

การมอบหมายงานให้กับพนักงานที่มีระดับความชำนาญหลากหลายโดยมีการเรียนรู้



นางสาวกัญญา ทองสนิท

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรดุษฎีบัณฑิต

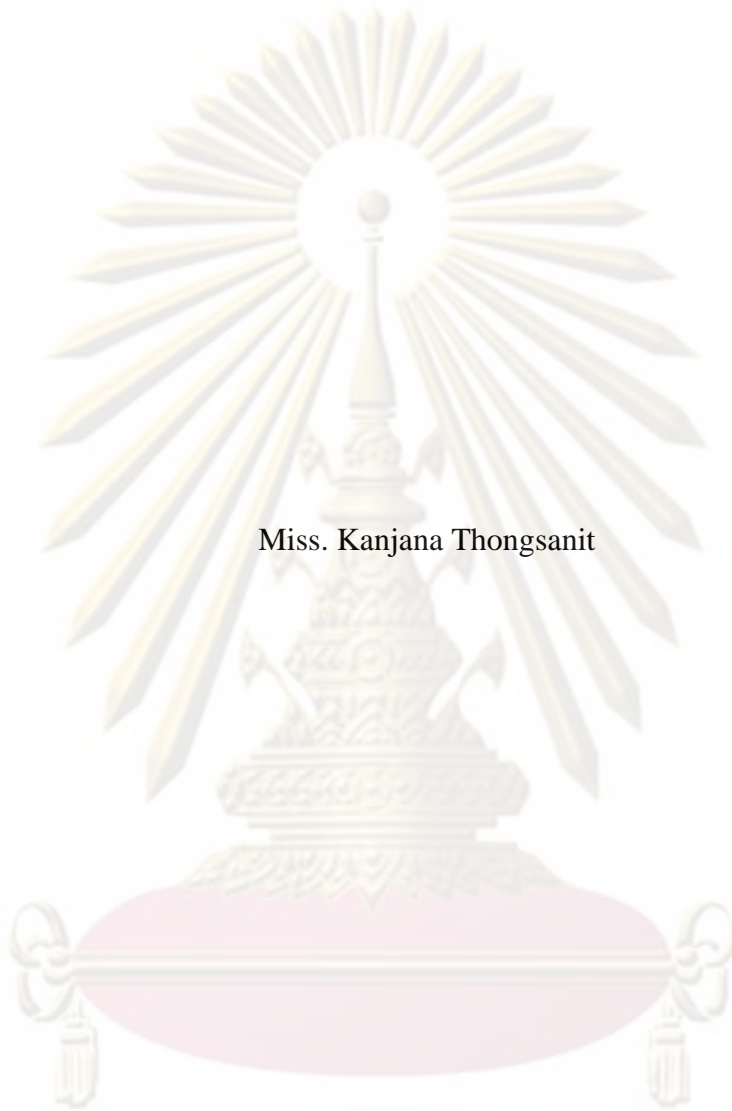
สาขาวิชาวิศวกรรมอุตสาหการ ภาควิชาวิศวกรรมอุตสาหการ

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

ปีการศึกษา 2552

ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย

ASSIGNMENT OF CROSS-TRAINED WORKERS WITH LEARNING EFFECT



Miss. Kanjana Thongsanit

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 2009

Copyright of Chulalongkorn University

Thesis Title ASSIGNMENT OF CROSS-TRAINED WORKERS WITH
 LEARNING EFFECT
By Miss Kanjana Thongsanit
Field of Study Industrial Engineering
Thesis Advisor Assistant Professor Rein Boondiskulchok, D.Eng.
Thesis Co-advisor Assistant Professor Wipawee Tharmmaphornphilas, 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 Lerthirunwong, Dr.Ing.)


THESIS COMMITTEE

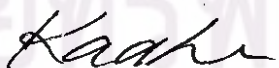
.....Chairman
(Assistant Professor Manop Reodecha, Ph.D.)

.....Thesis Advisor
(Assistant Professor Rein Boondiskulchok, D.Eng.)

.....Thesis Co-advisor
(Assistant Professor Wipawee Tharmmaphornphilas, Ph.D.)

.....Examiner
(Assistant Professor Paveena Chaovalitwongse, Ph.D.)

.....Examiner
(Assistant Professor Napassavong Rojanarowan, Ph.D.)


.....External Member
(Assistant Professor Karndee Leopairote, Ph.D.)

ศูนย์วิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

กัญญา ทองสนิท : การมอบหมายงานให้กับพนักงานที่มีระดับความชำนาญหลากหลาย โดยมีการเรียนรู้. (ASSIGNMENT OF CROSS-TRAINED WORKERS WITH LEARNING EFFECT). อ. ที่ปรึกษาวิทยานิพนธ์หลัก : ผศ. ดร. เจริญ บุญดีสกุลโชค, อ. ที่ปรึกษาวิทยานิพนธ์ร่วม : ผศ. ดร. วิภาวี ธรรมาภรณ์พิลาศ, 93 หน้า.

งานวิจัยนี้ได้ยกประเด็นปัญหาด้านคุณภาพคำตอบ ของปัญหาการมอบหมายงานที่ใช้ในอุตสาหกรรมว่าอาจไม่เหมาะสม ถ้าพนักงานที่ถูกมอบหมายมีระดับความสามารถที่ความแตกต่าง โดยทั่วไปแล้วในอุตสาหกรรมเครื่องนุ่งห่ม จะจัดสมดุลการผลิตโดยใช้เวลามาตรฐานในการจัดกลุ่มงาน เพื่อจัดงานเข้าสู่สถานีงาน หลังจากนั้นกลุ่มงานที่ถูกจัดไว้ในแต่ละสถานีจะถูกมอบหมายให้กับพนักงาน วิธีการจัดกลุ่มโดยไม่ได้พิจารณาถึงระดับความสามารถของพนักงานแบบนี้ อาจจำกัดคุณภาพของคำตอบ ดังนั้น งานวิจัยนี้จึงได้รวมปัญหาการจัดสมดุลการผลิตและการมอบหมายงานเข้าด้วยกัน นอกจากนี้แล้ว เนื่องจากในอุตสาหกรรมด้านแฟชั่น มักมีการผลิตผลิตภัณฑ์ แบบใหม่หรือสไตล์ใหม่ในทุกๆ ฤดูกาล ด้วยเหตุนี้จึงทำให้พนักงานต้องเรียนรู้งานใหม่อยู่ตลอด ในงานวิจัยนี้จึงพิจารณาระดับความสามารถของพนักงาน ทั้งที่กำหนดคงที่และ ที่ระดับความสามารถที่เปลี่ยนแปลงไปจากการเรียนรู้ โดย ได้สร้างโปรแกรมเชิงเส้นแบบผสมและแบบทวิภาค ของทั้งสองปัญหา โดยวัตถุประสงค์ของปัญหาคือ ต้องการให้รอบเวลาในการผลิตน้อยสุด กรณีที่ระดับความสามารถคงที่ และต้องการทำให้เวลาปิดงานเร็วที่สุด กรณีที่พิจารณาด้านการเรียนรู้ ในการหาคำตอบ ฮิวริสติกของแต่ละปัญหาได้ถูกพัฒนาขึ้น โดยมีการกำหนดขอบเขตล่างและของเขตบนของคำตอบ ประสิทธิภาพของฮิวริสติกถูกทดสอบโดยเปรียบเทียบกับคำตอบที่ดีที่สุด นอกจากนี้ในงานวิจัยยังยืนยันความไม่เหมาะสมของวิธีการมอบหมายที่ใช้กันในอุตสาหกรรม เมื่อพนักงานมีทักษะที่แตกต่างกันมาก ๆ และยืนยันความไม่สมในการมอบหมายงานที่กำหนดระดับความสามารถของพนักงานคงที่ในขณะที่ระดับความสามารถของพนักงานจะเปลี่ยนแปลงตลอดจากการเรียนรู้ โดยเฉพาะอย่างยิ่งเมื่อพนักงานมีการเรียนรู้ที่ต่างกัน

ภาควิชา วิศวกรรมอุตสาหกรรม
สาขาวิชา วิศวกรรมอุตสาหกรรม
ปีการศึกษา 2552

ลายมือชื่อผู้ผลิต กัญญา ทองสนิท
ลายมือชื่อ อ. ที่ปรึกษาวิทยานิพนธ์หลัก 
ลายมือชื่อ อ. ที่ปรึกษาวิทยานิพนธ์ร่วม วิภาวี

4771848321 : MAJOR: INDUSTRIAL ENGINEERING

KEYWORDS: ASSINGMENT PROBLEM / LEARNING / HEURISTIC

KANJANA THONGSANIT: ASSIGNMENT OF CROSS-TRAINED WORKERS WITH LEARNING EFFECT. ADVISOR : ASST. PROF. REIN BOONDISKULCHOK, Ph.D., COADVISOR : ASST. PROF. WIPAWEE THARMMAPHORNPHILAS, 93 pp.

This dissertation addresses the problem of the practical application of worker assignment among workers of highly varying skill levels. Generally, a heuristic method for the assembly line balancing problem is applied in garment industry, which is that the tasks or grouped tasks are first pre-determined based on a standard processing time and then are assigned to the workers. The process of grouping and assigning tasks without considering the skill level of the worker limits the quality of the solution. Consequently, an integrated approach to the assembly line balancing and worker assignment problem has been developed.

Due to the nature of fashion industry, new designs and new styles are launched every season and the workers in the industry are continuously required to learn new processes. This dissertation concerns the skill level of workers of both constant skill levels and learning ability. MIP models for both problems were developed and the objective is to minimize the cycle time and makespan respectively. To find a solution, a heuristic was proposed and a lower bound and the upper bound were determined. The performances of the heuristics were tested by comparing the solution obtained to the optimal solution. The result of the comparison of the solution from the heuristic and in practical application confirms the disadvantage if the practical application is applied for workers of highly varying skill levels. The constant skill level of workers learning ability is not appropriate in situations where workers have differing learning slopes.

Department : Industrial Engineering

Field of Study : Industrial Engineering

Academic Year : 2009

Student's Signature กัญญา ทองสนธิ์

Advisor's Signature เรณู บูณดิษฐ์

Co-Advisor's Signature วิภาวี

ACKNOWLEDGEMENTS

I would like to thank my advisor Asst. Prof. Rein Boondiskulchok, Ph.D. and my co-advisor Asst. Prof. Wipawee Thammaphornphilas , Ph.D. to provide advice and support all period of the study and I always remember the word “balancing your life also” as you recommendation. I would also like to thank Prof. Dr. William G. Ferrell and Assoc. Prof. Dr. Mary E. Kurz from Faculty of Industrial Engineering, Clemson University, U.S.A, for encouragement and suggestions.

I am very thankful to my committee members and faculty members Asst. Prof. Dr. Manop Reodecha, Asst. Prof. Dr. Paveena Chaovalitwongse, Asst. Prof. Dr. Napassavong Rojanarowan, Prof. Dr. Karndee Leopairote, Asst. Prof. Dr. Seerong Prichanont , Assoc. Prof. Dr. Parames Chutima , Ajarn Surapong Sirikulvadhana and Ajarn Poom for their useful suggestions.

I wish to acknowledge Thai Government and Silpakorn University for the sponsorship of my education. Thanks to friendship from my friends Ph.D. Student , Jitti, Surachai, Supakanya, Chaiyathuch, Kanya, Sujin, Ronnachai, Thanasan, Aniruth, Isarawit, Varaporn, Siravit and Krisada. I am thankful to my friends at Clemson U., Swaros, Onurai ,Sunarin, Arika, Jaqui, Esengul, Yongjun, Duanmu, Mahdieh, Xiaoyu for friendship and supports. I really appreciate their helps. I cannot forget to thank my KMITNB’s friends, Nattawut, Thidawan, Naratip, Nantakrit and my friends from Chiang Mai U. and Silpakorn U. who always support me. I also would like to thanks my friends from Thai Garment Development Foundation to support me the data about garment industry.

I would like to thank my family and my cousins who always take care me. I would like to thank my parents (Jaroon and Wongduan) who give me breath and also fulfill my life with endless love. A youngest sister would like to thank to my sisters (Asst. Prof. Dr. Pajaree and Dr. Jaruwat) for their supports and endless love. Finally, I would like to thank everything and everyone to make me completed this work.

TABLE OF CONTENTS

	Page
ABSTRACT (THAI)	iv
ABSTRACT (ENGLISH)	v
ACKNOWLEDGEMENTS	vi
TABLE OF CONTENTS	vii
LISTS OF TABLES	xi
LISTS OF FIGURES	xiii
CHAPTER I INTRODUCTION	1
1.1 Introduction.....	1
1.2 Statement of the problem.....	3
1.3 Dissertation objectives.....	4
1.4 Dissertation scope	5
1.5 Dissertation contribution.....	5
1.6 Dissertation methodology	6
1.7 Dissertation organization	7
CHAPTER II LITERATURE REVIEW	8
2.1 A review of assembly line balancing problems	8
2.1.1 Assembly line balancing problems	8
2.1.2 Solving the problems	10
2.2 The assignment problem.....	12
2.2.1 Variations on the assignment problem.....	13
2.2.1.1 Variations in agent.....	13
2.2.1.2 Variations in objective	13
2.2.1.3 An assignment problem with multiple tasks per agent.....	14
2.2.1.4 Others variations in assignment problems	15
2.2.2 Skill levels of workers and cross-training in assignment problems.....	16

	Page
4.5.1 Testing the problems.....	64
4.5.2 The parameters applied in the heuristic.....	65
4.5.3 Performance of heuristics on test problems	68
4.5.3.1 The computational time of the heuristic	69
4.5.3.2 The quality of the solution of the heuristic	74
4.5.3.3 The comparison of the quality of the solution of the heuristic to the heuristic for the constant skill level	82
4.6 Discussion of computational results	85
CHAPTER V CONCLUSION AND FUTURE RESEARCH.....	86
5.1 Conclusion	86
5.1.1 Introduction.....	86
5.1.2 Problem description: the task-worker assignment problem assuming constant skill levels.....	86
5.1.3 Mathematical model of the task-worker assignment problem assuming constant skill levels.....	87
5.1.4 Heuristic for the task-worker assignment problem assuming constant skill levels.....	87
5.1.5 Performance measurement of the heuristic assuming constant skill levels	88
5.1.6 Problem description: the task-worker assignment problem into account learning ability.....	88
5.1.7 Mathematical model of the task-worker assignment problem taking into account learning ability	89
5.1.8 Heuristic for the task-worker assignment problem taking into account learning ability.....	89
5.1.9 Performance measurement of the heuristic taking into account learning ability	90
5.2 Discussion and recommendations.....	90
5.2.1 Task-worker assignment assuming constant skill levels	90

	Page
5.2.2 Task-worker assignment taking into account learning ability	92
5.3 Future research.....	92
REFERENCES.....	94
VITA.....	102



ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

LIST OF TABLES

	page
Table 1.1. The task processing time for each worker (sec.)	2
Table 3.1 The number of feasible assignments or complexity of the problem	29
Table 3.2 Notations	30
Table 3.3 Problem Size.....	38
Table 3.4 The difference between the initial upper bound and initial lower bound.....	40
Table 3.5 The computational time of the heuristic and exact solution (sec.)	43
Table 4.1 The processing time of 5 tasks, 3 workers (A, B, C) and 3 items (sec.)	48
Table 4.2 Notations	50
Table 4.3 the summation of task processing time	58
Table 4.4 Problem Size.....	65
Table 4.5 The quality of the solution and CPU. of test when the parameters are varied	67
Table 4.6 CPU. time of the problem : No. tasks = $3 \times$ No. workers , 100 items.....	69
Table 4.7 CPU. time of the problem : No. tasks = $3 \times$ No. workers , 300 items.....	70
Table 4.8 CPU. time of the problem : No. tasks = $4 \times$ No. workers , 100 items.....	71
Table 4.9 The percentage difference of the problem : No. tasks = $3 \times$ No. workers , 100 items	76
Table 4.10 The percentage difference of the problem : No. tasks = $3 \times$ No. workers , 300 items	77
Table 4.11 The percentage difference of the problem : No. tasks = $4 \times$ No. workers , 100 items	78
Table 4.12 The percentage deviation of the heuristic for constant skill level from the heuristic solution	83

LIST OF FIGURES

	page
Figure 1.1 The research methodology	7
Figure 3.1 the solutions of grouping tasks to workstation and the assignment between workstations and workers.....	28
Figure 3.2 The overview of the heuristic.....	33
Figure 3.3 The computational time when weight between UB LB are varied (sec.) ...	41
Figure 3.4 The percentage difference of cycle time between heuristic and exact solution (%)	42
Figure 3.5 Computational time of the heuristic when the skills of worker are varied	43
Figure 3.6 The percentage of the difference between the optimal solution or the best found solution and the heuristic solutions (%).	44
Figure 3.7 The comparison of the percentage difference between the proposed heuristic and 2-stage heuristic solutions (%).....	45
Figure 4.1 Makespan from an assignment solution.....	49
Figure 4.2 An expression of the Makespan	53
Figure 4.3 Dijkstra's algorithm	55
Figure 4.4 the overview of the heuristic	57
Figure 4.5 An example of LB and UB.....	58
Figure 4.6 (A) Nodes of the group of tasks	59
Figure 4.6 (B) Feasible nodes of task-workstation assignment.....	59
Figure 4.6 (C) Nodes for Modified Dijkstra's algorithm	59
Figure 4.7 The pseudo code of Modified Dijkstra's algorithms.....	61
Figure 4.8 An example of the Modified Dijkstra's algorithm.....	63
Figure 4.9 An example to determine idle and sub-makespan.....	63
Figure 4.10 An example of an improvement of makespan.....	66
Figure 4.11 The quality of the solution of the test when the parameters are varied.....	68
Figure 4.12 CPU time of the test when the parameters are varied (sec.)	68

Figure 4.13 CPU Time of optimal sol. when No. Tasks increase (sec.).....	71
Figure 4.14 CPU Time of optimal sol. when No. Items increase (sec.).....	72
Figure 4.15 CPU Time of optimal sol. when No. learning slope varies (sec.).....	72
Figure 4.16 CPU Time of heuristic sol. when No. Tasks increase (sec.).....	73
Figure 4.17 CPU Time of heuristic sol. when No. Items increase (sec.).....	73
Figure 4.18 The fraction of CPU.Time (%).....	74
Figure 4.19 The percentage difference 7w28t100i (%).....	78
Figure 4.20 The percentage difference 7w21t100i (%).....	79
Figure 4.21 The percentage difference 7w21t300i (%).....	79
Figure 4.22 The percentage difference 8w32t100i (%).....	80
Figure 4.23 The percentage difference 8w24t100i (%).....	80
Figure 4.24 The percentage difference 8w24t300i(%).....	81
Figure 4.25 The percentage difference when No. Tasks and No.Items increase (%) ..	82
Figure 4.26 The percentage difference $\text{diff}_{\text{con_heu}}$ when No. Tasks increase (%).....	84
Figure 4.27 The percentage difference $\text{diff}_{\text{con_heu}}$ when No. Items increase(%) ..	84

CHAPTER I

INTRODUCTION

1.1 Introduction

Generally, in the garment industry, workers have different skill levels due to their experience and training. In addition, due to the high turnover rate in the industry, teams often consist of both new workers and experienced workers. For this reason, a high variation of operation time between workers occurs. The question is how to balance the line and assign workers of varying skill levels to workstations to maximize productivity.

Both the assembly line balancing problem and worker assignment problem are complicated by themselves and, as such, are typically resolved using a two-stage heuristic. First, the assembly line balancing problem is addressed by aggregating tasks using predetermined time standards. With this established, workers are assigned to the tasks. In the Thai garment industry, it is common for one group to do the line balancing and then a line supervisor to assign the workers to the grouped tasks. This basic approach is very poor for a number of reasons, the most serious of which is that grouping tasks for the line balancing effort does not consider the differences in worker skills whereas the supervisor who makes the actual assignments certainly does. This practical observation points to the general problem and defines the research problem addressed here, namely, investigation of integrated line balancing by task grouping and worker assignment to task groups with consideration of varying skill levels between workers for labor-intensive assembly operations.

To provide further motivation for addressing this problem, consider the following example. There are 3 workers assigned to 5 tasks, the task processing time for each worker and the average processing time are given in Table 1.1. The typical two-stage heuristic used in practice would apply average processing time to group tasks for line balancing. If the objective is to minimize the maximum processing time of the workstations, the assignments would be task 1 to workstation 1; tasks 2 and 3 to

workstation 2; and tasks 4 and 5 to workstation 3. This yields a maximum processing time of 5 using the average times of the workers. The second phase assigns one worker to each workstation with the objective of improving the solution. This yields the assignment of worker A to workstation 1, worker B to workstation 2 and worker C to workstation 3. The maximum processing time is 5 at workstation 2.

Table 1.1. The task processing time for each worker (sec.)

	Worker A	Worker B	Worker C	Average time
Task 1	5	2	5	4
Task 2	3	1	2	2
Task 3	4	4	1	3
Task 4	2	2	2	2
Task 5	5	2	2	3

However, if the both problems are integrated, worker skills are considered at the same time as task grouping. The optimal solution will be 4. The solution of task-workstation assignment is {1 2, 3, 4 5} and the solution of worker-workstation assignment is {B, A, C}. The maximum cycle time is $\max \{3, 4, 4\} = 4$. It should be noted that the quality of the solution solved in this way is better than the solution in the case of grouping tasks without considering worker skills and assigning groups of tasks to workers later. As such, there appears to be a strong potential for significant improvement by addressing the line balancing and worker assignment problems in an integrated fashion rather than sequentially. In this study, we are interested in developing a methodology for assigning tasks to workers of varying skill levels.

Not all cases however will have this magnitude of error between the integrated and two-stage approaches in practical application. The comparison of the quality of the solution between the two approaches can be questioned. What is the effect of the varying skill levels on the quality of the solution between the methods used in practical application versus an integrated approach to assembly line balancing and worker assignment?

In the fashion industry, new product styles are launched more frequently than in the past with smaller lot sizes due to increased market competition. Consequently, a trend in garment production in Thailand has emerged shifting from mass production to small lot production since it is more flexible and responsive compared to mass production. Garment production is labor intensive and the production rate mainly depends on worker skills. With different experience and training, each worker has different skill levels and learning ability. Generally, people learn and improve their performance by repeating operations, and as a result, they will require less time to produce the succeeding unit or gain proficiency with the repetition of the same task. This is called learning behavior.

In mass production, learning behavior is usually not considered. A constant production rate assumption is always assumed in developing a task-worker assignment since the learning period is only a small part compared to a whole production period. However, in the fashion industry, since new product styles are launched more frequently and lot sizes are smaller, the learning period becomes a more substantial part of production time. A task-worker assignment with a constant production rate assumption may not directly apply since it may not provide the optimal solution in practice. For this reason, learning should be considered in a task-worker assignment in the fashion industry.

In this study we are interested in developing a methodology of assigning tasks to workers of varying skill levels taking into account learning. What is the effect of applying the constant skill level in situations that account for the workers learning ability on the quality of the solution?

1.2 Statement of the problem

In this dissertation, an integrated approach to assembly line balancing and worker assignment is studied. The system considered in this problem is an assembly line which is a set of sequential workstations. The problem consists of a simultaneous solution to a double assignment: tasks to workstations and workers to workstations. A workstation consists of a worker who operates the assigned tasks. There are i identical items. An item is processed through a number of tasks from the first task to the last task until the item is completed as a finished product. A worker will operate

on an item as soon as she/he has finished their work on the current item and has released it into buffer spaces before the next workstation. All items are processed along the same route which passes through all workstations. The problem assumes that the workers have multiple skills so they are able to do more than one task. The skill levels of the workers are different. The task processing time depends on the skills of workers who execute those tasks. This study looks at both the constant skill level and skill level with learning ability.

A task-worker assignment method where all tasks must be assigned to workers was developed. In it, each worker is assigned to at least one task and some workers can have multiple assignments. Moreover, a task cannot be split and assigned to more than one workstation. In the study, tasks are ordered in a series and consecutive tasks are only allowed in the multiple assignments because of the continuous flow of the production line. This problem also includes the following assumptions.

- 1) The number of tasks is greater than the number of workers.
- 2) The task processing time is given.
- 3) Other learning factors among the tasks at the same workstation are not considered e.g. the task similarity.
- 4) There is unlimited buffer space before each workstation or worker.
- 5) At the start of production, there is no work in progress in the line.
- 6) Learning ability depends on each worker.

1.3 Dissertation objectives

1.3.1 The objective of this research is to develop an efficient heuristic to solve the problem of an integrated approach to assembly line balancing and worker assignment assuming constant skill levels of workers in order to minimize cycle time or to minimize the maximum processing time of the workstations.

1.3.2 The objective of this research is to develop an efficient heuristic to solve the problem of an integrated approach to assembly line balancing and worker assignment taking into account learning ability of workers in order to minimize makespan or the completion time.

1.4 Dissertation scope

This study focuses on a fixed task-worker assignment for which there is a solution for the assignment problem. In the problem, the tasks are ordered in a series since it is an assembly line for a finished product. For learning behavior, the Log-Linear model was applied to represent the learning ability of each worker. Based on the Log-Linear model, if t_1 and t_n represent the task processing time of the first and the n^{th} item, and using ϕ in terms of learning slope, $t_n = t_1 \cdot n^{(\log \phi / \log 2)}$ (Wright, 1936).

The problem assumes that the task processing time depends on the learning ability of the worker who operated the task. Furthermore, the study sets the task in discrete processing time which was generated from the learning model. If a worker is assigned to more than one task, we use the sum of the task processing time that he/she performs to represent processing time of the combined tasks. We assume that the task similarity between the tasks is ignored.

1.5 Dissertation contribution

An integrated approach to the assembly line balancing and worker assignment was developed in this dissertation. The problem was that the quality of the solution of the assignment in a practical application or a two stage heuristic may not be appropriate when there is high worker skill variation. The result of the study confirms the existence of this problem and can serve to increase the level of awareness in the industry of the effect of the varying skills of workers on production rates.

It would be beneficial for the industry to take into account the impact on the performance of the production line when assuming constant skill levels in situations where worker learning ability is a factor. This can be a guideline for industry for developing a more efficient assignment process.

Mathematical models of an integrated approach to the assembly line balancing and worker assignment assuming constant skill levels and mathematical models of the problem accounting for learning ability were proposed in conducting this research.

Furthermore a heuristic was developed to solve the problem of constant skill level and its quality of the solution and computational time was compared to the solution from a commercial solver. It was found that in many cases, the heuristic can determine the optimal solution. We used the solution from the first heuristic to limit search space for the second problem.

The heuristic for solving the task-worker assignment problem accounting for learning ability was proposed and its quality of solution and computational time compared to the solution from a commercial optimization solver. It was found that in some cases, the heuristic achieves the optimal solution. We believe that the results of this dissertation can be used a guideline for other researchers to further develop similar heuristics or apply them to other problems.

1.6 Dissertation methodology

This section addresses the dissertation methodology. Figure 1.1 shows the dissertation methodology.

