We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

8

Heuristics for Unrelated Parallel Machine Scheduling with Secondary Resource Constraints

Jeng-Fung Chen
Feng Chia University
TAIWAN, R.O.C.

1. Introduction

This research deals with the problem of scheduling N jobs on M unrelated parallel machines. Each job has a due date and requires a single operation. A setup that includes detaching one die and attaching another from the appropriate die type is incurred if the type of the job scheduled is different from the last job on that machine. Due to the mechanical structure of machines and the fitness of dies to each, the processing time for a job depends on the machine on which the job is processed, and some jobs are restricted to be processed on certain machines. Furthermore, the required detaching and attaching times depend on both the die type and the machine. This type of problems may be encountered, for example, in plastics forming industry where unrelated parallel machines are used to process different components and setups for auxiliary equipment (e.g., dies) are necessary. This type of scheduling problems is also frequently encountered in injection molding departments where many different parallel machines are also used to produce different components and for which setups are required for attaching or detaching molds.

In general, the dies (or molds) are quite expensive (tens of thousands dollars each) and thus the number of each type of dies available is limited. Therefore, dies should be considered as secondary resources, the fact of which distinguishes this research from many past studies in unrelated parallel-machine scheduling in which secondary resources are not restricted.

This type of problems is NP-hard (So, 1990). When dealing with a large instance encountered in industry, in the worst case, it may not be able to obtain an optimal solution in a reasonable time. In this research heuristics based on guided search, record-to-record travel, and tabu lists from the tabu search (TS) are presented to minimize the maximum completion time (i.e., makespan or C_{max}) and maximum tardiness (i.e., T_{max}), respectively, to promote schedule performance. Computational characteristics of the proposed heuristics are evaluated through extensive experiments.

The rest of this research is organized in six sections. Previously related studies on parallel machine scheduling are reviewed in Section 2. The record-to-record travel and tabu lists are briefly described in Section 3. The proposed heuristic to minimize makespan and the computational results are reported in Section 4. The proposed heuristic to minimize maximum tardiness and the computational results are reported in Section 5. Conclusions and suggestions for future research are discussed in Section 6.

Source: Multiprocessor Scheduling: Theory and Applications, Book edited by Eugene Levner,
ISBN 978-3-902613-02-8, pp.436, December 2007, Itech Education and Publishing, Vienna, Austria

2. Previously related studies on parallel machine scheduling

Parallel machine scheduling problems have been widely studied in the literature. The machines considered in parallel scheduling problems may be divided into three classes (Allahverdi & Mittenthal, 1994): (1) identical machines, in which the processing time of a specific job is the same on all machines; (2) uniform machines, in which the processing time of a specific job on a given machine is determined by the speed factor of that machine; and (3) unrelated machines, in which the processing time of a specific job among machines may change arbitrarily.

Parker et al. (1977) formulated a parallel machine scheduling problem as a vehicle routing problem and developed algorithms to minimize the total changeover cost. Geoffrion & Graves (1976) developed a quadratic assignment heuristic to minimize the sum of changeover costs. Hu et al. (1987) presented optimum algorithms to minimize the sum of changeover costs. Sumichrast & Baker (1987) also presented an approach to minimize the number of machine changeovers. A branch-and-bound procedure to minimize the number of major setups was developed by Bitran & Gilbert (1990). Tang (1990) presented a heuristic and two lower bounds for the makespan problem. An assignment algorithm to minimize C_{max} was developed by Bitran & Gilbert (1990). Monma & Potts (1993) proposed two heuristics to minimize C_{max} with preemption allowed. Lee & Guignard (1996) developed a hybrid bounding procedure for the C_{max} problem. Weng et al. (2001) proposed several heuristics to minimize the total weighted completion time. Webster & Azizoglu (2001) presented dynamic programming algorithms to minimize total weighted flowtime.

Baker (1973) selected the unscheduled job based on earliest-due-date-first (*EDD*) and assigned it to a machine according to certain rules. Dogramaci & Surkis (1979) presented a list-scheduling heuristic that generates three schedules and selects the one with least total tardiness. Elmaghraby & Park (1974) proposed a branch-and-bound algorithm to minimize some penalty functions of tardiness. An improved algorithm for this case was proposed by Barnes & Brennan (1977). Dogramaci (1984) developed a dynamic programming procedure to minimize total weighted tardiness. Ho & Chang (1991) sorted jobs based on the "traffic congestion ratio" and assigned jobs to machines by applying the list-scheduling procedure of Dogramaci & Surkis (1979). Luh et al. (1990) presented a Lagrangian-relaxation based approach to minimize total weighted tardiness. An "earliest-gamma-date" algorithm to minimize total weighted tardiness was proposed by Arkin & Roundy (1991). Koulamas (1994) sorted jobs based on shortest-processing time-first and generated m copies of this list for the m machines. He then applied certain rules to select the next job to be scheduled to minimize total tardiness. In the later study, Koulamas (1997) presented a decomposition heuristic and a hybrid heuristic to minimize mean tardiness. Suresh & Chaudhuri (1994) presented a GAP-*EDD* algorithm to minimize maximum tardiness. Guinet (1995) employed a simulated annealing method to minimize mean tardiness. Randhawa & Kuo (1994) examined the factors that may have influence on the scheduling performance and proposed heuristics to minimize mean tardiness. Schutten & Leussink (1996) proposed a branch-and-bound algorithm to minimize the maximum lateness. Alidaee & Rosa (1997) developed a "modified-due-date" algorithm to minimize total weighted tardiness. Azizoglu & Kirca (1998) developed a branch-and-bound algorithm to minimize total tardiness. Dessouky (1998) considered that all jobs are identical and have unequal ready times. He proposed a branch-and-bound procedure and six single-pass heuristics to minimize maximum lateness. Balakrishnan et al. (1999) proposed a mixed integer formulation to minimize the sum of

earliness and tardiness. Funda & Ulusoy (1999) developed two genetic algorithms to minimize the sum of weighted earliness and tardiness.

