31 18
19 43 4 29

12 37 2 1 3 7 39 8 5 9 43 11 13 14 17 15 31 27
29 24 30
12 2 37 1 39 4 6 11 13 15 16 18 19 23 24 43 8
10 7 14 30
2 8 12 37 11 1 13 15 16 39 23 4 6 10 24 43 7 14
17 25 32
2 1 3 11 12 37 8 13 39 5 7 14 31 17 30 9 29 32
43 27 15
2 8 12 1 11 13 15 16 23 24 4 6 10 7 14 32 29 17
25 30 31
2 8 12 11 1 13 15 16 24 23 4 6 37 43 10 7 14 17
27 31 32
2 8 12 11 1 13 15 16 24 23 4 6 37 10 43 7 14 17
27 31 32
Columns 22 through 42
24 14 17 25 27 32 30 29 19 33 34 20 36 35 26 31 21
22 28 38 40
14 32 17 29 25 30 23 27 19 33 20 21 35 34 36 26 31
22 28 38 40
24 14 17 25 27 32 30 29 19 33 34 20 36 35 26 31 21
22 28 38 40
24 14 17 25 27 29 30 32 19 33 34 20 36 35 26 31 21
22 28 38 40
24 14 17 25 27 29 30 32 19 33 34 20 36 35 26 31 21
22 28 38 40
24 14 17 27 29 30 32 19 33 34 20 36 35 25 26 31 21
22 28 38 40
23 14 17 27 32 29 31 19 33 35 20 36 34 25 26 30 21
22 28 38 40
19 14 17 32 30 27 23 33 34 35 36 31 25 26 29 20 21
22 28 38 40
24 14 17 27 29 30 23 33 34 36 35 32 20 21 25 26 31
22 28 38 40
14 29 30 31 17 32 27 23 24 33 35 36 34 20 21 25 26
22 28 38 40
29 32 30 31 25 17 26 27 23 19 20 24 33 35 34 36 21
22 28 38 40
31 16 25 32 17 26 23 27 30 19 20 24 33 34 35 36 21
22 28 38 40
27 31 25 26 29 15 24 16 18 19 20 23 33 36 34 35 21
22 28 38 40
16 18 25 31 17 26 27 30 24 32 19 20 33 35 36 34 21
22 28 38 40
23 16 24 32 30 17 31 27 19 20 33 34 36 35 25 26 21
22 28 38 40
23 24 16 18 3 7 14 17 27 4 19 20 21 33 34 35 36
22 28 25 26
22 2 4 8 14 31 17 25 27 28 32 33 35 30 34 29 26
6 10 36 38

22 2 4 8 6 10 14 30 31 32 29 17 27 33 35 28 34
25 26 36 38
5 7 14 31 30 17 25 26 32 29 9 27 33 36 34 35 45
42 41 40 38
24 23 16 18 3 7 14 25 17 27 4 19 20 21 22 33 34
35 36 28 26
22 2 8 4 6 10 14 30 31 32 29 17 27 33 35 28 34
25 26 36 38
22 1 7 14 32 29 17 31 27 33 36 34 28 30 35 25 26
38 40 3 5
22 1 7 14 31 25 30 17 26 32 27 33 28 34 29 35 36
38 40 3 5
22 1 7 14 32 29 17 27 28 33 36 34 30 31 35 25 26
38 40 3 5
22 1 7 14 32 17 29 31 27 33 36 34 28 30 35 25 26
38 40 3 5
22 1 7 14 32 17 31 27 28 33 36 34 30 29 35 25 26
38 40 3 5
22 1 7 14 32 17 29 31 27 33 36 34 28 30 35 25 26
38 40 3 5
17 27 32 31 29 18 19 20 21 22 28 33 35 34 36 10 44
42 41 40 38
11 13 14 17 15 16 25 26 23 24 27 18 19 20 21 33 34
35 36 22 10
23 29 16 19 30 31 26 24 27 33 34 35 36 20 21 22 28
38 40 4 6
31 27 32 15 24 23 16 25 26 18 19 33 34 35 36 45 42
41 40 38 28
22 1 7 14 17 27 30 28 29 32 33 34 31 36 35 25 26
38 40 3 5
22 2 8 14 32 17 29 30 27 33 36 34 28 31 35 25 26
38 40 4 6
22 1 7 14 17 29 31 27 33 34 36 32 28 30 35 25 26
38 40 3 5
26 31 27 32 15 24 23 16 29 18 19 33 34 35 36 45 42
41 40 38 28
29 25 26 27 15 24 23 16 32 18 19 33 35 34 20 36 45
42 41 40 38
29 31 27 32 15 24 23 16 26 18 19 33 34 35 36 45 42
41 40 38 28
26 32 31 27 15 24 23 16 29 18 19 33 34 36 20 35 45
42 41 40 38
31 27 32 29 15 24 23 16 30 18 19 33 34 35 20 36 45
42 41 40 38
32 31 30 27 15 24 23 16 29 18 19 33 34 36 20 35 45
42 41 40 38
1 7 14 30 29 17 27 33 36 35 34 25 31 32 28 26 38
40 3 5 9
9 2 8 14 31 30 32 17 27 33 34 29 28 35 36 25 26
4 6 10 42

22 1 7 14 31 30 32 17 27 33 34 29 28 35 36 25 26
3 5 9 42
40 38 36 31 30 35 32 10 28 29 34 33 27 26 25 17 14
8 22 5 37
40 38 36 31 30 34 32 35 10 28 29 33 27 26 25 17 14
8 22 5 37
22 1 7 14 32 31 17 27 33 36 34 28 30 29 35 25 26
3 5 9 42
22 1 7 14 31 30 32 17 27 33 34 29 28 35 36 25 26
3 5 9 42
26 32 31 27 15 24 23 16 29 18 19 33 34 36 20 35 45
42 41 40 38
17 27 30 15 23 24 16 18 19 20 21 22 28 33 36 34 35
26 4 6 10
18 23 29 16 19 20 30 31 26 24 27 33 34 35 36 21 22
28 3 5 9
32 30 27 26 31 4 6 5 9 18 19 33 35 36 34 20 21
22 28 38 40
32 17 25 29 16 19 30 31 26 27 33 36 34 35 20 21 22
28 38 40 43
15 16 18 19 24 23 25 26 27 33 35 34 29 36 20 21 22
28 3 5 9
22 2 8 14 17 29 31 30 32 27 33 34 35 36 25 26 28
4 6 10 42
22 1 7 14 30 25 17 27 33 35 34 36 31 32 28 26 29
3 5 9 42
22 1 7 14 30 31 25 17 27 32 33 34 36 35 28 29 26
3 5 9 42
22 1 7 14 29 17 31 30 32 27 33 34 35 36 25 26 28
3 5 9 42
22 1 7 14 17 29 31 30 32 27 33 34 36 35 25 26 28
3 5 9 42
29 30 26 27 15 23 16 24 18 19 33 35 36 34 20 21 22
28 38 40 43
22 1 7 14 31 29 17 30 32 27 33 34 36 35 25 26 28
3 5 9 42
3 5 9 15 23 24 16 18 19 20 21 22 28 26 33 35 34
36 4 6 10
17 27 30 37 3 5 9 4 6 10 18 19 20 21 22 28 33
35 34 36 26
27 37 4 6 10 3 5 9 18 19 33 34 36 20 21 22 28
35 32 25 26
27 37 4 6 10 3 5 9 18 19 33 34 36 20 21 22 28
35 32 25 26
27 37 4 6 10 3 5 9 18 19 33 34 36 20 21 22 28
35 32 25 26
32 30 27 31 25 26 4 6 5 9 19 33 35 36 34 20 21
22 28 38 40
32 30 27 31 25 26 4 6 5 9 19 33 35 36 34 20 21
22 28 38 40

29 32 25 30 17 31 27 26 16 19 33 35 34 20 36 21 22
28 38 40 41
20 21 24 31 23 32 17 26 27 33 35 36 34 22 28 38 40
4 6 10 41
20 21 22 6 10 17 27 26 28 3 5 9 24 33 35 36 34
38 40 39 41
32 16 23 25 18 19 33 35 34 20 21 22 28 36 26 38 40
4 6 10 41
17 27 33 35 34 31 32 36 29 20 21 3 5 9 25 26 22
28 38 40 41
29 27 30 31 18 19 33 35 34 20 21 22 28 3 5 9 36
26 38 40 41
18 16 23 19 20 24 33 34 35 36 21 22 28 25 26 38 40
4 6 10 41
27 37 43 39 26 18 19 33 36 35 34 20 3 5 9 21 22
28 38 40 41
29 30 18 19 33 36 20 21 22 28 3 5 9 35 34 25 26
38 40 39 41
29 30 18 19 33 36 34 20 21 22 28 3 5 9 35 25 26
38 40 39 41

Columns 43 through 45

41 42 44
41 42 44
41 42 44
41 42 44
41 42 44
41 42 44
41 42 44
41 42 44
41 42 44
41 42 44
41 42 44
41 42 44
41 42 44
41 42 44
41 42 44
38 37 40
40 41 37
40 41 37
28 22 21
38 37 40
40 41 37
9 41 37
9 41 37
9 41 37
9 41 37
9 41 37
9 41 37
26 25 30
28 38 40

```
   10    41    37
   22    21    20
    9    41    37
   10    41    37
    9    41    37
   22    21    20
   28    22    21
   22    21    20
   28    22    21
   28    22    21
   28    22    21
   41    42    37
   41    40    38
   41    40    38
    2     4     6
    2     4     6
   41    40    38
   41    40    38
   28    22    21
   38    40    41
   41    40    38
   10    29    41
   45    42    41
   41    40    38
   41    40    38
   41    40    38
   41    40    38
   41    40    38
   41    40    38
   45    42    41
   41    40    38
   38    40    41
   38    40    41
   38    40    41
   38    40    41
   38    40    41
   10    29    41
   10    29    41
   42    45    44
   42    45    44
   42    45    44
   42    45    44
   42    45    44
   42    45    44
   42    45    44
   42    45    44
   42    45    44
   42    45    44
position =
  Columns 1 through 21
```

1 1 1 1 1 1 1 2 2 1 1 1 1 1 2 1 1 1 1
1 1
1 1 1 1 1 1 1 2 2 1 1 1 1 1 2 1 1 1 1
1 1
1 1 1 2 1 2 1 1 1 1 1 1 1 1 2 1 1 1 1
1 1
1 1 2 1 1 2 1 1 1 1 1 1 1 1 2 1 1 1 1
1 1
1 1 1 1 1 1 1 1 2 2 1 1 1 1 2 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 2 2 1 1 1 2 1 1 1 1
1 1
1 1 1 1 1 1 1 1 2 2 1 1 1 1 2 1 1 1 1
1 1
1 1 1 1 1 1 1 1 2 2 1 1 1 1 2 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 2 2 1 1 1 2 1 1 1 1
1 1
1 1 1 1 1 1 1 1 2 2 1 1 1 1 2 1 1 1 1
1 1
1 1 1 1 1 1 1 2 2 1 1 1 1 1 2 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1
1 1
1 1 1 2 2 2 2 2 2 2 2 2 2 2 1 1 2 2 1
1 1
1 1 1 1 1 2 2 2 1 1 2 1 1 1 1 1 1 1 1
1 1
1 1 1 1 1 2 2 2 1 1 1 1 1 1 1 1 2 1 1
1 1
1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1
1 1
1 1 1 2 2 2 2 2 2 2 2 2 2 2 1 1 2 2 1
1 1
1 1 1 1 1 2 2 2 1 1 1 1 1 1 1 1 1 2 1
1 1
1 1 2 2 1 2 1 1 1 2 1 1 1 1 1 1 1 1 1
1 1
1 2 2 1 2 1 1 2 1 1 1 1 1 1 1 1 1 1 1
1 1
1 2 2 1 2 1 1 2 1 1 1 1 1 1 1 1 1 1 1
1 1
1 2 2 1 2 1 1 2 1 1 1 1 1 1 1 1 1 1 1
1 1

2 1 2 2 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1
1 1
2 1 2 2 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1
1 1
1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 2 1 1 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1
1 1
1 1 1 2 2 1 2 2 1 1 1 1 1 1 1 1 1 1 1
1 1
1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 2 2 2 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1
1 1
1 1 1 2 2 2 1 1 1 1 1 2 1 1 1 1 1 1 1
1 1
1 2 2 2 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1
1 1
1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 2 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
2 2 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 2 2 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1
1 1
2 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2
2 2
1 1 2 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2 2
2 2
2 1 1 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 2 2 2 1 1 2 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 1 2 2 2 2 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 2 2 2 2 1 1 1 1 1 2 1 1 1 1 1
1 1

1 1 1 1 1 1 1 1 1 2 2 2 2 1 1 1 1 1 1
1 1
1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 2 2 2 2 1 1 2 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1

```
    1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
1     1
    1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
1     1
  Columns 22 through 42
    1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
1     1
    1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
1     1
    1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
1     1
    1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
1     1
    1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
1     1
    1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
1     1
    1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
1     1
    1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
1     1
    1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
1     1
    1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
1     1
    1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
1     1
    1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
1     1
    1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
1     1
    1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
1     1
    1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
1     1
    1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
1     1
    1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
1     1
    1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
1     1
    1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     2     2     2
2     2
    1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
1     1
    1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
1     1
    1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
1     1
```

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2
2 2
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2
2 2
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2
2 2
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2
2 2
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2
2 2
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2
2 2
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2
2 2
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2
2 2
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 2
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1
1 1
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1
1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 2
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 2

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2
2 2
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 2
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 2
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 2
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 2
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 2
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 2
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1

```
    1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
1     1
    1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
1     1
    1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
1     1
    1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
1     1
    1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
1     1
 Columns 43 through 45
    1     1     1
    1     1     1
    1     1     1
    1     1     1
    1     1     1
    1     1     1
    1     1     1
    1     1     1
    1     1     1
    1     1     1
    1     1     1
    1     1     1
    1     1     1
    1     1     1
    1     1     1
    1     1     1
    1     1     1
    1     1     1
    2     2     1
    1     1     1
    1     1     1
    1     1     1
    1     1     1
    1     1     1
    1     1     1
    1     1     1
    1     1     1
    2     1     1
    1     1     1
    1     1     1
    2     2     1
    1     1     1
    1     1     1
    1     1     1
    2     2     1
    2     2     1
    2     2     1
    2     2     1
    2     2     1
```

```
     2     2     1
     1     1     1
     2     2     1
     2     2     1
     1     1     1
     1     1     1
     2     2     1
     2     2     1
     2     2     1
     1     1     1
     2     2     1
     1     1     1
     2     2     1
     2     2     1
     2     2     1
     2     2     1
     2     2     1
     2     2     1
     2     2     1
     2     2     1
     2     2     1
     1     1     1
     1     1     1
     1     1     1
     1     1     1
     1     1     1
     1     1     1
     1     1     1
     1     1     1
     1     1     1
     1     1     1
     1     1     1
     1     1     1
     1     1     1
     1     1     1
     1     1     1
     1     1     1
     1     1     1
WT_DOW_J =
    2.6494  202.0318    7.0000
    2.9326  200.4040    7.0000
    3.1801  199.3020    7.0000
    3.2613  197.9551    7.0000
    3.4448  196.2664    7.0000
    3.6099  195.3294    7.0000
    3.8557  195.1723    7.0000
    4.2386  194.8584    7.0000
    4.4299  194.3178    7.0000
    4.6839  191.9549    7.0000
    4.7130  191.2224    7.0000
```

