國 立 交 通 大 學

資訊管理研究所

碩　士　論　文

平行流線型機組排程-總完工時間之最小化

Scheduling on Multiple Flowshops to Minimize the Makespan

研究生：林延聰

指導教授：林妙聰 博士

中華民國九十六年六月

平行流線型機組排程-總完工時間之最小化
Scheduling on Multiple Flowshops to Minimize the Makespan

研　究　生：林延聰　　Student: Yen-Tsung Lin
指導教授：林妙聰　　Advisor: B.M.T. Lin

國立交通大學
資訊管理研究所
碩士論文

A Thesis
Submitted to Institute of Information Management
College of Management
National Chiao Tung University
in Partial Fulfillment of the Requirements
for the Degree of
Master of Business Administration
in
Information Management
June 2006
Hsinchu, Taiwan, the Republic of China

中華民國九十六年六月

論文名稱：平行流線型機組排程-總完工時間之最小化

校院系：國立交通大學資訊管理研究所

研究生：林延聰　　　　　　　　　　　　　　　　　　　指導教授：林妙聰 博士

論文摘要

流線型機組排程是現今常見的排程問題之一，而總完工時間更是流線型機組最常探討的問題。在過往的研究中，都是以單一條流線型機組為基礎，進行各類問題討論，本研究以分析平行獨立流線型機組的總完工時間為主軸，在機組機器數量為兩台的情況下，進行討論分析。本問題在機組數量為一條時，為最原始的流線型機組問題；在機組機器數量為一台時，則為平行機組問題。單就平行機組問題已是 NP-Hard，因此再加上流線型機組的條件，此問題亦是 NP-Hard。

本研究設計了動態規劃及兩個啟發式演算法，利用實驗比較兩個啟發式演算法的優劣，再利用禁忌搜尋法，搭配兩個啟發式演算法和下界函數進行比較分析。本研究實驗結果顯示，我們所提個兩個啟發式演算法效能優越，在流線型機組數量愈多時，第二個啟發式演算法會拉大與第一個啟發式演算法的差距，且都僅需零到一秒的時間內，即可解出一百組工作的解。

關鍵字：流線型機組、平行機組、動態規劃、下界函數、啟發式演算法、禁忌搜尋法

Title of Thesis: Scheduling on Multiple Flowshops to Minimize the Makespan
Name of Institutes: Institute of Information Management
Student Name: Yen-Tsung Lin Advisor Name: Dr. Bertrand M.T. Lin

Abstract

In this thesis, we consider a multi-flowshop scheduling problem, where a set of jobs is to be processed on multiple identical flowshops. The objective is to minimize the makespan, i.e. the maximum completion time of the jobs. This problem is NP-hard because it is a generalization of the classical parallel-machine scheduling. A dynamic programming algorithm is designed for finding exact solutions. When the number of flowshops is fixed, the time complexity is pseudo-polynomial, the same as that for parallel identical-machine scheduling. To report approximate solutions in a reasonable time, we develop two dispatching heuristics and a tabu search algorithm. Extensive experiments are conducted to examine the performances of the proposed algorithms. Computational results show that the designed algorithm can produce solutions with errors within 1% for all test instances.

Keyword: Flowshop, Parallel machines, Dynamic Program, Lower Bound, Heuristic, Tabu Search

誌謝

交大資管所兩年的生活很快的就要結束了，能在短短兩年中學習到以往未曾接觸過的領域，我感到相當的滿足，很感謝一路來給我方向以及不同視野的筱嵐學姊及雅梅學姊，總在我迷網時點醒我該走的路；也很感謝昱劭和癸棠兩位學弟，在我研究苦悶時，陪我去運動打球，同時為實驗室帶來更多歡樂的氣氛；還有我最重要的研究伙伴彥廷，他那讓我擔心的脫線個性和那令我羨慕的聰明才智，使我相當欣賞也喜愛與他相處，能和他一同從交大資管所畢業，是我一生感到開心與驕傲的事；我也要感謝一路指導我的林妙聰老師，他苦心帶領我進到排程這個領域，我在學習上老是狀況百出，他卻總是包容與關懷的幫助我找出問題所在，很感謝他的耐心與細心指導，才能成就出今天這篇論文。

我更要感謝的人是這兩年來任勞任怨的父母，他們總深信著自己兒子是天才的美夢，雖然我嘗試說破這個白日夢，卻無法撼動他們心中的信仰，但也因為他們的信任，使我不斷督促自己往前邁進；還有做事少根筋的大姊芸蓮，總是在我需要鼓勵時，給我站起來的力量，以及擁有女強人氣質的二姊穎秀，只要有什麼好康的東西都不會忘了幫我買一份。我最感謝的人是兩年來和我一起成長進步，陪著我哭陪著我笑的怡君，無論任何情況你總是在的身旁支持與鼓勵，讓我能夠成就今天的榮耀，真的很謝謝你。最後，我感謝這兩年參與我生命的每一個人，謝謝大家，謝謝！

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

Scheduling is a decision-making process about how we allocate limited resources, including time, facilities, and work force, to tasks or activities. This type of the decision-making processes is important and common in procurement and production. Allocating limited resources may face different conditions and different objectives in each company. Therefore, the scheduling literature covers a variety of objectives, including for example makespan ($C_{\max}$) and total completion time ($\sum C_i$). The theoretical results of machine scheduling have abounded and applied since the 1950s. In this thesis, we consider a scheduling issue that arises in the production line of industry.

Production models can be categorized into several types, including single-machine, parallel-machine, flowshop, job shop and open shop. Among the shop models, flowshop scheduling is the most widely discussed due to its real-world applications and several interesting structure properties. Before we formally define the flowshop problem, we provide an example to illustrate its configuration and properties. There are two tasks (operations), washing and waxing, for each car (job) to be processed in a car washing site. Each car needs to wash before to wax. We assume machine one is for washing and machine two is for waxing. An ordering of all cars on machine one must be the same as that on machine two. This is called a two-machine flowshop in which all jobs need to visit machine one and then machine two, and a machine-two operation cannot start until the corresponding machine-one operation is finished and machine two is not occupied. With the two-machine model, we can generalize it to a multiple-machine model where a number of operations have to be done for every job and the operations have a fixed sequence of machines to visit, and the sequence is same to all jobs.

This thesis focuses on the flowshop model with two machines. The objective we investigate is the maximum completion time among the jobs, i.e. the makespan. Scheduling to minimize makespan in a two-machine flowshop is commonly denoted by the three-field notation $F2||C_{\max}$, in which $F2$ indicates the two-machine flowshop environment and $C_{\max}$ specifies the objective to optimize. The problem was proposed and investigated by Johnson (1954). In his seminal work, he proposed an $O(n \log n)$ algorithm to solve the problem to optimality. This algorithm is referred to as Johnson's algorithm. While the $F2||C_{\max}$ problem can be solved in polynomial time, this thesis investigates a more complicated variant where multiple independent flowshops are available for processing the jobs. We denote the problem by $lF2||C_{max}$, where $lF2$ specifies the machine configuration with l independent two-machine flowshops.

It is common for a plant to have more than one production line. Moreover, a plant can rent or borrow production lines from its peers in the industry to fulfill the production demand. Therefore, we the multiple independent flowshop models can better reflect the real-world applications. From the theoretical point of view, the proposed model can be regarded as a hybridization of the standard $F2$ model and the standard parallel-machine model. In a parallel-machine model ($P2$), each job has exactly one operation that can be processed on any of the machines. It is easy to see that $P2||C_{\max}$ is NP-hard in the ordinary sense because it is can be reduced from the PAPTITION problem. Therefore, the problem we want to study is also NP-hard.