Armentano & Yamashita (2000) presented a TS heuristic to minimize mean tardiness. Yalaoui & Chu (2002) derived some dominance properties and proposed a branch-and-bound procedure to minimize total tardiness. Park et al. (2000) applied neural networks to obtain some look-ahead parameters which were used to calculate the priority index of each job to minimize total weighted tardiness. Lee & Pinedo (1997) presented a three-phase heuristic to minimize total weighted tardiness. In the first phase, factors or statistics which characterize an instance are computed; in the second phase, a sequence is constructed by an ATCS rule; in the third phase, a simulated annealing (SA) method is applied to improve the solution obtained in the second phase. Eom et al. (2002) also proposed a three-phase heuristic to minimize total weighted tardiness with both family and job setup times. In the first phase, jobs are listed by *EDD* and are divided into job sets based on a decision parameter; in the second phase, each job set is organized into several families by using the ATCS algorithm and then a TS method is applied to improve the sequence of jobs in each family; in the third phase, jobs are allocated to machines by using a threshold value and a look-ahead parameter. An *SA* heuristic was presented by Kim et al. (2002) to minimize total tardiness.

Although parallel-machine scheduling has been studied extensively, not much research has considered the case in which a setup for dies is incurred if there is a switch from processing one type of job to another type, the number of dies of each type is limited, the processing time for a job depends on the machine on which the job is processed, and some jobs are restricted to be processed on certain machines. In this research, effective heuristics based on guided search, record-to-record travel, and tabu lists are proposed to deal with this type of scheduling problems so that maximum completion time and maximum tardiness can be minimized, respectively, to promote schedule performance. Computational characteristics of the proposed heuristic are evaluated through extensive experiments.

Underlying assumptions are considered in this research:

1. A setup that includes detaching one die and attaching another from the appropriate die type is incurred if there is a switch from processing one type of job to another type;
2. The detaching (attaching) time depends on both the die type and the machine on which the die is detached (attached);
3. The processing time for a job depends on both the job and the machine on which the job is processed, and each job is restricted to processing on certain machines; and
4. The number of dies of a die type is limited.

3. Record-to-record travel and tabu lists

The concept of record-to-record travel and tabu lists from the tabu search are briefly described in this section. First, the record-to-record travel is described.

3.1 Record-to-record travel

The record-to-record travel (*RRT*) was introduced by Dueck (1993). Basically, *RRT* is very similar to SA. The main difference between *RRT* and SA is the mechanism to determine whether a neighborhood solution (Y) is accepted or not. SA accepts a worse neighborhood solution with a controlled probability. *RRT* accepts a neighborhood solution if its solution value

(V(Y)) is not worse than the current best solution (i.e., *RECORD*) plus a controlled *DEVIATION*. The algorithm of record-to-record travel to minimization may be generalized as follows:

RRT for minimization

generate an initial solution
choose an initial DEVIATION > 0
set RECORD=the current best solution value
Repeat: generate a neighborhood solution that is a perturbation of the current solution (i.e., Y = PERTURB (X))
IF (V(Y)) <RECORD + DEVIATION
THEN accept the move (i.e., X = Y)
IF (V(Y)) <RECORD
THEN set RECORD = (V(Y))
IF no improvement on the solution quality after a number of iterations
THEN lower DEVIATION
IF the stop criterion is reached
THEN stop
GOTO Repeat

3.2 Tabu lists

Since neighborhood solutions not leading to improvement are accepted in *RRT*, it is possible to return to previously visited solutions and cause cycling solutions. Hence, tabu lists from the tabu search (Glover, 1989) are applied to overcome this problem. The tabu lists store attributes that identify certain moves are forbidden in the later search. By using tabu lists, the solutions previously searched may be avoided and new regions of the search space may be explored.

Theoretically the tabu lists need to store all previously visited solutions. However, this would require too much memory and computational efforts. An practical way is to store only the moves occurring in the last s iterations, in which s is known as the tabu size. By using an appropriate tabu size, the likelihood of cycling solutions may be avoided.

An aspiration criterion is used to free a tabu solution if it is of sufficient quality and possibly would not cause cycling solutions. Hence, a solution is not forbidden if its attributes are not tabu or it passes the aspiration criterion test.

4. Heuristic procedure to minimize C_{max} and computational results

The proposed heuristic to minimize C_{max}, *Heu_Cmax*, and computational results are presented in this section. The development of *Heu_Cmax* is based on observing secondary resource constraints and process restrictions, and applying a guide search to improve the solutions. In order to avoid being trapped in local optimum, the record-to-record travel mechanism is applied. In addition, tabu lists are used to prevent obtaining cycling solutions.

