Resource constrained scheduling in permutation flowshop problems

Michele Ciavotta
Università degli Studi di Milano-Bicocca
62 PUBLICATIONS **902** CITATIONS
SEE PROFILE

Rubén Ruiz
Universitat Politècnica de València
169 PUBLICATIONS **8,559** CITATIONS
SEE PROFILE

Resource constrained scheduling in permutation flowshop problems

Michele Ciavotta, Rubén Ruiz

Grupo de Sistemas de Optimización Aplicada, Instituto Tecnológico de Informática, Universidad Politécnica de Valencia. Camino de Vera s/n Edif. 8G Acc. B- 46022 Valencia, Spain
mciavotta@iti.upv.es, rruiz@eio.upv.es

Abstract

This short paper gathers the initial results obtained after including the consideration of additional resources, like for example, tools, fixings or man power, into flowshop scheduling problems. The literature on scheduling assumes that the only limiting resource is the machines. However, in real life, personnel operating the plant could be scarcer than the machines themselves. More in details, we consider a single resource with a non-negative and integer consumption from all the tasks in a flowshop. The optimization criterion is the minimization of the maximum completion time or makespan. An important consideration is that resource constrained flowshop problems can be easily transformed into the well-known resource constrained project scheduling problem or RCPSP. Therefore, a natural question is to see how good metaheuristics for the RCPSP behave for the considered resource constrained scheduling setting studied in this paper. Initial results show that simple adaptations of existing metaheuristics for the regular flowshop with the consideration of additional resources behave much better than RCPSP metaheuristics.

1 Introduction

The flowshop problem (FSP) is a typical production configuration where there is a set N of n independent jobs that have to be processed with a fixed, known in advance processing time p_{ij} on a set M of m machines. Each job j, $j = 1,\ldots,n$ has to be processed by all the m machines in a fixed sequence. The machine route is the same for all jobs and each machine processes the same sequence of jobs. Furthermore, no machine can process more than one job at the same time and once the processing of a job by a given machine has started, it has to continue until completion.
The FSP has been often criticized for being too theoretical as most real industry settings seldom fit into this model. Actually, there is a so-called "*scheduling gap*" between the cases studied in the literature and the manufacturing problems arising in real-life. In order to partially fulfill this gap, we study in this short paper the flowshop problem with the additional consideration of renewable resources that the processing of jobs needs, apart from machines. In this scenario, we also model, for example, complex fixtures, tooling or manpower needed to operate machines and/or jobs.
Although the first works on the topic of scheduling with limited resources have been published more than forty years ago, those studies were seldom directed to the flowshop problem. Indeed the parallel machines and jobshop problems are far more popular [7][11][17]. Moreover, the majority of papers that can be retrieved from the literature are concerned to very theoretical and simplified cases as 0-1 resource consumption, for example, or processing times depending on the quantity of the resources assigned. Blazewicz et al. (1983) [3] proposed a classification scheme for resource constraints and investigated the computational complexity of parallel machines, flowshop and openshop with unit processing time jobs, precedence constraints and maximum completion time. Janiak in [9] and [10] studied an extension of the two-machine flowshop problem to the case with task processing times being inversely proportional to or linearly compressible by the used amounts of continuously divisible resources. He studied the complexity, some efficiently solvable cases and proposed a heuristic procedure to tackle that problem. Huq et al. (2004) [8] described the development of a mixed integer linear programming model for a flowshop with multi-processor workstations and workers to assign to the machines. More recently, Ruiz-Torres and Centeno (2008) [14] consider a permutation flowshop problem

