รายงานวิจัยฉบับสมบูรณ์

โครงการ การจัดลำดับการผลิตของอุตสาหกรรมต่อเนื่องแบบผสม
เมื่อเครื่องจักรในแต่ละสถานีงานมีประสิทธิภาพการทำงาน
ไม่เท่ากัน

โดย

ผศ.ดร.กาญจนา เศรษฐนันท์

พฤษภาคม 2548

สัญญาเลขที่ TRG 4580020

รายงานวิจัยฉบับสมบูรณ์

โครงการ การจัดลำดับการผลิตของอุตสาหกรรมต่อเนื่องแบบผสมเมื่อเครื่องจักรในแต่ละสถานีงานมีประสิทธิภาพการทำงานไม่เท่ากัน

โดย

ผศ.ดร.กาญจนา เศรษฐนันท์

ภาควิชาวิศวกรรมอุตสาหกรรม

คณะวิศวกรรมศาสตร์

มหาวิทยาลัยขอนแก่น

สนับสนุนโดยสำนักงานกองทุนสนับสนุนการวิจัย

กิตติกรรมประกาศ

การวิจัยครั้งนี้ได้สำเร็จลุล่วงไปด้วยดี ผู้วิจัยต้องขอขอบคุณสำนักงานกองทุนสนับสนุนการวิจัย (สกว.) ที่ได้ให้โอกาสและให้ทุนการวิจัยเพื่อเป็นกาสนับสนุนการวิจัยให้แก่นักวิจัยรุ่นใหม่ได้มีโอกาสทำงานวิจัยและพัฒนาตัวเองอย่างต่อเนื่อง และสามารถที่จะผลิตผลงานตีพิมพ์ในวารสารวิชาการระดับนานาชาติได้

พร้อมกันนี้ผู้วิจัยขอกราบขอบพระคุณ รศ.ดร.สมเกียรติ รุจิเกียติกำจร หัวหน้าภาควิชาวิศวกรรมอุตสาหการ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น นักวิจัยพี่เลี้ยงที่ได้กรุณาเสียสละเวลาให้คำปรึกษา ชี้แนะแนวทางและความรู้ต่างๆ ตลอดเวลาที่ผู้วิจัยได้รับทุน นอกจากนี้ผู้วิจัยใคร่ขอขอบคุณเจ้าหน้าที่ของ สกว. ทุกท่านที่ได้อำนวยความสะดวกและให้ความช่วยเหลือต่างๆ ด้วยดีตลอดมา

สุดท้ายผู้วิจัยขอขอบพระคุณคุณแม่และคุณพ่อผู้อันเป็นที่รัก น้องสาว น้องชาย และตัวเองที่ได้ให้ความรัก และความอบอุ่น ตลอดจนกำลังใจที่ทำให้ฝ่าฝันอุปสรรคต่างๆ ได้เป็นอย่างดี

ผศ.ดร.กาญจนา เศรษฐนันท์

พฤษภาคม 2548

บทคัดย่อ

รหัสโครงการ : TRG 4580020

ชื่อโครงการ : การจัดลำดับการผลิตของอุตสาหกรรมต่อเนื่องแบบผสม เมื่อเครื่องจักรในแต่ละสถานีงานมีประสิทธิภาพการทำงานไม่เท่ากัน

ชื่อนักวิจัยและสถาบัน : ผศ.ดร.กาญจนา เศรษฐนันท์
ภาควิชาวิศวกรรมอุตสาหกรรม
คณะวิศวกรรมศาสตร์
มหาวิทยาลัยขอนแก่น

E-mail Address : skanch@kku.ac.th

ระยะเวลาโครงการ : 1 กรกฎาคม 2545 - เมษายน 2547

ในงานวิจัยนี้ ได้ศึกษาเกี่ยวกับการจัดลำดับการผลิตในรายการผลิตต่อเนื่องแบบผสม โดยมีการพิจารณาเวลาเตรียมงานที่ไม่อิสระในสายการผลิตต่อเนื่องแบบผสมจะประกอบด้วยสถานีงานทั้งหมด L สถานีงาน โดยแต่ละสถานีงานประกอบด้วยเครื่องจักรอย่างน้อย 1 เครื่องที่มีประสิทธิภาพไม่เท่ากันแบบไม่สัมพันธ์กัน (unrelated machines) วัตถุประสงค์ของงานวิจัยนี้ เพื่อทำให้เวลาแล้วเสร็จของการผลิตมีค่าน้อยที่สุด ในการวิจัยครั้งนี้รูปแบบทางคณิตศาสตร์ได้ถูกสร้างขึ้นเพื่อหาคำตอบที่ดีที่สุดสำหรับปัญหาขนาดเล็กที่ไม่ซับซ้อนและวิธีทางฮิวริสติค 2 ฮิวริสติค (IH และ TSearch) ซึ่งได้ถูกพัฒนาขึ้นเพื่อใช้ในการแก้ปัญหาขนาดใหญ่ที่เป็นจริงสำหรับอุตสาหกรรมโดยทั่วไป โดย IH เป็นฮิวริสติคที่ให้ค่าคำตอบเริ่มแรก (initial Solution) ที่ค่อนข้างดีและคำตอบที่ได้จาก IH ฮิวริสติคนี้จะถูกนำไปปรับปรุงให้ดีขึ้นโดย TSearch ฮิวริสติค โดย TSearch ฮิวริสติคนี้จะทาบูเสริช เมตะฮิวริสติค (Tabu Search Metaheuristic) ในการประเมินประสิทธิภาพของ 2 ฮิวริสติคนั้นจะใช้ Lower bounds (Forward LB และ Backward LB) ที่ได้พัฒนาขึ้นในการคำนวณ Lower bounds นั้นจะประกอบด้วย 4 ปริมาณคือ (1) เวลารอของเครื่องจักรที่สถานีงานที่ L (2) เวลารอของเครื่องจักรที่สถานีงานสุดท้าย (3) เวลาการผลิตทั้งหมด และ (4) เวลาเตรียมเครื่องจักร ดังนั้นในการประเมินประสิทธิภาพของฮิวริสติคที่พัฒนาขึ้นนั้นจะมีการวัดใน 2 ปริมาณคือ (1) สมรรถนะของฮิวริสติคโดยเปรียบเทียบกับ Lower bounds และ (2) เปอร์เซ็นต์การปรับปรุงของ TSearch ฮิวริสติคจากการใช้คำตอบจาก IH ฮิวริสติค จากการทดลองพบว่า ฮิวริสติกที่พัฒนาขึ้นมีประสิทธิภาพค่อนข้างดี และเปอร์เซ็นต์การปรับปรุงประสิทธิภาพจากการใช้ TSearch ฮิวริสติคจะอยู่ระหว่าง 12.2% - 25.5%

คำหลัก : Heuristics, Hybrid Flowshop, flexible Flowshop, Unrelated Muchines, Dependent Setup Times.

ABSTRACT

Project Code: TRG 45800200

Project Title: Heuristics for Scheduling Flexible Flowshops when Machines are Unrelated.

Investigator: Asst. Prof. Kanchana Sethanan, Ph.D.

E-mail Address: skanch@kku.ac.th

Project Period: July 1, 2002 – Apr 1, 2004

This research addresses the scheduling problem in a hybrid flowshop when machines in each stage are unrelated and sequence dependent setup times are considered. The production line consists of L production stages, each of which may have more than one non-identical (unrelated) machines. Prior to processing a job on a machine at the first stage, a setup time from idling is required. Also, sequence dependent setup times are considered on each machine in each stage. The objective of this research is to minimize the maximum makespan. Two mathematical models were formulated for small size problems and two heuristic algorithms (IH and TSearch) were developed to solve larger, more practical problems. In order to evaluate the Performance of the heuristic, normally, the heuristic solutions are compared to optimal solutions and/or lower bounds. The hybrid flowshop when machines in each stage are unrelated and sequence dependent setup times are considered is known to be NP-hard, and hence finding and optimal solution for average or large-size problems will be computationally intractable. The only alternative left is to develop lower bounds for the problem and use them to assess the quality of the heuristic solutions. Therefore, in this study, two lower bounds (Forward and Backward) were developed in order to evaluate the performance of the heuristics. Results obtained show that the heuristic algorithms are quite efficient. The relative improvement yielded by the TSearch algorithm was between 12.2 and 25.5 percent.

Keywords: Heuristics, Hybrid Flowshop, flexible Flowshop, Unrelated Machines, Dependent Setup Times.

TABLE OF CONTENTS

TABLE OF CONTENTS

Scheduling Flexible Flowshops with Sequence Dependent Setup Times and Machines in Each Stage Are Unrelated

CHAPTER 1
INTRODUCTION

Nowadays, manufacturers are faced with customer demands for a variety of high quality products. The companies must therefore make their production systems more flexible, respond rapidly to demand fluctuations, and reduce costs related to production. Hence, companies need to have advanced techniques. Manufacturing has been an interesting topic in production and operation management because of areas such as job scheduling or machine loading.

Scheduling problems arise whenever a set of resources such as workers or machines are required to perform a set of operations on jobs, also each operation can be accomplished in more than one way. Given a limited set of resources, the scheduling problem is to assign jobs to resources according to some process routing in order to obtain optimal performance measures while ensuring that all production constraints are satisfied. The development of production schedules is a remarkably important task in industry especially scheduling jobs through non-identical, parallel processors. Non-identical processors are processors that do not have equal capabilities and capacities. This type of production system where multiple products are made on parallel, non-identical production line is common in both service and manufacturing industries. For instance, workers in an office have different skills, an airline assigns a type of airplane to service a route, or paper plant assigns products to different paper machines. A parallel processing is the situation where a job can be done by more than one processor but only one processor can actually work on the job (Randhawa & Smith, 1995). Hence, scheduling problems involve the assignment of machines to various jobs and determination of the order in which the jobs will be performed in order to optimize some criteria (such as minimization of makespan, mean flow time, or lateness) while satisfying the shop constraints.

