A deconstruction-reconstruction metaheuristic for a job scheduling problem

Simon Thevenin[1] and Nicolas Zufferey[1]

Geneva School of Economics and Management, University of Geneva, Switzerland
simon.thevenin@unige.ch n.zufferey@unige.ch

1 Introduction and presentation of problem (P)

In the considered job scheduling problem (P), the production environment consists in a set of parallel and identical machines. Given a set J of n jobs, a subset $J' \subseteq J$ must be selected and scheduled before a global deadline D. The non selected jobs are rejected. With each job j is associated an integer processing time p_j and a gain g_j (incurred if j is performed). Preemptions are allowed at integer points in time. Some pairs of jobs are incompatible, i.e. it should be avoided to perform them at common time slots. A *conflict* occurs if two incompatible jobs are processed during a common time slot (there can be more than one conflict between two jobs). The problem is to find a solution s where each performed job j is given p_j time slots, and such that the number of conflicts $C(s)$ does not exceed a given upper bound K. Two objectives f_1 (to be maximized) and f_2 (to be minimized) are considered in a lexicographical order (i.e. f_1 is infinitely more important than f_2): $f_1(s)$ is the sum of the gains of completely performed jobs, and $f_2(s)$ is the number of parallel machines used in s.
(P) has applications in fast moving consumer good companies. Scheduling with *rejections* is particularly relevant in make-to-order production environments [4]. *Preemptions* are used in practical situations where setup times are negligible (e.g. in automated production). *Incompatibilities* occur when scarce resources are involved in the production system [1]. More precisely, two jobs which necessitate a common scarce resource cannot be performed simultaneously (they are incompatible). However, we assume in (P) that some additional resources can be mobilized up to a certain budget, and thus up to K conflicts are allowed. Papers on scheduling with incompatibilities include [2, 3, 5] and are often related to the graph multi-coloring problem. In particular, (P) is a generalization of the well-known and NP-hard k-coloring problem (if $K = 0$, $D = k$, and $p_j = 1$ for each j).

2 Solution methods for (P)

Five approaches are proposed: GR (a greedy algorithm), DLS (a descent local search), TS (a tabu search), TS^R (a tabu search with restarts), and DRM (a deconstruction-reconstruction metaheuristic). The time limit of each algorithm is $T = 60 \cdot n$ seconds. Note that if an algorithm stops before T, it is restarted, and the best solution is returned to the user.
GR starts from an empty solution and selects the next job to schedule with the largest gain g_j (ties are broken randomly). A_j denotes the set of feasible time slots for job j (i.e. not used by any job incompatible with j). If $p_j - |A_j| > K - C(s)$, job j is rejected. Otherwise, p_j slots are sequentially assigned to j and two situations can occur at each step: (1) if $p_j - |A_j| < 0$, the slot minimizing f_2 is chosen; (2) if $p_j - |A_j| \leq K - C(s)$, the slot minimizing the number of additional conflicts is selected (but j is rejected if more than K conflicts are created).
In DLS, a *move* (to generate a *neighbor* solution from the *current* solution) consists in rescheduling a job j. The way to reassign p_j slots to j depends on A_j. If $A_j \geq p_j$, p_j slots are sequentially chosen in A_j while minimizing f_2. Otherwise, the p_j slots are given one by one, by assigning at each step the slot minimizing the number of additional conflicts. Then, to maintain feasibility, some conflicts are removed with the following *Repair* method: while $C(s) > K$, the job involved in the largest number of conflicts is rejected (break ties with the gains). In TS, when a job j is rescheduled, it cannot be rescheduled for $tab = 10$ iterations. In TS^R, TS is restarted every $I = 100$ iterations.
DRM [6] is a population based meta-heuristic relying on powerful local search techniques, where at each *generation*, a solution of the population is first deconstructed, then reconstructed, and finally improved. To tackle (P), a population Pop with 10 solutions is used. It is initialized by generating 10 random solutions as follows. First, all the jobs of J are scheduled randomly, then, feasibility is reestablished with *Repair*, and finally the solution is improved with TS during $I = 100$ iterations.
DRM uses a deconstruction parameter q which is initially set to $q_{min} = n/20$ and cannot exceed

$q_{max} = n/3$. Then, while T is not reached, the below seven steps are performed, where s^b and s^w respectively denotes the best and worst solution of *Pop*.

(1) Select the least frequently chosen solution s in the population *Pop*.
(2) *Deconstruction*: reject q jobs in s, chosen randomly.
(3) *Reconstruction*: schedule some jobs (chosen randomly) until $C(s) = q + K$. The slots are assigned one by one to each job, while minimizing the number of conflicts (break ties with f_2). If ties occur again, they are broken with information from *Pop*: the slot t maximizing $\sum_{i \in J_t} Sim(i, j)$ is chosen, where J_t is the set of jobs processed during slot t, and $Sim(i, j)$ is the number of slots where jobs i and j are performed simultaneously in the solutions of *Pop*.
(4) *Reestablish feasibility:* while s has more than K conflicts, reject the job j with the smallest ratio $g_j/C_j(s)$, where $C_j(s)$ is the number of conflicts associated with job j in s.
(5) *Local search:* apply TS during I iterations, and denote s' the resulting solution.
(6) *Update Pop:* if s' is better than s^w, replace s^w with s' in *Pop*.
(7) *Update q:* if s' is better than s^b, set $q = q_{min}$; otherwise set $q = 1.05 \cdot q$ (if allowed).

On the one hand, DRM uses elements of *strategic oscillation* methods (see steps (2) and (3)): it explores unfeasible solutions but the distance from the feasibility border is controlled, as $K + q$ conflicts are allowed. On the other hand, DRM has features from *variable neighborhood search* (see steps (2) and (7)): it generates a deconstructed solution at a certain distance q from s, and q is updated according to the improvement or not of the best encountered solution.

3 Experiments

An instance (n, τ) is defined by its number n of jobs and its rate τ of allowed conflicts, from which we deduce $K = \tau \cdot n$. 15 instances were generated, with $n \in \{50, 100, 200\}$ and $\tau \in \{0, 0.02, 0.04, 0.1, 0.2\}$. Two jobs are incompatible with probability 0.5. Each p_j is randomly chosen in interval $[1, 10]$. The gain g_j is related to p_j as follows: a random number β is first chosen in interval $[1, 20]$, and we set $g_j = \beta \cdot p_j$. Finally, the deadline D was set small enough to prevent the scheduling of all jobs. The algorithms were implemented in C++ and executed on a computer with a processor Intel Quad-core i7 2.93 GHz with 8 GB of DDR3 RAM memory. 10 runs per instance were performed with $T = 60 \cdot n$ seconds. Aggregated results are given in Table 1, which shows for each method the average percentage gap according to the best ever found value for each objective (f_1, f_2). TS outperforms GR, which is slightly better than DLS: the obtained f_1 gaps are respectively 5.46%, 8.68% and 9.32%. The deconstruction and reconstruction steps in DRM are efficient, as the DRM gap is 2.29% for f_1 versus 6.87% for TS^R. It was observed that DRM obtained the best results for 13 instances. Note that the smaller is the f_1 gap, the larger is the f_2 gap, as f_1 and f_2 are conflicting objectives.

GR	DLS	TS	TS^R	DRM
(8.68, 5.01)	(9.32, 5.14)	(5.46, 9.65)	(6.87, 8.04)	(2.29, 14.89)

Table 1. Aggregated results obtained by the proposed methods.

References

1. C. Almeder and B. Almada-Lobo. Synchronisation of scarce resources for a parallel machine lotsizing problem. *International Journal of Production Research*, 49(24):7315–7335, 2011.
2. I. Blöchliger and N. Zufferey. Multi-coloring and job-scheduling with assignment and incompatibility costs. *Annals of Operations Research*, 211(1):83–101, 2013.
3. G. Even, M. M. Halldórsson, L. Kaplan, and D. Ron. Scheduling with conflicts: online and offline algorithms. *Journal of Scheduling*, 12(2):199–224, 2009.
4. S. A. Slotnick. Order acceptance and scheduling: A taxonomy and review. *European Journal of Operational Research*, 212(1):1 – 11, July 2011.
5. S. Thevenin, N. Zufferey, and J.-Y. Potvin. Multi-objective parallel machine scheduling with incompatible jobs. In *Proceedings for the ROADEF 2014 Conference*, Bordeaux, France, February 26-28 2014.
6. N. Zufferey and F. Glover. A reconstructing evolutionary metaheuristic for the vertex coloring problem. In *Proceedings of the 23rd Benelux Conference on Artificial Intelligence, BNAIC 2011*, pages 352 – 357, Gent, Belgium, November 3 – 4 2011.