ISSN 1992-6669

INTERNATIONAL JOURNAL of Computational Science

Volume 1
June 2007
Number 2

Editor-in-Chief
Kin Keung Lai

International Journal of Computational Science
1992-6669 (Print) 1992-6677 (Online) © Global Information Publisher
2007, Vol. 1, No. 2, 204-214

Constructive and Tabu Search Algorithms for Hybrid Flow Shop 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, Bangkok 10330 Thailand
jitti.j@student.chula.ac.th
[2] Faculty of Mathematics, Otto-von-Guericke-University,
P.O. Box 4120, D-39016 Magdeburg, Germany
frank.werner@mathematik.uni-magdeburg.de

Abstract. This paper investigates scheduling heuristics to seek the minimum of a positively weighted convex sum of makespan and the number of tardy jobs in a hybrid flow shop environment where at least one production stage is made up of unrelated parallel machines. Sequence-and machine-dependent setup times are considered. The problem is a combinatorial optimization problem which is difficult to be solved optimally, 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, a metaheuristic tabu search algorithm is proposed. Some tabu search parameters are briefly discussed. 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

This article is concerned with an industrial scheduling problem, which belongs to the class of NP-hard combinatorial optimization problems. Therefore, the search for efficient methods providing a

* Corresponding Author. Tel: +662-218-6815-6, Fax: +662-251-3969, Email: manop@eng.chula.ac.th.

good feasible solution continues to be a challenge. The heuristics concerned can be classified into two types: constructive and iterative heuristic algorithms. A constructive algorithm generates a single solution. Alternatively, one may also generate several solutions in parallel from which the best one is chosen as the heuristic solution. The interest in iterative algorithms results from the difficulty of solving real large-size problems. This study deals with one of the most popular iterative algorithms known as a tabu search (TS) algorithm, originally proposed by [1]. It has been often used for the approximate solution of combinatorial problems [2] and has been successfully applied to problems in many different areas [3].

In this paper, heuristics are 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, the production unit of which is characterized by a multi-stage manufacturing process with multiple production units per stage, called the hybrid flow shop (HFS) problem. Most textile companies are ageing while the technology changes rapidly. It is common to find more modern machines running side by side with older machines. Hence, these companies own machines of different ages, performing the same operations as the newer ones, but would generally require a longer operating time for the same operation. In addition, sequence-dependent setup times incur when machines often have to be reconfigured between jobs. 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.

The hybrid flow shop scheduling problem has attracted many researchers due to two main reasons [4]. Firstly, it is a category of problems which is difficult to solve [5]. Secondly, it finds many real-world applications. Although such a problem has been widely studied, most studies concentrate on problems with identical processors [6] – [9]. In this paper, the hybrid flow shop problem with unrelated parallel machines and sequence-dependent setup times is considered. The rest of this paper is organized as follows. The problem 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 as follows: a set of jobs $J = \{1,\ldots, j,\ldots, n\}$ has to be sequenced in a flow shop environment with k stages. For each stage t, a set $M^t = \{1,\ldots, i,\ldots, m^t\}$ of m^t unrelated machines is considered. Each job 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. Preemption is not permitted. Each job is processed by at most one machine at each stage without overlapping between the stages. 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 machine i at stage t. In addition, the setup times considered are classified into two types, namely a machine-dependent setup time and a sequence-dependent setup time. A setup time of a job is ma-

chine-dependent if it depends on the machine to which the job is assigned. It is assumed to occur only when a job is the first job assigned to this machine. ch^t_{ij} denotes the length of the machine-dependent setup time of job j if job j is the first job assigned to machine i at stage t. A sequence-dependent setup time depends on the job just completed on that machine. s^t_{lj} denotes the time needed to change over 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 assumed to be 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:

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

where C_{max} is the makespan, which corresponds 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

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 [10].

Firstly, a job sequence is determined according to a particular sequencing rule, see 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 two approaches for this subproblem. The first way is to order the jobs for the other stages, i.e. from stage two to stage k, according to their completion times at the previous stage, called the FIFO (First-In-First-Out) rule. The second way is to sequence the jobs for the other stages by using the same job sequence as for the first stage, which is called the permutation rule.