Heuristic *Heu_Cmax* consists of a procedure to generate an initial solution, a group *scheduling procedure* to improve makespans of machines, and several procedures to generate neighborhood solutions. Before proceeding to the details of *Heu_Cmax*, the following notations are defined:

group: a set of jobs that are allocated to the same machine and require the same type of die

sub_group: a subset of a *group*

j: index for jobs ($j = 1, 2, \ldots, N$)
m: index for machines ($m = 1, 2, \ldots, M$)
d: index for die types ($d = 1, 2, \ldots, D$)

4.1 Generation of initial solutions

A rule based on process efficiency is applied to assign jobs and allocates dies to machines. The jobs in each machine are then scheduled according to a *group scheduling procedure*. The initial solution is generated as follows:

Step 1. Assign each job $j \in J$ to its most efficient machine. If that machine is not allocated with a required die type, allocate a required die type to that machine.

Step 2. (*group scheduling procedure*) Form *groups* on each machine and schedule the *groups* with the longest detaching time last on each machine.

4.2 Generating neighborhood solutions

In order to minimize makespan, it is necessary to reassign jobs from the machine associated with maximum completion time to another machine. However, there are situations in which reassigning jobs from the latest completion machine to an earlier completion machine is not allowed or makespan cannot be reduced. Hence, it is sometimes necessary to reassign jobs from the latest completion machine to an intermediate machine and simultaneously reassign jobs from this intermediate machine to another machine. Moreover, sometimes it is more appropriate to reassign a *group* or several jobs than just a single job. Therefore, the neighborhood solutions are generated according to the following procedures (Chen, 2005).

4.2.1 Group reassignment

This procedure reassigns one *group* along with its required die from the machine with the latest completion time to another machine. The *group* and machine resulting in the least makespan are selected.

4.2.2 Job reassignment

This procedure reassigns one job from the machine with the latest completion time to another machine. The job and machine resulting in the least makespan are selected.

4.2.3 Sub_group reassignment

This procedure reassigns one *sub_group* from the machine with the latest completion time to another machine. First, each *group* in the machine with the latest completion time is divided into several equal *sub_groups* based on total processing time for the *group*. One *sub_group* is then reassigned to another machine. The *sub_group* and machine resulting in the least makespan are selected. The number of *sub_groups* in one *group* is randomly determined so that a different number of jobs are reassigned in each iteration.

4.2.4 Group and group chain reassignment

In this procedure, one *group* along with its required die from the machine with the latest completion time are reassigned to an intermediate machine and another *group* along with its required die from this intermediate machine are reassigned to another machine. The *group*s and machine resulting in the least makespan are selected.

4.2.5 Group and sub_group chain reassignment

In this procedure, one *group* along with its required die from the machine with the latest completion time are reassigned to an intermediate machine and one *sub_group* from this intermediate machine is reassigned to another machine. The *group*, *sub_group*, and machine resulting in the least makespan are selected.

4.2.6 Sub_group and sub_group chain reassignment

This procedure reassigns one *sub_group* from the machine with the latest completion time to an intermediate machine and simultaneously reassigns one *sub_group* from this intermediate machine to another machine. The *sub_groups* and machine resulting in the least makespan are selected.

4.2.7 Sub_group and job chain reassignment

This procedure reassigns one *sub_group* from the machine with the latest completion time to an intermediate machine and simultaneously reassigns one job from this intermediate machine to another machine. The *sub_group*, job, and machine resulting in the least makespan are selected.

4.2.8 Job and job chain reassignment

In this procedure one job from the machine with the latest completion time is reassigned to an intermediate machine and another job from this intermediate machine is reassigned to another machine. The jobs and machine resulting in the least makespan are selected. It is noted that in the above procedures a *sub_group* or job can be reassigned to a machine only if that machine is allocated with a required die or there is a required die not yet allocated to any machine.

4.2.9 Reattachment

The above reassignment procedures do not apply any reattachments of dies. It is possible that the maximum completion time can be reduced by reattaching dies to other machines. In this reattachment procedure one job from the machine with the latest completion time is reassigned to another machine that may not be allocated with a required die. The die to be reattached is taken from other machines allocated with the required die. This job may be processed very early or very late depending on the availability of the required die. If these arrangements are accepted, they are performed. This procedure is applied repeatedly to reduce the maximum completion time.

When performing the above procedures to generate neighborhood solutions, discard moves that are tabu (unless a tabu move results in an overall best solution). If the makespan obtained is accepted (i.e., $(V(Y)) < RECORD + DEVIATION$), perform the reassignment and update the current solution. If the makespan is improved, update the best solution.

4.3 Heuristic *Heu_Cmax*

Heuristic *Heu_Cmax* is now outlined as follows.

Step 0. Initialization

Initialize *Total_counter* and set *Inner_max*, *Outer_max*, and initial *DEVIATION RATE* (*DR*). Note that *Total_counter* is used to update *DR*.

Step 1. Generate an initial solution and set RECORD = initial solution value and DEVIATION = RECORD × DR.

Step 2. Apply the *group* reassignment procedure until an *Inner_max* number of moves are performed without any improvement to the best known solution.

Step 3. Apply the job reassignment procedure until an *Inner_ max* number of moves are performed without any improvement to the best known solution.

Step 4. Apply the *sub_ group* reassignment procedure until an *Inner__max* number of moves are performed without any improvement to the best known solution.

Step 5. Apply the *group* and *group* chain reassignment procedure until an *Inner__max* number of moves are performed without any improvement to the best known solution.

Step 6. Apply the *group* and *sub_ group* chain reassignment procedure until an *Inner _max* number of moves are performed without any improvement to the best known solution.

Step 7. Apply the *sub_ group* and sub*group* chain reassignment procedure until an *Inner_ max* number of moves are performed without any improvement to the best known solution.

Step 8. Apply the *sub_ group* and job chain reassignment procedure until an *Inner_ max* number of moves are performed without any improvement to the best known solution.