4.8669	186.7307	7.0000
5.1432	184.7886	7.0000
5.2341	184.4594	7.0000
5.4071	184.4184	7.0000
5.5972	183.9639	7.0000
5.6877	180.8818	7.0000
5.7260	179.3982	7.0000
6.0914	178.7446	7.0000
6.1468	178.4449	7.0000
6.1922	175.9728	7.0000
6.3984	174.8486	7.0000
6.5879	173.8872	7.0000
6.6837	172.3773	7.0000
6.9301	170.7194	7.0000
7.2668	170.4085	7.0000
7.3555	168.7506	7.0000
7.4973	168.3843	7.0000
7.7228	167.0783	7.0000
7.7561	166.8652	7.0000
7.8074	166.2142	7.0000
7.8252	165.5768	7.0000
7.9284	164.6277	7.0000
8.0277	163.9190	7.0000
8.1428	163.7552	7.0000
8.2608	163.2430	7.0000
8.3050	163.0464	7.0000
8.3266	161.4221	7.0000
8.4972	160.0752	7.0000
8.5750	159.3664	7.0000
8.7191	159.1729	7.0000
8.7229	157.9463	7.0000
8.8219	157.5150	7.0000
8.9060	156.7592	7.0000
9.0070	156.5974	7.0000
9.0338	156.0042	7.0000
9.0444	155.1819	7.0000
9.3092	154.5335	7.0000
9.3135	154.2820	7.0000
9.4666	153.9988	7.0000
9.5723	153.7313	7.0000
9.6029	153.5934	7.0000
9.6920	153.1956	7.0000
9.8921	149.7451	7.0000
10.0514	148.5487	7.0000
10.1310	148.1176	7.0000
10.1346	147.6398	7.0000
10.4922	145.9992	7.0000
10.6307	145.7729	7.0000
10.6390	144.9460	7.0000
10.7348	143.6620	7.0000

```
   11.0002  141.3930    7.0000
   11.0846  141.3148    7.0000
   11.1045  141.0325    7.0000
   11.1348  140.9703    7.0000
   11.3072  139.5715    7.0000
   11.3558  139.1114    7.0000
   11.6983  138.4320    7.0000
   11.8465  136.9107    7.0000
   11.9756  136.8553    7.0000
   12.1084  134.3763    7.0000
   12.5494  133.9509    7.0000
   12.6403  133.2421    7.0000
   12.8048  133.1606    7.0000
   12.8693  131.5843    7.0000
   13.3842  129.5621    7.0000
   15.2015  127.9043    7.0000
Define_Station =
  Columns 1 through 21
     1     1     1     1     1     1     1     1     2     2     2     2     2     3     3     3     3     3     3
4     4
     1     1     1     1     1     1     1     1     2     2     2     2     2     3     3     3     3     3     3
4     4
     1     1     1     1     1     1     1     2     2     2     2     2     2     3     3     3     3     3     3
4     4
     1     1     1     1     1     1     1     2     2     2     2     2     2     3     3     3     3     3     3
4     4
     1     1     1     1     1     1     1     1     2     2     2     2     2     3     3     3     3     3     3
4     4
     1     1     1     1     1     1     1     1     2     2     2     2     2     3     3     3     3     3     3
4     4
     1     1     1     1     1     1     1     1     2     2     2     2     2     3     3     3     3     3     3
4     4
     1     1     1     1     1     1     1     1     2     2     2     2     2     3     3     3     3     3     4
4     4
     1     1     1     1     1     1     1     1     2     2     2     2     2     3     3     3     3     3     3
4     4
     1     1     1     1     1     1     1     2     2     2     2     2     2     3     3     3     3     3     4
4     4
     1     1     1     1     1     1     1     2     2     2     2     2     2     3     3     3     3     3     3
4     4
     1     1     1     1     1     1     1     1     2     2     2     2     2     2     3     3     3     3     4
4     4
     1     1     1     1     1     1     1     1     2     2     2     2     2     2     3     3     3     3     4
4     4
     1     1     1     1     1     1     1     1     1     2     2     2     2     2     3     3     3     3     4
4     4
     1     1     1     1     1     1     1     1     2     2     2     2     2     2     3     3     3     3     4
4     4
     1     1     1     1     1     1     1     1     2     2     2     2     2     2     2     3     3     3     3
3     4
```

1 1 1 1 1 1 2 2 2 2 2 2 2 3 3 3 3 3 4
4 4
1 1 1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 4
4 4
1 1 1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3
3 4
1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 3 3 3 3
3 4
1 1 1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 4
4 4
1 1 1 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4
4 4
1 1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3 4
4 4
1 1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3 4
4 4
1 1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3 4
4 4
1 1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3 4
4 4
1 1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3 4
4 4
1 1 1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3
4 4
1 1 1 1 1 1 1 2 2 2 2 2 2 2 3 3 3 3 3
3 3
1 1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3 3
4 4
1 1 1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3
3 3
1 1 1 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4
4 4
1 1 1 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4
4 4
1 1 1 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4
4 4
1 1 1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3
4 4
1 1 1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3
3 3
1 1 1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3
4 4
1 1 1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3
4 4
1 1 1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3
4 4
1 1 1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3
4 4
1 1 1 1 1 1 1 2 2 2 2 2 2 2 3 3 3 3 3
4 4

1 1 1 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4
4 4
1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 3 3 3 3
3 4
1 1 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4
4 4
1 1 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4
4 4
1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 3 3 3 3
3 4
1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 3 3 3 3
3 4
1 1 1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3
4 4
1 1 1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3
3 3
1 1 1 1 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3
3 4
1 1 1 1 1 1 1 2 2 2 2 2 2 2 3 3 3 3 3
4 4
1 1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3 3
4 4
1 1 1 1 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3
3 3
1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 3 3 3 3
3 4
1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 3 3 3 3
3 4
1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 3 3 3 3
3 4
1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 3 3 3 3
3 4
1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 3 3 3 3
3 4
1 1 1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3
3 4
1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 3 3 3 3
3 4
1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 3 3 3
3 3
1 1 1 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 3
3 4
1 1 1 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 3
3 4
1 1 1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3
4 4
1 1 1 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 3
3 4
1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3
3 4

```
    1    1    1    1    1    1    1    1    1    2    2    2    2    2    2    3    3    3    3
3    4
    1    1    1    1    1    1    1    1    2    2    2    2    2    2    3    3    3    3    3
4    4
    1    1    1    1    1    1    1    1    2    2    2    2    2    2    3    3    3    3    3
3    3
    1    1    1    1    1    1    1    1    2    2    2    2    2    3    3    3    3    3    3
3    3
    1    1    1    1    1    1    1    1    2    2    2    2    2    2    3    3    3    3    3
3    3
    1    1    1    1    1    1    1    1    2    2    2    2    2    2    3    3    3    3    3
4    4
    1    1    1    1    1    1    1    1    2    2    2    2    2    3    3    3    3    3    4
4    4
    1    1    1    1    1    1    1    1    1    2    2    2    2    2    2    3    3    3    3
3    3
    1    1    1    1    1    1    1    2    2    2    2    3    3    3    3    3    3    4    4
4    4
    1    1    1    1    1    1    1    2    2    2    2    3    3    3    3    3    3    4    4
4    4
    1    1    1    1    1    1    1    2    2    2    2    3    3    3    3    3    3    4    4
4    4
  Columns 22 through 42
    4    4    5    5    5    5    5    5    5    5    5    6    6    6    6    6    6    7    7
7    7
    4    4    4    4    5    5    5    5    5    5    5    6    6    6    6    6    6    7    7
7    7
    4    4    5    5    5    5    5    5    5    5    5    6    6    6    6    6    6    7    7
7    7
    4    4    5    5    5    5    5    5    5    5    5    6    6    6    6    6    6    7    7
7    7
    4    4    5    5    5    5    5    5    5    5    5    6    6    6    6    6    6    7    7
7    7
    4    4    5    5    5    5    5    5    5    5    5    5    5    6    6    6    6    7    7
7    7
    4    4    5    5    5    5    5    5    5    5    5    5    5    6    6    6    6    7    7
7    7
    4    4    4    5    5    5    5    5    5    5    5    5    6    6    6    6    6    7    7
7    7
    4    4    4    4    5    5    5    5    5    5    5    5    5    6    6    6    6    7    7
7    7
    4    4    4    4    4    4    5    5    5    5    5    5    6    6    6    6    6    7    7
7    7
    4    4    4    4    4    5    5    5    5    5    5    5    6    6    6    6    6    7    7
7    7
    4    4    4    4    4    5    5    5    5    5    5    5    6    6    6    6    6    7    7
7    7
    4    4    4    4    4    4    5    5    5    5    5    5    6    6    6    6    6    7    7
7    7
```

4 4 4 5 5 5 5 5 5 5 5 5 6 6 6 6 6 7 7
7 7

4 4 4 5 5 5 5 5 5 5 5 5 5 6 6 6 6 7 7
7 7

4 4 4 4 5 5 5 5 5 5 5 5 6 6 6 6 6 7 7
7 7

4 4 5 5 5 5 5 5 6 6 6 6 6 6 6 6 6 7 7
7 7

4 4 5 5 5 5 5 5 6 6 6 6 6 6 6 6 7 7 7
7 7

4 4 4 4 4 4 5 5 5 5 5 5 5 6 6 6 6 6 6
6 6

4 4 4 4 5 5 5 5 5 5 6 6 6 6 6 7 7 7 7
7 7

4 4 5 5 5 5 5 5 6 6 6 6 6 6 6 6 7 7 7
7 7

4 4 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 7 7
7 7

4 4 5 5 5 5 5 5 6 6 6 6 6 6 6 6 6 7 7
7 7

4 4 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6 7 7
7 7

4 4 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 7 7
7 7

4 4 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6 7 7
7 7

4 4 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 7 7
7 7

4 4 4 4 4 4 4 4 5 5 5 6 6 6 6 6 6 7 7
7 7

4 4 4 4 4 4 5 5 5 5 5 6 6 6 6 6 6 7 7
7 7

4 4 4 4 4 4 5 5 5 5 5 5 5 5 6 6 6 7 7
7 7

4 4 4 4 4 4 5 5 5 5 5 5 5 6 6 6 6 6 6
6 7

4 4 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6 7 7
7 7

4 4 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 7 7
7 7

4 4 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 7 7
7 7

4 4 4 4 4 4 5 5 5 5 5 5 5 6 6 6 6 6 6
6 7

4 4 4 4 4 4 5 5 5 5 5 5 5 6 6 6 6 6 6
6 6

4 4 4 4 4 4 5 5 5 5 5 5 5 6 6 6 6 6 6
6 7

4 4 4 4 4 4 5 5 5 5 5 5 5 6 6 6 6 6 6
6 6

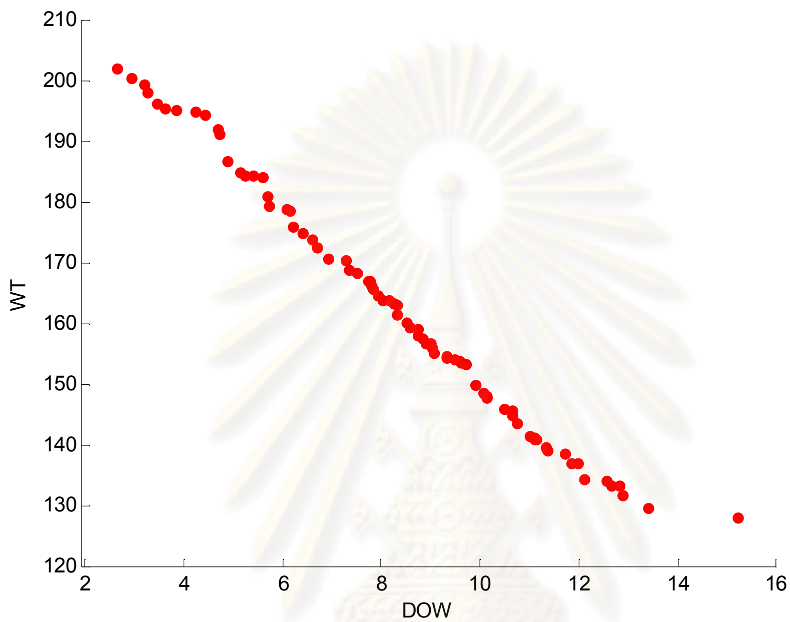
4	4	4	4	4	4	5	5	5	5	5	5	5	6	6	6	6	6	6	6	6
4	4	4	4	4	4	5	5	5	5	5	5	5	6	6	6	6	6	6	6	6
4	4	5	5	5	5	5	5	5	5	5	6	6	6	6	6	6	6	7	7	7
4	4	4	5	5	5	5	5	5	5	5	5	6	6	6	6	6	6	7	7	7
4	4	4	5	5	5	5	5	5	5	5	5	6	6	6	6	6	6	7	7	7
4	4	4	5	5	5	5	5	5	5	5	6	6	6	6	6	6	7	7	7	7
4	4	4	5	5	5	5	5	5	5	5	6	6	6	6	6	6	7	7	7	7
4	4	4	5	5	5	5	5	5	5	5	6	6	6	6	6	6	6	7	7	7
4	4	4	5	5	5	5	5	5	5	5	5	6	6	6	6	6	6	7	7	7
4	4	4	4	4	4	5	5	5	5	5	5	5	6	6	6	6	6	6	6	6
4	4	4	4	4	4	5	5	5	5	5	6	6	6	6	6	6	6	7	7	7
4	4	4	4	4	4	4	5	5	5	5	5	5	5	5	6	6	6	7	7	7
4	4	4	4	4	4	5	5	5	5	5	5	5	6	6	6	6	6	7	7	7
4	4	4	5	5	5	5	5	5	5	5	5	6	6	6	6	6	7	7	7	7
3	4	4	4	4	4	5	5	5	5	5	5	5	5	5	6	6	6	7	7	7
4	4	4	5	5	5	5	5	5	5	5	5	6	6	6	6	6	6	7	7	7
4	4	4	5	5	5	5	5	5	6	6	6	6	6	6	6	6	6	7	7	7
4	4	4	5	5	5	5	5	5	5	6	6	6	6	6	6	6	6	7	7	7
4	4	4	5	5	5	5	5	5	5	5	5	6	6	6	6	6	6	7	7	7
4	4	4	5	5	5	5	5	5	5	5	5	6	6	6	6	6	6	7	7	7
4	4	4	4	4	4	5	5	5	5	5	5	6	6	6	6	6	7	7	7	7
4	4	4	5	5	5	5	5	5	5	5	5	6	6	6	6	6	6	7	7	7
3	3	4	4	4	4	5	5	5	5	5	6	6	6	6	6	6	6	7	7	7
4	4	4	4	4	4	5	5	5	5	5	5	5	6	6	6	7	7	7	7	7
4	4	4	4	4	4	5	5	5	5	5	5	5	5	6	6	6	7	7	7	7

```
    4     4     4     4     4     4     5     5     5     5     5     5     5     5     6     6     6     7     7
7     7
    4     4     4     4     4     4     5     5     5     5     5     5     5     5     6     6     6     7     7
7     7
    4     4     4     4     4     4     4     5     5     5     5     5     5     6     6     6     6     6     7
7     7
    4     4     4     4     4     4     4     5     5     5     5     5     5     6     6     6     6     6     7
7     7
    4     4     4     4     5     5     5     5     5     5     5     5     6     6     6     6     6     7     7
7     7
    3     4     4     4     5     5     5     5     5     5     5     6     6     6     6     6     6     6     7
7     7
    3     4     4     4     5     5     5     5     5     5     6     6     6     6     6     7     7     7     7
7     7
    3     4     4     4     4     4     5     5     5     5     5     6     6     6     6     6     6     6     7
7     7
    4     4     4     4     4     4     4     5     5     5     5     5     6     6     6     6     6     7     7
7     7
    4     4     4     4     4     4     5     5     5     5     5     6     6     6     6     6     7     7     7
7     7
    3     3     4     4     4     4     4     4     5     5     5     5     6     6     6     6     6     6     7
7     7
    4     4     4     4     4     4     5     5     5     5     5     5     5     5     6     6     6     7     7
7     7
    4     4     4     4     4     4     4     5     5     5     6     6     6     6     6     6     7     7     7
7     7
    4     4     4     4     4     4     4     4     5     5     5     6     6     6     6     6     6     7     7
7     7
 Columns 43 through 45
    7     7     7
    7     7     7
    7     7     7
    7     7     7
    7     7     7
    7     7     7
    7     7     7
    7     7     7
    7     7     7
    7     7     7
    7     7     7
    7     7     7
    7     7     7
    7     7     7
    7     7     7
    7     7     7
    7     7     7
    7     7     7
    7     7     7
    7     7     7
    7     7     7
```

7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7
7	7	7

```
     7     7     7
     7     7     7
     7     7     7
     7     7     7
     7     7     7
     7     7     7
```

Elapsed time is 6092.348994 seconds.