To the best of our knowledge, there is no previous work on the $lF2||C_{\max}$ problem. In the following, we introduce previous results on flowshop scheduling and parallel-machine scheduling. The solution algorithm for the $F2||C_{\max}$ problem was proposed by Johnson (1954). He resolved the two machines of flowshop problem and then identified two special cases of three-machine flowshops which can be solved in polynomial time. An integer programming formulation of $Fm||C_{\max}$ is presented by Wagner (1959). The NP-hardness proof of $F3||C_{\max}$ was presented by Garey, Johnson, and Sethi (1976). Monma and Rinnooy Kan (1983) gave an overview of $Fm||C_{\max}$ models to elaborate the special structures that make the problem polynomially solvable.

To minimize the makespan in a parallel-machine environment, the shortest processing time first (SPT) rule is the most well studied heuristic. The worst case analysis of the LPT rule for $Pm||C_{\max}$ was presented by Graham (1969). Graham gives a general bound, a modified system and some special case. Finally, he provides an algorithm and some theorems for list scheduling. The first examples of worst case analyses of heuristics are presented by Graham (1966). It also provides a worst case analysis of an arbitrary list schedule for $Pm||C_{\max}$. A more sophisticated heuristic for $Pm || C_{\max}$, with a tighter worst case bound, called MULTIFIT, was proposed by Coffman, Garey, and Johnson (1978) and Friesen (1984a). MULTIFIT is well known to have an error bound better than the LPT rule at the cost of a longer running time. Lee and Massey (1988) analyze a heuristic that is based on the LPT rule as well as on the MULTIFIT. They design the combined algorithm in order to integrate each advantage.

In this thesis, we design a dynamic programming algorithm to optimally solve the multiple flowshops problem. Furthermore, we also design a lower bound and two heuristics, which are used to produce an initial solution of the tabu search approach. The rest of this thesis is organized as follows. We define the multiple flowshops problem and design the dynamic programming algorithm in Chapter 2. In Chapter 3, we introduce the existing lower bounds, develop two heuristics and design our tabu search approach. Computational experiments and analysis of the results are included in Chapter 4. Finally, we give our conclusion in Chapter 5.

Chapter 2 Problem Formulation and Dynamic Program

In this chapter, we introduce the definition of the parallel-flowshop scheduling problem, followed by a numerical example. Then, we propose a dynamic programming algorithm for producing optimal schedules.

2.1 Problem Formulation

There is a set of jobs $N = \{J_1, J_2, \ldots, J_n\}$ to be processed in l simultaneously available identical flowshops F_1, F_2, …, and F_l, each of which consists of two machines. Any job can be assigned to any flowshop. Any job J_i in N has two operations that will be performed by the two machines of the flowshop it is assigned to. The processing times of the two operations of job J_i are denoted by p_i and q_i, respectively. Each machine can process at most one job at a time, and no preemption is allowed. Moreover, once a job is dispatched to a flowshop, it cannot be transferred to any other flowshop. The objective is to minimize the makespan, i.e. the maximum completion time of the jobs.

To illustrate the problem definition, we consider the following set of 7 jobs $N = \{J_1, J_2, \ldots, J_7\}$ to process in two independent flowshops. Assume jobs $\{J_1, J_5, J_7\}$ are dispatched to F_1 and jobs $\{J_2, J_3, J_4, J_6\}$ to F_2, and the jobs are sequenced in increasing order of their indices in flowshops. The Gantt chart of the schedule is shown in Figure 1. We can see from the chart that the maximum completion time is 30 units of time. Figure 2 shows another solution that has a better makespan of only 26 units of time.

Table 1: Example of multiple flowshops

Job *i*		J_1	J_2	J_3	J_4	J_5	J_6	J_7
processing times	p_i	1	3	4	5	6	7	22
	q_i	22	7	6	3	2	1	1

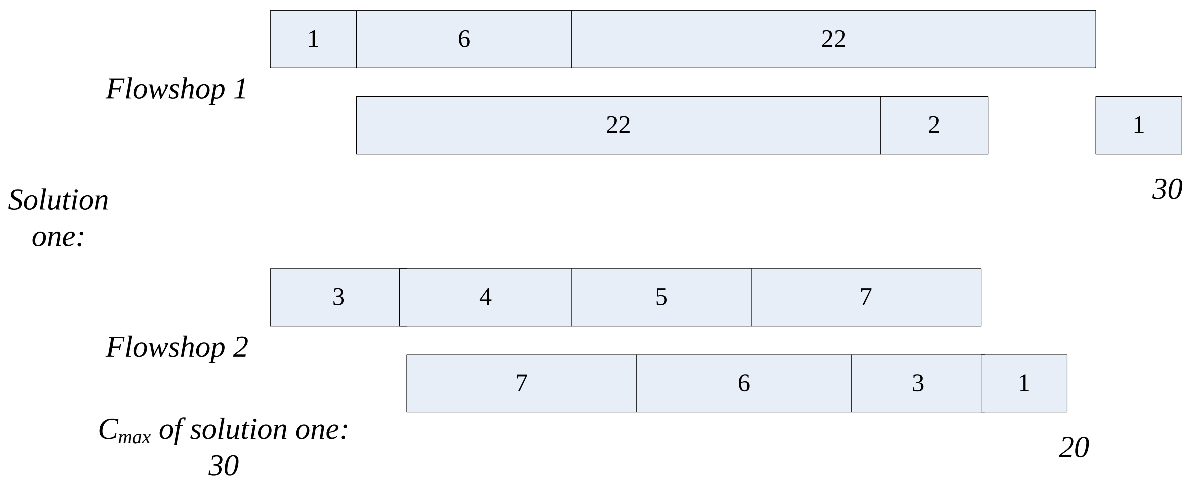

Figure 1: Gantt chart of solution 1

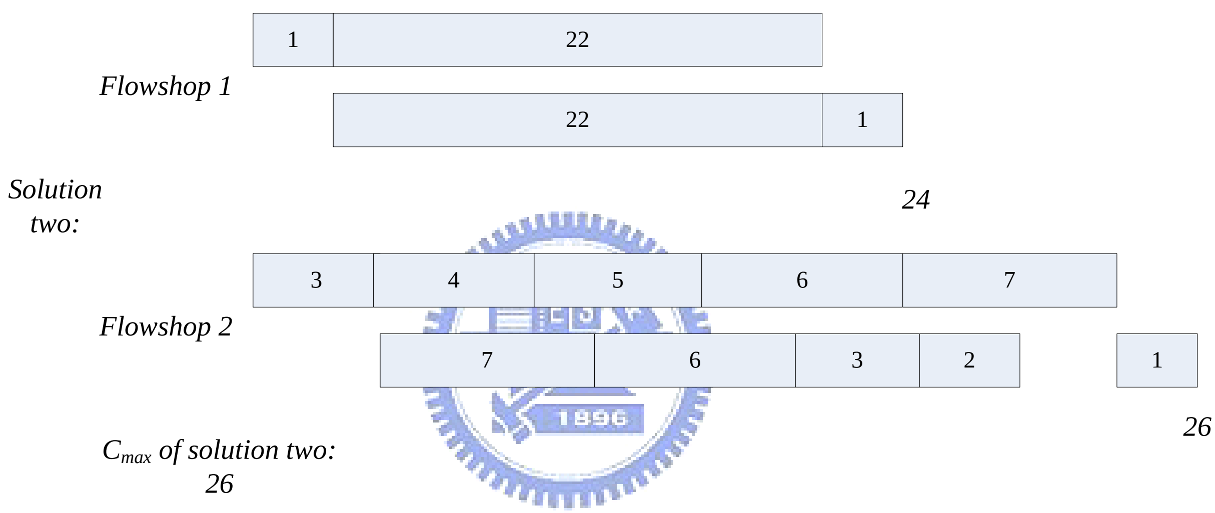

Figure 2: Gantt chart of solution 2

From the above example, we learn that the problem we are studying consists of two decisions: dispatching jobs to flowshops and sequence the jobs on each individual flowshop. Consider some special cases. If there is only one flowshop, i.e. l = 1, then the problem becomes scheduling in the classical two-machine flowshop and thus solvable in $O(n \log n)$ time using Johnson's algorithm (Johnson 1954). On the other hand, if we assume that the machine-one operations of all jobs are null or the machine-two operations of all jobs are null, then the problem is scheduling with parallel identical machines which is known to be NP-hard in the strong sense (Garey and Johnson, 1979). Therefore, the parallel-flowshop scheduling problem is strongly NP-hard.