3.1 Constructive Heuristics

In order to determine the job sequence for the first stage, 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.

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 (ERD) rule is equivalent to the well-known FIFO rule. The Earliest Due Date (EDD) rule schedules the jobs according to non-decreasing due dates of the jobs. The Minimum Slack Time (MST) 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 [11].

Palmer's heuristic [12] 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. 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 [13] develop one of the most significant heuristic methods for the makespan problem known as CDS algorithm. In so doing, $k - 1$ sub-problems are created and Johnson's rule is applied to each of the sub-problems. Gupta [14] 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 [15] develops a method, denoted by DAN. Furthermore, the CDS and PAL algorithms are also exhibited. Nawaz, Enscore and Ham [16] develop a method, called the NEH algorithm, which 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.

3.2 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 SPT and LPT) 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 π_r at position r, which is shifted to position i, while leaving all other relative job orders unchanged. The PI neighborhood exchanges a pair of chosen jobs π_r and π_i, while leaving all other jobs in the original positions. 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. 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. To escape from a local optimum, 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 a worse objective function value. Nevertheless, subsequent iterations may cause the search to move repeatedly back to the same local optimum. 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 control parameter of the TS algorithm. This is accomplished by keeping the attributes of the forbidden moves in a list, called the tabu list.

4.1 Choice of an Initial Solution

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 those obtained from the constructive algorithms and the fast improvement (SM, PI) heuristics.

5 Computational Results

Firstly, the constructive algorithms and different fast improvement heuristics are studied. The constructive algorithms are the simple dispatching rules (i.e., the SPT, LPT, ERD, EDD, MST and S/P rules) and the flow shop makespan heuristics (i.e., the PAL, CDS, GUP, DAN and NEH rules). Then, we applied the fast polynomial improvement heuristics based on the four variants discussed

in Section 3.2. We used problems with 10jobs×5stages, 30jobs×10stages, and 50jobs×20 stages and $\lambda \in \{0, 0.001, 0.005, 0.01\ 0.05, 0.1, 0.5, 1\}$ in the objective function. Ten different instances for each problem size have been run. An experiment was conducted to test with the following data: The standard processing times are generated uniformly from the interval [10,100]. The number of unrelated parallel for each stage is generated uniformly from the interval [1,5], but at least one stage is made up of parallel machines. The relative speeds are distributed uniformly in the interval [0.7,1.3]. Sequence- and machine-dependent setup times are generated uniformly from the interval [0,50]. 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 is similar to the approach presented by [17] and is as follows:

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

First, we have observed in our tests that the improvement heuristics from Section 3.2 improve the quality of the constructive algorithms by about 60 – 70 percent. In general, the A-PI algorithm should be selected as the improvement algorithm. Hence, we use in the following only the A-PI improvement heuristics.

Next, we tested the constructive algorithms that are separated into four main groups. The first heuristic group includes the simple dispatching rules, and the second heuristic group includes the flow shop makespan heuristics adapted. 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 the A-PI improvement heuristics (hereafter, these algorithms are denoted by the first letter "I" in front of the letters describing the heuristics of the first two groups). Among the simple dispatching rules (heuristic Group I), the SPT, LPT and ERD rules are good dispatching rules. However, in general the SPT 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 the 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 [18], where the results for small problem sizes are compared with the optimal solutions.

Thirdly, we studied the TS algorithm with a random initial solution. The favorable TS parameters, i.e., the number of neighbors (10 through 50, in step of 10), the neighborhood structure (PI and SM), and the size of tabu list (5, 10, 15, and 20) were investigated. 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. In Table 1, we give the average

