Proceedings of the 7th Asia Pacific Industrial Engineering and Management Systems Conference 2006
17-20 December 2006, Bangkok, Thailand

Sequencing and Tabu Search Heuristics for Hybrid Flow Shops with Unrelated Parallel Machines and Setup Times

Jitti Jungwattanakit†, Manop Reodecha, Paveena Chaovalitwongse
Department of Industrial Engineering, Faculty of Engineering
Chulalongkorn University, Bangkok 10330 Thailand
+662-218-6855, Email: jitti.j@student.chula.ac.th.

Frank Werner
Faculty of Mathematics
Otto-von-Guericke-University, P.O. Box 4120, D-39016 Magdeburg, Germany
Phone: +49-391-6712025, fax: +49-391-6711171, Email: frank.werner@mathematik.uni-magdeburg.de

Abstract. The goal of this paper is to investigate scheduling heuristics to seek the minimum of a positively weighted convex sum of makespan and the number of tardy jobs in a static hybrid flow shop environment where at least one production stage is made up of unrelated parallel machines. In addition, sequence - and machine - dependent setup times are considered. The problem is a combinatorial optimization problem which is too difficult to be solved optimally for large problem sizes, and hence heuristics are used to obtain good solutions in a reasonable time. Some dispatching rules and flow shop makespan heuristics are developed. Then this solution may be improved by fast polynomial heuristic improvement algorithms based on shift moves and pairwise interchanges. In addition, metaheuristic proposed is a tabu search algorithm. Three basic parameters (i.e., number of neighbors, neighborhood structure, and size of tabu list in each iteration) of a tabu search algorithm are briefly discussed in this paper. The performance of the heuristics is compared relative to each other on a set of test problems with up to 50 jobs and 20 stages.

Keywords: Hybrid flow shop scheduling; Unrelated parallel machines; Setup times; Constructive algorithms; Improvement heuristics; Tabu Search algorithm.

1. INTRODUCTION

Industrial scheduling presents a complex decision-making scenario. Operations managers are confronted with the tough task to find an optimal solution from the large number of possible combinations that should be considered. This type of problem is also related to combinatorial optimization and NP-hardness, and consequently the search for efficient methods providing a good feasible solution continues to be a challenge. Once efficient algorithm methods are found, computational tools can be built that will allow managers to make rapid decisions with flexibility and efficiency.

This article has been concerned with heuristics to provide good and quick feasible solutions. They obtain solutions to large problems with limited computational effort. The heuristics concerned in this paper can be classified into two types; constructive (conventional) and iterative (modern) heuristic algorithms.

In a constructive algorithm, single or several solutions are generated, but the only best one is chosen as the final solution. In this paper, several simple dispatching rules and flow shop heuristics are adapted to find a solution for the problem. Additionally, we investigate how to improve the quality of the solution by using several fast polynomial improvement algorithms.

The interest in iterative algorithms is due to the difficulty of solving real large-size problems by using an exact algorithm, while such metaheuristic algorithms can treat large complex problems and for this reason, they have got a considerable research attention over the last decades. This study will limit to one of the popular iterative algorithms known as a Tabu Search (TS) algorithm. It is originally proposed by Glover (1986). The TS algorithm is among the most cited and used metaheuristics for the combinatorial problems (Blum and Roli 2003). It has been successfully applied in a lot of different areas: scheduling, transportation, telecommunications, layout and circuit design, graphs, expert systems, and so on (see Glover and Laguna 1993 for a survey).

†: Corresponding Author

Hence, in this paper TS-based algorithms will be used to solve the problem of scheduling a given set of n jobs at k stages on m unrelated parallel machines. Such a problem occurs in real world problems such as e.g. in the textile industry.

A textile manufacturer supervises workers who make products that contain fibers, such as clothing, tires, and yarn. Whatever the industry, the task of a textile manufacturer is the same: to convert raw products into usable goods. Typically, a textile production unit can hardly fit in any classical scheduling model. Instead, such a production unit is characterized by a multi-stage manufacturing process with multiple production units per stage (i.e., parallel machines), which makes production management quite complex. This combined model is referred to as the hybrid flow shop (HFS) or flexible flow shop (FFS) problem as shown in Figure 1. It can be noted that this problem is also known as the flow shop problem because the process follows a flow shop characteristic, but there are some processing stages having parallel machines to increase the overall capacities, to balance the capacities of the stages, or either to eliminate or reduce the impact of bottleneck stages on the overall shop floor capacities. Most textile companies are ageing while the technology changes rapidly. It is common to find newer or more modern machines running side by side with older and less efficient machines. Hence, these companies own machines of different ages, which may perform the same operations as the newer ones, but would generally require a longer operating time for the same operation.

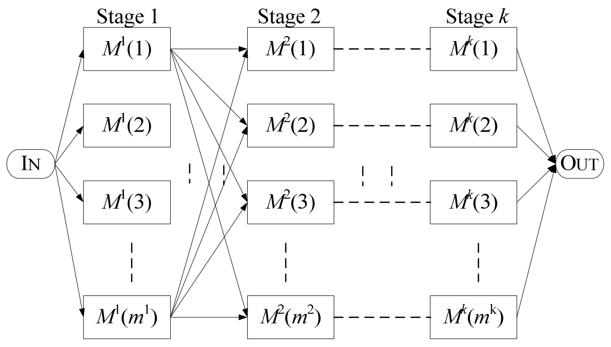

Figure 1: A general schematic of the hybrid flow shop.

Sequence-dependent setup times and costs incur when machines often have to be reconfigured or cleaned between jobs. This process is known as a changeover or setup. If the length of the setup depends on the job just completed and on the one about to be started, then the setup times are sequence-dependent. For instance, in the weaving phase the setup times are depending on the types of clothes being processed in sequence. Another example is dying operations which often require setups. Every time a new color is used, the painting devices must be cleaned. The cleanup time often depends on the color just used as well as the color about to be used. Such a resection is called a sequence-dependent setup time.

For the past three decades, the hybrid flow shop scheduling problem has attracted many researchers. Numerous research articles have been published on this topic (see e.g. Wang 2005). There are two main reasons for this, among many others. Firstly, a hybrid flow shop environment is a category of machine scheduling problems which is difficult to solve (Garey and Johnson 1979; Gupta 1971). Thus, it is unlikely that polynomial time algorithms exist for the exact solution of the general problem. Secondly, this type of machine scheduling problem can find many real-world applications.

Although the hybrid flow shop problem has been widely studied in the literature, most of the studies related to hybrid flow shop problems concentrate on problems with identical processors, see for instance, Gupta, Krüger, Lauff, Werner and Sotskov (2002), Alisantoso, Khoo, and Jiang (2003), Lin and Liao (2003), and Wang and Hunsucker (2003). In this paper, however, the hybrid flow shop problem with unrelated non-identical parallel machines and sequence-dependent setup times is considered. This complex problem, which is encountered by many industries, is very difficult to solve.

Consequently, in this paper the hybrid flow shop with unrelated parallel machines like the textile industry will be solved by using some constructive and TS-based algorithms. The rest of this paper is organized as follows: The problem considered in this paper is described in Section 2. Some heuristic algorithms are proposed in Section 3. A TS algorithm is presented in Section 4. Computational results and conclusions are shown in Section 5 and Section 6, respectively.

2. STATEMENT OF THE PROBLEM

The hybrid flow shop system is defined by a set $O = \{1,\ldots, t,\ldots, k\}$ of k processing stages. At each stage t, $t \in O$, there is a set $M^t = \{1,\ldots, i,\ldots, m^t\}$ of m^t unrelated machines. The set $J = \{1,\ldots, j,\ldots, n\}$ of n independent jobs has to be processed on machine of set $M^1,\ldots, M^k$. Each job j, $j \in J$, has its release date $r_j \geq 0$ and a due date $d_j \geq 0$. It has its fixed standard processing time for every stage t, $t \in O$. Owing to the unrelated machines, the processing time p^t_{ij} of job j on machine i at stage t is equal to ps^t_j / v^t_{ij}, where ps^t_j is the standard processing time of job j at stage t, and v^t_{ij} is the relative speed of job j which is processed by the machine i at stage t.