2.2 Dynamic Programming Algorithm

To optimally solve an NP-hard problem enumerative approaches are considered. Among the enumerative approaches, dynamic programming strategy and branch and bound algorithm are most commonly adopted. In this section, we design dynamic programming algorithm to solve the problem optimally. Dynamic programming strategy is a very useful technique to solve many combinatorial optimization problems. It is based on a concept called the principle of optimality. In this thesis, we design a backward reasoning dynamic programming algorithm. The following property gives the base for development of a dynamic programming algorithm.

PROPERTY 1: There is an optimal solution in which the jobs in each flowshop are arranged by Johnson's rule.

With PROPERTY 1, we can then re-index the jobs of N in Johnson's rule. To design our dynamic programming algorithm, we define binary function $G(j, t_{1,1}, t_{1,2}, \ldots, t_{k,1}, t_{k,2}, \ldots, t_{l,1}, t_{l,2})$ as follows: Given that jobs 1, 2, …, j are considered, the function value is 1 if for each flowshop F_k the completion times of the last job on the two machines are exactly $t_{k,1}$ and $t_{k,2}$; 0, otherwise. Considering the last job J_j, there are l choices to dispatch it. Therefore, the recursion will take into account of all l different scenarios. On any machine job J_j is considered, it is scheduled last. Therefore, we need to examine the possible completion times of the two machines if we removed job J_j for recursion. A dynamic programming algorithm is accordingly defined in the backward recursive form as shown below.

Dynamic Programming Algorithm

Initial conditions:

$$G(j, t_{1,1}, t_{1,2}, \ldots, t_{l,1}, t_{l,2}) = \begin{cases} 1, & \text{if } j = t_{1,1} = t_{1,2} = \cdots = t_{l,1} = t_{l,2} = 1 \\ \infty, & \text{otherwise.} \end{cases}$$

Recursive formula: For $1 \le j \le n$,

$G(j, t_{1,1}, t_{1,2}, \ldots, t_{l,1}, t_{l,2}) = 1$,

if there exists some flowshop F_k in which there exists some time point t', $t_{k,1} - p_j \le t' \le t_{k,2} - q_j$, such that

$G(j-1, t_{1,1}, t_{1,2}, \ldots, t_{k,1} - p_j, t', \ldots t_{l,1}, t_{l,2}) = 1$.

Otherwise, it is 0.

Goal:

Minimize $\min\left\{\max\left\{t_{k,2}\right\} \mid G(n, t_{1,1}, t_{1,2}; \ldots; t_{k,1}, t_{k,2}; \ldots; t_{l,1}, t_{l,2}) = 1\right\}$

The above formulation consists of $O(nP^l C_{JS}^l)$ entries, where $P = \sum_{i=1}^{n} p_i$ and C_{JS} is the makespan of the Johnson's sequence of all jobs. In the recursion formula, each entry is determined in $O(C_{JS})$ time. Therefore, the overall time complexity of the above algorithm is $O(nP^l C_{JS}^{l+1})$. The complexity is clearly exponential. But when the number of flowshops l is constant, the complexity becomes pseudo-polynomial. It is interesting to note that the complexity status is the same as that of parallel-machine scheduling. Although each machine in a parallel-flowshop is replaced by a two-machine flowshop, the complexity remains the same.

From a practical point of view, the dynamic programming algorithm is not appropriate for implementation taking into account memory space and flexibility. In later sections, approximation algorithms will be developed to produce approximate solution in a reasonable time.

Chapter 3 Approximation Algorithms

As the parallel-flowshop scheduling problem is strongly NP-hard, it is very unlikely to design algorithms that have can solve the problem in polynomial time. Therefore, we seek to design heuristic and meta-heuristic algorithms for deriving approximate solution in an acceptable time. In this section, we first proposed to heuristics that are based upon simple dispatching rules. We also develop a tabu search algorithm to provide better solutions. We use the solutions that are derived by the heuristics as the initial solutions for the tabu search algorithm to start out with.

3.1 Heuristic Algorithm H_1

The first heuristic H_1 is based on a greed approach. By Property 1, we arrange the jobs by Johnson's rule. We then dispatch the jobs to the flowshop one by one. Let job i be the first unscheduled job and flowshop k has the minimum completion time among all flowshops, then job i is dispatched to flowshop k. The example given in Figure 3 illustrates the greedy approach. The completion times of the two flowshops are 17 and 16, respectively. Therefore, the first unscheduled job is dispatched to the second flowshop.

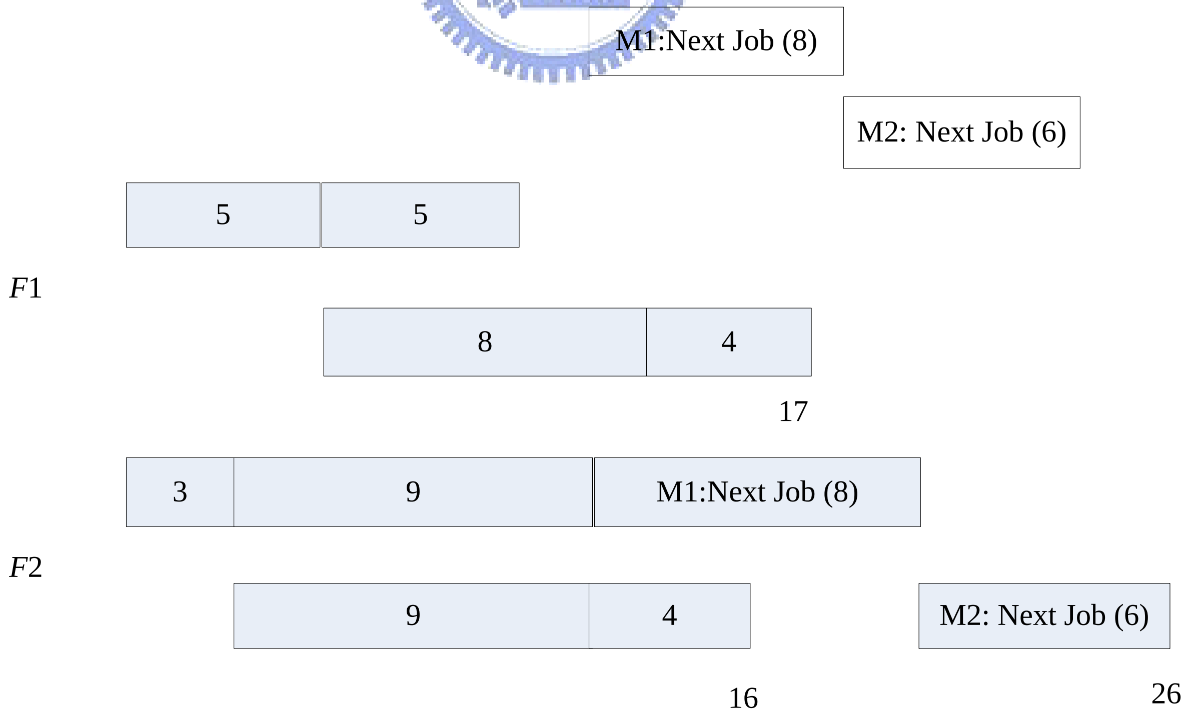

Figure 3: Example of Heuristic H_1

The time complexity of sorting the jobs is $O(n \log n)$. We maintain the job list in a min-heap by Johnson's rule. Another min-heap is used to store the current completion times of all flowshops. Removing the first job from the job list and adjust the heap takes $O(\log n)$ time. The time complexity for a loop of $O(n)$ iterations is thus $O(n \log n)$. We also need to identify the flowshop with the earliest completion time. The required run time for $O(n)$ iterations is $O(n \log l)$. Because $l \leq n$, we therefore have that the overall time complexity of Heuristic H_1 is $O(n \log n)$.

