IE NETWORK 2006

Sequencing Heuristics for Flexible Flow Shop Scheduling Problems with Unrelated Parallel Machines and Setup Times

Jitti Jungwattanakit[1], Manop Reodecha[1], Paveena Chaovalitwongse[1], and Frank Werner[2]

[1]Department of Industrial Engineering, Faculty of Engineering,
Chulalongkorn University, Phayathai Rd., Patumwan, Bangkok 10330
Tel: 0-2218-6855, 0-2218-6814-6, FAX: 0-2251-3969, Email: jitti.j@student.chula.ac.th
[2]Faculty of Mathematics, Otto-von-Guericke-University, P.O. Box 4120,
D-39016 Magdeburg, Germany Tel: Tel.: +49-391-6712025; fax: +49-391-6711171,
Email: frank.werner@mathematik.uni-magdeburg.de

Keywords : sequencing heuristics, flexible flow shop, unrelated parallel machines

Abstract

This paper presents an investigation of scheduling procedures to seek the minimum of a positively weighted convex sum of makespan and the number of tardy jobs in a static flexible flow shop scheduling environment. The flexible flow shop problem is a scheduling of jobs in a flow shop environment consisting of series of production stages, some of which may have only one machine, but at least one stage must have multiple machines. In addition, sequence - and machine - dependent setup times are considered. No preemption of jobs is allowed. Some dispatching rules and flow shop makespan heuristics are adapted for sequencing in the flexible flow shop problem. The improvement heuristic algorithms such as shift and pairwise interchange algorithms have been proposed. Upon these heuristics, the first stage sequence is generated by using them. The first stage sequence will now be used, in conjunction with either permutation or FIFO rules, to construct a schedule for the problem. The solution is then set equal to the best function value obtained by both rules. The performance of the heuristics is compared relative to the optimal solution on a set of small-scale test problems.

1. Introduction

This paper is primarily concerned with the scheduling problem occurring in the production industries such as the glass-container (Paul, [1]), rubber (Yanney and Kuo, [2]), photographic film (Tsubone, Ohba, Takamuki, and Miyake, [3]), steel (Finke and Medeiros, [4]), textile, and food industries. These industries are established as multistage production flow shop facilities where a production stage may be made up of parallel machines known as flexible flow shop or hybrid flow shop environment, that is, it is a generalization of the classical flow shop model. There are *k* stages and some stages have only one machine, but at least one stage must have multiple machines such that all jobs have to pass through a number of stages in the same order. A machine can process at most one job at a time and a job can be processed by at most one machine at a time. Preemption of processing is not allowed. It consists of assigning jobs to machines at each stage and sequencing the jobs assigned to the same machine so that some certain optimality criteria are minimized.

Although the flexible flow shop problem has been widely studied, most of the studies related to flexible flow shop problems are concentrated on problems with identical processors, see for instance, Gupta, Krüger, Lauff, Werner and Sotskov [5], Alisantoso, Khoo, and Jiang [6], Lin and Liao [7] and Wang and Hunsucker [8]. However, 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 flexible 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. In this paper, both sequence- and machine-dependent setup time restrictions are taken into account as well.

Since most scheduling problems are combinatorial optimization problems which are too difficult to be solved optimally, heuristic methods are used to obtain good solutions in acceptable times. Consequently, the purpose of this paper is to present the constructive heuristics based on both the simple dispatching rules such as the Shortage Processing Time (SPT) and the Longest Processing Time (LPT) rules and the flow shop makespan heuristics such as the Palmer [9], Campbell, Dudek, and Smith [10], Gupta [11], and Dannenbring [12] algorithms as well as Nawaz, Enscore and Ham [13] algorithm adapted to solve the flexible flow shop problem with unrelated parallel machines and sequence-dependent setup times. In addition, the criteria of makespan and the number of tardy jobs have been receiving considerable attention in this paper. One reason for this consideration is the increasing pressure of high competition while customers expect ordered goods to be delivered on time. The remainder of this paper is structured as follows: in Section 2, the problem under study is explained. Section 3 shows the heuristic algorithms. The computational results are explained in Section 4. The conclusions are shown in Section 5.

