MISTA 2017

On the exact solution of a large class of parallel machine scheduling problems

Teobaldo Bulhões · Ruslan Sadykov · Eduardo Uchoa · Anand Subramanian

1 Introduction

We consider the problem of scheduling a set of jobs J ($n = |J|$) on a set of machines of different types $k \in M$ ($m = |M|$) without preemption. A job j is not allowed to be processed before its release date r_j, and its processing time on a machine of type $k \in M$ is denoted as p_j^k. Also, s_{ij}^k denotes the setup time required to process job j immediately after job i on a machine of type k. Each job j is associated to a cost function $f_j(C_j)$ defined over its completion time C_j. The objective function is to minimize $\sum_{j \in J} f_j(C_j)$. This function is very general and can model many criteria. For example, suppose each job has an earliness $E_j = \max\{d_j - C_j, 0\}$ and a tardiness $T_j = \max\{C_j - d_j, 0\}$ that is computed based on its due date d_j. A classical objective is to minimize the total weighted earliness and tardiness given by $\sum_{j \in J}(w'_j E_j + w_j T_j)$, where w'_j and w_j are penalty coefficients associated with job j. Remark that cost functions that include earliness penalties are not regular and may have optimal solutions that include idle times between jobs. In this work we present a novel exact algorithm that is capable of solving problem $R|r_j, s_{ij}^k| \sum f_j(C_j)$ and the large class of problems that can be derived as particular cases from it. The proposed algorithm consists of a branch-cut-and-price approach that combines several features such as non-robust cuts, strong branching, reduced cost fixing and dual stabilization. To our knowledge, this is the first exact algorithm for unrelated machines with earliness and/or tardiness that can solve consistently instances with more than 20 jobs. We report improved bounds for instances of problems $R|r_j, s_{ij}^k| \sum w'_j E_j + w_j T_j$ and $R|| \sum w'_j E_j + w_j T_j$ with up to 80 and 120 jobs, respectively.

Teobaldo Bulhões, Eduardo Uchoa
Universidade Federal Fluminense, Niterói, Brazil
E-mail: tbulhoes@ic.uff.br, uchoa@producao.uff.br

Ruslan Sadykov
Inria Bordeaux - Sud Ouest, France
E-mail: Ruslan.Sadykov@inria.fr

Anand Subramanian
Departamento de Sistemas de Computação, UFPB, João Pessoa, Brazil
E-mail: anand@ci.ufpb.br

2 Mathematical formulation

We start by defining some notation required to introduce the mathematical formulation. Let T be an upper bound on the maximum completion time of a job in some optimal solution. Let $N_k = (V_k = R_k \cup O_k, A_k = A_k^1 \cup A_k^2 \cup A_k^3 \cup A_k^4)$ be the acyclic graph associated with each machine type $k \in M$, where set $R_k = \{(j,t,k) : j \in J, t = r_j + p_j, \ldots, T\}$ contains the vertices associated with the jobs. We adopt the convention that idle times only occur after the job has been processed. We assume that job j has been processed when arriving at vertex (j,t,k), but note that j did not necessarily finished at time t because of the existence of idle times. Set $O_k = \{(0,t,k) : t = 0, \ldots, T\}$ contains the vertices associated with a dummy job 0. For brevity, we denote arc $((i,t,k),(j,t+s_{ij}^k+p_j^k,k))$ as (i,j,t,k). Set $A_k^1 = \{(i,j,t,k) = ((i,t,k),(j,t+s_{ij}^k+p_j^k,k)) : (i,t,k) \in R_k, (j,t+s_{ij}^k+p_j^k,k) \in R_k, j \in J \setminus \{i\}\}$ contains the arcs connecting the vertices of R_k. Set $A_k^2 = \{(0,j,t,k) = ((0,t,k),(j,t+s_{0j}^k+p_j^k,k)) : (j,t+s_{0j}^k+p_j^k,k) \in R_k, j \in J\}$ contains all arcs connecting the vertices from O_k to R_k. Set $A_k^3 = \{(j,0,t,k) = ((j,t,k),(0,T,k)) : (j,t,k) \in R_k\}$ contains all arcs connecting the vertices from R_k to O_k. Set $A_k^4 = \{(j,j,t,k) = ((j,t,k),(j,t+1,k)) : (j,t,k) \in R_k \cup O_k, (j,t+1,k) \in R_k \cup O_k\}$ contains the arcs associated with idle times. Let f_a be the cost of an arc $a = (i,j,t,k) \in A_k^1 \cup A_k^2$, which is the cost incurred if job j finishes to be processed at time $t + s_{ij}^k + p_j^k$. Note that $f_a = 0, \forall a \in A_k^3 \cup A_k^4$. Moreover, let set $R_k^j = \{(i,t,k) \in R_k : i = j\}$ denote the vertices associated with job j. Finally, for each subset $S \subseteq V_k$, let $\delta^-(S)$ and $\delta^+(S)$ be the sets representing the arcs entering and leaving S, respectively. The proposed arc-time-indexed formulation is as follows.