3.2 Heuristic Algorithm H_2

The second heuristic H_2 is modified from H_1. The jobs are initially sorted by Johnson's sequence. The first job retrieved from the list is tentatively assigned to each flowshop. We identify the flowshop having the earliest completion time. Then, the job is permanently assigned to that flowshop and removed from other flowshops. The process is illustrated in Figure 4. Because the next job can produce a shorter completion time in flowshop 1 than in flowshop 2, therefore it is dispatched to flowshop 1.

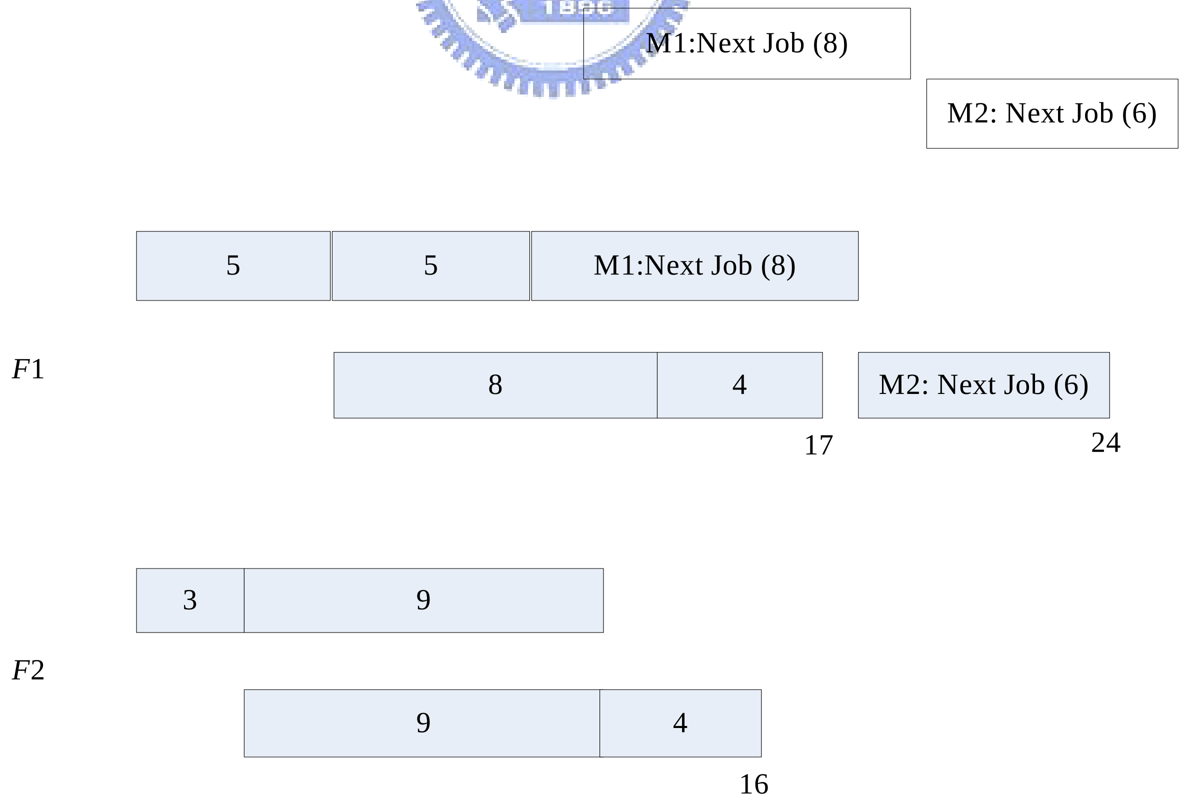

Figure 4: Example of Heuristic H_2

The complexity of Heuristic H_2 is different from that of Heuristic H_1. It is due to the fact that we need to compute the completion time of each flowshop resulted from appending the first job in the job list. The run time of this part is $O(ln)$ for n insertions. Therefore, the overall time complexity is $O(\max\{n \log n, ln\})$.

3.3 Tabu Search

Tabu Search was first proposed by Glover (1989) and has been used to solve numerous combinatorial optimization problems. The theme of tabu search is to develop procedures that can avoid the traps resulted from the local optima which are commonly encountered in the deployment of the steepest gradient approach. Tabu search encourage broad probing through the solution space and utilize a tabu mechanism to prevent cycling.

Applied to the permutation parallel-flowshop scheduling problem, our tabu search algorithm starts from some initial solution and move successively among neighboring solutions. In each iteration, a move is made to the best solution in the neighborhood of the current solution which may not be better than the current one. When the best solution in the neighborhood is in the area which we can control, even it cannot improve current solution we still choosing it. The purpose of such a relaxation is to allow the search proceed towards unexplored regions of the solution space. To prevent the cycling problem a tabu list is used. Tabu list is a special sort of memory mechanism, based on dependencies such as reflected in measures of recency and frequency. A tabu list consists of attributes of the latest moves. The size of the list can be fixed or variable along the course of explorations. Generally, tabu search consists of the following elements:

- Generation of initial solutions;
- Mechanism for generating some neighborhood of the current solution;
- Tabu list;
- Stopping criteria.

For a more complete description of tabu search, the interested reader is referred to the papers of Glover (1989, 1990 and 1997). In the following, we introduce the design of a tabu search algorithm for solving the parallel-flowshop problem. The flowchart is presented in Figure 5.

