Zeszyty Naukowe WWSI, No. 11, Vol. 8, 2014, pp. 29-39

Algorithms Using List Scheduling and Greedy Strategies for Scheduling in the Flowshop with Resource Constraints

Ewa Figielska*

Warsaw School of Computer Science

Abstract

The paper addresses the problem of scheduling in the two-stage flowshop with parallel unrelated machines and renewable resource constraints. The objective is minimization of makespan. The problem is NP-hard. Fast heuristic algorithms using list scheduling and greedy strategies are proposed. For evaluation of the performance of the algorithms computational experiments are performed on randomly generated test problems, and results are reported.

Keywords – Flowshop, Parallel machines, Resource constraints, Scheduling, Heuristic

1 Introduction

The paper addresses the problem of scheduling in the two-stage flowshop with parallel unrelated machines and renewable resource constraints.

A two-stage flowshop with parallel machines (also called a two-stage hybrid flowshop) is a system which consists of two 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. Each operation is performed at one processing

* E-mail: efigielska@poczta.wwsi.edu.pl

Manuscript received July 28, 2014

stage. All jobs pass through the stages in the same technological order. At a stage with parallel machines a job can be processed on any machine. In the considered flowshop, the parallel machines are unrelated which means that processing times of a job are different on different machines. Jobs, during their processing on the machines, use some additional renewable resources which are available in limited quantities at any time. All required resources are granted to a job before its processing begins and they are returned by the job after finishing its processing at a stage. Job resource requirements are arbitrary integer numbers (different for different jobs and different machines) or are assumed to be of 0-1 type (i.e. a job uses 1 unit of a resource or does not use a resource). The objective is to find a feasible schedule with the minimal makespan (length). The problem is known to be NP-hard.

The problem of scheduling in the two-stage hybrid flowshop with resource constraints was solved in our previous works [1,2,3,4] with the use of linear programming and metaheuristics.

Table 1: Machine selection rules

Machine selection rule M#	Description
M1	Random selection.
M2	For a given job choose a machine that becomes free first.
M3	For a given job choose a machine on which this job has the smallest processing time.
M4	For a given job choose a machine on which this job has the smallest average resource requirements.

In this paper, we propose algorithms applying list scheduling and greedy strategies, which promise to perform very fast. First, we create a list of jobs according to a priority rule based on the Johnson's rule [5] (the Johnson algorithm provides an optimal makespan for the classical two-machine flowshop). Next, if the first stage has one machine, the jobs from the list are executed one by one on this machine. If stage 1 has parallel unrelated machines, we choose the machines to which successive jobs from the list are assigned. Each job starts processing on the assigned to it machine when this machine is free and resource constraints are satisfied. If the parallel machines are at stage 2, a machine is assigned to a job after finishing its processing at stage 1. The choice of the machine for a given job is made in a greedy way according to the one of the machine selection rules presented in Table 1 which promises a short schedule at the stage with parallel machines.

The remainder of the paper is organized as follows. In the next sections, the algorithms are presented. The results of the computational experiment are reported in Section 3. Section 4 summarizes the paper.

2 Algorithms

2.1 Algorithms for scheduling in the flowshop with parallel machines at stage 1 and one machine at stage 2

In this section, we consider the problem of scheduling in a two-stage flowshop with parallel unrelated machines at the first stage and a single machine at the second stage. All the jobs are ready for processing at stage 1 at time 0. The jobs during their processing use additional renewable resources which are available in limited quantities at any time. Resource requirements of jobs are arbitrary integer numbers (different for different jobs and different machines) or are assumed to be of 0-1 type.

The algorithms solving this problem proceed as follows.

1. Create a list of jobs ordered according to the priority rule.
2. For each successive job j from the list:
3. Choose machine i for processing job j at stage 1 according to a machine selection rule.
4. Start processing job j on machine i as early as possible so that resource constraints are satisfied at any moment.
5. For each job the processing of which finished at stage 1, start its processing at stage 2 when the second stage machine becomes free.