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

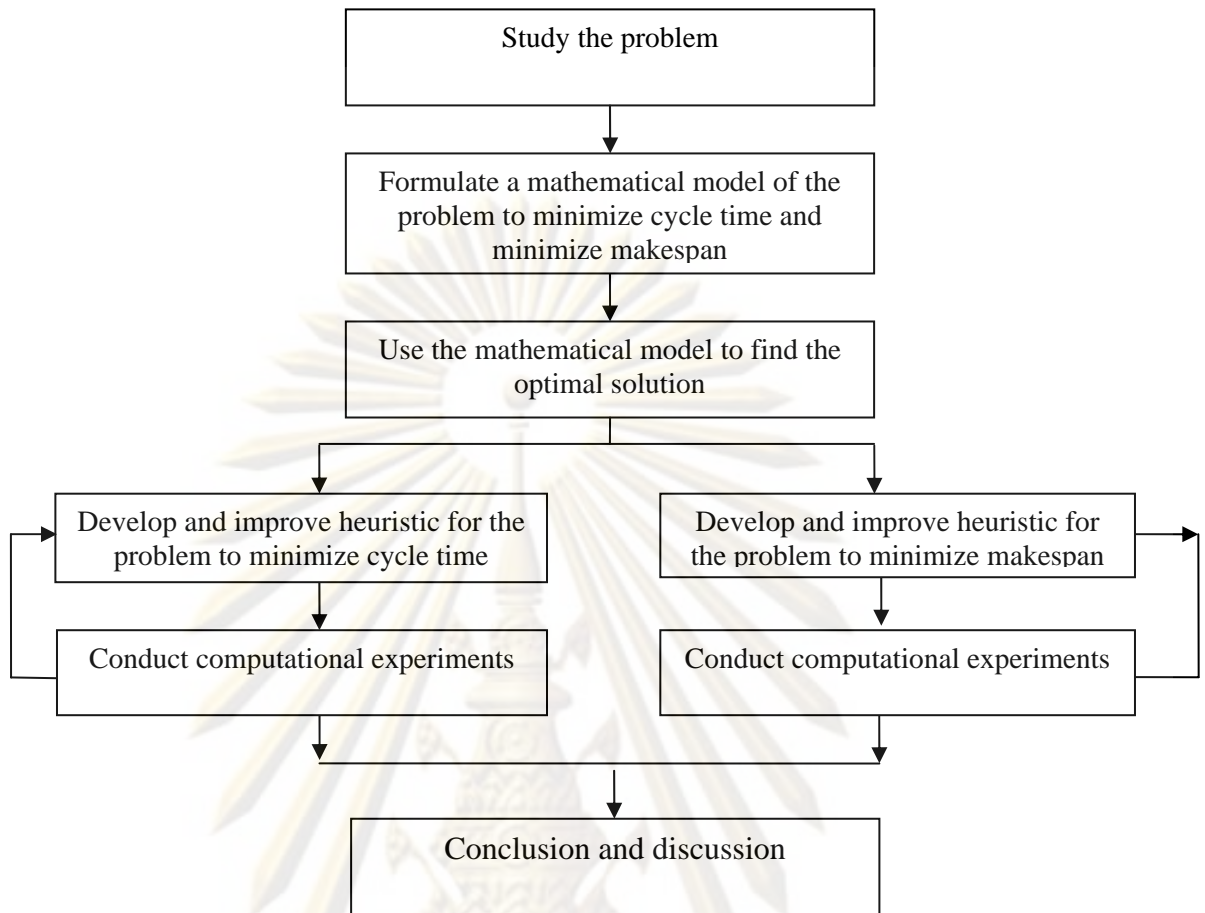


Figure 1.1 The research methodology

1.7 Dissertation Organization

The outline of this dissertation is as follows. The relevant literature is reviewed in Chapter II. In Chapter III, task-worker assignment assuming constant skill level is formulated and a heuristic to solve the problem and a performance measurement of the heuristic and the computational results is proposed. In Chapter IV, the task-worker assignment accounting for learning ability is formulated and a heuristic to solve the problem and a performance measurement of the heuristic and the computational results is proposed. Finally, the conclusion and suggestions for future research are presented in Chapter V.

CHAPTER II

LITERATURE REVIEW

This chapter is organized as follows: Firstly, a review of assembly line balancing problems, which consists of the problem description and solving method. Secondly, assignment problems which consist of the problem description, variations on assignment problems and a survey of problems related to worker assignment. Thirdly, a survey related to assembly line balancing problems and assignment problems with a constant skill level and finally, a survey related to assembly line balancing problems and assignment problems with learning ability.

2.1 A review of assembly line balancing problems

An assembly line consists of a series of workstations. One objective in designing a flow line is to attempt to allocate equal amounts of work to each workstation (Wile, 1972). This is known as line balancing. The problem of line balancing is to distribute the total work content to the workstations in the line, such that idleness of resource at each station is minimized (Gavett, 1968). The cycle time is the available time for an operator to complete his unit of work at his station. The bottleneck workstation is the workstation in which work content is equal to the cycle time.

2.1.1 Assembly line balancing problems

The simple assembly line balancing problem (SALBP) is one of assigning the tasks to stations according to some criteria. In the problem, the tasks have a precedence requirement, so the tasks cannot be processed in an arbitrary sequence. All tasks must be processed. The task processing time does not depend on the workstation, so the task processing time is same for an assignment at any workstation. The total line is considered to be serial. SALBP is designed for the mass-production of a single product. Mainly, there are two types of simple assembly line balancing problems (SALBP). The objective of SALBP-I is to minimize the

number of stations along the line and the cycle time is given. The objective of SALBP-II is to minimize the cycle time or maximize the production rate and the number of stations is given (Baybars, 1986). There are many generalized assembly line balancing problems which occur in a system with several products or different models and different line layouts. For example, more than one worker operates the tasks or the parallel workstations. Furthermore, the system has assignment restrictions. For example, some tasks have to be assigned to the same workstation or incompatible tasks have to be assigned to different workstations.

The problem addressed in this dissertation is the SALBP-II type. Since the number of workers is known, we want to minimize the cycle time. However, this problem in the study deviates from the original SALBP since workstations are non-identical. A worker who performs the tasks in a workstation has different levels of performance.

The mathematical formulation of the SALBP-II problem is the following:

Index

- S = a set of workstations , $s \in S$,for $s = 1, \dots, k$
 J = a set of tasks , $j \in J$,for $j = 1, \dots, o$
 ARC = a set of arcs from (j, u) which is an arc from task j to u to symbolize that j is the immediate predecessor of u

Parameter

- k = the number of workstations or the number of workers
 o = the number of tasks
 t_j = the processing time of task j .

Variable

- x_{js} = $\begin{cases} 1 & \text{if task } j \text{ is assigned to workstation } s \\ 0 & \text{otherwise} \end{cases}$
 $Cycle$ = Cycle time

$$\text{Minimize } Cycle \tag{2.1}$$

Subject to

Cycle time constraint:

$$\sum_{j \in J} t_j \times x_{js} \leq Cycle \quad \forall s \quad (2.2)$$

$$\sum_{s \in S} x_{js} = 1 \quad \forall j \quad (2.3)$$

$$\sum_{s \in S} s \times x_{js} \leq \sum_{s \in S} s \times x_{us} \quad \forall (j, u) \in ARC \quad (2.4)$$

Constraints 2.2 are to ensure that the cycle time is not exceeded by the workstation time of any workstation. Constraints 2.3 are to ensure that each task is assigned to just one station. Constraints 2.4 represent the precedence constraints to ensure that no task is assigned to an earlier station than its predecessor.

2.1.2 Solving the problems

There are a variety of procedures for solving SALBP-I, whereas only a few procedures solved SALBP-II directly. Most research has applied SALBP-I to solve SALBP-II by increasing the cycle time until a balance is achieved. The initial lower bound or the minimal cycle time is determined, and then an assignment of tasks to workstations following the precedence constraints using SALBP-I method is applied. The trial cycle time is successively increased until a feasible solution found. This procedure is called the iterated method (Baybars, 1986; Scholl and Klein, 1999; Scholl and Becker, 2006). The search method starts with the lower bound, and the cycle time is successively increased by one until a feasible solution is found. This is called the Lower Bound Method. The lower bound can be determined by several methods. The lower bound (LB) for SALBP-II can be obtained by omitting the precedence constraints. Let t_{\max} be the maximum task time, t_{sum} be the sum of task times for all tasks and k be the number of workers, $LB = \max \{t_{\max}, t_{\text{sum}}/k\}$. For another way to set the lower bound, Talbot and Patterson (1984) have restricted the possible assignment of each task to a station interval which is bounded by the concept of an earliest and latest station. The lower bound is the minimal cycle time in which the earliest station is less than or equal to the latest station for all tasks. Pastor and Ferrer (2008) developed a mathematical model for SALBP based on the concept of a feasible assignment from the earliest to latest interval. Sprecher (1999) and Klein and Scholl (1996) have calculated the lower bound using different methods and selected the best lower bound among the results.

Not only does the search start with the lower bound, but it also starts with the upper bound, and the cycle time is successively decreased by a step until a feasible solution is found or it equals the lower bound. It is called the Upper Bound Method. Furthermore, the search can be in the interval [LB, UB]. This is called a binary search. The interval is successively subdivided into two sub-intervals by selecting the mean value, $(LB+UB)/2$. If a feasible solution is found, the UB is set to the maximum station time in the corresponding solution. Otherwise, LB is set to c plus step. These are the general search procedures applied in the previous research. The exact algorithms for solving SALBP-II are based on the branch and bound principle. Furthermore, several rules were developed in order to reduce computation and improve the search process (Askin and Standridge, 1993).

For example, only maximal station loads have to be considered. A station is termed maximal if no task can be assigned to it without violating the precedence and the cycle time constraints, i.e. a workstation should never close while “fittable” tasks remain. A fittable task is an unassigned task that can be completed in the remaining idle time of the station. Moreover, for the dominance rule, suppose there is a station where one of its tasks, u , could be feasibly replaced by a longer task, v , and all the successors of u must also follow v . If v is substituted for u , the remaining workload is reduced without losing any possible sequence completions. Suppose we have three tasks 1, 2, 3 with task time (2, 4, 2) respectively and task 1 and task 2 are completed before their successors task 3 can be started. In this case, task 2 dominates task 1 since task time of task 2 > task time of task 1 and all of task 1’s successors. Let cycle time be $c = 4$. If we place task 1 first, the solution is (1, 2, and 3). The workstation time will be {2, 4, 2}. However, if b is placed first, the solution is (2)(1,3). The workstation time will be {4, 4}. The dominated task is ignored. The solution is better than the previous solution. The partial sequence that places the dominated task is fathomed. The dominance rule is applied in many heuristics. Furthermore, the bound violation is a rule that determines the upper bound of the largest workstation to which each task can be assigned. After a workstation is assigned, the unassigned task is checked. If any such task has an upper bound less than or equal to the order of the assigned workstation, then this partial solution is fathomed.

Regarding the assembly line balancing problem, when the operation time for every task is different depending on who executes the task, Miralles, et al. (2008) applied branch and bound using the bound violation rule. Finally, the rule of

excessive idle time is the rule that uses the total idle time to fathom the partial sequence. Let the total idle time be $k \times c - t_{\text{sum}}$. Thus whenever the cumulative idle time exceeds $k \times c - t_{\text{sum}}$, the partial solution is fathomed. The fathom rule has been applied in many algorithms. Furthermore, there are many researchers who have proposed heuristics for SALBP-II e.g. meta-heuristic (Fatih Ugurdag et al.,1997; Liu, et al., 2008; Tasan and Tunali ,2008). Liu, et al., (2008) proposed two-stage heuristics for SALBP-II. First, the initial solution was determined then it was improved by swapping tasks among workstations.

In this study, we are interested in applying the maximal station loads rule to limit the number of tasks assigned to a worker within the trial cycle time, then determine the feasible assignment from the alternative of the tasks. After the groups of tasks are generated based on the maximal station load rule, the feasible assignments from matching the group of tasks that validate the assignment requirement are searched. With this method, the number of the alternative groups of tasks will be reduced and the feasible solution will be generated.

The UB and LB will be developed. To determine lower bound, we believe that the lower bound considering the precedence constraints is better than the lower bound omitting the precedence constraints.

2.2 The assignment problem

The classical assignment problem is to find a one-to-one match between n tasks and m agents; the objective is to minimize the total cost of the assignments. The mathematical model for the classic assignment problem may be given as:

$$\text{Minimize } z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$$

Subject to

$$\sum_{i=1}^n x_{ij} = 1 \quad j = 1, 2, \dots, n \quad (2.5)$$

$$\sum_{j=1}^n x_{ij} = 1 \quad i = 1, \dots, m \quad (2.6)$$

$$x_{ij} = 0 \text{ or } 1$$

;where $x_{ij} = 1$ if agent i is assigned to task j , 0 if not, and c_{ij} = the cost of assigning agent i to task j .

2.2.1. Variations on the assignment problem

Pentico (2007) and Burkard (2002) proposed a survey paper of the variation of assignment problems. For example, an assignment problem that has variations in agent and objective, and an assignment problem with multiple tasks per agent, among others, were proposed.

2.2.1.1 Variations in agent

In practice, agents or workers have different skill levels. The industry sets a rating for the workers e.g. A, B and C and only the worker who is qualified is allowed to perform a given task (Dell'Amico and Martello, 1997). In assigning workers to machines, there are multiple alternatives in the assignment. The assignment can involve only a subset of workers and machines to be assigned (Prins , 1994).

For the proposed assignment problem in this dissertation, we assume that the workers have different skill levels; however it is assumed that all workers should be assigned in the problem.

2.2.1.2 Variations in objective

The original minimizing total cost has been modified based on the situation. For example, Martello, et al. (1984) have pointed out the problem of minimizing the difference between the maximum and minimum assignment values, which is called the balanced assignment problem. In addition, Duin and Volgenant (1991) have proposed the minimum deviation by minimizing the difference between the maximum and average assignment costs. For example, in a cutting problem, the objective is to minimize the waste from cutting a standard size edge-piece down to the individual sizes.

The bottleneck assignment problem differs from the classic assignment problem in that the objective changes from minimizing the sum of the costs of assigning tasks to agents to minimizing the maximum of costs of the assignments. An example of the bottleneck assignment problem from Lev and Weiss (1982) is as follows: A foreman takes a four-person work crew from Philadelphia to Atlantic City in order to fix some equipment. The four workers have the ability to use all machines. Due to the different skill levels, the length of time will vary for each worker to fix each machine. The crew will return to Philadelphia together. Therefore they will leave Atlantic City when the last crew member finishes. Thus the foreman needs to assign the tasks so that the largest task time is minimized.

The objective of the mini-max formulation is the same as the objective in the assembly line balancing problem. It is applied to the Simple Assembly Line Balancing Problem (SALBP-II) with the objective of minimizing cycle time or minimizing the maximum workstation time (Scholl, 1999). Generally, mini-max in time will be maxi-min in production rate. Suer (1998) has proposed the maxi-min formulation to maximize the minimization of the assembly rate for designing parallel assembly lines.

2.2.1.3 An assignment problem with multiple tasks per agent

There is one type of problem called the Generalized Assignment Problem (GAP). In the model, an agent may be assigned more than one task. The generalized assignment problem is the problem of assigning each task specifically to one agent, so the total cost of processing all tasks is minimized and no agent exceeds its resource capacity. Applications of the GAP appear in many fields such as vehicle routing, fixed charge location problems, grouping and loading for flexible manufacturing systems, scheduling projects, allocating storage space, and designing communication networks.

The GAP with m agents and n tasks can be formulated as an integer programming problem by defining the zero-one decision variables x_{ij} , where $x_{ij}=1$ if task j is assigned to agent i , and $x_{ij}=0$ otherwise:

$$\text{Minimize} \quad \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$$

Subject to

$$\sum_{j=1}^n a_{ij} x_{ij} \leq b_i \quad i = 1, \dots, m \quad (2.7)$$

$$\sum_{i=1}^m x_{ij} = 1 \quad j = 1, \dots, n \quad (2.8)$$

$$x_{ij} \in \{0,1\} \quad i = 1, \dots, m; j = 1, \dots, n$$

The GAP assumes that there is one resource available to the agents. Gavish and Pirkul (1986) have proposed the Multi-Resource Generalized Assignment Problem (MRGAP) which consumes several resources processed by the agents. This problem has been applied to the distribution of petroleum products; for example, an oil company intending to minimize the costs of delivering petroleum products (super, unleaded petrol etc.). Shtub and Kogan (1998) have presented an extension of MRGAP to the case where demand varies over time and capacity assignments are dynamic.

For the proposed assignment problem in this dissertation, multiple assignments are allowed for a worker. However it is different from GAP since the resource capacity is not limited. Furthermore, this dissertation considers that the assignments should validate the precedence constraint.

2.2.1.4 Others variations in assignment problems

Multi-dimensional assignment problems match the members of three or more sets. For example, the problem could be matching jobs with workers and machines or assigning students and teachers to classes and time slots. Franz and Miller (1993) have discussed the multi-period assignment problem for assigning medical residents and rotations at a teaching hospital. The objective was to maximize the expressed preferences of the residents. Kouvelis and Yu (1997) have presented a solution for the assignment problem which contains uncertainties.

2.2.2 Skill levels of workers and cross-training in assignment problems

Workers have different skill levels according to experience, capability, knowledge and background. Skill levels are represented by production rate, percent of efficiency and performance (Spragg et.al, 1999). For an assignment based on the skill level, many researchers have defined skill level in different patterns. Hassamonts (2004) has set skill levels with ratings. For example the “A” rate is the rate that a worker can complete a standard task at least 20% faster than the specified standard time. Chan, et al. (1997) rated skill ranging from 0 (unskilled) to 1.5 (fully – skilled) which depends on the performance of the worker performing a sewing operation. Song, et al. (2006) defined the operator’s efficiency as the ratio of the garment quantity of a piece finished by an operator divided by time. The applications of skill levels also appeared in planning (e.g. workforce planning, staffing), promoting (evaluating performance of a worker), motivating workers and using data of assignment.

For the flexibility of the production line, multi-skilled workers or cross-trained workers are required. It is recognized as a tool for increasing production flexibility when addressing changes in demand, worker assignment, and absenteeism. Labor flexibility has a positive effect on operational performance indicators, such as the throughput time and the delivery performance of jobs (e.g. Treleven, 1989). Cross-training also increases the possibility that workers may help each other and share their workloads. Moreover, cross-training can mitigate turnover because flexible workers can more easily replace the workers who leave, and can improve motivation. For example, some cross-trained workers feel that they have experienced professional growth.

Multi-functionality is a benefit for organizing the workforce (Zulch et al., 2004). The multi-skilled worker applies both in the industry and service sectors e.g. power stations (Eitzen and Panton, 2004), production lines (Bokhorst et al., 2004), and hospitals (Brusco, 1998). Furthermore, a manager or supervisor who has multiple skills may be set to be a floater to attend to the operation when it is backed up (Hopp, 2004). Farrar (1993) analyzed the potential performance and benefits achievable with a floater in serial production systems.

However, total flexibility of the workforce is not desirable in practical situations. The requirement of training all workers for all machines would be very costly. Moreover, a high level of labor flexibility may also involve considerable productivity loss due to the shift of workers between machines (Slomp et al., 2005). Wong, et al. (2005) studied the impact of the different levels of skill inventory of workers on the assembly. The number of skills was investigated in order to find out the optimal number of task skills that an operator should possess in the apparel assembly process.

Several researchers studied and developed cross-training policies in order to determine the distribution of workers' skills. Yang (2007) has compared a set of cross-training policies studying the different numbers of cross-trained workers (i.e. one, two, three or four from each department), additional skills per cross-trained worker, and additional machines (i.e. four workers / five machines having worker-to-machine ratios 80%) to analyze the performance of the policies. Inman (2005) has compared policies of total cross-training, reciprocal pairs (each cross-trained worker serving the paired unit) and chained cross-training (one worker trained some units linking the units in a chain). Cross-training has been investigated by many researchers such as Brusco and Johns (1998) who studied characteristics of cross-training policies in the number of work activity categories and the level of productivity. Jordan, et al. (1995) have stated that chained cross-training results in a robust change in workload characteristics. Inman, et al. (2004) applied chaining to cross-training assembly line workers to mitigate the impact of absenteeism.

2.2.3 Surveys related to assignment problems

Suer (1996) has addressed the problem of finding the optimal manpower assignment and cell loads simultaneously. A single workstation, or "cell", consists of a number of operators assigned to each task type. A two-phase hierarchical method is proposed using two models of mixed integer and integer programming formulations. The first model determines the alternative cell configurations in order to maximize the production rate. The second model is to allocate worker assignment and load to cell to maximize the production of the cell and utilize the least number of workers. Suer and Bera (1998) have expanded the previous models of Suer (1996) by allowing multiple products to be assigned to

multiple cells (lot-splitting). This model does not address the operator's skill level for each task.

Nakade and Ohno (1999) have considered the optimal worker allocation problem in a U-shaped production line. The multi-skilled worker operates multiple machines and visits each machine once for each unit of production. They derived the lower bound of the number of workers under the required cycle time and proposed an algorithm for finding an optimal allocation of workers to machines in order to minimize the cycle time using the minimum number of workers. Nevertheless, they do not consider the worker allocation problem when there are different skill levels among workers.

Slomp and Molleman (2002) have investigated the impact of cross-training on team performance under conditions where there is a fluctuating demand and supply of human resources. A task assignment heuristic is used for comparing the cross-training policies.

Slomp and Bokhorst (2005) have proposed a model that considers trade-offs between training costs and the workload balance among workers in a manufacturing cell. The integer programming model was proposed to calculate which workers have to be trained for which machines. They used a bottleneck worker to determine workload balance. The assumption of the problem is based on the idea that a bottleneck worker is used for determining the efficiency of the manufacturing cell.

Futatsuishi, et al. (2002) addressed the problem of minimizing the total elapsed time from the start up time to the completion of a job in an environment where a single worker needs more than one skill and one task can be processed by several persons.

Norman, et al. (2002) have proposed a model accounting for human skills. The skill levels of workers are permitted to change by providing them with additional training. A mixed integer programming model to assign workers to tasks in manufacturing cells is proposed in order to maximize the effectiveness of the organization (function of productivity, output quality and training cost).

Corominas, et al. (2006) have studied the assignment of tasks to the members of a multi-functional staff in a work center. The problem focuses on service industries that train workers on every task. The different tasks require varying levels of attention and responsibility. The number of consecutive periods operated by

workers should fall within the specified interval limited by a minimum and a maximum number of periods working at each type of task.

Campbell and Diaby (2002) have proposed an assignment heuristic for allocating cross-trained workers to multiple departments. Each worker has different capabilities for working in each department.

Caron, et al. (1999) have studied assignment with seniority and job priority constraints in the daily scheduling of nurses in a hospital.

Askin and Chen (2006) have studied dynamic task assignment for throughput maximization with worksharing. Two types of worksharing can be found: Dynamic assembly-Line Balancing (DLB) and Moving Worker Modules (MWN). MWM applications usually have more machines than workers. Workers carry work pieces along the line within zones and share use of machines. DLB matches machines and workers with some tasks assigned to a designated worker, which are called fixed tasks. Other tasks can be performed by either of an adjacent pair of workers. These are called shared tasks. A worker chooses to either pass on a job with the shared task undone or complete the shared task, according to specific rules.

2.3 Surveys related to assembly line balancing problems and assignment problems concerning constant skill level

Although the integrated line balancing and worker assignment problems appear to have tremendous relevance in practice, there are few references indicating past research in this area. Hassamontr (2004) proposed two-phased heuristics for assembly line balancing with operator's skill and machine constraints, which is basically the approach described above. The first phase assigns operations to workstations using standard times and the second phase assigns workers to workstations. Wong, et al. (2005) developed a line balancing technique using Genetic Algorithms (GA) in which each worker has an efficiency factor related to skill level. The required number of workers is computed and worker assignments are then generated using GA. Song, et al. (2006) proposed a recursive algorithm that first determines the number of workers for each task that is required, then identifies the skill(s) of each worker, and finally makes the assignments.

An extremely interesting paper by Miralles, et al. (2008) addresses grouping and assigning together considering worker skill. Their approach uses mathematical programming along with a branch and bound solution procedure. There are a number of key differences between this work and the problems under consideration here because their research focuses on designing a work environment that helps disabled workers develop capabilities where some workers cannot operate some tasks. Then Chaves (2009) proposed a hybrid meta-heuristic to solve the problem.

Chen et al., (2009) addressed the problem of assigning tasks to workstations and assigning machines in the workstation in order to balance the load. However all machines are assumed to have the same performance. Assignment restrictions have been added to the assembly line balancing problem. Scholl, et al. (2010) examined the assembly line balancing problem with assignment restrictions including task restriction, resource restriction and workstation restriction. For example, incompatible tasks must be assigned to different workstations and an assignment should not exceed the available space. However, the difference in task processing times between workstations is not considered. In regard to multi objectives, Zhang, et al. (2008) examined the problems of minimizing cycle time, the variation of workload and the total cost. The problem is modelled in a non-linear approach and solved by a genetic algorithm. Zhang and Gen (2009) formulated a non-linear model for mixed – model assembly line balancing and solved the problem using a genetic algorithm with the objective of minimizing cycle time, increasing the line efficiency and reducing the total cost.

Corominas, et al. (2008) proposed a process of rebalancing a motorcycle-assembly line considering two groups of workers: skilled and unskilled workers. The skilled workers can perform all the tasks, whereas the unskilled workers can only perform a subset of tasks. The unskilled workers will take longer to perform tasks than the skilled workers. However, the study assumed that the task processing time of workers in the same group (skilled and unskilled) is equal.

2.4 Surveys related to assignment problems concerning learning ability

Generally, people will learn and improve by repeating operations. They will require less time to produce succeeding units. This is learning behavior and can be studied and represented by a mathematical model (Wright, 1936, Dar-El, 2000). Learning is time-dependent. In this dissertation, we focus on the Log-Linear model. Based on the Log-Linear model, if t_1 and t_n represent the task processing times of the first and the n^{th} item, and use ϕ in terms of learning slope, $t_n = t_1 \cdot n^{(\log \phi / \log 2)}$ (Wright, 1936). The property of the learning phenomenon is that whenever the total quantity of units produced doubles, the time per unit to produce the unit decreases by a constant rate (known as learning rate) (Sumanth, 1985). For example, if the time taken to produce the first unit is 10 hours, and if the learning rate is 80 percent, then the time taken to produce the second, fourth, eighth, and sixteenth units are as follows:

$$\begin{aligned} \text{Second unit} &= 0.8 \times 10 &&= 8 \text{ Time units.} \\ \text{Fourth unit} &= 0.8 \times 8 &&= 6.4 \text{ Time units.} \\ \text{Eighth unit} &= 0.8 \times 6.4 &&= 5.12 \text{ Time units.} \\ \text{Sixteenth unit} &= 0.8 \times 5.12 &&= 4.096 \text{ Time units.} \end{aligned}$$

In general, the percent rate of learning =

$$\frac{\text{Cumulative average/unit at a given level of production}}{\text{Cumulative average time/unit at half the given production level}}$$

For example, time taken for eighth unit / time taken for fourth unit = $5.12/6.4 = 0.8$

The learning curve is given by a hyperbola of the form:

$$t_n = t_1(n)^s$$

where

t_n = time to produce the n th unit

t_1 = time to produce the first unit

n = unit number

s = slope of the learning curve when it is represented on the

log scale

$$= \log \text{ of learning rate} / \log 2$$

$$= \log r / \log 2 : r = \text{learning rate}$$

$$\text{Since learning rate } (r) = t_n / t_{n/2} = \frac{t_1(n)^s}{t_1\left(\frac{n}{2}\right)^s} = (2)^s$$

$$\text{Therefore, } s = \frac{\log r}{\log 2}$$

$$\text{,and } t_n = t_1 \cdot n^{(\log r / \log 2)}$$

Many factors influence learning e.g. job complexity, the number of repetitions, previous experience and training (Dar-El, 2000). Learning is addressed in many studies related to production e.g. scheduling (Mosheiov, 2001), assembly line balancing (Chakravarty, 1988), and allocation or worker selection (Nemhard, and Osothsilp, 2005). Workers vary in learning which is affected by many factors: individual ability, individual variability, financial incentives, organizational norms and constraints, training and the nature of the social environment (Uzumeri and Nemhard, 1998). Learning data can be kept and analyzed due to systems support. Many organizations have installed data acquisition systems to record detailed production and quality data to enable product tracking, quality control, and piece-rate wage tracking (Nemhard et al., 2000). Moreover, computer technology allows organizations to record individual worker activities at shorter time intervals at a dramatically lower cost (Uzumeri and Nemhard, 1998). Therefore, many researchers can examine the mathematical form for individual learning. Examples of learning models are: log-linear, exponential function, and hyperbolic functions.