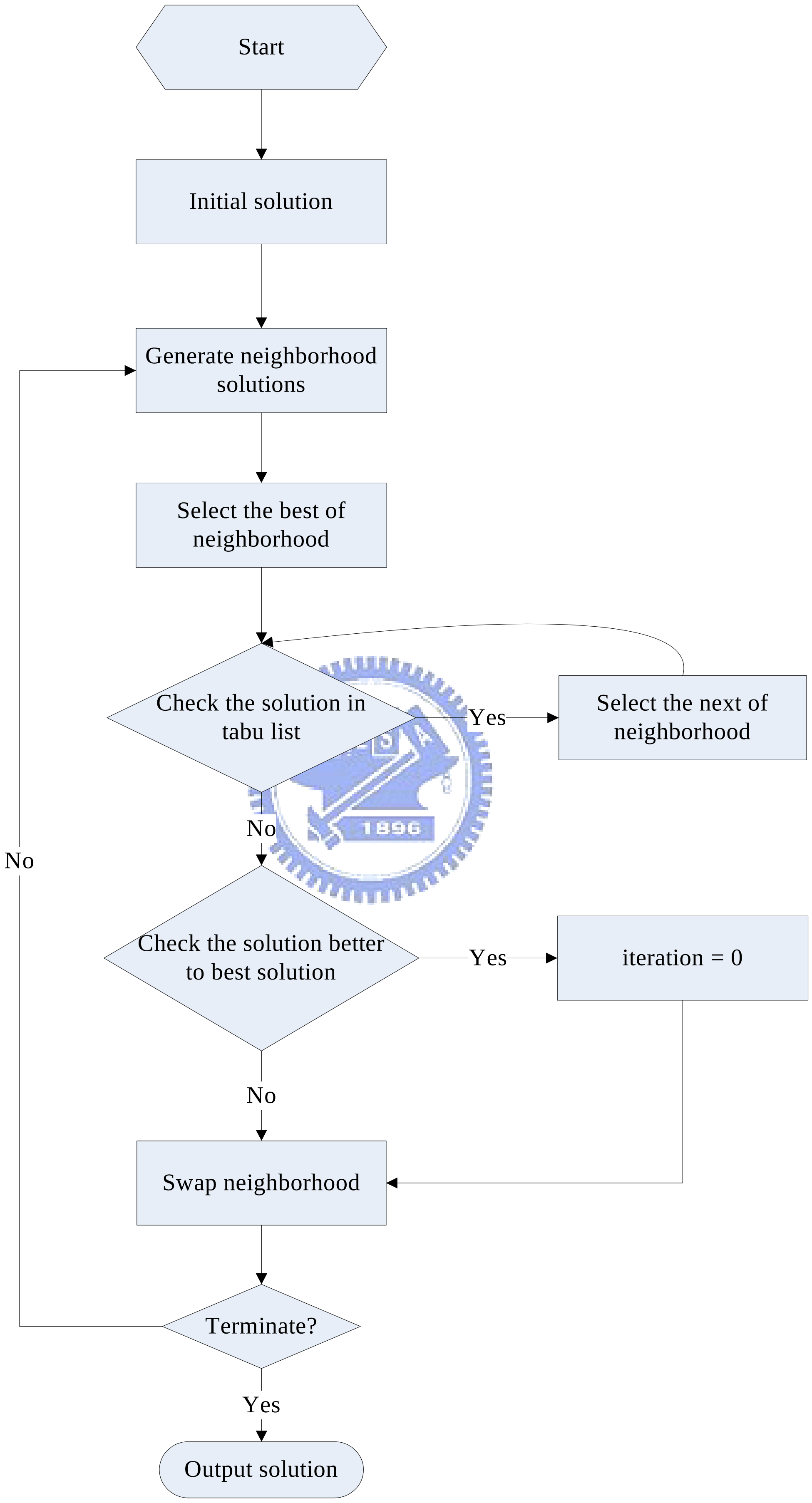
Start
Initial solution
Generate neighborhood solutions
Select the best of neighborhood
Check the solution in tabu list
Yes
Select the next of neighborhood
No
Check the solution better to best solution
Yes
iteration = 0
No
Swap neighborhood
Terminate?
No
Yes
Output solution

Figure 5: Flow chart of Tabu search

3.3.1 Initial solutions

To get an initial starting solution, we considered two heuristics: H_1 and H_2. These two methods were complete introduced before.

3.3.2 Neighborhood structure

We adopt a swapping scheme to construct the neighborhood structure of the tabu search. Given a solution *s*, let *N*(*s*) denote the set of all solutions which can be obtained from *s* using a swapping scheme. The scheme used in our development consists of two parts:

(1) Neighborhood structure one: Randomly select two flowshops *f* and *f'* and a position *i* in the two list of the two flowshops. The range of position *i* is the minimum number of jobs in *f* and *f'*. The scope of random position *i* will lie between the first and final jobs but not include them. We swap the jobs at position *i*. By PROPERTY 1, the positions of the swapped jobs in the two flowshops need to be adjusted so as to meet Johnson's rule. (Figure 6)

(2) Neighborhood structure two: Assume flowshop *f* has the largest complete time. We consider flowshop *f* and each individual flowshop *f'*. And selecting the *f* 's and *f'* 's first job and final job at the same time. Swap the first jobs of *f* and *f'*. Swap the last jobs of *f* and *f'*. (Figure 7)

Neighborhood *N*(*s*) consists of 20 neighbors generated by structure 1 and all neighbor generated by structure 2. And the size of tabu list is 7.