There are processing restrictions of the jobs as follows: (1) jobs are processed without preemptions on any machine; (2) every machine can process only one operation at a time; (3) operations have to be realized sequentially, without overlapping between the stages; (4) job splitting is not permitted.

Setup times considered in this problem are classified into two types, namely a machine-dependent setup time and a sequence-dependent setup time. A setup time of a job is machine-dependent if it depends on the machine to which the job is assigned. It is assumed to occur only when the job is the first job assigned to the machine. ch^t_{ij} denotes the length of the machine-dependent setup time, (or changeover time), of job *j* if job *j* is the first job assigned to machine *i* at stage *t*. A sequence-dependent setup time is considered between successive jobs. A setup time of a job on a machine is sequence-dependent if it depends on the job just completed on that machine. s^t_{lj} denotes the time needed to changeover from job *l* to job *j* at stage *t*, where job *l* is processed directly before job *j* on the same machine. All setup times are known and constant.

The scheduling problem has dual objectives, namely minimizing the makespan and minimizing the number of tardy jobs. Therefore, the objective function to be minimized is

$$\lambda C_{max} + (1 - \lambda)\eta_T \qquad (1)$$

where C_{max} is the makespan, which is equivalent to the completion time of the last job to leave the system, η_T is the total number of tardy jobs in the schedule, and λ is the weight (or relative importance) given to C_{max} and η_T, ($0 \le \lambda \le 1$).

3. HEURISTIC ALGORITHMS

Heuristic algorithms have been developed to provide good and quick solutions. They obtain solutions to large problems with acceptable computational times, but they do not generate optimality and it may be difficult to judge their effectiveness. They can be divided into either constructive or improvement algorithms. The former algorithms build a feasible solution from scratch. The latter algorithms try to improve a previously generated solution by normally using some form of specific problem knowledge. However, the time required for computation is usually larger compared to the constructive algorithms.

3.1 Heuristic Construction of a Schedule

Since the hybrid flow shop scheduling problem is NP-hard, algorithms for finding an optimal solution in polynomial time are unlikely to exist. Thus, heuristic methods are studied to find approximate solutions. Most researchers develop existing heuristics for the classical hybrid flow shop problem with identical machines by using a particular sequencing rule for the first stage. They follow the same scheme, see Santos, Hunsucker, and Deal (1996).

Firstly, a job sequence is determined according to a particular sequencing rule, and we will briefly discuss the modifications for the problem under consideration in the next section. Secondly, jobs are assigned as soon as possible to the machines at every stage using the job sequence determined for the first stage. There are basically two approaches for this subproblem. The first way is that for the other stages, i.e. from stage two to stage k, jobs are ordered according to their completion times at the previous stage. This means that the FIFO (First in First out) rule is used to find the job sequence for the next stage by means of the job sequence of the previous stage. The second way is to sequence the jobs for the other stages by using the same job sequence as for the first stage, called the permutation rule.

Assume now that a job sequence for the first stage has already been determined. Then we have to solve the problem of scheduling *n* jobs on unrelated parallel machines with sequence- and machine-dependent setup times using this given job sequence for the first stage. We apply a greedy algorithm which constructs a schedule for the *n* jobs at a particular stage provided that a certain job sequence for this stage is known (remind that the job sequence for this particular stage is derived either from the FIFO rule or from the permutation rule), where the objective is to minimize the flow time and the idle time of the machines. The idea is to balance evenly the workload in a heuristic way as much as possible.

3.2 Constructive Heuristics

In order to determine the job sequence for the first stage by some heuristics, we remind that the processing and setup times for every job are dependent on the machine and the previous job, respectively. This means that they are not fixed, until an assignment of jobs to machines for the corresponding stage has been done. Thus, for applying an algorithm for fixing the job sequence for stage one, an algorithm for finding the representatives of the machine speeds and the setup times is necessary.

The representatives of machine speed v'^t_{ij} and setup time s'^t_{lj} for stage *t*, t=1,...*k*, use the minimum, maximum and average values of the data. Thus, the representative of the operating time of job *j* at stage *t* is the sum of the processing time ps^t_j/v'^t_{ij} plus the representative of the setup time s'^t_{lj}. Nine combinations of relative speeds and setup times will be used in our algorithms. The job sequence for the first stage is then fixed as the job sequence with the best function value obtained by all combinations of the nine different relative speeds and setup times.

For determining the job sequence for the first stage, we adapt and develop several basic dispatching rules and constructive algorithms for the flow shop makespan scheduling problem. Some of the dispatching rules are related to tardiness-based criteria, whereas others are used mainly for comparison purposes.

The Shortest Processing Time (SPT) rule is a simple dispatching rule, in which the jobs are sequenced in non-decreasing order of the processing times, whereas the Longest Processing Time (LPT) rule orders the jobs in non-increasing order of their processing times. The Earliest Release Date first (ERD) rule is equivalent to the well-known first-in-first-out (FIFO) rule. The Earliest Due Date first (EDD) rule schedules the jobs according to non-decreasing due dates of the jobs. The Minimum Slack Time first (MST) rule is a variation of the EDD rule. This rule concerns the remaining slack of each job, defined as its due date minus the processing time required to process it. The Slack time per Processing time (S/P) is similar to the MST rule, but its slack time is divided by the processing time required as well (Pinedo and Chao 1999).

The hybrid SPT and EDD (HSE) rule is developed to combine both SPT and EDD rules. Firstly, consider the processing times of each job and determine the relative processing time compared to the maximum processing time required. Secondly, determine the relative due date compared to the maximum due date. Next, calculate the priority value of each job by using the weight (or relative importance) given to C_{max} and η_T for the relative processing time and relative due date.

Palmer's heuristic (1965) is a makespan heuristic denoted by PAL in an effort to use Johnson's rule by proposing a *slope order index* to sequence the jobs on the machines based on the processing times. The idea is to give priority to jobs that have a tendency of progressing from short times to long times as they move through the stages. Campbell, Dudek, and Smith (1970) develop one of the most significant heuristic methods for the makespan problem known as CDS algorithm. Its strength lies in two properties: (1) it uses Johnson's rule in a heuristic fashion, and (2) it generally creates several schedules from which a "best" schedule can be chosen. In so doing, $k – 1$ sub-problems are created and Johnson's rule is applied to each of the sub-problems. Thus, $k – 1$ sequences are generated. Since Johnson's algorithm is a two-stage algorithm, a k-stage problem must be collapsed into a two-stage problem.

Gupta (1971) provides an algorithm denoted by GUP, in a similar manner as algorithm PAL by using a different slope index and scheduling the jobs according to the slope order. Dannenbring (1977) develops a method, denoted by DAN, by using Johnson's algorithm as a foundation. Furthermore, the CDS and PAL algorithms are also exhibited. Dannenbring constructs only one two-stage problem, but the processing times for the constructed jobs reflect the behavior of PAL's slope index. Its purpose is to provide good and quick solutions.

Nawaz, Enscore and Ham (1983) develop the probably best constructive heuristic method for the permutation flow shop makespan problem, called the NEH algorithm. It is based on the idea that a job with a high total operating time on the machines should be placed first at an appropriate relative order in the sequence. Thus, jobs are sorted in non-increasing order of their total operating time requirements. The final sequence is built in a constructive way, adding a new job at each step and finding the best partial solution. For example, the NEH algorithm inserts a third job into the previous partial solution of two jobs which gives the best objective function value under consideration. However, the relative position of the two previous job sequence remains fixed. The algorithm repeats the process for the remaining jobs according to the initial ordering of the total operating time requirements.

Again, to apply these algorithms to the hybrid flow shop problem with unrelated parallel machines, the total operating times for calculating the job sequence for the first stage are calculated for the nine combinations of relative speeds of machines and setup times.

3.3 Improvement Heuristics

Unlike constructive algorithms, improvement heuristics start with an already built schedule and try to improve it by some given procedures. Their use is necessary since the constructive algorithms (especially some algorithms that are adapted from pure makespan heuristics and some dispatching rules such as the SPT, and LPT rules) do not consider due dates. In this section, some fast improvement heuristics will be investigated to improve the overall function value by concerning mainly the due date criterion.

The iterative algorithms described in the following and in Section 4 are based on the shift move (SM) and the pairwise interchange (PI) neighborhoods.