Nemhard and Uzumeri (2000B) have categorized learning modeling into two broad areas: organizational learning and individual learning. Organization learning research has focused on the overall implications of learning across large organizational units (Nemhard and Uzumeri, 2000B). The organization learning curves use data that are aggregated across many individuals and many processes (Nemhard and Uzumeri, 2000B). Individual learning research has examined the microstructure of the learning curve in order to understand the mechanisms by which learning occurs at the individual level (Nemhard and Uzumeri, 2000A). The purpose of individual learning is to provide management the information that will give the assignment more efficient allocation with specific characteristics. This study focuses on the individual learning of each worker.

Task complexity is a factor that affects the learning function. Uzumeri and Nemhard (1998) have found that workers have variations in a task. Workers learn more quickly in a location task (fixed location) than in a more difficult search task (randomized location) and it was also found that there were greater performance improvements in tasks performed in the order of “difficult to easy” than those performed in the order of “easy to difficult.” Nemhard (2000) has studied the effects of task complexity on learning and forgetting. The results indicate that task complexity significantly affects learning and forgetting rates.

In a sewing process, three attributes of skill-complexity, which are method, machine and material, are involved. The method consists of the sewing stitch, the length of the stitch, and the section of the unit that is assembled. The machine consists of the type of equipment employed and the amount of automation relevant for the task. The material consists of the specific grade, density, fiber type and accounts for whether the fabric is knitted or woven.

Nemhard (2000) has also examined the effects of task complexity and experience on parameters of learning and forgetting in the garment industry. The study found that the effect of task complexity on learning and forgetting parameters depends on the experience of workers. Workers who have experience with the task method, machine, or material will learn more rapidly and forget more rapidly. Nemhard (2001) proposed a heuristic worker-task assignment policy by assigning the more rapid learners to the shorter production run tasks, and the more gradual learners to the longer production run tasks. The results indicate that the heuristic method significantly improves overall productivity under empirically observed conditions and under many experimental conditions.

Leopairote (2003) has investigated workforce flexibility in a labor constrained flow line system assuming that workers are heterogeneous with respect to learning-forgetting effects. The distribution of individual learning and forgetting behaviors was obtained from an automotive company. The study focused on an unpaced and asynchronous flow line system and addressed selecting appropriate workers for the production lines. The number of stations performed by each worker and level of task sharing are determined. Workers are then assigned to stations and the worker schedule and rotation pattern over a production period is determined in order to maximize throughput of flow line.

Sayin and Karabati (2007) have proposed assigning cross-trained workers to departments. This model has two objectives which are maximum utility and skill improvement. The department utility is a function of departmental labor shortage. Two stages of optimization were studied. The first stage of the model needs to maximize total departmental utility subject to typical assignment constraints. The second stage of the model seeks to maximize total skill improvement by using the outcome of total utility value of the first stage as a parameter input in a constraint of the second stage model. A worker must be allocated to a department. One department allows many workers to work. The skill level of each worker is modeled by a hyperbolic learning curve. The research proposes that once a worker is assigned to a department, his skill levels should improve according to his individual learning curve and the improved skill levels are used for the assignment in the next period.

The problem in this dissertation focuses on assigning the tasks to workstations when the tasks have a precedence requirement, which is the assembly line balancing problem. Generally, the conventional assembly line balancing problem is designed for large batch problems, so the given processing time of the problem is the same for all units; whereas in an assembly line balancing problem for small batch problems, learning cannot be ignored (Karni and Herer, 1995). Consequently, the objective of the conventional problem, which is minimizing the maximum workstation time, cannot be applied when learning is relevant since the bottleneck time dynamically changes based on the reduction of learning slope of an assignment. For this reason, minimizing makespan or completion time is incorporated in this problem.

An assembly line balancing problem with learning consideration is studied under different assumptions. For example, the learning for all tasks is same (Toksari, et al. 2008), learning depends on the task which is assigned (Chakravarty 1988; Karni and Herer 1995; Dar-El 1998; Cohen, Vitner et al. 2006), and learning depends on the worker who operates the task (Cohen, Y., 2008). Furthermore, the processing time which represents learning behavior is set in different ways e.g. discrete (Karni and Herer, 1995 ; Chakravarty, 1988) and continuous, (Cohen and Dar-El 1998 ; Cohen, Vitner et al. 2006 ; Cohen, Y., 2008) which is represented by a learning model. Moreover some studies assumed that tasks can be divisible (Cohen, Vitner et al. 2006 and Cohen, Y., 2008), whereas others (Karni and Herer, 1995 ; Chakravarty, 1988 ; Cohen and Dar-El 1998) assumed the tasks cannot be split.

Regarding the production system, most previous studies focused on the assembly line which is a non-buffered system in which all units are transferred between workstations simultaneously based on the bottleneck station. The upper envelope concept was developed and applied in the non-buffered system (Cohen, Vitner et al. 2006). The upper envelope is formed by the largest workstation time value. It represents the production rate of each stage. Thus the makespan is the sum of the production times under the envelope. For this reason, minimizing area under the envelope is minimizing makespan. Cohen, Vitner, et al. (2006) and Cohen, Y. (2008) developed a method to determine the optimal assignment based on the upper envelope concept. Cohen and Dar-El (1998) proposed an outline of a heuristic procedure to solve the problem. The idea is that the flat learning slope will give the small slope in makespan value, so the heuristic will start with limiting the small slope, then determine the task-worker assignment which is valid for the allowance. Using the total processing time of each task, an assembly line balancing technique is determined. The allowance is re-adjusted in increments and solving the problem is repeated until a feasible solution is found. In this dissertation, we focus on the buffered production line. However, in previous studies, there is rarely research which focuses on the problem where a buffer is allowed in the system.

2.5 Surveys related task-worker assignment in the garment industry

Many researchers have studied problems in the garment industry. Examples of research study are simulation modeling (Khan, 1999), assembly line balancing (Masaru, et al., 1981; Betts and Mahmoud, 1992; Chan, 1997; Hui and Ng; 1999), scheduling (Wong and Chan; 2001; Tomastik, 1996; Chen, et al., 1992), and allocation (Spragg, et al., 1999 and Hui et al., 2002).

Hui and Ng (1999) have studied the effect of time variation for assembly line balancing. They have reported that the time variance should be taken into consideration for improving the effectiveness of line balancing. In real situations, there can be a wide variation in the average skill of workers. A lot of factors cause variations in the operational time of the task such as the fabrics and sub materials, performance of the machinery, working environment and quality level of the product.

Betts and Mahmoud (1992) have studied assembly line balancing in the cloth industry for varying skill of workers. They have stated that the problem of varying skill of workers has multiple optimum solutions that allow the line balancer increased flexibility in the choice of a particular solution. Since the assembly line involves different operations being performed at different production rates, balance control is necessary to make sure that the right person is assigned the right task. Chan, et al. (1997) have presented a Genetic Algorithm that can be used for solving the assembly line balancing problem in an effective manner to meet the realistic production conditions in which workers have arbitrary skill levels. However, they assumed the skill levels at a constant production rate. Moreover, they have allowed that a worker performs only one task and that a task can be assigned to only one worker.

In practice, the performance of the assignment depends on the skill and experience of supervisors who are important for it to be successful. Spargg, et al. (1999) have proposed a model accounting for a supervisor to monitor, analyze and repair production schedules; whereas, Hui, et al. (2002) have captured the knowledge of experienced supervisors and proposed a rule based system for determining the right number of operators to be moved in and out of a sewing section.



ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

CHAPTER III

TASK-WORKER ASSIGNMENT ASSUMING CONSTANT SKILL LEVEL

This chapter presents a problem of an integrate approach to assembly line balancing problem and worker assignment problem assuming constant skill levels of workers. The remainder of this chapter is organized as follows. A problem description and mathematical model are presented in Section 3.1- 3.2. A heuristic to solve the problem is presented in Section 3.3. A performance measurement of heuristic and the computational results are presented in Section 3.4. The conclusions of this work and discussion are presented in Section 3.5.

3.1 Problem description

This problem concerns a flow shop where all job routings are identical and involve all workstations that are in series in the line. A workstation consists of a worker who operates the assigned tasks. An order includes i identical items must be processed through those tasks. We consider the process that includes o tasks and k workers where $o \geq k$. Workers with potentially multiple skills and different skill levels must be assigned to perform tasks. Workers with multiple skills are allowed to perform more than one task as long as they are consecutive in the routing. The differing skill levels of workers are reflected in different processing times for each worker on the same task.

The problem is to jointly assign tasks so that the line is balanced and assign workers to those tasks. To simplify the problem, artificial workstations are established to represent grouped tasks. Tasks within the same workstation must be performed consecutively by the same worker. Recall, some workers have multiple skills and can perform multiple tasks. Since all workers must be assigned, the number of workstations must equals to the number of workers but the number of tasks can be greater than these because some workers can perform more than one task. As a result, the decision becomes worker-workstation assignment.

Figure 3.1 refers to the earlier example of 5 tasks to 3 workstations and illustrated alternative assignment between workstations and workers. For example, on the first row, tasks 1, 2, and 3 are assigned to workstation 1; the task 4 is assigned workstation 2; task 5 is assigned to workstation 3. At the bottom of the Figure 3.1, alternatives of worker-workstation assignment are illustrated.

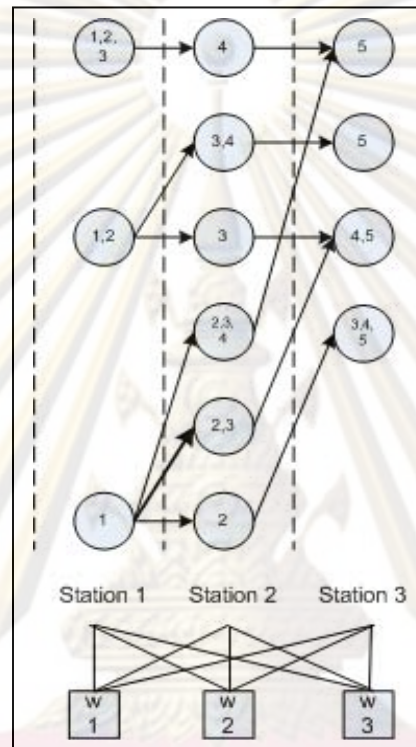


Figure 3.1 The solutions of grouping tasks to workstation and the assignment between workstations and workers.

The total number of feasible assignments is $\binom{o-1}{k-1} \times k!$ which is rather large for reasonably sized, practical problems. It is computed by the number of possible ways to group consecutive tasks to workstations multiplied by the number of possible assignment between workstations and workers. Table 3.1 shows the number of feasible assignments or complexity of the problem. The goal of the research is to develop a task-worker assignment where all tasks must be assigned to workers and each worker is assigned to at least one task. If worker is assigned to more than one task, the length of time they spend on that group of tasks equal to the sum of the

processing time of those tasks. The consecutive tasks are only allowed in case of the multiple assignments. It is further assumed that there is unlimited buffer space between each workstation and the travelling time between workstations is zero.

Table 3.1 The number of feasible assignments or complexity of the problem

No.	No.	complexity(1)	complexity(2)	(1)*(2)
Worker	Task	of grouping	assignment	Complexity of the problem
3	5	6	6	36
5	10	126	120	15,120
5	15	1,001	120	120,120
5	20	3,876	120	465,120
6	18	6,188	720	4,455,360
6	24	33,649	720	24,227,280
7	21	38,760	5,040	195,350,400
7	28	296,010	5,040	1,491,890,400
8	24	245,157	40,320	9,884,730,240
8	32	2,629,575	40,320	106,024,464,000
9	24	490,314	362,880	177,925,144,320
9	32	7,888,725	362,880	2,862,660,528,000
12	36	417,225,900	479,001,600	199,851,873,661,440,000
12	48	17,417,133,617	479,001,600	8,342,834,869,956,790,000
15	45	114,955,808,528	1,307,674,368,000	150,324,764,264,781,000,000,000
15	60	13,298,522,298,180	1,307,674,368,000	17,390,136,741,606,400,000,000,000

3.2 Model formulation

The mathematical model presents below to determine which tasks $\{ j=1,\dots,o \}$ and workers $\{ w =1,\dots,k \}$ have to be assigned to the workstations $\{ s =1,\dots,k \}$. The model has one type of continuous decision variable and three types of binary decision variables. The continuous variable is the objective function value or cycle time (*Cycle*). The binary decision variables relate to the assignments that are to determine whether the task is assigned in a workstation (y_{js}), whether the worker is chosen for a workstation (r_{ws}). And also the three dimensional variables, a_{wjs} combine the assignment solution of both variables y_{js} and r_{ws} . A number of constraints are formulated to ensure feasibility of assignment. The first set of constraints is required to represent the calculation of cycle time or bottleneck time. The maximum processing time of workstation is examined to evaluate the cycle time. The second set of constraints involves worker – workstation assignment. They are required to ensure that all workers must be assigned to operate tasks in a workstation. The last set of constraints involves task – workstation assignment. They are required

to ensure that all tasks are grouped and assigned to workstations. The MIP model for this problem will be defined using the following notations (Table 3.2):

Table 3.2 Notations

Index	
S	= a set of workstations , $s \in S$,for $s = 1, \dots, k$
J	= a set of tasks , $j \in J$,for $j = 1, \dots, o$
W	= a set of workers , $w \in W$,for $w = 1, \dots, k$
Parameter	
k	= the number of workstations or the number of workers
o	= the number of tasks
P_{wj}	= the processing time it takes worker w to complete task j .
Variable	
a_{wjs}	= 1 if task j is assigned to worker w in workstation s 0 otherwise
y_{js}	= 1 if task j is assigned to workstation on s or to any workstation on the precedes s 0 otherwise
r_{ws}	= 1 if worker w is assigned to workstation on s 0 otherwise
<i>Cycle</i>	= Cycle time

The model, then is a follow:

$$\text{Minimize } \textit{Cycle} \tag{3.1}$$

Subject to

Cycle time constraint:

$$\sum_{w \in W} \sum_{j \in J} P_{wj} a_{wjs} \leq Cycle \quad \forall s \quad (3.2)$$

Worker – workstation assignment constraint:

$$\sum_{j \in J} \sum_{s \in S} a_{wjs} \geq 1 \quad \forall w \quad (3.3)$$

$$\sum_{j \in J} a_{wjs} \leq o \times r_{ws} \quad \forall w, \forall s \quad (3.4)$$

$$\sum_{s \in S} r_{ws} = 1 \quad \forall w \quad (3.5)$$

$$\sum_{w \in W} r_{ws} = 1 \quad \forall s \quad (3.6)$$

Grouping task – workstation assignment constraint:

$$y_{ok} = 1 \quad (3.7)$$

$$y_{j+1s} \leq y_{js} \quad 1 \leq j \leq J-1, \forall s \quad (3.8)$$

$$y_{js} \leq y_{j,s+1} \quad \forall j, 1 \leq s \leq S-1 \quad (3.9)$$

$$y_{j1} = \sum_{w \in W} a_{wj1} \quad \forall j \quad (3.10)$$

$$y_{js} - y_{j,s-1} = \sum_{w \in W} a_{wjs} \quad \forall j, 2 \leq s \leq S \quad (3.11)$$

$$a_{wjs} \in (0,1), y_{js} \in (0,1), r_{ws} \in (0,1), Cycle \geq 0$$

The objective function (3.1) minimizes cycle time. Constraints (3.2) determine the cycle time which is the sum of the processing times of the workstation in which tasks are assigned. Constraints (3.3) through (3.6) relate worker assignments to workstations. Constraints (3.3) force all workers to be assigned to at least one task. Constraints (3.4) ensure that if a worker is assigned to the tasks at a workstation, he or she must be assigned to that workstation while constraints (3.5) and (3.6) deal with the required one to one assignment between workers and workstations. Constraints (3.7) through (3.11) are focused on grouping tasks and assigning them to workstations. Constraints (3.7) through (3.9) force a structure on the task-workstation assignments: (3.7) assigns the last task o to the last workstation k , (3.8) forces the required precedence relationships among the tasks in the y variables. That is, if

$y_{j+1s} = 1$ then $y_{js} = 1$. (e.g., If $y_{53} = 1$, then $y_{43} = 1$ and, in turn, if $y_{43} = 1$, then $y_{33} = 1$.) Similarly, (3.9) forces the correct structure for the precedence relationships on the workstations: if $y_{js} = 1$ then $y_{j,s+1} = 1$ (e.g., if $y_{11} = 1$, then $y_{12} = 1$ and since $y_{12} = 1$ then $y_{13} = 1$). Constraints (3.10) and (3.11) ensure that a task cannot be split among workers; thus, each task will be assigned to only one worker.

To illustrate how these variables are interpreted, consider the 5 task, 3 workstations example discussed previously. The solution on the first row of Figure 3.1 would be represented in the model as $\bar{y}_1^T = [1 \ 1 \ 1 \ 0 \ 0]$, $\bar{y}_2^T = [1 \ 1 \ 1 \ 1 \ 0]$ and $\bar{y}_3^T = [1 \ 1 \ 1 \ 1 \ 1]$. To interpret these vectors, $y_{11} = y_{21} = y_{31} = 1$ in \bar{y}_1^T means that tasks 1, 2 and 3 are grouped and assigned to workstation 1. For the other 2 workstations ($s= 2$ and 3), the assignments are determined by $y_{js} - y_{j,s-1}$ for each j . So, for example, $\bar{y}_2^T - \bar{y}_1^T = [0 \ 0 \ 0 \ 1 \ 0]$ means that task 4 is assigned to workstation 2. This is how the constraint set restricts consecutive tasks to the same workstation. It is proposed by Glass and Herer, 2006.

3.3 Heuristic description and numerical example

The proposed heuristic for finding a solution to this problem involves two steps: first, tasks are grouped and, then, the problem is transformed into an assignment problem between the groups of tasks and workers. As noted earlier, there are a very large number of possible task groupings and this is critical to the quality of the solution, so developing a methodology to generate groups of task efficiently is one focus of this research. The proposed methodology starts by determining the upper bound (UB) and lower bound (LB) of the cycle time. A trial value between UB and LB is set to be an upper limit of group size. Then, consecutive tasks are grouped based upon the maximal station load rule. Once tasks are grouped, the problem becomes an assignment problem between workers and groups of tasks. Figure 3.2 illustrates an overview of the heuristic.

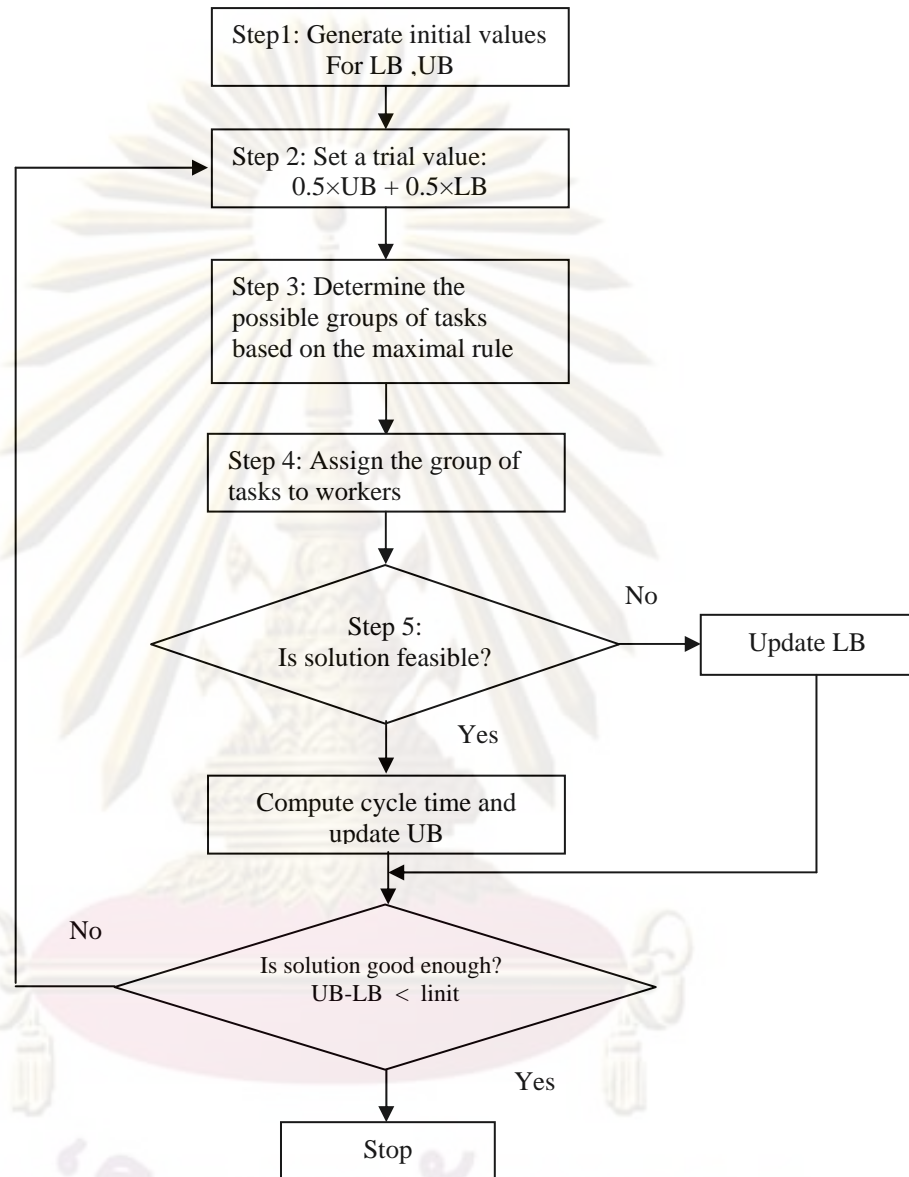


Figure 3.2 The overview of the heuristic

The grouping methodology is now presented as a step by step procedure and then computational experiments are provided.

Step 1: Generate the initial UB and LB

To determine the LB of the cycle time, the processing time of the fastest worker who performs each task is used to represent the processing time of that task. The tasks are assigned to the workstation as in SALBP-II which can be solved by an exact algorithm to minimize cycle time. Groups of tasks are obtained once the LB is found. All precedence constraints are considered to determine the LB. It should be noted that since the LB is obtained using the processing time of the fastest workers, it may not be feasible when all workers are assigned.

To illustrate, again consider our 5 task, 3 worker example. The minimum processing time of each task is {2, 1, 1, 2, 2} which is the time of the fastest worker for each of the 5 tasks. To minimize cycle time using the exact algorithm, the tasks are grouped for the 3 workers as {1 and 2, 3 and 4,5} with corresponding workstation time of {3, 3, 2}. The initial lower bound is $\max\{3,3,2\} = 3$. However the assigned worker solution is {B, C, C}, which is infeasible.

The initial UB is defined as a feasible solution. This is achieved by a one-to-one assignment between the groups of tasks from LB and workers. This assignment is not an original assignment problem since the objective is to minimize the maximum workstation time or cycle time; therefore, a mathematical model of the problem is formulated. This solution will be the initial feasible solution and the cycle time will be the initial UB. In the example, the initial UB is $\max\{3, 3, 5\} = 5$ with solution task grouping {1,2, 3,4, 5} executed by worker assignment {B, C, A}.

Step 2: Set the “trial value” of the cycle time

Once the LB and UB are determined, a trial value is set. A trial value is chosen to be the bound on the cycle time for generating groups of tasks. This can be done in any number of ways as long as it is between the LB and the UB so it can be thought of as the linear combination of the two, $Bt = \alpha \times LB + (1 - \alpha) \times UB$, $0 \leq \alpha \leq 1$. For example, other research that uses this general approach frequently begins with the midpoint, $\alpha = 0.5$. This parameter will certainly have an effect on how efficiently the algorithm finds a high quality solution as well as the computation time of the algorithm so it is adjusted for specific situations using computational experimentation to achieve the desirable balance for the decision maker.

Step 3: Determine the groups of tasks

A fundamental property associated with optimal solutions of this class of problems is that the consecutive tasks will be grouped with the maximum group size and without exceeding the current trial value. This idea is the foundation behind this step in the heuristic because groups of tasks will only be generated based on the maximal station load rule. With this method, the number of the groups of tasks will be reduced and a feasible solution will be generated.

This idea is implemented by using the maximal station load rule to generate groups of tasks for that workstation. Tasks are added sequentially until the trial value is exceeded; that is, only the groups that have a cycle time (e.g, the sum of task processing time) within the trial value is considered. Referring back to our 5 task, 3 worker example, if $LB = 3$, $UB = 5$ and $\alpha = .5$, the first trial value is 4. The possible groups of tasks based on the rule of worker *A* are {task2}, {task3} and {task4}. Worker *A* cannot operate task number 1 since his or her processing time of this task is 5 which is greater than 4. Using this logic, the possible groups of tasks for worker *B* are {task1, task2}, {task2}, {task3}, {task4, task5} and {task5}. Worker *B* cannot perform task 1 alone because the his or her processing time is below the LB so tasks 1 and 2 are grouped (processing time 3 seconds) which is above the LB and below the trial value. Therefore the first task groups for worker *B* is {task1, task2}. The possible groups of tasks for worker *C* are {task2, task3}, {task3, task4}, {task4, task5} and {task5}.

Step 4: Determine the assignment between the possible groups of tasks and workers

The tasks, the worker who performs the tasks and sizes of the possible groups of tasks are transformed in parameters so the assignment between the possible groups of tasks ($g = 1, 2, \dots, G$) and workers is obtained from the following mathematical model with the objective of minimizing cycle time.

Inputs

$$b_{jg} = \begin{cases} 1 & \text{if task } j \text{ is assigned in group } g \\ 0 & \text{otherwise} \end{cases}$$

$$r_{wg} = \begin{cases} 1 & \text{if worker } w \text{ operates the tasks in group } g \\ 0 & \text{otherwise} \end{cases}$$

$e_g =$ group size of group g

Variables

$$x_g = \begin{cases} 1 & \text{if group } g \text{ is selected} \\ 0 & \text{otherwise} \end{cases}$$

Minimize *Cycle*

Subject to

$$\sum_{g \in G} b_{jg} x_g = 1 \quad \forall j \quad (12)$$

$$\sum_{g \in G} r_{wg} x_g = 1 \quad \forall w \quad (13)$$

$$e_g x_g \leq \text{Cycle} \quad \forall g \quad (14)$$

$$x_g \in (0,1), \text{Cycle} \geq 0$$

For example,

$$b_{jg} =$$

Group	1	2	3	4	5	6	7	8	9	10	11	12
j=1	1	0	0	0	0	0	0	0	0	0	0	0
j=2	1	1	1	1	0	0	0	0	0	0	0	0
j=3	0	0	0	1	1	1	1	0	0	0	0	0
j=4	0	0	0	0	0	0	1	1	1	1	0	0
j=5	0	0	0	0	0	0	0	0	1	1	1	1

$$r_{wg} =$$

Group	1	2	3	4	5	6	7	8	9	10	11	12
A	0	1	0	0	1	0	0	1	0	0	0	0
B	1	0	1	0	0	1	0	0	1	0	1	0
C	0	0	0	1	0	0	1	0	0	1	0	1

$$e_g =$$

Group	1	2	3	4	5	6	7	8	9	10	11	12
	3	3	1	3	4	4	3	2	4	4	2	2

$$x_g = \begin{array}{c|cccccccccccc} \text{pattern} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\ \hline & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{array}$$

$$\text{cycle} = 4$$

Step 5: Check the feasibility of the solution

In this step, the UB or LB is altered based on the previous results to reduce the search space for the next iteration. If a feasible solution is found using the trial value, the UB is set to the maximum workstation time in this feasible solution. If not, the LB is set to the trial value. After the UB or LB is updated, the heuristic will check the gap between UB and LB. The search stops when a gap, $UB - LB$, is less than a predefined amount.