The priority rule used in the algorithms is based on the Johnson's rule and operates as follows. First, it sorts jobs with $p_j^{min} \leq s_j$ in non-decreasing order of p_j^{min}. Then, it sorts the remaining jobs in non-increasing order of s_j, where s_j is the processing time of job j at stage 2, $p_j^{min} = \min_{i=1,\ldots,m} p_{ij}$, where m is the number of machines and p_{ij} is the processing time of job j on machine i at stage 1. The machine selection rules used by the algorithms are presented in Table 1.

The algorithms for solving the problem with parallel unrelated machines at stage 1 and one machine at stage 2, implementing rules M1-M4 will be, respectively, referred to as RM1-RM4.

2.2 Algorithms for scheduling in the flowshop with one machine at stage 1 and parallel machines at stage 2

In this section, we consider the two-stage flowshop with one machine at stage 1 and parallel unrelated machines at stage 2. Resources available in limited quantities at any moment are shared among the stages, which means that they can be used at the same time at different stages. So, if a resource is used at one stage, its quantity available at the same time at the other stage lessens. Resource requirements of jobs are assumed to be arbitrary integers.

At stage 2 with parallel machines successive jobs have different ready times which are equal to their completion times at stage 1. The resource availability at stage 2 is different in different time intervals between the ready times of the successive jobs. Therefore, the algorithms developed in this section proceed in a different way than those presented in the previous section. These algorithms can be outlined as follows.

1. Create a list of jobs ordered according to the priority rule and execute jobs from the list one by one on the machine at stage 1.
2. For each successive time interval k:
3. Let J_k be the set of all jobs that are available for processing at stage 2 in time interval k (i.e. jobs that finished their processing at stage 1 before interval k begins) and were not yet executed at stage 2.
4. For each successive job j from the job list, which belongs to set J_k:
5. Choose machine i for processing job j at stage 2 according to a machine selection rule.
6. Start processing job j on machine i in interval k as early as possible so that resource constraints are satisfied at every moment. If job j cannot start in interval k, remove it from set J_k.

As a priority rule, we use the best priority rule from [4]. It is based on the Johnson's rule, but beside jobs processing times, it takes into account also resource requirements of jobs. This rule operates as follows. First, it sorts jobs with $p_j \le s_j^{min}$ in non-decreasing order of $p_j/\bar{\alpha}_j$. Then, it sorts the remaining jobs in non-increasing order of $s_j^{min}\bar{\beta}_j$, where: p_j is the processing time of job j at stage 1, $s_j^{min} = \min_{i=1,\dots,m} s_{ij}$, where m is the number of machines and s_{ij} is the processing time of job j on machine i at stage 2, $\bar{\alpha}_j = \frac{1}{l}\sum_{r=1}^{l} \alpha_{jr}/W_r$, $\bar{\beta}_j = \frac{1}{l}\sum_{r=1}^{l} \beta_{kjr}/W_r$, where $\{k = i: s_{ij} = s_{ij}^{min}, i = 1, \dots, m\}$, l is the number of resources, W_r is the availability of

resource r, α_{jr} and β_{ijr} are the numbers of units of resource r required by job j, respectively, on the machine at stage 1 and on machine i at stage 2.

The algorithms for solving the problem with resources shared among the stages use rules M1-M4, and will be, respectively, referred to as SM1-SM4.

3 Computational experiments

In this section, the results of a computational experiment are presented.

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

$$\delta = \frac{C_{max}-LB}{LB} \times 100\% \quad (1)$$

where C_{max} is the best makespan (maximal completion time) found by the heuristic algorithm, and LB is the lower bound on the optimal makespan. We use the lower bounds derived in [2,3,4].

Results are analyzed in terms of the following parameters:

- the number of jobs n,
- the number of machines m,
- the resource availability W,
- the ranges of job processing times at stage 1, $[p]$, and at stage 2, [s],
- the ranges of job resource requirements at stage 1, $[\alpha]$, and at stage 2, $[\beta]$,
- the values of the dominance factor, θ (for the problems with parallel machines at stage 1), which is defined as the ratio of the optimal length of the schedule at stage 1 to the sum of the job processing times at stage 2 (see [2]).