1.1 A Hybrid Flowshop Environment

In real industries, a hybrid flowshop is more commonly seen than traditional flowshop. A hybrid flowshop is a generalization of the flowshop and the parallel processor environments. A hybrid flowshop is alternatively called a flexible flowshop (FFs). In a hybrid flowshop environment, there are L workstations, each of which consists of at least one machine. The machines in each stage may identical, uniform, or unrelated. In a hybrid flowshop, each job is processed first at stage 1, then

at stage 2, and so on. Normally, a job requires only one machine in each stage and any machine can process any job as shown in the schematic representation in Figure 1.

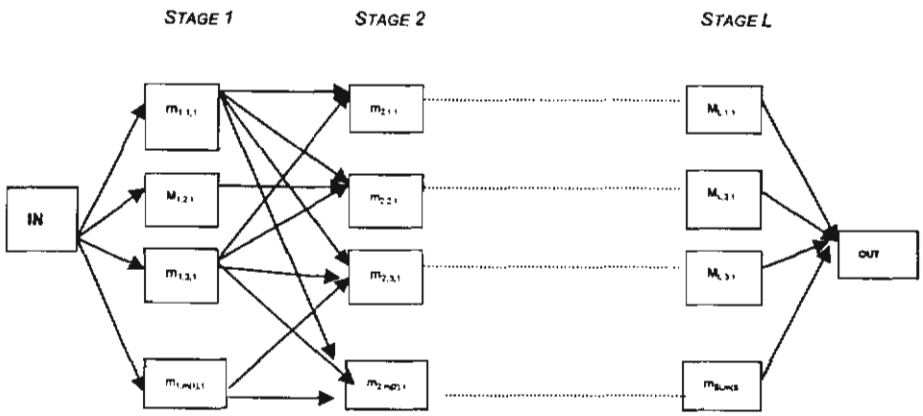

Figure 1: A Schematic representation of a Flexible Flowshop Manufacturing Environment

1.2 Dependent Setup Time

The requirements of setup times of jobs are very common in many real manufacturing situations such as inspecting material, setting tools, and cleanup. There are two types of setup times: sequence independent and sequence dependent setup times. Sequence dependent setup times are considered to be very important factors in the manufacturing environment, especially, when a shop floor is operated at or near its full capacity (Wilbrecht & Prescott, 1969). Sequence dependent setups occur especially in process industry operations, where machine setup time is significant and is needed when products change. The magnitude of setup time depends on the similarity in technological processing requirements for successive jobs (Srikan & Ghosh, 1986). Normally, similar technological requirements for two consecutive jobs would require lesser setup.

Even though there exists an enormous amount of research on the flowshop scheduling problem, research study has rarely been conducted in the case where setup times are sequence dependent (Allahverdi et al., 1995). Hence, the results of these research studies lack a practical solution for applications that require the treatment of setup times. For this reason, dependent setup times must be allowed for a realistic description and hence are considered in this research. The paper is organized as follows. In the next section, the literature review is presented. In section 3, the problem and propose is described, then two optimizing algorithms are developed in section 4. The heuristic algorithms are described in Section 5. In section 6, the lower bounds for the problem are determined. Section 7 concludes the paper with a discussion of this research and future extension.

CHAPTER 2

LITERATURE REVIEW

The problem of scheduling *n* jobs on m machines is one of the classical problems in flowshop manufacturing that has been interested researchers for many years. According to Gupta (1994), a heuristic was developed to solve a special case when there is only one machine in the second in order to minimize the makespan. Computational experiments show that the effectiveness of the proposed heuristic increases as the problem-size increases. Brah and Hunsucker (1991) develop branch and bound algorithms for the multiple stage hybrid flowshop (Pm_1, Pm_2,...,Pm_s)//C_{max}). The computational results show that the algorithms can solve only small-sized problems. Portmann et al. (1998) improved the lower bound of Brah's and reduced the number of branches used in the search tree. They could solve the problems with up to five stages and fifteen jobs.

For the dependent setup time problem, Ruiz et al. (2005) presented two advanced genetic algorithms as well as several adaptations of existing advanced metaheuristics that have shown superior performance when applied to a regular flowshop with sequence dependent setup times. In the same year, Ruiz and Maroto developed a metaheuiristic, in the form of genetic algorithm, for hybrid flowshops with sequence dependent setup times and machines eligibility. The results indicate that the proposed algorithm is more effective than all other adaptations. Tahar et al. (2005), developed a linear programming model and a heuristic algorithm for identical parallel machine scheduling with job splitting and sequence-dependent setup times in order to minimize the makespan. According to Kurz and Askin (2005) developed an integer programming model based on the TSP for the flexible flow lines with sequence dependent setup times. Several heuristics such as Insertion heuristic and Random keys Genetic Algorithm also developed.

In addition, Sethanan (2001), the mixed integer programming is formulated to solve the flexible flowshops with sequence dependent setup times when machines in each stages are uniform (FFs(Qm_1,Qm_2,...,Qm_S)/S_{ipm}/ C_{max}). Since the FFs(Qm_1,Qm_2,...,Qm_S)/S_{ipm}/ C_{max} problem is known to be NP-hard (Allahverdi, 1999) and hence finding an optimal solution for average or large-size problems will be computationally intractable. Therefore, in her study, the algorithm and two lower bounds were developed to solve the problem.

From the reviewed literature, there is no literature in scheduling multiple hybrid flowshop lines with sequence dependent times when machines in each stage are unrelated. Hence, this research focuses on scheduling multiple hybrid flowshop lines when machines are unrelated and sequence dependent times also are considered.

CHAPTER 3

THE PROBLEM DESCRIPTION

This research involves scheduling multiple products through non-identical parallel production lines. There are many production stages in each production line. Each stage may comprise more than one machine. Resource and technological constraints are considered in this production system. Resource constraints generally refer to processor capacities and limitations. Technological constraints are considered as product routing and precedence constraints. In this study, all products can be manufactured on every machine in a stage, and the machine cannot process a new product until the previous product has been completely finished.

Prior to processing a job on a machine, there is an associated setup time. Thus, setup times are considered significant and typically depend on the sequence of the jobs through the processors. Setup times in general are large when compared to the unit processing time. As much of the current industrial competition is a time-based, the reduction of the production lead time is an important key. Hence, the objective of this research is to minimize the maximum completion time of the products called the makespan. Two mathematical models are formulated to solve the problem and to produce an optimal schedule in order to minimize the total makespan.

3.1 Description of the model

The assumption made in formulating the model are followed:

1. It is assumed that the decisions have been made from the long and intermediate-range planning.
2. Production is make-for-stock; hence, there are no due dates associated with products.
3. All jobs and machines are available at the beginning of the scheduling process.
4. In the production line, there are many stages. Each stage of the hybrid flowshop production may have several non-identical machines.
5. Jobs can wait between two production stages and the intermediate storage is unlimited.
6. Setup times for jobs on each machine are dependent on the order in which jobs are processed.
7. Products cannot be split between machines in the same stage.
8. There is no job preemption.

CHAPTER 4

OPTIMIZING ALGORITHMS

A brief description of the problem is reviewed in order to help in understanding the mathematical formulation. In this research, there is only one production line considered. The problem involves the scheduling of multiple products in a flexible flowshop environment with sequence dependent setup times ($FFS(R_{m1},R_{m2},\ldots, R_{mS})/s_{ipm}/C_{max}$). The production line consists of many stages, which may have one or more non-identical (unrelated) parallel machines. In each stage, machines can process all products but differ in their performances and the machines cannot process a new product until the previous product has been completely finished.

The products have to be manufactured on only one of the machines in each stage, and the processing of products cannot start until the entire batch is completed in the previous stage. Each product, e.g., product *i* requires P(i,s,m) units processing time on machine *m* of stage *s*. Machine setup times are needed between any two products. In this study, it is assumed that setup times are equal for every machine in the same stage when changing from one product to another.

This section presents an optimal algorithm for the FFS with uniform machines at each stage. A 0-1 mixed integer programming model is developed with the criterion to minimize makespan for this problem. Parameters and decision variables used in formulating the model are defined as presented in Table 1. The 0-1 mixed integer programming formulation is presented below with a brief explanation of each constraint.

Table 4.1: The Notation Used in the Mixed Integer Programming Model

Type of Variables	**Notation**	**Explanation**
Decision Variables	F(i,s,m)	Finish time of product i on machine m of stage s
	E	The makespan
Binary decision variables	x(i,s,m)	= 1 , if product i is assigned to machine m of stage s = 0 , otherwise
	w(i,p,s,m)	= 1 , if product i immediately precedes product p on machine m of stage s = 0 , otherwise
	w(i,0,s,m)	= 1 , if product i is the last product processed on machine m of stage s = 0 , otherwise
	w(0,i,s,m)	= 1 , if product i is the first product processed on machine m of stage s = 0 , otherwise
Parameters	i,p,h	Product indices
	s	Stage index

	m(s)	The number of machines in stage s
	n	Total number of products
	M(s)	The set of machines in stage s ; M(s) = {1,2,..,m(s)}
	L	The number of stages in the production line
	P(i,s,m)	The processing time of product i on machine m of stage s
	ch(i,p,s)	The number of time units required to changeover from production i to product p at stage s
	V	a very large positive number.

