Combining a Genetic Algorithm with a Column Generation Algorithm for Solving a Scheduling Problem

Ewa Figielska[1]

[1] Warsaw School of Computer Science, Warsaw, Poland,
e-mail: efigielska@poczta.wwsi.edu.pl

Abstract. This paper deals with the problem of minimal makespan scheduling in a two-stage flowshop with parallel unrelated machines and renewable resources at the first stage and a single machine at the second stage. A heuristic for solving this problem is developed which combines a genetic algorithm with a column generation algorithm starting from a good feasible solution provided by a fast approximate algorithm. The results of a computational experiment show that the proposed heuristic requires almost three times less CPU time than the heuristic with a standard column generation algorithm, while producing the same quality solutions.

1 Introduction

This paper deals with the problem of preemptive scheduling in a flowshop with unrelated parallel machines and limited renewable resources. The objective is to minimize the makespan. A flowshop with parallel machines (FSPM is a system which consists of a set of two or more processing centers (processing stages) with at least one center having two or more parallel machines. A job in such a system consists of a sequence of operations processed at successive stages, and all jobs pass through processing stages in the same order. At a stage with parallel machines a job can be processed on any machine. The flowshops with parallel machines have received considerable attention from researchers during last years. However most literature in this area addresses problems with identical parallel machines and nonpreemptive jobs, and include among others [1,2,9,10,13,14]. A handful of papers concerns makespan minimization problem in the flowshop with parallel machines that are not identical [15,18]. Scheduling problems with additional renewable resource constraints have been widely investigated only for the one stage systems [16,17,19,4]. In the previous works [5,6] we have extended this research to the more general and encountered in real-life systems case, when a stage with additional renewable resources is a part of a multistage system.

In this paper a new heuristic is developed for finding minimum makespan schedule in the two-stage flowshop with parallel unrelated machines and renewable resource constraints at the first stage and a single machine at the second stage. This heuristic combines a genetic algorithm (GA) with a column generation (CG) algorithm starting from a suboptimal solution. A CG algorithm is an iterative procedure which in successive iterations, solves linear programming (LP) problems each of which is of greater size than the previous one. Due to this fact, the computational effort needed for solving an LP problem at a single iteration, increases as the

number of iterations grows. However, the number of iterations can be reduced when the CG algorithm starts from a suboptimal feasible solution. In this paper we propose a fast approximate algorithm which yields a good initial solution for the CG algorithm. The developed heuristic is tested as to its computation time and effectiveness in finding a minimum makespan flowshop schedule and compared with the heuristic using the standard column generation algorithm with a simple initial solution (proposed by the author in [5]).

2 Problem Description

The considered in this paper flowshop scheduling problem with parallel machines and additional resource constraints can be described as follows. There are n preemptive jobs to be processed at two stages in the same technological order, first at stage 1 and then at stage 2. At stage 1 there are m parallel unrelated machines, stage 2 has one machine. Jobs for their processing at stage 1, besides machines, require additional resources. There are l types of renewable resources. A resource of type r (r=1,…,l) is available in an amount limited to W_r units at a time. The total usage of resource r at any moment by jobs simultaneously executed on parallel machines cannot exceed the availability of this resource. Job j during its processing on machine i at stage 1 uses α_{ijr} units of the resource of type r at every moment. At stage 1, a job can be processed on any of the parallel machines, and its processing times may be different on different machines. The processing time of job j (j=1,…,n) is equal to p_{ij} if it is executed on machine i (i=1,…,m) at stage 1, and the processing time of job j at stage 2 is equal to s_j. The objective is to find a feasible schedule which minimizes makespan, $C_{\max}$, which is equal to the maximum job completion time at stage 2.

This problem is NP-hard in the strong sense since a simpler problem of preemptive scheduling in the two-stage flowshop without resource constraints with two identical parallel machines at one stage and one machine at another is NP-hard in the strong sense [12].