The values of θ close to 1 indicate that the stages are balanced, which is the most important case from the practical point of view, but, which, at the same time, is the most difficult one to solve. If θ is less or greater than 1, one of the stages dominates the other.

3.1 Computational experiments for the problem with parallel machines at stage 1, one machine at stage 2 and resource requirements of 0-1 type

In this section we present the results of the experiment carried out for the problem with parallel machines at stage 1 and one machine at stage 2 in the case when jobs have resource requirements of 0-1 type (i.e. a job uses 1 unit of a resource or does not use a resource). In the experiment, the data sets from [2] were used.

Table 2: Computational results for the problem with parallel machines at stage 1, one machine at stage 2 and resource requirements of 0-1 type

					δ [%]			CPU time [s]	A5T	
n	m	W	$[p]$	θ	RM1	RM2	RM3	RM2	δ [%]	CPU time [s]
40	4	2	[30,60]	0.68	19.5	3.0	0.9	0.008	0.23	1.34
			[50,100]	1.12	72.7	41.5	43.3	0.011	1.59	1.29
			[70,140]	1.55	82.6	46.0	42.7	0.011	1.97	1.26
40	8	4	[50,100]	0.62	18.2	1.8	0.4	0.016	0.12	1.31
			[70,140]	0.86	58.5	19.1	16.8	0.018	0.34	1.28
			[90,180]	1.12	87.1	43.8	40.9	0.017	2.30	1.31
			[110,220]	1.34	104.9	55.5	50.0	0.016	4.73	1.31
			[130,260]	1.59	104.7	54.8	52.1	0.016	3.84	1.30
40	4	4	[30,60]	0.80	56.6	19.7	23.6	0.011	0.21	1.19
			[50,100]	1.36	99.9	48.2	60.5	0.012	1.19	1.20
			[70,140]	1.86	104.4	52.2	61.4	0.011	1.08	1.20
40	8	8	[50,100]	0.77	47.6	10.7	10.3	0.024	0.28	1.26
			[70,140]	1.07	88.2	37.3	37.4	0.024	1.05	1.33
			[90,180]	1.38	109.0	52.2	55.0	0.024	3.23	1.33
			[110,220]	1.78	103.0	49.9	52.7	0.023	2.35	1.28
			[130,260]	2.02	103.2	50.4	53.4	0.023	2.20	1.32
80	4	2	[30,60]	0.66	16.2	1.2	0.3	0.059	0.09	2.44
			[50,100]	1.08	80.6	45.6	41.9	0.059	0.83	2.49
			[70,140]	1.52	80.0	47.5	44.2	0.048	0.93	2.53
80	8	4	[50,100]	0.56	13.2	0.6	0.1	0.088	0.07	2.62
			[70,140]	0.79	55.1	20.2	12.5	0.089	0.12	2.64
			[90,180]	1.03	91.9	47.1	43.5	0.090	0.69	2.61
			[110,220]	1.27	99.8	55.2	49.4	0.090	2.30	2.62
			[130,260]	1.54	99.6	54.2	45.0	0.093	1.96	2.59
80	4	4	[30,60]	0.74	58.5	18.1	23.5	0.071	0.13	2.50
			[50,100]	1.23	117.6	61.0	75.7	0.072	0.59	2.54
			[70,140]	1.74	114.8	60.5	71.9	0.070	0.51	2.59
80	8	8	[50,100]	0.68	45.8	7.9	7.6	0.138	0.11	2.68
			[70,140]	0.97	99.6	43.2	45.4	0.137	0.45	2.66
			[90,180]	1.25	111.4	58.3	62.4	0.137	1.70	2.63
			[110,220]	1.50	125.3	59.9	62.8	0.135	1.71	2.63
			[130,260]	1.83	121.9	60.0	61.3	0.144	1.27	2.65
			Average		81.0	38.3	39.0	0.056	1.26	1.95

The results are shown in Table 2.