with secondary resources with the objective of minimizing the number of tardy jobs. As in previous cases, in this paper the amount of resources assigned influences the processing times of the jobs. Finally, Figielska in [4-6] tackled the problem of preemptive scheduling in a two-stage hybrid flowshop with one machine or parallel unrelated machines at each stage and renewable resources at both the stages. The resource requirements are of a 0-1 type or arbitrary integers. The objective is the minimization of makespan. Several heuristic approaches as well as more complex method are presented.
This paper gathers the initial findings of a work-in-progress and some early results obtained from simple metaheuristics like Iterated Greedy algorithms, Ant Colony Optimization methods and other similar approaches for a more general flowshop scheduling problem with additional resources.
Of interest is the fact that the regular flowshop problem, even without the consideration of additional resources, can be easily transformed into the well-known Resource Constrained Project Scheduling Problem (RCPSP). A natural question arises: How effective are existing RCPSP algorithms for these transformed scheduling problems with resources? In this paper we also test a well-known Genetic Algorithm for the RCPSP. In our opinion, in a research oriented towards solving resource constrained scheduling problems, a first step is to take a look at the performance of RCPSP approaches.

2 Scheduling with additional resources in flowshop problems. Transformation into RCPSP

In this work we study the permutation flowshop problem with the consideration of additional resources. Each job needs to be processed at each machine. This gives us a total of $n \cdot m$ tasks. Each one of these tasks requires a non-negative, known, deterministic, fixed and integer amount of units of one single resource for its processing (apart from the machine itself). When the task of a given job is about to be processed, the needed machine, as well as all necessary units of the resource, must be available. Then, the machine and the amount of resource needed are busy during the processing of the task and once completed they are freed and returned to the resources pool. The total quantity of the available resource is limited and fixed during the processing horizon. The minimization of the maximum completion time or makespan is the considered optimization criterion. According to the notation of Blazewicz [3], the studied problem is $F/prmu,res1\cdot\cdot/C_{\max}$. Figure 1 shows an example of a small flowshop with 10 jobs and four machines. Below the Gantt chart we can see the graph with the usage of the single additional resource. As we can see, some tasks must be delayed as there are not enough resources available. One side effect is that there are idle times in the first machine, something that in regular flowshops never happens.

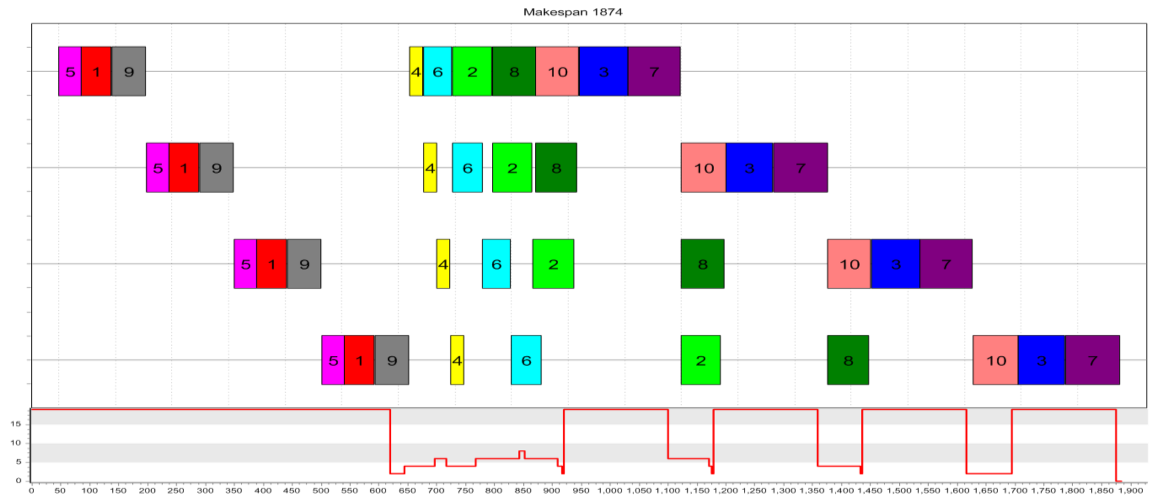

Figure 1: A simple example of a 10 job, four machine flowshop where it is also shown the utilization of single resource.

This problem can be easily transformed into a RCPSP by considering the machines as resources with a maximum availability of 1. The steps are as follows: first, each job is constituted by a sequence of m tasks and this is represented by a chain of m activities in the transformed RCPSP. The chains of activities are obtained using precedence constraints. Second, each machine i, i=1,…,m is modeled as a resource R_i with an availability of one unit and added to the original resource pool. Third, each activity in the position i, i =1,…,m of the chain only needs one unit of resource R_i, and a certain amount of real resource. It is to be noted that this transformation leads to a more general version of the problem (general flowshop instead of the permutation flowshop). Something similar has been done in [20].

