Constructive and Simulated Annealing Algorithms for Hybrid Flow Shop Problems with Unrelated Parallel Machines

Jitti Jungwattanakit, Manop Reodecha, Paveena Chaovalitwongse
Department of Industrial Engineering, Faculty of Engineering,
Chulalongkorn University, Bangkok 10330 Thailand. jitti.j@student.chula.ac.th
Frank Werner
Faculty of Mathematics, Otto-von-Guericke-University, P.O. Box 4120,
D-39016 Magdeburg, Germany. Frank.Werner@Mathematik.Uni-Magdeburg.DE

Abstract

Most scheduling problems are combinatorial optimization problems which are too difficult to be solved optimally, and hence heuristics are used to obtain good solutions in a reasonable time. The specific 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. Some simple dispatching rules and flow shop makespan heuristics are adapted for the sequencing problem under consideration. Then, this solution may be improved by a fast polynomial reinsertion algorithm. Moreover, a simulated annealing algorithm is presented in this paper. Three basic parameters (i.e., cooling schedules, neighborhood structures, and initial temperatures) of a simulated annealing 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; Constructive algorithms; Improvement heuristics; Simulated Annealing algorithms.

1. Introduction

This paper is primarily concerned with industrial scheduling problems, where one first has to assign jobs with limited resources and then to sequence the assigned jobs on each resource over time. It is mainly concerned with processing industries that are established as multi-stage production facilities with multiple production units per stage (i.e., parallel machines), e.g. a textile company (Karacapilidis and Pappis, [1]), an automobile assembly plant (Agnetis *et al.*, [2]), a printed circuit board manufacturer (Alisantoso *et al.*, [3]), and so on. In such industries, at some stages the facilities are duplicated in parallel to increase the overall capacities, or to balance the capacities of the stages, or either to eliminate or to reduce the impact of bottleneck stages on the shop floor capacities. The mixed character between flow shop and parallel machines is known as a hybrid or flexible flow shop environment.

Although the hybrid flow shop problem has been widely studied in the literature, most of the studies related to hybrid flow shop problems are concentrated on problems with identical processors, see for instance, Gupta et *al.*, [4] and Wang and Hunsucker [5]. In a real world situation, it is common to find newer or more modern machines running side by side with older and less efficient machines. Even though the older machines are less efficient, they may be kept in the production lines because of their high replacement costs. The older machines may perform the same operations as the newer ones, but would generally require a longer operating time for the same operation. In this paper, the hybrid flow shop problem with unrelated parallel machines is considered, i.e., there are

different parallel machines at every stage and speeds of the machines are dependent on the jobs. Moreover, several industries encounter setup times which result in even more difficult scheduling problems.

A detailed survey for the hybrid flow shop problem has been given in Linn and Zhang [6] and Wang [7]. Most of the earlier literature has considered the simple case of only two stages. Arthanari and Ramamurthy [8] and Salvador [9] are among the first who define the hybrid flow shop problem. They propose a branch and bound method to tackle the problem. However, it can only be applied to very small instances. Other exact approaches are proposed by many authors, e.g. Brah and Hunsucker [10] and Moursli and Pochet [11].

When an exact algorithm is applied to large problems, such an approach can take hours or days to derive a solution. On the other hand, a heuristic approach is much faster but does not guarantee an optimum solution. Gupta [12] proposes heuristic techniques for a simplified hybrid flow shop makespan problem with two stages and only one machine at stage two. The proposed heuristics are based on extensions of Johnson's algorithm. Sriskandarajah and Sethi [13] develop simple heuristic algorithms for the two-stage hybrid flow shop problem. They discuss the worst and average case performance of algorithms for finding minimum makespan schedules. Guinet *et al.*, [14] propose a heuristic for the makespan problem in a two-stage hybrid flow shop. They compare this heuristic with the Shortest Processing Time (SPT) and the Longest Processing Time (LPT) dispatching rules. They conclude that the LPT rule gives good results for the two-stage makespan problem. Gupta and Tunc [15] consider the two-stage hybrid flow shop scheduling problem where there is one machine at stage one and the number of identical machines in parallel at stage two is less than the total number of jobs. The setup and removal times of each job at each stage are separated from the processing times. They propose heuristic algorithms that are empirically tested to determine the effectiveness in finding an optimal solution. Santos *et al.*, [16] investigate scheduling procedures which seek to minimize the makespan in a static hybrid flow shop. Their method is to generate an initial permutation schedule based on the Palmer, CDS, Gupta and Dannenbring flow shop heuristics, and then it is followed by the application of the First-In-First-Out (FIFO) rule.

To obtain a near-optimal solution, metaheuristic algorithms have also been proposed. For example, Nowicki and Smutnicki [17] propose a tabu search (TS) algorithm for the hybrid flow shop makespan problem. Gourgand *et al.*, [18] present several simulated annealing (SA)-based algorithms for the hybrid flow shop problem. A specific neighborhood is used and the authors apply the methods to a realistic industrial problem. Jin *et al.* [19] propose two approaches to generate the initial job sequence and use an SA algorithm to improve it. It can be seen that the SA algorithm has been successfully applied to various combinatorial optimization problems. For an extensive survey of the theory and applications of the SA algorithm, see Koulamas *et al.*, [20].

