Two-agent scheduling problem with flowtime objective: Analysis of problem and exact method

Paz Perez-Gonzalez[1], Jose M. Framinan[1], Manuel Dios[1]

Industrial Management, University of Seville, Spain
pazperez@us.es, framinan@us.es, mdios@us.es

Keywords: single machine, two-agent scheduling problem, interfering jobs problem, flowtime.

1 Introduction

Interfering jobs problems, or multi-agent scheduling problems, consist on scheduling jobs from different sets, each one with its own objective, and competing for the same machines (Agnetis *et. al.* 2004). This is an emergent topic, and a recent review of the problem is presented by Perez-Gonzalez and Framinan (2013), from whom we adopt the notation. In this paper, we consider a single machine scheduling problem with two sets of jobs $\mathcal{J}^A$ and $\mathcal{J}^B$ ($\mathcal{J}^A \cap \mathcal{J}^B = \varnothing$), each one with n^A and n^B jobs respectively. Let $\mathcal{J} = \mathcal{J}^A \cup \mathcal{J}^B$ with $n = n^A + n^B$ jobs. Given a sequence σ formed by jobs in both sets, the completion time of job $i \in \mathcal{J}^A$ is denoted as $C_i^A(\sigma)$ and $C_{sum}^A(\sigma) = \sum_{i \in \mathcal{J}^A} C_i^A(\sigma)$ is the total flowtime of σ for jobs in $\mathcal{J}^A$. The total flowtime of jobs in $\mathcal{J}^B$ is $C_{sum}^B(\sigma)$. The objective considered here is to minimize C_{sum}^A subject to $C_{sum}^B \leq \epsilon$, and the problem is denoted $1||\epsilon(C_{sum}^A/C_{sum}^B)$. This problem is shown to be weakly NP-hard by Agnetis *et. al.* (2004), who present a Dynamic Programming (DP) algorithm with running time $\mathcal{O}(n^A n^B \epsilon)$. In this paper, we try to gain some understanding of this problem. In Section 2, we analyse their structure of solutions, as it is well-known that NP-hard problems can be easy to be solved by heuristic methods since there may be many solutions close to the optimal), and vice versa. In Section 3, we derive some specific properties of this problem and a more efficient codification of solutions, which is embedded in a Branch and Bound procedure that outperforms existing exact methods.

2 Analysis of the structure of solutions

The problem under consideration is weakly NP-hard, so it is not possible to find the optimal solution in polynomial time. However, there are strongly NP-hard problems for which is not "difficult" to find solutions close to the optimum due to its structure of solutions. For example, Perez-Gonzalez and Framinan (2009) study a strongly NP-hard scheduling problem which, in some cases, almost all solutions (99.6%) have an approximation percentage to the optimal value of the objective function less than 2%. So, finding a good solution by heuristic methods is "easy". Taillard (1990) and Armentano and Ronconi (1999) concludes that other scheduling problems are "harder" using the same approach. To the best of our knowledge, this kind of studies have been not carried out for interfering jobs problems. In our case, not all schedules are feasible so the analysis must be based on two aspects:

1. The percentage of feasible solutions. The hardness of the problem depends on the percentage of feasible solutions with respect to the total number of solutions. This ratio clearly depends on the value of ϵ. In our case, we compute ϵ as in Agnetis *et. al.* (2009), $\epsilon \in [\epsilon_{min}, \epsilon_{max}]$, for a given $\delta \in (0,1)$: $\epsilon = \epsilon_{min} + \delta(\epsilon_{max} - \epsilon_{min})$, with $\epsilon_{\min} = C_{sum}^B(\sigma_{SPT}^B \cup \sigma_{SPT}^A)$, and $\epsilon_{\max} = C_{sum}^B(\sigma_{SPT}^A \cup \sigma_{SPT}^B)$.

2. The distribution of solutions provides the distance to the optimal solution of each feasible solution obtained by complete enumeration for each instance. Unfeasible solutions are discarded. If a high proportion of feasible solutions are close to the optimal solution, the problem is considered to be "easy".

We have generated ten small instances (since the computational time to evaluate all sequences of each instance is high) for each problem combining all values of $n^A \in \{5, 10, 15\}$, $n^B \in \{5, 10, 15\}$ and $\delta \in \{0.2, 0.4, 0.6, 0.8\}$. Table 1 shows the percentage (average) of feasible solutions for the 10 instances solved for each problem. Moreover, the distributions of feasible solutions for all cases have the same shape (see Figure 1 as an example for instances of size 15×15). Regarding δ, Table 1 shows that as ϵ (δ) increases, the number of feasible solutions increases too. Figure 1 shows that as ϵ increases, the solutions are more distant to the optimal. Then, the difficulty of the problem suggests different approaches to tackle it depending on the value of ϵ, i.e.: a) Smaller values of ϵ ($\delta \in \{0.2, 0.4\}$) mean few feasible solutions, however, the distribution shows that feasible solutions in this case are close to the optimal, so any feasible solution can be a good solution. Then, the main focus is on finding feasible solutions. b) Bigger values of ϵ ($\delta \in \{0.6, 0.8\}$) mean a great percentage of feasible solutions, however, the distribution shows that feasible solutions in this case may be far from the optimal. Then, the main difficulty here is on finding good solutions (among the feasible solutions).