Model 1:

The objective function: Min E

Constraints:

$$F(i,1,m) \geq ch(0,i,s) + \{P(i,1,m) \cdot x(i,1,m)\} \quad (1)$$

i = 1,2,...,n ; and m=1,2,...,m(1)

$$F(i,s,m) \geq F(i,s-1,mp) + \{P(j,i,s,m) \cdot x(j,i,s,m)\} \quad (2)$$

i = 1,2,...,n ; s = 2,3,...,L, m = 1,2,...,m(s), and mp = 1,2,...,m(s-1)

$$F(i,s,m)-F(p,s,m)-ch(p,i,s)+(V)(1-w(p,i,s,m)) \geq \{P(i,s,m) \cdot x(i,s,m)\} \quad (3)$$

i = 1,2,...,n, s = 1,2,..., L, m = 1,2,...,m(s)

$$F(i,L,m) \leq E \quad (4)$$

i = 1,2,...,n ; and m = 1,2,...,m(L)

$$\sum_{m=1}^{m(s)} x(i,s,m) = 1 \quad (5)$$

i = 1,2,..., n ; and s = 1,2,..., L

$$x(p,s,m) - w(0,p,s,m) - \sum_{p=1}^{n} w(i,p,s,m) = 0 \quad (6)$$

p = 1,2,..., n; s = 1,2,...,S; and m = 1,2,...,m(s)

$$x(i,s,m) - w(i,0,s,m) - \sum_{i=1}^{n} w(i,p,s,m) = 0 \quad (7)$$

i = 1,2,...,n ; s = 1,2,..., L; and m = 1,2,...,m(s)

$$\sum_{p=1}^{n} w(0,p,s,m) = 1 \quad (8)$$

s = 1,2,..., L; and m = 1,2,...,m(s)

$$\sum_{p=0}^{n} w(p,0,s,m) = 1 \qquad (9)$$

s = 1,2,..., L; and m = 1,2,...,m(s)

Constraints (1) is a completion time forcing constraints. It ensures that all products are scheduled and the completion time of any product on any machine of the first stage is at least the amount of processing time required for the product on that machine. Constraints (2) ensures that the completion time of product i produced on machine m in the current stage (stage s) must be greater than its completion time in a previous stage (stage s-1). Constraints (3) is about product sequencing on all the L stages. If product p manufactured on machine m at stage s immediately precedes product i manufactured on the same machine and stage, then the value of w(p,i,s,m) equals to one. Hence, the completion time of product i (manufactured on that machine with the same stage) is greater than the completion time of product p. The difference is by the sum of the setup time from product p to product i and the required processing time of product i on that machine with the same stage. Constraints (4) are needed to ensure that the makespan is equal to or greater than the completion time of each of the jobs in the last stage. Constraints (5) ensures that, for each product, it can be manufactured on only one of machines in that stage of a production line. Constraints (6) ensures that, except for the first product, a product scheduled on any machine is preceded by exactly one different product. Constraints (7) ensures that, except for the last product, a product scheduled on any machine must be immediately followed by only one product. Constraints (8) &(9) ensure that a machine can have exactly one first and one last product.

Model 2:

The objective function: Min E

Constraints:

$$F(i,L,m) \leq E \qquad (1)$$

i = 1,2,...,n ; and m = 1,2,...,m(L)

$$\sum_{i=1}^{n} w(p,i,s,m) \leq 1 \qquad (2)$$

i = 1,2,..., n ; m = 1,2,...,m(L), and s = 1,2,..., L

$$\sum_{p=0}^{n} \sum_{m=1}^{m(s)} w(p,i,s,m) = 1 \qquad (3)$$

i = 1,2,..., n ; and s = 1,2,..., L

$$\sum_{p=0}^{n} w(p,h,s,m) - \sum_{i=0}^{n} w(h,i,s,m) = 0 \qquad (4)$$

h = 1,2,..., n ; m = 1,2,...,m(L), and s = 1,2,..., L

$$F(i,s,m) - F(p,s,m) - ch(p,i,s,m) + V(1 - w(j,i,s,m)) \geq w(p,i,s,m) \cdot P(i,s,m) \qquad (5)$$

i,p = 1,2,..., n ; m = 1,2,...,m(L), and s = 1,2,..., L

$$F(i,s,m) \geq F(i,s-1,mp) + P(i,s,m) \cdot w(p,i,s,m) \qquad (6)$$

i = 1,2,...,n ; s = 2,3,...,L, m = 1,2,...,m(s), and mp = 1,2,...,m(s-1)

$$F(i,1,m) \geq P(i,1,m) \cdot w(p,i,1,m) \qquad ; i = 1,2,..., n ; m = 1,2,...,m(1) \qquad (7)$$

Constraints (1) are needed to ensure that the makespan is equal to or greater than the completion time of each of the jobs in the last stage. Constraints (2) ensures that each machine has to be assigned to at most one job. Constraints (3) ensures that each job is processed once and once in each stage. Constraints (4) ensures that, except for the first product, a product scheduled on any machine is preceded by exactly one different product and, except for the last product, a product scheduled on any machine must be immediately followed by only one product. Constraints (5) is about product sequencing on all the L stages detailed as in model 1. Constraints (6) ensures that the completion time of product i produced on machine m in the current stage (stage s) must be greater than its completion time in a previous stage (stage s-1). The difference must be equal to or greater than the amount of processing time required in the current stage. Constraints (7) is a completion time forcing constraints. It ensures that all products are scheduled and the completion time of any product on any machine of the first stage is at least the amount of processing time required for the product on that machine.

CHAPTER 5

IH ALGORITHM

Phase 1: Obtaining an Initial Solution Using the IH Algorithm

The heuristic developed in this phase schedules one product group at a time on the machines of the first stage. The algorithm then proceeds by scheduling products to the machines of all other stages. Prior to the presentation of the IH algorithm, the notation and variables used are defined.

Notation:

Let

i,p	=	product indices
j,q	=	product group indices
s	=	stage index
G	=	set of all product groups; G = {1,2,...,N}
m*	=	the minimum value of m(s); $m^* = \min_{s \in \Psi} m(s)$
M*	=	set of m* product groups selected to schedule as the first product group on each machine in stage one through stage *L*
G_R	=	set of all remaining product groups after assigning the first m* groups; G\{M*}
g_j	=	number of products in product groups j; j ∈ G
P_j	=	set of products in product groups j; j ∈ G
	=	{1,2,...,g_j}
Ψ	=	set of stages in a production line
	=	{1,2,...,L}
m(s)	=	number of machines in stage s; s ∈ Ψ
M(s)	=	set of machines in stage s
	=	{1,2,...,m(s)}
EM(1) .	=	set of m(1)-m* remaining machines
$v_{s,m}$	=	speed of machine m at stage s
ch(q,p,j,i,s)	=	The number of time units required to changeover from product i of group j to product p of group q at stage s

ST(j,i,s,m) = start time of product i of group j on machine m of stage s. There are 8 possible ways of determining the value of ST(j,i,s,m). A detailed description of these ways is presented in Appendix A.

PT(j,i,s,m) = processing time of product i of group j on machine m of stage s; $j \in G$, $i \in P_j$, $s \in \Psi$, and $m \in M(s)$.

T(j,i) = processing time of product i of group j on the standard machine in the first stage (i.e., speed of machine = 100%)

CT(j,i,s,m) = completion time of product i, group j on machine m of stage s. This time is equal to the sum of its start time and processing time.

= ST(j,i,s,m) + PT(j,i,s,m); $j \in G$, $i \in P_j$, $s \in \Psi$, and $m \in M(s)$

5.1 A Detailed Description of the IH Algorithm

The detailed description of the IH Algorithm is presented below in Parts 1 through 4.

Part 1: Assign the first m* groups to the machines at stage 1 through stage L-1.

Case 1: m* = m(1)

Step 1: Select the m* groups in order to assign them as the first group on the machines at stage 1 through L-1.

1.1 Find product i* amd m' with $\min_{i \in F_j} CH(0,0,j,i,s,m) + \sum_{s=1}^{S-1} PT(j,i,s,m)$ for $j \in J$, $s \in \Psi \setminus \{L\}$, $m \in M(s)$

1.2 Find j* where $i^* \in j^*$

1.3 Update $J = J \setminus \{j^*\}$; $M(s) = M(s) \setminus \{m'\}$; and count_j = count_j + 1

1.4 Check whether $J \neq \phi$ or count_j < m*.

If $J \neq \phi$, go to Step 1.1; otherwise, go to Step 2 of this pant.

Step 2: Rearrange the products of the m* groups on the first stage machines assigned to them

2.1 Schedule Product i* of group j* on machine m' i*.

2.2 Calculate completion times of the scheduled product where

$CTime(j,i,s,m') = STime(j,i,s,m') + PT(j,i,s,m')$.

2.3 Update $P_j = P_j \setminus \{i^*\}$.

If $P_j \neq \phi$, go to Step 2.4.

If $P_j = \phi$, update $G = G \setminus \{j^*\}$. If $M^* = \phi$, go to Part 2; otherwise, go to Step 1.

2.4 Find the next product.

Find i^* and m' with:

$$\min_{i \in G_j} \{ PT(j,i,s,m) + ch(j,p,j,i,1); j \in G \text{ and } m \in M(1)$$

where p is the last product scheduled so far on machine m at the first stage.

Then, go to Step 2.1 of this pant.

Case 2: $m^* < m(1)$

Step 1: Schedule m^* groups on the m^* machines using the same procedure as case 1 (begin with step 1 through step 2).

Step 2: Find group j^* and machine m' with $\max_{\forall j} CTime(j,i_{jp},s=1,m)$; for $m \in M(1)$ where $j \in J^*$ and i_{jp} is the product scheduled last in group j.

Step 3: Find i' and m'' with $\min_{i \in F_{j^*}} CH(0,0,j,i,s,mm) + PT(j,i,s=1,mm)$ where $mm \in EM(1)$.

Step 4: Schedule i' of group j^* on machine m'' and then calculate its completion time on this machine which is $CTime(q,p_{jp},s=1,m'') + ch(q,p_{jp},j^*,i',s=1) + PT(j^*, i', s, m'')$ and p_{jp} is the product scheduled last in group q.

Step 5: Rearrange the remaining products of group j^* on machine m' after schedule the removed product (i.e., product i') and then update the latest completion of the last product of group j^* on that machine.

Step 6: Check if the latest completion time is improved, perform the product reschedule and return to Step 2; otherwise do not remove the product from machine m', and go to Step 7.