Table 1. The effect of the tabu search parameters

λ	Problem	Number of Neighbors					Neighborhood Structures		Size of Tabu List			
	size	10	20	30	40	50	PI	SM	5	10	15	20
0	10×5	0.029[a]	**0.017**	0.033	0.050	0.083	**0.033**	0.052	0.057	**0.030**	0.040	0.043
	30×10	0.400	**0.242**	0.313	0.392	0.463	**0.337**	0.387	0.380	0.367	**0.347**	0.353
	50×20	**0.050**	0.146	0.346	0.533	0.625	**0.163**	0.517	0.380	0.347	**0.287**	0.347
	Sum	0.479	**0.404**	0.692	0.975	1.171	**0.533**	0.955	0.817	0.743	**0.673**	0.743
0.001	10×5	0.954[b]	0.989	**0.943**	1.477	3.281	**0.911**	2.146	2.254	1.152	**0.924**	1.786
	30×10	7.912	5.236	**4.940**	5.640	6.250	**5.923**	6.068	6.046	5.912	6.412	**5.612**
	50×20	0.981	**0.945**	2.040	3.409	4.250	**2.050**	2.600	2.511	**2.221**	2.339	2.228
	Sum	9.847	**7.170**	7.923	10.526	13.781	**8.884**	10.814	10.811	**9.285**	9.675	9.626
0.005	10×5	1.136	0.648	**0.618**	0.799	1.282	**0.826**	0.967	1.036	**0.707**	0.894	0.949
	30×10	6.057	3.942	**3.611**	4.134	4.195	4.650	**4.125**	**4.325**	4.354	4.341	4.532
	50×20	1.875	**1.663**	2.623	3.493	4.099	**2.635**	2.867	2.837	2.746	2.711	**2.710**
	Sum	9.068	**6.253**	6.852	8.426	9.576	8.111	**7.959**	8.198	**7.807**	7.946	8.191
0.01	10×5	0.781	**0.419**	0.474	0.730	1.099	**0.667**	0.735	0.855	**0.534**	0.552	0.863
	30×10	5.264	3.653	**3.549**	3.931	4.390	4.413	**3.903**	**4.097**	4.165	4.238	4.131
	50×20	2.171	**1.807**	2.783	3.744	4.161	**2.759**	3.108	2.933	3.134	**2.687**	2.979
	Sum	8.216	**5.879**	6.806	8.405	9.650	7.839	**7.746**	7.885	7.833	**7.477**	7.973
0.05	10×5	0.535	0.176	**0.166**	0.191	0.332	0.379	**0.181**	0.261	**0.239**	0.257	0.364
	30×10	4.585	3.727	**3.632**	3.777	4.119	4.298	**3.638**	3.939	4.019	3.989	**3.926**
	50×20	2.410	**1.734**	2.793	3.338	3.905	2.839	**2.834**	**2.667**	2.905	2.903	2.869
	Sum	7.530	**5.637**	6.591	7.306	8.356	7.516	**6.653**	**6.867**	7.163	7.149	7.159
0.1	10×5	0.381	**0.119**	0.158	0.154	0.344	0.324	**0.138**	0.278	**0.150**	0.230	0.267
	30×10	4.067	3.542	**3.458**	3.773	3.714	4.004	**3.418**	3.684	3.769	**3.682**	3.708
	50×20	2.174	**1.491**	2.313	2.925	3.555	**2.407**	2.576	2.513	2.444	**2.427**	2.582
	Sum	6.622	**5.152**	5.929	6.851	7.613	6.735	**6.132**	6.475	6.363	**6.339**	6.557
0.5	10×5	0.331	0.164	**0.108**	0.228	0.282	0.283	**0.162**	0.219	**0.190**	0.217	0.264
	30×10	3.705	**2.962**	3.168	3.182	3.569	3.561	**3.073**	3.379	**3.219**	3.375	3.295
	50×20	2.008	**1.304**	2.098	2.860	3.510	**2.310**	2.402	2.369	2.340	2.431	**2.283**
	Sum	6.044	**4.430**	5.374	6.269	7.360	6.154	**5.637**	5.967	**5.749**	6.023	5.842
1.0	10×5	0.358	**0.127**	0.152	0.218	0.327	0.290	**0.183**	0.290	**0.167**	0.207	0.283
	30×10	3.523	**2.805**	2.877	3.169	3.299	3.341	**2.928**	3.171	3.105	**3.085**	3.177
	50×20	2.129	**1.415**	2.321	2.861	3.551	**2.345**	2.566	2.434	2.451	2.521	**2.416**
	Sum	6.011	**4.347**	5.351	6.249	7.177	5.976	**5.677**	5.895	**5.723**	5.813	5.876