3 The metaheuristic approach: algorithms and preliminary results

In order to tackle this problem we decided to modify a set of algorithms designed for the permutation flowshop without the resource constraints. The modification does not modify the basic structure of such methods since the solution representation is not altered. The only change is the process to evaluate candidate solutions as with the considered problem, resources are calculated.
The process starts considering the first task of the first job in the solution and scheduling it in the first time slot in which a sufficient amount of resources are available. A structure containing the information about the quantity of resource available at each time is updated. Then the procedure considers the second task of the first job and the first task of the second job. The one with the smallest requirement of resources is chosen and scheduled on the correspondent machine in the first available time period. The set of tasks to be evaluated is then updated considering the unscheduled tasks whose predecessors have been already processed. These steps are repeated until a complete solution is built. This extension in the calculation of schedules has been embedded into the well-known NEH heuristic of Nawaz et al. (1983) [12] into the Iterated greedy (IG) method of Ruiz and Stützle (2007) [15], the PACO algorithm of Rajendran and Ziegler (2004) [13] and into the Hybrid Genetic Algorithm (HGA) of Ruiz et al. (2006) [16]. These algorithms can be considered the state-of-the-art methods as far as the PFSP is concerned.

4 Computational evaluation and preliminary results

Two different experiments have been carried out in this preliminary study. The first is aimed at evaluating the performance of RCPSP algorithms when dealing with regular scheduling problems without additional resources. For this experiment we transformed the well-known instances of Taillard [18] into PSPLIB-like test files as explained in section 2. This set is composed of 12 groups of instances with 10 replicates at each group. The groups are formed after the following combinations of n and m: 20×5, 20×10, 20×20, 50×5, 50×10, 50×20, 100×5, 100×10, 100×20, 200×10, 200×20 and 500×20.
The second experiment is aimed at assessing the effectiveness of algorithms designed for the classical flowshop when there is one limited and renewable additional resource. Hence, for this experiment we decided to create a large set of instances whose structure has been inspired in that of Taillard's instances. The only difference is the presence of resource demands for each job. This demand is uniformly distributed in the range [0, 9] and the same value is assigned to all the tasks of the same job. The total quantity of the available resource q is used to create several instance sets of different difficulty. This value is chosen to be respectively the 60%, 80%, 100%, 120% and 140% of the average needs of resource. This value is calculated taking into account that if we have m machines, at each moment at most m jobs are being processed at the same time and, therefore, on average, the total needs of resource is m·{the *statistical expected value* of U[0,9]}. Where this expected value is, according to the uniform distribution, 4.5. Finally, for each combination of m = {5,10,20}, n ={20,50,100,200} and q, we generated 10 instances for a grand total of 600. Another set of 100 instances has been generated by choosing a random combination of the previous parameters. The aim of this second set is to be used in the calibration phase of the algorithms.
We compare the NEH, IG, PACO and HGA algorithms against the Genetic Algorithm of Alcaraz and Maroto [1] that was specifically proposed for the RCPSP. This GA uses a special solution representa-

