Machine scheduling by column-and-row generation on the time-indexed formulation

Ruslan Sadykov (Speaker) [*] François Vanderbeck [†]

1 Introduction

We consider the minimum cost scheduling of jobs $j \in J = \{1, \dots, n\}$ with processing times $p_j \in I\!N$, on a single machine, a single job at a time, with no preemption. Let $T \geq \sum_j p_j$ be the length of the planning horizon. Period t represents time interval $[t-1, t)$ for $t = 1, \dots T$. We assume a generic cost function: the inputs allow us to compute values c_{jt} representing the cost of processing job j starts at the ouset of period t.

One of the approaches to solve this problem uses the following time-indexed Integer Programming formulation. Let a binary variable z_{jt}, $j \in J$, $t = 1, \dots, T$, equals to one if job j starts at the outset of period t. Let also job 0 with processing time 1 model the machine idle time. The computationally most efficient such time-indexed formulation is the so-called "flow" reformulation [1]:

$$[R] \equiv \min \Big\{ \sum_{jt} c_{jt}\, z_{jt} : \sum_{t=1}^{T-p_j+1} z_{jt} = 1 \ \forall j \in J, \quad \sum_{j=0}^{n} z_{j1} = 1, \\ \sum_{j=0}^{n} (z_{jt} - z_{j,t-p_j}) = 0 \ \forall t \in \{2, \dots, T\}, \quad z_{jt} \in \{0,1\} \ \forall j, t \Big\}$$

where the first group of constraints models the assignment of each job to a time period, while the others enforce the "one-job-at-a-time" restriction. The formulation has nT variables and $(n+T)$ constraints (note that T is pseudo-polynomial in the input size).

The linear programming (LP) relaxation of this formulation is known to produce very tight lower bounds. However, its size becomes impractical for instances with a large time horizon. One of the methods to overcome this difficulty is to apply the column generation approach based on the totally uni-modular subsystem formed by the one-job-at-a-time constraints, as done by [5]. This method consists in defining a reformulation:

$$[M] \equiv \min \Big\{ \sum_{g \in G} c^g \,\lambda_g : \sum_{g \in G} \sum_{t=1}^{T-p_j+1} z^g_{jt}\, \lambda_g = 1 \ \forall j \in J, \ \sum_{g \in G} \lambda_g = 1, \ \lambda_g \in \{0,1\} \ \forall g \in G \Big\}$$

where G is the set of "pseudo-schedules" (in which each job does not necessarily appears exactly once), vector z^g, scalar c^g define the associated solution and cost for a solution

[*]`Ruslan Sadykov@inria.fr`. INRIA Bordeaux — Sud-Ouest, 351 cours de la Libération, 33405 Talence, France

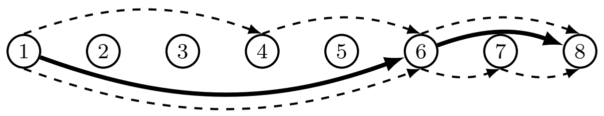
[†]`fv@math.u-bordeaux1.fr`. Université Bordeaux I, 351 cours de la Libération, 33405 Talence, France

$g \in G$. The LP relaxation of [M] is solved by column generation. The pricing subproblem can be modeled as the search for a shortest path: $z^* = \arg\min\{\sum_{jt}(c_{jt} - \pi_j)\, z_{jt} : \sum_{j=0}^{n} z_{j1} = 1,\ \sum_{j=0}^{n}(z_{jt} - z_{j,t-p_j}) = 0\ \forall t > 1,\quad z_{jt} \in \{0,1\}\ \forall j,t\}$, where π_j is a dual solution to the linear relaxation of [M]. Thus, each pseudo-schedule defines a path in a graph whose nodes represent periods and where a job j is represented by arcs $(t, t+p_j)$, and idle times by arcs $(t, t+1)$.

2 Column-and-row generation approach

An alternative approach is a column-and-row generation for the LP relaxation of [R]. The method is reviewed in [4]. Variables z are generated dynamically, not one at the time, but by lots. To do it, we solve the above pricing subproblem (where π is the dual solution of the assignment constraints of [R]), and add to [R] the components of its solution z^* with a negative reduced cost in the LP relaxation of [R] along with the flow conservation constraints that are binding for that solution. The components of z^* with a non-negative reduced cost are stored in the column pool and added to [R] on one of the subsequent iterations if their reduced cost becomes negative. In [4], we indeed showed that either the current LP value of [R] is optimal, or some components of z^* must have a negative reduced cost in the LP relaxation of [R]. Therefore, this column-and-row generation approach solves the LP relaxation of [R] after a finite number of iterations.

Compare to a standard column generation approach for [M], the interest of this alternative approach is to allow for the recombination of previously generated pricing problem solutions, and thus to accelerate the convergence. To illustrate what is meant by recombination, we picture below two pseudo-schedules (dashed) and a new pricing problem solution z^* (bold) that can be obtained by recombining these two without the need to explictly generate it through pricing.



Note that such recombination is not feasible in [M] where the only feasible solutions are those defined by the convex combinations of previously generated columns. Such column-and-row generation approach applies to any problem admitting a decomposition in which the subproblem is solved by the shortest path problem or, more generally, by the min-cost flow problem or by dynamic programming.

3 Computational results

We performed a computational comparison of three approaches on the same computer: solving the LP relaxation of [R] directly using *Cplex 12.1*; solving the LP relaxation of [M] by standard column generation; and solving the LP relaxation of [R] by column-and-row generation. Column[-and-row] generation algorithms were implemented using BaPCod — a generic Branch-and-Price code developed by the INRIA RealOpt team in Bordeaux. A problem-specific implementation is likely to produce better results.

The three approaches were tested on instances with 25, 50, and 100 jobs and the total weighted tardiness objective function. The test instances were generated using the procedure from [3] which is the most used in the literature. The objective is to minimize the total weighted tardiness. Processing times of jobs are uniformly distributed in interval $[1, 100]$. For each n, we generated 25 instances, each for different pairs of two parameters, varying the relative range of due dates and the average tardiness factor.

The results are presented in Table 1. "*cpu*" is the solution time (in seconds), "*it*" is the number of iterations in the column[-and-row] generation procedure, "*sp*" is the number of calls to the pricing subproblem solver, and "*%z*" is the percentage of z variables generated in the column-and-row generation approach (from the total number of z variables in [R]). The column-and-row generation approach outperforms the other two. Moreover, its advantage increases with the increase of n.

	Cplex for [R]	Column generation for [M]			Column-and-row generation for [R]			
n	*cpu*	*it*	*sp*	*cpu*	*it*	*sp*	*%z*	*cpu*
25	11.2	343	343	2.1	208	69	5.8%	1.5
50	153.0	1270	1270	39.4	339	106	4.5%	16.9
100	2233.0	8784	8784	2891.5	466	139	4.5%	169.1

Table 1: Computational results

4 Perspectives

Our further research agenda is (*i*) to combine the column-and-row generation with an enumeration algorithm to solve the scheduling problem to optimality; (*ii*) to check whether the combination of the column-and-row generation approach with a cutting plane method is computationally advantageous; and (*iii*) to speed-up the column-and-row generation using standard stabilization techniques for column generation and variable fixing based on reduced cost (as in [2]). We also plan to experiment this column-and-row generation approach on the arc-time indexed formulation in which each binary variable z_{ijt} determines whether job i immediately precedes job j at time moment t. LP relaxation of this formulation generates even better lower bounds [2].

References

[1] Yunpeng Pan and Leyuan Shi. On the equivalence of the max-min transportation lower bound and the time-indexed lower bound for single-machine scheduling problems. *Mathematical Programming*, 110:543–559, 2007.

[2] Artur Pessoa, Eduardo Uchoa, Marcus Poggi de Aragão, and Rosiane Rodrigues. Exact algorithm over an arc-time-indexed formulation for parallel machine scheduling problems. *Mathematical Programming Computation*, 2:259–290, 2010.

[3] Chris N. Potts and Luk N. Van Wassenhove. A Branch and Bound Algorithm for the Total Weighted Tardiness Problem. *Operations Research*, 33(2):363–377, 1985.

[4] Ruslan Sadykov and François Vanderbeck. Column generation for extended formulations. Working paper, http://hal.inria.fr/inria-00539870/en/, 2011.

[5] J.M. van den Akker, C.A.J. Hurkens, and M.W.P. Savelsbergh. Time-indexed formulations for machine scheduling problems: Column generation. *INFORMS Journal on Computing*, 12(2):111–124, 2000.