Back to our example, a feasible solution can be obtained from the task groups identified in Step 3. The assignment is {task1, task2} for worker B, {task3} for worker A and {task4, task5} for worker C. The cycle time of the assignment is $\max\{3,4,4\}=4$. Since the solution is feasible, UB is set to 4. For $LB=3$ and $UB=4$, the next trial value is 3.5. After groups of tasks are generated and the assignment problem is solved, no feasible solution is found. Hence, LB is set to 3.5. If the stopping criteria is for the gap size to be 1 or less, the heuristic stops because $UB - LB < 1$ and the minimal cycle time from the heuristics is 4.

3.4 Performance measurement of heuristic

3.4.1 Testing problem

The computational experiments are designed to exercise the model and heuristic in a way that illustrates some of the features of each as well as gain some insights that numerical examples can provide. Normally in a modular production in the garment industry, the number of workers is between 6 and 15. The problem parameters were chosen to reflect a realistic situation in the garment industry. In this testing, the number of tasks was three times to four times the number of workers, since generally in the garment industry a worker is assigned less than three to four tasks. The experiments set the number of workers at 8, 12 and 15 and the number of tasks at 3 times and 4 times the number of workers. Hence, there are six problem sizes: 8 workers with 24 tasks (8w24t), 8 workers with 32 tasks (8w32t), 12 workers

with 36 tasks (12w36t), 12 workers with 48 tasks (12w48t), 15 workers with 45 tasks (15w45t) and 15 workers with 60 tasks (15w60t). For each problem size, an experiment was run with the standard processing times and skill of the workers set at two levels. The standard processing times of each task are generated randomly on the intervals [1, 10] and [1, 30] according to a uniform distribution. The skill level of workers is generated on the intervals [-20%, 20%] and [-50%, 50%] also according to a uniform distribution. The skill level of worker is the percentage deviation of the task processing time from the standard processing time. Each standard processing time intervals are paired with both skill level intervals for each combination of workers and tasks, so there are 4 tests for each problem size. For example, if the standard processing time of task1 is 10 and the skill level of worker A is 20%, so the task1's processing time of worker A becomes 12. 5 replications are used in each test meaning that the processing times and skill levels were randomly generated 5 times for each number of tasks and workers. All algorithms have been implemented using C++ a PC with an Intel Core™2 Duo 2.00 GHz CPU and 1.93 GB of RAM. All mathematical programs were solved using the AMPL CPLEX 8.0. There are 4 tests for each problem size, 20 replicates for a problem size, 8 worker and 5 replicates for 12 workers and 15 workers as shown in Table 3.3.

Table 3.3 Problem Size

Problem Code	Number of Worker	Number of Task	Instances	CPLEX Sol.
8w23t	8	23	20	Optimal Sol.
12w36t	12	36	20	Limit time*
15w45t	15	45	20	Limit time*
8w32t	8	32	20	Optimal Sol.
12w48t	12	48	20	Limit time*
15w60t	15	60	20	Limit Time*

Note * Limit time = 54,000 sec. or (15 hours)

3.4.2 Computational experiments

The computational experiments are designed to exercise the model and heuristic in a way that illustrates some of the features of each as well as gain some insights that numerical examples can provide. Normally in a modular production system in the garment industry, the number of workers is between 6 and 15. The problem parameters were chosen to reflect a realistic situation in the garment industry. In this testing, the number of tasks was three times to four times the number of workers, since generally in the garment industry a worker is assigned less than three to four tasks. The experiments set the number of workers at 8, 12 and 15 and the number of tasks at 3 times and 4 times the number of workers. Hence, there are six problem sizes: 8 workers with 24 tasks (8w24t), 8 workers with 32 tasks (8w32t), 12 workers with 36 tasks (12w36t), 12 workers with 48 tasks (12w48t), 15 workers with 45 tasks (15w45t) and 15 workers with 60 tasks (15w60t). For each problem size, an experiment was run with the standard processing times and skill of the workers set at two levels. The standard processing times of each task are generated randomly on the intervals [1, 10] and [1, 30] according to a uniform distribution. The skill level of workers is generated on the intervals [-20%, 20%] and [-50%, 50%] also according to a uniform distribution. The skill level of worker is the percentage deviation of the task processing time from the standard processing time. Each standard processing time intervals are paired with both skill level intervals for each combination of workers and tasks, so there are 4 tests for each problem size. For example, if the standard processing time of task1 is 10 and the skill level of worker A is 20%, so the task1's processing time of worker A becomes 12. 5 replications are used in each test meaning that the processing times and skill levels were randomly generated 5 times for each number of tasks and workers. All algorithms have been implemented using C++ a PC with an Intel Core™2 Duo 2.00 GHz CPU and 1.93 GB of RAM. All mathematical programs were solved using the AMPL CPLEX 8.0.

3.4.2.1 The performance of UB and LB of the heuristic

The initial UB and the initial LB are obtained by the heuristic given in Section 3.3. As mentioned, we expect the range between the initial UB and initial LB is in short-range. The experiment is designed to investigate the effect of the

factor of the variation of skills of worker on the quality of the initial UB and the initial LB. The average on the difference between initial UB and initial LB is presented in Table 3.4.

Table 3.4 The difference between the initial upper bound and initial lower bound.

Processing Time	Skill Level	Problem					
		8w24t	8w36t	12w36t	12w48t	15w45t	15w60t
U[1,10]	U[-20,20]	1.7947	2.2831	1.3490	2.4396	1.4325	2.2299
U[1,10]	U[-50,50]	7.7667	13.4670	9.5528	14.2486	10.1590	13.5106

From the data obtained in Table 3.4, it is found that when the skills of worker is vary in high level, the interval [-50,50], the difference between the initial UB and initial LB is higher than its when the skills of worker is vary in low level, the interval [-20,20]. The search space of the trial bound will be increased when skills of worker vary in high level.

3.4.2.2 The parameters applied in the heuristic

There are two parameters associated with the heuristic: the gap between UB and LB that will serve as the stopping criterion and the weights to be placed on the upper and lower bounds for determining the trial value. It is anticipated that the value of the gap that terminates the heuristic affects the trade-off between the quality of the solution and the computational time. When the gap size is small the quality of the solution will be better but the computation time will be longer because of the increased number of iterations. With wider gaps the reverse is seen. As mentioned previously, experimentation is one way to set these parameters at levels that are acceptable to the decision maker. As such, an experimental study was conducted with the gap between UB and LB at four levels, (0.001, 0.003, 0.03, and 0.3). The weights associated with the UB and LB correspond to $\alpha = .8, .5, .2$ and $.1$ meaning the ratio of the UB:LB weights are (20:80, 50:50, 80:20, 90:10). For example, 90:10 ratio places the trial value 10% of the interval below the UB. This is strictly trial and error because the sole purpose here is to determine suitable parameters to use in the heuristic for the experimental comparisons. The experiments

were performed on problem 8w24t and the measures that were recorded were computational time and quality of the solution. Quality is measured by the normalized difference between the heuristic solution and the CPLEX obtained optimal solution using:

$$\% \text{ Difference} = \frac{\text{Heuristics solution} - \text{Optimal solution}}{\text{Optimal solution}} \times 100$$

Figure 3.3 indicates that more weight on the UB leads to shorter computational times for all stopping criteria conditions although the difference is not very dramatic because the computational time is rather short for a 30% gap. Regarding solution quality, Figure 3.4 shows that all gaps except 30% found the optimal solution to this problem. As such, we selected a 90:10 weight ratio and a gap of 0.03 in an effort to achieve a good solution rather quickly.

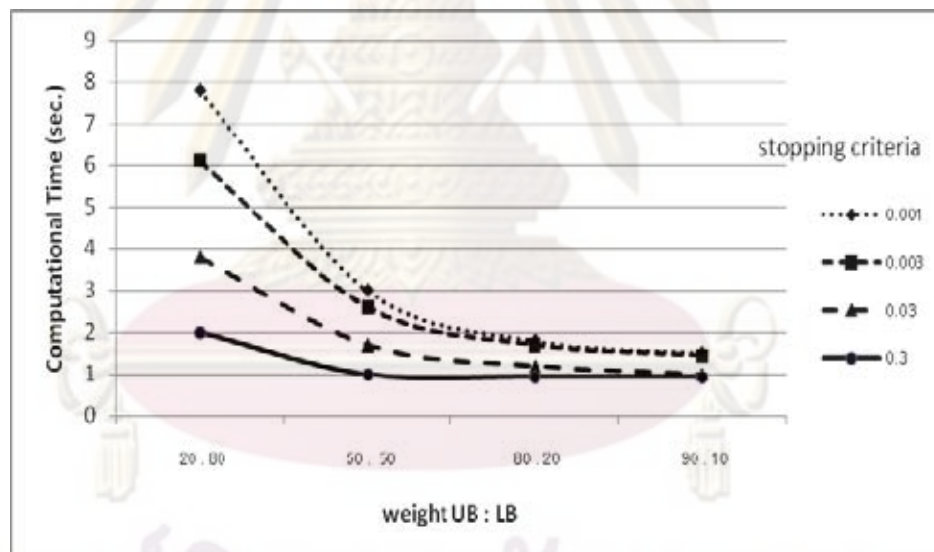


Figure 3.3 The computational time when weight between UB LB are varied(sec.)

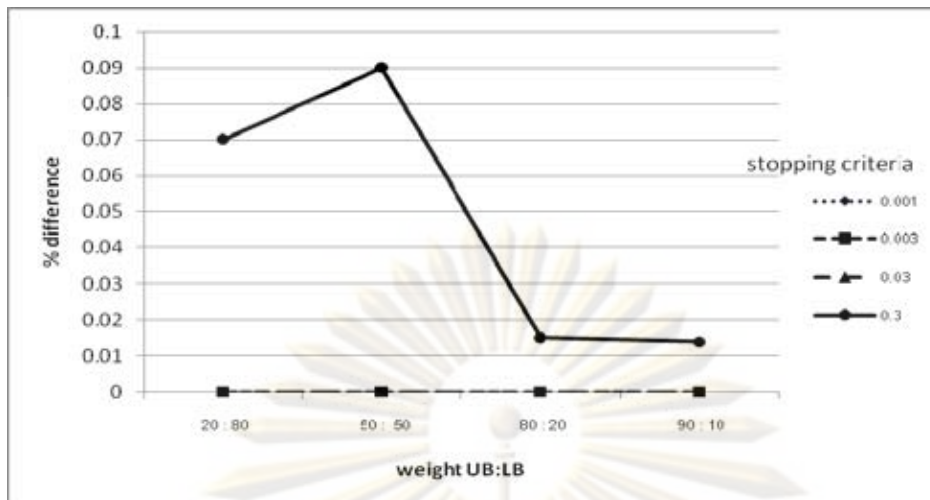


Figure 3.4 The percentage difference of cycle time between heuristic and exact solution (%)

As expected, the ability to find a high quality feasible solution depends on the position of trial value. In these experiments, it is noted that the optimal solution was found routinely with and rather quickly with most of the weight on the UB so we hypothesize that the UB in the heuristic is very good. It would take extensive experimentation to generalize this over wide range of situation; however, in these experiments this is true.

3.4.2.3 The computational time and the quality of the solution of the heuristic

The computational time of the heuristic is compared to the computational time of optimal solution which is found by solved the mathematical model presented in Section 4 using CPLEX. The run time of the problem is limited to 15 hours (54,000 sec.) and the results are presented in Table 3.5. Blank entries indicate that no optimal solution is found within 15 hours. For the problem with 8 workers, the optimal solution can be found for all tests whereas for the problem with 12 workers, the optimal solution is only found by CPLEX in certain cases so the average run time is calculated from the cars where the optimal solutions were found within 15 hours. For 15 workers, CPLEX could not find the optimal solution for any cases. As expected, the computational time of the heuristic is significantly lower than

the time to find the exact solution and even for larger problems the computing time was less than 10 seconds.

Table 3.5 The computational time of the heuristic and exact solution (sec.)

Test	8w24t	8w36t	12w36t	12w48t	15w45t	15w60t
Run time : Optimal (sec.)	2115	7549	25,026	41,582	-	-
Run time : Heuristic(sec.)	1.156	1.520	2.189	3.983	4.375	9.105

Figure 3.5 compares the computational time of the heuristic versus different skill levels of the workers. It can be seen that computational time is dramatically higher when the variability of the skill levels is high., that is, the skill levels are generated using the broader uniform distribution between [1, 50]. The high variation of skill level results the big difference between UB and LB since the fastest worker in the high variation of skill level reflects the lower LB. The number of search increases when the difference between UB and LB is high. With the reason the computational time increases.

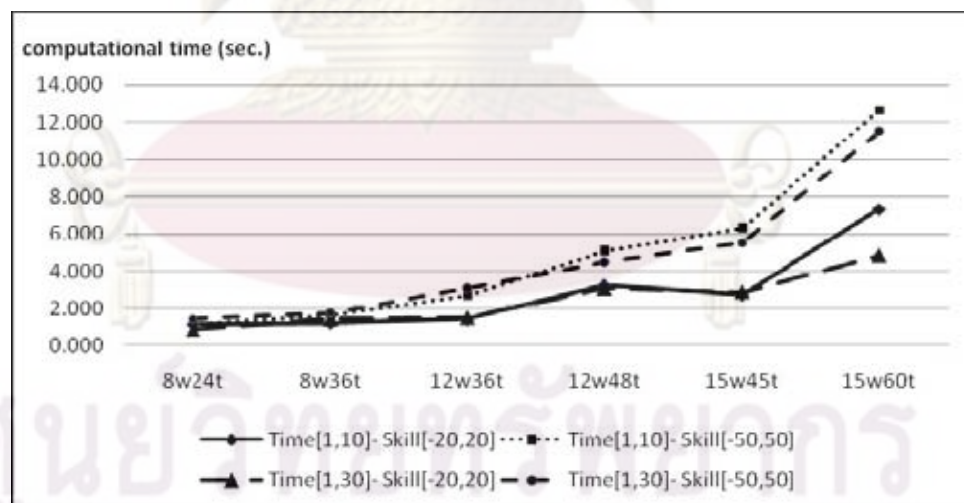


Figure 3.5 Computational time of the heuristic when the skills of worker are varied

For the quality of solution, % *difference* is used as described in earlier. For problems 8w24t and 8w32t, the heuristic found the optimal solution in almost all cases. Since the optimal solutions was rarely found by CPELX within 15

hours for problems with 12 and 15 workers, the comparison is made between the best know solution from CPLEX and the heuristic. As can be seen in Figure 3.6, the heuristic solution is better than the best solution from CPLEX for all the cases. Furthermore, the difference between the best found solution and the optimal solution increases when the problem size increases and especially when skills of workers vary in high level as shown in Figure 3.6. The heuristic works well compared to CPLEX when the problem size increase.

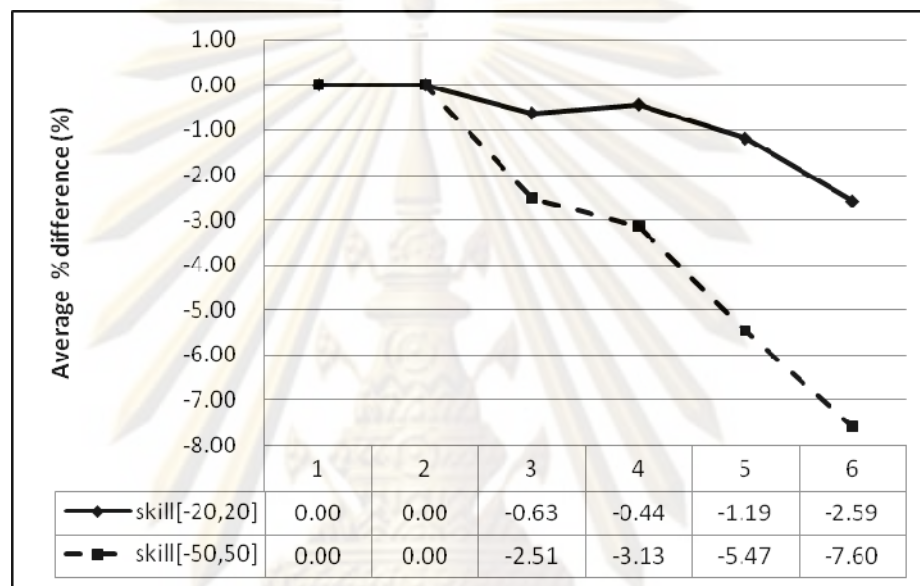


Figure 3.6 The percentage of the difference between the optimal or the best found solution and the heuristic solutions(%)

3.4.2.4 Comparison with the heuristic in practical application

A final set of experiments is conducted to compare the solution quality found using the proposed heuristic solution to the solution quality found by applying the 2-stage heuristic commonly found in the Thai garment industry that has been outlined earlier in this paper. For comparison, a slight modification to the previous percentage difference is made:

$$\% \text{Difference} = \frac{\text{2-stage solution} - \text{Heuristic solution}}{\text{Heuristic solution}} \times 100$$

The skill levels of the workers are distributed uniformly in four intervals for these tests: [-10,10], [-20,20], [-30,30], [-50,50]. Figure 3.7 displays the results and they are very interesting. The 2-stage heuristic is less than 5% worse than the proposed

heuristic when the deviation of skills of worker is small (e.g., generated by the $[-20,20]$ interval). The performance degrades to 10-15% when the deviation of skills of worker increases as reflected in a generation interval of $[-30, 30]$ and the degradation is dramatic in the highest level of variance, increasing to 25-40%. We submit that this degree of error in labor intensive industries where competitive position can be lost with small decreases in efficiency, it is important for decision makers to consider using an integrated approach such as this proposed algorithm.

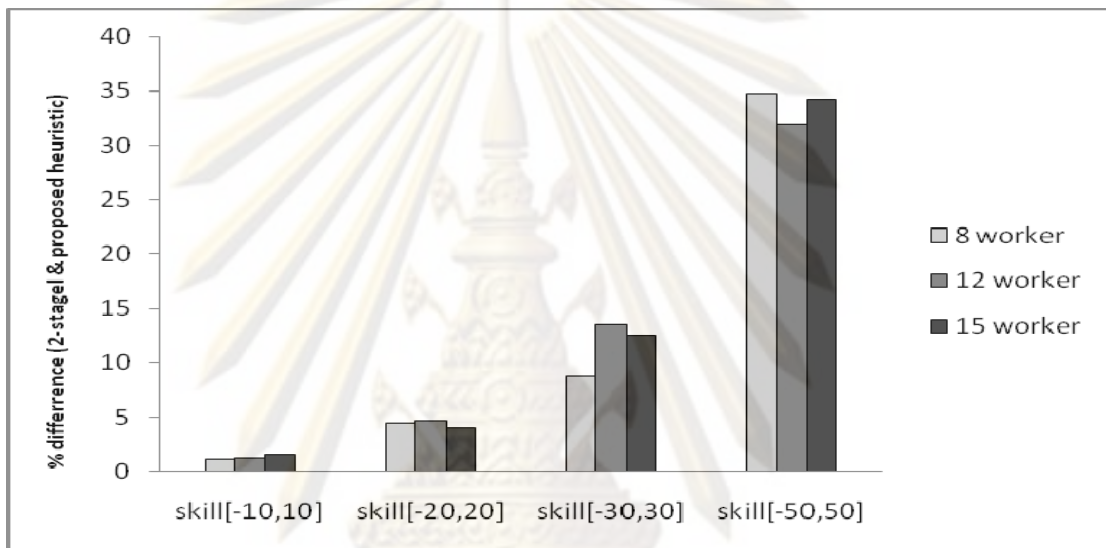


Figure 3.7 The comparison of the percentage difference between the proposed heuristic and a 2-stage heuristic solutions

3.5 Conclusion

This chapter has proposed a problem of an integrate approach to assembly line balancing problem and worker assignment problem assuming constant skill levels of workers. The problem with the objective to minimize cycle time is modeled. For solving the problem, a heuristic is developed. The lower bound and upper bound are determined. The binary search is modified to determine the sequence of trial cycle time. The heuristic groups the tasks based on the maximal station load rule within a trial value limit. Then, an assignment between the groups of tasks and the workers is performed.

The effectiveness of the heuristic is evaluated in term of computational time and quality of solution compared to the optimal solution. Moreover, the proposed

heuristic is compared to the 2-stage heuristic in term of the quality of solutions. It confirms the disadvantage if the 2-stage heuristic is applied when the skills of worker vary highly.



ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

CHAPTER IV

TASK-WORKER ASSIGNMENT TAKING INTO ACCOUNT LEARNING ABILITY

This chapter presents a problem of an integrated approach to assembly line balancing and worker assignment taking into account learning ability of workers. The remainder of this chapter is organized as follows. A problem description and mathematical model presented in Section 4.1- 4.2. A heuristic to solve the problem is presented in Section 4.3. The detail of the heuristic and a numerical example shows in Section 4.4. A performance measurement of heuristic and the computational results will be presented in Section 4.5. The conclusions of this work and discussion are presented in Section 4.6.

4.1 Problem description

This problem concerns an assembly line which is a set of sequential workstations. Buffer spaces are set up between workstations. A workstation consists of a worker who carries out assigned tasks. An order includes i , which are identical items that must be processed along the same route pass through all workstations. This problem assumes that the workers have multiple skills and are able to do more than one task. The skill levels of workers differ; therefore, workers' processing time vary.

Moreover, workers have different learning abilities, i.e. the processing time of the succeeding item being shorter than the preceding item. In the problem, the processing time of each worker for each item is based on the skill level and learning of the worker. Table 4.1 provides an example of processing time applied in the problem.

Table 4.1 The processing times of 5 tasks, 3 workers (A, B, C) and 3 items (sec.)

Worker A	items 1	items 2	items 3
Task 1	4	4	3
Task 2	3	2	1
Task 3	7	4	4
Task 4	2	1	1
Task 5	4	4	3

Worker B	items 1	items 2	items 3
Task 1	5	4	3
Task 2	3	3	2
Task 3	6	6	4
Task 4	3	1	1
Task 5	5	4	2

Worker C	items 1	items 2	items 3
Task 1	4	2	2
Task 2	4	2	2
Task 3	7	6	5
Task 4	2	2	1
Task 5	6	2	2

We want to establish worker assignments where all tasks must be assigned to workers. Each worker is assigned to at least one task. Since it is assumed that the number of tasks is greater than the number of workers, some workers can perform multiple tasks. Only consecutive tasks are allowed in the multiple assignments to smooth out the line. If a worker is assigned to more than one task, the worker processing time is determined by the sum of the processing time of all tasks that s/he performs. For example, if task 1 and task 2 are assigned to worker A, then the task processing times are 7, 6, 4 for items 1, 2, and 3 respectively. Tasks cannot be split.

This problem is also based on the following assumptions:

- 1) The other learning factors among the tasks at the same workstation are not considered such as the task similarity.
- 2) There is unlimited buffer space between each workstation. The objective is to minimize the makespan, which is the completion time of the last item of the last task. The problem is formulated in an integer linear programming model.

The objective is to minimize the makespan which is the completion time of the last item of the last task. Figure 4.1 illustrates makespan from an assignment solution

i.e. the solution of assignment i.e. worker A – operation 1 and 2, worker B – operation 3 and worker C – operation 4 and 5.

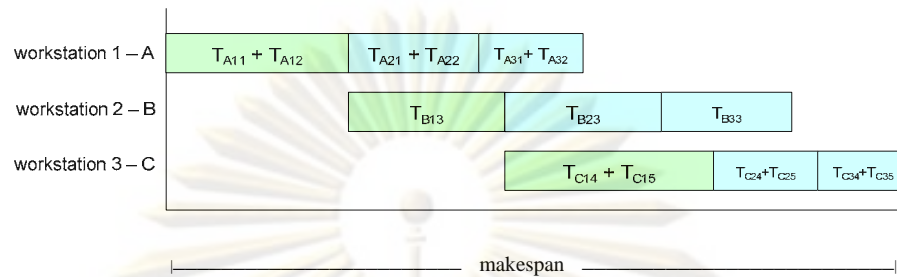


Figure 4.1 Makespan from an assignment solution

4.2 Model formulation

The mathematical model presented below determines which tasks $\{j=1, \dots, o\}$ and workers $\{w=1, \dots, k\}$ are assigned to the workstations $\{s=1, \dots, k\}$. Workers must complete all items $\{m=1, \dots, i\}$ with the minimum completion time. The model uses three types of continuous decision variables and three types of binary decision variables. The continuous variables relate to the objective function value or makespan (C_{\max}), the processing time (q_{ms}), and completion time (π_{ms}) of items m in workstations s . The binary decision variables relate to the assignments that are to determine whether the task is assigned to a workstation (y_{js}), whether the worker is chosen for a workstation (r_{ws}), and also the three dimensional variables, a_{wjs} which combine the assignment solution of both variables y_{js} and r_{ws} . Different types of constraints are formulated to ensure feasibility of assignment. The first set of constraints represents the mechanism of the flow line in which all items follow the same sequence of operations. The processing time of tasks was used to evaluate the completion time of each item. The second set of constraints involves worker – workstation assignment. They are required to ensure that all workers must be assigned to operate tasks within a workstation. The last set of constraints involves task – workstation assignment. They are required to ensure that only consecutive tasks are grouped and assigned to workstations. The MIP model for this problem was developed using the notations in Table 4.2:

Table 4.2 Notations

Index		
M	= set of items	, $m \in M$,for $m = 1, \dots, i$
S	= set of workstations ,	, $s \in S$,for $s = 1, \dots, k$
J	= set of tasks	, $j \in J$,for $j = 1, \dots, o$
W	= set of workers	, $w \in W$,for $w = 1, \dots, k$
Parameter		
i	= the number of items	
k	= the number of workstations / the number of workers	
o	= the number of tasks	
T_{wmj}	= the processing time of item m task j operated by worker w	
Variable		
a_{wjs}	= 1 if task j is assigned to worker w in workstation s	
	0 otherwise	
y_{js}	= 1 if task j is assigned in or before workstation s	
	0 otherwise	
r_{ws}	= 1 if worker w works in workstation s	
	0 otherwise	
q_{ms}	= the processing time of item m in the workstation s	
π_{ms}	= the completion time of item m in workstation s	

$$\text{Minimize } \pi_{ik} \quad (4.1)$$

Subject to

Completion time constraint:

$$q_{ms} = \sum_{w \in W} \sum_{j \in J} T_{wmj} a_{wjs} \quad \forall m, \forall s \quad (4.2)$$

$$\pi_{11} = q_{11} \quad (4.3)$$

$$\pi_{ms} \geq \pi_{m s-1} + q_{ms} \quad 1 \leq m \leq M, \quad 2 \leq s \leq S \quad (4.4)$$

$$\pi_{ms} \geq \pi_{m-1s} + q_{ms} \quad 2 \leq m \leq M, \quad 1 \leq s \leq S \quad (4.5)$$

Worker – workstation assignment constraint:

$$\sum_{j \in J} \sum_{s \in S} a_{wjs} \geq 1 \quad \forall w \quad (4.6)$$

$$\sum_{j \in J} a_{wjs} \leq o \times r_{ws} \quad \forall w, \forall s \quad (4.7)$$

$$\sum_{s \in S} r_{ws} = 1 \quad \forall w \quad (4.8)$$

$$\sum_{w \in W} r_{ws} = 1 \quad \forall s \quad (4.9)$$

Grouping task – workstation assignment constraint:

$$y_{ok} = 1 \quad (4.10)$$

$$y_{j+1s} \leq y_{js} \quad 1 \leq j \leq J-1, \forall s \quad (4.11)$$

$$y_{js} \leq y_{j,s+1} \quad \forall j, 1 \leq s \leq S-1 \quad (4.12)$$

$$y_{j1} = \sum_{w \in W} a_{wj1} \quad \forall j \quad (4.13)$$

$$y_{js} - y_{j,s-1} = \sum_{w \in W} a_{wjs} \quad \forall j, 2 \leq s \leq S \quad (4.14)$$

Objective function (4.1) is to minimize the makespan. Constraints (4.2) are used to calculate the sum processing time of all tasks that are assigned in workstation s for item m . Constraint (4.3) reflects that the production line starts empty, no WIP at the beginning of production so the completion time of first item in the first workstation equals q_{11} . Constraints (4.4) represent that workstation s can process an item only after the previous workstation $s-1$ has finished the operation on the item m . Similarly, an item m can be operated on a workstation only after the previous item $m-1$ has completed the operation on the workstation as shown in constraints (4.5). Constraints (4.6) force all workers to be assigned to at least one task. Constraints (4.7) represent that if a worker is assigned to any tasks in workstation, they must work in that workstation. Constraints (4.8)-(4.9) represent one - one assignment between workers and workstations.