61 tasks

Kim_61task_cycle600_27:27:7

```
TS_task_minWS =
  Columns 1 through 16
    18    15    14    46    26    17     9     8    12    11    10    39    40    41    51    52
    39    40     4    15    46    26    16    18    19    51    52     1     2     3    20    24
    46    15    14    18    19    35    36    37    38    51    52    39    40    41    42    20
    46    15    14    18    19    51    52    39    40    41     1     2     3    20    24    25
    15    14    13    16    46    39    40    41    18    19    51    52     1     2     3    20
  Columns 17 through 32
     1     2     3    19    20    24    35    36    37    38     4     5     6     7    28    29
    28    29    30    31    34    32    33     5     6     7     8    10     9    13    11    12
    24    25     1     2     3    21    28    29    30    31    32    33     4     5     6     7
    21    35    36    37    38    42     4     5     6     7     8     9    10    11    12    28
    24    25    21    28    29    30    31    34    32    33     4     5     6     7     8     9
  Columns 33 through 48
    30    34    31    32    33    42    43    44    48    45    47    49    50    21    22    23
    41    35    36    37    38    42    43    44    48    45    47    49    50    21    22    23
     8     9    10    11    12    34    43    44    47    48    45    49    50    22    23    27
    29    30    34    31    32    33    43    44    47    48    45    49    50    22    23    27
```

```
   10    11    35    36    37    38    42    43    44    48    45    47    49    50    22    23
  Columns 49 through 61
   25    27    53    54    57    55    56    59    60    58    61    16    13
   25    27    53    54    57    55    56    59    60    58    61    17    14
   53    54    57    59    60    55    56    58    16    26    61    17    13
   53    54    57    59    60    55    56    58    16    26    61    17    13
   27    53    54    57    55    56    59    60    58    26    61    17    12

position =
  Columns 1 through 16
     1     2     2     2     2     2     2     2     2     2     2     1     1     1     1     1
     1     1     1     2     2     2     2     1     1     1     1     1     1     1     1     1
     2     2     2     1     1     1     1     1     1     1     1     1     1     1     1     1
     2     2     2     1     1     1     1     1     1     1     1     1     1     1     1     1
     2     2     2     2     2     1     1     1     1     1     1     1     1     1     1     1
  Columns 17 through 32
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
  Columns 33 through 48
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
  Columns 49 through 61
     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     2     1
WT_DOW_J =
   26.4259  533.8860   10.0000
   26.6329  512.7287   10.0000
   28.1561  469.2010   10.0000
   29.9250  465.8266   10.0000
   42.9362  428.5206   10.0000
Define_Station =
  Columns 1 through 16
     1     1     1     1     1     1     1     2     2     2     2     2     3     3     3     3
     1     1     1     1     1     2     2     2     2     2     2     2     3     3     3     3
     1     1     1     1     1     1     1     2     2     2     2     2     2     3     3     3
     1     1     1     1     1     1     1     2     2     2     2     2     2     3     3     3
     1     1     1     1     1     2     2     2     2     2     2     2     3     3     3     3
  Columns 17 through 32
     3     3     3     4     4     4     4     4     4     5     5     5     5     5     5     5
     3     4     4     4     4     4     4     5     5     5     5     5     5     5     6     6
     3     3     4     4     4     4     4     4     4     5     5     5     5     5     5     6
```

3	3	4	4	4	4	4	4	4	5	5	5	5	5	5	5
3	4	4	4	4	4	4	5	5	5	5	5	5	5	6	6

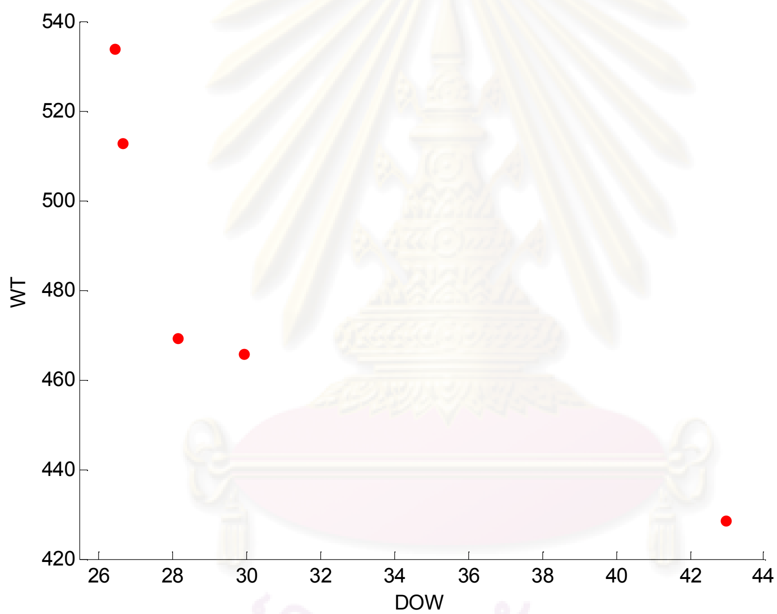
Columns 33 through 48

6	6	6	6	6	6	7	7	7	7	7	7	7	8	8	8
6	6	6	6	6	7	7	7	7	7	7	7	8	8	8	8
6	6	6	6	6	6	7	7	7	7	7	7	7	8	8	8
6	6	6	6	6	6	7	7	7	7	7	7	7	8	8	8
6	6	6	6	6	7	7	7	7	7	7	8	8	8	8	8

Columns 49 through 61

8	8	9	9	9	9	9	9	10	10	10	10	10
8	9	9	9	9	9	9	10	10	10	10	10	10
8	8	9	9	9	9	9	10	10	10	10	10	10
8	8	9	9	9	9	9	10	10	10	10	10	10
8	9	9	9	9	9	9	10	10	10	10	10	10

Elapsed time is 12551.515471 seconds.



An example of results of COIN at the side ratio 1:1:1 (1/3)

70 tasks

Tongue_70task_cycle251_23:23:24

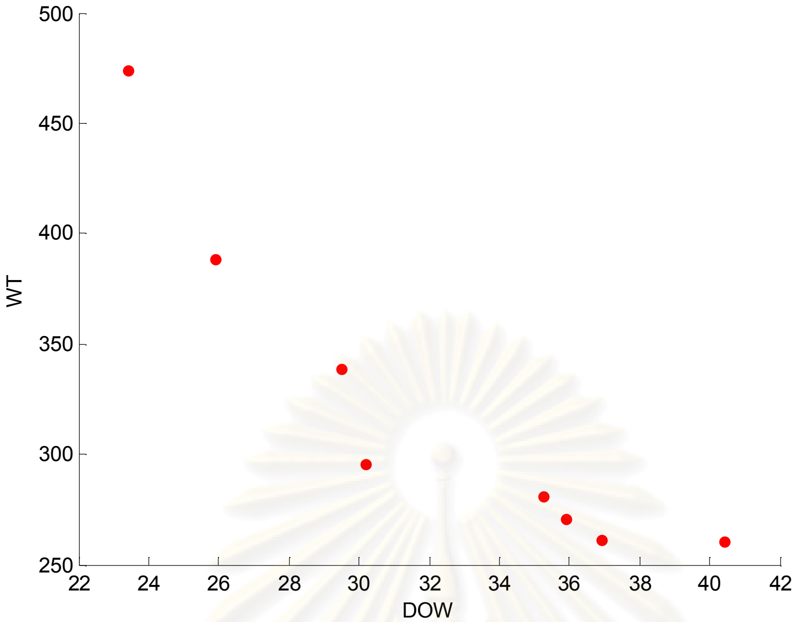
TS_task_minWS =

Columns 1 through 16

15	1	69	70	9	10	11	5	30	41	24	16	18	17	19	57
15	5	16	18	9	10	17	19	24	22	1	57	58	69	30	41
15	5	30	1	16	18	69	9	10	41	70	11	24	17	2	3
9	10	11	15	1	16	2	3	41	68	70	17	69	4	7	5
15	1	16	69	9	5	70	41	10	30	24	17	11	2	3	18
15	1	5	16	41	9	17	70	30	24	69	2	18	10	3	4

```
  15    1   16   18    9   10   17   70   41    2    3   68    5   24   19   30
  15    5   16    9   18   17   24   10   11    1   70   41    2   30    3   69
 Columns 17 through 32
  20   21   22   58    2   59    3   68    4    7    6    8   12   14   13   23
   2   70   11   59    3    4    7    6    8   68   12   20   21   14   13   23
   4    7    6    8   68   12   19   57   22   14   20   13   58   59   21   23
  30   24    6   18    8   12   19   57   13   20   58   59   14   22   21   23
  68    4    7    6    8   12   19   22   14   20   21   13   23   25   28   33
   6   19   11   57   58    7    8   22   12   59   13   20   21   14   23   68
  22   69   57   11    4    7    6    8   20   12   58   14   59   13   21   23
  19   68    4    6   57    7   22    8   58   12   20   14   21   13   59   23
 Columns 33 through 48
  25   29   31   27   32   33   26   34   28   35   61   56   48   44   62   60
  25   28   26   33   31   27   32   34   29   35   61   36   60   53   51   56
  33   31   34   32   25   28   27   29   26   35   61   60   56   48   51   49
  33   25   27   28   29   34   26   31   32   35   51   48   49   36   52   61
  31   32   57   58   26   27   59   34   29   35   53   56   61   51   62   44
  33   31   32   34   25   28   26   29   27   35   61   60   53   36   56   48
  25   29   26   33   28   31   27   34   32   35   61   44   62   48   51   56
  25   28   33   29   31   32   26   34   27   35   44   61   60   36   45   48
 Columns 49 through 64
  53   45   51   46   49   63   47   36   52   64   66   67   54   37   38   65
  52   62   48   54   44   63   64   65   37   38   39   40   42   66   55   67
  44   62   52   53   54   45   36   63   55   37   38   39   40   64   66   46
  53   56   62   44   54   45   46   47   55   60   37   63   64   38   67   66
  63   64   67   66   60   45   36   52   37   48   54   55   49   38   39   46
  49   44   62   37   51   63   64   52   66   54   55   65   45   46   38   47
  53   63   36   45   46   49   47   37   38   64   52   66   67   54   55   60
  46   53   62   56   47   49   37   63   51   64   52   54   38   67   66   55
 Columns 65 through 70
  39   40   42   43   50   55
  45   49   43   46   47   50
  47   65   67   42   43   50
  39   65   40   42   43   50
  65   40   42   47   43   50
  67   39   40   42   43   50
  65   39   40   42   43   50
  39   65   40   42   43   50
Station =
 Columns 1 through 16
   1    1    1    1    1    4    5    5    5    5    5    5   10   10   10   10
   1    1    1    8    8   10   10   10   10   10   10   11   11   11   11   11
   1    1    1    1    1    4    4    4    5    5    5    5    5    6    6    6
   2    2    2    3    3    3    4    4    4    4    5    5    5    5    5    5
   2    2    2    2    3    3    3    3    3    3    3    4    4    4    6    6
   1    1    1    1    1    5    5    5    5    5    5    6    6    7    7    7
   2    2    2    9    9   10   10   10   10   10   12   12   12   12   12   12
   2    2    2    3    3    4    4    4    4    5    5    5    5    5    5    5
 Columns 17 through 32
  11   11   11   11   11   13   13   13   14   16   17   17   16   16   15   15
```

```
  12   12   12   12   14   15   15   16   17   17   16   16   16   15   14   13
   6    7    7    7    9    9    9    9    9   10   10   12   12   12   12   13
   5    5    8    8    9    9    9    9   11   11   11   11   13   13   13   15
   7    7    7    7    8    8    8    8   11   11   11   12   12   13   13   14
   8    8    8    8    8    9    9    9    9    9   12   12   12   13   13   15
  12   15   15   15   15   15   16   16   17   17   17   17   14   14   14   13
   6    6    6    6    7    7    7    7    7    8    8    8    9    9    9   11
 Columns 33 through 48
  14   13   12   12   12   12    9    9    9    9    9    8    8    8    8    8
  13   11    9    9    9    9    7    7    7    7    7    7    6    6    6    6
  13   13   14   14   15   15   16   16   17   17   17   17   17   16   16   16
  16   17   17   16   16   15   15   15   14   14   14   14   14   12   12   12
  14   14   14   14   16   16   16   17   17   17   17   16   16   15   15   15
  15   15   16   16   17   17   16   16   14   14   14   14   14   14   11   11
  13   11   11   11    8    8    8    7    7    7    7    7    6    6    6    6
  11   12   12   12   12   13   13   13   13   13   14   14   14   14   14   14
 Columns 49 through 64
   7    7    7    7    7    6    6    4    3    3    3    3    3    2    2    2
   6    6    5    5    5    4    4    4    4    4    4    4    3    3    3    3
  14   11   11   11   11    8    8    8    3    3    3    3    3    3    3    3
  12   12   10   10   10    7    7    7    6    6    6    6    1    1    1    1
  15   10   10   10   10   10   10    9    9    9    9    5    5    5    5    5
  11   11   10   10   10   10    4    4    4    4    3    3    3    3    2    2
   6    5    5    5    4    4    4    4    3    3    3    3    3    3    1    1
  15   15   15   15   16   16   16   17   10   10   10   10    1    1    1    1
 Columns 65 through 70
   2    2    2    2    2    2
   3    3    3    2    2    2
   2    2    2    2    2    2
   1    1    1    1    1    1
   1    1    1    1    1    1
   2    2    2    2    2    2
   1    1    1    1    1    1
   1    1    1    1    1    1
WT_DOW_J =
  23.4151  473.9390   17.0000
  25.8890  388.3469   17.0000
  29.4973  338.9291   17.0000
  30.1873  295.8768   17.0000
  35.2480  281.3333   17.0000
  35.8846  270.7768   17.0000
  36.9307  261.1324   17.0000
  40.4166  260.6821   17.0000

Elapsed time is 2561.965907 seconds.
```



An example of results of COIN at the side ratio 1:4:4 (1/9)

111 tasks

Arcus_111task_cycle17067_50:50:11

TS_task_minWS =

Columns 1 through 16

84	81	89	80	86	1	88	87	90	85	2	3	4	7	10	12
84	85	89	88	1	80	86	81	87	90	2	3	4	7	9	5
85	1	80	86	84	81	2	3	87	88	90	89	4	7	5	6
85	1	81	2	3	4	6	5	84	8	9	87	80	86	89	7
85	1	81	2	84	89	80	3	4	87	5	10	11	8	88	12
84	81	89	87	1	88	85	90	80	86	2	3	4	6	8	10
84	81	85	89	87	1	80	86	2	90	3	88	4	7	8	10
85	84	81	87	80	86	1	89	88	2	3	4	6	5	8	90
84	1	81	2	3	87	85	88	90	80	86	4	7	5	8	10
84	85	88	89	80	86	81	1	87	2	90	3	4	6	8	10
85	1	81	87	84	80	2	3	4	86	8	10	11	88	9	90
81	1	85	2	80	87	86	90	3	4	7	6	84	10	9	8
84	81	88	87	80	85	89	90	1	86	2	3	4	7	5	6
84	81	80	86	85	89	87	1	88	2	3	4	6	5	8	10
84	88	85	1	81	2	3	4	6	8	10	5	11	12	89	18
85	1	81	2	87	80	86	84	3	4	88	5	90	7	10	11
85	84	89	88	81	1	80	2	3	4	86	6	8	10	9	7
81	80	85	86	84	1	89	88	87	90	2	3	4	7	9	6
84	81	85	1	89	80	88	2	87	86	90	3	4	7	5	8
1	84	81	80	2	89	86	87	3	4	9	85	5	6	10	11
84	1	81	85	88	87	80	86	90	89	2	3	4	10	7	11

85	1	81	80	86	2	84	87	90	3	4	6	8	88	7	10
84	89	85	1	81	2	3	88	87	80	86	4	90	10	9	8
84	81	85	1	87	88	2	3	4	9	5	8	10	80	86	12
81	1	85	2	84	89	88	87	90	3	4	6	80	10	5	8
84	1	88	80	86	2	81	85	89	87	90	3	4	6	7	5
84	81	88	87	80	85	89	86	1	90	2	3	4	9	10	5
85	1	81	87	80	2	90	3	4	6	8	86	10	9	12	7
84	1	81	2	88	89	87	90	80	85	3	4	8	10	5	6
84	81	88	89	87	80	86	1	90	85	2	3	4	7	5	8
85	84	81	1	80	86	2	3	4	6	8	10	89	7	5	87
85	1	80	86	84	2	3	4	5	88	8	10	12	11	13	18
84	1	81	2	3	88	85	89	80	86	4	6	8	5	87	10
1	81	85	87	80	90	86	2	84	88	3	4	89	7	5	6
80	86	1	81	2	3	4	85	8	10	12	84	11	15	89	20
84	81	87	80	85	1	89	90	2	3	4	7	9	88	5	8
84	1	81	2	3	4	6	8	80	86	5	85	7	9	87	90
84	1	89	88	85	80	86	81	2	3	4	6	8	10	9	11
1	81	2	3	87	80	85	90	4	6	9	8	10	84	12	11
84	81	80	1	89	87	90	88	85	86	2	3	4	6	8	5
81	80	86	84	1	89	88	85	2	87	90	3	4	10	5	6
84	80	86	1	81	88	89	85	2	3	87	4	7	9	8	5
85	84	81	1	88	2	3	4	6	8	10	87	7	5	89	12
84	85	89	80	88	86	1	2	3	81	4	7	87	90	8	10
84	1	88	80	85	86	81	2	3	4	9	8	10	5	89	7
84	1	81	80	86	2	3	87	88	90	89	85	4	6	5	8
85	84	1	89	2	3	4	6	5	8	80	86	9	88	7	81
84	81	88	87	90	85	1	89	80	86	2	3	4	9	8	6
84	88	85	1	80	2	3	4	6	8	9	89	7	5	10	12

Columns 17 through 32

6	8	11	13	19	5	21	20	17	15	26	82	16	9	14	22
8	6	10	12	11	19	17	15	21	26	82	16	20	14	23	22
8	10	12	9	11	17	14	23	16	28	32	18	19	20	24	15
90	10	11	12	16	19	21	18	17	30	38	28	27	14	23	22
18	9	29	16	90	37	20	15	45	13	6	21	27	35	43	48
9	7	12	11	19	14	23	25	33	18	17	28	29	37	30	15
12	5	6	11	20	19	17	15	13	9	21	26	82	18	14	16
7	10	9	11	12	20	16	17	21	28	18	15	26	13	82	27
89	11	12	14	22	15	18	25	31	29	21	34	42	13	47	56
9	11	12	14	23	25	34	42	33	24	47	13	15	26	5	56
89	7	5	6	12	16	21	18	15	19	30	38	27	20	13	46
88	5	89	11	12	21	14	23	25	24	13	16	33	19	30	38
8	10	9	11	12	13	17	15	26	20	21	28	19	30	38	16
90	7	12	11	17	21	28	9	14	23	33	32	15	26	20	36
17	9	15	29	19	21	30	38	80	46	13	28	37	45	87	51
12	13	6	8	19	9	21	89	20	17	15	26	18	14	23	22
87	5	11	12	14	23	22	18	15	90	26	32	83	82	19	17
10	12	11	17	13	18	28	5	14	16	8	22	15	26	82	36
6	9	10	11	12	20	17	19	21	14	23	22	25	13	15	33
12	17	13	16	88	90	7	18	8	14	22	21	29	31	37	27
12	17	20	5	8	6	14	23	22	33	24	31	18	21	29	39