The first 5 columns of the table indicate values of the problem parameters: n, m, W, $[p]$ and θ. The range [s] was set at [10,20]. In columns 6-8 of the table, the average (over 50 problem instances) deviations δ obtained by algorithms

RM1-RM3 are presented. Column 9 shows the computation times for RM2 (computation times of the other algorithms are almost the same). The last two columns show deviations δ and computation times for algorithm A5T from our previous work [2], which uses linear programming and a tabu search algorithm.

Table 3: Computational results for the problem with parallel machines at stage 1, one machine at stage 2 and arbitrary resource requirements of jobs

				δ [%]				CPU time [s]	HS	
n	m	$[p]$	θ	RM1	RM2	RM3	RM4	RM3	δ [%]	CPU time [s]
20	2	[1,100]	0.37	4.6	2.4	0.0	2.4	0.001	0.02	1.85
		[1,200]	0.73	64.0	45.5	15.2	37.5	0.001	0.20	1.82
		[1,400]	1.52	130.4	101.0	54.5	106.9	0.001	1.55	1.84
		[1,600]	2.30	134.7	105.4	53.5	97.8	0.001	1.01	1.83
	4	[1,200]	0.35	54.0	42.2	0.8	7.3	0.002	0.02	2.69
		[1,400]	0.66	220.6	173.0	26.7	80.6	0.002	0.58	2.74
		[1,600]	1.02	335.8	289.6	69.5	149.3	0.002	4.06	2.74
		[1,800]	1.17	384.8	348.3	87.3	168.5	0.002	5.55	2.83
		[1,1000]	1.71	377.5	320.7	83.0	174.8	0.002	3.87	2.82
		[1,1200]	2.05	369.2	343.4	94.4	170.5	0.002	3.75	2.83
		[1,1400]	2.20	399.5	333.7	88.5	196.7	0.002	2.66	2.92
60	2	[1,100]	0.37	1.8	1.1	0.0	0.9	0.022	0.01	11.19
		[1,200]	0.72	71.3	50.0	15.7	42.5	0.020	0.00	11
		[1,400]	1.43	140.6	113.0	60.1	98.5	0.020	0.34	11.42
		[1,600]	2.14	141.3	114.2	62.1	100.1	0.020	0.24	11.6
	4	[1,200]	0.30	55.8	39.2	0.0	2.0	0.023	0.00	13.53
		[1,400]	0.62	207.9	174.2	22.0	59.6	0.024	0.01	14.29
		[1,600]	0.95	341.1	295.9	79.6	132.9	0.023	0.35	14.25
		[1,800]	1.22	405.8	352.9	97.3	160.6	0.024	1.46	14.15
		[1,1000]	1.50	420.3	362.9	101.1	169.0	0.023	1.02	13.89
		[1,1200]	1.89	389.7	337.1	96.7	165.0	0.025	0.67	14.14
		[1,1400]	2.19	402.6	350.0	100.6	157.5	0.025	0.55	13.59
		Average		229.7	195.3	54.9	103.7	0.012	1.27	7.73

In Table 2, we can see that the algorithms using machine selection rules M2 and M3 (let us recall that, for a given job, M2 chooses the first free machine, M3 chooses the machine with the smallest processing time of this jobs), provide about two times better results than the algorithms with rule M1 (which chooses

the machines randomly). The smallest deviations δ are obtained when dominance factor θ is less than 1 (i.e. when stage 2 dominates stage 1). The values of δ increase with increasing θ, until θ is close to 1 (i.e. stages become balanced). Further increase in θ do not deteriorate the solutions. The number of jobs and machines does not seem to affect the quality of the solutions. The average deviations δ obtained by algorithms RM2 and RM3 are slightly less than 40%. The computation times do not exceed 0.15 seconds.

Algorithm A5T provided the average deviations δ equal to 1.26% using from about 1 to about 3 seconds of computation time [2].

The computations were carried out on a PC with Celeron 2.4GHz processor and 512GB RAM.