The SM neighborhood repositions a chosen job. This means that an arbitrary job π_r at position r is shifted to position i, while leaving all other relative job orders unchanged. If $1 \leq r < i \leq n$, it is called a right shift and yields $\pi' = (\pi_1, \ldots, \pi_{r-1}, \pi_{r+1}, \ldots, \pi_i, \pi_r, \ldots, \pi_n)$. If $1 \leq i < r \leq n$, it is called a left shift and yields $\pi' = (\pi_1, \ldots \pi_r, \pi_i, \ldots, \pi_{r-1}, \pi_{r+1}, \ldots, \pi_n)$. For instance, assume that randomly one solution in the current generation is selected, say [8 9 4 3 1 7 6 5 2], and then randomly a couple of job positions for performing the shift is selected, e.g. positions 2 and 7 (in this case, it is a right shift). The new solution will be [8 4 3 1 7 6 ***9*** 5 2]. However, if positions 7 and 2 are randomly selected (i.e. it is a left shift), the new solution will be [8 ***6***

9 4 3 1 7 5 2]. In the SM neighborhood, the current solution has $(n-1)^2$ neighbors.

The PI neighborhood exchanges a pair of arbitrary jobs π_r, and π_i, where $1 \le i, r \le n$ and $i \ne r$. Such an operation swaps the jobs at positions r and i, which yields $\pi' = [\pi_1, \ldots, \pi_{r-1}, \pi_i, \pi_{r+1}, \ldots, \pi_{i-1}, \pi_r, \pi_{i+1}, \ldots, \pi_n]$. For example, assume that the current solution is [8 9 4 3 1 7 6 5 2], and then randomly the couple of job positions to be exchanged is selected, e.g. positions 1 and 3. Thus, the new solution will be [***4*** 9 ***8*** 3 1 7 6 5 2]. In the PI neighborhood, the current solution has $n \times (n-1)/2$ neighbor*s*.

In order to find a satisfactory solution of the due date problem, we apply fast polynomial heuristics by applying either the shift move (SM) algorithm as an improvement mechanism based on the idea that we will consider the jobs that are tardy and move them left and right or the pairwise interchange (PI) algorithm, where tardy jobs are swapped to different job positions left and right, either to randomly determined two positions (denoted by the number “2”) or to all other positions (denoted by the letter “A”). The best schedule among the generated neighbors is then taken as the result.

4. TABU SEARCH ALGORITHM

A TS algorithm is an iterative improvement approach designed to avoid terminating prematurely at a local optimum for combinatorial optimization problems. Similar to a simulating annealing (SA) algorithm (see e.g. Jungwattanakit, Reodecha, Chaovalitwongse, and Werner 2006a,b), the TS algorithm is based on the idea of exploring the solution space of a problem by moving from one region of the search space to another in order to look for a better solution. However, to escape from a local optimum, the SA algorithm accepts an inferior solution, which may lead to better solutions later by using an acceptance probability. In contrast, the TS algorithm allows the search to move to the best solution among a set of candidate moves as defined by the neighborhood structure, although it can move to a neighbor with an inferior solution. Nevertheless, subsequent iterations may cause the search to move repeatedly back to the same local optimum. In order to prevent cycling back to recently visited solutions, it should be forbidden or declared tabu for a certain number of iterations, called the size (or length) of the list. Its size is a key controllable parameter of the TS algorithm. This is accomplished by keeping the attributes of the forbidden moves in a list, called the tabu list.

Additionally, an aspiration criterion is defined to deal with the case in which a move leading to a new best solution is tabu. If a current tabu move satisfies the aspiration criterion, its tabu status is canceled and it becomes an allowable move. The use of the aspiration criterion allows TS to lift the restrictions and intensify the search into a particular solution region.

4.1 Choice of an Initial Solution

A TS algorithm has been shown to be effective for many combinatorial optimization problems (see Glover and Laguna 1993), and it seems easy to apply such an approach to scheduling problems. To improve the quality of the solution finally obtained, we also investigated the influence of the choice of an appropriate initial solution by using particular constructive algorithms. We used as an initial solution that obtained from the constructive algorithms SPT, LPT, ERD, EDD, MST, S/P, HSE, PAL, CDS, GUP, DAN and NEH, as well as the other fast improvement (SM, PI) heuristics, respectively.

5. COMPUTATIONAL RESULTS

Firstly, the overall constructive algorithms and different fast improvement heuristics are studied. The constructive algorithms (denoted by letter “CA”) are the simple dispatching rules such as the SPT, LPT, ERD, EDD, MST, S/P, and HSE rules, and the flow shop makespan heuristics adapted such as the PAL, CDS, GUP, DAN, and NEH rules. Then, we applied the fast polynomial improvement heuristics based on four cases stated above in Section 3.3. They are denoted by 2-SM, A-SM, 2-PI, and A-PI, respectively. We used problems with 10 jobs × 5 stages, 30 jobs × 10 stages, and 50 jobs × 20 stages. For all problem sizes, we tested instances with $\lambda \in \{0, 0.001, 0.005, 0.01\ 0.05, 0.1, 0.5, 1\}$ in the objective function. Ten different instances for each problem size have been run.

An experiment was conducted to test with data such as the standard processing times, relative machine speeds, setup times, release dates and due dates. The standard processing times are generated uniformly from the interval [10,100]. Due to the unrelated machine problem, the relative speeds are distributed uniformly in the interval [0.7,1.3]. The setup times, both sequence- and machine-dependent setup times, are generated uniformly from the interval [0,50], whereas the release dates are generated uniformly from the interval between 0 and half of their total standard processing time mean. The due date of a job is set in a way that it is similar to the approach presented by Rajendaran and Ziegler (2003) and is as follows:

$$d_j = \text{total of mean setup time of a job on all stages} + (n-1)\times(\text{mean processing time of a job on one machine})\times U(0,1) + \sum_{t=1}^{k} ps_j^t + r_j \qquad (2)$$

The results for the constructive algorithms and improvement heuristics are given in Table 1. We give the average (absolute for $\lambda = 0$ resp. percentage for $\lambda > 0$) deviation of a particular algorithm from the best solution in these tests for three problem sizes $n\times k$.

Table 1: Average overall performance of constructive and improvement heuristics.

λ	Problem size	CA	2-SM	A-SM	2-PI	A-PI
0	10×5	3.025[a]	1.525	**1.192**	1.650	1.200
	30×10	7.050	3.933	3.008	4.267	**2.050**
	50×20	9.567	5.717	4.575	5.550	**2.192**
	Sum	19.642	11.175	8.775	11.467	**5.442**
0.001	10×5	78.540[b]	28.530	**19.060**	32.590	21.360
	30×10	88.360	36.950	23.810	36.840	**19.040**
	50×20	35.280	12.600	12.180	9.710	**5.500**
	Sum	202.180	78.080	55.050	79.140	**45.900**
0.005	10×5	41.340	15.070	**9.490**	17.900	11.780
	30×10	40.100	16.200	10.620	17.490	**8.740**
	50×20	19.775	8.748	8.126	8.416	**4.536**
	Sum	101.215	40.018	28.236	43.806	**25.056**
0.01	10×5	29.640	10.860	**6.910**	13.530	8.430
	30×10	27.977	12.313	7.857	14.122	**6.802**
	50×20	15.136	8.373	7.512	8.679	**5.397**
	Sum	72.753	31.546	22.279	36.331	**20.629**
0.05	10×5	17.267	6.292	**4.703**	8.344	5.394
	30×10	16.803	8.225	6.185	9.980	**5.019**
	50×20	9.697	5.697	5.348	6.533	**5.035**
	Sum	43.767	20.214	16.236	24.857	**15.448**
0.1	10×5	15.783	5.520	**4.187**	8.520	4.847
	30×10	14.945	6.759	4.827	8.614	**3.761**
	50×20	9.162	5.255	5.128	5.776	**4.766**
	Sum	39.890	17.534	14.142	22.910	**13.374**
0.5	10×5	15.531	5.675	**4.043**	7.762	4.537
	30×10	14.780	7.244	5.583	8.332	**4.340**
	50×20	8.984	5.269	4.993	6.602	**4.676**
	Sum	39.295	18.188	14.619	22.696	**13.553**
1.0	10×5	15.832	5.213	**4.338**	7.894	4.617
	30×10	14.887	7.070	5.361	9.309	**4.314**
	50×20	8.879	5.340	4.862	6.051	**4.632**
	Sum	39.598	17.623	14.561	23.254	**13.563**

[a] average absolute deviation for $\lambda = 0$, [b] average percentage deviation for λ

From these results it is obvious that the improvement heuristics can improve the quality of constructive algorithms by about 60–70 percent. In addition, we have found that for the problem size 10 jobs × 5 stages the all shift moves are slightly better than the others, whereas the all pairwise interchange -based improvement heuristics are the best algorithm otherwise. However, in general the all pairwise interchange algorithm should be selected as the improvement algorithm. Consequently, in this paper we use in the following only the all pairwise interchange-based improvement heuristics. However, when comparing between the 2-SM and 2-PI algorithms whose CPU time is smaller than both the A-SM and A-PI algorithms, we have found that the 2-SM algorithm certainly became better than the 2-PI algorithm.