12	9	11	13	21	18	15	29	17	5	89	26	82	28	19	37
6	5	11	12	14	23	33	17	21	20	15	26	22	28	32	82
90	7	6	11	16	21	18	89	13	15	29	26	14	17	82	28
11	12	86	16	9	20	7	15	14	19	17	30	38	21	27	46
8	10	9	11	12	16	20	27	35	19	21	18	15	29	26	37
11	12	6	19	17	21	28	15	7	14	23	30	38	36	44	13
5	11	16	17	84	21	15	20	26	82	19	13	18	27	14	23
86	7	12	9	11	19	14	22	30	38	16	31	27	35	23	17
10	9	11	12	14	23	22	33	32	24	31	41	83	18	17	20
88	12	9	11	17	14	23	22	13	18	21	20	33	19	83	29
6	81	21	29	37	87	90	16	14	45	25	19	27	35	30	9
9	11	12	18	90	7	29	19	21	37	15	45	30	38	46	52
9	8	10	11	12	21	14	20	22	17	15	23	18	33	24	29
17	28	88	87	16	5	90	18	27	26	21	36	29	19	7	6
10	11	86	6	12	14	23	24	22	18	32	31	29	39	83	40
88	89	10	11	12	21	20	14	15	18	17	16	13	23	27	35
7	87	12	13	18	14	23	19	90	24	21	22	29	83	25	32
16	17	21	7	28	20	13	15	36	5	19	30	27	18	38	88
10	9	11	12	21	13	18	14	19	24	16	23	30	32	7	25
11	8	7	9	12	16	13	15	26	17	14	23	20	27	28	19
6	10	11	12	90	17	15	16	21	20	28	36	19	14	22	24
90	80	86	11	16	13	18	20	15	26	9	29	19	17	14	23
11	12	13	5	20	15	26	82	16	17	14	23	22	27	24	6
87	12	6	11	17	15	26	82	90	21	19	20	30	28	38	14
10	12	7	9	11	15	14	23	13	17	21	18	24	32	41	29
10	11	12	19	16	21	20	18	15	29	30	38	26	17	13	46
7	5	10	11	12	15	26	21	13	18	17	20	16	28	19	27
11	13	18	17	20	15	29	28	81	26	21	36	82	44	37	87

Columns 33 through 48

24	23	25	33	30	38	34	46	32	28	41	52	42	27	35	43
83	31	25	32	28	41	34	18	42	29	37	13	47	60	30	33
26	82	27	22	21	31	29	25	36	33	83	41	30	38	34	42
15	29	83	35	46	13	36	24	20	25	44	34	52	26	32	37
51	14	49	24	25	53	19	17	30	34	42	28	86	36	26	23
26	82	16	38	36	27	22	5	34	45	44	20	13	83	32	24
28	23	25	34	42	29	30	22	27	35	36	33	47	32	83	41
19	36	44	35	30	50	38	29	43	14	22	83	46	24	49	48
55	59	61	57	67	9	62	24	58	63	71	26	54	37	83	82
54	57	60	68	64	55	59	67	17	7	72	63	62	20	21	18
14	52	24	22	29	17	35	37	31	43	23	25	26	82	28	32
27	17	28	46	36	44	35	52	20	18	15	29	50	43	48	37
36	44	18	29	82	50	37	46	27	35	43	45	48	14	51	25
16	19	13	82	27	22	30	83	41	38	46	44	52	24	31	25
20	16	7	26	86	14	90	23	25	36	24	22	82	27	35	52
30	38	25	16	31	29	82	28	32	36	37	41	24	27	35	43
30	16	31	33	29	38	25	13	46	27	35	20	28	52	41	43
27	21	35	29	23	31	25	39	37	45	33	24	51	34	42	47
32	41	83	31	34	16	39	42	26	82	40	47	55	58	66	18
25	83	20	35	39	15	26	82	28	43	36	49	44	40	23	24
25	40	19	28	37	9	36	15	83	16	44	26	32	82	30	50

45	14	23	16	33	51	20	22	30	83	27	35	36	44	24	50
16	24	41	36	27	44	13	18	31	29	7	25	35	43	49	34
23	36	27	32	20	33	37	24	35	45	22	44	51	50	19	83
52	24	23	18	25	22	83	31	13	32	26	34	42	39	47	57
45	43	48	82	51	30	49	14	22	25	38	17	24	31	39	53
24	22	46	33	26	20	50	25	31	52	16	8	83	27	35	43
22	89	30	38	88	29	28	32	24	35	43	48	41	31	25	33
15	13	46	21	32	24	26	82	52	83	20	43	28	25	34	18
15	29	25	21	19	6	34	28	13	42	47	26	36	39	16	44
25	31	24	32	39	16	27	35	41	43	48	40	34	53	30	38
15	23	17	33	26	89	20	22	28	7	82	83	24	51	34	42
26	51	14	22	17	13	82	20	31	16	28	23	24	36	33	32
25	32	37	31	45	51	34	16	26	42	39	13	83	40	82	19
30	14	23	22	44	13	24	50	33	37	83	31	35	25	9	45
13	33	20	21	19	25	16	17	41	37	15	26	82	28	36	45
26	32	82	28	19	22	43	48	33	24	53	31	29	36	44	37
30	15	31	33	34	42	47	20	16	38	5	37	60	17	28	56
86	46	52	89	26	14	22	23	25	34	29	44	32	50	37	31
38	46	29	33	37	41	15	26	82	52	22	27	35	45	83	34
30	32	35	24	22	43	48	31	25	49	53	34	39	33	18	21
83	23	13	32	25	18	27	44	41	30	38	35	46	50	31	33
22	33	28	27	35	37	82	45	30	83	32	51	25	41	43	36
28	19	33	18	35	83	31	36	25	9	29	21	39	32	40	34
22	24	25	31	36	44	46	18	29	52	39	16	50	83	27	35
26	28	19	37	82	45	30	33	20	16	51	22	38	27	35	83
52	27	35	43	48	87	49	90	14	25	22	23	83	82	28	53
36	82	35	43	48	14	24	30	38	25	49	53	22	44	34	83
45	14	19	24	86	22	50	16	51	31	23	25	39	40	34	42

Columns 49 through 64

18	31	29	83	37	39	40	49	36	44	48	53	45	47	56	60
59	67	38	56	54	64	72	36	44	24	27	46	58	66	50	45
47	35	57	60	59	67	54	58	39	65	43	48	46	44	13	37
50	31	42	47	39	33	57	60	54	59	56	55	64	72	88	61
22	38	46	52	32	44	31	33	7	47	39	59	82	40	83	41
31	35	46	50	51	41	39	42	43	47	54	57	52	60	58	49
31	44	38	60	59	43	48	39	24	53	40	67	55	68	37	46
37	53	45	23	51	52	25	31	39	40	34	42	47	58	57	33
65	23	19	33	17	66	20	16	28	27	30	38	6	32	41	36
71	22	32	83	61	65	82	19	74	31	16	39	40	76	29	27
39	34	83	41	42	48	40	49	36	33	44	53	45	47	59	60
34	42	32	41	53	45	22	31	26	49	82	51	47	83	57	58
49	52	53	24	22	23	33	83	31	39	34	32	40	42	47	56
35	43	18	34	42	48	49	39	40	29	37	53	50	45	47	56
34	42	47	56	83	57	60	58	59	67	54	44	43	68	55	32
48	49	34	83	45	42	46	52	44	47	56	60	54	58	39	68
34	42	39	37	24	21	36	45	47	60	55	62	57	48	58	59
55	20	54	57	40	59	61	67	44	58	66	83	56	60	50	62
27	30	60	54	35	24	68	38	57	29	37	43	45	61	65	70
45	33	48	50	19	32	53	51	30	38	34	46	41	42	47	59
45	27	35	43	48	38	46	13	34	42	47	57	55	62	58	59

25	31	34	39	43	49	40	42	47	60	68	38	55	59	61	32
19	39	42	47	83	37	55	62	58	59	56	66	71	60	64	72
41	31	25	34	42	47	57	60	54	43	30	38	46	52	39	55
40	56	41	55	59	67	29	62	71	64	65	33	28	58	35	37
34	28	23	83	40	32	33	36	44	42	47	13	56	41	54	58
82	39	34	40	49	42	47	18	55	32	56	54	48	59	62	58
53	39	37	40	34	83	46	52	49	36	44	45	51	42	47	55
39	29	41	48	40	42	53	33	47	54	57	36	60	59	67	37
37	27	35	43	50	54	59	56	55	62	63	71	45	30	61	70
37	15	26	82	45	46	52	90	28	42	47	51	55	59	61	54
31	38	32	47	60	55	39	54	57	59	43	48	65	68	58	46
83	25	41	39	44	27	35	50	34	42	40	47	55	62	71	60
41	47	60	28	30	58	68	55	56	64	63	72	62	38	61	57
43	38	34	42	47	56	55	62	63	58	32	66	48	59	67	49
27	35	43	48	53	51	34	42	30	47	60	56	54	57	65	68
83	41	49	45	30	38	25	51	34	46	52	50	39	42	47	54
54	27	55	39	35	64	58	46	68	40	36	44	45	72	61	62
41	45	24	39	42	47	83	59	67	54	60	56	64	35	72	33
42	20	17	43	47	56	60	54	68	31	55	64	51	48	61	70
83	29	38	36	46	42	52	41	37	47	60	59	67	56	54	57
39	29	26	40	34	37	42	52	82	47	57	65	43	60	55	59
24	31	48	38	34	42	39	40	44	49	47	56	55	21	63	46
30	43	38	48	53	46	52	37	45	51	42	47	57	56	54	64
43	48	23	13	53	34	40	33	42	47	60	49	57	68	32	37
31	25	34	42	47	56	54	64	60	39	36	40	44	43	72	58
36	44	37	50	32	31	39	41	40	34	33	24	45	42	47	55
50	46	52	29	37	31	23	42	47	56	54	59	67	32	39	41
47	90	55	59	67	54	62	57	30	83	27	65	58	71	32	41

Columns 65 through 80

55	64	59	67	51	50	62	57	65	68	54	63	72	61	58	71
68	55	39	57	61	35	52	69	70	74	65	63	43	73	51	48
50	55	62	71	66	53	40	49	56	45	68	63	61	70	52	73
68	63	45	62	43	74	41	67	82	51	70	76	71	69	49	48
50	57	65	58	56	54	64	60	67	66	55	62	63	71	72	68
68	48	59	21	56	67	64	72	65	66	55	74	53	61	70	62
57	45	50	61	51	52	62	56	63	71	65	58	66	49	64	54
60	56	66	64	54	59	32	41	65	68	55	72	63	67	62	61
39	40	35	45	46	52	44	43	50	70	60	49	68	69	51	48
35	28	30	38	36	37	43	41	70	49	48	44	69	73	58	53
51	55	62	56	67	61	50	54	71	68	70	73	69	75	78	77
65	66	56	54	59	64	39	55	72	62	71	67	61	69	60	40
59	67	54	58	66	60	57	65	41	55	64	62	68	63	71	61
54	51	55	59	67	64	57	60	72	58	66	65	61	62	69	63
64	65	48	72	33	66	31	53	39	41	63	49	40	61	62	74
33	53	55	64	66	51	57	65	62	61	71	50	69	40	63	59
67	53	40	44	65	71	68	61	54	49	50	70	69	73	66	56
71	64	32	43	41	19	70	68	69	30	49	38	63	48	46	52
73	46	52	28	75	36	48	51	59	62	44	79	49	53	50	71
60	55	67	52	56	62	58	66	57	54	65	71	63	68	61	70
52	51	67	53	61	65	54	49	66	41	70	60	56	73	75	63

58	57	65	48	41	66	53	56	54	70	46	52	73	67	64	72
68	40	48	53	74	50	30	54	63	45	76	67	61	69	38	46
62	58	66	48	61	71	68	69	70	73	65	49	40	53	75	79
45	72	61	82	63	43	48	66	53	51	36	60	54	70	73	68
57	65	66	50	64	55	72	59	46	67	74	63	52	60	61	62
71	57	65	63	66	41	61	67	64	60	69	70	73	68	72	53
62	50	59	67	61	54	70	56	58	57	60	64	71	65	73	68
49	68	55	61	62	58	66	45	69	51	65	70	73	75	77	78
67	64	73	75	40	49	48	79	53	38	72	51	74	58	69	60
57	65	62	56	67	64	72	69	49	70	71	60	74	36	73	63
67	62	71	40	56	52	53	63	66	41	36	44	49	61	70	69
43	48	56	54	68	61	58	66	69	70	59	67	64	72	74	57
66	27	74	54	35	70	69	36	65	71	43	44	50	59	49	67
71	60	53	57	46	52	41	39	68	61	64	54	69	51	40	72
49	38	59	64	55	72	63	67	61	44	69	70	74	58	46	50
58	66	59	56	55	64	57	60	40	68	63	67	61	72	70	74
71	50	74	52	57	65	43	48	41	49	76	69	70	51	66	73
55	62	82	51	58	66	74	57	65	71	68	40	61	76	69	70
39	59	62	71	53	58	66	69	49	40	28	72	57	73	67	75
55	62	63	82	40	65	68	58	44	71	64	72	61	50	70	74
61	62	58	66	54	68	48	69	70	71	67	53	73	49	56	63
52	53	57	65	58	66	59	61	50	60	54	68	70	62	71	69
60	55	62	72	59	67	58	68	74	65	63	41	66	71	76	49
58	66	55	59	56	67	64	54	63	45	72	65	62	41	61	70
59	67	46	68	74	66	55	76	52	57	48	50	61	65	49	70
62	71	57	60	56	64	58	66	65	68	61	54	70	59	63	67
33	57	60	64	55	62	63	71	72	40	68	61	69	70	74	45
61	33	70	56	73	63	66	60	69	64	68	75	38	77	72	35

Columns 81 through 96

69	70	73	74	66	75	91	78	93	94	76	92	79	95	97	77
49	40	62	71	76	75	78	77	79	91	93	94	53	92	95	97
69	64	72	75	78	77	91	79	74	94	76	93	51	92	95	96
58	66	65	73	53	40	75	78	77	79	91	94	93	92	95	100
74	61	69	70	76	73	75	91	93	94	77	79	78	92	95	98
71	76	69	63	73	75	91	40	94	79	92	78	77	93	95	97
70	69	73	72	75	91	78	93	92	77	74	94	79	76	95	97
70	73	71	74	75	91	92	93	94	76	69	95	97	78	77	100
64	53	72	73	74	76	75	91	78	93	94	77	92	95	99	97
46	52	50	75	79	91	93	94	77	66	78	45	92	95	98	99
58	66	64	72	63	79	74	57	65	76	91	92	93	94	95	97
74	76	63	68	70	73	75	91	92	77	94	78	93	79	95	100
72	74	76	69	70	73	75	91	92	79	77	93	94	95	98	99
71	68	74	70	73	75	79	91	78	93	94	76	92	77	95	104
76	69	70	73	75	79	50	78	77	71	91	92	93	94	95	97
67	72	74	76	70	73	75	78	77	79	91	93	94	92	95	97
64	51	72	75	91	78	93	94	79	74	92	77	95	63	97	100
65	73	75	53	78	77	79	72	91	94	92	74	93	76	95	97
67	56	69	64	78	77	72	91	94	63	74	92	93	95	100	96
69	64	73	75	78	72	79	74	91	94	92	77	93	76	95	97
71	68	64	79	72	69	78	77	91	93	94	74	76	92	95	99