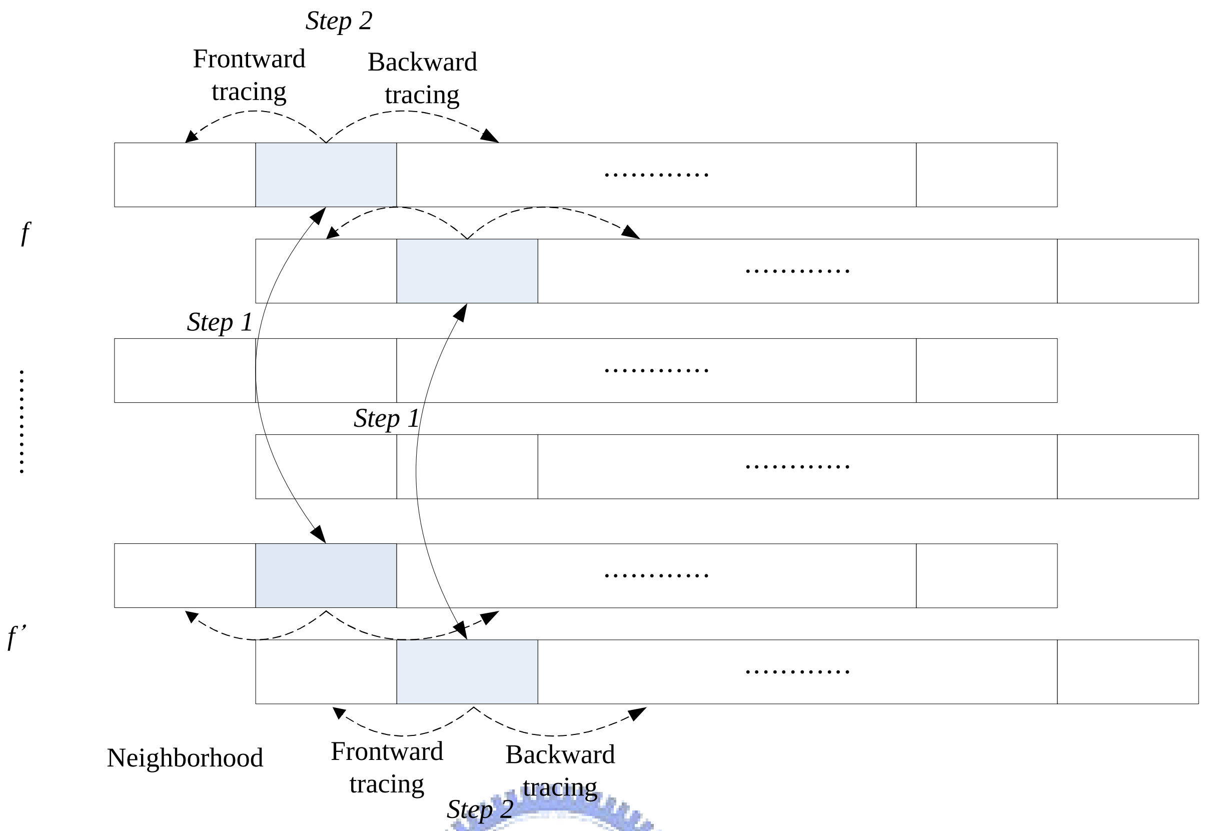

Figure 6: Neighborhood structure 1

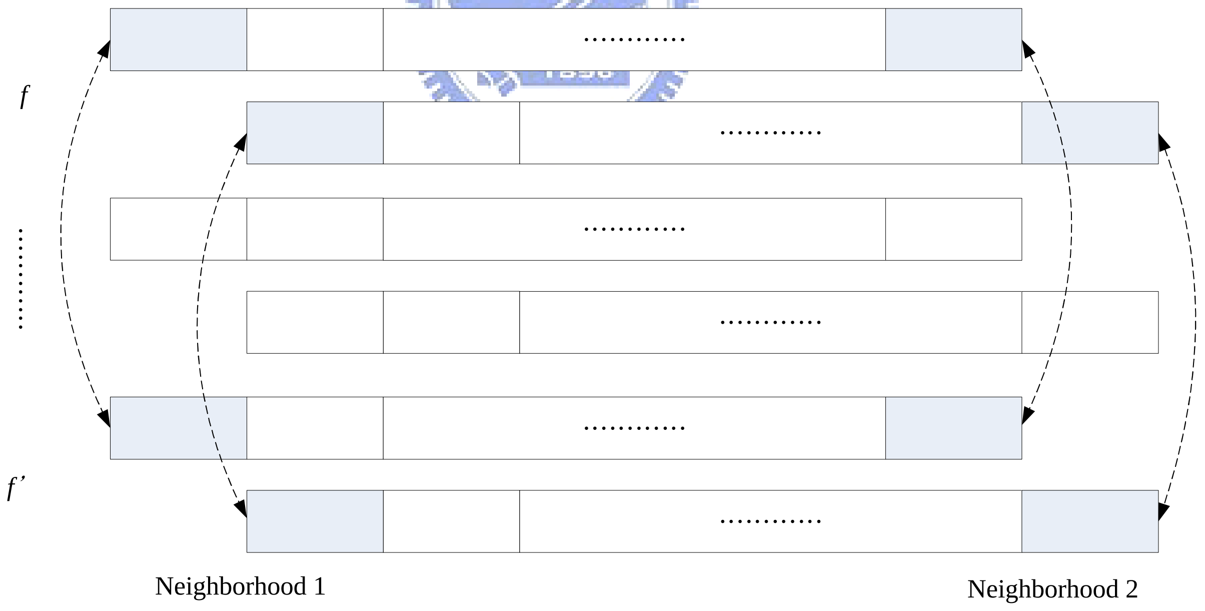

Figure 7: Neighborhood structure 2

3.3.3 Selection of Neighbors

We select the best solution *N*(*s*) and the best one of *N*(*s*), say *s*'. If solution of *s*' is better than *s* and *s*' is not in the tabu list then we move to solution *s*' and use it as the new current solution. If the move from *s* to *s*' happens to be in the tabu list, then we discard *s*' and select the next one from *N*(*s*). To avoid early convergence and local optimal traps, if *s*’ is worse than *s* with a deviation of less than 30%, then we still move on to *s*’ to have a better chance to visit other portions of the solution space.

3.3.4 Stopping criteria

The tabu search approach can be stopped when it conforms to some specified conditions, such as iteration schema and the number of calls schema. In other words, it can be stopped by a maximum number of iterations or a maximum number of calls. In our design, the tabu search algorithm will terminate when the incumbent value is not improved with 500 consecutive iterations.

Chapter 4 Computational Experiments

In this Chapter, we present the computational experiments designed to study the performance of the algorithms proposed in Chapter 3. In the experiment, we use a personal computer with an Intel Pentium(R) 4 CPU (3.4GHz) with RAM 504MB running Microsoft Windows XP professional. The programs are coded in C++. The processing times of all job operations are randomly drawn from uniform interval [1, 100]. In the experiment setting, the number of flowshops, l, is two, three, five and ten. The test instances include n = 50, 100, 200, 300, 400 and 500 jobs. For each combination of l and n, 100 independent data sets were generated and tested.

The results of the experiments are shown in Tables 2, 3, 4 and 5, each of which is presented for a different number of flowshops. In the tables, we include lower bound values and the objective values produced by two heuristics and two tabu search algorithms. The cited values are the average over each 100 instances. We also present the four average deviations. For example, for n = 50 the deviation of H_1 is calculated by

$$\frac{Z_{H_1} - LB}{LB} \times 100\%,$$

where LB the lower bound value and Z_{H1} is the solution value given by heuristic H_1. The value of LB is determined in the following way. First, we sequence all jobs by Johnson's rule as in a single two-machine flowshop and determine its makespan $C_{\max}$. Then, we divide the makespan by the number of flowshops, l, and take the least integer greater than or equal to the derived value, i.e. $LB = \lceil C_{\max}/l \rceil$. The column entitled "# of the best" contains the number of instances for which the corresponding algorithm produces the best solution among the four algorithms.

First, we observe the trend of deviations. As we can expect, meta-heuristics outperform simple heuristic rules in the aspect of deviations and number of the best solutions. We compare two simple heuristic algorithms and two tabu search algorithms. For example, the deviations produced by H_1 are 1.16405% and 0.09012% for 50-job instances and 500-job instances, respectively. The reduction is very significant and applied to all algorithms. When there are only two flowshops, no significant difference exists between H_1 and H_2. However, the superiority of algorithm H_2 over algorithm H_1 is clear when there are ten flowshops. The comparison can be visualized by the bar charts shown in Figure 8 and Figure 9. For ten flowshops, algorithm H_2 clearly dominates algorithm H_1. The same observation can be observed for H_2+Tabu_Search and H_1+Tabu_Search.

We discuss the deviations made by the four algorithms in terms of the problem scales. Within each table, we can easily find that the deviations of each individual algorithm decrease rapidly when the problem scale becomes large. Figure 10 shows the curves of deviations of four algorithms given three flowshops. Therefore, our algorithms provide better solutions when there are more jobs to process. Across the tables, we look into the effects of the number of flowshops, l. The results show that the deviations deteriorate when there are more flowshops. The two trends can be attributed to the following reasoning. In flowshop scheduling, if there are more jobs, then we consequently have more jobs whose machine-one operations are shorter than machine-two operations and thus created scheduling buffers for the remaining jobs. Therefore, if there are more jobs, then it is easier to construct better, even optimal, schedules. Therefore, the deviations decrease when the number of jobs increases. On the other hand, given a fixed number of jobs, if the number of flowshops increases, then the number of jobs distributed to each flowshop will decrease and thus reaching good schedules becomes more difficult and the deviations become larger.

Another observation we are interested in is the “# of the best” values of the heuristics provided different numbers of flowshops. Through analysis of the “# of the best” values across the tables, we find that the corresponding values of heuristic H_1 decrease when the number of flowshops increases. For example, the average values of “# of the best” produced by H_1 are 72 and 22 for 2-flowshop instances and for 10-flowshop instances. The values of heuristic H_2 also decrease as the number of flowshops increases.(Figure 11) But the change is not sharp. The observations suggest that when there are more flowshops available for processing the given jobs, heuristic H_2 will outperforms heuristic H_1 for most test cases.

Table 2: Computational results of two flowshops

n	Lower Bound	H_1			H_2		
		Value	Deviation (%)	# of the best	Value	Deviation (%)	# of the best
50	1312.66	1327.94	1.16405	66	1324.25	0.88294	85
100	2608.71	2624.15	0.59186	68	2619.78	0.42435	77
200	5157.58	5167.59	0.19408	74	5165.95	0.16229	78
300	7706.19	7716.23	0.13029	77	7716.35	0.13184	70
400	10253.8	10266.7	0.12531	73	10262.3	0.08290	76
500	12805.4	12816.9	0.09012	75	12813.4	0.06247	74

n	H_1+Tabu Search			H_2+Tabu Search		
	Value	Deviation (%)	# of the best	Value	Deviation (%)	# of the best
50	1315.73	0.23388	91	1315.67	0.22931	96
100	2610.43	0.06593	95	2610.40	0.06478	97
200	5158.88	0.02521	97	5158.89	0.02540	97
300	7707.31	0.01453	99	7707.33	0.01479	99
400	10255.0	0.01083	98	10254.9	0.01063	100
500	12806.4	0.00804	99	12806.4	0.00797	100