With respect to problem sizes, in Table 1 it can be seen that, on average, the problem seems more difficult when $n^A \leq n^B$: 5×10 (64.5704) vs 10×5 (68.4882); 5×15 (64.6340) vs 15×5 (68.7018); 10×15 (68.6646) vs 15×10 (68.7524). This conclusion is the opposite that the obtained by Agnetis *et. al.* (2009) for the weighted case $1||\epsilon(C^A_{wsum}/C^B_{wsum})$.

Table 1. Percentage of feasible solutions

		δ				
n^A	n^B	0.2	0.4	0.6	0.8	Aver.
5	5	19.2063	55.3968	84.2857	96.6270	63.8790
	10	14.9950	56.2970	88.3350	98.6547	64.5704
	15	13.1127	56.4880	89.8013	99.1338	64.6340
10	5	19.8968	63.4266	91.5018	99.1275	68.4882
	10	13.8711	65.6915	95.3331	99.8345	68.6825
	15	11.2345	66.8304	96.6568	99.9368	68.6646
15	5	18.2598	64.0473	92.9870	99.5130	68.7018
	10	11.5637	66.8426	96.6568	99.9464	68.7524
	15	8.6749	68.3683	98.0317	99.9868	68.7654

3 Codification of Solutions and Branch and Bound procedure

The classical encoding scheme used to represent a sequence for one-machine scheduling problems is the *permutation codification*, where each job $j \in \mathcal{J}$ is represented by a number, $j = 1, \ldots, n$. However, our problem has some properties that allow developing a more efficient codification. More specifically, Agnetis *et. al.* (2004) show that, in an optimal schedule, jobs in $\mathcal{J}^A$ and jobs in $\mathcal{J}^B$ follow the shortest processing time first rule (SPT). Schedules verifying this property are called *SPT schedules*. Note that the SPT rule does not apply for jobs belonging to different sets. Without loss of generality, we will assume that processing times of jobs in $\mathcal{J}^A$ and $\mathcal{J}^B$ are given in SPT order respectively. Based on

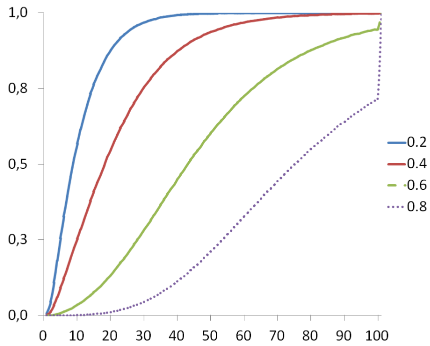

Fig. 1. Distribution of feasible solutions for $n^A \times n^B = 15 \times 15$ for different δ

this property, we can define a new encoding scheme called *binary codification*, where jobs in $\mathcal{J}^A$ are coded by zeros, and jobs in $\mathcal{J}^B$ by ones. Any schedule formed by zeros and ones represents only one SPT schedule. The first zero in the schedule is the job in $\mathcal{J}^A$ with the smallest processing time, the second is the second one in $\mathcal{J}^A$ with smallest processing time, and so on. Note that this codification reduces the search space from $n!$ possible schedules to only $\frac{(n^A+n^B)!}{n^A!n^B!}$ of SPT schedules.

Taking into account this binary codification as well as some properties of the problem that are omitted due to the lack of space, we develop a Branch and Bound (B&B) algorithm (pseudo-code in Figure 2), where UB is the upper bound used in the method, computed by the total flowtime of $\mathcal{J}^A$ for a given sequence provided by the fastest method presented by (Perez-Gonzalez and Framinan 2012). It is compared to the DP algorithm by Agnetis *et. al.* (2004) and to a MILP model of the problem using the Gurobi solver. Ten instances of sizes $n^A \in \{5, 10, 15, 20\}$ and $n^B \in \{5, 10, 15, 20\}$ are generated with random $[1, 99]$ processing times. δ has been randomly generated in the interval $[0.4, 0.6]$. Table 2 shows the average CPU time for each problem size. Note that Gurobi and DP are slower (not being able to find the optimal solution in less than 24 hours per instance for the largest size), while the B&B is faster for all instances except for the smallest sizes (5×5 and 5×10).

References

Agnetis, A. and De Pascale, G. and Pacciarelli, D. A lagrangian approach to single-machine scheduling problems with two competing agents *Journal of Scheduling*, 51(4):401-415, 2009.

Agnetis, A. and Mirchandani, P. B. and Pacciarelli, D. and Pacifici, A. Scheduling problems with two competing agents *Operations Research*, 52(2):229-242, 2004.

Armentano, V. A. and Ronconi, D. P. Tabu search for total tardiness minimization in flowshop scheduling problems *Computers & Operations Research*, 26(3):219-235, 1999.

P. Perez-Gonzalez and J. M. Framinan. Scheduling permutation flowshops with initial availability constraint: Analysis of solutions and constructive heuristics. *Computers & Operations Research*, 36(10):2866-2876, 2009.

P. Perez-Gonzalez and J. M. Framinan. A common framework and taxonomy for multicriteria scheduling problems with interfering and competing jobs: Multi-agent scheduling problems *European Journal of Operational Research*, 235(1):1-16, 2014.

P. Perez-Gonzalez and J. M. Framinan. Approximate algorithms for one-machine scheduling with interfering jobs *XXV European Conference On Operational Research*, Vilnius, Lituania, 2012.

Taillard, E. D. Some efficient heuristic methods for the flow shop sequencing problem *European Journal of Operational Research*, 47(1):65-74, 1990.