2. Statement of the Problem

The flexible flow shop system is defined by the 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 a set $M = \{M^1,\ldots, M^k\}$. 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_{ij}^t of job j on machine i at stage t is equal to ps_j^t / v_{ij}^t, where ps_j^t is the standard processing time of job j at stage t, and v_{ij}^t 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) a job cannot be processed before its completion of the previous operation; (3) every machine can process only one operation at a time; (4) operations have to be realized sequentially, without overlapping between stages; (5) 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_{ij}^t 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_{lj}^t 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

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

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, $(0 \leq \lambda \leq 1)$.

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 [14].

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 $v_{ij}^{/t}$ and setup time $s_{lj}^{/t}$ 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 $ps_j^t / v_{ij}^{/t}$ plus the representative of the setup time $s_{lj}^{/t}$. 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 (Baker [15], and Pinedo and Chao [16]).

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 [9] 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 [10] 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 [11] provides an algorithm denoted by GUP, in a similar manner as algorithm PAL by using a different slope index and schedules the jobs according to the slope order. Dannenbring [12] 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. Its purpose is to provide good and quick solutions.

Nawaz, Enscore and Ham [13] 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 that 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 uses are 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) without due date considerations. In this section, some fast improvement heuristics will be investigated to improve the overall function value by concerning mainly the due date criterion.

In order to find a satisfactory solution of the due date problem, the 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 that the tardy jobs are swapped to the different job positions left and right in both random two positions (denoted by letter "2") and all other positions (denoted by letter "A"). The best schedule among the generated neighbors is then taken as the result.

4. Computational Results

Firstly, the overall constructive algorithms and different 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 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 between three to seven jobs. For all problem sizes, we tested instances with $\lambda \in \{0, 0.001, 0.005, 0.01\ 0.05, 0.1, 0.5, \text{and } 1\}$ 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 [17] and is as follows:

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

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 constructive algorithm from the optimal solution obtained by formulating the 0-1 mixed linear integer programming and running a commercial mathematical programming software, CPLEX 8.0.0 and AMPL, with an Intel Pentium 4 2.00GHz CPU; however, the CPU time is limited at most 2 hours.

Table 1 average overall performance of constructive algorithms and improvement heuristics.

λ	Problem size	CA	2-SM	A-SM	2-PI	A-PI	λ	Problem size	CA	2-SM	A-SM	2-PI	A-PI
0	3 jobs	0.304[a]	0.067	0.067	0.063	**0.054**	0.05	3 jobs	8.022	3.790	3.741	3.948	**3.527**
	4 jobs	0.579	0.229	**0.225**	0.267	0.233		4 jobs	9.750	4.382	3.736	5.153	**3.645**
	5 jobs	0.804	0.300	0.283	0.333	**0.213**		5 jobs	9.200	2.945	2.295	3.680	**2.134**
	6 jobs	1.083	0.479	0.438	0.513	**0.413**		6 jobs	10.681	3.845	3.231	4.698	**2.871**
	7 jobs	1.317	0.550	0.475	0.642	**0.433**		7 jobs	7.460	2.096	**1.886**	3.204	2.058
	Sum	4.088	1.625	1.488	1.817	**1.346**		Sum	45.113	17.058	14.889	20.683	**14.235**
0.001	3 jobs	36.310[b]	6.490	6.240	7.140	**5.860**	0.1	3 jobs	7.742	3.828	3.778	4.012	**3.603**
	4 jobs	60.680	25.110	**25.030**	29.250	26.820		4 jobs	9.571	4.227	3.839	4.956	**3.620**
	5 jobs	25.550	5.870	4.570	5.820	**3.890**		5 jobs	8.915	2.778	2.394	3.543	**2.139**
	6 jobs	64.610	35.800	31.790	30.140	**25.180**		6 jobs	10.823	3.712	3.306	4.735	**2.815**
	7 jobs	35.250	8.620	6.820	10.870	**6.310**		7 jobs	6.938	1.906	1.774	2.457	**1.455**
	Sum	222.400	81.890	74.450	83.220	**68.060**		Sum	43.989	16.451	15.091	19.703	**13.632**
0.005	3 jobs	14.640	4.680	4.590	4.800	**4.340**	0.5	3 jobs	7.695	4.011	3.961	4.178	**3.819**
	4 jobs	21.840	9.020	**8.650**	9.850	8.870		4 jobs	9.396	3.647	3.429	4.484	**3.110**
	5 jobs	16.876	4.347	3.565	4.651	**3.310**		5 jobs	8.552	2.621	2.251	3.777	**1.876**
	6 jobs	24.810	10.770	9.310	10.210	**7.580**		6 jobs	10.974	3.972	3.557	5.494	**3.022**
	7 jobs	18.720	6.330	5.630	7.590	**5.280**		7 jobs	7.253	2.426	2.130	2.983	**1.746**
	Sum	96.886	35.147	31.745	37.101	**29.380**		Sum	43.870	16.677	15.328	20.916	**13.573**
0.01	3 jobs	10.904	4.133	4.088	4.242	**3.875**	1.0	3 jobs	7.811	4.156	4.030	4.318	**3.890**
	4 jobs	15.275	6.355	**5.888**	6.762	6.010		4 jobs	9.381	3.596	3.362	4.527	**3.081**
	5 jobs	13.216	3.617	3.059	4.114	**2.963**		5 jobs	9.512	3.541	3.276	4.496	**2.871**
	6 jobs	17.330	7.140	5.870	7.180	**4.850**		6 jobs	11.697	4.245	3.802	5.767	**3.341**
	7 jobs	13.000	4.150	3.660	5.110	**3.510**		7 jobs	8.392	3.023	2.513	3.763	**2.256**
	Sum	69.725	25.395	22.565	27.408	**21.208**		Sum	46.793	18.561	16.983	22.871	**15.439**