3 Heuristic

The proposed in this paper heuristic for solving the considered problem proceeds in three steps. In the first step, a two-phase approximate algorithm creates a suboptimal solution to the resource constrained scheduling problem which occurs at stage 1. In the second step the solution obtained in the first step is employed as an initial solution in the CG algorithm which solves to optimality the resource constrained scheduling problem at stage 1 of the flowshop. The schedule at stage 1 created by the CG algorithm is composed of a number of partial schedules. In a partial schedule at most m jobs are assigned to m machines for simultaneous processing during some period of time so that resource constraints are satisfied at every moment. The makespan of the schedule at stage 1 does not depend on the ordering of the partial schedules but completion times of jobs depend on the order in which the partial schedules are executed. So, when we change the ordering of the partial schedules, completion times of jobs at stage 1 will change. On the other hand the makespan in the flowshop (which is equal to the maximum job completion time at stage 2) depends on ready times of jobs at stage 2, which are equal to corresponding completion times at stage 1. So, the makeapan in the flowshop depends on the ordering of the partial schedules. In the third step of the heuristic, a GA is applied to finding a sequence of the partial schedules minimizing the makespan in the flowshop. For each partial schedule sequence generated in the search process a schedule at stage 2 is constructed using completion times of jobs at stage 1

(ready times of jobs at stage 2 are equal to corresponding completion times at stage 1) and processing times at stage 2, and then the makespan in the flowshop is determined. The schedule in the flowshop consists of the schedule at stage 1 and the schedule at stage 2.

3.1 Approximate Two-Phase Algorithm

The idea of the proposed approximate algorithm is based on the two-phase method developed by Slowinski [17] for solving to optimality the problem of resource constrained unrelated machine scheduling with job resource requirements of 0-1 type.

The approximate algorithm proposed in this paper produces a suboptimal solution to the more general problem of scheduling of parallel unrelated machines with arbitrary job resource requirements (resource requirements are arbitrary integers). This algorithm proceeds in two phases.

In the first phase, the LP problem with relaxed resource constraints (resource constraints at every moment are replaced by the resource constraints over the time needed for finishing all jobs) is solved. The time T needed for finishing all jobs is minimized. Besides T, the values of t_{ij} (i=1,...,m, j=1,...,n), where t_{ij} is the time during which job j is processed on machine i, are found.

In the second phase, having the values of t_{ij} and T, a schedule which consists of a number of partial schedules is constructed by an iterative procedure. In successive iterations partial schedules are created. For this purpose criticalities of machines $\varepsilon_i^I = \sum_{j=1}^{n} t_{ij} / T$ (i=1,...,m), jobs $\varepsilon_j^{II} = \sum_{i=1}^{m} t_{ij} / T$ (j=1,...,n) and resources $\varepsilon_r^{III} = \sum_{i=1}^{m} \sum_{j=1}^{n} \alpha_{ijr} t_{ij} / W_r T$ (r=1,…,l) are calculated. Pairs (i, j) (determining the assignment of jobs to machines) with the greatest values of criticalities ε_i^I, ε_j^{II}, and ε_r^{III}, are chosen. In each iteration, the value of t_{ij} decreases by the current partial schedule length. The procedure stops when $t_{ij} = 0$ for each i and j.

3.2 Column Generation Algorithm

The theoretical basis of the CG technique has been provided by Dantzig and Wolfe in [3]. Using a column generation technique, we avoid the difficulty of explicitly generating all columns of a problem, which in our work correspond to all feasible partial schedules, by working with only a subset of the columns and adding new columns as needed (when they improve the solution). Such an approach was suggested by Gilmore and Gomory [7] for solving cutting stock problems.

The CG algorithm is an iterative procedure which starts from an initial feasible solution and, in subsequent iterations, in turns generates a new column and solves a linear programming (LP) problem with all columns generated so far. Due to this fact, the size of the LP problem (and consequently the computational effort needed for solving this problem) at a single iteration increases as the number of iterations grows. The number of iterations in which a CG algorithm finds an optimal solution can be however reduced by improving the initial solution of the algorithm.