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

(absolute for $\lambda = 0$ resp. percentage for $\lambda > 0$) deviation of a particular algorithm from the best solution in these tests. 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, 20 and 30 non-tabu neighbors are good parameters, but generating 20 nontabu neighbors is the best variant. The PI moves are better than the 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 that of SM, it has been found that there is a statistically significant interaction between the neighborhood structure and the number of neighbors, that is, for 20 non-

Table 2. Average performance of tabu search algorithms with biased initial solutions

		Group I			Group II			Group III			Group IV		
λ	Problem size	SPTTS	LPTTS	ERDTS	PALTS	CDSTS	NEHTS	ISPTTS	ILPTTS	IERDTS	IPALTS	ICDSTS	INEHTS
0	10×5	0.020[a]	0.000	0.020	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.020	0.000
	30×10	0.220	0.220	0.260	0.260	0.340	0.180	0.280	0.220	0.220	0.220	0.300	0.160
	50×20	0.060	0.060	0.100	0.120	0.020	0.000	0.100	0.120	0.080	0.140	0.040	0.000
	Sum	0.300	0.280	0.380	0.380	0.360	0.180	0.380	0.340	0.300	0.360	0.360	0.160
0.001	10×5	0.014[b]	0.090	0.071	0.163	0.100	0.166	0.531	0.575	0.044	1.024	1.056	0.534
	30×10	3.674	4.649	4.377	4.142	3.715	3.413	4.107	3.829	4.898	4.915	4.934	3.421
	50×20	0.667	0.938	1.415	0.648	0.544	0.343	0.847	1.014	0.797	1.087	0.380	0.350
	Sum	4.355	5.677	5.863	4.953	4.359	3.922	5.485	5.418	5.739	7.026	6.370	4.305
0.005	10×5	0.306	0.984	0.912	0.615	1.179	1.076	0.573	1.279	1.044	0.832	1.103	0.721
	30×10	2.892	3.177	2.847	2.776	2.967	3.353	2.906	3.068	3.140	3.182	3.329	3.461
	50×20	1.363	1.223	1.491	1.395	0.876	0.540	0.932	1.745	2.025	2.089	0.676	0.543
	Sum	4.561	5.384	5.250	4.786	5.022	4.969	4.411	6.092	6.209	6.103	5.108	4.725
0.01	10×5	0.598	0.551	0.356	0.420	0.779	0.664	0.468	0.481	0.477	0.628	0.387	0.706
	30×10	3.063	3.562	2.373	3.490	2.851	3.068	3.824	3.346	2.963	3.274	3.109	2.994
	50×20	1.068	0.851	1.180	0.970	1.371	0.539	1.241	1.775	1.635	1.782	1.124	0.539
	Sum	4.729	4.964	3.909	4.880	5.001	4.271	5.533	5.602	5.075	5.684	4.620	4.239
0.05	10×5	0.039	0.033	0.029	0.039	0.029	0.042	0.076	0.052	0.057	0.059	0.054	0.029
	30×10	3.252	3.513	2.357	3.036	3.030	2.128	3.094	2.692	2.743	3.512	3.238	2.233
	50×20	1.064	0.959	1.146	1.083	1.421	0.491	1.027	1.146	1.700	1.531	0.891	0.491
	Sum	4.355	4.505	3.532	4.158	4.480	2.661	4.196	3.891	4.500	5.101	4.183	2.753
0.1	10×5	0.039	0.057	0.009	0.017	0.025	0.030	0.010	0.035	0.031	0.009	0.021	0.012
	30×10	2.698	2.633	1.947	2.515	2.295	1.839	2.569	2.628	2.107	2.682	2.356	1.832
	50×20	0.947	1.183	1.166	0.822	1.182	0.415	0.946	1.304	1.302	1.266	1.036	0.416
	Sum	3.685	3.872	3.122	3.354	3.502	2.285	3.524	3.966	3.440	3.957	3.413	2.260
0.5	10×5	0.061	0.015	0.055	0.006	0.039	0.039	0.027	0.035	0.061	0.019	0.041	0.039
	30×10	2.307	2.396	1.872	2.549	2.426	1.407	2.179	2.256	2.004	1.999	2.288	1.456
	50×20	0.938	1.047	1.228	0.904	1.306	0.327	0.927	1.249	1.172	1.495	0.876	0.314
	Sum	3.306	3.458	3.155	3.458	3.771	1.773	3.133	3.540	3.237	3.514	3.205	1.809
1.0	10×5	0.096	0.059	0.054	0.042	0.069	0.047	0.037	0.065	0.066	0.016	0.024	0.035
	30×10	2.264	2.517	1.808	2.787	2.468	1.741	2.366	2.433	2.082	2.654	2.181	1.747
	50×20	0.878	0.981	1.107	1.172	1.356	0.362	0.972	1.226	1.172	1.328	1.033	0.362
	Sum	3.237	3.557	2.969	4.000	3.893	2.150	3.375	3.724	3.320	3.998	3.237	2.143

