A Branch-and-Bound Procedure to Minimize Total Tardiness on One Machine with Arbitrary Release Dates

In this paper authors present a Branch-and-Bound procedure to minimize total tardiness on one machine with arbitrary release dates. They consider the scheduling situation where **n** jobs $J_1,...,J_n$ have to be processed by a single machine and where the objective is to minimize total tardiness. Associated with each job J_i, are a processing time p_i, a due date d_i, and a release date r_i. A job cannot start before its release date, preemption is not allowed, and only one job at a time can be scheduled on the machine. The tardiness of a job J_i is defined as $T_i = \max(0, C_i - d_i)$, where C_i is the completion time of J_i. The problem is to find a feasible schedule with minimum total tardiness $\sum T_i$. The problem, denoted as $1|r_i| \sum T_i$, is known to be NP-hard in the strong sense.

A lot of research has been carried on the problem with equal release dates $1|| \sum T_i$. Powerful dominance rules have been introduced in Emmons's research "One-machine sequencing to minimize certain functions of job tardiness". Lawler in his work "A pseudo-polynomial algorithm for sequencing jobs to minimize total tardiness," has proposed a dynamic programming algorithm that solves the problem in pseudo-polynomial time. Later, this algorithm has been extended for scheduling serial batching machines by Baptiste and Jouglet. And, finally, Du and Leung have shown that the problem is NP-Hard. Most of the exact methods for solving $1|| \sum T_i$ strongly rely on Emmons' dominance rules. The best results have been obtained by Szwarc, Della Croce and Grosso with a Branch-and-Bound method that efficiently handles instances with up to 500 jobs.

They rely on the edge-finding branching scheme. Rather than searching for the starting times of jobs, authors look for a sequence of jobs. This sequence is built both from the beginning and from the end of the schedule. Throughout the search

tree, they dynamically maintain several sets of jobs that represent the current state of the schedule

Branch-and-Bound procedure is very easy to implement on top of a constraint-based scheduling system such as **Ilog Scheduler**. Unfortunately, it does not perform well since no specific technique has been used in the above formulation to solve the total tardiness problem.

A dominance rule is a constraint that can be added to the initial problem without changing the value of the optimum, i.e., there is at least one optimal solution of the problem for which the dominance holds. Dominance rules can be of prime interest since they can be used to reduce the search space. However they have to be used with care since the optimum can be missed if conflicting dominance rules are combined.

Later authors present a generalization of Emmons dominance rules that are valid either if preemption is allowed or if jobs have identical processing times. They also propose some dominance rules relying on the sequence P and introduce an Intelligent Backtracking scheme. These rules allow us to deduce some precedence relations between jobs.

Emmons Rule 1. $\forall_i, k(i \neq k)$, *if* $p_i \leq p_k$ *and* $d_i \leq max(\sum Jj \in B_k p_j + p_k, d_k)$ *then* J_i *precedes* J_k ($Ji \in B_k$ *and* $J_k \in A_i$).

Emmons Rule 2. $\forall_i, k(i \neq k)$, *if* $p_i \leq p_k$ *and* $d_i > max(\sum Jj \in B_k p_j + p_k, d_k)$ *and* $d_i + p_i \geq J_j \in / A_k \, p_j$, *then* J_k *precedes* J_i.

Emmons Rule 3. $\forall_i, k(i \neq k)$, *if* $p_i \leq p_k$ *and* $d_k \geq \sum J_j \in / A_i \, p_j$, *then* J_i *precedes* J_k.

Authors recall Emmons Rules and generalize them to take into account release dates. Unfortunately, these rules are valid if preemption is allowed (or, for some of them, if jobs have identical processing times).

Several dominance properties have been introduced in “A Branch-and-Bound algorithm to minimize total tardiness with different release dates” work. These rules focus on the jobs in NS plus the last job of P. They determine that some precedence constraints can be added. Such constraints allow us to adjust release dates and to filter PF. All these rules are used in our Branch-and-Bound procedure. On top of this, authors also consider dominance properties that take into account the complete sequence P . Informally speaking, they most basic rule states that if the current sequence P can be “improved”, then it is dominated and they can backtrack.

Relaxing non-preemption is a standard technique to obtain lower bounds for non-preemptive scheduling problems. Unfortunately, the problem with no release dates (for which the preemption is not useful) is already NP-Hard, so some additional constraints have to be relaxed to obtain a lower bound in polynomial time. From now on, assume that jobs are sorted in non-decreasing order of due dates. First, Chu’s lower bound is recalled then, a new lower bound is described and finally, they show how the Generalized Emmons Rules can be used to improve this lower bound.

Focacci in his work “Solving Combinatorial Optimization Problems in Constraint Programming” has recently proposed an original approach based on Constraint Programming to compute a lower bound of $1|r_i|\sum T_i$. In this approach, each job is associated with a constrained variable identifying all possible positions (first, second, third, etc.) that the job can assume in a schedule. Following this idea, authors present some rules that deduce that a job cannot be executed in some

positions. This information allows them to adjust the release dates and to “filter” the sets PF and PL.

From now on, they assume that jobs are sorted in non-decreasing order of due dates. To simplify the presentation, they also assume that no job has been sequenced in the right part of the schedule, i.e., $P = \emptyset$. If P is not empty, they can “remove” the jobs of P and apply the rules described below. Of course, the tardiness of the jobs that were in P has to be added to the lower-bounds.

Suppose now that they want to compute a lower bound of the total tardiness under the hypothesis that J_i is scheduled in the k^{th} position. They first assign the completion time $C_{[k]}$ to J_i and they reassign all other completion times to all other jobs. Following these new assignments, they have a new lower bound. If it is greater than $\bar{T}$, then J_i cannot be in the k^{th} position.

To implement this constraint propagation rule, they just have to use the O(nlogn) algorithm of Chu, to compute the values $C_{[1]},C_{[2]}...,C_{[n]}$. Then, for each job J_i and for each position k, the lower-bound can be recomputed in linear time thanks to the reassignment rules provided above. This leads to an overall time complexity of $O(n^3)$.

Actually, this algorithm can be improved as follows. Assume that they have computed the lower bound under the assumption that Ji is scheduled in position k. To compute the lower bound under the assumption that Ji is scheduled in position k −1, they just have to exchange the assignments of job i and job [k − 1]. The modification of the lower bound is then max(0, C[k−1] − di)−max(0,C[k]−di) +max(0,C[k]−d[k−1])−max(0,C[k−1]−d[k−1]). Hence, they can “try” all possible positions for Ji in linear time. All impossible positions can thus be computed in O(n2).

Computational results show that the proposed approach outperforms the best known procedures. All “ingredients” described in this paper are useful to reduce the search space. However, the look ahead technique presented earlier seems to be very costly in terms of CPU time compared to the corresponding reduction of the search tree. This is due to the relatively high complexity of the lower bound lb_2 that is used several times in the look ahead. They tried to use some weaker lower bound like lb_1 but it does not reduce the search space.

Authors have presented new lower-bounds, new constraint propagation techniques and new dominance properties for $1|r_i| \sum T_i$. Computational results show that the proposed approach outperforms the best known procedures. They think that several techniques presented in this paper can be extended to more complex criteria such as weighted tardiness or weighted flow time.