In this paper we use the CG algorithm with a suboptimal initial solution for solving to optimality the problem of resource constrained preemptive scheduling of parallel unrelated machines which occurs at stage 1 of the considered flowshop. The suboptimal solution to this problem is provided by the approximate two-phase algorithm presented in Section 3.1. The details of the CG algorithm are presented in a previous paper [5].

3.3 Genetic Algorithm

In this paper, a GA finds the ordering of the partial schedules which provides the schedule for the two-stage flowshop with minimum makespan.

A GA [11] 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.

In this paper, a solution to the sequencing problem solved by the GA is coded as a single chromosome whose genes represent the indices of partial schedules. An initial population of chromosomes is randomly generated. The value of the 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. As a selection method the binary tournament selection is used. The two-point crossover operator PMX [8] is applied to each pair of parent chromosomes with the probability P_{crs}. 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 the probability P_{mut}. 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: the size of a population was set at 20, P_{crs} =0.8, P_{mut} =0.01, the number of iterations without any improvement of the best solution found so far was set at 500.

4 Illustrative Example

To illustrate the problem and the solution method we present the following example. Consider the case of the two-stage flowshop with 2 machines at stage 1 and a single machine at stage 2. The number of jobs n =10, the resource availability at any moment, W_1=10. Job processing times and resource requirements are shown in Figure 1.

Figure 2 presents two schedules for the first stage of the flowshop: (a) the suboptimal schedule created by the approximate algorithm, and (b) the optimal schedule provided by the CG algorithm starting from the suboptimal schedule depicted in Figure 2a. We can observe that the schedule provided by the CG algorithm contains 6 partial schedules with the same assignments of jobs to machines as the assignments created by the approximate algorithm.

Figure 3 presents two flowshop schedules for this instance. Each flowshop schedule consists of the first stage schedule and the second stage schedule. The first stage schedules in Figures 3a and 3b are composed of the same 10 partial schedules. In each of the partial schedules at most 2 jobs are processed simultaneously and the total resource usage does not exceed the resource availability, W_1=10, e.g. in partial schedule S_1 jobs 8 and 5 are processed simultaneously and use at every moment 1 and 8 units of the resource, respectively. The first stage schedules in Figures

3a and 3b have the same length, but completion times of jobs in the first stage schedule in Figure 3a are different from those in the first stage schedule in Figure 3b. For example, in Figure 3a, job 7 completes its processing at stage 1 at 44 time units, and at 1 time units in Figure 3b. The processing of a job on the machine at stage 2 can be started after completing its processing at stage 1. The machine at stage 2 starts working when at least one job has completed its processing at stage 1 (before that, the machine of stage 2 remains idle). After all jobs finish their processing at stage 1, the machine at stage 2 works for at least the time needed for completing all jobs from the last partial schedule (the machines of stage 1 remain idle for that time). Besides, the machine at stage 2 remains idle while it is waiting for finishing processing a job at stage 1 (after it completes all jobs whose processing have been already finished at stage 1). For example in Figure 3a, the machine at stage 2 remains idle (does not work) when it is waiting for finishing processing job 10 at stage 1.

Figure 3a presents the flowshop schedule with a randomly chosen sequence of the partial schedules. Figure 3b shows the schedule with the sequence of the partial schedules which was obtained by the GA so as to minimize the makespan in the flowshop. We can see that the shortening of the schedule achieved by the GA is significant.

job processing times at stage 1

job	machine 1	machine 2
1	8	20
2	21	30
3	11	11
4	15	2
5	23	18
6	25	4
7	1	34
8	30	33
9	27	3
10	40	16

job processing times at stage 2

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

resource requirements at stage 1

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

resource availability = 10

Figure 1. Data for the illustrative example

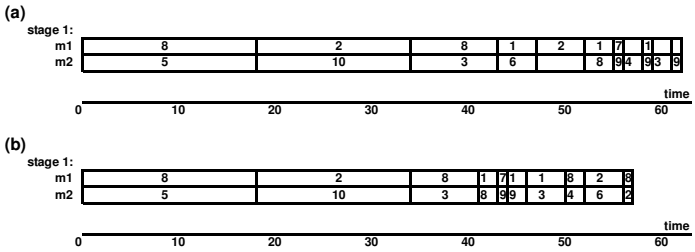

Figure 2. Schedules for the first stage of the flowshop. (a) A suboptimal schedule created by the two-phase approximate algorithm (b) An optimal schedule obtained by the CG algorithm starting from the suboptimal solution.