Constraints (4.10) through (4.14) concern grouping tasks and assigning them to workstations. Constraints (4.10) through (4.12) force a structure on the task-workstation assignments. Constraint (4.10) assigns the last task o to the last workstation k . Constraints (4.11) force the required precedence relationships among the tasks in the y variables. That is, if $y_{j+1s} = 1$ then $y_{js} = 1$. (e.g., If $y_{53} = 1$, then $y_{43} = 1$ and, in turn, if $y_{43} = 1$, then $y_{33} = 1$.) Similarly, constraints (4.12) force the correct structure for the precedence relationships on the workstations: if $y_{js} = 1$ then $y_{j,s+1} = 1$ (e.g., If $y_{11} = 1$, then $y_{12} = 1$ and since $y_{12} = 1$ then $y_{13} = 1$).

Constraints (4.13) and (4.14) also ensure that a task cannot be split among workers; thus, each task will be assigned to only one worker.

To illustrate how these variables are interpreted, consider the 5 task, 3 workstations example discussed previously. If tasks 1, 2, and 3 are assigned to workstation 1, task 4 is assigned to workstation 2 and task 5 is assigned to workstation 3, this assignment would be represented in the model as $\vec{y}_1^T = [1 \ 1 \ 1 \ 0 \ 0]$, $\vec{y}_2^T = [1 \ 1 \ 1 \ 1 \ 0]$ and $\vec{y}_3^T = [1 \ 1 \ 1 \ 1 \ 1]$. To interpret these vectors, $y_{11} = y_{21} = y_{31} = 1$ in \vec{y}_1^T means that tasks 1, 2 and 3 are grouped and assigned to workstation 1. For the other 2 workstations ($s= 2$ and 3), the assignments are determined by $y_{js} - y_{j,s-1}$ for each j . So, for example, $\vec{y}_2^T - \vec{y}_1^T = [0 \ 0 \ 0 \ 1 \ 0]$ means that task 4 is assigned to workstation 2. This is how the constraint set restricts consecutive tasks to the same workstation.

4.3 Heuristic description

The idea behind the heuristic is applying the concept of an upper envelope of a non-buffered system in order to determine the upper bound (UB) and lower bound (LB) of the solution to limit search space. The solutions are searched only between UB and LB. The heuristic starts by grouping tasks into workstations then, assigning workers to perform grouped tasks. Groups of consecutive tasks are generated based on the trial value, which is a predetermined value between UB and LB. The objective of the problem is to determine the assignment of the grouped tasks and workers which has the minimum makespan.

Regarding the relation between idle time and makespan, the idle time has the recurrence relation between the idle time of the previous workstation and the idle time of the previous item on the same workstation. Bellman (1982) identified makespan in two simple expressions. For the first expression, makespan is calculated by the idle time on the last station plus the summation of the task's processing time of the last workstation. For the second expression, it is calculated by the flow time of a last item from the completion time on the first workstation to the completion on the last workstation plus the summation of task's processing time of the first station as shown in Figure 4.2.

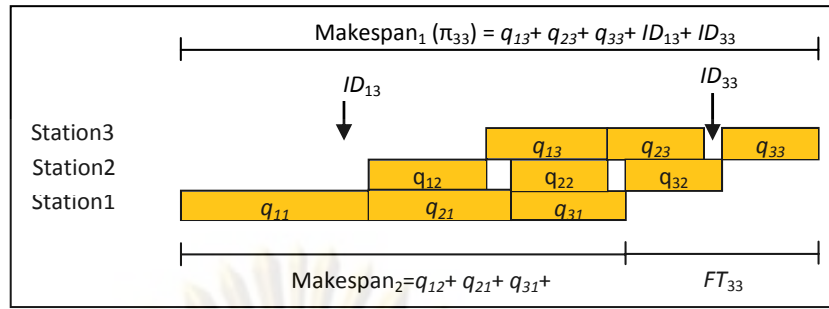


Figure 4.2 An expression of the Makespan

Makespan depends on the idle time on the last station plus the total task processing time of the last workstation, and the idle time on the last station depends on the idle time of previous stations. The recurrence relation of the idle times of the previous workstation is shown in Section 4.3.1. We developed a methodology to minimize the idle time of every workstation via a recurrence relation. The details of the heuristic are presented in Section 4.3.3.

4.3.1 The recurrence relation between the idle times

This section shows the recurrence relation between the idle times of the previous workstation. Let FT_{ms} be the flow time of an item m from the completion time of the first workstation to the completion of workstation s as shown in an equation 4.15, i.e. $FT_{33} = q_{32} + q_{33}$, let ID_{ms} be the idle time of an item m at workstation s , there are two types of expression of the makespan. Using the idle time on the last workstation, the makespan (π_{ik}) = $\sum_{m=1}^i q_{mk} + \sum_{m=1}^i ID_{mk}$, the first term is the sum of the task processing time and the second term is the sum of the idle times on the last workstation. Using the flow time, the makespan (π_{ik}) = $\sum_{m=1}^i q_{m,1} + FT_{ik}$, the first term of the right side is the sum of the task processing time at the first workstation and flow time of the last item at the last workstation.

$$FT_{ms} = \pi_{ms} - \pi_{m1} \quad (4.15)$$

$$\text{Since } \pi_{ms} = \sum_{z=1}^m q_{zs} + \sum_{z=1}^m ID_{zs} \text{ and } \pi_{m1} = \sum_{z=1}^m q_{z1},$$

$$\text{so } FT_{ms} = \sum_{z=1}^m ID_{zs} + \sum_{z=1}^m (q_{zs} - q_{z1}) \quad (4.16)$$

Since the start of the processing of item m at workstation s follows both the completion of item m at the previous workstation $s-1$ and the completion of the previous item $m-1$ on workstation s , the recurrence relation hold for every flow time, so

$$FT_{ms} = q_{ms} + \max[FT_{m-1s} - q_{m1}, FT_{ms-1}] \quad (4.17)$$

By replacing FT_{m-1s} and FT_{ms-1} in an equation 4.17 by the equation 4.16,

$$\begin{aligned} \sum_{z=1}^m ID_{zs} + \sum_{z=1}^m (q_{zs} - q_{z1}) = \\ q_{ms} + \max\left[\sum_{z=1}^{m-1} ID_{zs} + \sum_{z=1}^{m-1} (q_{zs} - q_{z1}) - q_{m1}, \sum_{z=1}^m ID_{zs-1} + \sum_{z=1}^m (q_{zs-1} - q_{z1})\right] \end{aligned} \quad (4.18)$$

By subtracting the quantity, $\sum_{z=1}^m (q_{zk} - q_{z1})$ from both sides of the equation 4.18, the recurrence relation for idle time is obtained.

$$\sum_{z=1}^m ID_{zs} = \max\left[\sum_{z=1}^{m-1} ID_{zs}, \sum_{z=1}^m ID_{zs-1} + \left(\sum_{z=1}^m q_{zs-1} - \sum_{z=1}^{m-1} q_{zs}\right)\right] \quad (4.19)$$

An equation 4.19 shows the recurrence relation between the idle time of the previous workstation and the idle time of the previous item on the same workstation. For example given a partial solution on workstation 1, which is worker1 operating task 1, 2 on station 1, so when a worker is assigned on the workstation2, the idle on the workstation 2 will depend on the idle on the workstation 1 and the idle time of the previous item on the workstation2. The relation is the same for the next workstation until the last workstation which is a part of makespan. From this fact of the recurrence relation, we expect that minimizing idle time of partial solution of assignment from the first workstation to the last workstation can lead to a good solution of full assignment to minimizing makespan.

4.3.2 The Dijkstra's algorithm

Generally, Dijkstra's algorithm is algorithm to determine the shortest path. For example Dijkstra's algorithm is used to determine the shortest path between any two nodes the start node, s and the end node, t . The principle behind Dijkstra's algorithm is that if node x is a node between the start node, s and the end node, t , the shortest path from s to t , then s, \dots, x had better be the shortest path from s to x . The length of the shortest path to each node is improved if there is a better way from s to other node through x . This algorithm is dynamic programming strategy where the distance from s to all nearby nodes is kept, then it is use to find the shortest path to other distant nodes, Skiena (1998). Figure 4.3 shows the step of Dijkstra's algorithm. The Dijkstra's algorithm is modified to determine the feasible assignment between the consecutive grouped tasks and workers which has minimum makespan.

```

Dijkstra's algorithm
1  FOR  each ( $v$ ) in all nodes
2      SET  weight of each node ( $Weight_v$ ) = infinite
3      SET  status of the previous node ( $previous_v$ )= NULL
4  END  Loop
5  SET   $Weight_{start\_node}$  of the start node = 0
6  SET   $Se$  = empty set
7  SET   $Q$  = set of solutions in all nodes
8  WHILE  ( $Q$  is not an empty set loop)
9      Determine ( $u$ ) node which has the minimum weight in set  $Q$ ,
10      $u$  is removed from set  $Q$ 
11      $u$  is added in set  $Se$ 
12     FOR each node ( $b$ ) which is connected from node  $u$  using a single arc
13         IF ( $(distance_b > distance_u + weight_b)$  ) then
14              $distance_b = distance_u + Weight_b$ 
15             SET  $u = the\ previous_v$  of node  $b$ 
16         END IF
17     END Loop
18 END WHILE

```

Figure 4.3 Dijkstra's algorithm

4.4 The detail of the heuristic and numerical example

The heuristic simplifies the problem by grouping tasks first. After tasks are grouped, the feasible assignment between the groups of tasks and workers will be determined by the modified Dijkstra's algorithm. We developed a procedure to generate groups of tasks. First, the UB and LB of the solution are determined. Then the trial value between UB and LB is set as a bound for task grouping. They are detailed in the following steps. Figure 4.4 shows the flowchart of the heuristic.

Step 1: Generate the initial UB and LB

To determine UB and LB, the summation of task processing time is determined in Table 4.3 based on the data in Table 4.1 and it is solved as an assembly line balancing problem to minimize the maximum workstation time. The objective function value in this stage becomes the LB of the original problem. The solution of worker - task assignment is used to evaluate makespan. This makespan becomes the UB of the original problem.

For example, to minimize the maximum workstation time, the solution of the task-workstation assignment is {12, 3, 45}, the worker-workstation assignment is {C, B, A} and the objective value is 16, so the LB is 16. The makespan of this solution and also the UB is 29, as shown in Figure 4.5.

Step 2: Set the trial value (TV)

Once LB and UB are determined, a trial value is set. A trial value is chosen to be the bound for generating groups of tasks. The trial value is 10% of the UB-LB difference increased from the LB or $TV = LB + (UB-LB) \times 0.1$. Therefore, the first trial value of the example is $16 + (29-16) \times 0.1 = 17.3$. The UB or the makespan value of the solution consists of the idle time plus workstation time at the last workstation. The LB is the minimum size of grouped tasks. However, the solution of the minimum groups of tasks may not give the optimal solution in makespan value. For this reason, we set the trial value by increasing the LB with the small amount (10%) of the UB-LB difference in order to search for a better solution in a close area.

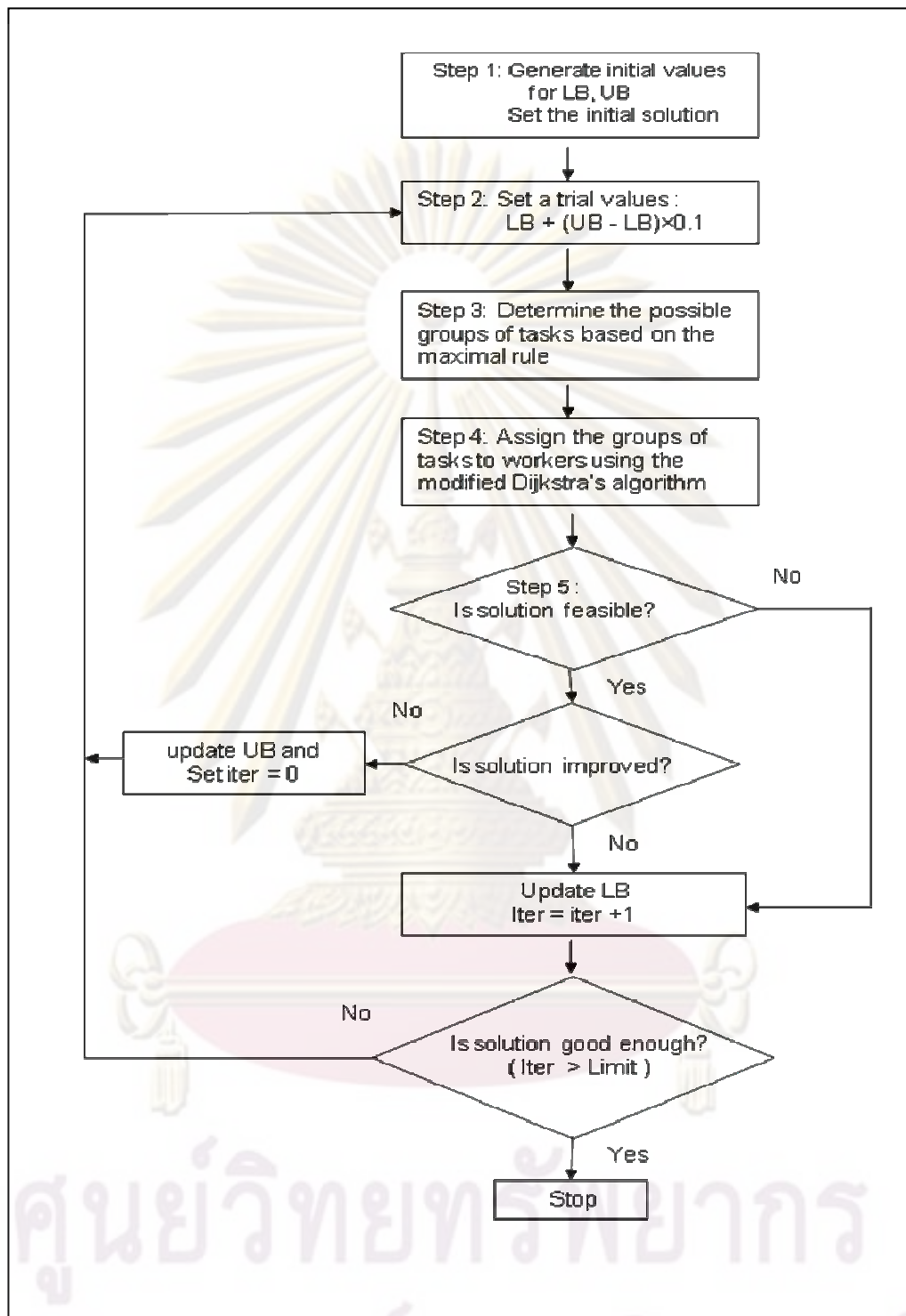
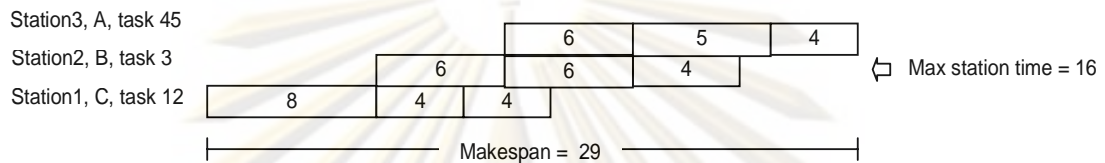


Figure 4.4 the flowchart of heuristic

Table 4.3 the summation of task processing time (sec.)

	Task 1	Task 2	Task 3	Task 4	Task 5
Worker A	11	6	15	4	11
Worker B	12	8	16	5	11
Worker C	8	8	18	5	10

**Figure 4.5 An example of LB and UB**

Step 3: Determine the groups of tasks

The idea is that we will generate only the groups of tasks based on the maximal station load rule which is that a workstation will never close if fittable tasks remain. The consecutive tasks will be grouped without exceeding the current trial value and with the maximum group size. Only the groups of tasks based on the maximal station load rule are generated. Beginning with each task, the next consecutive task will be in the groups. Only the maximum group size (the sum of task processing time) within the trial value is considered. The groups of tasks are generated based on the trial value. For example, the first trial value is 17.3. The possible groups of tasks based on the rule of worker *A* are {tasks1, 2}, {task2} {task3} {tasks4, 5}, and {task 5}. Worker *A* does not operate task 1 alone and 4 alone because the sum processing time of tasks 1 and 2 is 17 seconds and the sum processing time of tasks 4 and 5 is 15 seconds which are below the maximum group size of 17.3 seconds. The possible groups of tasks for worker *B* are {task1}, {task2}, {task3}, {tasks4, 5} and {task5}. Worker *B* does not operate tasks 4 alone because the sum processing time of tasks 4 and 5 is 16 seconds which is below the maximum group size of 17.3 seconds. The possible groups of tasks for worker *C* are {tasks1, 2}, {task2}, {tasks 4, 5} and {task5}. Worker *C* cannot operate task 3 since the processing time of task 3 is 18 seconds which is greater than 17.3.

In Figure 4.7(A), the number in the nodes shows a group of consecutive tasks within the trial value. In this step, a worker who is chosen to carry out each group of tasks is not considered. However, workers who can do those tasks within the trial value are shown at upper right of the nodes. For node 12, workers A and C can be chosen. In the next step, the heuristic will determine the feasible paths. A feasible path must include all tasks using the number of nodes equal to the number of workers. The infeasible nodes are cut off as in Figure 4.7(B). The feasible path is duplicated based on the workers who can operate tasks in those nodes. Therefore, node 12 will be extracted to node 12A and 12C as shown in Figure 4.7(C). After that, the step of the modified Dijkstra's algorithm is run to determine the path which leads to the minimum makespan.

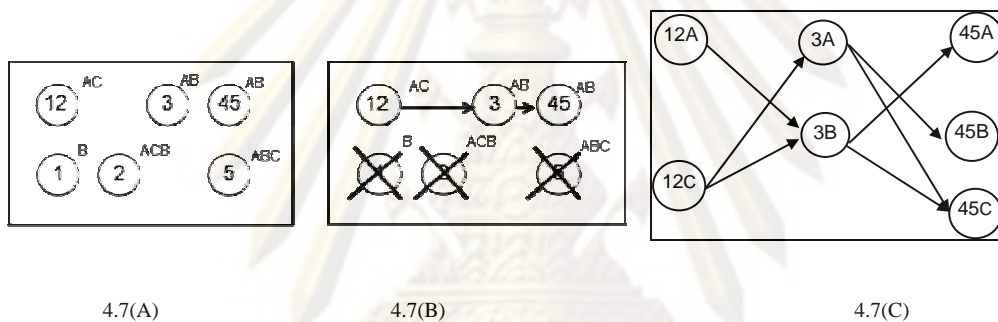


Figure 4.6 (A) Nodes of the group of tasks, 4.7(B) Feasible nodes of task-workstation assignment, 4.7(C) Nodes for Modified Dijkstra's algorithm

Step 4: Determine the assignment of the possible groups of tasks and workers

This step applies the idea of Dijkstra's algorithm. Generally, Dijkstra's algorithm is an algorithm to determine the shortest path. A node in this problem is a workstation that includes an assignment between the consecutive tasks and a worker and a path means the assignment starting from the first workstation to the last workstation. In the Dijkstra's algorithm, the shortest path from the starting node to every other node is determined in order to develop the shortest path to the ending node. In the heuristic, the shortest path to each node becomes the minimum idle time of that node which is designed to search for a solution minimizing makespan. If the

minimum idle time of the last station is found, the minimum makespan would tend to be reached.

The search will start at a randomly selected initial node. The initial nodes are the ones which own the first task e.g. node 12A and 12C. The algorithm will repeat at other initial nodes that are not selected. All initial nodes are possible to be chosen.

Figure 4.7 shows the pseudo code of the modified Dijkstra's algorithm. Firstly, the best makespan is set to UB (line 0). The initial node is randomly selected (line 1) and the variables to keep a solution are initialized (line 2 to line 10). In each step, the heuristic will determine which node in set Q which has the minimum idle time. It will be node u as shown in line 12. In the first loop, node u will be the initial node. If node u is found, the node u will be added to set Se and removed from set Q (line 14 to line 15). The nodes which are connected from node u using a single arc will be node b . The algorithm will check the feasibility of assignment of node b from node u . The one to one assignment between worker and workstation is examined. If the assignment is feasible (line 17), the sub-makespan and the idle time on the workstation b will be determined (line 18).

If the new idle time, new_idle_b , is less than idle time, $idle_b$ of node b and the sub makespan is less than the best makespan, the idle time of the node, $idle_b$ is updated (line 19 - 20). Then the assignment solution is kept (line 21). For example, the initial node is 12A, so node 12A will be node u . Node 3B will be node b . This assignment is feasible. The sub-makespan will be 23 and the idle time will be 7.

Figure 4.9 shows an example of the idle and the sub-makespan. Set $previous_b$, the previous node of node b equals node 12A. The steps will repeat for all node b . Then the new node u , which has minimum idle time, is determined again. The step will repeat until no node having minimum idle is found or idle time of every node in Q equals infinity. If the last tasks is in node b and $s_makespan_b$ is less than the best makespan, then the best makespan is updated (line 22-23). For example, node u is 3B and node 45C is node b . Previously, the best makespan was infinity. It is updated to 28 with the assignment solution (12A, 3B, 45C) as shown in Figure 4.8. Since the heuristic is designed to minimize idle time that is not a direct algorithm to minimize makespan, it is better to keep a set of solutions in a node.

Step 5: Check the improvement of the solution and Stopping Criteria

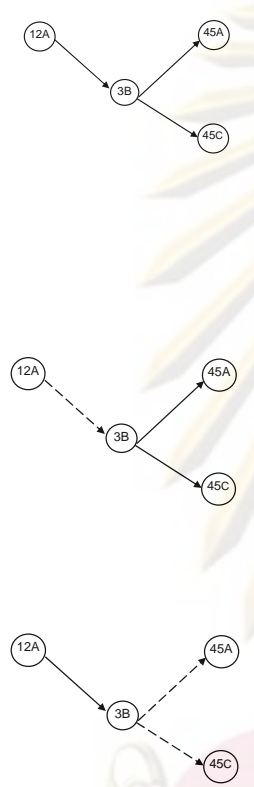
This step is to select whether UB and LB should be updated in order to further limit search space. If a better makespan of a feasible solution is found, the UB is set to the makespan value in the corresponding solution. Otherwise LB is set to the previous trial value. Then new trial value is re-calculated. The stopping criteria depend on the cumulative number of iterations which is not improving the makespan. For example, the limit of iterations is 2; the heuristic will stop if a solution is not improved in 2 consecutive iterations

```

Modified Dijkstra's algorithm
0  Set best_makespan = UB
1  Randomly select of initial node
2  FOR each node  $v$ 
3      SET idle time of each node ( $idle_v$ ) = infinite
4      SET sub makespan of each node ( $s\_makespan_v$ ) = infinite
5      SET status of the previous node ( $previous_v$ ) = NULL
6  END Loop
7  SET  $idle_v$  of the selected starting node = 0
8  SET  $Se$  = empty set
9  SET  $Q$  = set of unselected nodes
10 SET  $Have\_min$  = true
11 WHILE ( $Have\_min$  = true)
12     Determine node ( $u$ ) which has the minimum idle time in  $Q$ ,
13     IF node  $u$  is found THEN
14          $u$  is removed from set  $Q$ 
15          $u$  is added in set  $Se$ 
16     FOR each node  $b$  in  $Q$ , a node that is connected from node  $u$  using a single arc
17         IF (the assignment of  $b$  from  $u$  valid feasible assignment) THEN
18             Determine idle time,  $new\_idle_b$  and sub makespan,  $s\_makespan_b$ 
19             IF (( $new\_idle_b < idle_b$ ) AND ( $s\_makespan_b < best\_makespan$ )) THEN
20                  $idle_b$  is replaced by  $new\_idle_b$ 
21                 SET  $u = the\ previous_v$  of node  $b$ 
22                 IF node  $b$  is the last workstation
23                     IF  $s\_makespan_b < best\_makespan$  THEN update  $best\_makespan$ 
24                     END IF
25                 END IF
26             END IF
27         END IF
28     END Loop
29     ELSE  $Have\_min$  = false End IF
30 END WHILE