In this paper, a hybrid flow shop problem with unrelated parallel machines and setup times is studied. The goal is to seek a schedule which minimizes, a positively weighted convex sum of makespan and the number of tardy jobs. The constructive heuristics based on dispatching rules and pure flow shop makespan heuristics are adapted and SA-based algorithms as iterative algorithms are proposed.

The rest of this paper is organized as follows: The problem under consideration is described in Section 2. Heuristic algorithms are sketched in Section 3. Section 4 and Section 5 present the variants of the SA algorithm. Computational results with the heuristics are briefly discussed in Section 6 and conclusions are given in Section 7.

2. Problem Statement

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 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 of a job have to be realized sequentially, without overlapping between stages; (4) job splitting is not permitted.

Setup times considered in this problem are classified into two types, namely machine-dependent setup time and 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 on the machine. ch^t_{ij} denotes 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 data are known and constant.

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

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

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. 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 forms of specific problem knowledge. However, the time required for computation is usually greater compared to the constructive algorithms. The drawback of heuristic algorithms is that they do not generate optimality and it may be difficult to judge their effectiveness (Youssef *et al.*, [21]).

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 *et al.*, [16].

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 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 (the job sequence for this particular stage is derived either from the FIFO 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, it is noted 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''_{ij} and setup time s''_{lj} for stage t 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''_{ij} plus the representative of the setup time s''_{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, while 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 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 concerns the remaining slack of each job, defined as its due date minus its processing time. The Slack time per Processing time (S/P) is the slack time divided by the processing time required (Baker, [22]).

Palmer's heuristic [23] is a makespan heuristic denoted by PAL 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 [24] develop one of the makespan heuristic methods known as CDS algorithm. Since Johnson's rule is a two-stage algorithm, a k-stage problem must be collapsed into a two-stage problem. In so doing, $k - 1$ sub-problems are created and Johnson's rule is applied to each of the sub-problems. Then, a "best" sequence is selected. Gupta [25] 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 [26] denoted by DAN develops a method 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.

Nawaz, Enscore and Ham [27] develop a flow shop makespan heuristic, 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 that gives the best objective function value under consideration (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.

To apply the algorithms to this problem, 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. The best solution is selected from them.

3.3 Improvement Heuristics

Improvement heuristics start with an already built schedule and attempt to improve it by some given procedure. Their use is necessary since the constructive algorithms (especially some algorithms that are adapted from pure makespan heuristics and some dispatching rules such as SPT, LPT) do not consider due dates. We will improve the overall function value concerning the due date criterion. In order to find a satisfactory solution of the given problem involving due dates, we use a fast polynomial heuristic by applying the shift move (SM) neighborhood as an improvement mechanism.

The SM neighborhood repositions a chosen job. 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 example, assume that one solution in the current generation is selected, say [5 9 8 7 3 1 6 2 4], and then 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 [5 8 7 3 1 6 ***9*** 2 4]. However, if positions 7 and 2 are selected (i.e. it is a left shift), the new solution will be [5 ***6*** 9 8 7 3 1 2 4]. In the SM neighborhood, the current solution (S_{cur}) has $(n-1)^2$ neighbors.

We apply the SM neighborhood by considering only jobs that are tardy in a left-to-right scan and move each of them left and right to all $n-1$ possible positions (all shift-move algorithm). In each step, the best schedule is selected if it improves the objective function value. Since every job is considered at most once (if all jobs under consideration are late), at most $O(n^2)$ job sequences are examined by the improvement heuristics).

4. Simulated Annealing Heuristic

A simulated annealing (SA) heuristic has been introduced by Kirkpatrick *et al.*, [28]. It is an enhanced version of local optimization, in which an initial solution is repeatedly improved by making small local alterations, but an SA procedure often accepts a poor solution to avoid being trapped in a poor local optimum.

A basic SA algorithm starts from an initial solution $s \in \boldsymbol{S}$, and it generates a new solution $s' \in \boldsymbol{S}$ in the neighborhood of the initial solution s by using a suitable operator. This new point's objective function value $f(s')$ is then compared to the initial point's value $f(s)$ (the objective function value of the full schedule generated from the job sequence for the first stage is taken). The change in the objective function value, $\delta = f(s') - f(s)$, is calculated. If the objective function value decreases ($\delta < 0$), it is automatically accepted and it becomes the point from which the search will continue. If the objective function value increases ($\delta \geq 0$), then higher values of the objective function may also be accepted with a probability, usually determined by a function, $\exp(-\delta/T)$, where $T \in \Re$ is a control parameter of an SA algorithm called the *temperature*. The role of the temperature T is significant in the operation of an SA algorithm. This temperature, which is simply a positive number, is periodically reduced every NT iterations, where NT denotes the epoch length, so that it moves gradually from a relatively high value to near zero as the method progresses according to a function referred to as the *cooling schedule*. In our tests, we investigated in particular the influence of the chosen neighborhood and the cooling scheme for controlling the temperature. We used a geometric (i.e., $T_{new} = \alpha \times T_{old}$) and a Lundy-Mees reduction [29] (i.e., $T_{new} = T_{old}/(1+\beta \times T_{old})$) scheme and tested the parameters of these schemes (initial temperatures, temperature reductions and neighborhood structures).