Step 7: Repeat Step 2 through Step 6 with the product scheduled before the product used in the last removal attempt. If all attempts have been exhausted, proceed with Part 2.

Part 2: Assign the remaining groups to the machines at the first stage

Step 1: Find i' and m'' with $\min_{i \in F_j}$ CTime(q,$p_{[j]}$,s=1,m) +CH(q,$p_{[j]}$,j,i,s=1,m)+ PT(j,i,s=1,m) for j $\in G_R$ and m $\in$ M(1) and $p_{[j]}$ is the product scheduled last in group q.

Step 2: Find j* where i* $\in$ j* and update $G_R = G_R \setminus \{j^*\}$; if $J \neq \phi$, go to Step 3; otherwise, go to Part 3.

Step 3: Schedule group j* on m'' starting with product i'.

Step 4: Rearrange the products of group j* with the same procedure as Step 2 of Part 1. Except that, in Step 2.4, if $G = \phi$, go to Part 3.

Part 3: Balancing the Production Times of Machines at the First Stage

Step 1: Balance the production times of machines at the first stage.

Balancing the production times of machines at the first stage is performed by moving one or more of the products of a product group from the machine with the latest completion time to other machines such that the latest completion time of the first-stage machines is reduced. Balancing is performed after the assignment of all products to machines at the first stage has been completed. The procedure used to balance the production times of the first-stage machines is presented below:

1.1 Find the machine with the latest completion time (e.g., machine m')

1.2 Remove the last product scheduled on machine m'.

1.3 Calculate the latest completion time on each of the machines after scheduling the removed product last within its product group if scheduled on the machine;

otherwise, last on the machine. Select the one with the smallest updated completion time and the corresponding latest completion time.

1.4 If the latest completion time is improved, perform the product re-schedule and return to Step 1; otherwise, do not remove the product from machine m', and go to Step 5.

1.5 Repeat Steps 1 through 4 with the product scheduled before the product used in the last removal attempt. If all attempts have been exhausted, proceed with Part 4.

Part 4: Scheduling All products on All other Stages (i.e., stages 2,3,4,...,L)

After all products are completely assigned to the first-stage machines, the assignment of these products on machines at the succeeding stages needs to be performed. A Look Ahead (LA) rule developed by Sethanan (2001) (details are described in Appendix B) was developed to sequence the products on machines at stages 2 through S, in order to obtain low product finish times and a low makespan. The steps for Part 4 are given below.

Step 1: Schedule all products on all other stages (i.e., stage 2, 3, ... , L) and calculate the makespan

1.1 Set s = 2.

1.2 Set H = the set of products arranged in non-decreasing order of finish times from machines in stage s-1.

1.3 Schedule the first product (e.g., product i) in set H on one of the machines of stage s using the LA rule.

1.4 Update H = H \ {i}. If $H \neq \phi$, go back to Step 1.3. If $H = \phi$, update s = s + 1. If $s \leq L$, go to Step 1.2; otherwise, calculate the makespan and go to Phase 2.

Phase 2: Improving the Initial Solution Using the TSearch Algorithm

The initial solution obtained from Phase 1 (using the IH algorithm) may not be close to the optimal solution. A different heuristic is required to generate better schedules. The final solution of the first phase can be considered as an initial solution that will be improved in this phase. The heuristic of the second phase has three main steps: 1) moving groups between

(or within) machines at the first stage, 2) moving products between (or within) machines at the first stage, and 3) finding the best sequence resulting in the minimum makespan.

1. Implementing the TSearch Heuristic with the FFs(Rm_1,Rm_2...,Rm_s)/S_{ipnt}/C_{max} Problem

In the tabu search, a decision is made from the set of admissible candidates. The candidate decisions are evaluated and the best one is selected. A candidate is admissible either if it is not tabu or if its tabu status can be overridden by the aspiration criterion. As suggested by Laguna et al. (1993) and Barnes & Laguna (1993), there are four key elements to be considered in the TS:

- To identify the attributes (i.e., the criteria used to define or characterize a move) of a move that will be used to generate the tabu classification. Attributes of moves, e.g., indices of jobs (or jobs numbers), positions of jobs, and weights of jobs, are identified and recorded in the tabu list in order to prevent move reversals.

- To identify the actual tabu restriction based on the attributes.

- To identify a good data structure to keep track of moves that have a tabu status, and to free those moves from their tabu condition when their short-term memory has expired.

- To identify an aspiration condition in an effort to allow the tabu status of a move to be overridden if it yields a better solution.

Two popular types of moves found in the literature for the flowshop problem are: (1) exchanging jobs (i.e., swap move) and (2) removing the job placed at the x^{th} position and then putting it at the y^{th} position (i.e., insertion move). Taillard's (1990) experiments showed that the insertion move is the most efficient in terms of quality and computation time. Hence, only the insertion move will be considered in this research.

Part 5: Moving Groups between Machines (and within a Machine) at the First Stage

In this part, the groups scheduled on machines at the first stage are moved between machines (or within a machine) in an effort to minimize the makespan. This process is not performed for the other stages as it takes a large amount of computation time, and yields very little improvement. The best solution obtained from the previous Phase will be used as the initial solution. For each iteration, all the admissible moves within the neighborhood in the

current schedule are evaluated and the best move is selected. The tabu list, neighborhood size, and tabu restrictions are applied in the process of moving groups between machines at the first stage. The details of these three components are described below, and are followed by the notation used in this part and the detailed procedure of the TSearch algorithm.

Tabu List

Let N be the total number of groups. The size of the tabu list is determined as follows:

1. $m(1) = 1$.

   Based on the studies of Laguna et al. (1993), the size of the tabu list when jobs are moved within a machine is determined as described below.

   1.1 $N \leq 12$

   $|T| = \lfloor N / 2 \rfloor$

   where, $|T|$ = size of the tabu list

   1.2 $N > 12$

   $|T| = 10$

2. $m(1) > 1$

   1.1 If $2 \leq N \leq 10$, $1 \leq |T| \leq 3$.

   1.2 If $11 \leq N \leq 20$, $3 \leq |T| \leq 5$.

   1.3 If $21 \leq N \leq 50$, $5 \leq |T| \leq 10$.

   1.4 If $N > 51$, $10 \leq |T| \leq 15$.

Neighborhood Size and Tabu Restriction

In general, defining a good size of d depends on the structure of the problem. Based on studies by Laguna et al. (1993) and Barnes and Laguna (1993), the value of d can be obtained as follows:

Let $nf_{s,m}$ be the number of groups schedule on machine m in stage s.

- For $np_{s,m} \leq 30$

  $d = \lfloor np_{s,m}/2 \rfloor - 1$

  where $\lfloor h \rfloor$ = the largest integer less than or equal to h

- For $np_{s,m} > 30$

  $d = (\lfloor np_{s,m}/2 \rfloor / 2) \times c/4$

  where c is determined experimentally (Laguna et al., 1993 and Brandao & Mercer, 1997). The value of c is usually a number between 1 and 4 (Laguna et al., (1993)).

In this research, the neighborhood size and tabu restriction are determined as below:

1. For $m_1 = m_2 = m$
   - If $nf_{s,m} = 2$, d = 1.
   - If $3 \leq nf_{s,m} \leq 5$, d = 2.
   - If $6 \leq nf_{s,m} \leq 9$, d =3.
   - If $nf_{s,m} > 9$, the value of d is calculated using the same formula presented in the case of $nf_{s,m} \leq 30$. If $nf_{s,m} > 30$, the value of c is equal to 2.
2. For $m_1 \neq m_2$
   - If $nf_{s,m\,2} = 1$, or 2, d = 1.
   - If $nf_{s,m\,2} = 3$, d = 2.
   - If $4 \leq nf_{s,m\,2} \leq 9$, d = 3.
   - If $nf_{s,m\,2} \geq 10$, the value of d is calculated using the same formula presented in the case of $nf_{s,m} \leq 30$. If $nf_{s,m}\,2 > 30$, the value of c is equal to 2.

Notation

iter_gr	=	current iteration number for the process of moving groups between machines at the first stage
iter_max_gr	=	maximum number of iterations allowed to be performed in the product group insertion move procedure
best_value_gr	=	the minimum makespan found so far
best_seq_gr	=	the best schedule found so far
tor_iter_gr	=	maximum number of iterations allowed between two successive improvements
best_iter_gr	=	iteration where the best solution was found so far
size_tabu_list_gr	=	size of tabu list

move_value_gr = the minimum makespan obtained from the evaluation of all admissible moves in the iteration

move_seq_gr = the schedule that yields the minimum makespan in the iteration

Step 1: Initialize all parameters used in the process of moving groups between the machines at the first stage.

Set	iter_gr	=	0
	best_value_gr	=	makespan obtained in Phase 1 (Part 4)
	best_iter_gr	=	0
	iter_max_gr	=	500
	tor_iter_gr	=	200
	size_tabu_list_gr	=	10 for 10 groups (100 products)
		=	15 for 20 groups (180 products).

Step 2: Update the number of current iterations.

Increment the number of iterations (iter_gr) by 1.

Step 3: Check if the search should be stopped.

In this step, two stopping criteria are used:

3.1 Stop the search if the number of the current iterations (iter_gr) is greater than max_iter_gr, or

3.2 Stop the search if the number of successive iterations without improvement is greater than tor_iter_gr.

If the search is not stopped, go to Step 4; otherwise, go to Part 6 to proceed with the movement of products.

Step 4: Move groups between (or within) machines.

Groups that were divided between machines are treated as individual sub-groups. Sequences of products within groups (or sub-groups) are not changed in this step.

4.1 For each admissible move, perform the following:

- determine the tentative schedule of groups on machines in stage 1 after performing the move for the entire product group (or sub-product group).
- tentatively re-schedule all products on machines in stages 2 through L using the procedure detailed in Step 8 and find the corresponding makespan.