```

Figure 4.7 The pseudo code of Modified Dijkstra's algorithms

Trial	The Modified Dijkstra's algorithm																																																																																																												
The 1 st Trial UB=29 LB=16	Trial value = $16 + (29-16) \times 0.1 = 17.3$ Best makespan = UB = 29																																																																																																												
	<p>Start at node 12A $Se = \{\emptyset\}$, $Q = \{12A, 3B, 45A, 45C\}$ best_makespan = 29</p> <table border="1" data-bbox="630 571 1204 728"> <thead> <tr> <th>Node</th> <th>12A</th> <th>3B</th> <th>45A</th> <th>45C</th> </tr> </thead> <tbody> <tr> <td>$idle_v$</td> <td>0</td> <td>∞</td> <td>∞</td> <td>∞</td> </tr> <tr> <td>$s_makespan_v$</td> <td>∞</td> <td>∞</td> <td>∞</td> <td>∞</td> </tr> <tr> <td>$previous_v$</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> </tr> </tbody> </table> <p>Node u = 12A $Se = \{12A\}$, $Q = \{3B, 45A, 45C\}$ best_makespan = 29</p> <table border="1" data-bbox="630 884 1204 1052"> <thead> <tr> <th colspan="2"></th> <th colspan="4">Node b</th> </tr> <tr> <th>Node</th> <th></th> <th>12A</th> <th>3B</th> <th>45A</th> <th>45C</th> </tr> </thead> <tbody> <tr> <td>$idle_v$</td> <td></td> <td>0</td> <td>7</td> <td>∞</td> <td>∞</td> </tr> <tr> <td>$s_makespan_v$</td> <td></td> <td>-</td> <td>23</td> <td>-</td> <td>-</td> </tr> <tr> <td>$previous_v$</td> <td></td> <td>-</td> <td>12A</td> <td>-</td> <td>-</td> </tr> </tbody> </table> <p>Node u = 3B $Se = \{12A, 3B\}$, $Q = \{45A, 45C\}$</p> <table border="1" data-bbox="630 1176 1204 1344"> <thead> <tr> <th colspan="2"></th> <th colspan="2">Node b¹</th> <th colspan="2">Node b²</th> </tr> <tr> <th>Node</th> <th></th> <th>12A</th> <th>3B</th> <th>45A</th> <th>45C</th> </tr> </thead> <tbody> <tr> <td>$idle_v$</td> <td></td> <td>0</td> <td>7</td> <td>∞</td> <td>13</td> </tr> <tr> <td>$s_makespan_v$</td> <td></td> <td>-</td> <td>23</td> <td>-</td> <td>28</td> </tr> <tr> <td>$previous_v$</td> <td></td> <td>-</td> <td>12A</td> <td>-</td> <td>3B</td> </tr> </tbody> </table> <p>Node b = 45A : Infeasible assignment solution(12A,3B,45A) Node b = 45C : assignment solution (12A,3B,45C) best_makespan = 28</p> <p>Start at node 12C $Se = \{\emptyset\}$, $Q = \{12A, 3A, 3B, 45A, 45B, 45C\}$ best_makespan = 28</p> <table border="1" data-bbox="630 1624 1388 1780"> <thead> <tr> <th>Node</th> <th>12C</th> <th>3A</th> <th>3B</th> <th>45A</th> <th>45B</th> <th>45C</th> </tr> </thead> <tbody> <tr> <td>$idle_v$</td> <td>0</td> <td>∞</td> <td>∞</td> <td>∞</td> <td>∞</td> <td>∞</td> </tr> <tr> <td>$s_makespan_v$</td> <td>∞</td> <td>∞</td> <td>∞</td> <td>∞</td> <td>∞</td> <td>∞</td> </tr> <tr> <td>$previous_v$</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> </tr> </tbody> </table> <p>Node u = 12C $Se = \{12C\}$, $Q = \{3A, 3B, 45A, 45B, 45C\}$ best_makespan = 28</p>	Node	12A	3B	45A	45C	$idle_v$	0	∞	∞	∞	$s_makespan_v$	∞	∞	∞	∞	$previous_v$	-	-	-	-			Node b				Node		12A	3B	45A	45C	$idle_v$		0	7	∞	∞	$s_makespan_v$		-	23	-	-	$previous_v$		-	12A	-	-			Node b ¹		Node b ²		Node		12A	3B	45A	45C	$idle_v$		0	7	∞	13	$s_makespan_v$		-	23	-	28	$previous_v$		-	12A	-	3B	Node	12C	3A	3B	45A	45B	45C	$idle_v$	0	∞	∞	∞	∞	∞	$s_makespan_v$	∞	∞	∞	∞	∞	∞	$previous_v$	-	-	-	-	-	-
Node	12A	3B	45A	45C																																																																																																									
$idle_v$	0	∞	∞	∞																																																																																																									
$s_makespan_v$	∞	∞	∞	∞																																																																																																									
$previous_v$	-	-	-	-																																																																																																									
		Node b																																																																																																											
Node		12A	3B	45A	45C																																																																																																								
$idle_v$		0	7	∞	∞																																																																																																								
$s_makespan_v$		-	23	-	-																																																																																																								
$previous_v$		-	12A	-	-																																																																																																								
		Node b ¹		Node b ²																																																																																																									
Node		12A	3B	45A	45C																																																																																																								
$idle_v$		0	7	∞	13																																																																																																								
$s_makespan_v$		-	23	-	28																																																																																																								
$previous_v$		-	12A	-	3B																																																																																																								
Node	12C	3A	3B	45A	45B	45C																																																																																																							
$idle_v$	0	∞	∞	∞	∞	∞																																																																																																							
$s_makespan_v$	∞	∞	∞	∞	∞	∞																																																																																																							
$previous_v$	-	-	-	-	-	-																																																																																																							

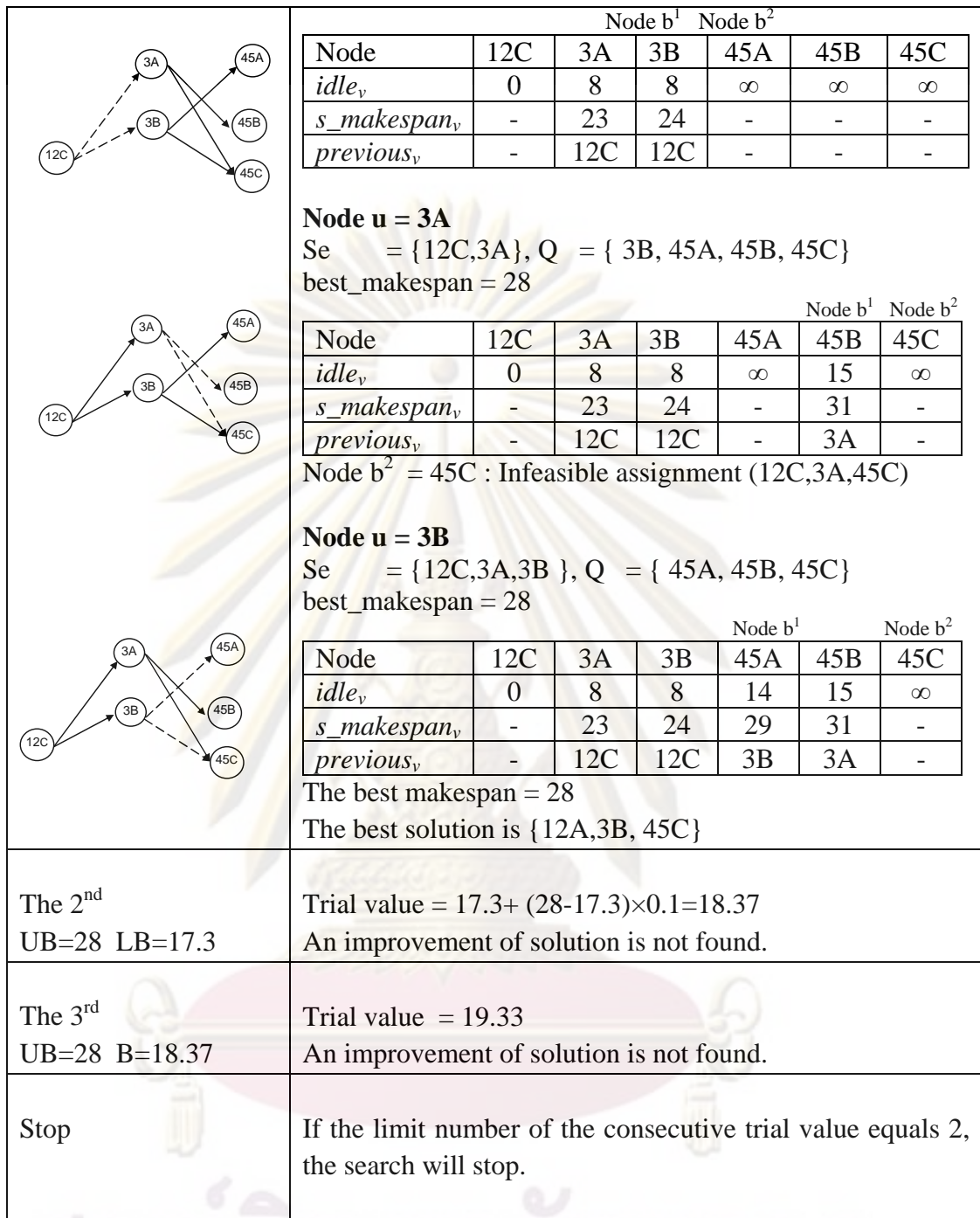


Figure 4.8 An example of the Modified Dijkstra's algorithm

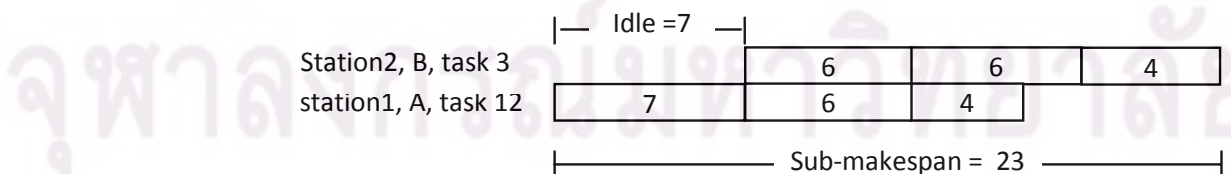


Figure 4.9 An example of idle and sub-makespan

4.5 Performance measurement of the heuristic

4.5.1 Testing the problems

A heuristic algorithm was implemented using the C++ programming language. The quality of the solution and computational time of the proposed heuristic were compared to those obtained from the mathematical model which was solved by CPLEX 8.0. Both CPLEX and the heuristic algorithm were run on a PC with an Intel Core™2 Duo 2.00 GHz CPU and 1.93 GB of RAM.

Normally in a modular production system in the garment industry, the number of workers is between 6 and 15. However, only the problem with 7-9 workers was tested since a problem size of 10-15 workers is too large to achieve optimal solutions using the mathematical model. The problem parameters were chosen to reflect a realistic situation in the garment industry. In this testing, the number of tasks was three times to four times the number of workers, since generally in the garment industry a worker is assigned less than three to four tasks. The numbers of items that were examined were 100 and 300 for the difference of lot sizes. Furthermore, the difference in learning period is also reflected. Regarding the learning process, a major reduction in the processing time occurs during the beginning of the process. Therefore, for the 100 item problem, major reduction is the large part of the process, whereas the 300 item problems include both major and minor reduction. The proportion of minor reduction is greater than that of major reduction

Table 4.4 shows all the problem sets of this test. The parameters of the problem instances were developed. Based on Log-Linear model, if t_1 and t_n represent the task processing times of the first and the n^{th} item, and using ϕ in terms of learning slope, $t_n = t_1 \cdot n^{(\log \phi / \log 2)}$ (Wright, 1936). The learning slope for apparel manufacturing is reported by Rosenwasser (1982) and is 0.77-0.85. In this test, the learning slope was uniformly generated from the interval between [0.77, 0.85] and [0.70, 0.90] in two levels. Since the task processing of the sewing process varies, it was uniformly generated. The mean task processing times of the first item (t_1) were generated uniformly in the intervals [1, 10] and [1, 30]. The differences in performance of experienced and inexperienced workers on the sewing line are represented by the percent deviations from the mean task processing times. The percent deviations are

uniformly distributed in the intervals $[-20\%, 20\%]$ and $[-50\%, 50\%]$. For example, if the mean task processing time of the first item is 10 and the percentage of deviation 20%, t_1 becomes 12.

Table 4.4 Problem Size

Problem Code	Number of Worker	Number of Task	Number of item	Instances	CPLEX Sol.
7w21t100i	7	21	100	80	Optimal Sol.
8w24t100i	8	24	100	80	Optimal Sol.
9w27t100i	9	27	100	40	Limit time*
7w21t300i	7	21	300	80	Optimal Sol.
8w24t300i	8	24	300	40	Optimal Sol.
9w27t300i	9	27	300	40	Limit Time**
7w28t100i	7	28	100	80	Optimal Sol.
8w32t100i	8	32	100	80	Optimal Sol.

Note * Limit time = 259,200 sec. or (3days), **Limit time = 345,500 sec. or (4days)

There are 8 tests per problem size. Each test has 10 replicates, but for 8w24t300i, 9w27t100i and 9w27t300i, we used 5 replicates, so the number of instances are 40 for 8w24t300i, 9w27t100i and 9w27t300i and 80 for the other problems. For the problem sets 7w21t100i, 7w21t300i, 8w24t100i and 8w24t300i, CPLEX was designed to run until the optimal solution was found. Since the problem size of 9 workers is quite large to achieve the optimal solutions, the run time of the problem was limited. For 9 workers with 100 items, 9w27t100i, the limited run time was 259,200 sec. (3 days). For 9 workers with 300 items, 9w27t300i, the limited run time was 345,500 sec. (4 days) due to the larger size of the problem.

4.5.2 The parameters applied in the heuristic

Four parameters were applied in the heuristic, i.e. the gap to set the trial value (*Gap*), the limit number of the loop (*Limit*), % of special group and the number of solutions that were kept.

The gap to set trial value is to the position of trial value between the UB and LB and was calculated from the percentage of the difference between the two. The gap value affects the computational time and quality of the solution and the computational time is reduced for the large gap. However, the optimal solution may

be missed if the gap is too large. To determine the suitable parameter, the gap to set trial value was designed in two levels, *Gap* {0.2, 0.3}.

The limit number of loop (*Limit*) is the stopping criteria. For example, the heuristic will stop if the best solution is not improved greater than or equal to 3 loops of the trial makespan. Figure 4.10 shows an example of an improvement of makespan when the limit number is three. The computational time is reduced when the limit number of the loop is small. However the optimal solution may be missed if the loop stops too early. To determine a suitable parameter, the limit number of the loop (*Limit*) was designed in two levels, *Limit* {3, 5}.

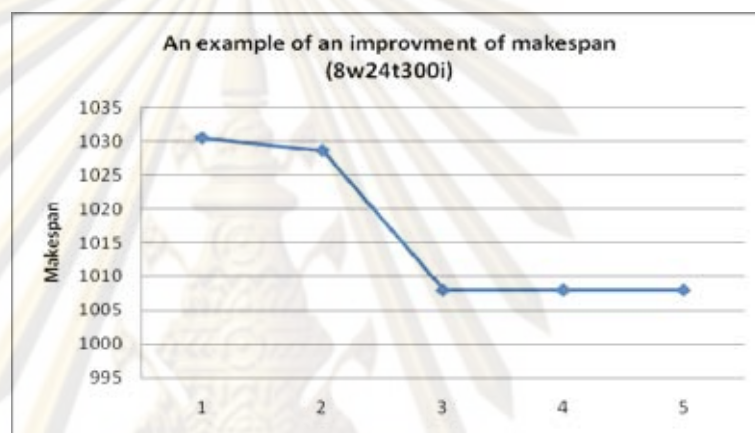


Figure 4.10 An example of an improvement of makespan

The heuristic also allows a sub set of the groups which has a summation of task processing times close to the generated group to add in the heuristic. If the number of groups increase, the chance to find the optimal solution will increase while the computational time also increases e.g. Task {1, 2, 3} is the generated group of tasks which has 10 sec. Let % of *special group* be equal to 10%, if the tasks 1, 2 consume 9 sec. The group of tasks 1, 2 will be the special group which is included in the heuristic. To determine a suitable parameter, % of *special group* was designed in two levels, {10%, 15%}.

Since the heuristic was designed to minimize idle time that is not a direct algorithm to minimize makespan, it is better to keep a set of solutions in a node. In the modified Dijkstra's algorithm, more than one solution is kept, so the computational time will increase if the number of solutions increases. In the heuristic, the number of solutions is designed in two levels, two times and three times the

number of workers, {2w, and 3w}. If the number of the workers is 7, then the number of the solutions will be 14 and 21.

The problem size 7w21t100i was tested. The test started at the level of parameters which gives the quickest computational times which are, $Gap\{0.3\}, Limit\{3\}, \% \text{ of special group}\{10\}$ and the number of solutions that are stored in a node{2w} in Test 1. Then the quality of the solution and the computational time were investigated. The results of the tests are in Table 4.5. The percentage difference of the heuristic solution from the optimal solution was calculated as the following equation:

$$\% \text{ difference from the optimal solution} = \text{diff} = \frac{Heu_{sol} - Opt_{sol}}{Opt_{sol}} \times 100$$

where Heu_{sol} is the makespan obtained by a heuristic, and Opt_{sol} is the makespan of the optimal solution obtained by using an exact algorithm.

Each factor was adjusted to another level one at a time and then the quality of the solution and the computational time were investigated. The results of the test are shown in Table 4.5, Figures 4.11 and 4.12. Since no test significantly improves the quality of the solution of Test 1 as shown in Table 4.5, the parameters of Test 1 were selected to be suitable parameters which are $Gap\{0.3\}, Limit\{3\}, \% \text{ of special group}\{10\}$ and the number of solutions that are stored in a node{2w}.

Table 4.5 The quality of the solution and CPU Time of the test when the parameters are varied

	<i>Gap</i>	<i>Limit</i>	<i>% of special group</i>	<i>No. sol. stored in a node</i>	Avg. diff (%)	Max. diff (%)	CPU Time (sec.)
Test 1	0.3	3	10%	14	0.109	2.762	5.420
Test 2	0.2	3	10%	14	0.239	2.768	5.751
Test 3	0.3	5	10%	14	0.109	2.762	7.647
Test 4	0.3	3	15%	14	0.094	2.762	7.218
Test 5	0.3	3	10%	21	0.109	2.762	7.647

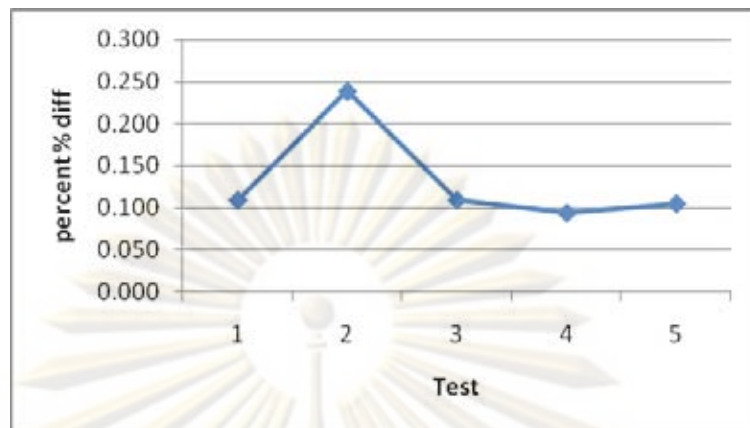


Figure 4.11 The quality of the solution of the test when the parameters are varied(%)

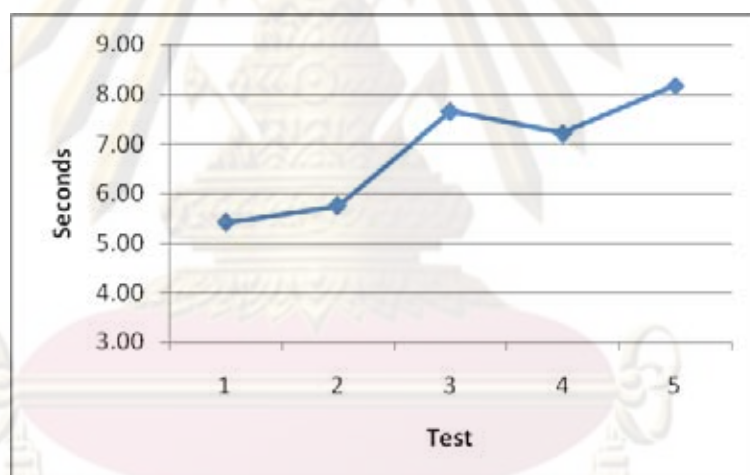


Figure 4.12 The CPU of the test when the parameters are varied(sec.)

4.5.3 Performance of heuristics on test problems

The purpose of these experiments was to evaluate the performance of the proposed algorithm on the test problems. In Sections 4.5.3.1- 4.5.3.2 the performance of heuristics are tested in two aspects; solution quality, and computational time. The comparison of the quality of the solution between the problems applying the constant skill level and the skill level including learning ability are investigated in section 4.5.3.3.

4.5.3.1 The computational time of the heuristic

This dissertation compares the computational time of the proposed heuristic to the computational time from CPLEX solution. The task processing time of the first item, the % deviation of the processing time among the workers and the learning slope are varied in two levels as mentioned in Section 4.5.1. Table 4.6 presents the computational time of the problem when the number of tasks is equal to 3 times the number of workers with 100 items. Table 4.7 presents the computational time of the problem when the number of tasks is equal to 3 times the number of workers with 300 items. Table 4.8 presents the computational time of the problem when the number of tasks is equal to 4 times the number of workers with 100 items for problems using 7 and 8 workers.

Table 4.6 CPU time of the problem : No. tasks = 3× No. workers , 100 items(sec.)

Processing Time	Skill Level	Learning Slope	Computational time (sec.)					
			7w21t100i		8w24t100i		9w27t100i	
			Opt.	Heu.	Opt.	Heu.	Opt.	Heu.
U[1,10]	U[1,20]	U[0.77,0.85]	801.845	3.821	1,479.828	10.74	38,847.11	43.66
U[1,10]	U[1,20]	U[0.70,0.90]	2,704.773	5.931	15,317.14	14.04	221,328.11	71.85
U[1,10]	U[1,50]	U[0.77,0.85]	1,253.58	5.26	13,872.80	14.73	75,117.24	63.93
U[1,10]	U[1,50]	U[0.70,0.90]	3,220.02	8.18	18,364.12	19.81	133,728.19	42.29
U[1,30]	U[1,20]	U[0.77,0.85]	427.74	2.12	1,920.55	27.08	8,508.04	91.15
U[1,30]	U[1,20]	U[0.70,0.90]	1,474.55	6.76	21,773.07	19.23	127,300.91	76.73
U[1,30]	U[1,50]	U[0.77,0.85]	529.93	4.85	17,706.71	12.24	87,128.28	54.82
U[1,30]	U[1,50]	U[0.70,0.90]	2,383.27	4.56	61,631.84	16.55	211,616.94	51.86

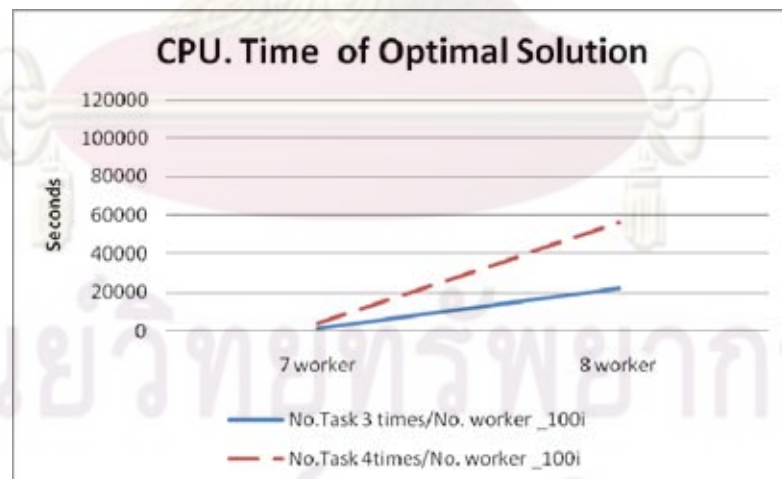
Table 4.7 CPU time of the problem : No. tasks = 3× No. workers , 300 items(sec.)

Processing Time	Skill Level	Learning Slope	Computational time (sec.)					
			7w21t300i		8w24t300i		9w27t300i	
			Opt.	Heu.	Opt.	Heu.	Opt.	Heu.
U[1,10]	U[±20]	U[0.77,0.85]	801.845	3.821	16,687.96	39.79	-	86.95
U[1,10]	U[±20]	U[0.70,0.90]	2,704.773	5.931	160,363.91	45.37	-	189.15
U[1,10]	U[±50]	U[0.77,0.85]	5,964.87	7.93	41,142.10	35.01	-	148.73
U[1,10]	U[±50]	U[0.70,0.90]	12,423.63	12.62	237,607.59	37.21	-	105.21
U[1,30]	U[±20]	U[0.77,0.85]	3,148.66	5.43	26,227.23	25.34	-	239.47
U[1,30]	U[±20]	U[0.70,0.90]	14,189.63	13.37	185,995.06	39.26	-	138.38
U[1,30]	U[±50]	U[0.77,0.85]	3,297.25	9.73	110,749.60	45.85	-	228.19
U[1,30]	U[±50]	U[0.70,0.90]	11,183.15	10.10	189,963.72	56.49	-	198.97

For the computational time of the optimal solution, it was found that when the number of tasks and the number of workers are increased, the computational time of the optimal solution will increase as shown in Figure 4.13. And when the number of items and the number of workers are increased, the computational time of the optimal solution will dramatically increase as shown in Figure 4.14.

Table 4.8 CPU time of the problem : No. tasks = 4× No. workers , 100 items(sec.)

Processing Time	Skill Level	Learning Slope	Computational time (sec.)			
			7w28t100i		8w32t100i	
			Opt.	Heu.	Opt.	Heu.
U[1,10]	U[±20]	U[0.77,0.85]	1,262.44	9.87	6,015.19	14.50
U[1,10]	U[± 20]	U[0.70,0.90]	7,420.15	26.63	56,597.09	40.93
U[1,10]	U[±50]	U[0.77,0.85]	4,844.74	21.80	86,142.74	39.30
U[1,10]	U[±50]	U[0.70,0.90]	4,809.68	19.75	85,285.44	40.93
U[1,30]	U[±20]	U[0.77,0.85]	1,262.19	13.81	22,177.36	29.34
U[1,30]	U[±20]	U[0.70,0.90]	4,232.49	10.61	57,779.92	35.12
U[1,30]	U[±50]	U[0.77,0.85]	3,880.08	8.98	39,668.41	34.66
U[1,30]	U[±50]	U[0.70,0.90]	5,822.06	10.65	92,853.20	42.91

**Figure 4.13 CPU Time of optimal sol. when No. Tasks increase (sec.)**

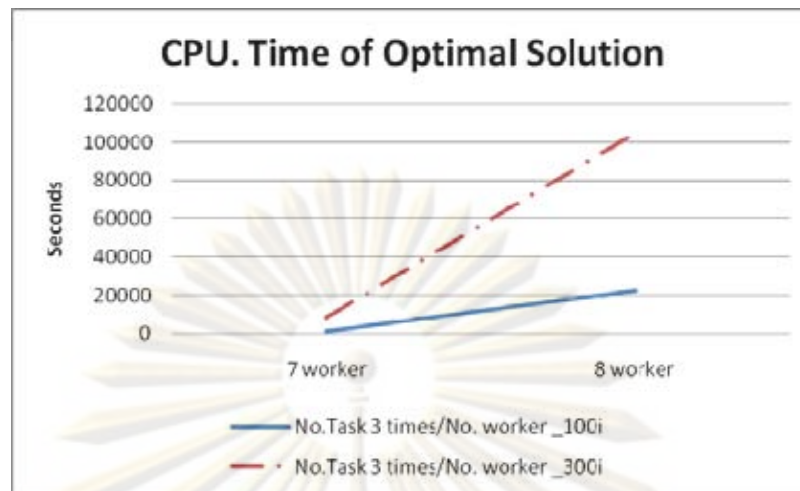


Figure 4.14 CPU Time of optimal sol. when No. Items increase (sec.)

Furthermore, when the learning slope of the worker varies at a high level, it was found that the computational time of the optimal solution is greater than the computational time when learning slope varies at a low level as seen in Figure 4.15.



Figure 4.15 CPU. Time of optimal sol. when learning slope varies (sec.)

For the computational time of the proposed heuristic, it was found that the total computational time of the heuristic is less than the computational time from CPLEX solution. When the number of tasks and the number of workers are increased, the computational time of the heuristic solution will slightly increase as shown in Figure 4.16. However when the number of items and the number of workers are increased, the total computational time of the heuristic solution will increase, especially for 9 workers as shown in Figure 4.17.