Concerning the neighborhood, we considered both an SM neighborhood (see Section 3.3) and a pairwise interchange (PI) neighborhood. The idea for the PI neighborhood is to exchange a pair of arbitrary jobs, π_r and π_i, where $1 \leq i, r \leq n$ and $i \neq r$. Such an operation swaps the job at position r and one at position 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 instance. assume that the current solution is [5 9 8 7 3 1 6 2 4], 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 [***8*** 9 ***5*** 7 3 1 6 2 4]. For the selection of a neighbor, one of all possible $n \times (n-1)/2$ PI neighbors is checked and then compared to the starting one.

5. Choice of an initial solution

An SA algorithm has been shown to be effective for many combinatorial optimization problems (see Koulamas *et al.*, [20]), 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 and fast improvement algorithms. We used one constructive algorithm of: SPT, LPT, ERD, EDD, MST, S/P, PAL, CDS, GUP, DAN and NEH; we also used another fast improvement heuristics as an initial solution.

6. Computational results

Firstly, we studied the algorithms from Section 3 (determination of an initial solution for SA) which are separated into four main groups. The first heuristic group includes the simple dispatching rules such as SPT, LPT, ERD, EDD, MST, and S/P. The second heuristic group contains the flow shop makespan heuristics adaptation such as PAL, CDS, GUP, DAN, and NEH. The third and fourth heuristic groups are generated from the first two heuristics by applying additionally an all-shift-move algorithm (see Section 3.3), and they are denoted by the letter "I" before the letters for the constructive heuristics. 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 $\lambda \in \{0, 0.05, 0.1, 0.5, \text{and } 1\}$ in the objective function. Ten different instances for each problem size have been run.

The results for the algorithms from Section 3 are given in Table 1. We give the average (absolute for $\lambda = 0$ and percentage for $\lambda > 0$) deviation of a particular algorithm from the best solution in these tests for all problem sizes $n \times k$ (the overall best variant is given in bold face).

Table 1 Average performance of constructive and fast improvement algorithms

λ	Problem size	SPT	LPT	ERD	EDD	MST	S/P	PAL	CDS	GUP	DAN	NEH
0	10×5	2.3[a]	1.7	2.8	3.1	3.2	3.0	1.9	1.7	1.8	2.0	**0.5**
	30×10	8.0	8.9	8.3	12.4	12.3	12.2	8.0	6.4	7.8	7.7	**2.4**
	50×20	7.4	8.6	7.7	16.2	16.2	14.3	9.7	7.3	7.9	9.3	**2.3**
	Sum	17.7	19.2	18.8	31.7	31.7	29.5	19.6	15.4	17.5	19.0	**5.2**
0.05	10×5	18.82[b]	12.81	24.23	24.09	22.21	22.10	11.66	10.03	14.12	11.47	**2.52**
	30×10	17.78	14.72	19.61	20.91	18.21	17.61	16.80	12.35	14.71	14.77	**0.59**
	50×20	8.53	8.28	10.14	11.75	10.96	9.87	7.90	6.99	8.03	8.43	**0.30**
	Sum	45.14	35.81	53.98	56.75	51.37	49.58	36.36	29.36	36.86	34.67	**3.41**
0.1	10×5	17.90	11.82	22.91	22.67	20.21	20.52	10.71	8.79	13.05	10.32	2.86
	30×10	16.61	13.12	18.46	19.14	16.38	15.71	15.59	11.17	13.30	13.61	**0.40**
	50×20	8.13	7.75	9.72	10.47	9.65	8.77	7.27	6.45	7.57	7.79	**0.08**
	Sum	42.64	32.69	51.08	52.28	46.25	45.00	33.57	26.40	33.92	31.72	**3.33**
0.5	10×5	17.48	11.21	22.17	21.94	18.81	19.21	10.33	7.87	12.44	9.74	2.98
	30×10	16.12	12.29	18.01	18.13	15.30	14.46	15.03	10.50	12.60	13.04	**0.26**
	50×20	8.11	7.59	9.68	9.71	8.84	8.12	7.04	6.28	7.50	7.56	**0.09**
	Sum	41.71	31.09	49.86	49.78	42.94	41.79	32.41	24.65	32.55	30.34	**3.33**
1.0	10×5	17.48	11.21	22.17	21.94	18.81	19.21	10.33	7.87	12.44	9.74	2.98
	30×10	16.34	12.46	18.24	18.29	15.43	14.57	15.24	10.68	12.77	13.24	**0.38**
	50×20	8.12	7.58	9.69	9.63	8.75	8.05	7.03	6.27	7.50	7.54	**0.08**
	Sum	41.94	31.24	50.10	49.85	42.98	41.82	32.59	24.82	32.72	30.52	**3.44**