$$\text{(F1)} \quad \min \sum_{k \in M} \sum_{a \in A_k} f_a x_a \tag{1}$$

$$\text{s.t.} \quad \sum_{k \in M} \sum_{a \in \delta^-(R_k^j)} x_a = 1, \qquad \forall j \in J \tag{2}$$

$$\sum_{a \in A_k^2} x_a \leq m_k, \qquad \forall k \in M \tag{3}$$

$$\sum_{a \in \delta^-(\{v\})} x_a - \sum_{a \in \delta^+(\{v\})} x_a = 0, \qquad \forall k \in M, \forall v \in V_k \setminus \{(0,0,k),(0,T,k)\} \tag{4}$$

$$x \geq 0, x \text{ integer} \tag{5}$$

Objective function (1) minimizes the completion time dependent costs. Constraints (2) state that each job $j \in J$ must be processed exactly once. Constraints (3) impose that at most m_k machines of type $k \in M$ can be used. Constraints (4) are related to the flow conservation. Define P_k as the set of paths in graph N_k that start at vertex $(0,0,k)$ and end at vertex $(0,T,k)$. Let b_p^a be the number of times path p traverses $a \in A_k$ and let λ_p be a binary variable that assumes value 1 if p is in the solution. Formulation F1 can be rewritten in terms of variables λ_p by means of a Dantzig-Wolfe decomposition as follows.

$$\text{(DW-F1)} \quad \min \sum_{k \in M} \sum_{p \in P_k} \Big(\sum_{a \in A_k} b_a^p f_a \Big) \lambda_p \tag{6}$$

$$\text{s.t.} \quad \sum_{k \in M} \sum_{p \in P_k} \Big(\sum_{a \in \delta^-(R_k^j)} b_a^p \Big) \lambda_p = 1, \qquad \forall j \in J \tag{7}$$

$$\sum_{p \in P_k} \Big(\sum_{a \in A_k^2} b_a^p \Big) \lambda_p \leq m_k, \qquad \forall k \in M \tag{8}$$

$$\lambda \text{ binary} \tag{9}$$

3 Branch-cut-and-price algorithm

This section briefly describes the proposed branch-cut-and-price (BCP) algorithm over formulation DW-F1. The solution of every node has the following three steps. **(i)** A lower bound for the problem is obtained by solving the linear relaxation of DW-F1 (possibly with cuts and branching constraints) through column generation. The pricing subproblem for a machine type $k \in M$ corresponds to finding a least-cost path in graph N_k where arc costs are associated with reduced costs. In this step, the pricing subproblem is solved by a labeling algorithm and a dual stabilization technique is applied. **(ii)** A reduced cost fixing procedure is executed for every machine type $k \in M$ to remove from graph N_k the arcs that can not be in a solution that improve the current upper bound. **(iii)** Cuts are separated and added to the master problem. Steps **(i)**, **(ii)** and **(iii)** are repeated as long as a tailing off condition is not reached and the pricing time is smaller than a predefined threshold, after which a strong branching technique is applied. The cuts adopted are obtained by a Chvátal-Gomory rounding of the n constraints (7). Such cuts are often referred to as non-robust, since each single cut requires additional states in the dynamic programming algorithm used for solving the pricing subproblem. Therefore, those cuts should be added in a limited fashion to avoid combinatorial explosion. To mitigate the impact of the Rank-1 cuts on the pricing subproblem, we have adopted the limited arc memory technique described in [4].