63	62	75	69	71	74	91	93	94	92	78	77	95	97	100	96
52	51	70	57	65	73	75	78	77	91	93	94	92	79	95	98
78	77	56	63	59	64	72	91	67	93	74	94	76	92	95	97
49	69	44	74	76	50	75	91	94	77	93	78	79	92	95	97
69	70	76	73	68	75	71	79	91	78	93	94	77	92	95	99
74	29	75	77	79	37	45	91	78	93	94	76	92	95	98	103
69	63	75	66	77	79	72	91	93	94	74	78	76	92	95	97
79	71	44	56	50	63	64	72	91	93	92	74	76	94	95	97
66	68	57	76	78	77	91	65	92	46	52	94	82	93	95	98
75	78	77	79	68	76	58	66	91	44	93	94	92	50	95	97
73	75	78	64	79	72	91	94	74	92	76	93	50	77	95	100
76	53	65	49	73	75	91	93	78	79	94	77	92	63	95	98
46	52	73	48	75	53	76	78	79	91	94	77	92	93	95	97
82	70	74	65	73	75	79	91	93	94	77	78	92	95	98	97
76	73	62	75	79	77	78	66	52	71	91	94	92	93	95	97
65	73	69	75	79	62	78	77	71	76	91	94	92	93	95	97
63	75	26	77	82	78	59	91	67	79	92	53	93	94	95	100
73	75	63	79	77	78	43	48	49	91	92	93	94	53	95	97
63	79	77	36	78	91	65	92	93	94	74	95	76	98	99	44
45	66	69	73	75	76	91	94	77	93	51	78	92	79	95	98
64	45	75	79	78	72	51	74	76	91	94	93	77	92	95	98
73	75	79	64	77	78	72	91	93	94	74	92	95	67	96	104
61	69	70	73	75	78	77	44	79	91	94	92	50	93	95	100
73	74	76	75	51	79	69	78	77	71	91	92	93	94	95	104
69	63	73	75	78	77	62	79	53	71	91	93	94	92	95	98
73	75	79	72	51	74	76	69	77	91	78	93	94	92	95	98
58	66	73	75	91	78	93	94	76	65	77	92	79	95	98	103
43	48	49	78	53	74	91	76	93	94	46	92	79	95	96	99

Columns 97 through 111

96	99	104	98	101	105	103	108	110	102	100	106	107	109	111
104	99	96	101	98	105	103	100	108	110	102	106	109	107	111
104	99	97	100	98	101	105	103	108	102	106	109	107	110	111
96	99	101	98	97	104	102	106	109	103	108	110	105	107	111
99	97	96	101	105	103	108	110	104	102	106	109	107	100	111
100	96	98	99	102	106	109	101	104	103	108	105	107	110	111
102	100	98	103	108	104	96	110	99	106	109	107	101	105	111
102	104	106	79	96	99	101	98	105	103	108	110	109	107	111
104	98	79	102	100	96	101	105	103	108	106	109	107	110	111
96	104	103	97	100	102	106	109	107	51	108	110	101	105	111
100	102	106	98	99	104	103	107	96	108	110	109	101	105	111
98	104	96	99	101	97	102	106	109	103	107	108	110	105	111
97	104	103	96	101	105	78	100	102	106	107	108	110	109	111
98	99	97	100	103	108	110	96	101	105	102	106	107	109	111
104	98	99	102	106	100	109	96	101	105	103	108	110	107	111
104	98	100	96	102	103	99	107	101	105	108	106	109	110	111
76	102	106	109	96	101	98	105	104	103	108	110	107	99	111
100	96	99	102	106	101	105	104	98	103	109	107	108	110	111
97	76	99	102	106	98	103	108	101	110	105	104	109	107	111
99	96	104	102	101	98	100	105	106	103	109	108	110	107	111
97	100	102	104	98	103	96	107	101	106	109	105	108	110	111

102	106	109	99	101	79	98	105	76	104	103	108	110	107	111
100	96	99	97	101	105	103	108	110	102	107	106	109	104	111
100	102	106	109	104	96	99	98	103	108	110	107	101	105	111
104	96	99	101	102	106	105	109	100	98	103	108	107	110	111
100	96	104	98	103	101	105	108	110	97	102	106	109	107	111
96	101	51	105	108	99	97	100	104	110	102	106	107	109	111
100	96	101	104	98	103	99	108	102	106	107	105	110	109	111
104	98	100	103	108	99	102	106	96	107	101	110	105	109	111
99	104	97	100	102	106	109	103	108	107	110	96	101	105	111
100	96	102	106	104	98	99	101	105	103	108	107	109	110	111
96	104	101	98	99	97	102	106	109	103	108	105	107	110	111
100	104	97	102	103	108	106	96	101	110	107	105	99	109	111
100	102	106	109	99	98	104	96	101	105	103	108	107	110	111
102	104	96	101	105	76	106	109	103	99	107	100	108	110	111
96	102	104	99	98	100	101	105	103	106	109	107	108	110	111
104	96	102	106	109	98	99	101	105	103	108	110	107	100	111
98	99	104	103	96	101	108	110	105	97	102	106	109	107	111
100	96	99	102	106	104	98	109	103	108	110	107	101	105	111
103	108	96	50	97	104	100	101	110	105	102	106	107	109	111
99	100	96	104	97	102	106	103	109	107	101	108	105	110	111
103	96	97	100	101	104	102	106	99	105	109	108	110	107	111
97	100	76	98	103	108	99	102	110	101	105	106	109	107	111
104	96	99	101	97	102	106	98	103	108	105	110	107	109	111
98	99	97	100	96	101	103	108	110	105	102	106	109	107	111
99	96	100	97	103	108	101	104	102	106	107	105	109	110	111
99	97	104	103	108	110	102	107	96	101	100	105	106	109	111
99	97	100	102	106	104	51	109	107	108	110	96	101	105	111
52	104	98	100	97	101	102	106	109	105	103	108	107	110	111

Station =

Columns 1 through 16

1	1	1	1	1	1	2	2	2	2	2	2	2	2	2	3
1	1	1	1	1	1	1	2	2	2	2	3	3	3	3	3
1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2
1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2
1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2
1	1	1	1	2	2	2	2	2	2	2	3	3	3	4	4
1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2
1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2
1	1	1	1	1	1	2	2	2	2	2	2	2	2	2	3
1	1	1	1	1	1	1	2	2	2	2	2	3	3	3	3
1	1	1	1	1	1	1	1	1	1	1	1	2	2	2	2
1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2
1	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2
1	1	1	1	1	2	2	2	2	2	2	2	3	3	3	3
1	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2
1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	3
1	1	1	1	2	2	2	2	2	3	3	3	3	3	3	3

1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2
1	1	1	1	1	2	2	2	2	2	2	2	3	3	3	4
1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2
1	1	1	1	1	1	1	1	1	1	1	3	3	3	3	4
1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2
1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2
1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2
1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	1	2	2	2	2	2	3	3	3
1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	3
1	1	1	1	1	1	1	1	2	2	2	2	2	2	3	3
1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	2	2	2	2	2	2	2	3	3
1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2
1	1	1	1	1	1	2	2	2	2	2	2	2	2	2	3
1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	3
1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2

Columns 17 through 32

3	4	4	4	4	5	5	5	5	5	6	6	6	7	7	7
3	3	3	4	4	4	4	4	5	5	5	5	5	6	7	7
2	2	3	3	3	3	4	4	4	4	4	4	4	4	4	5
3	3	3	3	3	3	3	3	3	3	3	3	3	4	4	4
2	2	2	2	2	2	2	3	3	4	4	4	4	4	4	4
3	3	3	3	3	4	4	4	4	4	4	4	4	4	4	5
2	3	3	3	3	3	3	4	4	5	6	6	6	6	7	7
4	4	4	5	5	5	5	5	6	6	6	6	6	6	7	7
2	3	3	4	4	4	4	4	5	5	5	5	6	6	7	7
3	3	3	3	4	4	4	5	5	5	6	6	6	6	6	7
3	3	3	3	3	3	3	3	4	4	4	4	4	4	5	5
3	3	3	4	5	5	6	6	6	6	6	6	6	6	6	6
2	2	3	4	4	4	4	4	5	5	5	5	5	5	5	5
2	2	2	3	3	3	3	3	4	4	5	5	5	5	6	6
2	3	3	3	3	3	3	3	4	4	4	4	4	4	5	5
2	2	3	3	3	3	3	4	4	4	4	5	5	5	5	5
3	3	4	4	5	5	5	5	5	6	6	6	6	6	6	6
2	2	3	3	3	3	3	3	4	4	4	4	4	5	5	5
3	3	3	3	4	4	4	4	4	4	5	5	5	5	6	6
3	3	3	3	3	3	3	3	4	4	4	5	5	5	5	5

2	2	2	2	3	3	4	4	4	5	5	5	5	5	5	5
4	4	4	4	4	4	5	5	5	5	6	6	6	6	6	6
3	3	3	3	3	4	4	4	5	5	5	5	5	5	5	5
2	3	3	3	3	4	4	5	5	5	5	5	5	5	5	5
3	3	3	3	3	3	3	3	4	4	4	4	4	5	5	5
4	4	5	5	5	5	5	5	5	5	6	6	6	6	6	6
2	2	2	3	3	4	4	4	4	4	5	5	5	5	5	5
2	2	3	3	3	4	4	4	5	5	5	6	6	6	6	7
3	3	3	3	3	3	4	4	4	4	4	5	5	5	5	5
2	3	3	3	3	4	4	4	4	4	4	5	5	5	5	5
2	2	3	3	3	3	4	4	5	5	5	5	5	5	5	5
3	4	4	4	4	4	4	4	5	5	5	5	5	5	5	6
3	3	3	3	3	3	3	3	4	4	5	5	5	5	5	5
2	3	3	3	3	3	4	4	4	4	5	5	5	5	5	5
3	3	3	4	4	4	4	4	4	5	5	5	5	5	5	5
2	3	3	3	3	3	4	4	4	4	5	5	5	6	6	6
3	3	3	4	4	4	4	4	5	5	5	5	5	6	6	6
3	3	3	3	3	3	4	4	4	4	5	5	5	5	5	6
3	3	3	3	3	3	3	4	4	4	4	4	4	4	4	5
3	3	3	4	4	4	4	4	4	4	4	5	5	5	5	6
2	3	3	3	3	3	3	5	5	5	5	6	6	6	6	6
3	3	3	3	3	3	3	3	5	5	5	5	5	5	5	5
3	3	3	5	5	5	5	5	5	5	5	5	5	5	5	6
3	3	3	3	3	3	4	4	4	4	5	5	5	5	5	5
2	3	3	4	4	4	4	5	5	5	5	5	5	5	5	5
3	3	3	3	3	3	4	4	4	4	5	5	5	5	6	6
3	3	3	3	3	5	5	5	5	5	5	5	5	5	6	6
3	3	3	4	4	4	4	4	5	5	5	5	5	5	5	5
3	3	3	3	3	3	3	3	3	4	4	4	5	5	5	5

Columns 33 through 48

7	8	8	8	8	8	9	9	9	9	9	9	10	10	10	10
7	8	8	8	8	8	9	9	9	9	9	9	10	10	10	10
5	5	5	5	5	6	6	6	6	6	7	7	7	7	7	8
4	4	5	5	5	5	5	5	5	6	6	6	6	6	7	7
4	5	5	5	6	6	6	6	6	6	6	6	7	7	8	9
5	5	5	5	5	5	5	6	6	6	6	6	7	7	7	7
7	8	8	8	9	9	9	9	9	9	9	9	10	10	10	10
7	7	7	7	7	7	7	7	7	7	7	8	8	8	8	8
7	8	8	8	9	9	9	9	9	9	10	10	10	10	10	10
7	7	8	8	9	9	9	9	9	9	10	10	10	10	10	10
5	5	5	5	5	5	5	5	6	6	6	6	7	7	7	7
6	7	7	7	7	7	7	7	7	7	8	8	8	8	8	8
5	5	5	5	5	5	5	5	5	5	5	5	6	6	6	6
6	6	6	6	6	7	7	7	7	7	7	7	7	7	7	8
5	5	5	5	5	5	6	7	7	7	7	7	8	8	8	8
5	5	6	6	6	6	7	7	7	7	7	7	7	7	7	7
6	6	7	7	7	7	7	8	8	8	8	8	8	8	8	8
5	5	5	5	5	6	6	6	6	6	7	7	7	7	7	8
6	7	7	7	7	7	8	9	9	9	10	10	10	10	10	10
5	6	6	6	7	7	7	7	7	7	7	7	7	8	8	8

6	6	6	6	6	6	6	7	7	7	7	8	8	8	8	8
6	6	7	7	7	7	7	7	8	8	8	8	8	8	8	8
5	5	6	6	6	6	6	6	6	6	6	7	7	7	7	7
6	6	6	6	6	7	7	7	7	7	7	7	7	7	7	7
5	5	5	5	5	5	6	6	6	6	7	7	7	8	9	9
6	6	6	6	7	7	7	7	7	7	7	7	7	8	8	8
5	5	5	6	6	6	6	6	6	6	6	7	7	7	7	7
7	7	7	7	7	7	7	8	8	8	8	8	8	9	10	10
5	6	6	6	6	6	7	7	7	7	7	7	7	8	8	8
6	6	6	6	6	6	7	7	7	7	8	8	8	8	8	8
6	6	6	6	7	7	7	7	7	7	7	8	8	8	8	8
6	6	6	6	7	7	7	7	7	7	7	8	8	8	9	9
5	5	6	6	6	6	6	6	6	6	6	7	7	7	7	7
6	6	6	6	6	6	6	6	7	7	8	8	8	8	8	8
5	6	6	6	6	7	7	7	7	7	8	8	8	8	9	9
6	7	7	7	7	7	7	7	8	8	8	8	8	8	8	8
6	6	7	7	7	7	7	7	7	7	7	8	8	8	8	8
6	6	7	7	7	9	9	9	9	9	9	9	10	10	10	10
5	5	5	5	6	6	6	6	6	7	7	7	7	7	7	8
6	6	6	6	6	6	6	7	7	7	7	7	7	7	7	8
6	6	6	6	6	6	6	7	7	7	7	7	8	8	8	9
6	6	6	6	7	7	7	7	7	7	7	7	7	7	8	8
6	7	7	7	7	7	7	7	7	8	8	8	8	8	8	8
5	5	5	5	5	6	6	6	6	6	6	7	7	7	7	7
5	5	7	7	7	7	7	7	7	7	8	8	8	8	8	8
6	6	6	6	7	7	7	7	7	7	7	7	7	7	7	8
6	6	6	6	6	6	6	6	7	7	7	8	8	8	8	8
5	5	5	5	5	6	6	6	6	6	6	6	6	6	9	9
5	5	5	5	5	5	5	5	5	6	6	6	6	6	7	7

Columns 49 through 64

10	10	10	10	10	10	10	9	9	9	9	9	9	9	8	8
10	10	10	10	9	9	8	8	8	8	8	8	8	7	7	7
8	8	9	9	9	9	9	10	10	10	10	10	10	10	10	10
7	8	8	9	9	10	10	10	10	10	10	10	10	9	9	9
9	9	9	9	9	9	10	10	10	10	10	10	9	9	9	9
8	8	8	8	8	9	9	10	10	10	10	10	10	10	9	9
10	10	10	10	9	9	9	9	8	8	8	8	8	8	7	7
8	8	8	9	9	9	9	9	10	10	10	10	9	9	8	8
10	10	10	10	10	9	9	9	9	9	9	9	9	9	8	8
10	10	9	9	9	9	9	9	9	8	8	8	8	8	7	7
8	8	8	9	9	9	9	9	9	10	10	10	10	10	10	10
9	10	10	10	10	10	10	10	10	10	10	9	9	9	9	9
6	6	6	6	6	7	8	8	8	9	9	9	10	10	10	10
8	8	8	9	9	9	9	10	10	10	10	10	10	10	10	10
8	8	9	9	9	9	10	10	10	10	10	10	10	10	10	10
7	7	8	8	8	9	9	9	9	9	10	10	10	10	10	10
9	9	10	10	10	10	10	10	10	10	9	9	9	8	8	8
8	8	8	8	8	9	9	9	9	9	10	10	10	10	10	10
10	10	10	10	10	10	10	10	10	9	9	9	9	9	9	9
8	8	8	8	8	8	8	8	8	8	9	9	9	10	10	10

8	8	8	8	8	8	8	9	9	9	10	10	10	10	10	10
8	8	9	9	9	9	9	10	10	10	10	10	10	10	9	9
7	8	8	9	9	9	9	10	10	10	10	10	10	9	9	9
8	8	8	9	9	10	10	10	10	10	10	10	10	10	10	9
10	10	10	10	10	10	10	10	9	9	9	9	8	8	8	8
8	8	9	9	10	10	10	10	10	10	10	9	9	9	9	8
7	8	9	9	9	9	10	10	10	10	10	10	10	10	9	8
10	10	10	10	10	9	9	9	9	9	9	9	9	9	9	8
9	9	9	9	9	10	10	10	10	10	10	9	9	9	9	8
8	8	8	8	8	9	9	9	9	9	10	10	10	10	10	10
8	8	9	9	9	9	9	9	9	10	10	10	10	10	10	9
9	9	10	10	10	10	10	10	9	9	8	8	8	8	8	8
7	8	8	8	8	8	8	8	9	9	10	10	10	10	10	10
9	9	9	9	9	9	9	9	10	10	10	10	10	9	9	8
9	9	9	9	10	10	10	10	10	9	9	8	8	8	8	8
8	8	8	8	8	8	9	9	9	10	10	10	10	10	10	10
8	9	9	9	9	9	9	9	9	9	9	9	10	10	10	9
10	10	10	10	10	10	9	9	9	8	8	8	8	8	8	8
8	8	8	9	9	9	10	10	10	10	10	10	10	8	8	8
8	8	8	8	10	10	10	10	10	10	9	9	9	9	9	9
9	9	9	9	9	10	10	10	10	10	10	9	9	9	9	8
8	8	8	9	9	9	9	9	10	10	10	10	10	10	10	10
8	8	8	8	9	9	10	10	10	10	10	10	9	9	9	8
7	7	7	7	7	7	8	8	8	8	9	9	9	9	10	10
8	8	9	9	9	9	10	10	10	10	10	9	9	9	9	9
8	8	9	9	9	10	10	10	10	10	8	8	8	8	8	8
8	8	8	8	8	9	9	9	9	10	10	10	10	10	10	9
9	9	9	9	9	10	10	10	10	9	9	9	8	8	8	8
7	7	7	8	8	8	10	10	10	10	10	10	10	10	10	9