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

tabu 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 has been found that a size of 10 and 15 works best, but a size 10 of the tabu list is recommended.

Finally, we used the recommended TS parameters to test the choice of an appropriate initial solution (some results are given in Table 2). 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 the experiment, we have found that there are no statistically significant differences in the different initial solutions for the problem sizes 10jobs×5stages and 30jobs×10stages, but it became statistically significantly different for the problem size 50jobs×20stages, in particular for $\lambda \geq 0.05$ we have found that algorithms INEHTS and NEHTS are better than the others. In general, algorithms INEHTS and NEHTS are good choices for the TS algorithm when using a biased initial solution. These results are consistent with previous studies in which the NEH algorithm is usually applied to provide the initial solution for an iterative algorithm such as a tabu search [19].

6 Conclusions

In this paper, first some constructive algorithms have 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 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 best known solutions. We have found that among the simple dispatching rules the SPT, LPT and ERD rules are good algorithms whereas among the flow shop makespan heuristics, the NEH algorithm is clearly superior to the other constructive algorithms. When applying 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 metaheuristic algorithms. Before we studied the influence of the initial solution on the performance of the TS algorithm, we tested the TS parameters, i.e., the number of neighbors, the neighborhood structure, and the 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 the tabu list should be selected as 10. For the recommended TS parameters, we have investigated the influence of the starting solution by using several constructive and improvement algorithms. The variants INEHTS and NEHTS can both be recommended in general.

Further research can be done to use other metaheuristic 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.

Acknowledgements

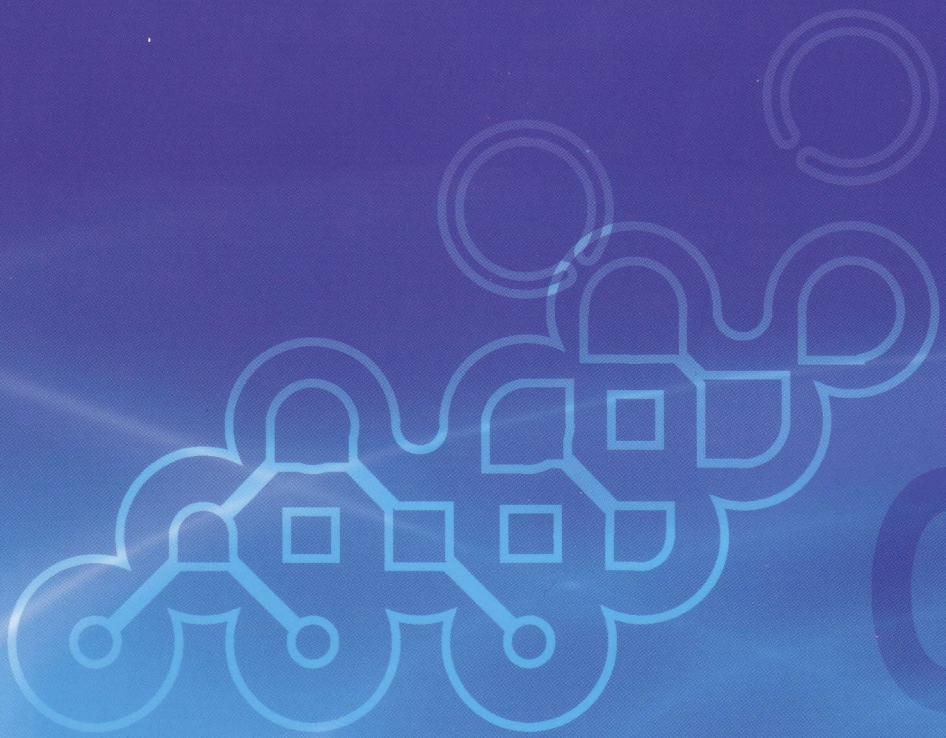
This work was supported in part by the Department of Industrial Engineering, Chulalongkorn University, and by INTAS (project 03-51-5501).