4 Computational results

Table 1 reports preliminary results obtained by the proposed BCP over the $R|r_j, s_{ij}^k| \sum w'_j E_j + w_j T_j$ instances of [3] and the $R|| \sum w'_j E_j + w_j T_j$ instances of [1]. The former instances were derived from the instances of [1] by adding two types of sequence-dependent setup times: small and large. All experiments were conducted on a Intel Xeon E5-2680 v3, running at 2.5 GHz with a single thread and a time limit of 12 hours. In this table, **Improv (%)** denotes the average percentage improvement over the best known solution (BKS) which were obtained by running the methods devised in [1,2], and **#New** corresponds to the number of new improved solutions. Regarding the $R|r_j, s_{ij}^k| \sum w'_j E_j + w_j T_j$ instances, it can be observed that most of them were solved to optimality and several new improved solutions were found. Also, it appears that the instances with large setup times are harder to be solved. More precisely, we can verify that the quality of the lower bound obtained by BCP as well as the upper bound found by running the heuristic proposed in [2] are worse for such instances. On the other hand, only 8 $R|| \sum w'_j E_j + w_j T_j$ instances considered were not solved to optimality. In contrast, 497 such instances were not solved by the method presented

in [1]. Moreover, the use of Rank-1 cuts improved significantly the performance of our method, resulting in much smaller BCP trees and CPU times.

Table 1 Average results for $R|r_j, s_{ij}^k|\sum w'_j E_j + w_j T_j$ and $R||\sum w'_j E_j + w_j T_j$

Instances			BCP				BKS		[1]		
n	m	Setups	#Solved	Gap (%)	Time (s)	#Nodes	Improv. (%)	#New	#Solved	Gap (%)	Time (s)
40	2	small	60/60	0.00	224	1.10	0.120	22	-	-	-
60	2	small	60/60	0.00	1695	3.53	0.330	42	-	-	-
60	3	small	60/60	0.00	2109	10.6	0.408	47	-	-	-
80	2	small	60/60	0.00	5832	5.70	0.136	41	-	-	-
80	4	small	48/60	0.52	16368	91.93	0.264	50	-	-	-
40	2	large	60/60	0.00	931	2.80	0.760	46	-	-	-
60	2	large	58/60	0.06	10589	23.16	1.340	58	-	-	-
60	3	large	45/60	1.21	20691	85.76	1.560	55	-	-	-
80	2	large	28/60	1.32	35368	48.76	0.798	54	-	-	-
80	4	large	10/60	3.91	39479	120.43	0.385	27	-	-	-
40	2	no	60/60	0.00	312	3.37	0.000	0	26/60	0.16	52
60	2	no	60/60	0.00	708	3.27	0.001	1	7/60	0.89	109
60	3	no	60/60	0.00	428	2.93	0.010	5	7/60	0.82	120
80	2	no	59/60	0.00	2425	5.36	0.003	3	2/60	0.90	134
80	4	no	60/60	0.00	964	3.86	0.063	15	0/60	4.54	228
90	3	no	60/60	0.00	2018	4.70	0.033	20	1/60	2.52	153
100	5	no	59/60	0.02	3397	26.73	0.103	27	0/60	8.83	297
120	3	no	56/60	0.04	10775	16.72	0.072	22	0/60	4.12	165
120	4	no	58/60	0.007	7944	17.66	0.171	31	0/60	6.98	217

As for future work, we intend to test our algorithm on other particular cases such as the single and parallel machine problems studied in [6] and [5], respectively.

References

1. H. Şen and K. Bülbül, A Strong Preemptive Relaxation for Weighted Tardiness and Earliness/Tardiness Problems on Unrelated Parallel Machines, *INFORMS Journal on Computing.* **27** (2015) 135–150.
2. A. Kramer and A. Subramanian, A unified heuristic and an annotated bibliography for a large class of earliness-tardiness scheduling problems, *Journal of Scheduling.* (2017) accepted.
3. A. Kramer, Um método heurístico para a resolução de uma classe de problemas de sequenciamento da produção envolvendo penalidades por antecipação e atraso, *Master's Thesis, Universidade Federal da Paraíba, Brazil* (2015). In Portuguese.
4. D. Pecin, C. Contardo, G. Desaulniers, E. Uchoa, New enhancements for the exact solution of the vehicle routing problem with time windows, *INFORMS Journal on Computing.* **29** (2017) 489–502.
5. A. Pessoa, E. Uchoa, M. de Aragão, R. Rodrigues, Exact algorithm over an arc-time-indexed formulation for parallel machine scheduling problems. *Mathematical Programming Computation.* **2** (2010) 259–290.
6. S. Tanaka, M. Araki, An exact algorithm for the single-machine total weighted tardiness problem with sequence-dependent setup times. *Computers & Operations Research.* **40** (2013) 344–352.