4.2 After all admissible moves have been performed, select the move that yields the minimum makespan. Denote the minimum makespan as move_value_gr and the corresponding schedule as move_seq_gr.

4.3 Check whether move_value_gr is less than the best_value_gr. If true, perform the following updates and go to Step 4.4

best_value_gr = move_value_gr,

best_seq_gr = move_seq_gr.

Otherwise, go to Step 4.4

4.4 Put the attribute of this move in the tabu list and go back to Step 1.

Part 6: Moving Products between (and within) Machines at the First Stage

In this part, the products are moved between (and within) machines in an effort to minimize the makespan. As in Part 5, the process of moving products between (and within) machines is performed only in the first stage. The best solution obtained in the previous part is used as the initial solution. The notation used in the implementation of the TS is described below and is followed by the procedure. Basically, the rules used to define the tabu list and to determine the tabu list size, neighborhood size, and tabu restriction are the same as in Part 5.

Notation

iter_pr = current iteration number for the process of moving products between machines at the first stage

iter_max_pr = maximum number of iterations allowed to perform in the process of products insertion procedure

best_value_pr = the minimum makespan found so far

best_seq_pr = the best schedule found so far

tor_iter_ pr = maximum number of iterations allowed between two successive improvements

best_iter_pr = iteration where the best solution has been found so far

size_tabu_list_ pr = size of tabu list

move_value_ pr = the minimum makespan obtained from the evaluation of all admissible moves in the iteration

move_ seq_pr = the schedule that yields the minimum makespan in the iteration

Details of this part are described as follows.

Step 1: Initialize all parameters used in the process of moving product between machines at the first stage.

Set iter_ pr = 0,

best_sol_ pr = makespan obtained in Part 5

best_iter_ pr = 0,

iter_max_ pr = 500,

tor_iter_ pr = 200,

size_tabu_list_pr = 10 for 100 products

= 15 for 180 products.

Step 2: Update the number of current iteration.

Increment the number of (iter_pr) by 1.

Step 3: Check if the search should be stopped.

The two stopping criteria used in Step 3 of Part 5 are also used in this step, as detailed below.

1. Stop the search if the maximum number of current iterations (iter_pr) is greater than max_iter_pr, or
2. Stop the search if the number of successive iterations without improvement is greater than tor_iter_pr.

If the search is not stopped, go to Step 4. Otherwise, go to Step 5.

Step 4: Move products between (or within) machines.

4.1 For each admissible move, perform the following:

- determine the tentative schedule of products on machines in stage 1 after performing a product move.
- tentatively re-schedule all products on machines in stages 2 through L using the procedure detailed in Step 8 and find the corresponding makespan.

4.2 After all admissible moves have been performed, select the move that yields the minimum makespan. Denote the minimum makespan as move_value_pr and the corresponding schedule as move_seq_pr.

4.3 Check if move_value_pr is less than best_value_pr. If true, perform the following updates and go to Step 4.4

best_value_pr = move_value_pr,

best_seq_pr = move_seq_pr.

Otherwise, go to Step 4.4

4.4 Put the attribute of this move in the tabu list and go back to Step 1.

Step 5: Determine the best makespan at the last stage and the best sequence found so far.

CHPATER 6

LOWER BOUNDS

6.1 Introduction

Normally, the quality of heuristic solutions is assessed by comparing their results to: (1) optimal solutions, (2) lower bounds, and/or (3) reference objective values obtained by the best known approximation algorithms. The flexible flowshop problem with sequence dependent setup is known to be NP-hard, and hence finding an optimal solution for average or large-size problems will be computationally intractable. The problem is also relatively new, and no approximation algorithms can be found for it in the literature. The only alternative left is to develop lower bounds for the problem and use them to assess the quality of the TSearch heuristic solutions.

6.2 Lower Bound Determination

Problem parameters and notation used in the development of the lower bound are defined below.

Notation

i, p	= product indices
Ψ	= set of stages in a production line
	= {1,2,..,L}
s	= stage index
n	= total number of products
N	= set of products
m(s)	= number of machines in stage s
M(s)	= set of machines at stage s
	= {1,2,..., m(s)}
V(s,m)	= the fastest speed of machine m at stage s to process products
$\lceil x \rceil$	= the least integer value greater than or equal to x.
SU(i)	= the setup time from idling for product i in stage 1
P(i,s)	= the processing time of product i on its fastest machine in stage s
ST(i,s)	= processing time of product i on a standard machine (i.e., speed = 100%) in stage s
CH(i,p,s)	= The number of time units required to changeover from production i to product p at stage s

CT(i,s) = the cumulative processing time of product i on its fastest machines from stage 1 through stage s-1

$= \sum_{s=1}^{s-1} P(i,s)$

MN(i,s) = the minimum setup time of product i at stage s. MN(i,s) is the lowest setup time for product i at stage s from any other product

$= \min_{i \neq p,} ch(p,i,s)$

ICT(i,s-1) = the sum of the setup time from idling of product i at the first stage and the cumulative processing times of product i on its fastest machines from stage 1 through stage s-1.

= SU(i) + CT(i,s-1)

δ = the minimum value between m(s) and m(1)

= min {m(s), m(1)}

xtra(s) = the difference between the number of machines in the first stage and that in stage s. If negative, a value of zero is used.

= max {0, m(1) - m(s)}

A = set of δ products with lowest values of SU(i)

B = set of δ products with lowest values of ICT(i,s-1)

C = N – B

D = set of m(1) products yielding the lowest values of SU(i)

$LB^{F,s}$ = the lower bound on the makespan calculated at stage s and obtained by the forward method

$LB^{F,L}$ = the lower bound on the makespan calculated at the last stage (stage L) and obtained by the forward method

$LB^{B,1}$ = the lower bound on the makespan calculated at the first stage (stage 1) and obtained by the backward method

LB^{F} = the best lower bound on the makespan obtained by the forward method

$= \max_{\forall s} \{LB^{F,s}\}$

BLB = the best lower bound

= max $\{LB^{F}, LB^{B,1}\}$

Based on the flow or routing of products, two methods were developed in this research to calculate a lower bound on the makespan: (1) the forward method and (2) the backward method. The lower bound on the makespan is a stage-based calculation, meaning that a value is calculated for each stage for the forward method, but it is calculated only for the first stage for the backward method. Then, the best lower bound (BLB) is obtained by taking the maximum value of the LB^{F} and $LB^{B,1}$, where the LB^{F} is the best lower bound on the makespan obtained by the forward method and calculated by taking the maximum value of the $LB^{F,s}$.

To calculate the lower bound on the makespan for the $FFS(R_{m1},R_{m2},\ldots,R_{mL})/s_{ipm}/C_{max}$ sequencing problem, the key idea is to consider a flexible flowshop structure with each machine in each stage as fast as its fastest speed. The makespan can be determined by considering the sum of three quantities: (1) the s-stage machine total waiting and idle times and (2) the total setup and production times on the s-stage machines, and (3) the last stage machine total waiting time. These three quantities can be divided into four components, as presented below.

- total waiting time at stage s (total_wait(s))
- total processing time of all products at stage s (total_proc(s))
- total setup time at stage s (total_setup(s))
- total waiting time when a products leave from stage s to the last stage (last_wait(s,L))

A detailed description of these components and how they are used to calculate $LB^{F,s}$ and $LB^{B,1}$ is presented in sections 6.2.1 and 6.2.2, respectively. For the forward method, the optimal makespan cannot be less than the sum of the two elements: (1) the sum of the first above three components divided by the number of machines in the s^{th} stage and (2) the machine waiting time at the last stage. Hence, using the forward method:

$$LB^{F,s} = \frac{1}{m(s)}[\text{total\_wait}(s) + \text{total\_proc}(s) + \text{total\_setup}(s)] + \text{last\_wait}(s,L)$$

Similarly, for the backward method (only consider the last stage), the last_wait(L) is not included. Hence, using the forward method:

$$LB^{B,1} = \frac{1}{m(1)}[\text{total\_wait}(1) + \text{total\_proc}(1) + \text{total\_setup}(1)]$$

6.2.1 Forward Method

1. Total waiting time at stage s (total_wait(s))

The total_wait(s) is the minimum amount of time that the machines at stage s have to wait until their first products are processed. This means that the first m(s) products have to complete their processing on stage 1 through stage s-1. Two cases are considered in calculating the total_wait(s).

Case 1: $m(s) \leq \delta$

In this case, there are two subcases:

1.1 Total waiting time at the first stage (total_wait(1))

$$\text{total\_wait(1)} = \sum_{i \in A} SU(i)$$

1.2 Total waiting time at the s^{th} stage; $2 \leq s \leq L$ (Total_wait(s))

The total_wait(s) is determined by summing the first δ, δ = m(s), smallest values of ICT(i,s-1).

Let

$\gamma(i,s) = SU(i) + CT(i,s-1)$

$\gamma[n,s]$ = the $\gamma(i,s)$ values sorted in non-decreasing order results in the n^{th} lowest value

$\gamma(i_n,s)$ = product i with the n^{th} lowest value of $\gamma[n,s]$

Hence:

$$\text{total\_wait(s)} = \sum_{i \in B} ICT(i,s-1) = \sum_{n=1}^{\delta} \gamma[n]$$

where $i_1 \neq i_2 \neq i_3 \neq \ldots \neq i_n$

Case 2: $m(s) > m(1)$

To find the total_wait(s) in this case, the machines in stage *s* are divided into two groups. The first group contains m(1) machines, and the second contains m(s) – m(1) machines (i.e. xtra(1)). The total waiting time for the machines in the first group (waiting_time_g1(s)) is calculated as the sum of the first δ smallest values of ICT(i): $\sum_{i \in B} ICT(i)$. For the second group, the key idea to find the minimum machine waiting time (wait_time_g2(s)) is to find the earliest start time of the remaining products on machine number m(1)+1, m(1)+2, ..., m(s). To calculate the wait_time_g2(s), the ratio (R) between xtra(1) and m(1) is determined and will be used. The R value is

determined as $\left\lceil \frac{m(s)-m(1)}{m(1)} \right\rceil$. Two cases are considered in calculating the machine waiting times in this group: (1) R = 1, and (2) R > 1. Details for each of these cases are described below.

