INFORMS Journal on Computing
Vol. 8, No. 4, Fall 1996

0899-1499/96/0804-0402 $01 25
© 1996 INFORMS

A Branch-and-Bound Algorithm for Single-Machine Earliness–Tardiness Scheduling with Idle Time

J. A. HOOGEVEEN / *Department of Mathematics and Computing Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands, Email address: slam@win.tue.nl*

S. L. VAN DE VELDE / *Department of Mechanical Engineering, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands, Email address: s.l.vandevelde@wb.utwente.nl*

(Received: February 1992; revised: September 1995; accepted: October 1995)

We address the NP-hard single-machine problem of scheduling *n* independent jobs so as to minimize the sum of α times total completion time and β times total earliness with $\beta > \alpha$, which can be rewritten as an earliness–tardiness problem. Postponing jobs by leaving the machine idle may then be advantageous. The allowance of machine idle time between the execution of jobs singles out our problem from most concurrent research on problems with earliness penalties. Solving the problem to optimality poses a computational challenge, since the possibility of leaving the machine idle has a major effect on designing a branch-and-bound algorithm in general, and on computing lower bounds in particular. We present a branch-and-bound algorithm which is based upon many dominance rules and various lower bound approaches, including relaxation of the machine capacity, data manipulation, and Lagrangian relaxation. The algorithm is shown to solve small instances with up to 20 jobs.

Recently, we have seen much interest in machine scheduling models that penalize both early and tardy completions of jobs. We refer to Baker and Scudder[1] for an overview. These types of models are supposed to capture the just-in-time concept, whose basic premise is to reduce costly inventories by enforcing on-time deliveries throughout the entire manufacturing process. In theory, on-time deliveries may be achieved by allowing machine idle time. In practice, however, idle time is highly controversial, because of lost production capacity, for instance.

In this paper, we consider an NP-hard single-machine earliness–tardiness problem in which the insertion of machine idle time is allowed, and present a branch-and-bound algorithm for its solution. As we will see later, the possibility to leave the machine idle poses a computational challenge and affects significantly the design of a branch-and-bound algorithm.

We consider the following problem. A set $\mathcal{J} = \{J_1, \ldots, J_n\}$ of n independent jobs has to be scheduled on a single machine that is continuously available from time zero onward. The machine can handle at most one job at a time. Job J_j ($j = 1, \ldots, n$) requires a positive integral uninterrupted processing time p_j and should ideally be completed exactly on its due date d_j. A *schedule* specifies for each job J_j a completion time C_j such that the jobs do not overlap in their execution. The order in which the machine processes the jobs is called the *job sequence*. For a given schedule, the earliness of J_j is defined as $E_j = \max\{0, d_j - C_j\}$ and its tardiness as $T_j = \max\{0, C_j - d_j\}$. In addition, we define *maximum earliness* as $E_{\max} = \max_{1 \leq j \leq n} E_j$ and *maximum tardiness* as $T_{\max} = \max_{1 \leq j \leq n} T_j$. Accordingly, J_j is called *early*, *just-in-time*, or *tardy* if $C_j < d_j$, $C_j = d_j$, or $C_j > d_j$, respectively. The cost of a schedule σ is the weighted sum of total completion time and total earliness, that is,

$$f(\sigma) = \alpha \sum_{j=1}^{n} C_j + \beta \sum_{j=1}^{n} E_j,$$

where α and β are given positive weights with $\beta > \alpha$. Without loss of generality, we also assume α and β to be integral and relatively prime. We are interested in the case $\beta > \alpha$, since insertion of machine idle time may be advantageous only in this case. The cost function $f(\sigma)$ arguably measures inventory costs in a machine scheduling environment: total completion time measures the work-in-process inventories, and total earliness measures the storage inventories due to early completions. The problem, referred to as problem (P), is to find a feasible schedule σ that minimizes $f(\sigma)$.

Problem (P) is NP-hard. By definition we have that $T_j = C_j + E_j - d_j$ for $j = 1, \ldots, n$, and the cost function can therefore alternatively be written as

$$(\alpha - \gamma) \sum_{j=1}^{n} C_j + (\beta - \alpha) \sum_{j=1}^{n} E_j + \gamma \sum_{j=1}^{n} T_j + \gamma \sum_{j=1}^{n} d_j,$$

for any $0 \leq \gamma \leq \alpha$. Garey, Tarjan, and Wilfong[5] prove that minimizing this cost function with $\gamma = \alpha$ and $\beta > \alpha$ is NP-hard.

Problem (P) was identified by Kanet and Christie[12] and studied by Fry and Leong.[3] They formulated it as an integer linear problem and used a standard code to find an optimal schedule. Not surprisingly, this method already requires excessive computation times for small instances.

Subject classifications: Single-machine scheduling.
Other key words. Earliness, tardiness, machine idle-time, branch and bound algorithm, Lagrangian relaxation.

Their formulation is 'weak' in that the linear programming relaxation gives weak lower bounds, which seriously impairs the performance of any standard integer linear program solver. Also, a general code does not take advantage of the problem structure. Fry, Leong, and Rakes[4] compare the performance of the integer linear programming approach with the performance of a rudimentary branch-and-bound algorithm for the problem $1\|\Sigma_{j=1}^{n}(\alpha C_j + \beta E_j + \gamma T_j)$ in which idle time is allowed; indeed, they find that their branch-and-bound algorithm is much faster. Inspecting the alternative rendition of the cost function we see that their problem is equivalent with ours.

Both our algorithm and the branch-and-bound algorithm by Fry, Leong, and Rakes hinge upon the observation that the search for an optimal schedule can be reduced to a search over the $n!$ different job sequences. This is possible, since there is a clear-cut method to insert machine idle time to minimize total cost for any *given* sequence. This method, which requires $O(n^2)$ time, is described in Section 1. In Section 2, we discuss the design of the branch-and-bound algorithm, including the upper bound, the branching rule, the search strategy, and the dominance rules. The derivation of lower bounds is significantly complicated by the possibility of machine idle time. The range of the due dates in proportion to the processing times mainly determines how much idle time is desired. To cope with the different problem instances, we present five approaches for lower bound computation, including Lagrangian relaxation, in Section 3. The branch-and-bound algorithm is based upon many dominance rules and various lower bound approaches. Unfortunately, it can handle only small problem sizes; the computational results presented in Section 4 exhibit that we can solve instances with up to 20 jobs. Conclusions are given in Section 5.

Throughout the paper, we follow the three-field notation of Graham, Lawler, Lenstra, and Rinnooy Kan[6] to classify scheduling problems.

1. The Insertion of Idle Time for a Given Sequence

In this section, we describe a procedure to insert machine idle time so as to minimize total cost for a *given* sequence. This procedure is not new. Similar methods have been presented (cf. Baker and Scudder[1]), including those presented by Fry, Leong, and Rakes[4] for the $1\|\Sigma_{j=1}^{n}(\alpha C_j + \beta E_j + \gamma T_j)$ problem and by Garey, Tarjan, and Wilfong[5] for the $1\|\Sigma_{j=1}^{n}(E_j + T_j)$ problem. This is not surprising: as we have already noted, $T_j = C_j + E_j - d_j$ for all j; for specific choices for α and β, our problem is equivalent with theirs. Since the basis of the procedure is well known, we just present our implementation. For a proof of correctness, see Hoogeveen and van de Velde[10].

Suppose that the scheduling order is $\sigma = (J_n, \ldots, J_1)$. Since no job can start before time zero, we need the constraint that $C_l \geq \Sigma_{j=l}^{n} p_j$ for $l = 1, \ldots, n$. We use an inductive procedure for finding an optimal schedule for σ. It finds an optimal schedule for the subsequence $(J_l, \ldots, J_1)$, given an optimal schedule for the subsequence $(J_{l-1}, \ldots, J_1)$, for $l = 2, \ldots, n$; we initiate the procedure by scheduling J_1 to be completed at time $\max\{d_1, \Sigma_{j=1}^{n} p_j\}$. When adding J_l to the subsequence $(J_{l-1}, \ldots, J_1)$, we first check whether $\Sigma_{j=l}^{n} p_j \leq d_l \leq C_{l-1} - p_{l-1}$; if so, then putting $C_l = d_l$ yields an optimal schedule for $(J_l, \ldots, J_1)$. If $d_l < \Sigma_{j=l}^{n} p_l$, then we obtain an optimal schedule for $(J_l, \ldots, J_n)$ by putting $C_l = \Sigma_{j=l}^{n} p_j$. If $d_l > C_{l-1} - p_{l-1}$, then we tentatively put $C_l = C_{l-1} - p_{l-1}$. Define $\mathcal{Q}_l$ as the set containing J_l and its immediate followers, that is, the group of jobs that are executed after J_l with no machine idle time in between. We now compute the optimal delay of the jobs in $\mathcal{Q}_l$, disregarding the jobs not in $\mathcal{Q}_l$. A delay of one unit of time increases the completion time of each job in $\mathcal{Q}_l$ by one and decreases the earliness of each early job in $\mathcal{Q}_l$ by one; the total effect on the cost is equal to the primitive directional derivative $g_l = \alpha|\mathcal{Q}_l| - \beta n_l$, where $|\mathcal{Q}_l|$ denotes the number of jobs in $\mathcal{Q}_l$ and n_l denotes the number of early jobs in $\mathcal{Q}_l$. Obviously, we defer the jobs in $\mathcal{Q}_l$ until g_l becomes nonnegative, that is, as long as $n_l \leq \alpha|\mathcal{Q}_l|/\beta$. Define $a = \lfloor \alpha|\mathcal{Q}_l|/\beta \rfloor$, $k = |\mathcal{Q}_l| - a$, and $E_{[k]}$ as the kth smallest value of the earliness of the jobs in $\mathcal{Q}_l$. If the jobs in $\mathcal{Q}_l$ are deferred by $\delta = E_{[k]}$, then at most a jobs in $\mathcal{Q}_l$ remain early and, due to the choice of a, g_l then becomes nonnegative. Deferring the jobs by δ is only feasible if δ is no larger than the length $\delta_{\max}$ of the period of idle time immediately after the last job in $\mathcal{Q}_l$. If $\delta \leq \delta_{\max}$, then we get an optimal schedule for $(J_l, \ldots, J_1)$ by deferring the jobs in $\mathcal{Q}_l$ by δ. If $\delta > \delta_{\max}$, then we defer the jobs in $\mathcal{Q}_l$ by $\delta_{\max}$. At this point, we repeat the process for J_l: we update $\mathcal{Q}_l$, and evaluate if additional delay of the jobs in $\mathcal{Q}_l$ is advantageous. We now give a step-wise description of the idle time insertion algorithm.

Idle Time Insertion Algorithm

```
Step 1. $C_1 \leftarrow \max\{d_1, \Sigma_{j=1}^{n} p_j\}$; $l \leftarrow 2$.
Step 2. If $l = n + 1$, then go to Step 10.
Step 3. If $d_l < \Sigma_{j=1}^{n} p_j$, then $C_l \leftarrow \Sigma_{j=l}^{n} p_j$; otherwise, put $C_l \leftarrow$
    $\min\{d_l, C_{l-1} - p_{l-1}\}$. If $C_l = d_l$, then go to Step 9.
Step 4. Determine $\mathcal{Q}_l$ and evaluate $g_l$. If $g_l \geq 0$, then go to
    Step 9.
Step 5. Compute $E_j$ for each job $J_j \in \mathcal{Q}_l$.
Step 6. Compute $\delta_{\max}$, i.e., the length of the period of idle
    time immediately after the last job in $\mathcal{Q}_l$.
Step 7. Let $a \leftarrow \lfloor |\mathcal{Q}_l|\alpha/\beta \rfloor$, and $k \leftarrow |\mathcal{Q}_l| - a$. Determine $\delta$ as
    the $k$th smallest earliness value for the jobs in $\mathcal{Q}_l$.
Step 8. Defer the jobs in $\mathcal{Q}_l$ by $\min\{\delta, \delta_{\max}\}$. If $\delta > \delta_{\max}$, then
    go to Step 4.
Step 9. $l \leftarrow l + 1$; go to Step 2.
Step 10. Stop: an optimal schedule for the sequence $(J_n, \ldots, J_1)$
    has been determined.
```

Theorem 1: *The idle time insertion algorithm generates an optimal schedule for a given sequence.*

For a proof, see [10]. As to the complexity of the algorithm, we have that a complete run through the main part of the algorithm, i.e., Steps 4 through 8, takes $O(n)$ time: this is needed to identify the set $\mathcal{Q}_l$, to compute the primitive directional derivative g_l, the values $\delta_{\max}$ and δ, and to defer the jobs, if necessary. The value δ is determined in $O(n)$ time through a median-finding technique. After each run through the main part of the algorithm, a gap between two successive jobs is closed. As at most $n - 2$ such gaps exist, the algorithm runs in $O(n^2)$ time. For the case $2\alpha = \beta$, i.e., for the problem $1\|\Sigma_{j=1}^{n}(E_j + T_j)$, Garey, Tarjan, and Wilfong[5] show

Shipments are Forbidden. *Computers and Operations Research 14,* 363–368.

4. T.D. FRY, G.K. LEONG, and T.R. RAKES (1987). Single Machine Scheduling: A Comparison of Two Solution Procedures. *Omega 15,* 277–282.
5. M.R. GAREY, R.E. TARJAN, and G.T. WILFONG (1988). One-processor Scheduling with Symmetric Earliness and Tardiness Penalties. *Mathematics of Operations Research 13,* 330–348.
6. R.L. GRAHAM, E.L. LAWLER, J.K. LENSTRA, and A.H.G. RINNOOY KAN (1979). Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey. *Annals of Discrete Mathematics 5,* 287–326.
7. N.G. HALL, W. KUBIAK, and S.P. SETHI (1991). Earliness–Tardiness Scheduling Problems II. Deviation of Completion Times about a Restrictive Common Due Date. *Operations Research 39,* 847–856.
8. J.A. HOOGEVEEN, H. OOSTERHOUT, and S.L. VAN DE VELDE (1994). New Lower and Upper Bounds for Scheduling around a Small Common Due Date. *Operations Research 42,* 102–110.
9. J.A. HOOGEVEEN and S.L. VAN DE VELDE (1991). Scheduling Around a Small Common Due Date. *European Journal of Operational Research 55,* 237–242.
10. J.A. HOOGEVEEN and S.L. VAN DE VELDE (1992). *Minimizing Total Inventory Cost on a Single Machine in Just-in-Time Manufacturing,* Memorandum COSOR 9220, Eindhoven University of Technology, Eindhoven, The Netherlands.
11. J.A. HOOGEVEEN and S.L. VAN DE VELDE (1996). Earliness–tardiness Scheduling Around Almost Equal Due Dates. *INFORMS Journal on Computing,* to appear.
12. J.J. KANET and D.P. CHRISTY (1984). Manufacturing Systems with Forbidden Early Order Departure. *International Journal of Production Research 22,* 41–50.
13. C.N. POTTS and L.N. VAN WASSENHOVE (1985). A Branch-and-Bound Algorithm for the Total Weighted Tardiness Problem. *Operations Research 33,* 363–377.
14. W.E. SMITH (1956). Various optimizers for single-stage production. *Naval Research Logistics Quarterly 1,* 59–66.
15. S.L. VAN DE VELDE (1991). *Machine Scheduling and Lagrangian Relaxation,* Doctoral Thesis, CWI, Amsterdam.