Table 3: Computational results of three flowshops

n	Lower Bound	H_1			H_2		
		Value	Deviation (%)	# of the best	Value	Deviation (%)	# of the best
50	882.05	907.54	2.88986	53	899.98	2.03276	82
100	1730.12	1751.43	1.23171	59	1745.96	0.91554	70
200	3433.85	3453.20	0.56351	61	3448.95	0.43974	72
300	5139.59	5161.39	0.42416	53	5154.31	0.28640	76
400	6846.08	6871.08	0.36517	56	6862.24	0.23605	73
500	8546.80	8568.00	0.24805	56	8561.40	0.17082	80

n	H_1+Tabu Search			H_2+Tabu Search		
	Value	Deviation (%)	# of the best	Value	Deviation (%)	# of the best
50	889.11	0.80041	58	888.72	0.75619	72
100	1734.34	0.24391	66	1733.93	0.22022	80
200	3436.49	0.07688	77	3436.42	0.07484	79
300	5142.02	0.04728	79	5141.97	0.04631	87
400	6848.18	0.03067	83	6848.11	0.02965	87
500	8548.62	0.02129	84	8548.64	0.02153	86

Table 4: Computational results of five flowshops

n	Lower Bound	H_1			H_2		
		Value	Deviation (%)	# of the best	Value	Deviation (%)	# of the best
50	529.48	565.61	6.82368	49	558.83	5.54317	67
100	1033.21	1066.76	3.24716	46	1059.52	2.54643	69
200	2076.48	2110.08	1.61812	44	2097.11	0.99351	73
300	3090.44	3126.55	1.16844	43	3112.32	0.70799	68
400	4093.27	4125.72	0.79277	39	4111.59	0.44756	78
500	5120.89	5154.57	0.65770	45	5141.27	0.39798	78

n	H_1+Tabu Search			H_2+Tabu Search		
	Value	Deviation (%)	# of the best	Value	Deviation (%)	# of the best
50	545.77	3.07660	51	544.70	2.87452	71
100	1045.78	1.21660	52	1044.44	1.08690	67
200	2085.74	0.44595	50	2083.85	0.35493	75
300	3099.28	0.28604	44	3097.03	0.21324	76
400	4100.46	0.17565	55	4099.71	0.15733	72
500	5126.99	0.11912	57	5126.15	0.10272	72

Table 5: Computational results of ten flowshops

n	Lower Bound	H_1			H_2		
		Value	Deviation (%)	# of the best	Value	Deviation (%)	# of the best
50	264.90	321.12	21.2231	16	305.45	15.3077	90
100	525.54	572.85	9.00217	21	557.05	5.99574	85
200	1021.90	1070.89	4.79401	27	1055.95	3.33203	73
300	1538.18	1586.34	3.13097	18	1566.23	1.82358	85
400	2053.00	2102.48	2.41013	25	2081.95	1.41013	80
500	2552.55	2604.38	2.03052	27	2583.34	1.20624	79

n	H_1+Tabu Search			H_2+Tabu Search		
	Value	Deviation (%)	# of the best	Value	Deviation (%)	# of the best
50	298.16	12.5557	28	294.55	11.1929	85
100	554.64	5.53716	29	548.39	4.34791	75
200	1049.26	2.67737	31	1043.91	2.15383	74
300	1562.93	1.60904	28	1555.11	1.10065	78
400	2076.74	1.15636	22	2068.52	0.75597	85
500	2576.20	0.92652	29	2567.85	0.59940	76