(1) R = 1

The following procedure is followed:

Let $\Omega(i)$ = SU(i) + P(i,1); i ∈ N

= the finish time function of product i as it is the first product scheduled on the first stage machines.

$\beta(j,s-1)$ = min {min{MN(p,1)}, MN(j,1)} + CT(j,s-1); where, p ∈ A and j ∈ B

= the start time function of the remaining products on the m(s) – m(1) machines in the s^{th} stage

1.1 Let x be the machine number in the second group, x = 1,2,..., xtra(1). Set x=1.

1.2 Determine the machine waiting time on machine x using the following steps.

1.2.1 Sort all values of Ω (i) in non-decreasing order. Let $\Omega[1]$, $\Omega[2]$, ..., $\Omega[c]$ be the values resulting from the order. Then, find the product with the first lowest value of $\Omega(i)$ (e.g., product j):

$$\Omega(j) = \Omega[1] = \min_{i \in N} \Omega(i)$$

1.2.2 Sort all values of $\beta(i,s-1)$ in non-decreasing order. Let $\beta[1,s-1]$, $\beta[2,s-1]$, ..., $\beta[c,s-1]$ be the values resulting from the order. Then, find the product with the first lowest value of $\beta(i,s-1)$ (e.g., product g):

$$\beta(g,s-1) = \beta[1,s-1] = \min_{i \in C} \beta(i,s-1)$$

1.2.3 Check if j = g. If not true, calculate the waiting time of machine x in stage s (waiting_time(x,s)) and update set N as follows.

waiting_time(x) = $\Omega(j) + \beta(g,s-1)$

N = N \ {j}, delete $\beta(g,s-1)$

and go to step 1.3; otherwise, go to step 1.2.4.

1.2.4 Find the product with the second lowest value of $\Omega(i)$ (e.g., product j'):

$$\Omega(j') = \Omega[2] = \min_{i \in N \setminus \{j\}} \Omega(i)$$

1.2.5 Find the product with the second lowest value of $\beta(i,s-1)$ (e.g., product g'):

$$\beta(g',s-1) = \beta[2,s-1] = \min_{i \in N \setminus \{g\}} \beta(i,s-1)$$

1.2.6 Calculate the minimum waiting_time(x,s) as follows:

waiting_time(x,s) = min {Ω(j) + β(g',s-1), Ω(j') + β(g,s-1)}

1.2.7 If Ω(j) + β(g',s-1) < Ω(j') + β(g,s-1), update C = C – {j} and β(g',s-1). Otherwise, update C = C– {j'} and delete β(g,s-1).

1.3 Update x = x + 1. If x is greater than m(s) - m(1), go to step 1.4; otherwise, go back to step 1.2.

1.4 Calculate total_wait(s) as follows:

$$\text{total\_wait(s)} = \sum_{i \in A} ICT(i) + \sum_{x=1}^{m(s)-m(1)} waiting\_time(x,s)$$

(2) R > 1

For this case, the machines in the second group are divided into smaller subgroups of m(1) machines (the last subgroup may have a smaller number). The minimum waiting tine of the machines in the first subgroup (i.e., machine number m(1)+1, m(1)+2, ..., 2m(1)) is determined using the procedure detailed in case (1) (i.e., R = 1). To calculate the minimum waiting time for the machines of the remaining subgroups, the same procedure is repeated with the following modifications.

(1) Function Ω(i) is replaced with function α(i, w_1, w_2 ,...,w_r) which is defined as follows.

α(i, w_1, w_2 ,...,w_r) = SU(i) + P(i,1) + $\sum_{\sigma=1}^{r} \{MS(w_\sigma,1) + P(w_\sigma,1)\}$

where, i, $w_\sigma \in N$, σ = 1,2,...,r, i $\neq w_1 \neq w_2 \neq \ldots \neq w_r$

To calculate the waiting time on each subgroup of machines in the last stage, function α(i,w_1,w_2 ,...,w_r) must be regenerated for each r until the value of r reaches R-1. For instance, when r =1, the quantity α(i, w_1) is used to calculate the waiting time for the second subgroup of machines (i.e., machines 2·m(1)+1, 2·m(1)+2,..., 3·m(1)). Likewise, when r = R – 1, the quantity α(i, w_1, w_2 ,...,w_r) is used to calculate the waiting time for the R^{th} subgroup of machines (i.e., machines (R – 1)·m(1)+1,..., m(s)).

In step 2.1.2.1, all values of α(i, w1, w2 ,...,wr) obtained from all combinations of i and w_σ are sorted in non-decreasing order and let α[1], α[2], α[3],..., α[n] be the values resulting from the order.

(2) In step 2.1.2.3 of Case 2.1, product g is checked to find if it is a member of set ϖ, where ϖ is set of products (i, w_1, w_2 ,...,w_r) that yielded $\alpha[1]$.

(3) Steps 2.1.2.4 through 2.1.2.6 are modified to find the combination of $\alpha(\varpi)$ and $\beta(g,s-1)$ such that g is not a member of ϖ, which yield the minimum value of the sum of $\alpha(\varpi)$ and $\beta(g,s-1)$. Step 2.1.2.7 is then modified to update C = C – ϖ and delete $\beta(g,s-1)$.

The value total_wait(s) when R > 1 is calculated as follows:

total_wait(s) = wait_time_g1(s) + wait_time_g2(s)

$$= \sum_{i \in A} ICT(i) + \sum_{x=1}^{m(s)-m(1)} waiting\_time(x,s)$$

2. Total processing time of all products at the s^{th} stage (total_proc(s))

A lower bound of the total processing times on the machines at the last stage is calculated as the sum of the processing times of all products when processed on machines with the average speed in that stage. The value of total_proc(s) is hence calculated as follows:

$$\text{total\_proc(s)} = \frac{\sum_{i \in N} ST(i,s) \cdot m(s)}{\sum_{m \in M(s)} v_{s,m}}$$

3. Total setup time at the s^{th} stage (total_setup(s))

In minimizing changeovers, the number of machines assigned to each product should be as few as possible. Thus, the minimum number of setups for the entire production schedule on the s^{th} stage machines is equal to N - m(s) setups. The value of total_setup(s) is hence determined as the sum of the N – m(s) smallest changeovers.

$$\text{total\_setup(s)} = \sum_{i \in C} MS(i,s)$$

4. Total waiting time when a products leave from stage s to the last stage (last_wait(s,L))

The total machine waiting time when a products leave from stage s to the last stage (last_wait(s,L)) is the minimum amount of time that the last stage machine has to wait until the first product from stage (s+1) to be processed on the last stage. Hence:

$$\text{last\_wait}(s,L) = \min_{\forall i}\{\sum_{s'=s+1}^{L} P(i,s')\};\ s' = s+1, s+2, \ldots, L-1$$

$$= 0;\ s' = L$$

The overall lower bound by the forward method at stage s ($LB^{F,s}$) and the best lower bound on the makespan obtained by this method (LB^{F}) are then calculated as follows:

$$LB^{F,s} = \frac{1}{m(s)}[\text{total\_wait}(s) + \text{total\_proc}(s) + \text{total\_setup}(s)] + \text{last\_wait}(s,L)$$

$$LB^{F} = \max_{\forall s}\{LB^{F,s}\}$$

6.2.2 Backward Method

Consider a schedule where products are processed from stage L to stage 1 (i.e., reverse order of machines), then its antithetical schedule (mirror image) yields the same makespan for the original problem when no setup times are considered. With setup times, the lower bound for the backward schedule would still remain a lower bound for the original problem, when calculated as in the forward method with the following two adjustments:

1. Setup times from idling for the first m(L) products in stage L must not be considered when calculating total_wait(L) (i.e., assume SU(i) = 0 for all products, where SU(i) in this case is the setup time for product i from idling at stage L).
2. The sum of the m(1) minimum setup times from idling in stage 1 (sum_setup_idle(1)) should be added to total_wait(1).

The backward lower bound will then be calculated as follows:

$$LB^{B,1} = \frac{1}{m(1)}[\text{total\_wait}(1) + \text{sum\_setup\_idle}(1) + \text{total\_proc}(1) + \text{total\_setup}(1)]$$

The best lower bound (BLB) is then determined as max $\{LB^{F}, LB^{B,1}\}$.

CHAPTER 7

COMPUTATIONALEXPERIENCE

7.1 Introduction

This section will focus on computational experience with the heuristic algorithms (IH and TSearch). Two quantities are investigated: (1) the performance of the heuristic algorithms, obtained by comparing their solutions to the lower bound and (2) the relative improvement of the solutions obtained by the IH algorithm with respect to those of the TSearch algorithm.