tion with a forward/backward bit and also another bit that decides if the project schedule is built with the serial or parallel schemes. This algorithm has been demonstrated to be state-of-the-art in several studies. This algorithm was later applied with great success to the Multi-Mode Resource Constrained Project Scheduling Problem by Alcaraz et al. [2]. We denote this method by GA_Alcaraz. Each algorithm has been executed 10 times (replicates) over each instance and the results have been averaged.
The NEH, IG, PACO and HGA algorithms have been coded in Delphi 2009 while GA_Alcaraz has been coded in Microsoft Visual C++ 6.0. Both experiments have been performed on a cluster of 30 blade severs each one with two Intel XEON 5254 processors running at 2.5 GHz with 16 GB of RAM memory. Each cluster has two processors with four cores each but experiments are carried out in virtualized Windows XP machines, each one with two virtualized processor and 2 GB of RAM memory.
An important issue related to the comparison among algorithms is the choice of a suitable termination criterion. We believe that the fairest way to confront different methods is assigning the same amount of CPU time to each one of them and more time to larger instances with respect to smaller ones. Therefore, we assign to each algorithm the same elapsed CPU time limit that depends on the size of the considered instance. The algorithms are then stopped after $n{\cdot}m/2{\cdot}$ t milliseconds of CPU time, where t is an input parameter that will be tested at different values. In this way, we assign more time to larger instances that are obviously more time consuming to solve. For the first experiment, t has been set to 60ms. Since we could not modify the stopping criterion of GA_Alcaraz, we decided to use the most common stopping criterion in RCPSP i.e., the number of evaluated calendars. We set this number to 500 in order to have execution times comparable to those of the other algorithms. In Table 1, we present the results of the first experiment. The cells contain the average percentage deviation from the best lower bound known of Taillard instances [18].

		Algorithms									
		GA_Alcaraz		HGA		PACO		IG		NEH	
# Jobs	#Machines	Time	%lb	Time	%lb	Time	%lb	Time	%lb	Time	%lb
20	5	0.128	17.691	0.300	0.050	0.300	0.213	0.300	0.041	0.000	3.348
	10	0.346	24.527	0.600	0.101	0.600	0.232	0.600	0.010	0.000	5.025
	20	0.904	21.725	1.200	0.066	1.200	0.168	1.200	0.023	0.000	3.731
50	5	0.436	17.324	0.750	0.008	0.750	0.037	0.750	0.001	0.000	0.844
	10	1.085	30.648	1.500	0.692	1.500	0.835	1.500	0.506	0.001	5.123
	20	3.471	36.518	3.000	2.758	3.000	3.004	3.000	2.398	0.002	6.346
100	5	1.064	16.655	1.500	0.022	1.500	0.038	1.500	0.006	0.002	0.463
	10	3.274	27.050	3.000	0.228	3.000	0.326	3.000	0.165	0.003	2.125
	20	10.881	36.824	6.000	2.246	6.000	2.854	6.000	1.952	0.006	5.234
200	10	10.320	25.034	6.000	0.169	6.002	0.173	6.000	0.083	0.012	1.430
	20	36.483	35.546	12.000	1.602	12.002	2.099	12.000	1.359	0.025	4.530
500	20	256.932	30.807	30.000	0.744	30.024	0.907	30.000	0.565	0.144	2.238
Average values		27.110	26.696	5.488	0.724	5.490	0.907	5.488	0.592	0.016	3.370

Table 1: Average percentage deviation from best known solution for flowshop methods (HGA, PACO, IG, NEH) and the RCPSP algorithm (GA_Alcaraz). Regular flowshop instances. Times in seconds.

As can be easily seen, all algorithms, including the simple NEH heuristic, return very good results, they are between 0.001% and 6.3% of the best lower bound known. The best algorithm is the IG which obtains better results for each instance subset. This result is consistent with those of Ruiz and Stützle [15] where IG was shown to outperform the existing literature on regular flowshop and makespan criterion. The GA_Alcaraz, however, presents results that are, on average, more than 25% worse than the best solution. This is partially due to very nature of the model used by RCPSP to describe a flowshop problem that needs a big quantity of activities and precedence constraints. This model results in a general flowshop, instead of a permutation flowshop. Considering that, for the smallest instances, the number of activities is equal to 100 while for the biggest are 10.000. Addition-

ally, and as can be seen, GA_Alcaraz needs more CPU time in almost all cases, with more than 8 times more CPU time for the instances of 500×20. It is clear that using RCPSP algorithms for solving scheduling problems is not a viable approach.
We think that these preliminary observations call for an intensification of the research on the scheduling problems with limited resources from a different perspective respect to RCPSP. We believe, hence, that these problems need a scheduling approach that does not consider the machine set as a resource and makes the problem simpler and easier to be successfully solved. We are now in the process of extending this analysis by making new and more exhaustive tests considering, for example, different stopping times and other RCPSP algorithms as that of Valls et al. [19].
The results of the second experiment are summarized in Table 2. The results presented are averaged over ten replicates for each algorithm and two stopping times (*t* = 200ms. and *t* = 1000ms.) are considered. In this case, we do not have a known lower bound for the considered problem (recall that in this second test, the additional resource is being considered). Hence, we calculate the percentage deviation with respect to the best solution found in all the runs of all the algorithms. All the considered algorithms use the NEH heuristic to generate an initial solution.

			Algorithms							
			HGA		PACO		IG		NEH	
t	#Jobs	#Machines	Time	%Best	Time	%Best	Time	%Best	Time	%Best
1000ms	20	5	50.010	2.137	50.008	1.433	50.012	1.154	0.245	6.367
		10	100.029	1.767	100.026	1.227	100.034	0.957	0.518	6.457
		20	200.076	1.365	200.074	0.936	200.085	0.766	1.145	4.435
	50	5	125.126	1.969	125.103	1.141	125.101	0.932	3.889	2.913
		10	250.261	2.346	250.239	1.377	250.268	1.113	8.890	3.475
		20	500.895	2.560	500.642	1.292	500.900	1.123	21.940	3.574
	100	5	250.859	1.568	250.477	0.728	250.753	0.778	33.897	1.600
		10	501.074	2.092	501.200	1.129	500.885	1.068	84.725	2.093
		20	1006.984	1.706	1003.270	0.931	1006.758	0.833	231.692	1.707
	200	5	504.385	1.240	502.548	1.124	504.181	1.086	323.205	1.240
		10	1014.567	1.002	1007.100	0.931	1014.213	0.904	901.202	1.002
		20	2741.007	1.083	2735.782	1.083	2722.523	1.083	2723.064	1.083
Average values			603.773	1.736	602.206	1.111	602.143	0.983	361.201	2.995
200ms	20	5	10.011	3.790	10.009	2.702	10.009	2.535	0.244	6.367
		10	20.021	3.451	20.026	2.391	20.040	2.227	0.521	6.457
		20	40.054	2.763	40.069	1.922	40.099	1.742	1.138	4.435
	50	5	25.038	2.898	25.100	1.859	25.162	1.756	3.847	2.913
		10	50.206	3.474	50.246	2.291	50.173	2.120	8.812	3.475
		20	101.096	3.573	100.622	2.260	101.006	2.197	21.823	3.574
	100	5	50.855	1.600	50.431	1.454	50.821	1.415	33.784	1.600
		10	102.488	2.092	101.371	1.944	102.424	1.919	84.306	2.093
		20	235.574	1.706	231.625	1.707	231.714	1.707	231.063	1.707
	200	5	325.354	1.240	322.256	1.240	323.006	1.240	321.205	1.240
		10	907.916	1.002	900.126	1.002	900.591	1.002	897.104	1.002
		20	2738.516	1.083	2720.356	1.083	2722.404	1.083	2709.368	1.083
Average values			383.927	2.389	381.020	1.821	381.454	1.745	359.435	2.995

Table 2: Average percentage deviation from best known solution for flowshop methods. Resource constrained flowshop instances. Times in seconds.

Again, the IG performs best and returns the best results for each instance subset. However, it must be noted that the tested algorithms barely improve the NEH, in some cases no more than a 6%. This calls for additional attention and more focused work. Of particular interest is also the sheer amount of

CPU time employed. Note also how in the largest instances, the time needed by the NEH is almost the same as the time needed by all other tested methods. This means that the other methods did not have time to start the search after the initialization. It is clear then that adding an additional resource has a profound effect over the scheduling problem. For this second test we ran into problems when testing GA_Alcaraz, possibly memory problems as the resulting scheduling problem with the additional resource is even greater than the before.

5 Conclusions and future research

In this short paper we have presented an advancement about the consideration of a flowshop problem in which all tasks make use of an additional scarce resource. This resource might model, for example, expensive tooling or personnel at production plants. With this we strive to help in closing the widely recognized "*scheduling gap*" that precludes advances in scheduling research from their applications in practice. We have focused our attention on the possibility of transforming resource constrained scheduling problems into resource constrained project scheduling problems, for which there is a plenty literature and research.