Table 2: Average performance of constructive (Group I) algorithms.

λ	Problem size	SPT	LPT	ERD	EDD	MST	S/P	HSE
0	10×5	3.000[a]	3.200	3.500	4.600	4.100	4.100	*2.800*
	30×10	*6.900*	7.900	7.700	7.900	8.400	7.900	7.200
	50×20	8.700	8.200	11.100	15.800	14.600	14.100	*7.900*
	Sum	18.600	19.300	22.300	28.300	27.100	26.100	*17.900*
0.001	10×5	90.920[b]	94.290	*87.880*	102.460	91.560	90.930	90.910
	30×10	89.090	104.510	94.730	90.250	100.510	91.170	*87.730*
	50×20	31.830	34.420	42.570	49.770	45.600	43.960	*30.970*
	Sum	211.840	233.220	225.180	242.480	237.670	226.060	*209.610*
0.005	10×5	45.290	*44.130*	44.710	58.240	52.180	52.520	45.250
	30×10	42.140	45.420	43.800	43.640	46.770	41.540	*41.360*
	50×20	18.812	*18.338*	23.140	28.685	26.281	25.233	18.798
	Sum	106.242	107.888	111.650	130.565	125.231	119.293	*105.408*
0.01	10×5	33.300	*30.430*	31.040	41.170	36.990	37.630	33.270
	30×10	30.633	30.954	30.780	31.752	33.282	*28.880*	29.870
	50×20	14.895	*14.199*	17.734	21.330	19.445	18.625	14.996
	Sum	78.828	*75.583*	79.554	94.252	89.717	85.135	78.136
0.05	10×5	22.154	*16.778*	17.176	21.889	20.413	19.662	21.591
	30×10	20.477	17.413	19.306	21.227	21.110	*16.986*	19.431
	50×20	10.406	*9.721*	11.872	12.748	11.476	10.838	10.425
	Sum	53.037	*43.912*	48.354	55.864	52.999	47.486	51.447
0.1	10×5	21.084	*15.177*	15.656	19.457	18.163	17.181	20.257
	30×10	18.691	15.309	17.482	19.453	19.007	*15.058*	17.658
	50×20	10.029	*9.384*	11.335	11.772	10.554	9.935	10.053
	Sum	49.804	*39.870*	44.473	50.682	47.724	42.174	47.968
0.5	10×5	21.203	*14.852*	15.456	18.446	17.310	16.114	20.073
	30×10	18.759	15.021	17.524	19.528	18.653	*14.916*	17.669
	50×20	9.985	*9.394*	11.181	11.244	10.068	9.446	10.011
	Sum	49.947	*39.267*	44.161	49.218	46.031	40.476	47.753
1.0	10×5	21.473	*15.061*	15.696	18.567	17.426	16.214	21.473
	30×10	18.793	15.018	17.551	19.567	18.630	*14.923*	18.793
	50×20	9.892	9.308	11.073	11.087	9.918	*9.296*	9.892
	Sum	50.158	*39.387*	44.320	49.221	45.974	40.433	50.158

[a] average absolute deviation for $\lambda = 0$, [b] average percentage deviation for λ

Next, we present the constructive algorithms that are separated into four main groups. The first heuristic group includes the simple dispatching rules such as SPT, LPT, ERD, EDD, MST, S/P, and HSE. The second heuristic group includes the flow shop makespan heuristics adapted such as PAL, CDS, GUP, DAN, and NEH. The third and fourth heuristic groups are generated from the first two groups of heuristics where the solutions are improved by the selected polynomial improvement algorithm based on all pairwise interchange-based improvement heuristics, and they are denoted by the first letter "I" in front of the letters describing the heuristics of the first two groups.

Table 3: Average performance of constructive (Group II) algorithms.

λ	Problem size	PAL	CDS	GUP	DAN	NEH
0	10×5	2.700[a]	2.200	2.800	2.600	***0.700***
	30×10	7.700	6.100	7.300	7.800	*1.800*
	50×20	9.400	6.600	8.600	8.800	*1.000*
	Sum	19.800	14.900	18.700	19.200	***3.500***
0.001	10×5	73.220[b]	61.630	77.800	70.370	***10.470***
	30×10	101.410	77.990	94.510	98.410	*29.980*
	50×20	37.030	27.000	34.620	35.200	*10.320*
	Sum	211.660	166.620	206.930	203.980	*50.770*
0.005	10×5	38.870	31.010	41.540	36.150	***6.250***
	30×10	44.290	34.330	42.650	43.740	*11.500*
	50×20	20.046	15.791	18.867	18.902	*4.411*
	Sum	103.206	81.131	103.057	98.792	***22.161***
0.01	10×5	28.570	22.230	30.560	25.780	***4.710***
	30×10	29.870	23.506	29.744	29.561	*6.887*
	50×20	15.440	12.655	14.646	14.514	***3.160***
	Sum	73.880	58.391	74.950	69.855	***14.757***
0.05	10×5	18.069	12.421	19.019	15.633	***2.399***
	30×10	16.766	13.394	17.207	15.928	***2.386***
	50×20	10.355	8.400	9.898	9.457	***0.765***
	Sum	45.190	34.215	46.124	41.018	***5.550***
0.1	10×5	17.073	11.196	17.585	14.471	***2.097***
	30×10	14.637	11.722	15.071	13.784	***1.470***
	50×20	9.877	8.016	9.523	8.985	***0.479***
	Sum	41.587	30.934	42.179	37.240	***4.046***
0.5	10×5	17.373	11.176	17.221	14.448	***2.700***
	30×10	14.368	11.768	14.794	13.488	***0.869***
	50×20	9.754	7.933	9.489	8.866	***0.436***
	Sum	41.495	30.877	41.504	36.802	*4.005*
1.0	10×5	17.674	11.400	17.418	14.701	***2.885***
	30×10	14.367	11.785	14.780	13.477	***0.964***
	50×20	9.651	7.837	9.399	8.766	***0.428***
	Sum	41.692	31.022	41.597	36.944	***4.277***

[a] average absolute deviation for $\lambda = 0$, [b] average percentage deviation for λ

The results for the constructive and polynomial improvement algorithms are given in Table 2 through Table 5. From these results it can be seen that the improvement algorithms in the fourth heuristic group improved from flow shop makespan heuristics from the second heuristic group (i.e., PAL, CDS, GUP, DAN, and NEH) are better than the dispatching rules in the first heuristic group (i.e., SPT, LPT, ERD, EDD, MST, S/P, and HSE) as well as the third heuristic group improved from them.

Table 4: Average performance of improvement (Group III) algorithms.