Two sets of problems, with four types of data characteristics in each set, were generated to evaluate the above two quantities:

Set 1: 70-85 products (10 groups)

Set 2: 135-155 products (20 groups)

Four types (A, B, C and D) of data characteristics were generated for each set, and 5 test problems were generated for each data type. The parameters for each data type, processing times of products on a standard machine (speed = 1) at each stage (PTime(j,i,s,m)), machine speed deviations ($v_{s,m}$), changeover times between products at each stage (ch(j,i,q,p,s)), and setup times from idling of products at the first stage (ch(0,0,j,i,s)), were randomly selected from different uniform distributions as shown in Table 7.1

Table 7.1: Values of Parameters Used with the Different Data Types

Parameter	Type			
	A	B	C	D
Total number of machines and stages	12 machines, 4 stages (3,3,3)	20 machines, 5 stages (4,4,4,4,4)	12 machines, 4 stages (3,3,3)	20 machines, 5 stages (4,4,4,4,4)
PTime(j,i,s,m)	U[20,50]	U[20,50]	U[20,50]	U[20,50]
$v_{s,m}$	U[0.80, 1.20]	U[0.80, 1.20]	U[0.70, 1.30]	U[0.70, 1.30]
ch(j,i,q,p,s)	U[20%, 40%] of Time(j,i,s,m)	U[20%, 40%] of Time(j,i,s,m)	U[20%, 40%] of Time(j,i,s,m)	U[20%, 40%] of Time(j,i,s,m)
ch(j,i,j,p,s)	U[5%, 15%] of Time(j,i,s,m)	U[5%, 15%] of Time(j,i,s,m)	U[5%, 15%] of Time(j,i,s,m)	U[5%, 15%] of Time(j,i,s,m)
ch(0,0,j,i,s)	U[15%, 25%] of Time(j,i,s,m)	U[15%, 25%] of Time(j,i,s,m)	U[15%, 25%] of Time(j,i,s,m)	U[15%, 25%] of Time(j,i,s,m)

Changeover times between products at each stage (ch(j,i,q,p,s) and setup times from idling at the first stage (ch(0,0,j,i,s)) are identical on all machines at the same stage. Types A and B generate problems with small deviations in the speed of machines. Conversely, types C and D generate problems with large deviations in the speeds. Characteristics of the data types can be summarized as follows:

A: A small number of stages, small deviations in machine speeds, and small, identical number of machines in each stage.

B: A large number of stages, small deviations in machine speeds, and large, identical number of machines in each stage.

C: A small number of stages, large deviations in machine speeds, and small, non-identical number of machines in each stage.

D: A large number of stages, large deviations in machine speeds, and large, identical number of machines in each stage.

In section 7.2, the computational results obtained with the heuristics are presented and compared to the lower bounds for the large size problems. Section 7.3 presents the relative improvement of the solutions obtained by the IH algorithm with the application of the TSearch algorithm.

7.2 Comparison of the Results of Heuristic Algorithms with the Lower Bounds

The heuristic algorithms were coded in JAVA and run on a 2.0 GHz PC, with 256 MegaBytes of RAM, for testing and evaluation. In this section, the heuristic algorithms are evaluated using two performance measures: (1) solution quality, and (2) computational speed. The quality of a solution generated by the heuristics is measured in terms of their performance (HP), as presented below.

$$HP = (sol_{LB}/sol_{heu}) \times 100$$

where,

HP = the heuristic performance (%)

sol_{LB} = the lower bound of the solution

sol_{heu} = the solution obtained from the heuristic algorithms

The computational speed of the algorithms is measured by the amount of CPU time required to execute the algorithms. The CPU time includes compiling, linking, and execution times, and is reported in seconds and seconds per iteration for the IH and TSearch algorithms, respectively.

For each combination of problem set and data type, ten different test problems were generated. The solution of each test problem using the heuristic algorithm and its lower bound were obtained for all combinations of sets and data types. The results of these computations are presented in Tables 7.2-7.13. Table 7.14 shows the averages obtained for these results.

Table 7.2: Computational Results for Set 1 Type A:

Heuristic Algorithms vs. Lower Bound

Problem Number	CPU Time			Heuristic Performance (%)	
	IH (seconds)	*TSearch*		IH	TSearch
		seconds/iteration	*Number of Iterations (iterations)*		
1	*1.2*	2.1	*50*	*56.05*	*69.00*
2	*1.1*	2.2	*148*	*58.12*	*71.20*
3	*1.0*	2.0	*345*	*56.60*	*69.90*
4	*1.1*	2.1	*69*	*56.82*	*71.40*
5	*1.1*	2.0	*191*	*56.47*	*71.40*

Table 7.3: Computational Results for Set 1 Type B:

Heuristic Algorithms vs. Lower Bound

Problem Number	CPU Time			Heuristic Performance (%)	
	IH (seconds)	*TSearch*		IH	TSearch
		seconds/iteration	*Number of Iterations (iterations)*		
1	*1.0*	2.0	*150*	*58.87*	*69.10*
2	*1.1*	1.9	*156*	*55.42*	*68.50*
3	*1.0*	1.9	*245*	*53.69*	*69.60*
4	*1.1*	2.0	*145*	*56.50*	*71.10*
5	*1.1*	2.0	*89*	*55.60*	*68.90*

Table 7.4: Computational Results for Set 1 Type C:

Heuristic Algorithms vs. Lower Bound

Problem Number	CPU Time			Heuristic Performance (%)	
	IH (seconds)	*TSearch*		IH	TSearch
		seconds/iteration	*Number of Iterations (iterations)*		
1	*1.0*	2.0	*130*	*52.53*	*65.30*
2	*1.0*	2.0	*136*	*56.34*	*68.90*
3	*1.0*	2.1	*45*	*56.39*	*66.40*
4	*1.1*	2.0	*39*	*50.43*	*64.40*
5	*1.1*	2.0	*97*	*51.57*	*67.30*

Table 7.5: Computational Results for Set 1 Type D:

Heuristic Algorithms vs. Lower Bound

Problem Number	CPU Time			Heuristic Performance (%)	
	IH (seconds)	*TSearch*		IH	TSearch
		seconds/iteration	*Number of Iterations (iterations)*		
1	*1.0*	1.9	*120*	*50.04*	*67.10*
2	*1.1*	2.0	*135*	*50.80*	*65.60*
3	*1.1*	2.0	*128*	*50.33*	*64.10*
4	*1.2*	1.9	*123*	*58.61*	*67.70*
5	*1.0*	1.9	*171*	*53.45*	*68.70*

Table 7.6: Computational Results for Set 2 Type A:

Heuristic Algorithms vs. Lower Bound

Problem Number	CPU Time			Heuristic Performance (%)	
	IH (seconds)	*TSearch*		IH	TSearch
		seconds/iteration	*Number of Iterations (iterations)*		
1	*30.2*	*3.2*	*67*	*58.25*	*68.60*
2	*30.3*	*3.2*	*454*	*58.37*	*69.10*
3	*30.3*	*3.2*	*234*	*61.06*	*69.50*
4	*30.2*	*3.2*	*432*	*60.74*	*71.30*
5	*30.4*	*3.1*	*45*	*58.78*	*68.00*

Table 7.7: Computational Results for Set 2 Type B:

Heuristic Algorithms vs. Lower Bound

Problem Number	CPU Time			Heuristic Performance (%)	
	IH (seconds)	*TSearch*		IH	TSearch
		seconds/iteration	*Number of Iterations (iterations)*		
1	*31.2*	*3.2*	*453*	*57.93*	*68.70*
2	*31.1*	*3.3*	*234*	*58.11*	*69.90*
3	*31.1*	*3.3*	*458*	*58.90*	*69.90*
4	*31.2*	*3.4*	*120*	*58.41*	*67.60*
5	*31.1*	*3.1*	*56*	*58.27*	*70.40*

Table 7.8: Computational Results for Set 2 Type C:

Heuristic Algorithms vs. Lower Bound

Problem Number	CPU Time			Heuristic Performance (%)	
	IH (seconds)	*TSearch*		IH	TSearch
		seconds/iteration	*Number of Iterations (iterations)*		
1	*43.1*	*4.0*	*233*	*56.33*	*64.30*
2	*43.3*	*4.0*	*125*	*54.45*	*64.60*
3	*43.2*	*3.9*	*45*	*57.87*	*66.20*
4	*43.2*	*4.1*	*78*	*56.25*	*66.20*
5	*43.1*	*3.9*	*4551*	*56.18*	*64.90*

Table 7.9: Computational Results for Set 2 Type D:

Heuristic Algorithms vs. Lower Bound

Problem Number	CPU Time			Heuristic Performance (%)	
	IH (seconds)	*TSearch*		IH	TSearch
		seconds/iteration	*Number of Iterations (iterations)*		
1	*44.2*	*4.1*	*79*	*53.52*	*66.10*
2	*44.3*	*4.0*	*145*	*56.42*	*64.80*
3	*44.3*	*4.0*	*365*	*55.06*	*65.50*
4	*44.2*	*4.1*	*278*	*56.35*	*65.30*
5	*44.4*	*3.9*	*222*	*56.55*	*65.00*

Table 7.10: Averages of Computational Results for Sets 1 and 2 for all Data Types: Heuristic Algorithms vs. Lower Bound

Set	Type	CPU time			Heuristic Performance (%)	
		IH (seconds)	TSearch		IH	TSearch
			seconds/iteration	Number of iterations (iterations)		
1	A	1.10	2.08	160.60	56.81	70.58
	B	1.06	1.96	157.00	56.02	69.44
	C	1.04	2.02	89.40	53.45	66.46
	D	1.08	1.94	135.40	52.65	66.64
2	A	30.28	3.18	246.40	59.44	69.30
	B	31.14	3.26	264.20	58.32	69.30
	C	43.18	3.98	1006.40	56.22	65.24
	D	44.28	4.02	217.80	55.58	65.34

Based on these results, the average performance for set 1 ranges between 52.65-56.81% for the IH algorithm and 66.46-70.58% for the TSearch algorithm. For set 2, the average performance is lower than that of set 1, and ranges between 55.58-59.44% for the IH algorithm and 65.24-69.30% for the TSearch algorithm.

The computational times for the IH are extremely small-- less than 45 seconds. These times do significantly increase with the size of the problem. This means that the IH algorithm is sensitive to the problem size. In contrast, computational times for the TSearch algorithm seem to be high-- between 1.94 and 2.08 seconds per iteration for data set 1 and between 3.18 and 4.02 seconds per iteration for data set 2. These times increase significantly with the size of the problem in terms of numbers of products (product groups), stages, and machines.