Columns 65 through 80

7	7	7	7	7	7	7	6	6	6	6	6	6	6	5	5
7	7	7	7	6	6	6	6	6	6	6	6	6	6	5	5
10	10	10	10	9	9	9	9	9	8	8	8	7	7	6	6
8	8	8	8	7	7	7	7	7	7	7	7	6	6	6	6
9	9	8	8	8	8	8	8	7	7	7	7	7	7	7	6
9	9	9	9	8	8	8	8	8	8	8	8	8	7	7	7
7	7	7	7	7	7	7	7	6	6	6	6	5	5	5	5
8	8	8	8	7	7	7	7	7	7	6	6	6	6	5	5
8	7	7	7	7	7	7	7	7	7	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7	7	6	6	6	6	5
10	10	10	9	9	8	8	8	7	7	7	7	6	6	6	6
9	8	8	8	8	8	7	7	7	7	7	6	6	5	5	5
9	9	9	8	8	8	8	8	8	7	7	7	7	7	6	6
9	9	9	9	9	8	8	8	8	8	8	7	7	7	6	6
10	10	9	9	8	8	8	7	7	7	6	6	6	6	6	6
10	9	9	9	9	8	8	8	8	8	7	7	7	7	7	6
8	8	8	8	8	8	7	7	7	7	7	7	6	6	6	6
10	9	9	9	9	8	8	8	7	7	7	7	7	6	6	6
8	8	8	8	8	8	8	8	8	8	7	7	7	7	7	7
10	10	10	9	9	9	8	8	8	8	7	7	6	6	6	6

10	10	10	9	9	9	9	8	8	8	8	8	7	7	7	7
8	8	8	8	8	7	7	7	7	7	6	6	6	6	6	6
9	8	8	8	8	8	8	8	7	7	7	6	6	6	6	6
9	9	8	8	8	8	7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	7	7	7	7	7	7	7	6	6	6	6
8	8	7	7	7	7	7	7	6	6	6	6	6	6	5	5
8	8	8	8	7	7	7	7	7	6	6	6	5	5	5	5
8	8	8	8	8	7	7	7	6	6	6	6	6	5	5	5
8	8	8	8	8	7	7	7	7	7	7	7	7	7	6	6
10	10	9	9	8	8	8	8	7	7	7	7	7	7	6	6
9	9	9	9	8	8	8	7	7	7	7	6	6	6	6	6
8	8	8	8	7	7	7	7	7	7	6	6	6	6	6	5
9	9	9	9	8	8	7	7	7	7	6	6	6	6	5	5
8	8	8	8	7	7	7	7	7	7	6	6	6	6	6	6
8	7	7	7	7	7	7	7	7	6	6	6	6	6	6	5
9	9	9	9	8	8	8	7	7	7	7	7	6	6	6	6
9	9	8	8	8	8	8	7	7	7	7	7	7	7	6	6
8	8	8	8	8	7	7	7	7	7	7	6	6	6	6	6
8	8	8	8	8	7	7	7	7	7	7	6	6	5	5	5
9	9	8	7	7	7	7	7	7	7	7	7	6	6	5	5
8	8	8	7	7	7	7	7	6	6	6	6	5	5	5	5
10	9	8	8	8	8	7	7	7	7	7	6	6	6	6	5
8	8	8	7	7	7	7	7	7	7	7	6	6	6	6	5
10	10	10	10	10	8	8	8	8	8	8	8	8	8	6	6
9	8	8	8	8	7	7	7	7	6	6	6	6	6	6	6
8	7	7	7	7	7	7	7	7	7	7	7	7	7	6	6
9	8	8	8	7	7	7	6	6	6	6	6	6	6	5	5
8	8	8	7	7	7	7	7	7	6	6	6	6	6	5	5
9	9	9	9	9	8	8	8	8	8	5	5	5	5	5	5

Columns 81 through 96

5	5	4	4	4	4	4	3	3	3	3	3	3	3	2	2
5	5	5	5	4	4	4	4	3	3	3	2	2	2	2	2
6	6	6	5	5	5	4	4	4	3	3	3	3	3	3	2
6	5	5	5	5	5	5	5	5	4	4	3	3	3	3	2
6	5	5	5	5	5	5	4	4	4	4	3	3	3	3	3
6	6	6	6	5	5	5	4	4	4	4	4	3	3	3	2
5	5	5	5	4	4	4	4	4	4	3	3	3	3	3	2
5	5	4	4	4	4	3	3	3	3	3	3	2	2	2	2
6	6	6	5	5	5	4	4	3	3	3	3	3	3	2	2
5	5	5	5	5	5	4	4	4	3	3	3	3	3	2	2
5	5	5	5	5	5	4	4	4	4	4	3	3	3	3	2
5	5	5	4	4	4	4	4	3	3	3	3	3	2	2	2
6	5	5	5	5	4	4	4	3	3	3	3	3	3	3	3
6	6	6	6	6	5	5	5	4	4	4	4	4	4	3	3
6	5	5	5	5	5	5	5	4	4	4	3	3	3	3	2
6	6	6	6	5	5	5	4	4	4	4	3	3	3	3	2
5	5	5	5	4	4	4	4	4	4	4	3	3	2	2	2
6	6	6	5	5	5	5	4	4	3	3	3	3	3	3	2
6	6	6	6	5	5	5	4	4	4	3	3	3	3	2	2
6	6	5	5	5	5	4	4	4	4	4	3	3	3	2	2

6	6	6	6	5	5	5	4	4	3	3	3	3	3	3	2
5	5	5	5	5	4	4	4	3	3	3	3	3	2	2	2
6	6	6	6	5	5	5	4	4	4	3	3	3	3	3	2
6	6	6	5	5	4	4	4	4	3	3	3	3	3	3	2
5	5	5	5	5	4	4	4	4	4	4	4	3	3	3	2
5	5	4	4	4	4	4	4	3	3	3	3	3	2	2	2
5	5	5	5	4	4	4	4	3	3	3	3	3	3	3	3
5	5	5	5	5	4	4	4	4	4	3	3	3	3	3	2
6	6	5	5	5	5	4	4	4	4	4	4	3	3	3	2
6	6	5	5	5	5	4	4	3	3	3	3	3	3	3	2
5	5	5	5	5	4	4	4	4	3	3	3	3	3	3	3
5	5	5	5	5	4	4	4	4	4	3	3	3	3	3	2
5	5	5	5	5	5	4	4	4	4	4	4	3	3	3	2
5	5	5	5	5	5	5	4	4	4	4	4	4	4	3	2
5	5	5	5	4	4	4	4	4	4	4	3	3	3	3	3
6	5	5	5	5	5	4	4	4	4	4	3	3	3	3	2
6	6	5	5	5	5	5	5	4	4	3	3	3	3	3	3
6	5	5	5	4	4	4	4	4	3	3	3	3	3	3	2
5	5	5	4	4	4	4	4	4	4	3	3	3	3	3	2
5	5	5	4	4	4	4	3	3	3	3	3	3	2	2	2
5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	2
5	5	5	5	5	4	4	4	4	4	4	4	4	3	3	3
5	5	4	4	4	4	4	4	3	3	3	3	3	3	3	2
6	5	5	5	4	4	4	4	4	4	3	3	3	3	3	2
5	5	5	5	5	5	4	4	4	4	3	3	3	3	3	2
6	6	6	6	5	5	5	5	4	4	4	3	3	3	3	2
5	5	4	4	4	4	4	4	4	4	3	3	3	3	3	2
5	5	5	5	5	5	4	4	4	3	3	3	3	3	2	2
5	5	5	5	4	4	4	4	4	3	3	3	3	3	2	2

Columns 97 through 111

2	2	2	2	2	1	1	1	1	1	1	1	1	1	1
2	2	2	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	1	1	1	1	1	1	1	1	1	1
2	2	1	1	1	1	1	1	1	1	1	1	1	1	1
3	3	3	2	2	2	2	2	2	2	2	2	2	2	1
2	2	2	2	2	2	2	2	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	1	1	1	1	1	1	1	1
2	2	2	2	2	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	1	1	1	1	1	1	1	1	1	1
2	2	2	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	1	1	1	1	1	1
3	3	2	2	2	2	2	2	2	1	1	1	1	1	1
2	2	2	2	2	2	1	1	1	1	1	1	1	1	1
2	2	2	2	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	1	1	1	1	1	1	1	1
2	2	2	1	1	1	1	1	1	1	1	1	1	1	1
2	2	1	1	1	1	1	1	1	1	1	1	1	1	1

2	2	2	2	2	2	2	1	1	1	1	1	1	1	1
2	2	2	2	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	1	1	1	1	1	1	1	1
3	3	2	2	2	2	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	1	1	1	1	1	1
2	2	2	2	1	1	1	1	1	1	1	1	1	1	1
3	2	2	2	2	2	2	2	2	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	1	1	1	1	1	1
2	2	2	2	2	1	1	1	1	1	1	1	1	1	1
3	3	3	2	2	2	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	1	1	1	1	1	1
2	2	2	2	2	1	1	1	1	1	1	1	1	1	1
2	2	2	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	1	1	1	1	1	1	1	1
2	2	2	2	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	1	1	1	1	1	1
2	2	2	2	2	2	2	1	1	1	1	1	1	1	1
2	2	2	2	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	1	1	1	1	1	1
2	2	2	2	2	2	1	1	1	1	1	1	1	1	1
2	2	2	2	1	1	1	1	1	1	1	1	1	1	1

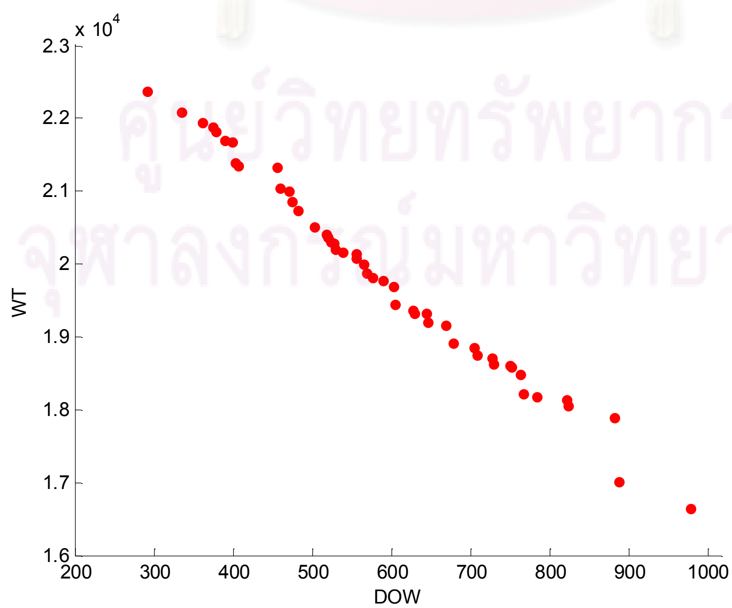
WT_DOW_J =

1.0e+004 *

0.0292	2.2373	0.0010
0.0335	2.2095	0.0010
0.0360	2.1955	0.0010
0.0375	2.1884	0.0010
0.0377	2.1822	0.0010
0.0389	2.1710	0.0010
0.0399	2.1678	0.0010
0.0402	2.1402	0.0010
0.0406	2.1343	0.0010
0.0456	2.1332	0.0010
0.0458	2.1048	0.0010
0.0471	2.1008	0.0010
0.0474	2.0851	0.0010
0.0482	2.0741	0.0010
0.0502	2.0513	0.0010
0.0517	2.0418	0.0010
0.0518	2.0377	0.0010
0.0523	2.0311	0.0010
0.0527	2.0281	0.0010

0.0529	2.0212	0.0010
0.0537	2.0164	0.0010
0.0554	2.0153	0.0010
0.0555	2.0078	0.0010
0.0564	1.9999	0.0010
0.0568	1.9874	0.0010
0.0576	1.9823	0.0010
0.0590	1.9781	0.0010
0.0601	1.9691	0.0010
0.0604	1.9441	0.0010
0.0627	1.9377	0.0010
0.0629	1.9323	0.0010
0.0644	1.9322	0.0010
0.0645	1.9211	0.0010
0.0668	1.9171	0.0010
0.0677	1.8918	0.0010
0.0704	1.8850	0.0010
0.0708	1.8754	0.0010
0.0727	1.8717	0.0010
0.0729	1.8623	0.0010
0.0750	1.8616	0.0010
0.0750	1.8579	0.0010
0.0762	1.8496	0.0010
0.0765	1.8213	0.0010
0.0783	1.8187	0.0010
0.0821	1.8140	0.0010
0.0822	1.8054	0.0010
0.0881	1.7896	0.0010
0.0886	1.7020	0.0010
0.0977	1.6649	0.0010

Elapsed time is 7195.530935 seconds.



An example of results of PSONK at the side ratio 1:1:1 (1/3)

297 tasks

Scholl&Klein_297task_cycle2787_99:99:99

TS_task_minWS =

Columns 1 through 16
1 2 3 4 111 134 221 83 27 86 5 6 90 259 266 8
138 191 1 2 3 4 109 94 48 134 27 259 5 26 83 90
138 1 2 3 4 94 259 27 86 109 48 93 221 40 111 247

Columns 17 through 32
136 22 94 247 138 191 9 95 48 7 139 56 25 61 65 109
247 95 136 139 266 40 6 8 9 31 30 111 221 297 7 142
105 191 223 31 110 116 162 172 179 83 266 122 26 225 90
82

Columns 33 through 48
24 93 60 10 31 179 12 14 19 142 223 13 29 121 18 17
105 56 86 93 116 122 129 82 10 22 60 68 34 24 110 179
88 89 22 25 95 56 5 6 10 175 9 134 136 61 139 142

Columns 49 through 64
15 20 11 253 128 172 40 82 152 164 21 68 225 44 73 33
12 17 20 11 15 172 175 88 180 252 258 265 272 16 275 280
24 29 7 30 34 129 65 297 60 68 152 8 12 17 180 44

Columns 65 through 80
26 16 49 38 105 88 89 53 116 30 175 122 297 34 110 129
290 162 141 13 73 25 253 29 33 38 21 61 223 225 65 44
49 141 33 38 121 13 14 128 253 11 252 15 164 151 167 73

Columns 81 through 96
23 35 42 46 51 162 141 260 167 180 267 273 55 171 41 45
14 19 18 151 41 49 53 58 152 164 167 171 23 28 32 89
18 53 163 171 258 19 21 16 35 42 46 260 41 58 45 50

Columns 97 through 112
50 151 163 28 81 87 276 281 291 37 32 54 58 59 64 72
45 50 54 296 37 36 39 43 47 52 57 76 62 71 63 66
20 54 296 51 81 87 265 272 275 280 290 267 273 55 23 276

Columns 113 through 128
78 80 125 36 79 39 296 43 47 52 57 76 252 192 201 62
69 74 67 35 42 77 163 260 267 273 276 281 291 70 75 84
281 291 59 64 72 99 103 78 192 79 125 201 80 85 92 98

Columns 129 through 144
100 99 103 258 265 272 66 69 74 275 280 71 85 92 77 290
91 96 97 101 106 112 117 123 124 257 145 155 46 51 147
158
102 107 113 118 126 100 104 108 115 120 150 114 292 119 127
157

Columns 145 through 160
104 98 102 108 63 107 114 292 119 113 118 126 67 70 75 84
121 81 87 149 160 146 156 264 148 159 55 59 64 72 78 125
161 28 37 32 36 39 43 47 52 57 62 76 71 77 63 67

Columns 161 through 176
91 96 97 101 106 112 117 124 123 146 147 145 158 115 149
155
79 192 201 99 103 128 80 100 104 108 115 120 114 292 85
92
66 69 74 70 75 97 84 91 96 101 106 112 117 123 146 156

Columns 177 through 192
160 257 264 156 120 150 127 157 148 159 130 144 161 154 166
131
98 102 107 113 118 126 119 127 157 150 161 130 144 154 131
132
148 149 145 124 130 144 131 133 257 159 155 154 132 264 160
147

Columns 193 through 208
133 170 132 135 137 140 143 169 153 200 165 176 168 173 177
174
133 135 137 140 200 143 153 165 168 173 177 166 170 176 169
174
158 166 170 135 137 140 200 143 153 169 174 287 178 288 165
176