λ	Problem size	ISPT	ILPT	IERD	IEDD	IMST	IS/P	IHSE
0	10×5	1.400[a]	1.300	1.200	*1.000*	1.300	1.300	1.100
	30×10	2.100	2.500	2.400	1.500	***1.200***	1.800	2.100
	50×20	*1.300*	2.300	3.000	3.000	4.900	3.300	1.700
	Sum	*4.800*	6.100	6.600	5.500	7.400	6.400	4.900
0.001	10×5	16.220[b]	24.020	*12.550*	13.710	18.940	18.870	23.880
	30×10	20.310	18.910	18.970	***13.200***	15.710	13.910	19.090
	50×20	6.730	5.150	7.690	5.010	4.040	6.960	*2.990*
	Sum	43.260	48.080	39.210	***31.920***	38.690	39.740	45.960
0.005	10×5	10.000	13.410	*8.170*	8.650	11.990	11.740	14.840
	30×10	9.690	8.150	8.610	7.530	7.520	*6.610*	10.300
	50×20	3.924	5.228	5.514	6.051	*3.554*	5.631	3.922
	Sum	23.614	26.788	22.294	*22.231*	23.064	23.981	29.062
0.01	10×5	8.890	9.450	7.030	*6.640*	7.690	9.540	9.080
	30×10	6.373	9.709	5.676	6.753	4.762	***4.076***	9.096
	50×20	*4.699*	6.427	5.749	6.264	6.890	6.583	5.198
	Sum	19.962	25.586	*18.455*	19.657	19.342	20.199	23.374
0.05	10×5	5.476	5.281	*4.900*	6.629	5.643	5.620	7.194
	30×10	4.820	6.313	*2.768*	6.397	5.431	4.893	6.431
	50×20	*4.778*	5.247	5.438	7.221	6.010	6.711	5.028
	Sum	15.074	16.841	*13.106*	20.247	17.084	17.224	18.653
0.1	10×5	*4.546*	5.404	4.787	6.318	5.721	4.877	5.763
	30×10	3.255	4.743	*1.718*	5.523	4.957	5.033	4.619
	50×20	5.169	*4.241*	5.024	6.681	5.831	5.788	5.173
	Sum	12.970	14.388	*11.529*	18.522	16.509	15.698	15.555
0.5	10×5	4.969	4.932	5.195	5.707	6.287	*4.629*	5.414
	30×10	3.929	5.283	*2.404*	6.727	5.135	6.897	4.730
	50×20	5.453	*4.244*	5.090	6.215	5.900	4.725	5.406
	Sum	14.351	14.459	*12.689*	18.649	17.322	16.251	15.550
1.0	10×5	5.018	5.073	5.268	5.741	5.935	*4.840*	5.018
	30×10	4.155	4.838	*2.421*	6.843	5.932	6.940	4.155
	50×20	5.346	*4.147*	5.107	6.079	5.731	4.523	5.346
	Sum	14.519	14.058	*12.796*	18.663	17.598	16.303	14.519

[a] average absolute deviation for $\lambda = 0$, [b] average percentage deviation for λ

Among the simple dispatching rules (heuristic Group I), the SPT, LPT, ERD, and HSE rules are good dispatching rules. However, in general the HSE rule outperforms the other dispatching rules for $\lambda < 0.01$, and the LPT rule is better than the other rules otherwise. Among the adapted flow shop makespan heuristics in heuristic Group II, the NEH algorithm is clearly the best algorithm among all studied constructive heuristics (but in fact, this algorithm takes the convex combination of both criteria into account when selecting partial sequences). The CDS algorithm is certainly the algorithm on the second rank (but it is substantially worse than the NEH algorithm even if the makespan portion in the objective function value is dominant, i.e. for large λ values). These results are similar to the conclusions of Jungwattanakit, Reodecha, Chaovalitwongse, and Werner (2006c) whose experiments compared the results for small problem sizes with the optimal solutions.

Table 5: Average performance of improvement (Group IV) algorithms.

λ	Problem size	IPAL	ICDS	IGUP	IDAN	INEH
0	10×5	1.400[a]	1.300	1.100	1.300	***0.700***
	30×10	2.500	2.300	2.100	2.300	*1.800*
	50×20	1.700	***0.700***	2.000	1.400	1.000
	Sum	5.600	4.300	5.200	5.000	***3.500***
0.001	10×5	35.200[b]	23.980	35.630	22.840	***10.470***
	30×10	*17.750*	18.110	21.450	21.140	29.980
	50×20	8.750	2.640	3.260	***2.460***	10.320
	Sum	61.700	*44.730*	60.340	46.440	50.770
0.005	10×5	13.980	14.060	15.260	12.970	***6.250***
	30×10	9.650	***6.050***	9.020	10.200	11.500
	50×20	5.453	***3.306***	3.734	3.703	4.411
	Sum	29.083	23.416	28.014	26.873	***22.161***
0.01	10×5	8.460	9.810	10.130	9.770	***4.710***
	30×10	7.585	*4.935*	8.988	6.782	6.887
	50×20	6.183	4.934	3.431	5.251	***3.160***
	Sum	22.228	19.679	22.549	21.803	***14.757***
0.05	10×5	6.322	4.229	5.675	5.365	***2.399***
	30×10	5.865	4.227	5.282	5.419	***2.386***
	50×20	5.486	3.139	5.538	5.064	***0.765***
	Sum	17.673	11.595	16.495	15.848	***5.550***
0.1	10×5	5.749	3.752	4.154	4.996	***2.097***
	30×10	4.193	2.241	3.848	3.537	***1.470***
	50×20	5.336	3.126	5.394	4.955	***0.479***
	Sum	15.278	9.119	13.396	13.488	***4.046***
0.5	10×5	4.327	2.790	4.147	3.346	***2.700***
	30×10	4.936	2.812	4.611	3.745	***0.869***
	50×20	5.163	3.450	5.125	4.906	***0.436***
	Sum	14.426	9.052	13.883	11.997	*4.005*
1.0	10×5	4.527	3.195	4.378	3.531	***2.885***
	30×10	4.910	2.405	4.666	3.543	***0.964***
	50×20	5.044	3.808	5.046	4.976	***0.428***
	Sum	14.481	9.408	14.090	12.050	***4.277***

[a] average absolute deviation for $\lambda = 0$, [b] average percentage deviation for λ

Thirdly, we studied the TS algorithm with a random initial solution. The purpose of this study is to determine the favorable TS parameters, i.e., number of neighbors, neighborhood structure, and size of tabu list in each iteration. Given the above three different problem sizes, the following TS parameter values were used in this test.

Number of neighbors	: 10 through 50, in step of 10
Neighborhood structures	: PI and SM
Sizes of tabu list	: 5, 10, 15, and 20

From the preliminary tests, we set the time limit equal to one second for the problems with ten jobs, ten seconds for the problems with 30 jobs, and 30 seconds for the problems with 50 jobs. Again, for all tests we considered instances with λ ∈ {0, 0.001, 0.005, 0.01 0.05, 0.1, 0.5, and 1}. Table 6 through Table 8 present the effect of the number of neighbors, neighborhood structure, and size of tabu list by using the average (absolute resp. relative) deviation from the best value as the performance measure.

Table 6: The effect of the various numbers of neighbors on the performance of the TS algorithm.

λ	Problem size	10	20	30	40	50
0	10×5	0.029[a]	**0.017**	0.033	0.050	0.083
	30×10	0.400	**0.242**	0.313	0.392	0.463
	50×20	**0.050**	0.146	0.346	0.533	0.625
	Sum	0.479	**0.404**	0.692	0.975	1.171
0.001	10×5	0.954[b]	0.989	**0.943**	1.477	3.281
	30×10	7.912	5.236	**4.940**	5.640	6.250
	50×20	0.981	**0.945**	2.040	3.409	4.250
	Sum	9.847	**7.170**	7.923	10.526	13.781
0.005	10×5	1.136	0.648	**0.618**	0.799	1.282
	30×10	6.057	3.942	**3.611**	4.134	4.195
	50×20	1.875	**1.663**	2.623	3.493	4.099
	Sum	9.068	**6.253**	6.852	8.426	9.576
0.01	10×5	0.781	**0.419**	0.474	0.730	1.099
	30×10	5.264	3.653	**3.549**	3.931	4.390
	50×20	2.171	**1.807**	2.783	3.744	4.161
	Sum	8.216	**5.879**	6.806	8.405	9.650
0.05	10×5	0.535	0.176	**0.166**	0.191	0.332
	30×10	4.585	3.727	**3.632**	3.777	4.119
	50×20	2.410	**1.734**	2.793	3.338	3.905
	Sum	7.530	**5.637**	6.591	7.306	8.356
0.1	10×5	0.381	**0.119**	0.158	0.154	0.344
	30×10	4.067	3.542	**3.458**	3.773	3.714
	50×20	2.174	**1.491**	2.313	2.925	3.555
	Sum	6.622	**5.152**	5.929	6.851	7.613
0.5	10×5	0.331	0.164	**0.108**	0.228	0.282
	30×10	3.705	**2.962**	3.168	3.182	3.569
	50×20	2.008	**1.304**	2.098	2.860	3.510
	Sum	6.044	**4.430**	5.374	6.269	7.360
1.0	10×5	0.358	**0.127**	0.152	0.218	0.327
	30×10	3.523	**2.805**	2.877	3.169	3.299
	50×20	2.129	**1.415**	2.321	2.861	3.551
	Sum	6.011	**4.347**	5.351	6.249	7.177

[a] average absolute deviation for $\lambda = 0$, [b] average percentage deviation for λ

From the full factorial experiment, we analyzed our results by means of a multi-factor Analysis of Variance technique using a 5% significant level. We have found that for all TS parameters, there are significant differences.