Step 9. Apply the job and job chain reassignment procedure until an *Inner_ max* number of moves are performed without any improvement to the best known solution.

Step 10. If an *Outer__max* number of moves are performed without any improvement to the best known solution, reinitialize *Total_counter*, update *DR*, and go to Step 11. Otherwise, set *Total_ counter* = *Total_counter* + *1*, update *DR*, and return to Step 2.

Step 11. Apply the reattachment procedure.

Step 12. If an *Outer_ max* number of moves are performed without any improvement to the best known solution, terminate *Heu_Cmax*. Otherwise, set *Total_counter* = *Total_counter* + *1*, update *DR*, and return to Step 11.

It is noted that the reattachment procedure may complicate the allocation of dies to machines; hence it is not performed until all the other procedures cannot improve makespan any further.

4.4 Computational results

A set of test problems are used to evaluate the computational characteristics of *Heu_Cmax*. The runtime and solution quality of *Heu_Cmax* are compared with a basic simulated annealing (*BSA*) method (Tamaki et al., 1993). Both *Heu_Cmax* and the *SA* method were coded in C and all of the experiments were performed on a Pentium 4 1.6 GHz PC with 256M SDRAM

4.4.1 Data sets

The test problems were generated "randomly" according to the following factors:

number of jobs (*N*),

number of machines (*M*),

number of die types (*D*), and

number of dies of a die type (b_d).

It is known that all of these factors have impacts on the size and complexity of this scheduling problem. The parameters for this data set are listed in Table 1, in which r_d is the number of jobs requiring die type d.

N	25, 30, 35, 40, 45	50, 55, 60, 65, 70
M	3, 4, 5	6, 7, 8
D	$\lfloor N/6 \rfloor+1, \lfloor N/7 \rfloor+1$	
b_d	$\lfloor r_d/3 \rfloor+1, \lfloor r_d/4 \rfloor+1$	
Speed factor for jobs of type d on machine m, f_{dm}	1/U[5, 15]	
Processing time for job j on machine m, p_{jm}	$1/f_{dm}$ *U[10, 40]	
Attaching time for a die of type d on machine m, $s1_{dm}$	U[10, 30]	
Detaching time for a die of type d on machine m, $s2_{dm}$	U[10, 30]	
Probability that a die type can be attached to a machine	0.5	

Table 1. Parameters of the test data

4.4.1 Parameter settings

For the *BSA* (Tamaki et al., 1993), the solution is represented by a binary string $\widetilde{X}=(\widetilde{x}_{11}\widetilde{x}_{12}\ldots\widetilde{x}_{MN}\,\widetilde{y}_{01}\widetilde{y}_{11}\ldots\widetilde{y}_{NN}\ldots\widetilde{z}_{01}\widetilde{z}_{11}\ldots\widetilde{z}_{NN}\,\widetilde{w}_{01}\widetilde{w}_{11}\ldots\widetilde{w}_{NN})$. The neighborhood of a string $\widetilde{X}$ is the set of strings with *Hamming distance* 1 from $\widetilde{X}$. A procedure is used to transform a binary string to a feasible schedule of

$$x_{mj}=\begin{cases}1 & \text{if job } j \text{ is assigned to machine } m\\ 0 & \text{otherwise}\end{cases},$$

$$y_{jj'}=\begin{cases}1 & \text{if jobs } j \text{ and } j' \text{ are assigned to the same machine and job } j' \text{ immediately succeeds job } j\\ 0 & \text{otherwise}\end{cases},$$

$$z_{jj'}=\begin{cases}1 & \text{if jobs } j \text{ and } j' \text{ require the same type of die and job } j' \text{ immediately succeeds job } j \text{ directly (if jobs } j \text{ and } j' \text{ are processsed on the same machine) or job } j' \text{ immediately succeeds job } j \text{ indirectly (if jobs } j \text{ and } j' \text{ are processed on different machines)}\\ 0 & \text{otherwise}\end{cases},$$

$$w_{jj'}=\begin{cases}1 & \text{if job } j \text{ uses the same die that job } j' \text{ does}\\ 0 & \text{otherwise}\end{cases},$$

The temperature of the k^{th} stage was set at $T(k) = 0.9^k \times 100$, the number of iterations in each stage was set at 1000, and the termination criterion was $T(k) \le 0.01$

For heuristic *Heu_Cmax*, each improvement procedure utilized one tabu list with a size of 5, *Inner_max* was set at 5, *Outer_max* was set at 5, and DR was updated by $0.9^{\lfloor (1+\text{Total\_counter})/5 \rfloor}$.

4.4.2 Results

According to the computational results, the performance of *Heu_Cmax* was very impressive. The *BSA* method did not obtain any better solution value than *Heu_Cmax* in all of the 120 test problems. *Heu_Cmax* was better than the *BSA* method not only on the solution value but also on the runtime. Table 2 shows the mean value comparisons of *Heu_Cmax* and *BSA*. According to Table 2, on an overall average the solution value and runtime were improved 10.98% and 97.77%, respectively.

The improvement of *Heu_Cmax* is more significant when *N* or *M* is large. The magnitude of N and M affects the size and complexity of this scheduling problem. Hence, this computational experience may indicate that *Heu_Cmax* may perform much better than the *BSA* method when this type of scheduling problems involves more jobs or more parallel machines. The improvement of *Heu_Cmax* is also more significant when b_d is small. The number of dies of each type affects the availability of the secondary resource. Hence, this computational experience may indicate that *Heu_Cmax* may perform much better than the *BSA* method when secondary resources are tightly constrained.