A Factorial Design was used to evaluate the performance of the heuristic algorithms (HP). The design has three factors: deviations in machine speeds, number of products, and number of machines and stages. The analysis was performed using SAS Software V8 for Windows and the results are presented in Appendix C. The statistical results show a significant effect only for two factors on the heuristic performance, number of products and deviation in machine speeds. Tukey's test was performed to compare between the three means obtained with different number of machines and stages. Results of the test (see Appendix C) indicate that the two means are different from each other.

The statistical results obtained from ANOVA and Tukey's test show that the heuristic performance declines with the increase of: (1) number of products, and (2) deviation in machine speeds. This decline is due mainly to the decrement in the value of the lower bound rather than the performance of the heuristics. The lower bound value may be affected by the following factors:

(1) the difference between the actual processing times and the smallest processing times of products used to calculate the first component of lower bound. The difference in processing times gets larger when the difference in the speeds between the fastest and the slowest machines increases.

(2) the difference between actual processing times and the processing times on the average speed machine of products used to calculate the second component of the lower bound, and

(3) the difference between actual setup times (both major and minor setup times) and the smallest setup times of the products, used to calculate components 3 and 4 of the lower bound.

If the differences were small, the lower bound would be relatively high resulting in higher algorithm performance, and vice versa. Larger deviations in machine speeds, a number of products (groups), and of machines and stages would most probably cause larger differences in processing times and setup times.

7.3 Comparison between the IH Algorithm and the TSearch Algorithm

In this section, the relative improvement of the solutions obtained from the IH algorithm after applying the TSearch is evaluated and presented below.

$$\text{Let } RI = \{(sol_{IH}/ - sol_{TSearch}) / sol_{IH}\} \times 100$$

where,

RI = the relative improvement (%) between sol_{IH} and $sol_{TSearch}$

sol_{IH} = the solution obtained from the IH algorithm

$sol_{TSearch}$ = the solution obtained from the TSearch algorithm

Two sets of relatively large size problems are used in this section. These sets are identical to those described in Section 7.2. For each combination of problem set and data type, 5 different test problems were generated. The solutions of each test problem using the IH and TSearch algorithms were obtained for all combinations of sets and data types. The results obtained are presented in Tables 7.11 and 7.12. Table 7.13 shows the averages obtained for these results.

Table 7.11: Relative Improvement Results for the Different Data Types in Set 1:

Problem Number	*Relative Improvement (%)*			
	Type			
	A	*B*	*C*	*D*
1	*18.82*	*14.79*	*19.56*	*25.44*
2	*18.38*	*19.15*	*18.18*	*22.61*
3	*19.06*	*22.86*	*15.10*	*21.53*
4	*20.41*	*20.49*	*21.75*	*13.39*
5	*20.96*	*19.25*	*23.40*	*22.23*

Table 7.12: Relative Improvement Results for the Different Data Types in Set 2:

Problem Number	*Relative Improvement (%)*			
	Type			
	A	*B*	*C*	*D*
1	*15.07*	*15.63*	*12.36*	*19.00*
2	*15.50*	*16.87*	*15.73*	*12.99*
3	*12.19*	*15.71*	*12.57*	*15.92*
4	*14.76*	*13.53*	*15.02*	*13.64*
5	*13.59*	*17.28*	*13.47*	*13.01*

CHAPTER 8

CONCLUSIONS AND RECOMMENDATIONS

This research was undertaken to minimize the makespan for the "flexible flowshop with sequence dependent setup times when machines in each stages are unrelated" problem. Two exact algorithms were first developed and used to solve small problems. Two heuristic algorithms (IH and TSearch) were then developed to solve larger and more practical problems. In order to evaluate the performance of the heuristic algorithms, two lower bounds were developed for the solution of the problem.

Since the optimal solution can be obtained for only small size problems (only for 5 jobs and 7 machiines), two heuristic algorithms (IH and TSearch) were developed. The first algorithm (IH) was developed to obtain a good initial solution and then improved in the second phase using the TSearch algorithm. To assess the quality of the heuristic algorithms, two methods were presented for obtaining a lower bound for the flexible flowshop with sequence dependent setup times when machines in each stages are unrelated problems: (1) forward method and (2) backward method. Machine waiting time, idle time, and the total setup and processing times on machines at the last stage were used to obtain the lower bounds.

For the computational experience, two data sets with four problem configurations for each set were generated, and five test problems were generated for each configuration. The performances of the heuristics were presented and evaluated using two measures: (1) solution quality and (2) computational speed. The quality of heuristic solutions was evaluated using lower bounds. The results showed a performance for the IH algorithm between 52.65-56.81% for data set 1 and 55.58-59.44% for data set 2. The performance for the TSearch algorithm ranged between 66.46-70.58% for data set 1 and 65.24-69.30% for data set 2. The performance of the algorithms declined with the increase of: (1) deviation in machine speeds and (2) number of products.

The computational times were very small for the IH algorithm, indicating that this algorithm is very efficient and not sensitive to problem size. Conversely, the computational times of the TSerach algorithm increased significantly with problem size--number of products, stages, and machines. For the relative improvement realized when applying the TSearch algorithm to the results obtained with the IH algorithm, the results indicated an improvement between 12.20 and 25.50%. This improvement increased as the deviations in machine speeds, number of stages, and machines increased. On the other hand, it decreased as the number of products (groups) increased.

8.1 Contribution of the Research

The exact algorithms as well as the heuristic algorithms and the lower bound methods developed can also be applied to identical, uniform, and unrelated parallel processing problems with or without dependent setup times. Computational experience showed that both heuristic algorithms are effective in solving the problem.

8.2 Recommendations for Future Research

The following recommendations are made for future research:

- The calculation of the lower bounds may be further enhanced
- Improvements may be made to the TSearch algorithm. The Tabu search was utilized in this research without using intensification or diversification strategies. These strategies, which are used to guide the search in a more intelligent way, need to be further studied.
- Other search methods (e.g., Neural Network or Genetic Algorithm) may be applied to solve this problem. Their performances may be compared to that of the Tabu Search algorithm.

CHPATER 9

OUTPUTS จากโครงการวิจัยที่ได้รับทุนจาก สกว.

9.1 **Software ของ Heuristic Algorithm *ในรูปแบบ* GUI (Graphic User Interface)** ซึ่งวิธีการใช้แสดงไว้ภาคผนวก *D*

9.2 **Software ของการคำนวณค่า Lower Bounds *ในรูปแบบ* GUI (Graphic User Interface)** ซึ่งวิธีการใช้แสดงไว้ภาคผนวก *E*

9.3 ***การนำเสนอผลงานการประชุมวิชาการระดับนานาชาติ*** **(International Conference)**

9.3.1 The 33rd International Conference on Computers and Industrial Engineering: Detailed is shown in Appendix F-1
(Held at Jeju, Korea, on March 25-27, 2004)

9.3.2 The Fifth Asia-Pacific Conference on Industrial Engineering and Management Systems: Detailed is shown in Appendix F-2
(Held at Gold Coast, Australia., on December 12-15, 2005)

REFERENCES

Allahverdi, A., Gupta, J. N. D., & Aldowaisan, T. (1999). *A Review of Scheduling. Research Involving Setup Considerations*, OMEGA, The International Journal of Management Science, 27: 219-239.

Barns, J. W. & Laguna, M. (1993). *Solving the Multiple-Machine Weighted Flow Time Problem Using Tabu search.* IIE Transactions, 25(2), 121-128.

Brah, S. A. & Hunsucker, J. L. (1991). *Branch and Bound Algorithm for the Flow Shop with Multiple Processors.* European Journal of Operational Research, 51, 88-99.

Gupta, J. N. D. & Tunc, E. A. (1994). *Scheduling a Two-Stage Hybrid Flowshop with Separable Setup and Removal Time.* European Journal of Operational Research, 77, 415-428.

Kurz M. E., & Askin, R. G. (2004). *Scheduling Flexible Flow lines with Sequence Dependent Setup Times.* European Journal of Operational Research, 159, 66-82.

Laguna, M. & Barns, J. W. & Glover, F. (1993). *Intelligent Scheduling with Tabu Search: An Application to Jobs with Linear Delay Penalties and Sequence-Dependent Setup Costs and Times.* Journal of Applied Intelligence, 3, 159-172.

Portmann, M-. C., Vignier, A., Dardilhac, D. & Dezalay, D. (1998). *Branch and Bound Crossed with GA to Solve Hybrid Flowshops*, European Journal of Operational Research, 107, 384-400.

Randhawa, S. U. & Smith, T.A. (1995). *An Experiment Investigation of Scheduling Non-Identical, Parallel Processors with Sequence-Dependent Setup Times and Due Date*, International Journal of Production Research, 33(1), 59-69.

Ruiz R., Maroto, C., & Alcaraz A. (2005). *Solving the Flowshop Scheduling Problem with Sequence Dependent Setup Times Using Advanced Metaheuristics.*, European Journal of Operational Research, 165, 34-54.

Sethanan, K. (2001). *Scheduling Flexible Flowshops with Sequence Dependent Setup Times.* Doctoral Dissertation, West Virginia University. West Virginia.

Simons Jr.,J. V., (1992). *Heuristics in Flowshop Scheduling with Sequence Dependent Setup Times.* OMEGA, The International Journal of Management Science, 20(2): 215-225.

Srikar, B. N. & Ghosh, S. (1986). *A MILP Model for the n-job, m-stage Flowshop with Sequence Dependent Set-up Times.* International Journal of Production Research, 24(6), 1459-1474.

Tahar D. N., Yalaoui F., Chu C., & Amodeo L. (2005). *A Linear Programming Approach for Identical Parallel Machine Scheduling with Job Splitting and Sequence dependent Setup Times.* International Journal of Production Economics. To Appear.

Tillard, E. (1990). *Some Efficient Heuristic Methods for the Flow Shop Sequencing Problem.* European Journal of Operational Research, 47(1), 65-74.

Wilbrecht, J. K. & Prescott, W. B. (1969). *The Influence of Setup Time on Job Shop Performance, Management Science*, 16, B274-B280.

Appendixes