Columns 209 through 224
288 287 178 181 189 188 183 187 295 196 193 182 190 185 197
186
287 178 181 183 196 185 182 189 186 197 187 190 184 288 194
188
168 173 177 181 187 189 197 196 295 186 183 193 198 182 184
188

Columns 225 through 240
184 194 195 205 199 198 227 203 206 209 211 208 202 204 207
210
295 195 227 230 232 205 203 206 209 211 193 198 271 199 289
208
194 185 190 195 205 199 202 251 227 230 271 289 232 204 250
207

Columns 241 through 256
213 212 214 234 215 250 229 235 230 289 232 271 236 239 279
286
202 251 204 207 210 213 212 214 234 215 216 217 218 219 238
285
212 203 208 229 235 236 239 279 286 206 209 211 210 213 214
234

Columns 257 through 272
216 217 218 219 220 222 224 226 228 231 233 237 238 285 240
243
229 235 250 236 239 279 286 220 222 224 226 228 231 233 237
240
256 263 270 215 238 285 216 217 218 219 220 222 224 226 228
231

Columns 273 through 288
241 242 244 255 245 262 251 246 248 249 284 294 254 261 268
269
241 242 244 255 262 243 245 246 248 249 254 261 268 269 256
263
233 237 240 241 242 244 255 262 243 246 245 248 249 254 261
268

Columns 289 through 297
256 263 270 274 278 277 282 283 293
270 274 278 277 282 283 293 284 294
269 274 278 277 282 283 293 284 294

Station =

Columns 1 through 16
2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3
2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3
1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3

Columns 17 through 32
3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4
3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4
3 3 3 3 3 3 5 5 5 5 5 5 5 5 5 5

Columns 33 through 48
4 4 4 4 4 8 8 8 8 8 8 8 8 8 8 10
4 4 5 5 5 5 5 5 5 5 5 6 6 6 6 6
5 5 5 7 7 7 7 7 7 7 7 7 7 7 7 8

Columns 49 through 64
10 10 10 10 10 10 10 11 11 11 11 11 11 11 11 11

6	6	6	6	6	10	10	10	10	10	10	10	10	10	10	10
8	8	8	8	8	8	8	8	8	9	9	9	9	9	9	9

Columns 65 through 80

11	11	15	15	15	15	15	15	15	18	18	18	18	18	18	18
10	10	11	11	11	11	11	11	11	11	11	11	11	11	12	12
9	9	9	10	10	10	10	10	10	10	14	14	14	14	14	14

Columns 81 through 96

18	18	21	21	21	21	21	21	21	21	21	21	21	21	22	22
12	12	12	12	12	12	12	12	15	15	15	15	15	15	15	15
14	14	14	14	14	14	21	22	22	22	22	22	22	22	22	22

Columns 97 through 112

22	22	22	22	22	22	22	22	22	22	22	23	23	23	23	23
15	15	15	20	20	20	20	20	20	20	22	22	22	22	22	22
22	22	28	28	28	28	28	29	29	29	29	29	29	29	29	29

Columns 113 through 128

23	23	23	23	23	23	25	25	26	26	26	26	27	27	27	28
22	22	22	22	22	24	24	24	24	24	24	24	24	24	25	26
30	30	30	30	30	30	30	30	30	29	29	29	29	29	28	28

Columns 129 through 144

28	28	28	28	29	29	29	29	29	29	29	30	30	30	30	30
26	27	27	27	27	27	27	27	27	27	27	27	28	28	28	28
28	28	27	27	27	27	27	27	27	27	27	27	27	27	27	26

Columns 145 through 160

30	30	30	30	30	30	29	29	29	29	29	29	28	28	28	28
29	30	30	30	30	30	30	30	30	30	30	30	30	30	29	29
26	26	26	26	26	26	26	25	25	25	25	25	25	25	25	25

Columns 161 through 176

28	27	27	27	26	26	26	25	24	24	24	24	24	24	24	24
29	29	29	29	29	29	28	28	28	28	28	28	28	28	28	26
25	24	24	24	24	24	24	24	24	24	23	23	23	23	23	23

Columns 177 through 192

24	24	24	24	24	24	24	20	20	20	20	20	20	20	20	19
26	26	26	26	26	25	25	25	23	23	23	23	23	23	23	23
23	23	23	23	23	23	23	21	20	20	20	20	20	20	20	20

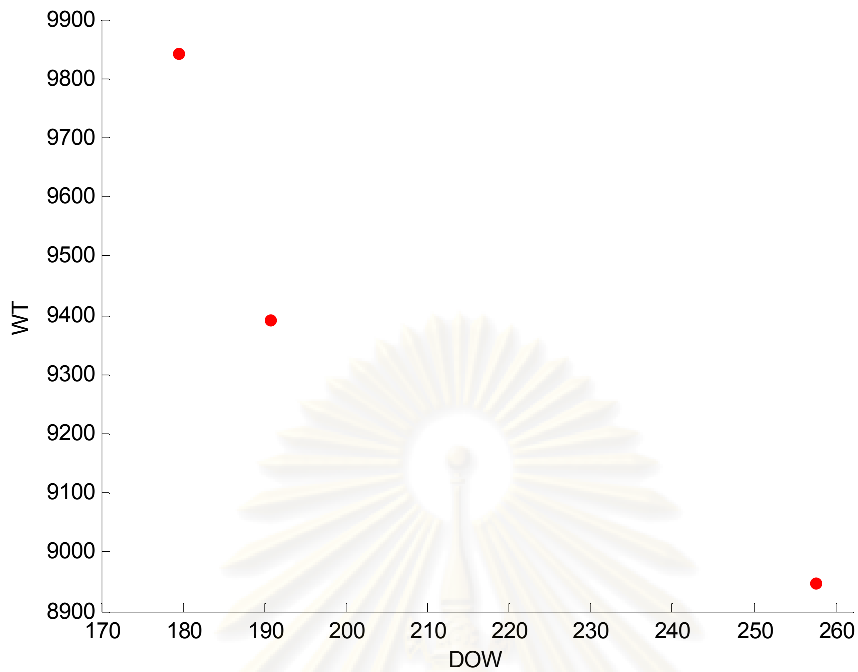
Columns 193 through 208

19	19	19	19	19	19	19	19	19	19	19	17	17	17	17	17
23	21	21	21	21	21	21	21	21	21	21	21	19	19	19	19
20	20	19	19	19	19	19	19	19	19	19	19	19	19	19	18

Columns 209 through 224

17	17	17	17	17	17	17	17	16	16	16	16	16	16	16	16

```
    19    19    19    19    19    19    19    19    18    18    18    18    18    18    18    18
    18    18    18    18    18    18    18    18    18    17    17    17    17    17    17    17

  Columns 225 through 240
    16    16    14    14    14    14    14    14    14    14    14    14    14    13    13    13
    18    17    17    17    17    17    17    17    17    17    17    17    17    16    16    16
    17    17    17    17    16    16    16    16    16    16    16    16    16    16    16    15

  Columns 241 through 256
    13    13    13    13    13    13    13    13    12    12    12    12    12    12    12    12
    16    16    16    16    16    16    16    16    16    16    14    14    14    14    14    14
    15    15    15    15    15    15    15    15    15    15    13    13    13    13    13    13

  Columns 257 through 272
    12    12     9     9     9     9     9     9     9     9     9     9     9     9     7     7
    14    13    13    13    13    13    13    13    13    13    13    13     9     9     9     9
    13    13    13    13    13    12    12    12    12    12    12    12    12    12    12    11

  Columns 273 through 288
     7     7     7     7     7     6     6     6     6     6     6     5     5     5     5     5
     9     9     9     8     8     8     8     8     8     7     7     7     7     7     7     7
    11    11    11    11    11    11     6     6     6     6     6     6     4     4     4     4

  Columns 289 through 297
     5     5     5     5     1     1     1     1     1
     7     7     7     7     1     1     1     1     1
     4     4     4     4     2     2     2     2     2


WT_DOW_J =

  1.0e+003 *

    0.1795    9.8422    0.0300
    0.1907    9.3935    0.0300
    0.2576    8.9494    0.0300

Elapsed time is 5503.337533 seconds.
```



An example of results of PSONK at the side ratio 1:4:4 (1/9)

36 tasks

CaseStudy_36task_cycle1371_16:16:4

WT_DOW =

1.0e+003 *

0.0821	1.1103	0.0060
0.0861	1.0585	0.0060
0.0935	1.0328	0.0060
0.0999	1.0062	0.0060
0.0999	0.9959	0.0060
0.1048	0.9513	0.0060
0.1095	0.9398	0.0060
0.1100	0.9382	0.0060
0.1133	0.9290	0.0060
0.1147	0.9061	0.0060
0.1147	0.8911	0.0060
0.1185	0.8896	0.0060
0.1193	0.8800	0.0060
0.1196	0.8768	0.0060
0.1216	0.8524	0.0060
0.1217	0.8441	0.0060
0.1237	0.8354	0.0060
0.1252	0.8211	0.0060

0.1292	0.8084	0.0060
0.1295	0.8016	0.0060
0.1300	0.7892	0.0060
0.1323	0.7809	0.0060
0.1356	0.7659	0.0060
0.1380	0.7659	0.0060
0.1395	0.7627	0.0060
0.1403	0.7620	0.0060
0.1404	0.7612	0.0060
0.1405	0.7471	0.0060
0.1412	0.7363	0.0060
0.1416	0.7298	0.0060
0.1422	0.7283	0.0060
0.1430	0.7267	0.0060
0.1440	0.7160	0.0060
0.1463	0.7105	0.0060
0.1467	0.7020	0.0060
0.1478	0.6961	0.0060
0.1502	0.6836	0.0060
0.1513	0.6779	0.0060
0.1516	0.6767	0.0060
0.1537	0.6741	0.0060
0.1538	0.6537	0.0060
0.1573	0.6374	0.0060
0.1609	0.6320	0.0060
0.1620	0.6180	0.0060
0.1650	0.6007	0.0060
0.1674	0.5846	0.0060
0.1702	0.5613	0.0060
0.1958	0.5551	0.0060
0.1970	0.5435	0.0060
0.2225	0.5313	0.0060
0.2246	0.5215	0.0060
0.2856	0.5206	0.0060
0.3825	0.5204	0.0060

TS_task_minWS =

Columns 1 through 18

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

```
    1    2   11    6    3   15    8    9    7   10   17   19   18   20   21   12   13   14
    1    2    6   17    3    4   11   10   15    8   12    7   18   19   20   21   13   14
    1    2   15    3   11   17   20   12   13   19   18   21    7   10    6    8    9   14
    1    2   15    6    7   17   20   18   19   10   11   12   13   14   16   22   21
23
    1    2   11    6    7   17   19    3    4    5    8   15    9   18   20   21   10   12
    1    2   10    7   17    3   15    8    9    6   11   12   13   14   16   22   18   19
    1    2    6   11   12   15    3   17    7   13   14   16   22   18   19   20   21
23
    1    2    6    7   15    3    4   11   10    8    9   17   19   18   20   12   13   14
    1    2   10    6    7   17   19   18   20   21    3   15    8    9   11   12   13   14
    1    2    7   10    3   15   17   19    8    9   18   20   11    6   12   13   14   16
    1    2    6    7   10   15    3   17   18   19    8    9   20   21   11   12   13   14
    1    2   10    7    8    9   17    3   15    6   11   12   13   14   16   22   18   20
    1    2   11    6   12    3   15   17   19   10    8   18   20   21   13   14    9   16
    1    2   17    8    6    9    3   19   18   20   21   15   10   11    7    4   12   13
    1    2    6   15    7   17    3    4   11   10    8   18   19   20   12   13   14   16
    1    2    6   10    8    9    7    3   15   17   18   19   20   21   11   12   13   14
    1    2    7   15    6   10   11   17    8    3    4   12   13   14    5   20   19   18
    1    2    8    9   11   15    7   10    3    4    5    6   12   13   14   16   17   22
    1    2   11   17   20   19    6   10    8    3    4    5   15    7   18   21   12   13
    1    2   10    7    8    9    3   15    6   11   12   13   14   16   22   17   19   18
    1    2   11    7    6   10   17   12    3    4   19   18   20   21   15    8    9   13
    1    2   15   17   20    3    4   11   12   13   18   10    6    7   19   21   14    5
    1    2    6   15    8    9   17   18   10    3   20   11   12   13   14   16   22   19
    1    2   10    7   17   20   11    6   12   15    8   18   19   21   13   14   16
22
    1    2   17    8    3    4   11   10   15   18   19    6    9   20   21   12   13   14
    1    2   17   18   19    6    7   10    3    4    5    8    9   20   11   12   15   21
    1    2   10    8    9    7   15   17    6   11   12   13   14   18   19   20   21   16
    1    2   11    8    3   17   20   12   15    9    7    6   10   19   18   21   13   14
    1    2   11    8    9    6   15    3   17   18   19   10    7    4   12   13   14   16
    1    2   10    3   17   18   20    7   15    8    9   19   21   11    6   12   13   14
    1    2   11    6   12   15   17   19   10   18   20    7    3    4    5   13   14   16
    1    2   17   19    6    7   10   11   12   13   14   15    3    4    5   18   20   21
    1    2   15   10   11    6   17   20   18   19   21   12   13   14   16   22   23
24
    1    2   17   20   11    6   15    7   10    8   18   19   21    3    4    5    9   12
    1    2   11    8    9    7   10   17    6   15    3   19   20   12   13   14   16   22
    1    2   17    7    8    3    4   15   18   19    6    5   10   11   20   12   13   14
    1    2   11    7    6   17   19   18   20    3   15    8    9   21   12   13   14   16
    1    2    6   17   18   19   20   11    8    7   10   12   21    3    4    5   13   14
    1    2    6   11   15    8    3    4    9   17    7   12   13   14   18   20   19   21
    1    2    6   17    3    4   11   12   15    8    9   13   14    7   18   19   16   22
    1    2   11    8    7   10    9    3    4    5   12   15   13   17   19   18   14    6
    1    2    6    7    8    9    3   15   10   11   12   13   14   16   22    4    5   17
    1    2   10    3   15    8    7   17   20   19    6   11   12   13   14   16   22   18
    1    2   15   17   19   10    3   11    8    7    6   12   13   14   16   22   18   20
```

Columns 19 through 36

13 14 16 22 23 24 25 26 27 28 29 30 31 32 33 34 35
36
16 22 23 24 25 26 27 28 4 5 29 30 31 32 33 34 35
36
14 16 22 23 24 25 26 9 27 28 29 30 31 32 33 34 35
36
13 14 16 22 23 24 25 26 27 28 29 30 31 32 33 34 35
36
16 22 23 24 25 26 27 28 4 5 29 30 31 32 33 34 35
36
16 22 23 24 25 26 27 28 4 5 29 30 31 32 33 34 35
36
16 22 23 24 25 26 5 6 27 28 29 30 31 32 33 34 35
36
22 21 23 24 25 26 9 27 28 5 29 30 31 32 33 34 35
36
16 22 19 21 23 24 25 26 27 28 29 30 31 32 33 34 35
36
16 22 23 24 25 26 27 28 4 5 29 30 31 32 33 34 35
36
9 5 16 22 23 24 25 26 27 28 29 30 31 32 33 34 35
36
16 22 23 24 25 26 27 28 4 5 29 30 31 32 33 34 35
36
24 25 26 3 4 5 8 9 27 28 29 30 31 32 33 34 35
36
13 14 16 22 23 24 25 26 27 28 29 30 31 32 33 34 35
36
20 21 23 24 25 26 27 28 4 5 29 30 31 32 33 34 35
36
10 8 9 24 25 26 27 28 4 5 29 30 31 32 33 34 35
36
16 22 21 23 24 25 26 27 28 5 29 30 31 32 33 34 35
36
16 22 23 24 25 26 27 28 4 5 29 30 31 32 33 34 35
36
22 21 23 24 25 26 27 28 4 5 29 30 31 32 33 34 35
36
16 22 23 24 25 26 27 28 4 5 29 30 31 32 33 34 35
36
19 21 23 24 25 26 27 28 4 5 29 30 31 32 33 34 35
36
22 23 24 25 26 7 27 28 4 5 29 30 31 32 33 34 35
36
14 16 22 23 24 25 26 27 28 5 29 30 31 32 33 34 35
36
22 5 9 21 23 24 25 26 27 28 29 30 31 32 33 34 35
36