λ	Problem size	ISPT	ILPT	IERD	IEDD	IMST	IS/P	IPAL	ICDS	IGUP	IDAN	INEH
0	10×5	1.0	0.7	1.4	0.7	1.0	0.9	0.6	**0.5**	0.9	1.0	**0.5**
	30×10	4.2	4.5	4.3	3.2	**2.4**	5.7	4.4	4.0	4.1	4.4	**2.4**
	50×20	3.6	4.3	3.5	3.5	3.9	7.8	4.6	4.5	4.8	5.3	**2.3**
	Sum	8.8	9.5	9.2	7.4	7.3	14.4	9.6	9.0	9.8	10.7	**5.2**
0.05	10×5	5.06	3.60	3.82	6.84	5.32	5.43	2.64	3.39	3.63	3.46	**2.52**
	30×10	6.03	6.66	7.75	8.00	11.15	8.29	7.07	4.78	6.84	6.24	**0.59**
	50×20	4.46	5.22	5.10	6.03	5.66	6.28	4.22	3.53	4.83	5.38	**0.30**
	Sum	15.55	15.49	16.67	20.88	22.14	19.99	13.93	11.70	15.30	15.08	**3.41**
0.1	10×5	4.63	3.95	4.60	6.40	5.78	4.48	**1.80**	2.01	3.20	1.94	2.86
	30×10	6.10	6.14	8.39	7.94	9.93	7.83	6.23	2.78	5.57	5.51	**0.40**
	50×20	4.27	5.22	4.78	5.56	5.66	4.61	3.78	3.15	4.88	4.73	**0.08**
	Sum	14.99	15.31	17.78	19.89	21.36	16.93	11.81	7.93	13.66	12.18	**3.33**
0.5	10×5	4.31	2.84	4.91	6.01	5.80	3.98	1.71	2.24	2.55	**0.80**	2.98
	30×10	6.13	6.10	8.94	7.85	9.18	7.20	5.19	1.65	4.88	5.95	**0.26**
	50×20	4.37	4.72	4.81	5.28	5.44	4.73	4.06	2.92	4.79	4.61	**0.09**
	Sum	14.81	13.66	18.66	19.14	20.42	15.91	10.96	6.82	12.22	11.36	**3.33**
1.0	10×5	4.31	2.84	4.91	6.01	5.80	3.98	1.71	2.24	2.55	**0.80**	2.98
	30×10	6.33	6.18	9.17	7.72	9.33	7.40	5.29	1.69	4.98	5.95	**0.38**
	50×20	4.39	5.01	4.91	5.17	5.45	4.09	4.05	2.93	4.77	4.59	**0.08**
	Sum	15.04	14.02	18.98	18.91	20.58	15.47	11.05	6.86	12.31	11.34	**3.44**

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

From these results it is obvious that the algorithms in the fourth heuristic group (i.e., IPAL, ICDS, IGUP, IDAN, and INEH) can improve the pure makespan heuristics from the second heuristic group (i.e., PAL, CDS, GUP, DAN, and NEH), and they are better than the dispatching rules in the first heuristic group (i.e., SPT, LPT, EDD, MST, and S/P) as well as the third heuristic group improved from them.

Among the simple dispatching rules (heuristic Group I), the SPT rule outperforms the other dispatching rules for $\lambda = 0$, and the LPT rule is better than the other rules for $\lambda > 0$. Among the adapted flow shop makespan heuristics in the heuristic Group II, the NEH algorithm is clearly the best algorithm among all studied constructive heuristics. The CDS algorithm is certainly the second rank algorithm, whereas the remaining algorithms differ slightly from each other.

When we apply a fast (re-)insertion algorithm (denoted by the letter "I" first) to the dispatching rules and adapted makespan heuristics, we have found that the quality of the solution can be improved by about 50–70 percent except for the NEH rule. It is noted that the NEH rule is not improved by using the improvement heuristics in algorithm INEH because the NEH algorithm is embedded by such an (re-) insertion algorithm itself. However, the improvement of the heuristics from the adapted pure makespan heuristics in the heuristic Group IV is better than the improvement of the heuristics derived from the dispatching rules in the heuristic Group III.

Secondly, we studied the SA algorithm with a random initial solution. The purpose of this study is to determine the favorable SA parameters, i.e., initial temperatures (100 through 1000, in steps of 100), neighborhood structures (PI and SM), and cooling schedules (CS1 – CS3 refer to a geometric reduction schedule with $\alpha \in \{0.85, 0.90, \text{and } 0.95\}$, and CS4–CS6 are the schedules by Lundy and Mees with $\beta \in \{0.0005, 0.001, \text{and } 0.002\}$).