3.2 Computational experiments for the problem with parallel machines at stage 1, one machine at stage 2 and arbitrary resource requirements

In this section, the computational experiment was carried out for the problem with parallel machines at stage 1 and one machine at stage 2 in the case when resource requirements of jobs are arbitrary integers taken from some interval. In the experiment, the data sets from [3] were used.

The results of the experiment are presented in Table 3.The first 4 columns of the table show the values of the problem parameters: n, m, $[p]$ and θ. The values of the remaining parameters are as follows: [s] = [1,100], W=10, and $[\alpha]$ = [1,10]. Columns 5-8 of the table contain the average (over 50 problem instances) values of deviations δ obtained by algorithms RM1-RM4. The computation times (of RM3 as a representative of all the algorithms) are indicated in column 9. The last 2 columns contain the results obtained in [3] by algorithm HS using column generation and simulated annealing procedures.

In Table 3, we can see that the algorithm applying rule M3 (favoring a machine with the smallest processing time for a given job) significantly outperforms the other algorithms. It provides the solutions about 4 times better than RM1 and RM2, and about 2 times better than RM4. The smallest deviations δ are obtained in the cases when the first stage schedules are about 3 times shorter than those at the second stage. For the most important problems with $\theta \approx 1$, deviations δ become greater and almost do not change with the further increase in the values of θ, i.e. for the problems where the first stage is a dominant one. The deviations δ increase with increasing the number of machines. The number of jobs seems not to influence the performance the algorithms. The average deviation δ obtained by RM3 is equal to

about 55%. The computation times of algorithms RM1-RM4 did not exceed 0.025 seconds for all tested problem instances.

Algorithm HS provided solution with average $\delta = 1.27\%$ using from about 2 about 14 seconds [3].

The computations were performed on a PC with Celeron 1.3GHz processor and 2GB RAM.

3.3 Computational experiments for the problem with one machine at stage 1, parallel machines at stage 2 and resources shared among the stages

In this section, the computational experiment was conducted for the problem with one machine at stage 1 and parallel machines at stage 2, where resources are shared among the stages. The data sets from [4] were used.

The results of the experiment are presented in Table 4. The first 3 columns of the table indicate the values of the problem parameters: W, n and m. The values of the remaining parameters are as follows: [p] = [1,100], [s] = [1,100m], [α] = [1,10] and [β] = [1,10]. In columns 4-7 of table the average (over 10 problem instances) percentage deviations δ produced by the algorithms are shown. Column 8 contains the computation times. The last two columns present deviations δ and computation times for the column generation based algorithm, denoted as A5, proposed in [4].

In Table 4, we can see that algorithm SM3 outperforms the other algorithms. It provides more than 3 times better results than algorithms SM1 and about 2 times better results than algorithms SM2 and SM4.

The performance of the algorithms considerably improves when resource constraints are weak ($W = 14$) in comparison with the case when resource constraints are strong ($W = 10$). If the number of machines grows, the performance of the algorithms deteriorates. The increase in the number of jobs slightly improves the results. On the average, deviations δ obtained by algorithm SM3 are equal to about 47% and 28% for problems with, respectively strong and weak resource constraints.

The computation times of algorithm SM3 did not exceed 0.81 seconds. They become smaller with weakening the resource constraints. The average computation times were equal to 0.27 and 0.18 seconds for problems with, respectively, strong and weak resource constraints.

Algorithm A5 used on the average 26 and 9 seconds of the computation time if resource constraints were, respectively, strong and weak, and produced the average deviations δ equal to 0.8% and 0.16%, respectively, for $W = 10$ and $W = 14$.

The computations were carried out on the PC with Celeron 1.3GHz processor and 2GB RAM.