For the number of neighbors, the 20 and 30 nontabu neighbors are good parameters, but 20 nontabu neighbors are better than the other. It is clear that pairwise interchange (PI) moves are better than shift moves (SM) for $\lambda < 0.005$, whereas for $\lambda = 0.005$ and the problem sizes 10 jobs × 5 stages as well as 50 jobs × 20 stages, there are not statistically significantly differences in both neighborhood structures, but they are statistically significant for the problem size 30 jobs × 10 stages. For the problem size 50 jobs × 20 stages and $\lambda \geq 0.1$, although the average main effect of the PI moves is better than the other, it has been found that there is a statistically significant interaction between the neighborhood structure and the number of neighbors, that is, for 20 nontabu SM neighbors become better than PI moves. Hence, in general the SM should be selected as the neighborhood structure for $\lambda \geq 0.005$. For the size of the tabu list, it is shown that a size of 10 and 15 works best, but the size 10 of the tabu list is recommended.

Table 7: The effect of the various neighborhood structures on the performance of the TS algorithm.

λ	Problem size	PI	SM
0	10×5	**0.033**[a]	0.052
	30×10	**0.337**	0.387
	50×20	**0.163**	0.517
	Sum	**0.533**	0.955
0.001	10×5	**0.911**[b]	2.146
	30×10	**5.923**	6.068
	50×20	**2.050**	2.600
	Sum	**8.884**	10.814
0.005	10×5	**0.826**	0.967
	30×10	4.650	**4.125**
	50×20	**2.635**	2.867
	Sum	8.111	**7.959**
0.01	10×5	**0.667**	0.735
	30×10	4.413	**3.903**
	50×20	**2.759**	3.108
	Sum	7.839	**7.746**
0.05	10×5	0.379	**0.181**
	30×10	4.298	**3.638**
	50×20	2.839	**2.834**
	Sum	7.516	**6.653**
0.1	10×5	0.324	**0.138**
	30×10	4.004	**3.418**
	50×20	**2.407**	2.576
	Sum	6.735	**6.132**
0.5	10×5	0.283	**0.162**
	30×10	3.561	**3.073**
	50×20	**2.310**	2.402
	Sum	6.154	**5.637**
1.0	10×5	0.290	**0.183**
	30×10	3.341	**2.928**
	50×20	**2.345**	2.566
	Sum	5.976	**5.677**

[a] average absolute deviation for $\lambda = 0$, [b] average percentage deviation for λ

Table 8: The effect of the various sizes of tabu list on the performance of the TS algorithm.

λ	Problem size	5	10	15	20
0	10×5	0.057[a]	**0.030**	0.040	0.043
	30×10	0.380	0.367	**0.347**	0.353
	50×20	0.380	0.347	**0.287**	0.347
	Sum	0.817	0.743	**0.673**	0.743
0.001	10×5	2.254[b]	1.152	**0.924**	1.786
	30×10	6.046	5.912	6.412	**5.612**
	50×20	2.511	**2.221**	2.339	2.228
	Sum	10.811	**9.285**	9.675	9.626
0.005	10×5	1.036	**0.707**	0.894	0.949
	30×10	**4.325**	4.354	4.341	4.532
	50×20	2.837	2.746	2.711	**2.710**
	Sum	8.198	**7.807**	7.946	8.191
0.01	10×5	0.855	**0.534**	0.552	0.863
	30×10	**4.097**	4.165	4.238	4.131
	50×20	2.933	3.134	**2.687**	2.979
	Sum	7.885	7.833	**7.477**	7.973
0.05	10×5	0.261	**0.239**	0.257	0.364
	30×10	3.939	4.019	3.989	**3.926**
	50×20	**2.667**	2.905	2.903	2.869
	Sum	**6.867**	7.163	7.149	7.159
0.1	10×5	0.278	**0.150**	0.230	0.267
	30×10	3.684	3.769	**3.682**	3.708
	50×20	2.513	2.444	**2.427**	2.582
	Sum	6.475	6.363	**6.339**	6.557
0.5	10×5	0.219	**0.190**	0.217	0.264
	30×10	3.379	**3.219**	3.375	3.295
	50×20	2.369	2.340	2.431	**2.283**
	Sum	5.967	**5.749**	6.023	5.842
1.0	10×5	0.290	**0.167**	0.207	0.283
	30×10	3.171	3.105	**3.085**	3.177
	50×20	2.434	2.451	2.521	**2.416**
	Sum	5.895	**5.723**	5.813	5.876

[a] average absolute deviation for $\lambda = 0$, [b] average percentage deviation for λ

Finally, we used the recommended TS parameters to test the choice of an initial solution. The letters before TS denote the heuristic rule as an initial solution for the TS algorithm. For example, SPTTS means that the SPT rule is used as an initial solution for the TS algorithm. From these results in Table 9 through Table 12, we have found that there are no statistically significant differences in the different initial solutions for the problem sizes 10 jobs × 5 stages and 30 jobs × 10 stages, but it became statistically significantly different in the problem size 50 jobs × 20 stages especially for λ ≥ 0.05 we have found that algorithms INEHTS and NEHTS are better than the others. Consequently, in general algorithms INEHTS and NEHTS are good choices for the TS algorithm with using a biased initial solution.

Table 9: Comparison of the TS algorithm with different initial solutions (Group I).

λ	Problem size	SPTTS	LPTTS	ERDTS	EDDTS	MSTTS	S/PTS	HSETS
0	10×5	0.020[a]	***0.000***	0.020	***0.000***	0.020	***0.000***	0.040
	30×10	0.220	*0.220*	0.260	0.240	0.260	0.240	0.220
	50×20	0.060	0.060	0.100	0.060	0.040	0.060	*0.020*
	Sum	0.300	0.280	0.380	0.300	0.320	0.300	*0.280*
0.001	10×5	***0.014***[b]	0.090	0.071	0.589	0.023	0.064	1.027
	30×10	*3.674*	4.649	4.377	4.152	4.544	4.928	3.954
	50×20	0.667	0.938	1.415	0.937	0.634	0.614	*0.512*
	Sum	*4.355*	5.677	5.863	5.678	5.201	5.606	5.493
0.005	10×5	0.306	0.984	0.912	*0.280*	1.067	0.770	0.421
	30×10	2.892	3.177	2.847	3.034	3.484	*2.735*	3.107
	50×20	1.363	1.223	1.491	1.285	*0.838*	1.044	0.992
	Sum	4.561	5.384	5.250	4.599	5.389	4.549	*4.520*
0.01	10×5	0.598	0.551	0.356	***0.206***	0.377	0.373	0.323
	30×10	3.063	3.562	**2.373**	3.003	3.189	2.577	3.237
	50×20	1.068	0.851	1.180	1.451	*0.690*	0.944	1.113
	Sum	4.729	4.964	3.909	4.660	4.256	**3.894**	4.673
0.05	10×5	0.039	0.033	0.029	0.049	0.059	*0.026*	0.095
	30×10	3.252	3.513	*2.357*	3.499	2.677	3.117	3.388
	50×20	1.064	0.959	1.146	1.031	*0.764*	1.040	1.206
	Sum	4.355	4.505	3.532	4.579	*3.500*	4.183	4.689
0.1	10×5	0.039	0.057	0.009	0.021	0.030	***0.008***	0.020
	30×10	2.698	2.633	*1.947*	2.440	2.231	2.477	2.769
	50×20	*0.947*	1.183	1.166	1.470	0.956	0.991	1.262
	Sum	3.685	3.872	*3.122*	3.931	3.218	3.476	4.052
0.5	10×5	0.061	0.015	0.055	0.039	0.031	0.019	0.107
	30×10	2.307	2.396	*1.872*	2.421	2.152	2.471	2.180
	50×20	0.938	1.047	1.228	1.336	1.027	*0.936*	1.041
	Sum	3.306	3.458	*3.155*	3.796	3.210	3.426	3.327
1.0	10×5	0.096	0.059	0.054	0.067	*0.011*	0.048	0.096
	30×10	2.264	2.517	*1.808*	2.342	2.560	1.925	2.266
	50×20	0.878	0.981	1.107	1.300	0.931	0.986	*0.878*
	Sum	3.237	3.557	2.969	3.709	3.502	*2.960*	3.239