Given the above three different problem sizes, the SA parameter values were tested. From our 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. Table 2 through Table 4 present the effect of the initial temperatures, neighborhood structures and cooling schedules by using the average (absolute resp. relative) deviation from the best value as the performance measure.

From the full factorial experiment, we analyzed our results by means of a multi-factor *Analysis of Variance* (*ANOVA*) technique using a 5% significance level. We have found that for neighborhood structures and cooling schedules, there are statistically significant differences, whereas there are not statistically significant differences in the initial temperatures. A low initial temperature is however slightly preferable (we recommend 100). It can be observed that PI

Table 2 The effect of various initial temperatures on the performance of the SA algorithm

λ	Problem size	100	200	300	400	500	600	700	800	900	1000
0	10×5	0.019[a]	0.022	0.019	**0.011**	0.025	0.019	0.019	0.017	0.022	0.025
	30×10	**2.747**	2.781	2.767	2.781	2.792	2.822	2.811	2.839	2.864	2.883
	50×20	**2.461**	2.531	2.561	2.561	2.539	2.603	2.628	2.608	2.653	2.681
	Sum	**5.227**	5.334	5.347	5.353	5.356	5.444	5.458	5.464	5.539	5.589
0.05	10×5	**1.954**[b]	2.140	2.171	2.079	2.010	2.195	2.195	2.192	2.261	2.251
	30×10	**7.662**	7.727	7.979	7.816	7.880	7.809	7.925	7.877	7.770	7.838
	50×20	**3.901**	4.010	4.100	4.161	4.117	4.145	4.151	4.232	4.197	4.277
	Sum	**13.517**	13.877	14.250	14.056	14.007	14.149	14.271	14.301	14.228	14.366
0.1	10×5	1.707	**1.647**	1.840	1.917	1.922	1.864	1.839	1.969	1.895	1.893
	30×10	**6.126**	6.137	6.218	6.237	6.218	6.254	6.304	6.291	6.386	6.361
	50×20	**3.440**	3.446	3.535	3.596	3.608	3.652	3.626	3.658	3.750	3.675
	Sum	11.273	**11.230**	11.593	11.750	11.748	11.770	11.769	11.918	12.031	11.929
0.5	10×5	**0.850**	0.884	0.873	0.947	0.962	0.959	0.968	1.048	1.025	1.030
	30×10	**3.723**	3.781	3.814	3.898	3.931	3.909	3.947	3.926	3.915	3.974
	50×20	**2.125**	2.240	2.285	2.312	2.360	2.404	2.381	2.377	2.414	2.338
	Sum	**6.698**	6.905	6.972	7.157	7.253	7.272	7.296	7.351	7.354	7.342
1.0	10×5	**0.513**	0.641	0.633	0.653	0.690	0.726	0.705	0.756	0.721	0.684
	30×10	**3.337**	3.392	3.470	3.452	3.504	3.497	3.520	3.554	3.546	3.534
	50×20	**1.761**	1.837	1.847	1.915	1.924	1.987	1.942	1.963	2.007	1.953
	Sum	**5.611**	5.870	5.950	6.020	6.118	6.210	6.167	6.273	6.274	6.171

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

Table 3 The effect of various neighborhood structures on the performance of the SA algorithm

λ	Problem size	PI	SM
0	10×5	**0.016**[a]	0.024
	30×10	**2.794**	2.823
	50×20	**2.522**	2.643
	Sum	**5.332**	5.490
0.05	10×5	2.270[b]	**2.020**
	30×10	8.098	**7.522**
	50×20	4.192	**4.067**
	Sum	14.560	**13.609**
0.1	10×5	1.973	**1.725**
	30×10	6.522	**5.985**
	50×20	3.646	**3.551**
	Sum	12.141	**11.261**
0.5	10×5	1.136	**0.773**
	30×10	4.249	**3.515**
	50×20	2.425	**2.222**
	Sum	7.810	**6.510**
1.0	10×5	0.865	**0.479**
	30×10	3.897	**3.065**
	50×20	2.049	**1.778**
	Sum	6.811	**5.322**

Table 4 The effect of various cooling schedules on the performance of the SA algorithm

λ	Problem size	CS1	CS2	CS3	CS4	CS5	CS6
0	10×5	**0.000**	0.002	0.028	0.033	0.022	0.035
	30×10	**0.653**	0.915	1.663	4.625	4.562	4.433
	50×20	**0.320**	0.638	1.880	4.237	4.283	4.137
	Sum	**0.973**	1.555	3.571	8.895	8.867	8.605
0.05	10×5	**0.936**	1.052	1.930	3.142	3.043	2.767
	30×10	**3.388**	3.443	4.614	12.314	12.034	11.068
	50×20	**1.059**	4.411	3.101	6.559	6.472	6.171
	Sum	**5.383**	8.906	9.645	22.015	21.549	20.006
0.1	10×5	**0.855**	0.959	1.561	2.770	2.696	2.255
	30×10	**2.741**	2.839	3.725	10.487	9.717	8.011
	50×20	**1.000**	1.295	2.620	6.008	5.758	4.911
	Sum	**4.596**	5.093	7.906	19.265	18.171	15.177
0.5	10×5	0.658	0.636	0.920	1.755	1.149	**0.610**
	30×10	**2.450**	2.560	2.940	6.833	4.816	3.693
	50×20	**0.959**	1.168	1.969	4.429	3.103	2.313
	Sum	**4.067**	4.364	5.829	13.017	9.068	6.616
1.0	10×5	0.590	0.546	0.782	1.122	0.628	**0.364**
	30×10	2.754	**2.700**	3.102	5.051	3.885	3.393
	50×20	**0.963**	1.157	1.827	3.167	2.400	1.968
	Sum	**4.306**	4.403	5.711	9.340	6.913	5.726

[a] average absolute deviation for $\lambda = 0$ and [b] average percentage deviation for $\lambda > 0$

moves are better than SM neighborhoods for λ = 0, whereas SM neighborhoods are better than PI moves for the other values. Consequently, the neighborhood structures should be based on PI moves for λ = 0 and on SM neighborhood otherwise. For the cooling schedules, we have observed that a geometric cooling scheme outperforms the other cooling schedules. In particular, the reduction scheme $T_{new}=0.85\times T_{old}$, where T_{new} and T_{old} denote the new and old temperatures, can be recommended.

Finally, we used the recommended SA parameters to test the choice of an appropriate initial solution. The letters before SA denote the heuristic rule for finding an initial solution for the SA algorithm. For example, SPTSA means that the SPT rule is used as an initial solution for the SA algorithm.

From these results in Table 5, we have found that there are no statistically significant differences when using different initial solutions. We have however found that the IEDDSA rule is a good algorithm for problems with λ= 0, and the NEHSA and INEHSA rules are slightly better than the others for problems with λ > 0. Consequently, in general the NEHSA and INEHSA algorithms are good choices for the SA algorithm with using a biased initial solution.

7. Conclusions

In this paper, we have investigated both constructive and iterative (SA-based) approaches 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 often occurs in the textile industry. 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. The constructive algorithms are compared to each other. It is shown that the NEH and CDS algorithms outperform the others, respectively. In particular, the NEH algorithm is most superior to the other constructive algorithms regardless of improvement heuristics. After the application of the fast improvement heuristics, the INEH algorithm based on the NEH rule is still better than the other algorithms.

In addition, we have used SA-based algorithms as improvement algorithms. Before we studied the influence of the initial solution on the performance of the SA algorithm, we tested the SA parameters, i.e., initial temperatures, neighborhood structures, and cooling schedules. We have found that a low initial temperature is slightly preferable (we recommend 100). The neighborhood structures should be based on PI moves for λ = 0 and on

SM neighborhoods otherwise. The geometric cooling scheme $T_{new}=0.85\times T_{old}$ is recommended. For the recommended SA parameters, we investigated the selection of a starting solution by using several constructive algorithms. The variants NEHSA and INEHSA can both be recommended in general.

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

Table 5 Comparison of the SA algorithm with different initial solutions

λ	Problem size	SPTSA	LPTSA	ERDSA	EDDSA	MSTSA	S/PSA	PALSA	CDSSA	GUPSA	DANSA	NEHSA
0	10×5	**0**[a]	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	30×10	0.84	0.76	0.86	0.82	0.76	0.76	0.76	0.78	0.84	0.78	0.82
	50×20	0.34	0.3	0.44	0.32	0.30	0.34	0.36	0.28	0.38	0.44	0.38
	Sum	1.18	1.06	1.3	1.14	1.06	1.10	1.12	1.06	1.22	1.22	1.20
0.05	10×5	0.70[b]	0.38	0.72	0.45	0.59	0.60	0.56	0.63	0.66	0.52	0.52
	30×10	2.43	2.69	2.83	2.53	2.70	2.63	2.79	2.63	2.39	2.57	2.33
	50×20	1.09	1.07	1.12	1.21	1.06	1.25	1.15	1.16	1.06	**1.01**	1.11
	Sum	4.22	4.14	4.67	4.19	4.35	4.48	4.49	4.42	4.11	4.10	3.96
0.1	10×5	0.64	0.51	0.39	0.43	0.65	0.55	0.50	0.44	0.59	0.55	0.48
	30×10	2.10	2.37	2.41	2.22	2.20	2.07	2.36	**1.96**	2.33	2.10	2.00
	50×20	0.86	0.90	0.92	0.96	1.03	0.96	1.08	0.90	1.10	1.13	**0.80**
	Sum	3.59	3.78	3.72	3.60	3.88	3.58	3.94	3.31	4.02	3.78	3.28
0.5	10×5	0.48	0.39	0.30	0.39	0.33	0.34	0.27	0.36	0.34	0.43	0.43
	30×10	1.99	2.05	1.82	1.98	1.88	2.06	2.10	1.79	1.94	2.10	1.70
	50×20	0.97	0.94	0.83	0.79	0.75	0.90	0.88	0.80	0.86	0.77	**0.63**
	Sum	3.44	3.37	2.95	3.16	2.96	3.30	3.25	2.96	3.14	3.29	2.76
1.0	10×5	0.40	0.33	0.34	0.24	0.23	0.38	0.35	0.30	0.27	0.28	0.36
	30×10	2.38	2.02	2.29	2.08	2.34	2.10	1.94	2.08	2.11	2.31	2.05
	50×20	0.83	0.92	0.80	0.81	0.90	0.85	1.02	0.86	0.96	0.73	**0.66**
	Sum	3.61	3.27	3.43	3.13	3.48	3.33	3.30	3.24	3.34	3.33	3.07

λ	Problem size	ISPTSA	ILPTSA	IERDSA	IEDDSA	IMSTSA	IS/PSA	IPALSA	ICDSSA	IGUPSA	IDANSA	INEHSA
0	10×5	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	30×10	0.82	0.78	0.88	0.68	0.78	0.86	0.86	0.72	0.74	0.72	0.74
	50×20	0.36	0.32	0.34	**0.12**	0.42	0.46	0.50	0.34	0.32	0.42	0.40
	Sum	1.18	1.10	1.22	**0.80**	1.20	1.32	1.36	1.06	1.06	1.14	1.14
0.05	10×5	0.51	0.54	0.59	0.49	**0.38**	0.61	0.54	0.45	0.62	0.46	0.50
	30×10	2.57	2.60	2.64	2.81	2.36	2.38	2.54	2.82	2.39	2.76	**2.32**
	50×20	1.17	1.12	1.30	1.15	1.17	1.16	1.13	1.11	1.11	1.21	1.09
	Sum	4.24	4.26	4.52	4.45	3.91	4.14	4.22	4.38	4.11	4.43	**3.91**
0.1	10×5	0.59	0.65	0.55	0.42	0.64	**0.36**	0.70	0.48	0.44	0.66	0.37
	30×10	2.14	2.21	2.12	2.45	2.16	2.41	2.04	2.22	2.03	2.37	2.04
	50×20	0.94	0.91	1.19	1.02	1.02	1.05	0.99	1.06	1.04	1.04	0.82
	Sum	3.67	3.77	3.86	3.90	3.82	3.82	3.73	3.76	3.51	4.07	**3.23**
0.5	10×5	0.28	0.38	0.33	0.32	0.32	**0.25**	0.40	0.42	0.37	0.28	0.37
	30×10	1.85	1.70	2.06	2.24	2.07	2.09	1.93	1.93	1.87	1.90	**1.51**
	50×20	0.83	0.83	0.78	0.81	0.83	0.80	0.90	0.96	0.89	0.89	0.63
	Sum	2.96	2.91	3.17	3.38	3.22	3.15	3.23	3.31	3.13	3.07	**2.52**
1.0	10×5	0.30	0.28	**0.16**	0.24	0.27	0.26	0.40	0.32	0.21	0.22	0.37
	30×10	2.15	2.09	2.25	2.05	2.15	2.30	**1.88**	2.07	2.15	1.94	2.03
	50×20	0.93	0.81	0.83	0.90	0.82	0.89	0.84	0.90	0.88	0.83	0.67
	Sum	3.38	3.19	3.25	3.18	3.24	3.45	3.11	3.29	3.24	**2.98**	3.07

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

Acknowledgements

This work was supported in part by INTAS (project 03-51-5501).

References

[1] Karacapilidis, N.I. and Pappis, C.P., Production Planning and Control in Textile Industry: A Case Study, Computers in Industry, Vol.30, No.2, pp.127–144, 1996.

[2] Agnetis, A., Pacifici, A., Rossi, F., Lucertini, M., Nicoletti, S., Nicolò, F., Oriolo, G., Pacciarelli D., and Pesaro, E., Scheduling of Flexible Flow lines in an Automobile Assembly Plant, European Journal of Operational Research, Vol.97, No.2, pp.348–362, 1997.

[3] Alisantoso, D., Khoo, L.P., and Jiang, P.Y., An Immune Algorithm Approach to the Scheduling of a Flexible PCB Flow Shop, The International Journal of Advanced Manufacturing Technology, Vol.22, No.11–12, pp.819–827, 2003.

[4] Gupta, J.N.D., Krüger, K., Lauff, V., Werner, F., and Sotskov, Y.N., Heuristics for Hybrid Flow Shops with Controllable Processing Times and Assignable due Dates, Computers & Operations Research, Vol.29, No.10, pp.1417–1439, 2002.

[5] Wang, W., and Hunsucker, J.L., An Evaluation of the CDS Heuristic in Flow Shops with Multiple Processors, Journal of the Chinese Institute of Industrial Engineers, Vol.20, No.3, pp.295–304, 2003.

[6] Linn, R. and Zhang, W., Hybrid Flow Shop Scheduling: A Survey, Computers & Industrial Engineering, Vol.37, No.1–2, pp.57–61, 1999.

[7] Wang, H., Flexible Flow Shop Scheduling: Optimum, Heuristics, and Artificial Intelligence Solutions, Expert Systems, Vol.22, No.2, pp.78–85, 2005.

[8] Arthanari, T.S. and Ramamurthy, K.G., An Extension of Two Machines Sequencing Problem, Opsearch, Vol.8, No.1, pp.10–22, 1971.

[9] Salvador, M.S., A Solution to a Special Case of Flow Shop Scheduling Problems, in: Elmaghraby SE (ed.), Symposium on the Theory of Scheduling and Applications, Springer, New York, pp.83–91, 1973.

[10] Brah, S.A. and Hunsucker, J.L., Branch and Bound Algorithm for the Flow Shop with Multiple Processors, European Journal of Operational Research, Vol.51, No.1, pp.88–99, 1991.

[11] Moursli, O. and Pochet, Y., Branch and bound algorithm for the hybrid flowshop, International Journal of Production Economics, Vol.64, No.1–3, pp.113–125, 2000.

[12] Gupta, J.N.D., A Functional Heuristic Algorithm for the Flowshop Scheduling Problem, Operations Research Quarterly, Vol.22, No.1, pp.39-47, 1971.

[13]Sriskandarajah, C. and Sethi, S.P., Scheduling Algorithms for Flexible Flowshops: Worst Case and Average Case Performance, European Journal of Operational Research, Vol.43, No.2, pp.143–160, 1989.

[14] Guinet, A., Solomon, M.M., Kedia, P.K., and Dussauchoy, A., A Computational Study of Heuristics for Two-stage Flexible Flowshops, International Journal of Production Research, Vol.34, No.5, pp.1399–1415, 1996.

[15] Gupta, J.N.D. and Tunc, E.A., Scheduling a Two-stage hybrid Flowshop with Separable Setup and Removal Times, European Journal of Operational Research, Vol.77, No.3, pp.415–428, 1994.

[16] Santos, D.L., Hunsucker, J.L., and Deal, D.E., An Evaluation of Sequencing Heuristics in Flow Shops with Multiple Processors, Computers & Industrial Engineering, Vol.30, No.4, pp.681-691, 1996.

[17] Nowicki, E., and Smutnicki, C., The Flow Shop with Parallel Machines: A Tabu Search Approach, European Journal of Operational Research, Vol.106, No.2–3, pp.226–253, 1998.

[18] Gourgand, M., Grangeon, N., and Norre, S., Metaheuristics for the Deterministic Hybrid Flow Shop Problem, In: Proceeding of the International Conference on Industrial Engineering and Production Management, IEPM'99, Glasgow, pp.136–145, 1999.

[19] Jin, Z., Yang, Z., and Ito, T., Metaheuristic Algorithms for the Multistage Hybrid Flowshop Scheduling Problem, International Journal of Production Economics, Vol.100, No.2, pp.322–334, 2006.

[20] Koulamas, C., Antony, S.R. and Jaen, R., A Survey of Simulated Annealing

Applications to Operations Research Problems, Omega International Journal of Management Science, Vol.22, No.1, pp.41–56, 1994.

[21] Youssef, H., Sait, S.M., and Adiche, H., Evolutionary Algorithms, Simulated Annealing and Tabu Search: a Comparative Study, Engineering Applications of Artificial Intelligence, Vol.14, No.2, pp.167–181, 2001.

[22] Baker, K.R., Introduction to Sequencing and Scheduling, John Wiley & Sons, New York, 1974.

[23] Palmer, D.S., Sequencing Jjobs Through a Multi-stage Process in the Minimum Total Time—A Quick Method of Obtaining a Near Optimum, Operational Research Quarterly, Vol.16, No.1, pp.101–107, 1965.

[24] Campbell, H.G., Dudek, R.A., and Smith, M.L., A Heuristic Algorithm for the n-job M-machine Sequencing Problem, Management Science, Vol.16, No.10, pp.B630–B637, 1970.

[25] Gupta, J.N.D., A Functional Heuristic Algorithm for the Flowshop Scheduling Problem, Operations Research Quarterly, Vol.22, No.1, pp.39–47, 1971.

[26] Dannenbring, D.G., An Evaluation of Flow Shop Sequencing Heuristics, Management Science, Vol.23, No.11, pp.1174-1182, 1977.

[27] Nawaz, M., Enscore, Jr. E.E., Ham, I., A Heuristic Algorithm for the M-machine, n-job Flow-shop Sequencing Problem, Omega International Journal of Management Science, Vol.11, No.1., pp. 91–95, 1983.

[28] Kirkpatrick, S., Gelatt, Jr. C.D., and Vecchi, M.P., Optimization by Simulated Annealing, Science, Vol.220, No.4598, pp.671–68, 1983.

[29]Lundy, M. and Mees, A., Convergence of an Annealing Algorithm, Mathematical Programming, Vol.34, No.1, pp.111–124, 1986.