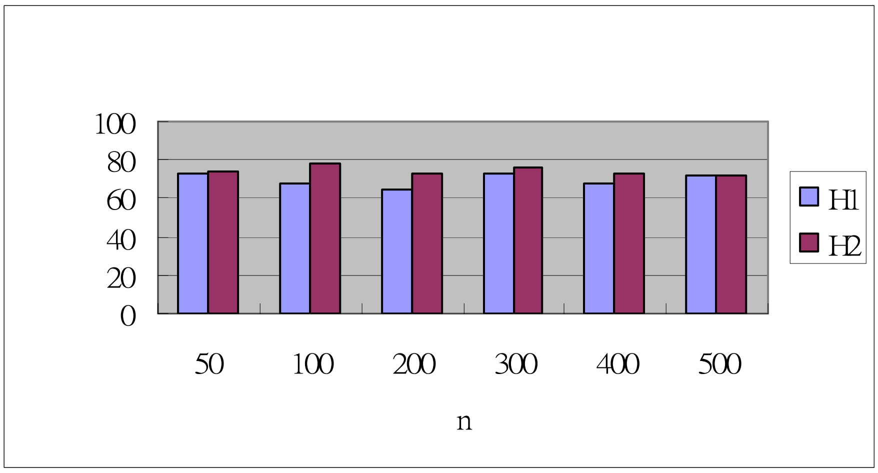

Figure 8: # of the best for two flowshops

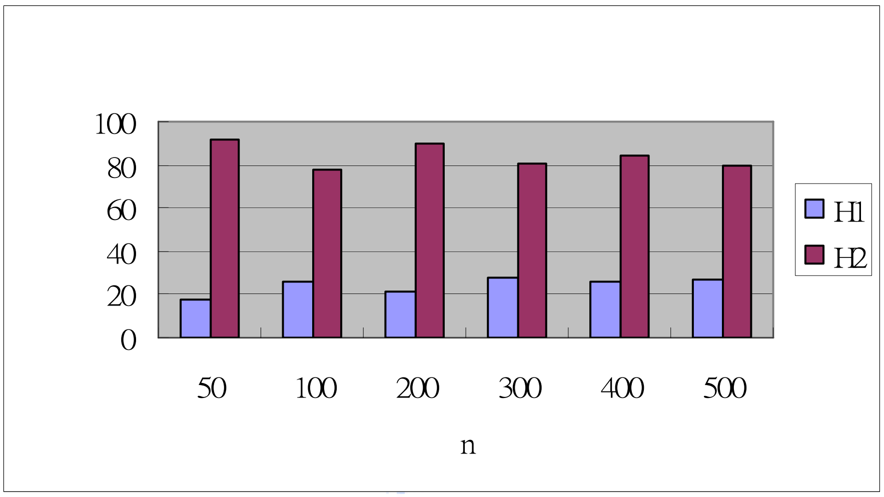

Figure 9: # of the best for ten flowshops

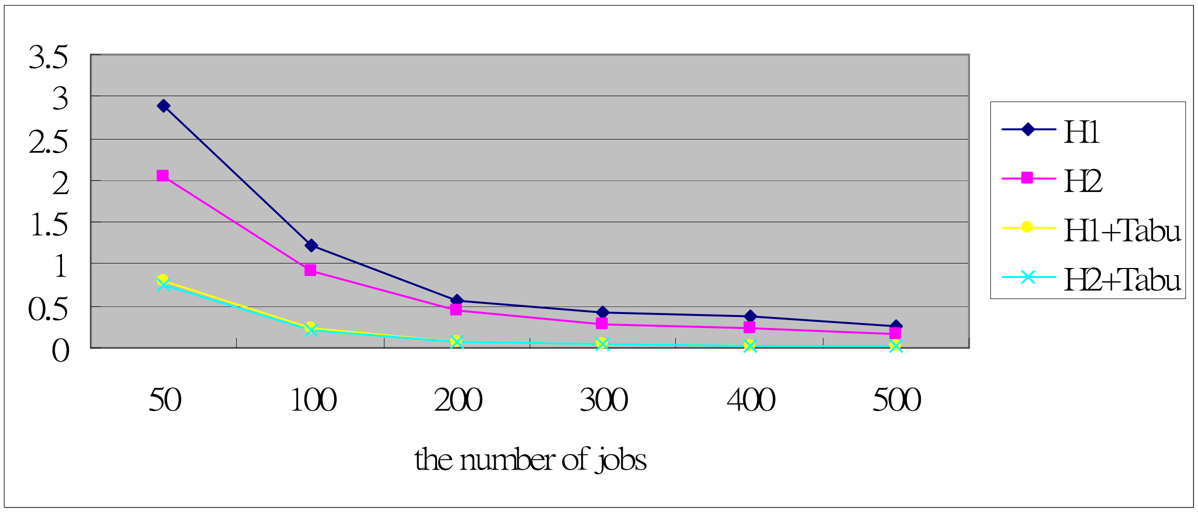

Figure 10: Deviations of three flowshops

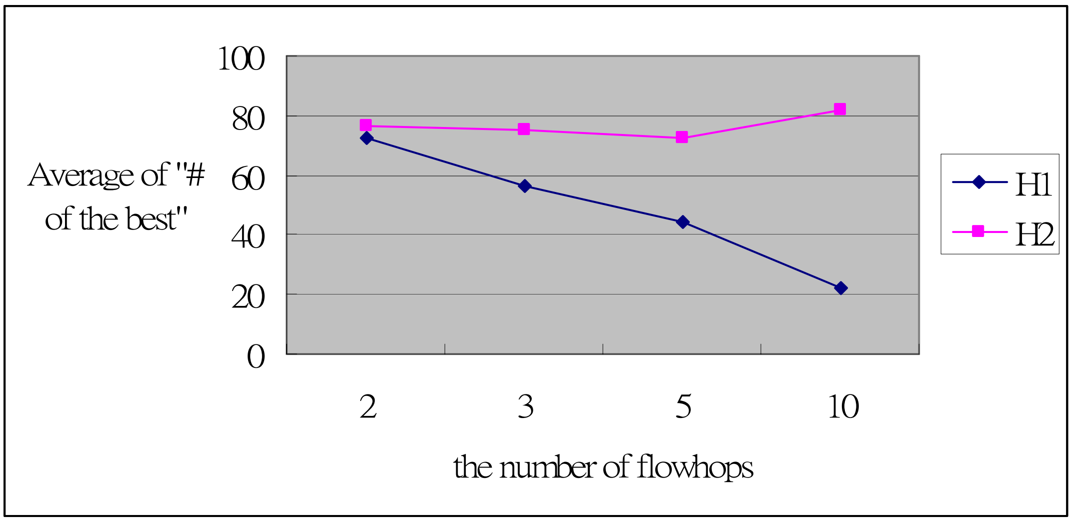

Figure 11: average of “# of the best”

Chapter 5 Conclusions

The thesis considers a manufacturing model consisting of independent and identical flowshops. The objective is to minimize the makespan. The problem is strongly NP-hard because it generalizes both the general flowshop problem and the parallel-machine problem. We discussed an optimality property of this scheduling problem and designed a dynamic programming algorithm that runs in pseudo-polynomial time when the number of flowshops is a constant. Two heuristics based on Johnson’s rule were proposed to derive approximate solutions in an acceptable time. And, we analyze the approximation of the two heuristics. Then, the two heuristics were embedded into tabu search algorithms for improving the solution quality. Computational experiments were designed and conducted to test the effectiveness of the proposed algorithms. Numerical results show that the heuristics and tabu search algorithms can provide solutions close to lower bound values, in other words, the effectiveness of the algorithms are very impressive.

The multiple flowshops model is very close to realistic situations in a factor or in a supply chain. Therefore, it suggests room of future research. A potential topic for future research is to study the general model in which each flowshop has m stages for $m > 2$. A challenging subject is to develop heuristic algorithms and theoretically analyze their performance ratios.

References

[1] Adiri, I., and Pohoryles, D. (1982), “Flowshop/No-idle or no-wait scheduling to minimize the sum of completion time”, *Naval Research Logistics Quarterly* 29, pp. 495-504.

[2] Ahmadi, R.H., and Bagchi, U. (1990), “Improved lower bounds for minimizing the sum of completion times of n jobs over m machines in a flow shop“, *European Journal of Operational Research* 44, pp. 331-336.

[3] Bansal, S.P. (1977), “Minimizing the sum of completion times of n-jobs over M-machines in a flowshop”, *AIIE Transactions* 9, pp. 306-311.

[4] Brucker, P. (1997), “Scheduling Algorithms”, *Springer*, New York.

[5] Chen, Z.-L. and Powell, W.B.(1999), “Solving Parallel Machine Scheduling Problems by Column Generation”, *INFORMS Journal of Computing*, Vol. 11, pp. 78-94.

[6] Coffman, E.G., Jr., Garey, M.R., and Johnson, D.S.(1978), “An Application of Bin-Packing to Multiprocessor Scheduling”, *SIAM Journal of Computing*, Vol. 7, pp. 177-201.

[7] Friesen, D.K.(1984a), “Tighter Bounds for the Multifit Processor Scheduling Algorithm”, *SIAM Journal of Computing*, Vol. 13, pp. 170-181.

[8] Garvey MR, Johson DS, and Sethi R.(1976), “The Complexity of Flowshop and Jobshop Scheduling”, *Mathematics and Operations Research*, Vol. 1, pp. 117-129.

[9] Glover, F. (1989), “Tabu search-Part I”, *ORSA Journal on Computing*, Vol.1, No.3, pp.190-206.

[10] Glover, F. (1990), “Tabu search-Part II”, *ORSA Journal on Computing*, Vol.2, No.1, pp.4-32.

[11] Glover, F., and Laguna M. (1997), “TABU SEARCH”, *Kluwer Academic Publishers*, M.A.

[12] Graham, R.L.(1966), “Bounds for Certain Multiprocessing Anomalies”, *Bell System Technical Journal*, Vol. 45, pp. 1563-1581.

[13] Graham, R.L.(1969), “Bounds on Multiprocessing Timing Anomalies”, *SIAM Journal of Applied Mathematics*, Vol. 17, pp. 263-269.

[14] Johnson, S.M.(1954), “Optimal Two- and Three-Stage Production Schedules With Setup Times Included”, *Naval Research Logistics Quarterly* 1, pp. 61-68.

[15] Lee, C.-Y. and Massey, J.D.(1988), “Multi-Processor Scheduling: Combining LPT and

MULTIFIT", *Discrete Applied Mathematics*, Vol. 20, pp.233-242.

[16] Monma, C.L. and Rinnooy Kan A.H.G.(1983), "A Concise Survey of Efficiently Solvable Special Cases of the Permutation Flow-Shop Problem", *RAIRO Recherche Operationelle*, Vol. 17, pp. 105-119.

[17] Pinedo M.(1985), "A Note on Stochastic Shop Models in Which Jobs have the Same Processing Requirements on Each Machine", *Management Science*, Vol. 31, pp. 840-845.

[18] Pinedo, M. (2002), "Scheduling Theory, Algorithms, and Systems", *New Jersey*.

[19] Sahni, S.(1976), "Algorithms for Scheduling Independent Tasks", *Journal of the Association of Computing Machinery*, Vol. 23, pp. 116-127.

[20] Shakhlevich, N., Hoogeveen, H., and Pindeo M.(1998), "Minimizing Total Weighted Completion Time in a Proportionate Flow Shop", *Journal of Scheduling*, Vol. 1, pp. 157-168.

[21] Smith, M.L., Panwalkar, S.S., and Dudek, R.A.(1975), "Flow Shop Sequencing with Ordered Processing Time Matrices", *Management Science*, Vol. 21, pp. 544-549.

[22] Smith, M.L., Panwalkar, S.S., and Dudek, R.A.(1976), "Flow Shop Sequencing Problem with Ordered Processing Time Matrices: A General Case", *Naval Research Logistics Quarterly*, Vol. 23, pp. 481-486.

[23] Szwarc, W.(1971), "Elimination Methods in the m × n Sequencing Problem", *Naval Research Logistics Quarterly*, Vol. 18, pp. 295-305.

[24] Szwarc, W.(1973), "Optimal Elimination Methods in the m × n Flow Shop Scheduling Problem", *Operations Research*, Vol. 21, pp. 1250-1259.

[25] Szwarc, W.(1978), "Dominance Conditions for the Three-Machine Flow-Shop Problem", *Operations Research*, Vol. 26, pp. 203-206.

[26] Wagner, H.M.(1959), "An Integer Programming Model for Machine Scheduling", *Naval Research Logistics Quarterly*, Vol. 6, pp. 131-140.