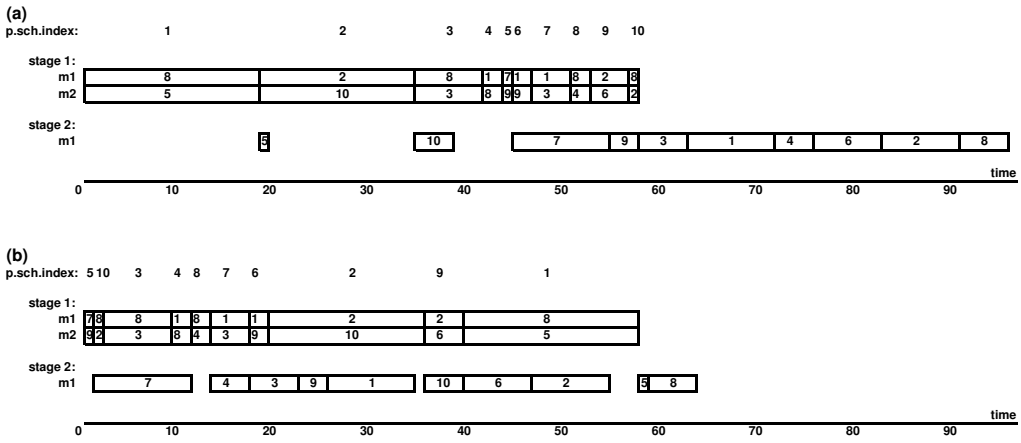

Figure 3. The resulting flowshop schedules: (a) the schedule with a random sequence of the partial schedules, (b) the final schedule with the sequence of the partial schedules minimizing the makespan in the flowshop.

5 Computational Experiment

A computational experiment was carried out to evaluate the performance of the proposed heuristic. The number of jobs was considered to be $n = 10$, 30, and 60, the number of machines, m, at stage 1 was set at 2 and 4. There was one additional resource type. Resource requirements were generated from $U[1,9]$ ($U[a,b]$ denotes the discrete uniform distribution in the range of $[a,b]$), whereas the resource availability was set at 10. Job processing times at stage 1 were generated from $U[1,200]$ and $U[1,400]$, job processing times at stage 2 were generated from $U[1,100]$.

To evaluate the quality of the heuristic solutions we used the values of the relative percentage deviation of solutions from the lower bound on the optimal makespan:

$$\delta = 100\% \times (C_{\max} - LB)/LB,$$

where $C_{\max}$ is the best makespan found by the algorithm, and LB is the lower bound on the optimal makespan defined as $LB = \max\{LB_1, LB_2\}$, where $LB_1 = \sum_{j=1}^{n} s_j + \min_{i=1,\ldots,m,\, j=1,\ldots,n}\{p_{ij}\}$ and $LB_2 = C_1^* + \min_{j=1,\ldots,n}\{s_j\}$, where C_1^* denotes the minimal makespan for the problem occurring at stage 1 (i.e. the minimum time needed to finish processing all jobs at stage 1).

All programs for the algorithms presented in the paper were run on a 2.4 GHz Celeron Processor. The LP problems were solved using CPLEX optimizer.

The results of a computational experiment are presented in Table 1. In this table, deviations, δ, and CPU times are presented both for the new heuristic developed in this paper and for the heuristic with a standard column generation algorithm. In the standard CG algorithm, a simple initial solution was created so that each job was assigned to a machine on which it required the smallest time to be completed, and in each partial schedule only one machine worked. In column 8 of Table 1, the ratio of the CPU time of the new heuristic to the CPU time of the heuristic with

a standard CG algorithm, $\sigma = \frac{\text{CPU time of the new heuristic}}{\text{CPU time of the heuristic with a standard CG algorithm}}$, is indicated. The numbers of iterations for the new CG algorithm and the standard CG algorithm are also shown.