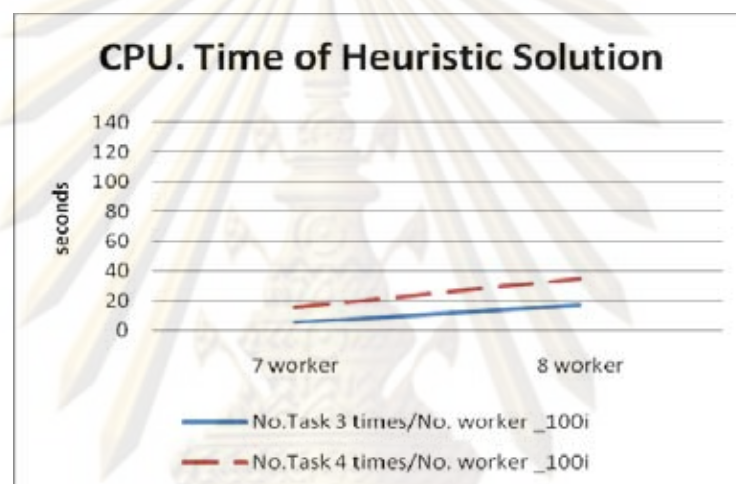


Figure 4.16 CPU. Time of heuristic sol. when No. Tasks increase(sec.)

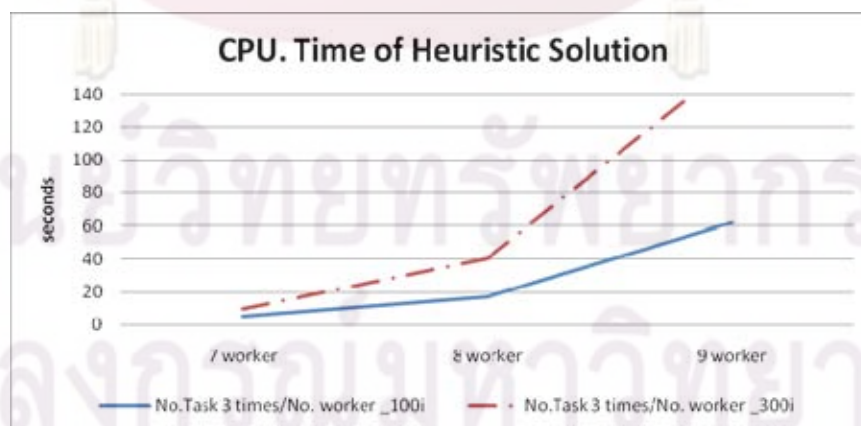


Figure 4.17 CPU. Time of heuristic sol. when No. Items increase(sec.)

To investigate the computational time of the heuristic, the heuristic step was divided into three parts which are generating UB and LB, determining the feasible path and applying the modified Dijkstra's algorithm. It was found that determining the feasible path consumes 81% of the total time, generating UB and LB consumes 18% of the total time, and applying the modified Dijkstra's algorithm takes only 1% of the total time as seen in Figure 4.18.

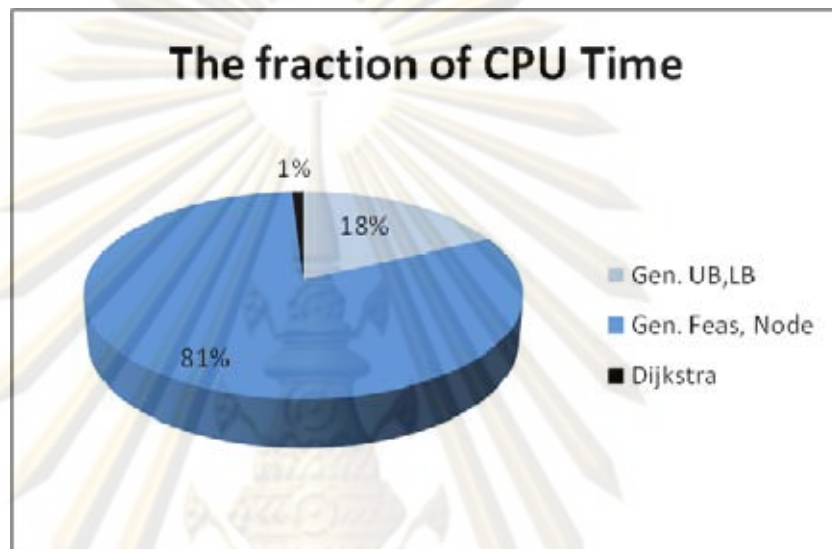


Figure 4.18 The fraction of CPU. Time (%)

4.5.3.2 The quality of the solution of the heuristic

The performance of heuristics is presented in the form of the percentage difference of a heuristic from the solution of CPLEX solution. The percentage difference of the heuristic solution from the optimal solution was calculated as the following equation:

$$\% \text{ difference from the optimal solution} \quad diff = \frac{Heu_{sol} - Opt_{sol}}{Opt_{sol}} \times 100$$

where Heu_{sol} is the makespan obtained by a heuristic, and Opt_{sol} is the makespan from the optimal solution obtained by using an exact algorithm.

The offset from the optimal including mean (Avg. diff), the standard deviation (Std. diff) and the maximum of the difference from the optimal solution (Max diff) were calculated. When the optimal solution was not achieved within the time limit for problem 9w27t100i and 9w27t300i, the best objective value within the time limit was compared to the heuristic solution. Table 4.9 shows the percentage difference of the problem when the number of tasks is equal to 3 times the number of workers for 100 items. Table 4.10 shows the percentage difference of the problem when the number of tasks is equal to 3 times the number of workers for 300 items. For the problem size of 9 workers, since the time of exact solution test is limited, some tests may found the optimal solution. In the column “Exact Opt”, the problem 9w27t300i has two parts which consist of the solutions which were found to be the optimal solution and the solutions which were not found to be the optimal solution within the time limit. The number of the optimal solutions for each test is presented in the column. For example, 1/5 means that from 5 tests, 1 test has the optimal solution within the limit time. Table 4.11 presents the percentage difference of the problem when the number of tasks is equal to 4 times the number of workers with 100 items for problems of 7 and 8 workers. In Figures 4.19 - 4.24, the percentage difference of the heuristic from the solution of exact solution of each problem size is presented.

It was found that the heuristic gives the percentage difference from the optimal solution at less than 0.80 %. The maximum of the difference was less than 4.6% for all tests.

Table 4.9 The percentage difference of the problem :No. tasks = 3× No. workers, 100item(%)

Processing Time	Skill Level	Learning Slope	The difference between the exact solution and the heuristic solution (%)								
			7w21t100i			8w24t100i			9w27t100i		
			<i>Avg. diff</i>	<i>Std. diff</i>	<i>Max. diff</i>	<i>Avg. diff</i>	<i>Std. diff</i>	<i>Max. diff</i>	<i>Exact Opt.</i>	<i>Avg. diff</i>	<i>Max. diff</i>
U[1,10]	U[±20]	U[0.77,0.85]	0.38	0.89	2.76	0.20	0.64	2.01	5/5	1.08	3.16
									Not Opt.	-	-
U[1,10]	U[±20]	U[0.70,0.90]	0.03	0.09	0.29	0.52	0.52	1.43	1/5	0.58	0.58
									Not Opt.	-1.39	0.01
U[1,10]	U[±50]	U[0.77,0.85]	0.13	0.31	0.98	0.14	0.46	1.44	5/5	0.49	1.50
									Not Opt.	-	-
U[1,10]	U[±50]	U[0.70,0.90]	0.08	0.22	0.69	0.21	0.35	0.98	3/5	1.00	2.62
									Not Opt.	4.48	7.50
U[1,30]	U[±20]	U[0.77,0.85]	0.05	0.13	0.42	0.39	0.91	2.86	5/5	0.38	0.89
									Not Opt.	-	-
U[1,30]	U[±20]	U[0.70,0.90]	0.08	0.22	0.71	0.33	0.54	1.44	4/5	0.71	2.09
									Not Opt.	- 0.68	- 0.68
U[1,30]	U[±50]	U[0.77,0.85]	0.00	0.01	0.03	0.43	1.32	4.18	4/5	0.13	0.36
									Not Opt.	- 1.39	- 1.39
U[1,30]	U[±50]	U[0.70,0.90]	0.04	0.06	0.16	0.29	0.37	1.08	3/5	1.49	1.99
									Not Opt.	1.15	1.70

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

Table 4.10 The percentage deviation of the problem : No. tasks = 3× No. workers, 300 items(%)

Processing Time	Skill Level	Learning Slope	The difference between the exact solution and the heuristic solution (%)								
			7w21t300i			8w24t300i			9w27t300i		
			<i>Avg. diff</i>	<i>Std. diff</i>	<i>Max. diff</i>	<i>Avg. diff</i>	<i>Std. diff</i>	<i>Max. diff</i>	<i>Exact Opt.</i>	<i>Avg. diff</i>	<i>Max. diff</i>
U[1,10]	U[±20]	U[0.77,0.85]	0.00	0.00	0.00	0.39	0.57	1.67	4/5	0.01	0.04
									Not Opt.	-1.27	-1.27
U[1,10]	U[± 20]	U[0.70,0.90]	0.06	0.19	0.61	0.70	1.26	3.53	0/5	-	-
									Not Opt.	-2.12	-0.38
U[1,10]	U[±50]	U[0.77,0.85]	0.14	0.33	1.05	0.07	0.20	0.63	2/5	0.91	1.79
									Not Opt.	-0.68	2.51
U[1,10]	U[±50]	U[0.70,0.90]	0.10	0.22	0.57	0.01	0.02	0.06	0/5	-	-
									Not Opt.	-1.71	-0.79
U[1,30]	U[±20]	U[0.77,0.85]	0.55	1.46	4.63	0.17	0.43	1.15	3/5	0.14	0.42
									Not Opt.	-0.53	-0.48
U[1,30]	U[±20]	U[0.70,0.90]	0.14	0.38	1.22	0.00	0.00	0.00	1/5	1.23	1.23
									Not Opt.	-0.82	1.03
U[1,30]	U[±50]	U[0.77,0.85]	0.26	0.81	2.56	0.03	0.04	0.10	2/5	0.00	0.00
									Not Opt.	- 1.87	- 1.39
U[1,30]	U[±50]	U[0.70,0.90]	0.14	0.33	1.05	0.00	0.00	0.00	0/5	-1.61	- 0.40
									Not Opt	1.15	1.70

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

Table 4.11 The percentage difference of the problem : No. tasks = 4× No. workers, 100 items(%)

Processing Time	Skill Level	Learning Slope	The difference between the exact solution and the heuristic solution (%)						
			7w28t100i			8w24t100i			
			Avg. diff	Std. diff	Max. diff	Avg. diff	Std. diff	Max. diff	Avg. diff
U[1,10]	U[±20]	U[0.77,0.85]	0.21	0.61	1.95	0.04	0.09	0.29	0.04
U[1,10]	U[± 20]	U[0.70,0.90]	0.02	0.06	0.17	0.02	0.03	0.07	0.02
U[1,10]	U[±50]	U[0.77,0.85]	0.23	0.54	1.65	0.05	0.12	0.37	0.05
U[1,10]	U[±50]	U[0.70,0.90]	0.00	0.00	0.01	0.34	0.70	2.28	0.34
U[1,30]	U[±20]	U[0.77,0.85]	0.00	0.00	0.00	0.13	0.34	1.07	0.13
U[1,30]	U[±20]	U[0.70,0.90]	0.00	0.00	0.00	0.43	0.66	1.68	0.43
U[1,30]	U[±50]	U[0.77,0.85]	0.08	0.21	0.66	0.20	0.39	0.98	0.20
U[1,30]	U[±50]	U[0.70,0.90]	0.11	0.33	1.05	0.08	0.25	0.79	0.08

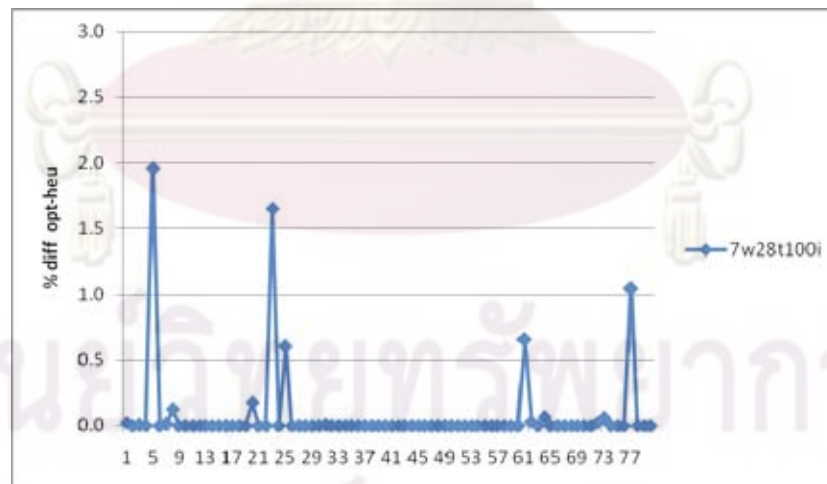


Figure 4.19 The percentage difference 7w28t100i(%)

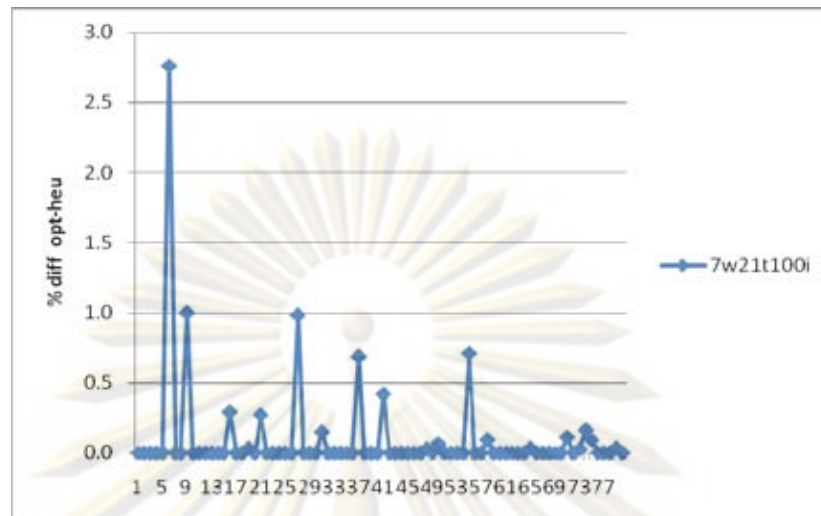


Figure 4.20 The percentage difference 7w21t100i (%)

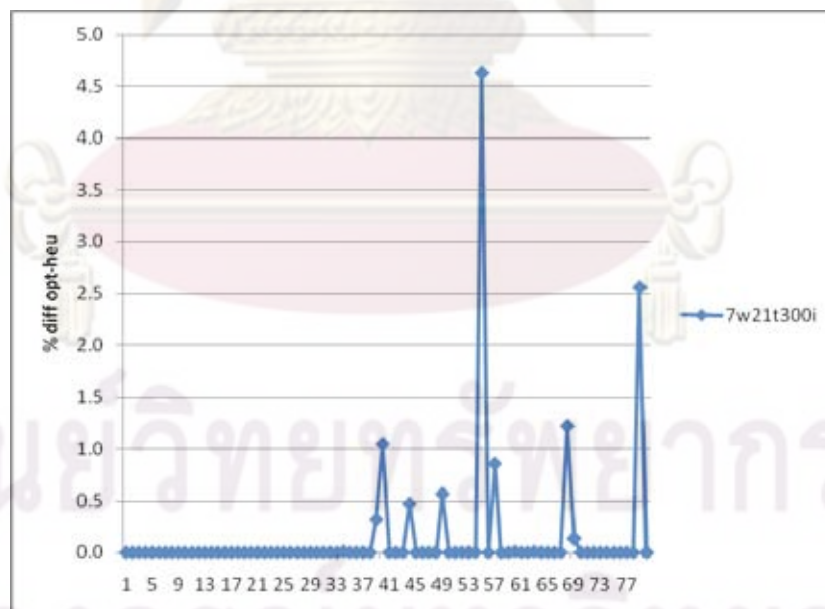


Figure 4.21 The percentage difference 7w21t300i (%)

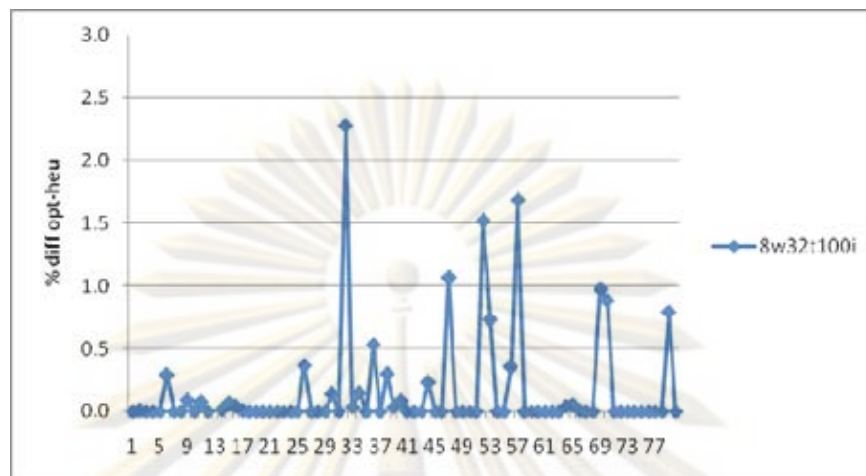


Figure 4.22 The percentage difference 8w32t100i(%)

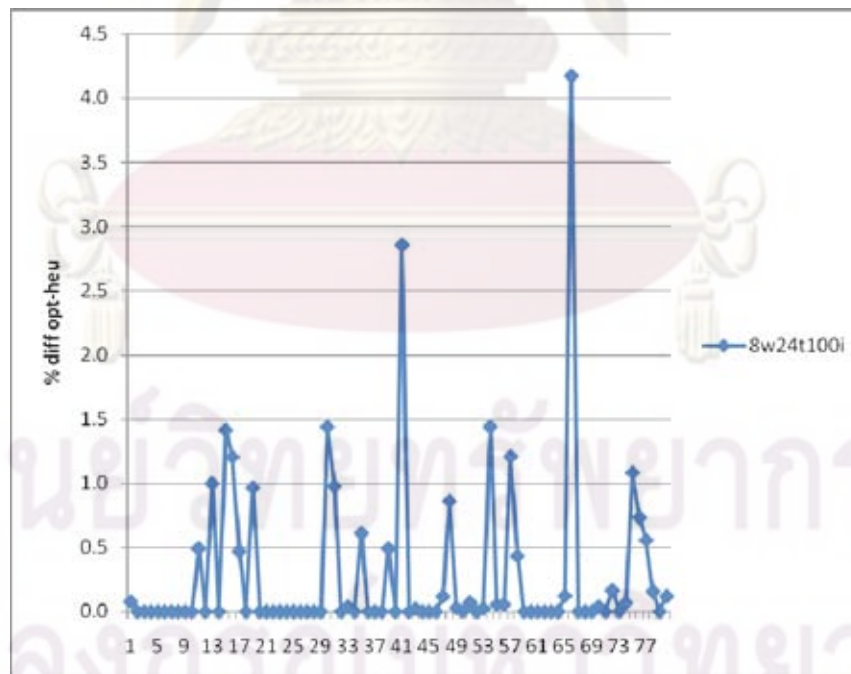


Figure 4.23 The percentage difference 8w24t100i (%)

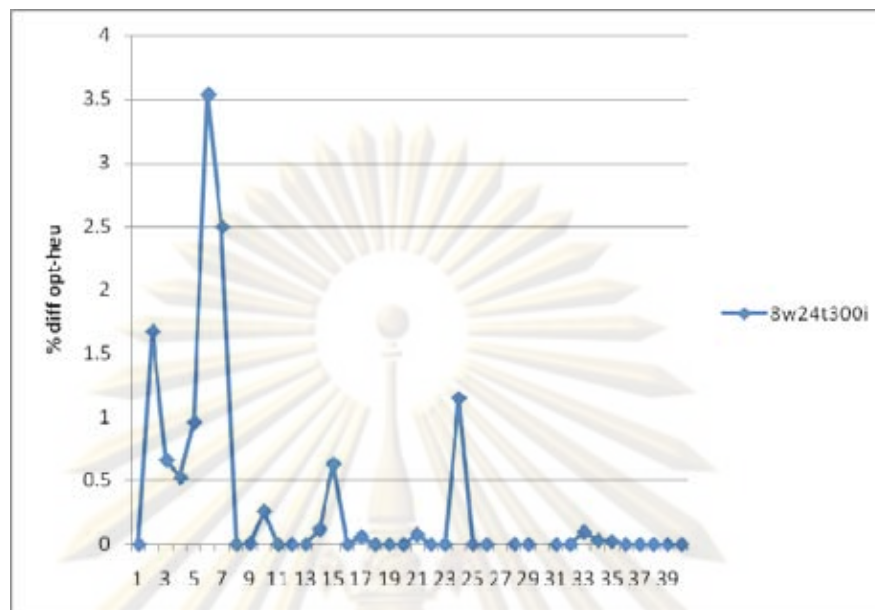


Figure 4.24 The percentage difference 8w24t300i (%)

It is found that the percentage difference will increase when the number of workers increases. Producing 100 items has the percentage difference higher than 300 items as seen in Figure 4.20. The reason is that the heuristic group tasks are based on the sum of processing times, so the sum of processing times from 300 items can better represent the learning effect than the sum of processing times from 100 items. For example, the sums of processing times of task 1 for 100 items from workers A and B are equal, while the learning slopes of worker A and worker B are different for 100 items. However, if the production is 300 items, the sum of processing times is significantly different. For this reason, the heuristic achieves the better solution.

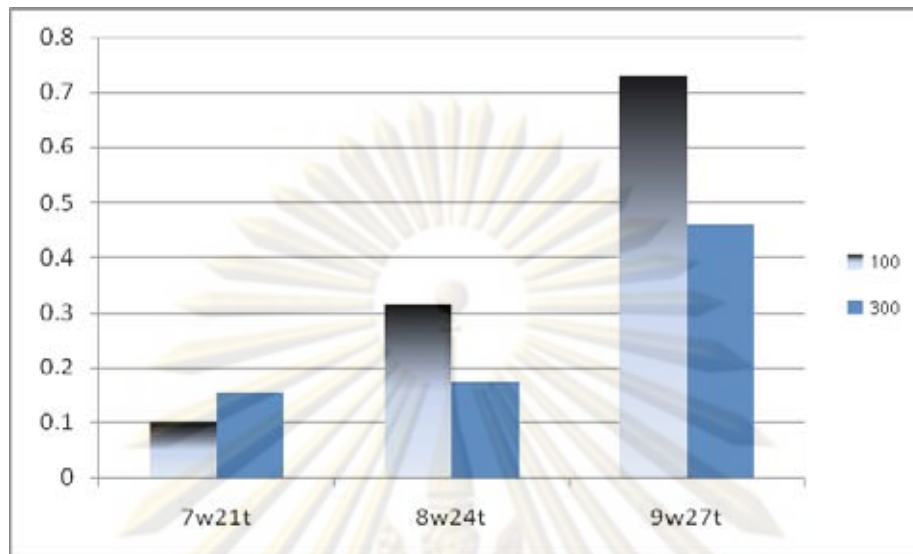


Figure 4.25 The percentage difference when No. Tasks and No.Items increase(%)

4.5.3.3 The comparison of the quality of the solution of the heuristic to the heuristic for the constant skill level

This section is to study the effect of using the constant skill level in situations accounting for worker learning ability. The quality of the solution was investigated. The constant skill level was determined from the summation of task processing times. Based on the data, the problem was solved according to the heuristic method in Chapter 3. We wanted to know the quality of the solution when it is used in situations where workers' learning ability is a factor. The performance is presented in the form of the percentage difference of makespan. The percentage difference of the constant skill level solution from the heuristic solution was calculated as the following equation:

$$\% \text{ difference from the heuristic solution } \quad diff_{con_heu} = \frac{Con_{sol} - Heu_{sol}}{Heu_{sol}} \times 100$$

where Con_{sol} is the makespan obtained by a heuristic for constant skill level, and Heu_{sol} is the makespan from the heuristic solution obtained by the heuristic in Section 4.3.

Table 4.12 shows the percentage difference between the constant skill level and the heuristic solution. It was found that when the learning slope among workers has high variation, their percentage difference will be higher as in Figure 4.26. It was also found that the percentage difference of the learning slope among workers which varies at a low level is low. When the learning slope among workers has high variation for both of 100 and 300 items, there is a higher percentage difference than the low variation as shown in Figure 4.27. This means that the assumption of constant skill level is appropriate for the small difference in learning slope among workers and for large lots.