References

1. Glover, F.: Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research* **13** (1986) 533–549
2. Blum, C., Roli, A.: Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys* **35** (2003) 268–308
3. Glover, F., Laguna, M.: Tabu search. In C.R.Reeves (ed), *Modern Heuristic Techniques for Combinatorial Problems (UK: Blackwell Scientific Publications)* Oxford, chapter 3 (1993) 70-150
4. Wang, H.: Flexible Flow Shop Scheduling: Optimum, heuristics, and artificial intelligence solution. *Expert Systems* **22** (2005) 78 – 85
5. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the theory of NP-completeness CA. W.H. Freeman and company, San Francisco (1979)
6. Gupta, J.N.D., Krüger, K., Lauff, V., Werner, F., Sotskov, Y.N.: Heuristics for hybrid flow shops with controllable processing times and assignable due dates. *Computers & Operations Research* **29** (2002) 1417-1439
7. Alisantoso, D., Khoo, L.P., Jiang, P.Y.: An immune algorithm approach to the scheduling of a flexible PCB flow shop. *The International Journal of Advanced Manufacturing Technology* **22** (2003) 819-827
8. Lin, Hung-Tso., Liao, Ching-Jong.: A case study in a two-stage hybrid flow shop with setup time and dedicated machines. *International Journal of Production Economics* **86** (2003) 133-143
9. Wang, W., Hunsucker, J.L.: An evaluation of the CDS heuristic in flow shops with multiple processors. *Journal of the Chinese Institute of Industrial Engineers* **20** (2003) 295-304
10. Santos, D.L., Hunsucker, J.L., Deal, D.E.: An evaluation of sequencing heuristics in flow shops with multiple processors. *Computers & Industrial Engineering* **30** (1996) 681-691
11. Pinedo, M., and Chao, X.: *Operations scheduling with applications in manufacturing and services*. Irwin/McGraw-Hill, New York (1999)
12. Palmer, D.S.: Sequencing jobs through a multi-stage process in the minimum total time--a quick method of obtaining a near optimum. *Operations Research Quarterly* **16** (1965) 101–107
13. Campbell, H.G., Dudek, R.A., Smith, M.L.: A Heuristic algorithm for the n-Job m-Machine sequencing problem. *Management Science* **16**(1970) B630–B637
14. Gupta, J.N.D.: A functional heuristic algorithm for the flowshop scheduling problem. *Operations Research Quarterly* **22** (1971) 39-47
15. Dannenbring, D.G.: An evaluation of flow shop sequencing heuristics. *Management Science* **23** (1977) 1174-1182
16. Nawaz, M., Enscore, Jr. E.E., Ham, I.: A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega* **11**(1983) 91–95

17. Rajendran, C., Ziegler, H.: 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** (2003) 513–522
18. Jungwattanakit, J., Reodecha, M., Chaovalitwongse, P., Werner, F.: (2006) Sequencing heuristics for flexible flow shop scheduling problems with unrelated parallel machines and setup times. *Proceedings of the 2006 IE Network National Conference* Bangkok, Thailand (2006) session F53 pp 1-8
19. Grabowski, J., Wodecki, M.: A very fast tabu search algorithm for the permutation flow shop problem with makespan criterion. *Computers & Operations Research* **31**(2004) 1891–1909

GLOBAL INFORMATION PUBLISHER
www.gip.hk



Price: US$ 150.00