Table 1. Computational results

			δ (%)		CPU time (s)			Number of iterations	
n	m	Job processing times at stage 1	New Heuristic	Heu. with standard CG alg.	New Heuristic	Heu. with standard CG alg.	σ	New CG alg	Standard CG alg.
10	2	U[1,200]	0.00	0.00	0.44	1.38	0.32	3	14
		U[1,400]	2.33	2.33	0.72	1.53	0.47	9	15
	4	U[1,200]	0.00	0.00	0.89	1.95	0.46	10	20
		U[1,400]	0.00	0.00	1.04	1.75	0.59	13	18
30	2	U[1,200]	0.00	0.00	2.69	8.24	0.33	34	85
		U[1,400]	1.37	1.32	2.05	5.63	0.36	25	55
	4	U[1,200]	0.00	0.00	3.09	7.78	0.40	36	69
		U[1,400]	0.00	0.00	2.76	6.77	0.41	31	64
60	2	U[1,200]	0.00	0.00	4.73	22.22	0.21	39	161
		U[1,400]	0.73	0.73	6.26	23.03	0.27	66	166
	4	U[1,200]	0.00	0.00	8.83	24.34	0.36	85	159
		U[1,400]	0.00	0.00	7.86	25.72	0.31	73	166
	Average		0.37	0.37	3.45	10.86	0.37	68.67	116.00

Job processing times at stage 2 were generated from U[1,100].

From Table 1 we can see that, with respect to the CPU time, the new heuristic significantly outperforms the heuristic with a standard CG algorithm, and it produces the results of the same very good quality as the previous heuristic.

The values of the ratio, σ, vary from 0.21 (what means that the new heuristic is almost 5 times faster than the previous one) to 0.59 (what means that the new heuristic is almost twice faster than the previous one). We can observe that the ratio, σ, decreases as the number of jobs grows, and tends to increase as the number of machines increases.

6 Conclusions

In this paper the heuristic combining a GA with a CG algorithm for solving the problem of scheduling in the two-stage flowshop with unrelated machines and resource constraints has been developed. An approximate algorithm has been proposed for generating a good initial solution for the CG algorithm. On the basis of the results of a computational experiment we can conclude that the proposed heuristic is able to produce very good quality solutions using much less computational effort than the heuristic with a standard column generation algorithm.

Bibliography

[1] 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

[2] 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.
[3] Dantzig, G.B. and P. Wolfe (1960). Decomposition principle for linear programs. *Oper. Res.* 8, 101-111.
[4] Figielska, E (1999). Preemptive scheduling with changeovers: using column generation technique and genetic algorithm, *Computers and Industrial Engineering* 37, 63-66.
[5] Figielska, E. (2006). Solving a flowshop scheduling problem with resource constraints, *Evolutionary Computation and Global Optimization, Prace naukowe PW, Elektronika* 156, 139-146.
[6] Figielska, E. (2006). Combining an optimizing method with a genetic algorithm to solve a flowshop scheduling problem with additional resources, *Proceedings of the 11th IEEE International Conference on Emerging Technologies and Factory Automation*, 1080-1086, Prague, Czech Republic.
[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] Nowicki, E., Smutnicki Cz. (1995). The flow shop with parallel machines: A tabu search approach. *Europ. J. of Opernl Res.* 106, 226-253.
[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 approaches to problems of resource allocation among project activities – A comparative study. J. Opl Res. Soc. 31, 711-723.
[17] Slowinski, R. (1981), L'ordonanacement des taches preemptives sur les processeurs independants en presence de ressources supplementaires. *RAIRO Inform./Comp. Science*, vol.15, No.2, 155-166.
[18] Suresh, V (1997). A note on scheduling of two-stage flow shop with multiple processors. *Int. J. of Prod. Econ.* 49, 77-82
[19] De Werra, D. (1988). On the two-phase method for preemptive scheduling. *Europ. J. of Opernl Res.* 37, 227-235.