[a] average absolute deviation for $\lambda = 0$, [b] average percentage deviation for λ

6. CONCLUSIONS

In this paper, some constructive algorithms have first been investigated for minimizing a convex combination of makespan and the number of tardy jobs for the hybrid flow shop problem with unrelated parallel machines and setup times, which is often occurring in the real world problems. All algorithms are based on the list scheduling principle by developing job sequences for the first stage and assigning and sequencing the remaining stages by both the permutation and FIFO approaches.

Table 10: Comparison of the TS algorithm with different initial solutions (Group II).

λ	Problem size	PALTS	CDSTS	GUPTS	DANTS	NEHTS
0	10×5	***0.000***	***0.000***	0.020	0.020	***0.000***
	30×10	0.260	0.340	0.220	0.280	*0.180*
	50×20	0.120	0.020	0.060	0.040	***0.000***
	Sum	0.380	0.360	0.300	0.340	*0.180*
0.001	10×5	0.163	*0.100*	1.041	1.528	0.166
	30×10	4.142	3.715	4.193	4.993	*3.413*
	50×20	0.648	0.544	0.761	0.470	***0.343***
	Sum	4.953	4.359	5.995	6.991	***3.922***
0.005	10×5	0.615	1.179	*0.602*	0.649	1.076
	30×10	*2.776*	2.967	3.014	3.069	3.353
	50×20	1.395	0.876	0.985	1.238	***0.540***
	Sum	4.786	5.022	*4.601*	4.956	4.969
0.01	10×5	0.420	0.779	*0.394*	0.398	0.664
	30×10	3.490	*2.851*	3.538	3.108	3.068
	50×20	0.970	1.371	1.107	1.384	***0.539***
	Sum	4.880	5.001	5.039	4.890	*4.271*
0.05	10×5	0.039	0.029	*0.022*	0.058	0.042
	30×10	3.036	3.030	3.032	3.255	***2.128***
	50×20	1.083	1.421	1.303	1.356	***0.491***
	Sum	4.158	4.480	4.357	4.670	***2.661***
0.1	10×5	0.017	0.025	0.014	0.033	*0.030*
	30×10	2.515	2.295	2.161	1.978	*1.839*
	50×20	0.822	1.182	1.064	1.327	***0.415***
	Sum	3.354	3.502	3.239	3.338	*2.285*
0.5	10×5	*0.006*	0.039	0.015	0.049	0.039
	30×10	2.549	2.426	1.801	2.341	***1.407***
	50×20	0.904	1.306	0.798	1.132	*0.327*
	Sum	3.458	3.771	2.615	3.522	***1.773***
1.0	10×5	0.042	0.069	*0.029*	0.075	0.047
	30×10	2.787	2.468	2.243	2.449	***1.741***
	50×20	1.172	1.356	1.041	1.289	***0.362***
	Sum	4.000	3.893	3.313	3.813	*2.150*

[a] average absolute deviation for $\lambda = 0$, [b] average percentage deviation for λ

The constructive algorithms are compared to the best solutions. It is shown that in particular, for the simple dispatching rules the SPT, LPT, ERD, and HSE rules are good algorithms whereas for the flow shop makespan heuristics, the NEH algorithm is most superior to the other constructive algorithms. When we have applied the polynomial improvement algorithm, we have found that the all-pairwise interchange algorithm is a good improvement algorithm. Next, we used TS-based algorithms as improving algorithms. Before we studied the influence of the initial solution on the performance of the TS algorithm, we tested the TS parameters, i.e., number of neighbors, neighborhood structure, and size of tabu list. We have found that a constant number of 20 neighbors works best. The neighborhood structures should be based on shift moves for $\lambda \geq 0.05$ and on pairwise interchanges of jobs otherwise. The size of tabu list should be selected as 10. For the recommended TS parameters, we investigated the selection of a starting solution by using several constructive and improvement algorithms. The variants INEHTS and NEHTS can both be recommended in general.

Table 11: Comparison of the TS algorithm with different initial solutions (Group III).

λ	Problem size	ISPTTS	ILPTTS	IERDTS	IEDDTS	IMSTTS	IS/PTS	IHSETS
0	10×5	**0.000**[a]	**0.000**	**0.000**	0.020	**0.000**	**0.000**	**0.000**
	30×10	0.280	0.220	0.220	0.260	0.240	0.240	*0.220*
	50×20	0.100	0.120	0.080	**0.000**	**0.000**	0.040	**0.000**
	Sum	0.380	0.340	0.300	0.280	0.240	0.280	*0.220*
0.001	10×5	0.531[b]	0.575	0.044	0.544	0.525	0.526	*0.043*
	30×10	4.107	3.829	4.898	3.386	**2.985**	3.963	3.743
	50×20	0.847	1.014	0.797	1.174	0.555	0.805	*0.358*
	Sum	5.485	5.418	5.739	5.104	*4.065*	5.294	4.144
0.005	10×5	0.573	1.279	1.044	0.589	**0.272**	0.413	0.795
	30×10	2.906	3.068	3.140	**2.514**	2.904	3.509	3.137
	50×20	0.932	1.745	2.025	2.273	*0.803*	1.730	1.122
	Sum	4.411	6.092	6.209	5.376	**3.979**	5.652	5.054
0.01	10×5	0.468	0.481	0.477	0.485	0.405	0.344	*0.312*
	30×10	3.824	3.346	2.963	*2.936*	3.065	2.968	3.497
	50×20	1.241	1.775	1.635	1.437	1.266	*1.132*	1.470
	Sum	5.533	5.602	5.075	4.858	4.736	*4.444*	5.279
0.05	10×5	0.076	0.052	0.057	0.052	0.065	0.041	*0.039*
	30×10	3.094	2.692	2.743	3.343	2.860	2.697	*2.588*
	50×20	*1.027*	1.146	1.700	1.494	1.056	1.316	1.152
	Sum	4.196	3.891	4.500	4.889	3.981	4.054	*3.779*
0.1	10×5	0.010	0.035	0.031	0.046	0.039	0.055	*0.010*
	30×10	2.569	2.628	*2.107*	2.528	2.259	2.166	2.661
	50×20	*0.946*	1.304	1.302	1.222	1.052	1.071	1.229
	Sum	3.524	3.966	3.440	3.796	3.350	*3.292*	3.899
0.5	10×5	0.027	0.035	0.061	0.043	0.043	0.031	**0.004**
	30×10	2.179	2.256	*2.004*	2.276	2.127	2.321	2.188
	50×20	*0.927*	1.249	1.172	1.326	1.214	1.185	1.061
	Sum	*3.133*	3.540	3.237	3.645	3.384	3.537	3.253
1.0	10×5	0.037	0.065	0.066	0.035	0.075	*0.035*	0.048
	30×10	2.366	2.433	2.082	2.177	*1.960*	2.116	2.203
	50×20	0.972	1.226	1.172	1.192	1.208	1.060	*0.972*
	Sum	3.375	3.724	3.320	3.404	3.243	*3.211*	3.223

[a] average absolute deviation for $\lambda = 0$, [b] average percentage deviation for λ

Further research can be done to use other improving algorithms such as genetic or ant colony algorithms. The choice of good parameters for them should be tested. The influence of the starting solution should be investigated. Moreover, hybrid algorithms should be developed by using a tabu search algorithm as a local search algorithm within a genetic algorithm or the other algorithms.