Our results show that even the best resource constrained project scheduling metaheuristics cannot compete with simple adaptations of existing flowshop heuristics, yielding results far worse than simple constructive heuristics. On the other hand, we have also shown in our initial experiments that the consideration of a single additional resource also deteriorates the results obtained by the adaptations of existing flowshop heuristics. More specifically, CPU times climb exponentially, verging on unacceptable spans of time to obtain solutions.

We are currently studying and enlarging the previous experiments. Tailored metaheuristics for the resource constrained flowshop scheduling problem are being developed in order to bring CPU times down and also, at the same time, to increase the quality of the obtained solutions. At the same time, we are also studying other scheduling problems, namely the parallel machines scheduling problem. Also from the RCPSP perspective, to see if simpler scheduling settings are more amenable to the RCPSP methods.

References

[1] J. Alcaraz and C. Maroto. A Robust Genetic Algorithm for Resource Allocation in Project Scheduling. *Annals of Operations Research*, 102:83-109, 2001.

[2] J. Alcaraz, and C. Maroto and R. Ruiz. Solving the multi-mode resource-constrained project scheduling problem with genetic algorithms. *Journal of the Operational Research Society*, 54:614-626, 2003.

[3] J. Blazewicz and J.K. Lenstra and Rinnoy Kan. Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics,* 5:11-24, 1983.

[4] E. Figielska. A new heuristic for scheduling the two-stage flowshop with additional resources. *Computers & Industrial Engineering,* 54:750-763, 2008.

[5] 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.

[6] E. Figielska. Heuristic algorithms for preemptive scheduling in a two-stage hybrid flowshop with additional renewable resources at each stage. *Computers & Industrial Engineering,* 59:509-519, 2010.

[7] J. Grabowski and A. Janiak. Job-shop scheduling with resource-time models of operations. *European Journal of Operational Research*, 28:58-73, 1987

[8] F. Huq and K. Cutright and C. Martin. Employee scheduling and makespan minimization in a flow shop with multi-processor work stations: a case study. *Omega*, 32:121-129, 2004.

[9] A. Janiak. Minimization of resource consumption under a given deadline in the two-processor flow-shop scheduling problem. *Information Processing Letters,* 32:101-112, 1989.

[10] A. Janiak. Minimization of the makespan in a two-machine problem under given resource constraints. *European Journal of Operational Research,* 107:325-337, 1998.

[11] M. Y. Kovalyov and Y. M. Shafransky. Uniform machine scheduling of unit-time jobs subject to resource constraints. *Discrete Applied Mathematics,* 84:253-257, 1998.

[12] M. Nawaz and E.E. Enscore and I. Ham. A heuristic algorithm for the m-machine, n-job flow shop sequencing problem. *Omega*, 11:91–5, 1983.

[13] C. Rajendran and H. Ziegler. Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs. *European Journal of Operational Research*, 155:426-438, 2004.

[14] A. Ruiz-Torres and G. Centeno. Minimizing the number of late jobs for the permutation flowshop problem with secondary resources. *Computers & Operations Research*, 35:1227-1249, 2008.

[15] R. Ruiz and T. Stützle. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 177:2033-2049, 2007.

[16] R. Ruiz, and C. Maroto and J. Alcaraz. Two new robust genetic algorithms for the flowshop scheduling problem. *Omega*, 34:461-476, 2006.

[17] J. L. Szwarcfiter. Job shop scheduling with unit time operations under resource constraints and release dates. *Discrete Applied Mathematics*, 18:227-233, 1987.

[18] Taillard, E. Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64:278-285, 1993.

[19] V. Valls and F. Ballestín and S. Quintanilla. A hybrid genetic algorithm for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 185:495-508, 2008.

[20] S. Voß, and A. Witt. Hybrid flow shop scheduling as a multi-mode multi-project scheduling problem with batching requirements: A real-world application. *International Journal of Production Economics*, 105:445-458, 2007.