5. Heuristic procedure to minimize T_{max} and computational results

The heuristic proposed in section 4 can be modified to minimize T_{max}. The modified heuristic is named *Heu_Tmax* and is described in the followings.

						(*BSA*-*Heu_Cmax*)/*BSA**100%	
		C_{max}	CPU sec.	C_{max}	CPU sec.	C_{max}	CPU sec.
N	25	1355.52	32.70	1235.66	1.17	8.84%	96.42%
	30	1635.08	49.95	1520.44	1.46	7.01%	97.09%
	35	2048.58	113.80	1897.71	1.69	7.36%	98.52%
	40	2401.92	214.60	2142.68	2.30	10.79%	98.93%
	45	2630.00	246.65	2411.11	2.86	8.32%	98.84%
	50	1744.20	289.80	1538.96	5.06	11.77%	98.25%
	55	1900.24	299.70	1681.33	7.88	11.52%	97.37%
	60	2075.28	365.50	1797.81	8.06	13.37%	97.79%
	65	2250.09	395.20	1941.77	12.62	13.70%	96.81%
	70	2415.63	442.20	2019.26	11.58	16.41%	97.38%
M	3	2772.22	133.65	2480.65	1.61	10.52%	98.80%
	4	1801.42	119.85	1722.22	1.64	4.40%	98.64%
	5	1469.02	141.15	1327.37	2.45	9.64%	98.27%
	6	2590.42	343.85	2238.58	9.34	13.58%	97.28%
	7	2024.74	364.60	1753.44	9.65	13.40%	97.35%
	8	1616.10	366.95	1398.96	8.13	13.44%	97.78%
D	$\lfloor N/6 \rfloor+1$	2048.40	244.25	1832.60	5.77	10.54%	97.64%
	$\lfloor N/7 \rfloor+1$	2042.91	245.75	1823.12	5.17	10.76%	97.90%
b_d	$\lfloor r_d/3 \rfloor+1$	2008.85	245.30	1825.82	6.07	9.11%	97.53%
	$\lfloor r_d/4 \rfloor+1$	2082.46	244.70	1829.90	4.87	12.13%	98.01%
Overall average		2045.65	245.00	1820.97	5.47	10.98%	97.77%

Table 2. Mean value comparisons of *Heu_Cmax* and *BSA*

5.1 Generation of initial solutions

For the initial solution, each job is first assigned to its most efficient machine. If that machine is not allocated with a required die, allocate a required die to that machine. The job sequence in each machine is then improved by a *rescheduling procedure*. The *rescheduling procedure* is to improve job sequence within a machine. It is applied whenever *group*, *sub_group*, and jobs are reassigned from one machine to another. The *rescheduling procedure* includes the following steps:

Step 1. Form *groups* in each machine and sequence jobs in the same *group* according to *EDD*.

Step 2. Schedule the *group* last of the entire *group*s unscheduled if it would incur the least maximum tardiness. Repeat this process until all *groups* are scheduled.

Step 3. Starting from the first job of the second *group* in the sequence, move the job along with all its predecessors in the same family forward to the best position to improve maximum tardiness.

5.2 Generating neighborhood solutions

The neighbourhood generation procedures are similar to those described in subsection 4.2 except that the *group*, *subgroup*, or job reassigned is selected from the machine associated with maximum tardiness and that the *group, subgroup*, or job and machine resulting in the least maximum tardiness are selected (Chen, 2006).

5.2.1 Group reassignment

This procedure reassigns one *group* along with its required die from the machine associated with the maximum tardiness to another machine. The *group* and machine resulting in the least maximum tardiness are selected.

5.2.2 Job reassignment

This procedure reassigns one job from the machine associated with the maximum tardiness to another machine. The job and machine resulting in the least maximum tardiness are selected.

5.2.3 Sub_group reassignment

This procedure reassigns one *sub_group* from the machine associated with the maximum tardiness to another machine. First, each *group* in the machine associated with maximum tardiness is divided into several equal *sub_group* based on the total processing time for the *group* and the due date of every job in the first *sub_group's* being earlier than that of any job in the second *sub_group*, and the due date of every job in the second *sub_group's* being earlier than that of any job in the third *sub_group* and so on. One *sub_group* is then reassigned to another machine. The *sub_group* and machine resulting in the least maximum tardiness are selected.

5.2.4 Group and group chain reassignment

In this procedure, one *group* along with its required die from the machine associated with the maximum tardiness are reassigned to an intermediate machine and another *group* along with its required die from this intermediate machine are simultaneously reassigned to

another machine. The *groups* and machine resulting in the least maximum tardiness are selected.

5.2.5 Group and sub_group chain reassignment

In this procedure, one *group* along with its required die from the machine associated with the maximum tardiness are reassigned to an intermediate machine and one *sub_group* from this intermediate machine is simultaneously reassigned to another machine. The group, *sub_group*, and machine resulting in the least maximum tardiness are selected.

5.2.6 Sub_group (job) and sub_group (job) chain reassignment

This procedure reassigns one *sub_group* (job) from the machine associated with the maximum tardiness to an intermediate machine and simultaneously reassigns one *sub_group* (job) from this intermediate machine to another machine. The *sub_group*(*s*), job(s), and machine resulting in the least maximum tardiness are selected.

5.2.7 Reattachment

In this procedure, one job from the machine associated with the maximum tardiness is reassigned to another machine that may not be allocated with a die. This job may be processed very early or very late depending on the availability of the required die. The job and machine resulting in the least maximum tardiness are selected.