Table 4.12 The percentage difference of the constant skill level solution from the heuristic solution (%)

Process	The difference between the constant skill level solution and the heuristic solution (%)									
	Skill Level	Learning Slope	7w21t 100i	7w28t 100i	7w21t 300i	8w24t 100i	8w32t 100i	8w24t 300i	9w27t 100i	9w27t 300i
U[1,10]	U[±20]	U[0.77,0.85]	3.43	5.36	1.64	5.36	2.86	3.43	5.36	1.64
U[1,10]	U[±20]	U[0.70,0.90]	5.20	6.37	4.95	6.49	5.33	5.20	6.37	4.95
U[1,10]	U[±50]	U[0.77,0.85]	4.63	3.26	2.97	4.01	2.20	4.63	3.26	2.97
U[1,10]	U[±50]	U[0.70,0.90]	6.83	5.77	5.37	4.25	7.57	6.83	5.77	5.37
U[1,30]	U[±20]	U[0.77,0.85]	4.65	4.12	1.40	5.23	1.57	4.65	4.12	1.40
U[1,30]	U[±20]	U[0.70,0.90]	7.99	7.72	4.66	7.71	1.77	7.99	7.72	4.66
U[1,30]	U[±50]	U[0.77,0.85]	3.13	4.19	2.24	6.19	2.13	3.13	4.19	2.24
U[1,30]	U[±50]	U[0.70,0.90]	5.20	8.47	5.80	7.55	4.42	5.20	8.47	5.80

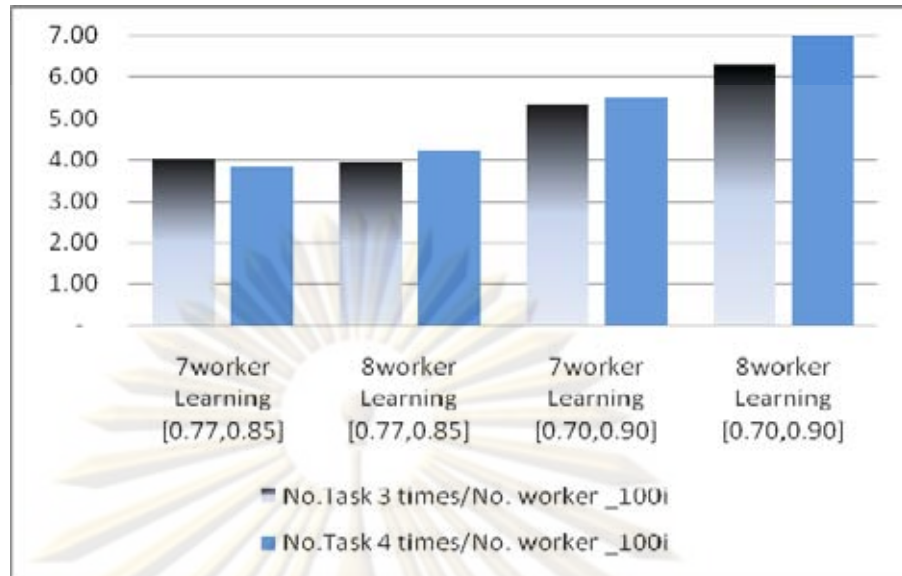


Figure 4.26 The percentage difference $diff_{con_heu}$ when No. Tasks increase

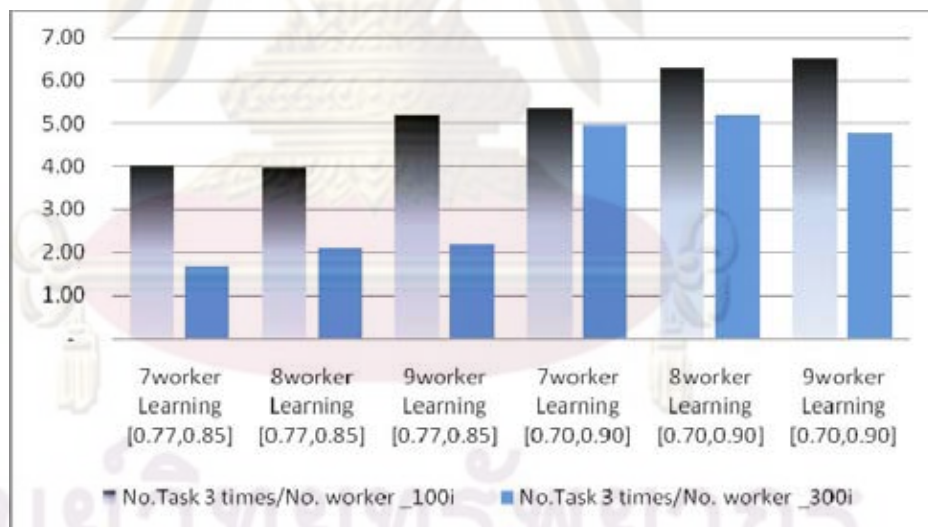


Figure 4.27 The percentage difference $diff_{con_heu}$ when No. Items increase(%)

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

4.6 Discussion of computation results

This chapter has proposed a problem of Assembly Line Worker Assignment and Balancing Problem with learning consideration. The problem which consists of the assembly line balancing problem and assignment problem with the objective of minimizing makespan was modeled. A heuristic was developed to solve the problem and the lower bound and upper bound were determined. For the sequence of trial makespan, the search starts from LB to UB. The heuristic groups the tasks were based on the value of the trial value. The groups of tasks were filtered and only the groups that were possible to give the feasible task-workstation assignment were kept. The worker-workstation assignment solution was determined based on the recurrence relation of the idle time among workstations in Section 4.3.1. The modified Dijkstra's algorithm was developed to determine the worker-workstation assignment.

The effectiveness of the heuristic was evaluated in terms of computational time and quality of the solution or makespan. It was found that the computational time of the proposed heuristic is significantly lower than the computational time from the exact solution. However, the computational time of the heuristic increases when the number of items and the number of workers increase. The heuristic found a solution within 0.80% from optimal solution on average.

To answer the question about the effect of using the constant skill level in situations accounting for worker learning ability, it was found that using the constant skill level in this situation is not appropriate since it has an effect on makespan. However, the constant skill level is suitable in situations where workers have slightly different learning slopes and a large production lot.

ศูนย์วิทยทรัพยากร

จุฬาลงกรณ์มหาวิทยาลัย

CHAPTER V

CONCLUSION AND FUTURE RESEARCH

5.1 Conclusion

5.1.1 Introduction

The appropriateness of using a two-stage heuristic for the practical application of worker assignment for workers with highly varying skill levels is addressed in this research. An integrated approach to the assembly line balancing problem and worker assignment problem was developed. The problem consists of a simultaneous solution to a double assignment: tasks to workstations and workers to workstations. This dissertation concerns both situations which are constant skill levels of workers and skill levels taking into account the learning ability of workers.

5.1.2 Problem description: the task-worker assignment problem assuming constant skill levels

The problem focuses on the final assembly line. The total line is considered to be serial with workstations consisting of one worker. Since the number of tasks is greater than the number of workers, a worker may be assigned more than one task. For the multiple assignments, consecutive tasks are only allowed since the tasks have precedence relation. There are i identical items which is processed in a number of tasks. After the worker has finished the tasks for processing an item, the item is sent to the next workstation along the line until it has passed through the last workstation. The processing time depends on the skill levels of workers who execute the tasks.

5.1.3 Mathematical model of the task-worker assignment problem assuming constant skill levels

A mathematical model is outlined in Chapter III. The model combines the original assembly line balancing problem and assignment problem. A variable a_{wjs} which represents that task j is assigned to worker w in workstation s which is designed to merge both problems. The main constraints were designed to cover the requirements of the system. To represent the group of consecutive tasks, the variable y_{js} was developed. The objective of the mathematical model is to minimize the maximum workstation time of cycle time.

5.1.4 Heuristic for the task-worker assignment problem assuming constant skill levels

For the proposed heuristic, starting with limiting search space by UB and LB, the heuristic uses the processing time of quickest worker of each task to determine LB. To set the UB close to the LB, the UB was developed based on the solution from the LB. The search space between UB and LB is split by the trial cycle time and the solution between the trial cycle time and LB is examined. No matter what solution is feasible or infeasible, the part of the search space separated by a trial value is discarded. Consequently, the search space is continuously reduced. The groups of tasks are generated based on the trial value, then assigned to the workers. Only the groups which have valid maximal station loads rule are considered. Tasks are added sequentially until the trial value is exceeded; that is, only the groups that have a cycle time within the trial value are considered. The maximal station loads rule is a classic rule for assembly line balancing problems. To determine the feasible assignment of the groups of tasks to workers, a mathematical model is modeled in the heuristic. The highlight of the model is that we added two parameters, b_{jg} and r_{wg} to indicate which tasks are contained in the group of tasks and which worker operates the group of tasks. Then a solver selects the groups of tasks on the list which are valid feasible assignments based on the information from both parameters.

5.1.5 Performance measurement of the heuristic assuming constant skill levels

The heuristic factors that have an effect on the quality of the solution, such as the position of the trial value between the UB and LB and stopping criteria of the search, were investigated. These parameters were tested in a small problem size, 8w24t. It was found that the position that is close to UB gives the best computation time. This means that the final solution is close to the upper bound and we can conclude that the heuristic has an efficient UB. The suitable stopping criterion of the heuristic which is limited by a gap between UB and LB is then determined.

Furthermore, the effect of the varying skill levels on the quality of the solution between the two-stage heuristic and the proposed heuristic was studied. Using the two-stage heuristic, firstly, the assembly line balancing problem is addressed by aggregating tasks using predetermined time standards and then with this established, the workers are assigned to the tasks. The percentage difference between the cycle time from the proposed heuristic and the cycle time from the two stages method is calculated. The result of the comparison confirms the disadvantage if the practical application is applied when the skills of workers vary greatly. The performance degrades to 10-15% when the deviation of skills of workers increases as reflected in a generation interval of [-30, 30] and the degradation is dramatic in the highest level of variance, increasing to 25-40%.

5.1.6 Problem description: the task-worker assignment problem taking into account learning ability

The second problem takes into account skill levels with learning consideration since it is the nature of fashion industry to launch new designs and new styles every season. For this reason, workers in the industry must continually learn about new fabrics and sewing processes. Based on the same problem, a mathematical model was developed. The objective of the problem is to minimize makespan. The objective of the conventional problem, which is minimizing the maximum workstation time, does not apply when learning is relevant since the bottleneck time

dynamically changes based on the reduction of learning ability for an assignment. Therefore, minimizing makespan or completion time was used as the objective in addressing this problem.

5.1.7 Mathematical model of the task-worker assignment problem taking into account learning ability

A mathematical model for the problem was developed as described in Chapter IV. However, this model is different from the mathematical model in Chapter III in regards to calculating the objective value, makespan, which is based on the completion time of the last item. The number of constraints in the problem is greater than in the problem in Chapter III due to the calculation of makespan, which depends on the number of items. Consequently, the computational time of the model is higher than the mathematical model proposed in Chapter III.

5.1.8 Heuristic for the task-worker assignment problem taking into account learning ability

A heuristic was developed to solve the problem of assigning tasks to workers factoring in learning consideration. The heuristic starts with limiting search space by UB and LB. We use the benefit of the solution from the heuristic of the problem using constant skill levels to determine UB and LB of the solution. The summation of the task processing time is represented as the constant skill level. The objective value or cycle time is the LB. It can be LB since it is obvious that the maximum workstation time is always less than the makespan value. The solution is evaluated using the makespan value as UB.

For the sequence of trial value between UB and LB, the search extends from LB to UB. The binary search could not be applied in the problem since the characteristics of both problems are different. A groups of tasks based on the trial value was generated. The groups of tasks were filtered and only the groups that were possible to give the feasible task-workstation assignment were kept. The assignment of the possible groups of tasks and workers was then determined

In the Dijkstra's algorithm, the shortest path from the starting node to every other node was determined in order to develop the shortest path to the ending node. In the heuristic, the shortest path to each node becomes the minimum idle time of that node which is designed to search for a solution minimizing makespan. If the minimum idle time of the last station is found, the minimum makespan would tend to be reached. The Dijkstra's algorithm was modified for the problem to fit the problem.

5.1.9 Performance measurement of the heuristic taking into account learning ability

The effectiveness of the heuristic was evaluated in terms of computational time and quality of the solution. It was found that the computational time of the proposed heuristic is significantly lower than the computational time of the exact solution. Furthermore, the proposed heuristic can keep the quality of the solution with a percentage deviation of less than 0.80 on average.

In addition, it was found that when the workers have different learning slopes, the assumption of a constant skill level in all situations is not appropriate since it has an effect on the quality of the solution. However, the constant skill level is suitable in situations where the workers have slightly different learning slopes and with the large production lots.

5.2 Discussion and recommendations

5.2.1 Task-worker assignment assuming constant skill levels

It was found that the quality of UB and LB was reduced when the workers have high variation in skill levels as seen in Section 3.4.2.1. For this reason, the computational time of the heuristic is higher when workers have high variation in skill levels. Consequently, we recommend resetting the UB in order to make the heuristic is not sensitive to high variation in skill levels.

There are three variables of assignment in the mathematical models of the both problems, which are a_{wjs} , y_{js} and r_{ws} . The three dimension variable a_{wjs} is the same assignment solution which is combined from y_{js} and r_{ws} . The mathematical model was developed based on background knowledge. Consequently, we recommend improving the model or resetting of precedence constraints.

Although the computational time of the heuristic is fast, when the number of tasks and workers increase, it tends to rapidly increase. It was found that the main computational time of the heuristic is from the assignment of the grouped tasks to the workers in which a solver is called to solve an assignment model. As a result, it was determined that the heuristic could be improved.

This heuristic may have a limitation in application since the problem is designed for tasks that are set in series. Nevertheless, the framework of this heuristic can be applied.

Since the model assumes that all workers are fully cross-trained, it is further assumed that they can be assigned to any task on a line. However in practice, a worker may have skills to work on only certain tasks. To solve this problem, the assumption can be relaxed in our model by assigning an infinite cost to any worker-task combination. Consequently, the number of the alternatives of grouped tasks is reduced, as is the problem size of the assignment groups of tasks to workers in Section 3.3. The computational time can be reduced when workers have skills for only certain tasks.

Since the heuristic generates the groups of tasks based on the trial value, the optimal solution is a solution in which all workers work as close to the final cycle time as possible. However in fact, there may be the other solutions that give the same cycle time which are not included in this heuristic.

Comparing the two stage assignments, which are the assembly line balancing problem and worker assignment problem, to the proposed problem; it was found that the proposed problem has more complexity since it addresses the two problems together. We recommend that the supervisor should consider the variation

of skills of the workers before making an assignment. The proposed problem is recommended to account for the high variation of skills of workers.

5.2.2 Task-worker assignment taking into account learning ability

When more than one task is assigned to a worker, the processing time of the combined tasks is determined by the sum of the processing time of the tasks that he/she performs. This assumption may not be suitable if the combined tasks are similar since there may be a learning factor as a result of the task similarity.

In addition, the computational time heuristic is sensitive when the number of items and the number of workers increase. The main computational time is obtained from the step generating the feasible path of task-workstation assignment. This part should be further refined.

5.3 Future research

There are several interesting possible extensions to the present work. A future research direction could be to consider a more general case. It can be developed as an integrated approach to assembly line balancing and worker assignment in different conditions in industrial manufacturing. The problem can be modified following generalized assembly line balancing. For example, it may be a system for several products or different models, or include a different line layout. For example, it could include parallel workstations or a group of tasks which can be operated by more than one worker or a system which has assignment restrictions e.g. some tasks have to be assigned to the same workstation or some tasks are incompatible and have to be assigned to a different station. Furthermore, industrial manufacturing may require more than a one-objective problem, so the objective of the problem can be considered in the form of a multi-objective problem.

Also regarding further solution development, a randomized-based heuristic can be added to improve the quality of the solution. A local search procedure or meta-heuristic may be more efficient for problems of a larger size.



ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

REFERENCES

- Altıparmak, F., Dengiz, B. and Bulgak, A.A. 2007. Buffer allocation and performance modeling in asynchronous assembly system operations: An artificial neural network metamodeling approach. **Applied Soft Computing**, 7: 946-956.
- Askin, R. G. and Chen, J. 2006. Dynamic task assignment for throughput maximization with worksharing. **European Journal of Operational research** 168: 853-869.
- Askin, R. G. and Standridge, C. R. 1993. **Modeling and Analysis of Manufacturing Systems**, 1st Ed. New York: John Wiley & Sons.
- Baybars, I. 1986. A Survey of Exact Algorithms for the Simple Assembly Line Balancing Problem. **Management Science**, 32
- Bellman, R., Esogbue, A.O. and Nabeshima, I. 1982. **Mathematical aspects of scheduling & Applications**. 1st ed.: Pergamon Press, 1982.
- Betts, J. and Mahmoud, KI. 1992. Assembly line balancing in the clothing industry allowing for varying skills of operatives. **International Journal of clothing Science and Technology** 4(4): 28-34.
- Bokhorst, A.C., Slomp, J. and Molleman, E. 2004. Development and evaluation of cross-training policies for manufacturing teams. **IIE Transactions** 36: 969-984.
- Brusco, M.J. and Johns, T.R. 1998. Staffing a multi-skilled workforce with varying levels of productivity: an analysis of cross-training policies. **Decision Sciences** 29: 499-515.
- Burkard, R.E. 2002. Selected topics on assignment problems. **Discrete Applied Mathematics** 123: 257-302.
- Caron, G., Hansen, P. and Jaumard 1999. The assignment problem with seniority and job priority constraints. **Operations Research** 47(3): 449-493.
- 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.

- Chakravarty, A. K. 1988. Line Balancing with Task Learning Effects **IIE Transactions** 20(2): 186-193.
- Chan, K.C.C. 1997. Handling the assembly line balancing problem in the clothing industry using a genetic algorithm. **International Journal of clothing Science and Technology** 10(1): 21-37.
- Chaves, A.A., Lorena, A.A.N. and Miralles, C. 2009. Hybrid Metaheuristic for the Assembly Line Worker Assignment and Balancing Problem. **Lecture Notes in Computer Science: Hybrid Metaheuristic**, Springer : Berlin, Heidelberg.
- Chen, C. , Racine, R. and Swift, F. 1992. A practical approach to the apparel production-planning and scheduling problem. **International Journal of clothing Science and Technology** 4(2): 9-18.
- Chen, J.C. , Hsaio, M.H., Chun-Chieh, C. and Cheng-Ju, S. 2009. A grouping genetic algorithm for the assembly line balancing problem of sewing lines in garment industry. **The eight International Conference on Machine Learning and Cybernetics**, China.
- Cohen, Y. , Vitner, G. and Sarin, S. 2007. Work allocation of work in assembly lines for lots with homogenous learning. **European Journal of Operational Research** 108: 922-931.
- Cohen, Y., Vitner, G. and Sarin, S. 2008. Work allocation to stations with varying learning slopes and without buffers. **European Journal of Operational Research** 184(2): 797-801.
- Corominas, A., Pastor, R., and Plans, J. 2008. Balancing assembly line with skilled and unskilled workers. **Omega** 36(6): 1126-1132.
- Corominas, A., Pastor, R. and Rodriguez, E. 2006. Rotational allocation of tasks to multifunctional workers in a service industry. **International Journal Production Economics** 103: 3-9.
- Dar-El, E.M. 2000. **Human Learning: From Learning Curves to Learning Organizations**, Kluwer: Boston, MA.
- Dar-El, E.M., Cohen, Y. 1998. Optimizing the number of stations in assembly lines under learning for limited production. **Production Planning & Control** 9(3): 230-240.
- Dell'Amico, M. and Martello, S. 1997. The k -cardinality assignment problem. **Discrete Applied Mathematics** 76: 103-131.

- Duin, C.W. and Volgenant, A. 1991. Minimum deviation and balanced optimization: A unified approach. **Operations Research Letters** 10(1): 43-48.
- Eitzen, G. and Panton, D. 2004. Multi-skilled Workforce Optimization. **Annals of Operations Research** 127: 359-372.
- Farrar, T.M. 1993. Optimal use of an extra server in a two station tandem queuing network. **IEEE Transactions on Automatic Control** 38: 1296-1299.
- Fatih Ugurdag, H., Rachamadugu, R., and Papachristou, C. A., 1997. Designing paced assembly lines with fixed number of stations. **European Journal of Operational Research** 102(3) pp: 488-501.
- Franz, L.S. and Miller, J.L. 1993. Scheduling medical residents to rotations:solving the large-scale multiperiod staff assignment problem. **Operation Research** 68(3): 269-279.
- Futatsuishi, Y., Watanabe, I and Nakanishi, T. 2002. A study of the multi-stage flow shop scheduling problem with alternative operation assignments. **Mathematics and Computers in Simulation** 59: 73-79.
- Gavett, J.W. 1968 **Production and operations management**, Harcourt, Brace & world Inc.
- Gavish, B., Pirkul, H. 1991. Algorithms for the multi-resource generalized assignment problem. **Management Science** 36(7): 583-590.
- Hassamontr, J 2004. Assembly Line Balancing with Operator's Skill and Machine Constraints. **Hawaii International Conference on Business (HIC 2004)**, June 20-24, 2004, USA.
- Hopp, W.J. and Van Oyen, M.P. 2004. Agile workforce evaluation: a framework for cross-training and coordination. **IIE Transactions** 36: 919-940.
- Hui, P.C.L. and Ng, S.F. 1999. A study of the effect of time variation for assembly line balancing in the clothing industry. **International Journal of clothing Science and Technology** 11(4): 181-188.
- Hui, P.C.L., Chan, C.C., Yeung, K.W. and Ng, S.F. 2002. Fuzzy operator allocation for balance control of assembly lines in apparel manufacturing. **IEEE Transaction on Engineering Management** 49(2): 173-180.
- Inman, R.R., Blumenfeld, D.E.and Ko, A. 2005. Cross-training Hospital Nurses to Reduce Staffing Costs. **Health Care Management Rev** 30(2): 116-125.

- Jordan, W.C. and Graves, S.C. 1995. Principles on the benefits of manufacturing process flexibility. **Management Science** 41: 577-594.
- Kan, M.R. 1999. Simulation modeling of a garment production system using a spreadsheet to minimize production cost. **International Journal of clothing Science and Technology** 11(5): 287-299.
- Karni, R. and Herer, Y. T. 1995. Allocation of tasks to stations in small-batch assembly with learning: basic concepts. **International Journal of Production Research** 33(11): 2973-2998.
- Kouvelis, P. and Yu, G. 1997. **Robust Discrete Optimization and Its Applications** Kluwer Academic Publishers, Dordrecht, The Netherlands.
- Leopairote, K. 2003. **Policies for multi-skilled worker selection assignment and scheduling** Dissertation, Industrial Engineering, University of Wisconsin-madison.
- Lev, B. and Weiss, H. 1982. **Introduction to Mathematical Programming Quantitative Tools for decision Making** 1st ed. New York:Elsevier North Holland,Inc.
- Liu, S. B., Ong, H. L., and Huang, H. C., 2003. Two bi-directional heuristics for the assembly line type II problem. **The International Journal of Advanced Manufacturing Technology** 9(9): 656-661.
- Martello, S., Pulleyblank, W.R., Toth, P and de Werra, D. 1984. Balanced optimization problems. **Operations Research Letters** 3 (5): 275–278.
- Masaru, N., Sei, U., Yochito, M., Katsuyuki, S.and Yukihiro, A. 1981. Line balancing of sewing systems (a simple calculating method to determine the number of workstations and cycle time). **Journal of the Textile Machinery Society of Japan** 27(2):57-62.
- Miralles, C., Garcia-Sabater, J. P., Andres, C. & Cardos, M. 2008. Branch and bound procedures for solving the Assembly Line Worker Assignment and Balancing Problem: Application to Sheltered Work centres for Disable. **Discrete Applied Mathematics** 156(3): 352-367.
- Mosheiov, G. 2001. Scheduling problems with a learning effect. **European Journal of Operational Research** 132(3): 687-693.

- Nakade, K. and Ohno 1999. An optimal worker allocation problem for a U-shaped production line. **International Journal Production Economics** 60-61: 353-358.
- Nembhard, D.A. 2000. The effects of task complexity and experience on learning and forgetting: A Field study. **Human factors** 42(2): 272-286.
- Nembhard, D.A. 2001. Heuristic approach for assigning workers to tasks based on individual learning rates. **International Journal of Production Research** 39(9): 1955-1968.
- Nembhard, D.A. and Osothsilp, N. 2005. Learning and forgetting-based worker selection for tasks of varying complexity. **Journal of the Operational Research Society** 56(2): 576-587.
- Nemhard, D.A. and Uzumeri, M V. 2000A. An individual-based description of learning within an organization. **IEEE Transactions on Engineering Management** 47(3): 370-378.
- Nemhard, D.A. and Uzumeri, M V. 2000B. Experiential learning and forgetting for manual and cognitive tasks. **International Journal of Industrial Ergonomics** 25(4),315-326.
- Norman, B.A., Tharmmaphornphilas, W, Needy, K.L., Bidanda, B. and Warner, R.C. 2002. Worker assignment in cellular manufacturing considering technical and human skills. **International journal production research** (40)6:1479-1492.
- Pastor, R. and Ferrer, L. 2008. An improved mathematical program to solve the simple assembly line balancing problem. **International Journal of Production Research** 47(11): 2943-2959.
- Pentico, D.W. 2007. Assignment problems: A golden anniversary survey. **European Journal of Operational Research** Volume 176(2): 774-793.
- Prins, C. 1994. An overview of scheduling problems arising in satellite communications. **Journal of the Operational Research Society** 45(6): 611-623.
- Scholl, A. 1999. **Balancing and sequencing of assembly lines** 2nd ed., Physica-Verlag, Germany.
- 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(3): 666-693.

- Scholl, A., Fliedner, M., and Boysen, N., 2010. Absalom: Balancing assembly lines with assignment restrictions. **European Journal of Operational Research** 200: 688-701.
- Scholl, A. and Klein, R. 1996. Maximizing the production rate in simple assembly line balancing - A branch and bound procedure. **European Journal of Operational Research** 91(3): 367-385.
- Sayin, S. and Karabati, S. 2007. Assigning cross-trained workers to departments: A two-stage optimization model to maximize utility and skill improvement. **European Journal of operational research** 176:1643-1658.
- Skiena, S.S. 2008. **The Algorithm Design Manual**. Springer: Newyork, USA.
- Slomp, J., Bokhorst, J.A.C. and Molleman, E. 2005. Cross-training in a cellular manufacturing environment. **Computers & Industrial Engineering** 48: 609-624.
- Slomp, J., Molleman, E. 2002. Cross-training policies and team performance. **International Journal production research** 50(5): 1193-1219.
- 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. **Computer & Industrial Engineering** 51(4): 585-608.
- Spragg, J.E., Fozzard, G. and Tyler, D.L. 1999. FLEAS:A flow line environment of automated supervision. **International Journal of clothing Science and Technology** 10(6): 322-327.
- 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.
- Suer, G.A. 1996 Optimal operator and cell loading in labor-intensive manufacturing cells. **Computer industrial Engineering** 31(1/2): 155-158.
- Suer, G.A. and Bera, I.S. 1998 Optimal operator assignment and cell loading when lot-splitting is allowed, **Computer industrial Engineering** 35(3-5): 431-434.
- Talbot, F.B. and Patterson, J.H. 1984. An integer programming algorithm with network cuts for solving the assembly line balancing problem. **Management Science** 30(1): 85-99.

- Tasan, S., and Tunali, S. 2008. A review of the current applications of genetic algorithms in assembly line balancing. **Journal of Intelligent Manufacturing** 19(1): 49-69.
- Toksari, M. D. and Isleyen, S. K. 2008. Simple and U-type assembly line balancing problems with a learning effect. **Applied Mathematical Modeling** 32(12): 2954-2961.
- Tomastik, R.N., Luh, P.B. and Liu G. 1996. Scheduling Flexible Manufacturing System for Apparel Production. **IEEE Transaction on Robotics and Automation** 12(5): 789-799.
- Treleven, M. 1989. A review of dual resource constrained systems research. **IIE Transactions** 21: 279-287.
- Uzumeri, M and Nembhard D.A. 1998. A population of learning: a new way to measure organizational learning. **Journal of Operations Management** 16: 515-528.
- Wild, R. 1972. **Mass-production management; the design and operation of production flow-line systems**, John Wiley & Sons.
- Wong, W.K., Chan C.K. and Ip, W.H. 2001. A hybrid flowshop scheduling model for apparel manufacture. **International Journal of clothing Science and Technology** 13(2):115-131.
- Wong, W.K., Mok, P.Y. and Leung, S.Y.S. 2005. Developing a Genetic Optimisation Approach to Balance an Apparel Assembly Line. **International Journal Advanced Manufacturing Technology** 28: 387-394.
- Wright, T.R. 1936. Factors affecting the cost of airplanes. **Journal of Aeronautical Sciences** 3(4): 122-128.
- Yang, K.K. 2007. A comparison of cross-training policies in different job shops. **International Journal of Production Research** 45(6): 1279-1295.
- 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** 20(3): 283-293.
- Zhang W., Lin L., and Gen M., 2008. A Multiobjective Genetic Algorithm *based Approach to Assembly Line Balancing Problem with Worker Allocation*. **International Conference on Systems, Man and Cybernetics (SMC)** Singapore.

Zulch, G., Rottinger, S. 2004. A simulation approach for planning and re-assigning of personnel in manufacturing. **International Journal of Production Economics** 90(2): 265-277.



ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

VITA

Miss Kanjana Thongsanit was born on April 10th, 1976 in Phitsanuloke province, Thailand. She graduated from Chiang Mai University, Thailand in academic year 1998 with a bachelor's degree in Industrial Engineering. She earned a Master's degree in Industrial Management Engineering from King Mongkut's University of Technology North Bangkok, Thailand in academic year 2001. After her graduation, she decided to work as a quality management consultant at Thai Garment Development Foundation (TGDF) from 2001-2002. In 2003, she changed her career to be a lecturer at Industrial Engineering and Management department, Silpakorn University (Sanamchan Campus), Nakornpatom. She got a scholarship from Thai government to study a Ph.D. in year 2004 in Industrial Engineering at Chulalongkorn University, Thailand.



ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย