Solving a Flowshop Scheduling Problem with Additional Resource Constraints

Ewa Figielska

Mila College School of Higher Education in Warsaw, Warsaw, Poland,
e-mail: efigielska@poczta.mila.edu.pl

Abstract. The paper considers the problem of preemptive scheduling in a two-stage flowshop with parallel unrelated machines at the first stage and a single machine at the second stage. At the first stage jobs require some amounts of additional renewable resources. The objective is the minimization of the makespan. The problem is NP-hard. A heuristic which combines column generation technique with a genetic algorithm is proposed. Several problems with randomly generated parameters are solved. The results indicate that the heuristic is a promising tool for solving the considered problem.

1 Introduction

In this paper we propose a new heuristic algorithm for a two-stage flowshop scheduling problem with parallel unrelated machines and additional resource constraints at the first stage and a single machine at the second stage. A number of jobs have to be processed at two stages, each job being processed first at stage 1 then at stage 2. At the first stage, a job can be processed on any machine and during its processing it requires some amounts of additional renewable resources. The total amounts of these resources available at a moment are limited. It is assumed that preemptions of jobs are allowed, and that each machine can work on at most one job at a time and a job can be processed on at most one machine at a time. A job can be taken to the second stage as soon as it completes its processing at the first stage. Unlimited intermediate storage space is available to hold jobs completed at the first stage. The aim is to find a feasible schedule which minimizes the makespan, denoted by $C_{\max}$, which is defined as the maximum completion time of jobs at the second stage.

To the best of our knowledge this problem has not been considered in the literature so far. However, such a problem may arise in a manufacturing environment in which products are initially processed on any of parallel machines (production lines) and then each product must go through a final testing operation, which is to be carried out on a common testing machine. In real-life systems that are encountered in process industries, such as chemical, food, cosmetics, preemptions of jobs are permitted and usually result in shorter schedules than in the case when preemptions are not allowed. Moreover, such systems are often subjected to some additional constraints e.g. on the availability of the additional resources such as manpower, fuel flow, tools.

During the last decade the flowshops with multiple machines, also called hybrid flowshops, received considerable attention from researchers. Most literature in this area addresses the minimum makespan problems under the assumption that preemptions of jobs are not allowed and parallel machines at each stage are identical (e.g. [9, 3, 10, 2, 13, 14]). Only a few papers concern the flowshop with parallel machines that are not identical [17, 15]. In these papers no resource

constraints were considered. A two-stage multiprocessor flowshop scheduling problem with preemptions is considered in [12], where it is shown to be NP-hard in the strong sense even in the case of two identical parallel machines at one stage and one machine at another.

2 The Framework of the Heuristic

The proposed heuristic can be outlined as follows.

1. The problem of unrelated parallel machine scheduling with additional resource constraints which occurs at the first stage is optimally solved by a column generation (CG) algorithm [5]. The objective is to minimize the makespan (the maximum job completion time at the first stage).
   - The solution to the first-stage problem is represented by a set of partial schedules. A partial schedule assigns some jobs (or part of jobs) to machines for parallel processing during a certain period of time, so that resource constraints are fulfilled at every moment. The makespan does not depend on the ordering (sequence) of partial schedules.
   - Completion times of jobs at stage 1 depend on the ordering of partial schedules.
   - Jobs that are finished at the first stage are ready to be processed at the second stage. They can be stored between the stages. A job starts on the second stage machine as soon as this machine is free.
   - The ready time of a job at stage 2 is equal to its completion time at stage 1.
   - The makespan in the two-stage flowshop, which is equal to the maximum job completion time at stage 2, depends on the processing times and ready times of jobs at stage 2, so, it depends on the ordering of partial schedules at stage 1.
   - A GA is used for minimizing the makespan in the two-stage flowshop.
2. A GA operates on a population of individuals (choromosomes) which define the ordering of the partial schedules. The makespan is used to evaluate a chromosome. For each generated sequence of the partial schedules, a schedule is constructed and the makespan is calculated taking into account the ready times and processing times of jobs at stage 2.

To illustrate the performance of the heuristic we present the following example. Consider an instance of 10 jobs with the job processing times and resource requirements as shown in Figure 2. There are two stages: the first stage contains two parallel unrelated machines, the second stage consists of one machines. The schedules are presented in Figure 1. In Figure 1a the ordering of the partial schedules is $S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10}$ (where S_k denotes the partial schedule of index k). We can see that at the beginning of the schedule jobs 1 and 9 are processed simultaneously on machines of the first stage during some period of time. These two jobs belong to the partial schedule of index 1 (S_1). Then go: partial schedule S_2 with jobs 7 and 2, S_3 with jobs 4 and 9, and so on. Job 1 is completely finished in S_1 (in fact it is the first job completed at stage 1), while job 9 is not completed until S_3. So, job 1, after its completion at stage 1, passes to stage 2 and, as this stage is free, the processing of job1 begins. The next job completed at stage 1 (in S_2) is the job of index 2. After its completion job 2 is stored between

the stages until the machine of stage 2 is released. The indices of the successive jobs completed at stage 1 are: 1, 2, 9, 7, 10, 3, 4, 5, 8, 6.

In Figure 1b the ordering of the partial schedules is (S_1 , S_7 , S_2 , S_4 , S_5 , S_8 , S_6 , S_6 , S_{10} , S_9 , S_3). This ordering has been found by the GA so as to minimize the maximum job completion time at stage 2 (the makespan of the whole schedule). We can see that the makespan in Figure 1b, which is equal to 423.17, is significantly smaller than the makespan in Figure 1a, equal to 470.

As for the resource constraints, each partial schedule satisfies resource constraints at a time. For the considered instance, the availability of an additional resource at any moment is 10 units. In S_1 , job 1 executed on machine 1 requires 6 units (see Figure 2) of an additional resource at a time, and job 9 executed on machine 2 requires 3 units of this resource at a time, so the total usage of the resource at any moment in this partial schedule is equal to 9 and is less than 10. Similarly, all remaining partial schedules satisfy resource constraints at any moment.

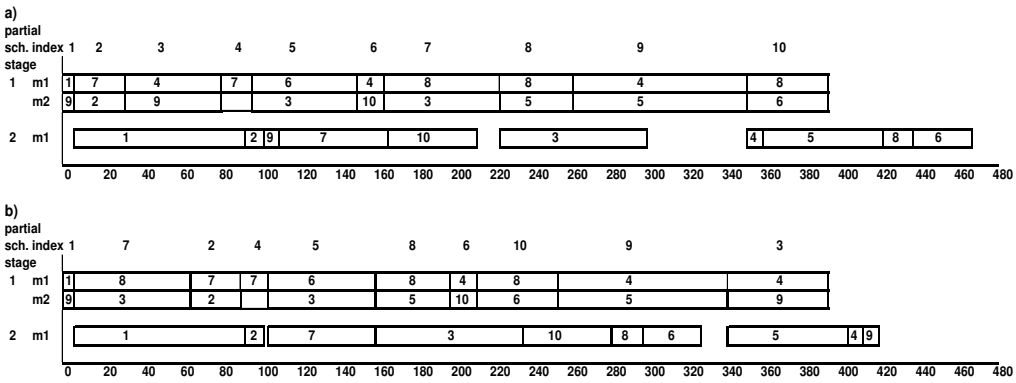

Figure 1. An illustrative example. The resulting schedules: a) the feasible schedule with a random sequence of the partial schedules, b) the final schedule with the sequence of the partial schedules minimizing the makespan.

processing times at stage 1

job:	machine 1	machine 2
1	6	170
2	138	25
3	174	115
4	154	154
5	178	127
6	100	92
7	40	184
8	141	147
9	176	57
10	172	14

processing times at stage 2

job:	machine 1
1	88
2	9
3	75
4	9
5	61
6	30
7	56
8	17
9	9
10	46

resource requirements at stage 1

job:	machine 1	machine 2
1	6	4
2	6	1
3	10	3
4	4	2
5	5	6
6	5	5
7	9	5
8	2	6
9	10	3
10	8	6

resource availability = 10

Fig. 2 The data for an illustrative example

3 Notation

In this paper jobs are indexed by j, parallel machines at stage 1 by i, resource types by r. The parameters of the problem considered are as follows:

n the number of jobs,
m the number of machines at stage 1,
l the number of types of renewable resources,
p_{ij} the processing time of job j on machine i at stage 1,
s_j the processing time of job j on the machine at stage 2,
W_r the number of units of resource r available at a time,
α_{ijr} the number of units of resource r required at every moment during processing job j on machine i at stage 1.

4 Heuristic Description

4.1 The First Stage Problem Solving

As stated in Section 2, first, the problem of resource constrained preemptive scheduling of parallel unrelated machines so as to minimize the makespan is solved. The solution to this problem is represented by a set S of partial schedules S_β, $\beta \in B$, where B is the set of indices of all feasible partial schedules. Partial schedule S_β is determined by its duration Δ_β and the values of v_{ij}^β ($j = 1,...,n$, $i = 1,...,m$) representing an assignment of jobs to machines, where $v_{ij}^\beta = 1$ if job j is processed on machine i in partial schedule S_β and $v_{ij}^\beta = 0$, otherwise. The problem is formally defined as follows:

$$\min \sum_{\beta \in B} \Delta_\beta \tag{1}$$

subject to:

$$\sum_{\beta \in B} \Delta_\beta \sum_{i=1}^{m} \frac{v_{ij}^\beta}{p_{ij}} = 1, \; j = 1,\ldots,n \tag{2}$$

$$\sum_{j=1}^{n} v_{ij}^\beta \le 1, \;\; i = 1,...,m, \;\; \beta \in B \tag{3}$$

$$\sum_{i=1}^{m} v_{ij}^\beta \le 1, \;\; j = 1,...,n, \beta \in B \tag{4}$$

$$\sum_{j=1}^{n} \sum_{i=1}^{m} \alpha_{ijr} v_{ij}^\beta \le W_r, \; r = 1,...,l, \beta \in B \tag{5}$$

$$\Delta_\beta \ge 0, \;\; \beta \in B \tag{6}$$

$$v_{ij}^\beta \in \{0,1\}, \;\; j = 1,...,n, i = 1,...,m, \beta \in B \tag{7}$$

where Δ_β ($\beta \in B$) and v_{ij}^β ($j = 1,...,n$, $i = 1,...,m$, $\beta \in B$) are decision variables. Constraints (2) ensure that all jobs are completed at the first stage of the two-stage flowshop.

Constraints (3) and (4) ensure that, respectively, each machine works on at most one job at a time and each job is processed on no more than one machine at a time. Due to constraints (5) the usage of each resource at every moment does not exceed its availability.

In the general case, the above problem is known to be NP-complete [16]. It can be optimally solved by means of a CG algorithm. The theoretical basis of the CG technique has been provided by Dantzig and Wolfe in [5] (for applications of the CG technique see e.g. [7, 1, 6, 4].

A CG algorithm does not generate explicitly all columns of the problem, which correspond to all partial schedules. It works only with a subset of columns and adds a new column which improves the solution. At each iteration of the CG algorithm, the schedule length is minimized by solving the LP problem of the form:

$$\min \sum_{\beta \in \tilde{B}} \Delta_\beta \tag{8}$$

subject to

$$\sum_{\beta \in \tilde{B}} \Delta_\beta \sum_{i=1}^{m} \frac{v_{ij}^\beta}{p_{ij}} = 1 \quad j = 1,...,n \tag{9}$$

$$\Delta_\beta \geq 0, \ \beta \in \tilde{B} \tag{10}$$

where $\tilde{B} \subset B$ denotes a subset of indices of columns, Δ_β ($\beta \in \tilde{B}$) are decision variables, and the values of v_{ij}^β ($\beta \in \tilde{B}$, $j = 1,...,n$, $i = 1,...,m$) are fixed (determined in the previous iterations) or (before the first iteration) given in advance. Let π_j^* ($j = 1,...,n$) be the optimal solution of the dual problem to the problem (8)-(10) (π_j are dual variables corresponding to constraints (9)). If there exists a column $\beta \in B \setminus \tilde{B}$ such that $\sum_{j=1}^{n} \sum_{i=1}^{m} \pi_j^* v_{ij}^\beta / p_{ij} - 1 > 0$, then the current set $\tilde{B}$ can be extended by this new index β and a new iteration of the CG algorithm is started. Otherwise the optimal solution is found and the algorithm stops.

4.2 Minimal Makespan Schedule Finding

The aim is the minimization of the makespan in the two-stage flowshop. A GA finds the ordering of the partial schedules which provides the schedule for the two-stage flowshop with minimum makespan.

A GA [11] is a search technique that imitates the natural selection and biological evolutionary process. GAs have been used in a wide variety of applications, particularly in combinatorial optimization problems and they were proved to be able to provide near optimal solutions in reasonable time.

A GA starts with a *population* of randomly generated candidate solutions (called *chromosomes*). A chromosome is represented by a string of numbers called *genes*. Each chromosome in the population is evaluated according to some fitness measure. Certain pairs of chromosomes (*parents*) are selected on the basis of their fitness. Each of these pairs combines to produce new chromosomes (*offspring*) and some of the offspring are randomly modified. A new

population is then formed replacing some of the original population by an identical number of offspring. The process is repeated until a stopping criterion is met.

Let $P(t)$ denotes the population at iteration t and pop_size is the population size. The GA applied in this paper can be outlined as follows.

1. Generate and evaluate the initial population $P(t)$, t=0.
2. Repeat the following steps until stopping condition is satisfied.
   2.1. Repeat the following loop $pop\_size/2$ times (pop_size is an even number).
      2.1.1. Select two parents from $P(t)$.
      2.1.2. Apply the crossover operator over the parent chromosomes and produce 2 offspring chromosomes.
      2.1.3. Apply the mutation operator over the offspring.
      2.1.4. Copy the offspring to population $P(t+1)$.
   2.2. Evaluate $P(t+1)$.
   2.3. Replace the worst chromosome of $P(t+1)$ by the best chromosome found so far.
   2.4. Set $t = t+1$.
3. Return the best chromosome found.

The factors which characterize the GA applied to the problem considered in this paper are determined as follows.

Solution representation. A solution to the sequencing problem solved by the GA is coded as a single chromosome whose genes represent the indices of partial schedules.

Initial population. An initial population of chromosomes is randomly generated.

Evaluation. The value of an objective function, which is equal to the makespan in the two-stage flowshop, is used to measure the fitness of a chromosome. For each partial schedule sequence (chromosome) generated in the search process, a schedule for the two-stage flowshop is constructed and the makespan is calculated taking into account ready times and processing times of jobs at the second stage.

Parent selection. The binary tournament selection method is used. In a binary tournament selection, two chromosomes are randomly chosen. The more fit (with a smaller objective function value) is then taken as a parent chromosome. Two binary tournaments are held to produce two parents.

Crossover. The two-point crossover operator PMX [9] is applied to each pair of parent chromosomes with a probability P_{crs} (*crossover probability*).

Mutation. The genes of each chromosome in the population are considered one by one, and the gene being considered swaps its value with another randomly generated gene of the same chromosome with a probability P_{mut} (*mutation probability*).

Stopping condition. The search process terminates when the best objective function value (makespan) found so far is not updated for a predetermined number of iterations.

On the basis of the preliminary computational experiment the following values of the genetic parameters which ensure a good performance of the algorithm were selected: pop_size =30, P_{crs} =0.8, P_{mut} =0.01, the number of iterations without any improvement of the best solution found so far is set at 250.

5 Computational Experiment

The heuristic was tested on six problem instances with the number of jobs $n = 20$, 60, and 100, the number of machines $m = 2$ and 4, and one resource type. Resource requirements α_{ijr} were generated from $U[1, 9]$ ($U[a, b]$ denotes the discrete uniform distribution in the range of $[a, b]$), whereas the resource availability, W_1, was set at 10. Processing times at the first stage, p_{ij}, were generated from $U[1, 200]$ and $U[1, 300]$ for instances with 2 and 4 machines, respectively, whereas processing times at stage 2, s_j, were generated from $U[1, 100]$ for all instances.

Table 1. Results of a computational experiment

n	m	LB	$C_{\max}$		CPU time (s)
20	2	1062.00	1098.84		0.74
	4	1005.00	1049.89		2.22
60	2	3191.00	3191.00	*	4.00
	4	2966.00	2966.00	*	12.61
100	2	5032.00	5032.00	*	10.39
	4	5268.00	5269.00		16.50

The results of the experiment are presented in Table 1. The first two columns show the size of an instance. In column 3, the lower bound on the makespan is presented which is defined as follows: $LB = \max\{LB_1, LB_2\}$, where $LB_1 = C^*_{CG} + \min_{j=1,\ldots,n}\{s_j\}$, C^*_{CG} denotes the minimal time needed to complete all jobs at stage 1 (obtained by the CG algorithm), and $LB_2 = \min_{\substack{i=1,\ldots,m \\ j=1,\ldots,n}}\{p_{ij}\} + \sum_{j=1}^{n} s_j$. The makespan for the two-stage flowshop scheduling problem found by the heuristic and, the CPU time (in seconds) are shown in columns 4 and 5, respectively.

From the Table we can see that the performance of the heuristic is good. For three instances (indicated by *) the optimal solution was found. For the instance with 100 jobs and 4 machines the solution is near-optimal one. The results suggest that the heuristic is able to produce very good results especially for problems with a great number of jobs.

The computation time increases with the number of jobs and machines but does not exceed a few seconds for all examined instances.

6 Conclusions

In this paper a heuristic combining the column generation algorithm with the genetic algorithm for solving the two-stage flowshop preemptive scheduling problem with parallel unrelated machines and resource constraints at the first stage, and a single machine at the second stage has been developed. The heuristic seems to be a promising tool for solving the considered problem.

Bibliography

[1] Barnhart, C, E. Johnson, G. Nemhauser, M. Savelsbergh and P. Vance (1998). Branch and price: column generation for solving huge integer problems. *Oper. Res.* 46, 316-329.

[2] Brah, S.A. and L.L. Loo (1999). Heuristics for scheduling in a flow shop with multiple processors. *Europ. J. of Opernl Res.*113,113-112

[3] Chen, B. 1995. Analysis of classes of heuristics for scheduling a two-stage flow shop with parallel machines at one stage, *J. of Opernl Res. Soc.* 46, 234-244.

[4] Chen, Z.-L. and C.-Y. Lee (2002). Parallel machine scheduling with a common due window, *Europ. J. of Opernl Res.* , 136, 512-527.

[5] Dantzig, G.B. and P. Wolfe (1960). Decomposition principle for linear programs. *Oper. Res.* 8, 101-111.

[6] Figielska, E. (1999). Preemptive scheduling with changeovers: using column generation technique and genetic algorithm. *Comp. and Ind. Engin.* 37, 63-66.

[7] Gilmore, P.C. and R.E. Gomory (1961). A linear programming approach to the cutting-stock problem, *Oper. Res.* 9, 849-859.

[8] Goldberg D.E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley.

[9] Gupta J.N.D. (1988). Two stage hybrid flowshop scheduling problem. *J. of Opernl Res. Soc.* 39, 359-364.

[10] Haouari, M and R. M'Hallah (1997). Heuristic algorithms for the two-stage hybrid flowshop problem, *Oper. Res. Let.* 21, 43-53.

[11] Holland J.H. (1975). *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI.

[12] Hoogeveen J.A., J.K. Lenstra and B. Veltman (1996). Preemptive scheduling in a two-stage multiprocessor flow shop is NP-hard. *Europ. J. of Opernl Res.* 89, 172-175.

[13] Linn, R, W. Zhang (1999). Hybrid flow shop scheduling: a survey. *Comp. and Ind. Engin.* 37, 57-61.

[14] Oğuz, C, M.F. Ercan, T.C.E. Cheng and Y.F. Fung (2003). Heuristic algorithms for multiprocessor task scheduling in a two-stage hybrid flow shop, *Europ. J. of Opernl Res.* 149, 390-403.

[15] Ruiz, R., C. Maroto (2006). A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. *Europ. J. of Opernl Res.* 169, 781-800.

[16] Slowinski, R. (1980). Two aproaches to problems of resource allocation among project activities - A comparative study. *J. Opl Res. Soc.* 31, 711-723.

[17] Suresh, V (1997). A note on scheduling of two-stage flow shop with multiple processors, *Int. J. of Prod. Econ.* 49, 77-82.