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

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 the all pairwise interchange -based improvement heuristics are better than the others in general. Consequently, in this paper we have shown only the all pairwise interchange-based improvement heuristics. However, compared to 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 became better than the 2-PI algorithm.

Next, we present the constructive algorithms that are separated into four main groups. The first heuristic group is the simple dispatching rules such as SPT, LPT, ERD, EDD, MST, S/P, and HSE. The second heuristic group is 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.

The results for the constructive algorithms are given in Table 2. From these results it can be seen that the constructive 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, EDD, MST, S/P, and HSE) as well as the third heuristic group improved from them.

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 slightly outperforms the other dispatching rules for $\lambda \leq 0.01$, and the LPT rule is better than the other rules for 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).

Table 2 Average performance of constructive algorithms.

λ	Problem size	SPT	LPT	ERD	EDD	MST	S/P	HSE	λ	Problem size	PAL	CDS	GUP	DAN	NEH
0	3 jobs	*0.250*[a]	0.300	0.350	0.400	0.500	0.450	0.350	0	3 jobs	0.250	0.250	0.200	0.250	*0.100*
	4 jobs	*0.350*	0.750	0.400	0.550	0.700	0.650	0.550		4 jobs	0.750	0.650	0.700	0.650	*0.250*
	5 jobs	0.800	*0.700*	0.950	1.200	1.200	1.200	1.050		5 jobs	0.800	0.500	0.600	0.500	***0.150***
	6 jobs	*0.800*	1.200	1.150	1.450	1.550	1.400	0.950		6 jobs	1.000	0.900	1.150	1.000	*0.450*
	7 jobs	*1.050*	1.150	1.900	1.900	1.800	1.750	1.350		7 jobs	1.300	0.900	1.200	1.050	*0.450*
	Sum	*3.250*	4.100	4.750	5.500	5.750	5.450	4.250		Sum	4.100	3.200	3.850	3.450	*1.400*
0.001	3 jobs	29.200[b]	45.270	50.920	27.560	53.020	38.570	*25.240*	0.001	3 jobs	33.970	45.810	30.860	46.660	*8.610*
	4 jobs	*38.000*	88.400	40.000	45.000	58.100	55.900	38.100		4 jobs	93.000	75.000	77.900	81.400	*37.500*
	5 jobs	27.970	*23.570*	30.770	38.670	38.360	38.020	27.850		5 jobs	26.530	16.310	18.310	14.560	*5.700*
	6 jobs	61.100	97.960	79.690	70.590	72.780	*51.980*	61.940		6 jobs	59.110	54.220	79.980	58.680	*27.320*
	7 jobs	49.410	*23.960*	58.860	39.380	39.710	36.480	48.620		7 jobs	37.330	20.230	31.270	30.550	*7.200*
	Sum	205.68	279.16	260.24	221.20	261.97	220.95	*201.75*		Sum	249.94	211.57	238.32	231.85	*86.330*
0.005	3 jobs	14.530	14.220	15.210	15.430	23.490	19.510	*12.070*	0.005	3 jobs	12.060	15.520	11.060	17.170	*5.370*
	4 jobs	*14.990*	29.860	16.010	19.530	26.260	24.680	15.050		4 jobs	28.720	23.350	26.260	24.870	*12.480*
	5 jobs	18.650	*15.180*	19.590	24.850	24.710	24.180	18.930		5 jobs	16.960	11.570	12.530	11.600	*3.760*
	6 jobs	*22.870*	31.400	29.030	30.290	31.980	25.780	22.870		6 jobs	23.610	20.560	26.820	23.390	*9.070*
	7 jobs	22.910	*14.180*	25.420	25.530	25.720	22.790	22.230		7 jobs	17.900	12.040	15.080	15.840	*4.970*
	Sum	93.950	104.84	105.26	115.63	132.16	116.94	*91.150*		Sum	99.250	83.040	91.750	92.870	*35.650*
0.01	3 jobs	11.968	9.820	*9.715*	11.908	17.926	15.258	9.950	0.01	3 jobs	8.342	11.076	7.718	12.784	*4.379*
	4 jobs	*11.270*	20.470	12.090	14.820	19.690	18.400	11.450		4 jobs	18.570	14.720	17.620	16.010	*8.180*
	5 jobs	14.613	*11.884*	14.834	18.888	19.050	18.452	14.974		5 jobs	13.074	9.574	10.168	10.180	*2.899*
	6 jobs	17.410	20.020	19.910	21.350	22.700	18.620	*16.880*		6 jobs	16.970	14.660	17.120	16.710	*5.680*
	7 jobs	16.630	*10.530*	16.970	18.930	18.950	16.200	15.870		7 jobs	11.240	8.170	9.560	10.020	*2.900*
	Sum	71.891	72.724	73.519	85.896	98.316	86.930	*69.124*		Sum	68.196	58.200	62.186	65.704	*24.038*
0.05	3 jobs	10.312	7.048	*5.450*	9.066	13.451	11.831	8.747	0.05	3 jobs	5.716	7.828	4.790	8.760	*3.267*
	4 jobs	*8.329*	12.651	9.325	10.469	13.761	12.397	8.415		4 jobs	10.349	7.778	10.437	8.568	*4.521*
	5 jobs	10.565	*8.846*	9.230	12.322	12.955	12.449	10.504		5 jobs	8.811	7.205	7.318	8.418	*1.778*
	6 jobs	13.236	*9.344*	12.157	12.929	13.703	11.504	12.112		6 jobs	11.348	9.563	8.533	10.493	*3.256*
	7 jobs	11.239	*6.291*	9.148	13.126	11.553	8.770	10.363		7 jobs	5.086	3.736	4.956	4.598	***0.650***
	Sum	53.681	*44.180*	45.310	57.912	65.423	56.951	50.141		Sum	41.310	36.110	36.034	40.837	*13.472*
0.1	3 jobs	10.185	6.828	*5.021*	8.851	12.889	11.400	8.686	0.1	3 jobs	5.463	7.569	4.584	8.283	*3.149*
	4 jobs	*8.614*	12.301	9.656	10.519	13.666	12.175	8.622		4 jobs	9.696	7.361	9.869	8.056	*4.315*
	5 jobs	10.419	*8.705*	8.828	11.719	12.349	11.900	10.211		5 jobs	8.597	7.356	6.990	8.334	*1.574*
	6 jobs	14.250	*9.172*	12.429	12.876	13.635	11.587	12.856		6 jobs	11.833	9.851	8.455	10.752	***2.186***
	7 jobs	10.902	*5.906*	8.415	12.347	10.805	8.017	9.695		7 jobs	4.480	3.392	4.947	4.014	***0.335***
	Sum	54.370	*42.912*	44.349	56.312	63.344	55.079	50.070		Sum	40.069	35.529	34.845	39.439	***11.559***
0.5	3 jobs	10.282	6.760	*4.867*	8.865	12.603	11.219	8.836	0.5	3 jobs	5.423	7.551	4.614	8.086	*3.237*
	4 jobs	8.842	12.063	9.957	10.607	13.521	12.006	*8.784*		4 jobs	9.077	6.876	9.320	7.563	*4.137*
	5 jobs	10.192	8.531	*8.441*	11.191	11.426	11.092	9.854		5 jobs	8.316	7.300	6.632	8.102	*1.548*
	6 jobs	14.536	*8.607*	12.698	12.921	13.572	11.651	13.138		6 jobs	12.250	10.305	8.481	11.010	*2.514*
	7 jobs	11.036	*6.840*	8.413	12.769	10.534	8.330	9.975		7 jobs	4.868	4.107	5.113	4.662	*0.391*
	Sum	54.888	*42.801*	44.376	56.353	61.656	54.298	50.587		Sum	39.934	36.139	34.160	39.423	*11.827*
1.0	3 jobs	10.299	6.750	*4.852*	8.871	12.572	11.201	10.299	1.0	3 jobs	5.412	7.547	4.622	8.059	*3.247*
	4 jobs	*8.871*	12.038	10.002	10.625	13.500	11.991	8.871		4 jobs	9.004	6.820	9.245	7.494	*4.107*
	5 jobs	11.078	*9.458*	9.482	12.282	12.401	12.042	11.078		5 jobs	9.117	8.204	7.620	8.948	*2.441*
	6 jobs	15.408	*9.368*	13.365	13.417	14.166	12.145	15.408		6 jobs	12.819	10.902	9.013	11.603	*2.747*
	7 jobs	12.467	*7.834*	9.482	13.948	11.688	9.519	12.467		7 jobs	5.719	5.185	5.837	5.712	***0.843***
	Sum	58.123	*45.448*	47.183	59.143	64.327	56.898	58.123		Sum	42.071	38.658	36.337	41.816	*13.385*

Table 3 Average performance of improvement algorithms.

λ	Problem size	ISPT	ILPT	IERD	IEDD	IMST	IS/P	IHSE	IPAL	ICDS	IGUP	IDAN	INEH
0	3 jobs	***0.050***	***0.050***	***0.050***	***0.050***	***0.050***	***0.050***	***0.050***	***0.050***	***0.050***	***0.050***	***0.050***	0.100
	4 jobs	0.200	0.250	0.200	0.200	0.250	0.250	***0.150***	0.250	0.300	*0.200*	0.300	0.250
	5 jobs	0.250	0.250	0.250	0.200	0.300	0.200	***0.150***	0.250	0.200	***0.150***	0.200	***0.150***
	6 jobs	0.450	0.400	0.450	0.400	0.450	0.350	***0.250***	0.400	*0.350*	0.500	0.500	0.450
	7 jobs	0.350	0.500	0.450	0.400	0.500	0.300	***0.300***	0.550	0.450	0.500	*0.450*	*0.450*
	Sum	1.300	1.450	1.400	1.250	1.550	1.150	***0.900***	1.500	*1.350*	1.400	1.500	1.400
0.001	3 jobs	5.230	5.180	5.730	5.800	6.040	***4.950***	5.180	5.360	6.410	*5.140*	6.640	8.610
	4 jobs	25.900	32.700	***8.000***	26.400	26.200	25.900	25.900	32.800	32.800	*15.200*	32.700	37.500
	5 jobs	4.560	***2.320***	2.740	4.610	6.310	4.350	4.180	***2.440***	4.270	2.570	2.690	5.700
	6 jobs	17.860	33.860	36.630	36.110	35.860	***16.320***	17.770	20.220	21.660	20.010	***18.540***	27.320
	7 jobs	7.700	7.140	8.080	5.970	5.360	*3.460*	8.740	7.170	***1.260***	7.800	5.900	7.200
	Sum	61.250	81.200	61.180	78.890	79.770	*54.980*	61.770	67.990	66.400	***50.720***	66.470	86.330
0.005	3 jobs	4.120	3.800	4.350	4.410	4.660	*3.570*	3.800	3.980	5.030	***3.750***	5.250	5.370
	4 jobs	8.530	9.370	***5.270***	9.540	9.160	8.490	8.520	9.650	9.540	***6.560***	9.390	12.480
	5 jobs	3.670	***2.380***	3.190	3.730	4.570	3.590	3.290	***2.590***	3.420	2.690	2.840	3.760
	6 jobs	6.070	8.990	9.650	9.200	8.890	***4.680***	5.940	6.920	8.000	7.240	***6.340***	9.070
	7 jobs	7.030	4.960	5.400	6.180	5.250	*3.270*	7.680	4.720	***2.800***	6.350	4.780	4.970
	Sum	29.420	29.500	27.860	33.060	32.530	***23.600***	29.230	27.860	28.790	***26.590***	28.600	35.650
0.01	3 jobs	3.780	3.371	3.922	3.989	4.233	***3.143***	3.371	3.554	4.600	***3.326***	4.828	4.379
	4 jobs	5.750	5.810	***4.320***	6.970	6.490	5.700	5.730	6.300	6.010	***5.000***	5.860	8.180
	5 jobs	3.201	***2.334***	3.287	3.288	3.805	3.172	2.822	***2.375***	2.947	2.626	2.805	2.899
	6 jobs	4.480	5.540	5.920	5.430	5.150	***3.000***	4.270	4.570	5.640	4.370	***4.210***	5.680
	7 jobs	5.180	2.920	3.350	5.120	4.040	*2.120*	5.780	2.650	***1.790***	3.530	2.690	2.900
	Sum	22.391	19.975	20.799	24.797	23.718	***17.135***	21.973	19.449	20.987	***18.852***	20.393	24.038
0.05	3 jobs	3.609	3.388	3.499	3.837	3.977	***2.887***	3.115	3.299	4.071	***3.164***	4.206	3.267
	4 jobs	3.535	3.366	3.728	5.296	4.177	***3.021***	3.489	***3.062***	3.212	3.160	3.175	4.521
	5 jobs	2.103	2.576	2.670	2.421	2.412	1.824	***1.725***	2.092	1.874	2.292	**1.842**	***1.778***
	6 jobs	2.747	*2.516*	3.400	3.579	2.816	2.753	2.731	2.622	3.117	2.490	***2.429***	3.256
	7 jobs	4.066	*0.960*	2.158	4.271	2.787	1.110	3.986	1.184	1.074	1.142	1.308	***0.650***
	Sum	16.060	12.806	15.455	19.404	16.169	***11.595***	15.046	12.259	13.348	*12.248*	12.960	13.472
0.1	3 jobs	3.662	3.343	3.714	4.131	4.019	***2.929***	3.157	3.340	4.228	3.321	4.247	*3.149*
	4 jobs	3.690	3.529	4.081	5.059	3.944	***2.763***	3.643	*2.900*	3.156	3.253	3.110	4.315
	5 jobs	2.333	2.417	2.919	2.836	2.502	2.059	*1.954*	2.007	***1.237***	2.110	1.723	1.574
	6 jobs	2.579	2.656	3.326	3.442	*2.417*	2.970	2.807	2.648	3.364	2.842	2.543	***2.186***
	7 jobs	1.195	*0.615*	2.055	3.832	2.865	0.900	1.194	1.112	0.984	1.098	1.280	***0.335***
	Sum	13.459	12.560	16.095	19.300	15.747	*11.621*	12.755	12.007	12.969	12.624	12.903	***11.559***
0.5	3 jobs	3.902	3.475	3.913	4.394	4.249	***3.149***	3.387	3.535	4.514	3.634	4.441	*3.237*
	4 jobs	3.053	2.776	3.414	4.362	3.666	*2.394*	3.005	***2.372***	2.840	2.586	2.709	4.137
	5 jobs	2.074	2.215	2.463	2.531	2.164	1.799	*1.696*	1.552	***1.087***	1.841	1.547	1.548
	6 jobs	3.244	*2.620*	3.422	3.215	2.941	2.911	2.876	2.716	3.948	***2.495***	3.358	2.514
	7 jobs	1.671	1.581	2.063	4.655	2.738	*1.120*	1.438	1.135	1.488	1.376	1.293	***0.391***
	Sum	13.944	12.667	15.275	19.157	15.758	*11.373*	12.402	***11.310***	13.877	11.932	13.348	11.827
1.0	3 jobs	3.937	3.492	3.942	4.432	4.283	***3.176***	3.937	3.553	4.545	3.671	4.460	*3.247*
	4 jobs	3.044	2.745	3.395	4.341	3.634	*2.352*	3.044	***2.331***	2.787	2.532	2.656	4.107
	5 jobs	3.026	3.192	3.448	3.533	3.138	*2.751*	3.026	2.676	***2.082***	2.803	2.340	2.441
	6 jobs	3.497	*2.811*	3.961	3.813	3.354	3.095	3.497	3.004	4.235	***2.507***	3.575	2.747
	7 jobs	2.197	2.139	2.770	5.045	3.306	*1.665*	2.197	1.451	1.831	1.896	1.731	***0.843***
	Sum	15.701	14.379	17.516	21.164	17.715	*13.039*	15.701	***13.015***	15.480	13.409	14.762	13.385

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

[b] average percentage deviation for $\lambda > 0$

When we apply the polynomial improvement ('interchange') algorithm (denoted by the letter "I" first) to the solutions obtained by the dispatching rules and adapted flow shop makespan heuristics, we have found that the quality of the solution in terms of the deviation from the optimal solution can be improved by about 70 percent except for the NEH rule. However, the improvement of heuristics from the adapted flow shop makespan heuristics in the heuristic Group IV is better than the improvement of heuristics from the dispatching rules in the heuristic Group III (since for most of the problems, there is an substantial portion in the objective function value resulting from the makespan). In Group III, the IS/P rule outperforms the others, whereas in Group IV, the algorithms can slightly improve the solutions obtained by the NEH algorithm.

5. Conclusions

In this paper, some constructive algorithms have been investigated for minimizing a convex combination of makespan and the number of tardy jobs for the flexible 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. The constructive algorithms are compared to the optimal 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 the good improvement algorithm. However, there are some deviations from the optimal value, so in the further research, the iterative heuristics such as simulated annealing, tabu search and genetic algorithms will be able to study in order to find the best solution.

References

[1] R.J.A. Paul. Production scheduling problem in the glass-container industry. Operations Research. 27(2): page 290–302, 1979.

[2] J.D. Yanney, and W. Kuo. A practical approach to scheduling a multistage, multiprocessor flow-shop problem. International Journal of Production Research. 27(10), page 1733–1742, 1989.

[3] H. Tsubone, M. Ohba, H. Takamuki, and Y. Miyake. Production scheduling system in the hybrid flow shop. Omega. 21(2): page 205-214, 1993.

[4] D.A. Finke and D.J. Medeiros. Shop scheduling using tabu search and simulation. Proceedings of the 2002 Winter Simulation Conference. page 1013-1017, 2002.

[5] J.N.D. Gupta, K. Krüger, V. Lauff, F. Werner, and Y.N. Sotskov. Heuristics for hybrid flow shops with controllable processing times. Computers and Operations Research. 29(10): page 1417–1439, 2002.

[6] D. Alisantoso, L.P. Khoo, and P.Y. Jiang. An immune algorithm approach to the scheduling of a PCB flexible flow shop. The International Journal of Advanced Manufacturing Technology. 22(11–12): page 819–827, 2003.

[7] H.T. Lin, and C.J. Liao. A case study in a two-stage hybrid flow shop with setup time and dedicated machines. International Journal of Production Economics. 86(2): page 133–143, 2003.

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

[9] D.S. Palmer. Sequencing jobs through a multi-stage process in the minimum total time--a quick method of obtaining a near optimum. Operational Research Quarterly. 16(1): page 101-107, 1965.

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

[11] J.N.D. Gupta. A functional heuristic algorithm for the flow-shop scheduling problem. Operations Research Quarterly. 22(1): page 39-47, 1971.

[12] D.G. Dannenbring. An evaluation of flow shop sequencing heuristics. Management Science . 23(11): page 1174-1182,. 1977.

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

[14] D.L. Santos, J.L. Hunsucker and D.E. Deal. An evaluation of sequencing heuristics in flow shops with multiple processors. Computers & Industrial Engineering. 30(4): page 681-691, 1996.

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

[16] M. Pinedo, and X. Chao. Operations scheduling with applications in manufacturing and services. Irwin/McGraw-Hill, New York, 1999.

[17] C. Rajendran, and H. Ziegler. 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): page 513–522, 2003.