16 22 23 24 25 26 27 28 4 5 29 30 31 32 33 34 35
36
21 16 22 23 24 25 26 9 27 28 29 30 31 32 33 34 35
36
18 20 19 21 23 24 25 26 27 28 29 30 31 32 33 34 35
36
14 16 22 23 24 25 26 9 27 28 29 30 31 32 33 34 35
36
20 21 23 24 25 26 27 28 4 5 29 30 31 32 33 34 35
36
14 16 22 23 24 25 26 27 28 5 29 30 31 32 33 34 35
36
8 9 16 22 23 24 25 26 27 28 29 30 31 32 33 34 35
36
21 23 24 25 26 7 27 28 4 5 29 30 31 32 33 34 35
36
23 24 25 26 9 27 28 3 4 5 29 30 31 32 33 34 35
36
5 16 22 23 24 25 7 26 27 28 29 30 31 32 33 34 35
36
13 14 16 22 23 24 25 26 27 28 29 30 31 32 33 34 35
36
22 23 24 25 26 27 28 3 4 5 29 30 31 32 33 34 35
36
16 22 23 24 25 26 27 28 4 5 29 30 31 32 33 34 35
36
22 5 20 21 23 24 25 26 27 28 29 30 31 32 33 34 35
36
16 22 23 24 25 26 27 28 4 5 29 30 31 32 33 34 35
36
22 21 23 24 25 26 8 9 27 28 29 30 31 32 33 34 35
36
16 22 23 24 25 26 8 9 27 28 29 30 31 32 33 34 35
36
25 26 7 8 9 27 28 3 4 5 29 30 31 32 33 34 35 36
13 14 16 22 23 24 25 26 27 28 29 30 31 32 33 34 35
36
18 21 23 24 25 26 27 28 4 5 29 30 31 32 33 34 35
36
16 22 9 21 23 24 25 26 27 28 29 30 31 32 33 34 35
36
22 23 10 4 5 24 25 26 27 28 29 30 31 32 33 34 35
36
15 9 16 22 23 24 25 26 27 28 29 30 31 32 33 34 35
36
16 22 23 10 24 25 26 27 28 5 29 30 31 32 33 34 35
36
10 5 20 21 23 24 25 26 27 28 29 30 31 32 33 34 35
36

```
   20   21   16   22   23   24   25   26   27   28   29   30   31   32   33   34   35
36
   20   19   18   21   23   24   25   26   27   28   29   30   31   32   33   34   35
36
   21   23   24   25   26    9   27   28    4    5   29   30   31   32   33   34   35
36
   21   23   24   25   26    9   27   28    4    5   29   30   31   32   33   34   35
36
```

Station =

Columns 1 through 18

```
 1  1  1  1  1  1  1  2  2  2  3  3  3  3  3  3  4  4
 1  1  1  1  1  1  2  2  2  2  3  3  3  3  3  3  4  4
 1  1  1  1  1  1  1  2  2  3  3  3  3  3  4  4  5  6
 1  1  1  1  1  1  2  2  2  3  3  3  3  3  4  4  4  5
 1  1  1  1  1  1  1  2  2  2  2  2  3  3  3  3  4  4
 1  1  1  1  1  2  2  2  2  3  3  3  3  4  4  4  4  4
 1  1  1  1  1  2  2  2  2  2  3  3  3  3  4  4  4  4
 1  1  1  1  1  1  2  2  2  2  3  3  3  3  4  4  4  4
 1  1  1  1  1  1  2  2  2  3  3  3  4  4  4  4  4  5
 1  1  1  1  1  1  2  2  2  3  3  3  3  4  5  5  5  6
 1  1  1  1  1  1  2  2  2  2  3  4  4  5  5  5  5  6
 1  1  1  1  2  2  2  2  2  3  3  3  3  4  4  5  5  5
 1  1  1  1  1  1  2  2  2  2  2  3  3  3  3  4  4  5
 1  1  1  1  1  1  2  2  2  2  3  4  4  4  4  4  4  5
 1  1  1  1  1  2  2  2  2  2  3  3  3  4  4  5  5  5
 1  1  1  1  2  2  2  2  2  2  3  3  3  3  4  4  5  5
 1  1  1  1  2  2  2  2  3  3  3  3  4  4  4  4  4  4
 1  1  1  1  2  2  2  2  2  3  3  3  4  4  5  5  5  5
 1  1  1  1  2  2  2  2  3  3  3  4  4  4  5  5  5  5
 1  1  1  1  2  2  2  2  2  2  3  3  4  4  5  5  5  5
 1  1  1  2  2  2  2  3  3  3  3  4  4  4  4  5  5  5
 1  1  1  1  2  2  3  3  3  3  3  3  3  4  4  5  5  5
 1  1  1  1  1  1  2  2  2  2  2  3  4  4  4  4  4  4
 1  1  1  1  1  1  2  2  3  3  4  4  4  5  5  5  5  5
 1  1  1  1  2  2  2  3  3  3  3  3  4  4  5  6  6  6
 1  1  1  1  1  2  2  2  3  3  4  4  4  5  5  5  6  6
 1  1  1  1  2  3  3  4  5  5  6  6  6  6  6  6  6  6
 1  1  1  1  1  2  2  2  3  3  4  4  5  5  5  5  5  5
 1  1  1  1  2  2  3  3  3  3  4  4  4  4  5  5  5  5
 1  1  1  1  1  2  2  3  3  3  3  3  4  4  5  5  5  5
 1  1  1  1  1  2  2  3  3  3  3  4  4  5  6  6  6  6
 1  1  1  1  2  2  2  2  3  3  3  3  4  4  4  4  5  5
 1  1  1  2  2  2  4  4  5  5  6  6  6  6  6  6  6  6
 1  1  1  1  1  2  2  2  2  2  4  4  4  5  5  5  5  6
 1  1  1  1  1  1  2  2  4  5  5  5  5  6  6  6  6  6
 1  1  1  2  2  3  3  3  3  4  4  4  4  4  5  5  5  5
```

1	1	1	2	2	2	2	3	3	3	4	4	4	5	5	6	6	6
1	1	1	2	2	2	3	3	3	3	3	3	4	4	4	4	4	4
1	1	1	2	2	2	3	4	4	4	4	4	5	5	5	5	5	6
1	1	1	1	2	2	2	2	3	3	3	4	4	5	5	5	6	6
1	1	1	1	1	2	2	3	3	3	4	4	4	5	5	5	6	6
1	1	1	2	2	2	2	3	3	4	4	5	5	5	5	6	6	6
1	1	1	1	2	2	2	2	3	3	3	3	3	5	5	5	5	6
1	1	1	2	2	3	3	3	3	4	6	6	6	6	5	5	5	5
1	1	1	1	2	2	2	3	3	3	3	5	5	5	6	6	6	6
1	1	1	2	2	2	2	2	3	5	5	5	5	6	6	6	6	6
1	1	1	1	1	1	2	2	3	3	3	3	3	4	4	4	4	6
1	1	1	1	2	2	2	4	4	4	4	5	5	6	6	6	6	6
1	1	1	1	2	2	2	2	4	4	4	4	5	5	5	5	5	6
1	1	1	2	2	2	2	3	3	3	3	5	5	5	6	6	6	6
1	1	1	1	2	2	2	2	4	4	4	4	5	5	5	5	5	5
1	1	1	2	2	2	2	4	4	4	4	5	5	5	5	5	5	5
1	1	1	1	1	2	2	2	2	4	4	4	4	5	5	5	5	6

Columns 19 through 36

4	5	5	6	6	6	6	6	6	6	5	5	5	4	3	2	2	1
4	5	5	6	6	6	6	6	6	5	5	5	4	4	3	2	2	1
6	6	6	6	6	6	6	5	5	5	5	5	4	4	3	2	2	1
5	6	6	6	6	6	6	6	5	5	5	5	4	4	3	2	1	1
4	5	5	6	6	6	6	6	6	5	5	5	4	4	3	3	1	1
4	5	5	6	6	6	6	6	6	5	5	5	4	3	3	2	1	1
4	5	5	5	5	5	6	6	6	6	6	6	5	4	3	1	1	1
5	5	5	6	6	6	6	6	6	6	5	5	4	4	3	1	1	1
5	6	6	6	6	6	6	6	5	5	5	5	4	3	2	1	1	1
6	6	6	6	6	6	5	5	5	5	4	4	4	3	3	2	2	1
6	6	6	6	6	6	5	5	5	5	4	4	4	3	3	3	2	1
5	5	5	6	6	6	6	6	6	5	4	4	4	3	2	2	1	1
6	6	6	6	5	5	5	5	5	4	4	4	3	3	2	1	1	1
5	5	5	6	6	6	6	6	6	6	5	5	4	3	3	1	1	1
5	6	6	6	6	6	6	6	5	5	4	4	4	3	3	1	1	1
5	6	6	6	6	6	6	5	5	5	4	4	3	3	1	1	1	1
4	5	5	5	6	6	6	6	6	6	5	5	4	3	1	1	1	1
5	6	6	6	6	6	6	6	5	5	4	4	3	2	1	1	1	1
5	6	6	6	6	6	6	6	5	5	4	4	3	3	2	1	1	1
5	6	6	6	6	6	6	6	5	5	4	4	3	3	1	1	1	1
6	6	6	6	6	6	6	5	5	5	4	4	3	2	1	1	1	1
5	5	6	6	6	6	6	6	5	5	4	4	4	2	1	1	1	1
5	5	6	6	6	6	6	6	6	5	5	5	3	3	3	1	1	1
6	6	6	6	6	6	5	5	5	5	4	4	3	3	2	1	1	1
6	6	6	6	5	5	5	5	5	5	4	4	3	2	1	1	1	1
6	6	6	6	6	6	5	5	5	5	4	4	3	2	2	1	1	1
6	6	5	5	4	4	4	4	4	3	3	3	2	2	2	1	1	1
6	6	6	6	6	6	6	5	5	5	4	4	3	2	2	1	1	1
6	6	6	6	6	6	5	5	5	5	4	4	3	2	2	1	1	1
6	6	6	6	6	6	6	5	5	5	4	4	2	2	2	1	1	1

6	6	5	5	5	5	5	5	4	4	4	4	3	3	2	1	1	1
6	6	6	6	6	6	5	5	5	5	4	4	2	2	1	1	1	1
5	5	5	5	4	4	4	4	3	3	3	3	2	2	1	1	1	1
6	6	6	6	6	5	5	5	4	4	4	4	3	3	3	3	2	1
6	6	5	5	4	4	3	3	3	3	3	3	2	2	1	1	1	1
6	6	6	6	6	6	6	5	4	4	3	3	2	2	1	1	1	1
6	6	6	6	5	5	5	5	5	5	4	4	3	3	1	1	1	1
4	5	6	6	6	6	6	6	5	5	5	5	2	2	1	1	1	1
6	6	6	6	6	6	5	5	5	4	3	3	3	2	1	1	1	1
6	6	6	6	5	5	5	4	4	4	3	3	2	2	1	1	1	1
6	6	6	6	5	5	5	4	4	3	3	3	2	2	1	1	1	1
6	6	6	5	5	5	5	4	4	3	3	3	2	2	1	1	1	1
6	6	6	6	6	6	4	4	4	4	4	4	3	2	1	1	1	1
5	5	5	5	4	4	4	4	4	4	3	3	2	2	1	1	1	1
6	6	6	6	5	5	4	4	4	4	4	4	3	3	1	1	1	1
6	6	6	4	4	4	4	4	3	3	3	3	2	2	1	1	1	1
6	6	6	6	6	6	5	5	5	5	5	5	2	2	1	1	1	1
6	6	5	5	5	5	5	4	4	4	3	3	3	3	1	1	1	1
6	6	6	6	5	5	4	4	4	4	4	3	3	3	1	1	1	1
6	6	6	6	6	5	5	5	5	5	5	4	4	4	1	1	1	1
5	6	6	6	6	6	4	4	4	4	4	3	3	3	1	1	1	1
6	6	6	5	5	5	4	4	4	4	4	3	3	3	1	1	1	1
6	5	5	5	5	5	4	4	4	4	4	3	3	3	1	1	1	1

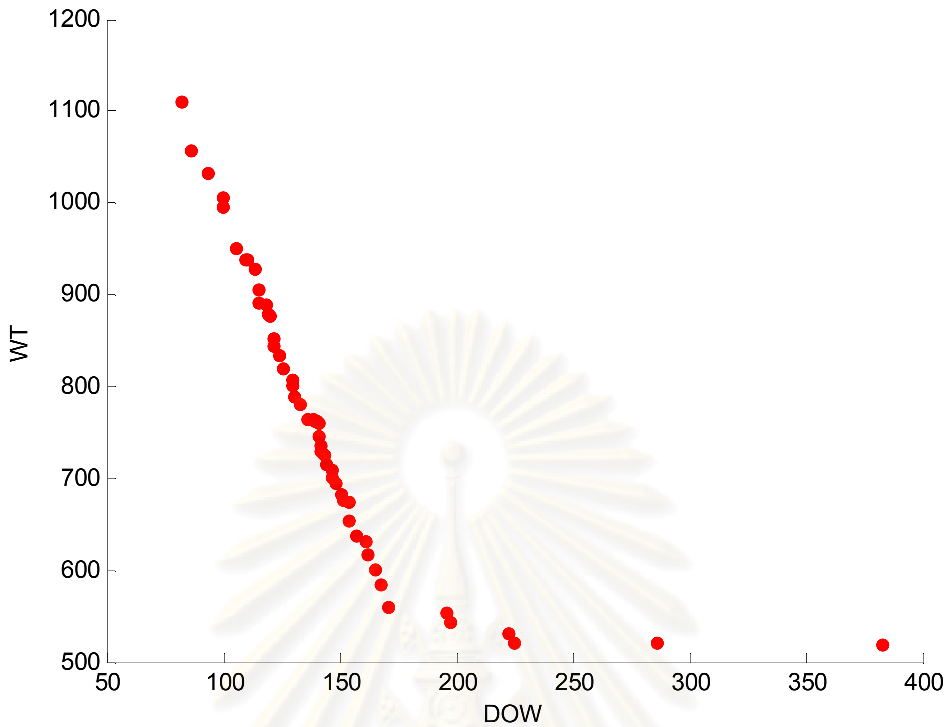
WT_DOW_J =

1.0e+003 *

0.0821	1.1103	0.0060
0.0861	1.0585	0.0060
0.0935	1.0328	0.0060
0.0999	1.0062	0.0060
0.0999	0.9959	0.0060
0.1048	0.9513	0.0060
0.1095	0.9398	0.0060
0.1100	0.9382	0.0060
0.1133	0.9290	0.0060
0.1147	0.9061	0.0060
0.1147	0.8911	0.0060
0.1185	0.8896	0.0060
0.1193	0.8800	0.0060
0.1196	0.8768	0.0060
0.1216	0.8524	0.0060
0.1217	0.8441	0.0060
0.1237	0.8354	0.0060
0.1252	0.8211	0.0060
0.1292	0.8084	0.0060
0.1295	0.8016	0.0060
0.1300	0.7892	0.0060

0.1323	0.7809	0.0060
0.1356	0.7659	0.0060
0.1380	0.7659	0.0060
0.1395	0.7627	0.0060
0.1403	0.7620	0.0060
0.1404	0.7612	0.0060
0.1405	0.7471	0.0060
0.1412	0.7363	0.0060
0.1416	0.7298	0.0060
0.1422	0.7283	0.0060
0.1430	0.7267	0.0060
0.1440	0.7160	0.0060
0.1463	0.7105	0.0060
0.1467	0.7020	0.0060
0.1478	0.6961	0.0060
0.1502	0.6836	0.0060
0.1513	0.6779	0.0060
0.1516	0.6767	0.0060
0.1537	0.6741	0.0060
0.1538	0.6537	0.0060
0.1573	0.6374	0.0060
0.1609	0.6320	0.0060
0.1620	0.6180	0.0060
0.1650	0.6007	0.0060
0.1674	0.5846	0.0060
0.1702	0.5613	0.0060
0.1958	0.5551	0.0060
0.1970	0.5435	0.0060
0.2225	0.5313	0.0060
0.2246	0.5215	0.0060
0.2856	0.5206	0.0060
0.3825	0.5204	0.0060

Elapsed time is 356.469689 seconds.


1200
1100
1000
900
800
700
600
500
WT
50
100
150
200
250
300
350
400
DOW

VITA

Mr. Ronnachai Sirovetnukul was born on April 13, 1975 in Bangkok, Thailand. After he had studied a junior high school from Assumption College and a senior high school from Suankularb Wittayalai School, he earned his Bachelor of Industrial Engineering degree from Kasetsart University in 1996. After he graduated his Master degree in Industrial Engineering from Chulalongkorn University in 1998, he had worked as a lecturer at the department of Industrial Engineering at Mahidol University. Prior to moving to pursue a doctorate, he gained industrial experience as a researcher with the federation of Thai industries. His research interests are Artificial Intelligence for Combinatorial Optimization; Balancing, Sequencing and Scheduling; Logistics and Supply Chain Management; Mass Customization Production Systems; and Simulation Modeling.

During his doctoral study at Chulalongkorn University, the relevant work was published at the Electrical Engineering Conference (33th) in Chiangmai, Thailand in 2010. He presented two international conference papers at the IEEE International Conference on Industrial Engineering and Engineering Management (IEEM) in Hong Kong in 2009 and Macao in 2010. He also published an international journal paper in Engineering Journal (Volume 14, Number 2, pp.53-78, 2010). At present, he works as a lecturer at the department of Industrial Engineering, Mahidol University which is addressed on 999 Phutthamonthon Sai 4 Road, Salaya, Phutthamonthon, Nakhonpathom, 73170, Thailand. His email address is egrsr@mahidol.ac.th.