Table 12: Comparison of the TS algorithm with different initial solutions (Group IV).

λ	Problem size	IPALTS	ICDSTS	IGUPTS	IDANTS	INEHTS
0	10×5	**0.000**[a]	0.020	**0.000**	0.020	**0.000**
	30×10	0.220	0.300	0.260	0.200	**0.160**
	50×20	0.140	0.040	0.060	0.040	**0.000**
	Sum	0.360	0.360	0.320	0.260	**0.160**
0.001	10×5	1.024[b]	1.056	0.544	*0.100*	0.534
	30×10	4.915	4.934	4.743	4.804	*3.421*
	50×20	1.087	0.380	0.369	0.758	*0.350*
	Sum	7.026	6.370	5.656	5.662	*4.305*
0.005	10×5	0.832	1.103	*0.527*	0.732	0.721
	30×10	3.182	3.329	*2.910*	3.278	3.461
	50×20	2.089	0.676	0.778	1.441	*0.543*
	Sum	6.103	5.108	*4.215*	5.451	4.725
0.01	10×5	0.628	0.387	*0.286*	0.603	0.706
	30×10	3.274	3.109	3.316	3.597	*2.994*
	50×20	1.782	1.124	0.942	1.557	**0.539**
	Sum	5.684	4.620	4.544	5.757	*4.239*
0.05	10×5	0.059	0.054	0.070	**0.022**	0.029
	30×10	3.512	3.238	2.731	2.439	*2.233*
	50×20	1.531	0.891	1.189	1.302	**0.491**
	Sum	5.101	4.183	3.990	3.763	*2.753*
0.1	10×5	*0.009*	0.021	0.012	0.010	0.012
	30×10	2.682	2.356	2.620	1.914	**1.832**
	50×20	1.266	1.036	0.782	1.284	*0.416*
	Sum	3.957	3.413	3.414	3.208	**2.260**
0.5	10×5	*0.019*	0.041	0.044	0.044	0.039
	30×10	1.999	2.288	2.121	2.049	*1.456*
	50×20	1.495	0.876	1.056	0.937	**0.314**
	Sum	3.514	3.205	3.221	3.030	*1.809*
1.0	10×5	0.016	0.024	0.047	**0.011**	0.035
	30×10	2.654	2.181	2.179	2.053	*1.747*
	50×20	1.328	1.033	1.020	1.358	**0.362**
	Sum	3.998	3.237	3.246	3.422	**2.143**

[a] average absolute deviation for $\lambda = 0$, [b] average percentage deviation for λ

ACKNOWLEDGMENT

This work was supported by Department of Industrial Engineering, Chulalongkorn University.

The fourth author has been supported by INTAS (project 03-51-5501).

REFERENCES

Alisantoso, D., Khoo, L.P., and Jiang, P.Y. (2003) An immune algorithm approach to the scheduling of a flexible PCB flow shop, *The International Journal of Advanced Manufacturing Technology,* ***22(11)***, 819-827.

Blum, C., and Roli, A. (2003) Metaheuristics in combinatorial optimization: Overview and conceptual comparison, *ACM Computing Surveys*, ***35(3)***, 268–308.

Campbell, H.G., Dudek, R.A., and Smith, M.L. (1970) A Heuristic algorithm for the n-Job m-Machine sequencing problem, *Management Science,* ***16(10)***, 630–637.

Dannenbring, D.G. (1977) An evaluation of flow shop sequencing heuristics, *Management Science,* ***23(11)***,1174-1182.

Garey, M.R., and Johnson, D.S. (1979) *Computers and Intractability: A Guide to the Theory of NP-Completeness*, CA: Freeman, San Francisco.

Glover, F., and Laguna, M. (1993) *Tabu search. In C.R.Reeves (ed), Modern Heuristic Techniques for Combinatorial Problems (UK: Blackwell Scientific Publications)*, Oxford, chapter 3, 70-150.

Glover, F. (1986) Future paths for integer programming and links to artificial intelligence, *Computers and Operations Research,* ***13(5)***, 533–549.

Gupta, J.N.D (1971) A functional heuristic algorithm for the flow-shop scheduling problem, *Operations Research Quarterly,* ***22(1)***, pp. 39-47.

Gupta, J.N.D., Krüger, K., Lauff, V., Werner, F., and Sotskov, Y.N. (2002) Heuristics for hybrid flow shops with controllable processing times and assignable due dates, *Computers & Operations Research,* ***29(10)***, 1417-1439.

Jungwattanakit, J., Reodecha, M., Chaovalitwongse, P., and Werner, F. (2006a) Solving the hybrid flowshop scheduling problem with unrelated parallel machines and sequence-dependent setup times by simulated annealing algorithm, *Proceedings of the 1st International Conference & 7th AUN/SEED-Net Fieldwise Seminar on Manufacturing and Material Process, Kuala Lumpur, Malaysia*, 640–645.

Jungwattanakit, J., Reodecha, M., Chaovalitwongse, P., and Werner, F. (2006b) Constructive and simulated annealing heuristics for hybrid flow shops with unrelated parallel machines, *Proceedings of the 3rd OR-CRN Operations Research conference 2006, Bangkok, Thailand*, 110–121.

Jungwattanakit, J., Reodecha, M., Chaovalitwongse, P., and Werner, F. (2006c) Sequencing heuristics for flexible flow shop scheduling problems with unrelated parallel machines and setup time, *Proceedings of the 2006 IE Network National Conference. Bangkok, Thailand.* (to appear)

Lin, Hung-Tso., and Liao, Ching-Jong (2003) A case study in a two-stage hybrid flow shop with setup time and dedicated machines, *International Journal of Production Economics,* ***86(2)***, 133-143.

Nawaz, M., Enscore, Jr. E., Ham, I. (1983) A heuristic algorithm for the m-machine, n-job flowshop sequencing problem, *Omega International Journal of Management Science,* ***11(1)***, 91–95.

Palmer, D.S. (1965) sequencing jobs through a multi-stage process in the minimum total time--a quick method of obtaining a near optimum, *Operations Research Quarterly,* ***16(1)***,101–107.

Pinedo, M., and Chao, X. (1999) *Operations scheduling with applications in manufacturing and services*, Irwin/McGraw-Hill, New York.

Rajendran, C., and Ziegler, H. (2003) Scheduling to minimize the sum of weighted flowtime and weighted tardiness of jobs in a flowshop with sequence-dependent setup times, *European Journal of Operational Research,* ***149(3)***, 513–522.

Wang, W. (2005) Flexible Flow Shop Scheduling: Optimum, heuristics, and artificial intelligence solution, *Expert Systems*, ***22(2)***, 78 – 85.

Wang, W., and Hunsucker, J.L. (2003) An evaluation of the CDS heuristic in flow shops with multiple processors, *Journal of the Chinese Institute of Industrial Engineers,* ***20 (3)***, 295-304.

AUTHOR BIOGRAPHIES

Jitti Jungwattanakit is a Ph.D. student in Industrial Engineering Department, Chulalongkorn University, Thailand. His main research areas are Scheduling, Operations Research, and Simulation. His email address is <jitti.j@student.chula.ac.th>.

Manop Reodecha is an Assistant Professor in Industrial Engineering Department, Chulalongkorn University, Thailand. He received a Ph.D. in Industrial Engineering from North Carolina State University. His main research areas are Logistics and Supply Chain Management, Production and Operations Management, and Information Systems. His email address is <manop@eng.chula.ac.th>.

Paveena Chaovalitwongse is an Assistant Professor in Industrial Engineering Department, Chulalongkorn University, Thailand. She received a Ph.D. degree from University of Florida. Her main research areas are Operations Research, Supply Chain and Logistics Management, and Production Planning and Control. Her email address is <paveena.c@chula.ac.th>.

Frank Werner is an Extraordinary Professor at the Faculty of Mathematics of the Otto-von-Guericke-University, Germany. He received a Ph.D. degree from the TU Magdeburg. His main research areas are Scheduling, Stability investigations, Graph coloring problems, Logistics, and Supply Chains. His email address is <frank.werner@mathematik.uni-magdeburg.de>.