Table 4: Computational results for the problem with one machine at stage 1, parallel machines at stage 2 and resources shared among the stages

			δ [%]				CPU time [s]	A5	
W	n	m	SM1	SM2	SM3	SM4	SM3	δ [%]	CPU time [s]
10	20	2	64.3	54.8	40.6	60.3	0.008	4.21	1.29
		4	164.5	132.6	58.8	94.9	0.0095	2.25	1.57
		6	232.7	203.2	54.1	121.4	0.0103	0.28	1.57
	40	2	59.7	55.2	40.0	61.1	0.0645	1.00	6.33
		4	123.6	122.8	49.2	83.9	0.0735	0.58	7.51
		6	211.8	200.2	61.8	99.2	0.0843	0.17	7.64
	60	2	56.1	46.6	37.1	48.1	0.2485	0.42	26.29
		4	137.0	104.7	44.2	81.5	0.283	0.10	28.02
		6	220.9	188.6	51.8	98.1	0.3112	0.16	35.53
	80	2	53.8	47.4	32.1	50.8	0.6338	0.24	53.79
		4	121.4	100.4	46.6	77.3	0.7482	0.06	62.55
		6	191.4	162.2	50.2	98.4	0.8057	0.05	77.77
Average			136.4	118.2	47.2	81.3	0.2734	0.79	25.82
14	20	2	40.7	27.5	19.6	42.4	0.0045	0.65	0.86
		4	100.9	93.9	35.9	69.6	0.0062	0.14	0.90
		6	177.9	139.2	35.9	100.5	0.0067	0.28	0.78
	40	2	37.4	28.1	18.5	27.9	0.0397	0.11	2.64
		4	87.0	61.1	27.7	50.5	0.0525	0.20	2.63
		6	162.9	122.8	42.8	98.7	0.0643	0.20	3.44
	60	2	41.2	36.4	18.3	34.4	0.1405	0.01	9.44
		4	88.9	51.0	29.1	50.5	0.1942	0.02	8.71
		6	146.6	107.1	33.9	92.1	0.2185	0.11	9.59
	80	2	18.6	14.8	9.4	20.1	0.3283	0.02	18.50
		4	64.8	48.1	29.2	46.1	0.5528	0.05	22.19
		6	133.2	94.1	29.1	78.7	0.581	0.07	22.41
Average			91.7	68.7	27.5	59.3	0.1824	0.16	8.51

4 Summary

In this paper, we proposed fast algorithms using list scheduling and greedy strategies for solving the resource constrained scheduling problem in the two-stage hybrid flowshop in two cases.

(1) Parallel unrelated machines and resources with availability limited at any moment are at the first stage. Moreover job resource requirements are of 0-1 type or arbitrary integers.
(2) The flowshop has parallel unrelated machines at stage 2. The resources with availability limited at any moment are shared among the stages. Jobs have arbitrary resource requirements. In both cases the objective was minimization of the makespan.

The extensive computational experiment was carried out using 2940 randomly generated problem instances. To evaluate the performance of the proposed algorithms the values of percentage deviation δ of the heuristic makespan from the lower bound on the optimal makespan were calculated. The results of the experiment show that the algorithms used in this paper are fast. The computation time and deviation δ of the best algorithms, on the average over all the instances used in the experiment, are equal, respectively, to 0.1 seconds and 42.7%.

We compared the results obtained in this paper with the results reported in our previous works [2,3,4] where the same problems were solved by the algorithms using linear programming and metaheuristcs. The computation time and deviation δ of those algorithms, on the average over all the instances used in the experiment, are equal, respectively, to 8.3 seconds and 1.0%.

References

[1] E. Figielska. A new heuristic for scheduling the two-stage flowshop with additional resources, *Computers & Industrial Engineering* 54, 750-763, 2008.
[2] E. Figielska. Linear programming & metaheuristic approach for scheduling in the hybrid flowshop with resource constraints. *Control and Cybernetics* 40 (4), 1209-1230, 2011.
[3] E. Figielska. A genetic algorithm and a simulated annealing algorithm combined with column generation technique for solving the problem of scheduling in the hybrid flowshop with additional resources, *Computers &Industrial Engineering* 56, 142-151, 2009.
[4] E. Figielska. A heuristic for scheduling in a two-stage hybrid flowshop with renewable resources shared among the stages, *European Journal of Operational Research* 236(2), 433–444, 2014.
[5] S.M. Johnson. Optimal two- and three-stage production schedules with setup times included, *Naval Research Logistics Quarterly* 1, 61-68, 1954.