5.3 Heuristic *Heu_Tmax*

The structure of heuristic *Heu_Tmax* is very similar to that of heuristic *Heu_Cmax*. Readers may refer to subsection 4.3.

5.4 Computational experiments

A set of test problems is used to evaluate the computational characteristics of *Heu_Tmax*. The runtime and solution quality of *Heu_Tmax* are compared with an *EDD*-based procedure and a basic *SA* method (*BSA*) (Tamaki et al., 1993).

5.4.1 Data Sets

The test problems were "randomly" generated based on the following factors:

5. number of jobs (N);
6. number of machines (M);
7. number of die types (D);
8. number of dies of a die type (b_d);
9. due date range factor (R); and
10. due date priority factor (τ).

The level settings for each factor are: 4 levels for N, 3 levels for M, and 2 levels each for the other factors. This results in a total of 192 test problems. The parameters for the test problems are given in Table 3. Note that in Table 3 the due dates of jobs were generated as suggested by Potts & Van Wassenhove (1982), where

$C_{max} = (\sum_j (\sum_{m \in M'_j} p_{jm} + s1_{jm} + s2_{jm}) / |M'_j|) / M$ and M'_j is set of machines that can process job j.

N	30, 50, 70, 90
M	4, 6, 8
D	$\lfloor N/5 \rfloor + 1, \lfloor N/7 \rfloor + 1$
b_d	$\lfloor r_d/4 \rfloor + 1, \lfloor r_d/6 \rfloor + 1$
Speed factor for jobs of type d on machine m, f_{dm}	$1/U[5, 15]$
Processing time for job j on machine m, p_{jm}	$1/f_{dm}$*U[10, 40]
Attaching time for a die of type d on machine m, $s1_{dm}$	U[10, 100]
Detaching time for a die of type d on machine m, $s2_{dm}$	U[10, 100]
R	0.4, 1
τ	0.4, 0.8
Due date	$U[C_{max}(1 - \tau - R/2), C_{max}(1 - \tau + R/2)]$
Probability that a die type can be attached to a machine	0.5

Table 3. Parameters for the test data

The *EDD-based* procedure selects jobs on the basis of *EDD* and assigns jobs to the machine where it can be completed as early as possible. However, if a required die is not available, the next job is selected. For heuristic *Heu_Tmax*, each neighborhood generation procedure use a tabu list of size 5, *Inner_max* and *Outer_max* were both set at 5, and DR was updated by $0.9^{\lfloor (1+\text{Total\_counter})/5 \rfloor}$.

5.4.2 Results

According to the computational results, *Heu_Tmax* outperformed *EDD* and *BSA* in terms of solution quality. *EDD* and *BSA* did not obtain better solutions than *Heu_Tmax* in all of the 192 tested instances. *EDD* and *Heu_Tmax* obtained the same solutions in 12 tested problems; *BSA* and *Heu_Tmax* obtained the same solutions in 24 tested problems. As for the runtime consumed, the *EDD*-based procedure required less than 1 second to solve each of the tested instances. Depending upon the problem sizes, the runtime of *Heu_Tmax* ranged from less than 1 second to near 6 minutes, which was much less than that of *BSA*.

Table 4 shows the corresponding mean values of *EDD*, *BSA*, and *Heu_Tmax*. According to Table 4, maximum tardiness increases as the number of jobs (i.e., *N*) increases or the number of machines (i.e., *M*) decreases. Maximum tardiness also increases when secondary resources are more restricted (i.e., $b_d = \lfloor r_d/6 \rfloor + 1$) or the due dates of job are tight (i.e., τ = 0.8). On an overall average, the solution value of *EDD* was improved 42.88%; the solution value and the runtime of *BSA* were reduced 27.92% and 90.48%, respectively. *Heu_Tmax* is significantly better than *EDD* and *SA* when *M* is large. The sizes of *M* affect the size and complexity of this scheduling problem. Hence, this computational experiment may indicate that the performance of Heu_Tmax may be much better than *EDD* and *BSA* when this type of scheduling problems involves more parallel machines.

Heu_Tmax is significantly better than *EDD* and *SA* when *R* is small (i.e., *R* = 0.4). The value of R affects the dispersion of job due dates. Hence, this computational experience may indicate that the performance of *Heu_Tmax* may be much better than *EDD* and *BSA* when the due dates of jobs are more dispersive. *Heu_Tmax* is also significantly better than *EDD* and *BSA* when τ is small (i.e., τ = 0.4). The value of τ influences the tightness of due dates. Hence, this computational experiment may indicate that the performance of *Heu_Tmax* may be much better than *EDD* and *BSA* when the due dates of jobs are loose.

		EDD	*BSA*		*Heu_Tmax*	
		T_{max}	T_{max}	CPU sec.	T_{max}	CPU sec.
N	30	672.19	508.13	33.73	410.81	1.35
	50	1012.77	853.58	120.93	645.79	4.17
	70	1382.63	1130.48	268.30	740.65	19.67
	90	1810.85	1373.35	468.83	989.21	59.69
M	4	1536.30	1241.33	219.69	909.64	25.71
	6	1162.59	946.70	220.41	702.67	24.19
	8	959.94	711.13	228.74	477.53	13.75
D	$\lfloor N/5 \rfloor+1$	1276.10	1017.23	224.64	712.64	26.18
	$\lfloor N/7 \rfloor+1$	1163.12	915.54	221.25	680.59	16.26
b_d	$\lfloor r_d/4 \rfloor+1$	1202.87	944.29	246.28	678.71	27.21
	$\lfloor r_d/6 \rfloor+1$	1236.35	988.48	199.61	714.52	15.23
R	0.4	1500.54	1090.60	221.85	661.97	18.32
	1	938.68	842.17	224.04	731.26	24.12
τ	0.4	459.09	212.14	223.02	51.04	6.13
	0.8	1980.13	1720.64	222.88	1342.19	36.31
Overall average		1219.61	966.39	222.95	696.62	21.22

Table 4. Mean value comparisons of *Heu_Tmax*, *EDD*, and *BSA*

6. Conclusions and suggestions for future research

This research has dealt with scheduling jobs on unrelated parallel machines with secondary resource constraints. Effective heuristics based on guided search, record-to-record travel, and tabu lists from tabu search have been proposed to minimize makespan and maximum tardiness, respectively. The solution quality of the proposed heuristics have been evaluated in empirical comparisons with an *BSA* method and *EDD*. Computational results have demonstrated that the presented heuristics outperform these method and procedures tested. It is expected that this research may provide an innovative approach for production managers to schedule jobs in the production environment where unrelated parallel machines are used to process different components and for which setups are required for auxiliary equipments. Since the development of the proposed heuristics observe secondary resource constraints, family setup times, process restrictions, hence it is believed that the proposed heuristics may also be effectively applied to solve the parallel-machine scheduling problems with family and job setup times.

As for future research, it may be desirable to develop and study effective heuristics for the dynamic case where jobs arrive over time. Considering that the jobs (orders) from important customers have strict due-date constraints is another important issue for future research to pursue.

7. Acknowledgements

This research is partially supported by the National Science Council on Grant number NSC 95-2213-E-035-082. Some of the material in this research is based on the work published in the International Journal of Advanced Manufacturing Technology.

8. References

Alidaee, B. & Rosa, D. (1997). Scheduling parallel machines to minimize total weighted and unweighted tardiness. *Computers & Operations Research*, 24, 775-788

Allahverdi, A. & Mittenthal, J. (1994). Scheduling on M parallel machines subject to random breakdowns to minimize expected mean flow time. *Naval Research Logistics*, 41, 677-682

Arkin, M.E. & Roundy, R.O. (1991). Weighted-tardiness scheduling on parallel machines with proportional weights. *Operations Research,* 39, 64-76

Armentano, V.A. & Yamashita, D.S. (2000). Tabu search for scheduling on identical parallel machines to minimize mean tardiness. *Journal of Intelligent Manufacturing,* 11, 453-460

Azizoglu, M. & Kirca, O. (1998). Tardiness minimization on parallel machines. *International Journal of Production Economics*, 55, 163-168

Baker, K.R. (1973). Procedures for sequencing tasks with one resource type. *International Journal of Production Research*, 11., 125-138,

Balakrishnan, N.; Kanet, J. & Sridharan, S.V. (1999). Earliness/tardiness with sequence dependent setups on uniform parallel machines. *Computers & Operations Research*, 26, 127-141

Barnes, WJ. & Brennan, JJ. (1977). An improved algorithm for scheduling jobs on identical machines. *AIIE Transactions*, 9, 25-31,

Brian, G.R. & Gilbert, S.M. (1990). Sequencing production on parallel machines with two magnitudes of sequence-dependent setup cost. *Journal of Manufacturing and Operations Management*, 3, 24-52

Chen J.-F. (2005). Unrelated parallel machine scheduling with secondary resource constraints. *International Journal of Advanced Manufacturing Technology*, 26, 285-292

Chen J.-F. (2005). Minimization of maximum tardiness on unrelated parallel machines with process restrictions and setups. *International Journal of Advanced Manufacturing Technology*, 29, 557-563

Dessouky, M.I. (1998). Scheduling identical jobs with unequal ready times on uniform parallel machines to minimize the maximum lateness. *Computers and Industrial Engineering*, 34, 793-806

Dogramaci, A. & Surkis, J. (1979). Evaluation of a heuristic for scheduling independent jobs on parallel identical processors. *Management Science*, 25, 1208-1216

Dogramaci, A. (1984). Production scheduling of independent jobs on parallel identical machines. *International Journal of Production Research*, 16, 535-548

Dueck, G. (1993). New optimization heuristics: the great deluge algorithm and the record-to-record travel. *Journal of Computational Physics*, 104, 86-92

Elmaghraby, S.E. & Park, S.H. (1974). Scheduling jobs on a number of identical machines. *AIIE Transactions*, 6., 1-13

Eom, D.-H.; Shin, H.-J.; Kwun, I.-H.; Shim, J.-K. & Kim, S.-S. (2002). Scheduling jobs on parallel machines with sequence-dependent family set-up times. *International Journal of Advanced Manufacturing Technology*, 19, 926-932

Funda, S.-S. & Ulusoy, G. (1999). Parallel machine scheduling with earliness and tardiness penalties. *Computers & Operations Research*, 26, 773-787

Geoffrion, A.M. & Graves, G.W. (1976). Scheduling parallel production lines with changeover costs: practical application of a quadratic assignment/LP approach. *Operations Research*, 24, 595-610

Glover, F. (1989). Tabu search-part I. *ORSA Journal on Computing*, 1, 190-206

Guinet, A. (1993). Scheduling sequence-dependent jobs on identical parallel machines to minimize completion time criteria. *International Journal of Production Research*, 31, 1579-1594

Guinet, A. (1995). Scheduling independent jobs on uniform parallel machines to minimize tardiness criteria. *Journal of Intelligent Manufacturing*, 6, 95-103

Ho, J.C. & Chang, Y.L. (1991). Heuristics for minimizing mean tardiness for m parallel machines. *Naval Research Logistics*, 38, 367-381

Hu, T.C.; Kuo, Y.S. & Ruskey, F. (1987). Some optimum algorithms for scheduling problems with changeover costs. *Operations Research*, 35, 94-99

Kim, D.-W.; Kim, K.-H.; Jang, W. & Chen, F.F. (2002). Unrelated parallel machine scheduling with setup times using simulated annealing. *Robotics and Computer-Integrated Manufacturing*, 18., 223-231,

Koulamas, C. (1994). The total tardiness problem: review and extensions. *Operations Research*, 42, 764-775

Koulamas, C. (1997). Decomposition and hybrid simulated annealing heuristics for the parallel-machine total tardiness problem. *Naval Research Logistics*, 44, 109-125

Lee, H. & Guignard, M. (1996). A hybrid bounding procedure for the workload allocation problem on parallel unrelated machines with setups. *Journal of the Operational Research Society*, 47, 1247-1261

Lee, H. & Pinedo, M. (1997). Scheduling jobs on parallel machines with sequence-dependent setup times. *European Journal of Operational Research*, 100, 464-474

Luh, P.B.; Hoitomt, D.J.; Max, E. & Pattipati, K.R. (1990). Schedule generation and reconfiguration for parallel machines. *IEEE Transactions on Robotics and Automation* 6, 687-696

Monma, C.L. & Potts, C.N. (1993). Analysis of heuristics for preemptive parallel machine scheduling with batch setup times. *Operations Research*, 41, 981-993

Park, Y.; Kim, S. & Lee, Y.-H. (2000). Scheduling jobs on parallel machines applying neural network and heuristic rules. *Computers & Industrial Engineering*, 38, 189-202

Parker, R.G.; Deane, R.H. & Holmes, R.A. (1977). On the use of a vehicle routing algorithm for the parallel processor problem with sequence dependent change over costs. *AIIE Transactions.*, 9, 155-160

Potts, C.N. & Van Wassenhove, L. (1982). Decomposition algorithm for the single machine total tardiness problem. *Operations Research Letters*, 5, 177-181

Randhawa, S.U. & Kuo, C.H. (1997). Evaluating scheduling heuristics for non-identical parallel processors. *International Journal of Production Research*, 35, 969-981

Schutten, J.M.J. & Leussink, R.A.M. (1996). Parallel machine scheduling with release dates, due dates and family setup times. *International Journal of Production Economics*, 46-47, 119-125

So, K.C. (1990). Some heuristics for scheduling jobs on parallel machines with setups. *Management Science*, 36, 467-489

Sumichrast, R.T. & Baker, J.R. (1987). Scheduling parallel processors: an integer linear programming based on heuristics for minimizing setup time. *International Journal of Production Research*, 25, 761-771

Suresh, V. & Chaudhuri, D. (1994). Minimizing maximum tardiness for unrelated parallel machines. *International Journal of Production Economics*, 223-229

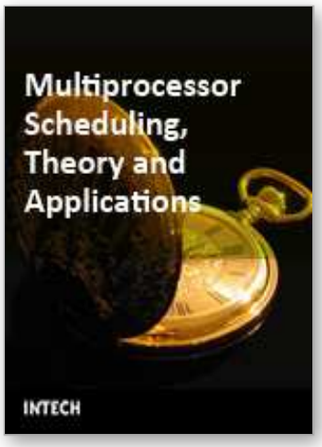
Tamaki, H.; Hasegawa, Y.; Kozasa, J. & Araki, M. (1993). Application of search methods to scheduling problem in plastics forming plant: a binary representation approach. *Proceedings of the 32nd IEEE Conference on Decision and Control,* pp. 3845-3850, San Antonio, TX, December, 1993

Tang, C.S. (1990). Scheduling batches on parallel machines with major and minor set-ups. *European Journal of Operations Research*, 46, 28-37

Webster, S. & Azizoglu, M. (2001). Dynamic programming algorithms for scheduling parallel machines with family setup times. *Computers & Operations Research*, 28, 127-137

Weng, M.; Lu, X.J. & Ren, H. (2001). Unrelated parallel machine scheduling with setup consideration and a total weighted completion time objective. *International Journal of Production Economics*, 70, 215-226

Yalaoui, F. & Chu, C. (2002). Parallel machine scheduling to minimize total tardiness. *International Journal of Production economics*, 76, 265-279



Multiprocessor Scheduling, Theory and Applications

Edited by Eugene Levner

ISBN 978-3-902613-02-8

Hard cover, 436 pages

Publisher I-Tech Education and Publishing

Published online 01, December, 2007

Published in print edition December, 2007

A major goal of the book is to continue a good tradition - to bring together reputable researchers from different countries in order to provide a comprehensive coverage of advanced and modern topics in scheduling not yet reflected by other books. The virtual consortium of the authors has been created by using electronic exchanges; it comprises 50 authors from 18 different countries who have submitted 23 contributions to this collective product. In this sense, the volume can be added to a bookshelf with similar collective publications in scheduling, started by Coffman (1976) and successfully continued by Chretienne et al. (1995), Gutin and Punnen (2002), and Leung (2004). This volume contains four major parts that cover the following directions: the state of the art in theory and algorithms for classical and non-standard scheduling problems; new exact optimization algorithms, approximation algorithms with performance guarantees, heuristics and metaheuristics; novel models and approaches to scheduling; and, last but least, several real-life applications and case studies.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen