A Fully Polynomial Time Approximation Scheme in Scheduling Deteriorating Jobs

Jin-Yi Cai* Pu Cai† Yixin Zhu‡

Abstract

We consider a scheduling problem with a single machine and a set of jobs which have to be processed sequentially. While waiting for processing, jobs may deteriorate, causing the processing requirement of each job to grow after a fixed waiting time t_0. We prove that the problem of minimizing the makespan—completion time for all jobs—is NP-hard. A fully polynomial time approximation scheme is developed for a special case where the job requirement grows linearly at a job-specific rate after t_0.

1. Introduction

Typical single processor scheduling models deal with N jobs waiting to be processed sequentially, with job i having positive processing time p_i. It is easy to see that the makespan (completion time of the N jobs) is invariant under any scheduling policy that disallows idleness. Thus, research has focused on minimizing weighted flow times and maximizing rewards.

Browne and Yechiali (1990), however, introduced the case where jobs can deteriorate as they await service, causing their processing times to grow while they wait. For these types of models, the makespan is no longer invariant and is a function of the scheduling policy, as are the actual processing times. Specifically, Browne and Yechiali (1990) considered the problem of minimizing the makespan to schedule N jobs on a single machine, in the following situation. The jobs are all available at time 0, with initial processing

*Department of Computer Science, State University of New York at Buffalo, Buffalo, NY 14260. Email: cai@cs.buffalo.edu. Supported in part by NSF grants CCR 9057486 and CCR 9319093, and an Alfred P. Sloan Fellowship.

†Department of Mathematics, Temple University, Philadelphia, PA 19122

‡BNR Inc. 2201 Lakeside Blvd Richardson, TX 75082

time p_i for job i. If job $i's$ processing is delayed until time t, then the processing requirement grows linearly with delay to $p_i + \alpha_i t$, where α_i is job $i's$ processing growth rate. They found that the makespan will be minimized if the jobs are scheduled in an increasing order of p_i/α_i, the ratio of the initial processing time to the growth rate. For the case $p_i = p$, Mosheiov (1991) considered the problem of minimizing the flow time—the sum of completion times—and showed that the optimal sequence to minimize the flow time is V-shaped: Jobs are arranged in descending order of growth rate if they are placed before the minimal growth rate job, and in ascending order if placed after it. Efficient heuristics were also developed there.

In practice, however, one often encounters situations in which jobs deteriorate if they are not processed before certain time. For example, deteriorating in processing time may occur when the machine is losing efficiency while processing a batch of jobs. The machine is assumed to be at maximal efficiency for the first t_0 units of time and starts to lose efficiency after time t_0. The efficiency loss is reflected in the fact that a job which is processed later in time has a longer processing time. Other examples include scheduling of maintenance, repair or cleaning assignments, etc.

In this paper, we consider a general time deteriorating job scheduling problem in which the processing requirement of job i is $p_i + \alpha_i f(t_0, t)$ if the job i starts at time t, where $f(t_0, t) \geq 0$ is an increasing function of t. We prove that the problem of minimizing the makespan for time deteriorating jobs is NP-hard. In fact, for $t_0 \neq 0$, the problem remains NP-hard even for the simple step function $1_{t-t_0>0}$. For the time delayed deteriorating job scheduling problem, i.e., $f(t_0, t) = \max(t - t_0, 0)$, the problem is intuitively more complicated as it is more time dependent, and we leave it as an open problem that it remains NP-hard in this case.

Obviously, the model studied in Browne and Yechiali (1990) will be a special case with $t_0 = 0$. For the case of $f(t_0, t) = \max(t - t_0, 0)$, where $t_0 > 0$, a fully polynomial time approximation scheme is developed. We say that an approximation scheme is a fully polynomial time approximation scheme (cf. Cormen et. al. (1990)) if for any given $\epsilon > 0$, the algorithm provides a solution which approximates the optimal solution by a factor of at most $1 + \epsilon$, and its running time is polynomial both in $1/\epsilon$ and in the size n of the input instance. In many ways, fully polynomial time approximation schemes are the best that one might hope for in solving NP-hard problems.

The rest of the paper is organized as follows. In section 2, we prove that the problem of minimizing the makespan of N deteriorating jobs with time requirement $p_i + \alpha_i f(t_0, t)$ for job i is NP-hard. For the special case of

$f(t_0, t) = \max(t - t_0, 0)$, some preliminary results are given in section 3 and a fully polynomial time approximation scheme is developed in section 4.

2. NP-Hardness

Theorem 1 *The problem of minimizing the makespan of N deteriorating jobs with time requirement $p_i + \alpha_i f(t_0, t)$ for job i starting at time t is NP-hard, where $f(t_0, t) \geq 0$ is some increasing function of t.*

Proof: If $\sum_{i \leq N} p_i \leq t_0$, then the scheduling problem is trivial as no penalty applies. So we suppose $\sum_{i \leq N} p_i > t_0$.

Let $f(t_0, t) = 1_{(t - t_0 > 0)}$, where $1_{(E)}$ is the indicator function of the set E. Consider any scheduling of jobs $i_1, i_2, \ldots, i_N$. Let $k = \max\{j \mid \sum_{s<j} p_{i_s} \leq t_0\}$, then jobs $i_1, i_2, \ldots, i_k$ are all not penalized, since their starting times are all $\leq t_0$, but jobs after that, $i_{k+1}, \ldots, i_N$, if any, are penalized. The makespan under this scheduling is

$$\sum_{i \leq N} p_i + \sum_{j > k} \alpha_{i_j}.$$

Hence a scheduling amounts to a selection of a subset $S \subseteq \{1, 2, \ldots, N\}$ with $\sum_{j \in S} p_j \leq t_0$, and a selection of the next job if any.

For any chosen subset $S \subseteq \{1, 2, \cdots, N\}$ with $\sum_{j \in S} p_j \leq t_0$, we may select at least one more job $j^* \in S^c$ scheduled just after job set S such that $\alpha_{j^*} = \max\{\alpha_j : j \in S^c\}$ without being penalized. Thus, the problem of minimizing the makespan can be formalized as the following optimization problem:

$$\begin{aligned} \min \quad & \sum_{j \in S^c - \{j^*\}} \alpha_j \\ \text{such that}: \quad & \sum_{j \in S} p_j \leq t_0 \\ & \alpha_{j^*} = \max\{\alpha_j : j \in S^c\} \\ & S \cup S^c = \{1, 2, \ldots, N\} \end{aligned} \qquad (1)$$

Obviously, the optimization problem is equivalent to the following problem

$$\max \quad \sum_{j \in S \cup \{j^*\}} \alpha_j$$

$$\text{such that}: \quad \sum_{j \in S} p_j \leq t_0 \tag{2}$$
$$\alpha_{j^*} = \max\{\alpha_j : j \in S^c\}$$
$$S \cup S^c = \{1, 2, \ldots, N\}$$

In the following, we are going to reduce the knapsack problem to the above problem.

Consider the knapsack problem: Given $t_0 > 0$ and $N - 1$ pairs of positive integers $\{(p_1, \alpha_1), \ldots, (p_{N-1}, \alpha_{N-1})\}$, we want to find a subset $S \subseteq \{1, \ldots, N-1\}$ which

$$\max \quad \sum_{j \in S} \alpha_j$$
$$\text{such that}: \quad \sum_{j \in S} p_j \leq t_0 \tag{3}$$

We get an instance for the optimization problem in (2) by letting $p_N > t_0$ and $\alpha_N > \max\{\alpha_j : j = 1, \ldots, N-1\}$. Then in any optimal scheduling, $N \in S^c$ and $j^* = N$, and an optimal solution of (2) yields an optimal solution for the knapsack problem (3). This completes our proof since the knapsack problem is NP-hard. *Q.E.D.*

For a general deteriorating function Theorem 1 shows that the problem of minimizing the makespan is an NP-hard problem. In fact, the problem remains NP-hard even if the deteriorating function is a simple step function. The problem of minimizing the makespan of the time deteriorating jobs with processing time requirement $p_i + \alpha_i \max(t - t_0, 0)$ for job i is intuitively more complicated than the case with the simple step function, as the makespan is more time dependent. We leave it as an open problem to show that the problem of minimizing the makespan is still NP-hard, if the deteriorating function $f(t_0, t)$ is $\max(t - t_0, 0)$.

3. Preliminary Results

In this section, we will present some preliminary results for the time deteriorating job scheduling problem if $f(t_0, t) = \max(t - t_0, 0)$. Obviously, if $t_0 = 0$, then it is the model studied in Borwne and Yechiali (1990).

Let us consider a job sequence $\pi = \{1, 2, \ldots, N\}$, the makespan for the sequence is the completion time of the last job N. Let

$$k = \max\{j; \; p_1 + p_2 + \cdots + p_{j-1} \leq t_0\}.$$

Let Y_i be the actual processing time of the i^{th} job of the policy π, then

$$Y_i = p_i, \quad i = 1, \ldots, k,$$

and

$$Y_i = p_i + \alpha_i(\sum_{j=1}^{i-1} Y_j - t_0), \quad i = k+1, \ldots, N.$$

The makespan under the sequence π is therefore

$$\begin{aligned} M(\pi) &= \sum_{j=1}^{N} Y_j \\ &= t_0 + \left[\sum_{j=1}^{k} p_j - t_0\right] \prod_{r=k+1}^{N} (1+\alpha_r) + \sum_{j=k+1}^{N} p_j \prod_{r=j+1}^{N} (1+\alpha_r). \quad (4) \end{aligned}$$

By using an interchanging argument or the lemma in Browne and Yechiali (1990), we have

Theorem 2 *Given the set of jobs that start at time $\leq t_0$, the makespan is minimized when the jobs that start after t_0 are scheduled in a non-decreasing order of p_i/α_i.*

From theorem 2, the following two corollaries can be easily obtained.

Corollary 1 *If all jobs have the same basic processing time, i.e., p_i is same for all jobs, processing the jobs in a non-increasing order of α_i will minimize the makespan.*

Corollary 2 *If job deteriorating rates are the same for all jobs, i.e., α_i is the same for all the jobs, then the makespan will be minimized when jobs are scheduled in a non-decreasing order of p_i.*

4. A Fully Polynomial Time Approximation Algorithm

In this section, a fully polynomial time approximation algorithm is developed for the problem of minimizing the makespan when $f(t_0, t) = \max(t - t_0, 0)$.

The strategy for the design of this algorithm is the following. For notational convenience, assume there are altogether $N+1$ jobs to be processed.

As shown in the previous section, all jobs starting after the deadline t_0 should be ordered in the non-decreasing order of p_i/α_i, thus giving an optimal scheduling is equivalent to giving the set of jobs S which are finished before t_0, and one more job j^* that carries over the deadline t_0. (We assume the non-trivial case that $\sum_{i\geq 1} p_i > t_0$.) There are at most $N+1$ choices for j^*, thus we can assume it is already given, for the purpose of obtaining a polynomial time algorithm. For any such a choice of j^*, we will inductively, for $i = 1$ to N, build a "net of solutions", which "well approximates" any possible scheduling, including the optimal scheduling. The "net of solutions" is bounded by a polynomial in both $1/\epsilon$ and N in number, and all schedulings in the "net of solutions" are computed in polynomial time in both $1/\epsilon$ and N.

For a chosen j^*, we can rename all remaining jobs as $\{1, 2, \ldots, N\}$. We will denote $\{1, 2, \ldots, i\}$ by $[i]$ and the power set of $[i]$ by $\mathcal{P}([i])$. The cardinality of a set S is denoted by $|S|$. Let $S \subseteq [i]$, let $[i] - S = \{j_1, \ldots, j_{i-|S|}\}$ where

$$\frac{p_{j_1}}{\alpha_{j_1}} \leq \cdots \leq \frac{p_{j_{i-|S|}}}{\alpha_{j_{i-|S|}}}.$$

We define the following functions

$$U_i(S) = \prod_{r=1}^{i-|S|} (1 + \alpha_{j_r}) \tag{5}$$

$$V_i(S) = \sum_{l=1}^{i-|S|} p_{j_l} \prod_{r=l+1}^{i-|S|} (1 + \alpha_{j_r}). \tag{6}$$

For a given (S, j^*), we define the following makespan function

$$M_i(S, j^*) = t_0 + \left(\sum_{j \in S} p_j + p_{j^*} - t_0 \right)^+ U_i(S) + V_i(S),$$

where $x^+ = \max(x, 0)$.

For $i = N$ we will denote $U_N(\cdot)$, $V_N(\cdot)$ and $M_N(\cdot, \cdot)$ by simply $U(\cdot)$, $V(\cdot)$ and $M(\cdot, \cdot)$ respectively. For any S, with $\sum_{j \in S} p_j \leq t_0$, the following scheduling is denoted by (S, j^*): first schedule all jobs in S (in any order), followed by job j^*, and then all remaining jobs in non-decreasing order of p_i/α_i.

As a technical comment, we note that this so-called makespan function $M(\cdot, \cdot)$ does not necessarily represent the actual makespan of a scheduling.

However, if $i = N$, $\sum_{j\in S} p_j \leq t_0$, and $\sum_{j\in S} p_j + p_{j^*} > t_0$, then $M(S, j^*)$ is the makespan of the scheduling (S, j^*) represented by the set $S \subseteq [N]$ and j^*. If we only have $\sum_{j\in S} p_j \leq t_0$ but not $\sum_{j\in S} p_j + p_{j^*} > t_0$, then $M(S, j^*)$ corresponds to the makespan of the scheduling represented by the set S and j^* with an idleness inserted at the end of job j^* till t_0. Equivalently, we can think of the job j^* is prolonged from p_{j^*} to $t_0 - \sum_{j\in S} p_j$. In particular, $M(S, j^*)$ is an upper bound for the actual makespan of the scheduling (S, j^*) represented by the set S and j^*. In any case, the following Theorem holds.

Theorem 3 *For any given $0 < \epsilon < 1$, let $\delta = \epsilon/(2N)$. Let (S, j^*) be an optimal scheduling in the time deteriorating job problem, and let S^* satisfy the following conditions:*

Condition A	$\sum_{j\in S^*} p_j \leq \sum_{j\in S} p_j$
Condition B	$U(S^*) \leq U(S)(1+\delta)^N$
Condition C	$V(S^*) \leq V(S)(1+\delta)^N$

Then, the scheduling (S^*, j^*) has relative error at most ϵ from the optimal scheduling (S, j^*).

Proof: Since the optimal scheduling achieves minimal makespan, we only need to show that the makespan M of the approximating scheduling (S^*, j^*) satisfies

$$\frac{M - M(S, j^*)}{M(S, j^*)} \leq \epsilon.$$

By Condition A, $M(S^*, j^*)$ is an upper bound for the actual makespan M, thus we need only to show that

$$\frac{M(S^*, j^*) - M(S, j^*)}{M(S, j^*)} \leq \epsilon.$$

This follows from the following calculations

$$\begin{aligned} M(S^*, j^*) &= t_0 + \left(\sum_{j\in S^*} p_j + p_{j^*} - t_0\right)^+ U(S^*) + V(S^*) \\ &\leq t_0 + \left(\sum_{j\in S} p_j + p_{j^*} - t_0\right)^+ U(S)(1+\delta)^N + V(S)(1+\delta)^N \end{aligned}$$

$$
\begin{aligned}
&\leq \left[t_0 + \left(\sum_{j\in S} p_j + p_{j^*} - t_0\right)^+ U(S) + V(S)\right](1+\delta)^N \\
&= M(S, j^*)(1+\delta)^N, \qquad (7)
\end{aligned}
$$

and

$$
\begin{aligned}
\frac{M(S^*, j^*) - M(S, j^*)}{M(S, j^*)} &\leq (1+\delta)^N - 1 \\
&\leq e^{\epsilon/2} - 1 \\
&\leq \epsilon \qquad \text{for } \epsilon < 1,
\end{aligned}
$$

where we substituted $\delta = \epsilon/(2N)$. *Q.E.D*

Therefore, the key is to find an approximating schedule which satisfies conditions A, B and C.

Let us inductively construct $\mathcal{L}_i \subseteq \mathcal{P}([i])$, such that for any $S^* \in \mathcal{L}_i$, $\sum_{j\in S^*} p_j \leq t_0$; moreover, for any $S \subseteq [i]$ such that $\sum_{j\in S} p_j \leq t_0$, there exists an $S^* \in \mathcal{L}_i$, satisfying

$$
\begin{aligned}
(i) \quad & \sum_{j\in S^*} p_j \leq \sum_{j\in S} p_j \\
(ii) \quad & U_i(S^*) \leq U_i(S)(1+\delta)^i \\
(iii) \quad & V_i(S^*) \leq V_i(S)(1+\delta)^i
\end{aligned}
$$

For $i = 1$, let $\mathcal{L}_1 = \{\phi\}$ or $\{\phi, \{1\}\} = \mathcal{P}([1])$, depending on whether $p_1 > t_0$ or not. Clearly, the above conditions are satisfied.

Suppose $\mathcal{L}_i$ has the properties (i), (ii), (iii) described above. Let

$$
\mathcal{L}'_{i+1} = \mathcal{L}_i \cup \{S_1 \cup \{i+1\} \mid S_1 \in \mathcal{L}_i \text{ and } \sum_{j\in S_1} p_j + p_{i+1} \leq t_0\}.
$$

Consider the set of triples $\{\, (\sum_{j\in S} p_j,\ U_{i+1}(S), V_{i+1}(S))\ \ |S \in \mathcal{L}'_{i+1}\}$. Divide the range of all values $(U_{i+1}(S), V_{i+1}(S))$ into a "net" of polynomially many subranges, where for each subrange, any two pairs of values (u, v) and (u', v') differ by a factor of at most $1+\delta$, in both components u and v. Now in each $(1+\delta)\times(1+\delta)$ subrange in U_{i+1} and V_{i+1} which is non-empty, choose an $S \in \mathcal{L}'_{i+1}$ which minimizes $\sum_{j\in S} p_j$. These sets S form $\mathcal{L}_{i+1}$.

Thus, for any $S_1 \in \mathcal{L}'_{i+1}$, there exists $S_2 \in \mathcal{L}_{i+1}$, such that

$$\begin{array}{rl} (i') & \sum_{j\in S_2} p_j \le \sum_{j\in S_1} p_j \\ (ii') & U_{i+1}(S_2) \le U_{i+1}(S_1)(1+\delta) \\ (iii') & V_{i+1}(S_2) \le V_{i+1}(S_1)(1+\delta) \end{array}$$

Now to verify (i), (ii), (iii) for $\mathcal{L}_{i+1}$, consider any $S \subseteq [i+1]$, such that $\sum_{j\in S} p_j \le t_0$.

Case 1. If $S \subseteq [i]$, then from the induction hypothesis, there exists $S^* \in \mathcal{L}_i$ which satisfies the conditions (i), (ii), (iii). Since $\sum_{j\in S} p_i \le t_0$, we have $\sum_{j\in S^*} p_i \le t_0$ by (i). Since $S^* \in \mathcal{L}'_{i+1}$, there exists $S^{**} \in \mathcal{L}_{i+1}$ which satisfies (i'), (ii'), (iii'), thus,

$$\begin{aligned} \sum_{j\in S^{**}} p_j &\le \sum_{j\in S^*} p_j \le \sum_{j\in S} p_j, \\ U_{i+1}(S^{**}) &\le U_{i+1}(S^*)(1+\delta) \\ &= U_i(S^*)(1+\alpha_{i+1})(1+\delta) \\ &\le U_i(S)(1+\alpha_{i+1})(1+\delta)^{i+1} \\ &= U_{i+1}(S)(1+\delta)^{i+1}, \\ V_{i+1}(S^{**}) &\le V_{i+1}(S^*)(1+\delta) \\ &= [V_i(S^*)(1+\alpha_{i+1}) + p_{i+1}](1+\delta) \\ &\le [V_i(S)(1+\alpha_{i+1})(1+\delta)^i + p_{i+1}](1+\delta) \\ &\le V_{i+1}(S)(1+\delta)^{i+1}. \end{aligned}$$

Case 2. If $i+1 \in S$, let $\tilde{S} = S - \{i+1\} \subseteq [i]$. Since $\sum_{j\in S} p_j \le t_0$, we have $\sum_{j\in\tilde{S}} p_j \le t_0 - p_{i+1}$. In particular $\sum_{j\in\tilde{S}} p_j \le t_0$. Thus, there exists $S^* \in \mathcal{L}_i$ which satisfies conditions (i), (ii), (iii), therefore,

$$\sum_{j\in S^*} p_j \le \sum_{i\in\tilde{S}} p_j \le t_0 - p_{j+1}$$

Hence, $S^* \cup \{i+1\} \in \mathcal{L}'_{i+1}$; and also,

$$U_{i+1}(S^* \cup \{i+1\}) = U_i(S^*), \quad U_{i+1}(S) = U_i(\tilde{S}),$$

$$V_{i+1}(S^* \cup \{i+1\}) = V_i(S^*), \quad V_{i+1}(S) = V_i(\tilde{S}).$$

Let $S_1 = S^* \cup \{i+1\}$, then there exists $S_2 \in \mathcal{L}_{i+1}$ which satisfies (i'), (ii') (iii'). So

$$\begin{aligned}
\sum_{j \in S_2} p_j &\leq \sum_{j \in S_1} p_j \\
&\leq \sum_{j \in \tilde{S}} p_j + p_{i+1} \\
&= \sum_{j \in S} p_j, \\
U_{i+1}(S_2) &\leq U_{i+1}(S_1)(1+\delta) \\
&= U_i(S^*)(1+\delta) \\
&\leq U_i(\tilde{S})(1+\delta)^{i+1} \\
&= U_{i+1}(S)(1+\delta)^{i+1}, \\
V_{i+1}(S_2) &\leq V_{i+1}(S_1)(1+\delta) \\
&= V_i(S^*)(1+\delta) \\
&\leq V_i(\tilde{S})(1+\delta)^{i+1} \\
&= V_{i+1}(S)(1+\delta)^{i+1}.
\end{aligned}$$

This completes the inductive construction of the sequence $\mathcal{L}_i \in \mathcal{P}[i], i = 1, 2, \cdots, N$.

Thus we have developed the approximation scheme which satisfies conditions A, B and C for any $i = 1, 2, \cdots, N$. If n bounds the input size of the problem, then for scheduling time deteriorating jobs on a single machine, there are $N \leq n$ stages and there are $N \leq n$ different j^* to try. Since each stage has $O(n/\delta)^2$ sets and $\delta = \Omega(\epsilon/n)$, the total time requirement is $O(n^2)O(n^4/\epsilon^2) = O(n^6/\epsilon^2)$.

References

[1] Baker, K. and G. Scudder, "Sequencing With Earliness and Tardiness Penalties: A Review," *Operations Research* **38** (1990), 22-38.

[2] Browne, S. and U. Yechiali, " Scheduling Deteriorating Jobs on a Single Processor," *Operations Research* **38** (1990), 495-498.

[3] Chu, C., "Efficient heuristics to minimize total flow time with release dates," *Operations Research Letters* **12** (1992), 321-329.

[4] Cormen, T., C. Leiserson, R. Rivest. *Introduction to Algorithm* The MIT Press, Cambridge, Massachusetts. 1990.

[5] Mosheiov, G., "V-shaped Policies for Scheduling Deteriorating Jobs," *Operations Research* **39** (1991), 495-498.