

Scheduling Resumable Deteriorating Jobs

Stanisław Gawiejnowicz

Adam Mickiewicz University, Umultowska 87, 61-614 Poznań, Poland
stgawiej@amu.edu.pl

Alexander Kononov

Sobolev Institute of Mathematics, Prospekt Akademika Koptyuga no. 4, 630090 Novosibirsk, Russia
alvenko@math.nsc.ru

1 Introduction

We consider the following single-machine time-dependent scheduling problem with constraints on the machine availability. There is given a set of independent and deteriorating jobs $J_j, 1 \leq j \leq n$, to be processed on a single machine, starting from time $t_0 > 0$. The processing time p_j of job J_j is a proportional function of the starting time S_j of this job, $p_j = \alpha_j S_j$, where $S_j \geq t_0$ and deterioration rate $\alpha_j > 0$ for $1 \leq j \leq n$. The machine is not continuously available and there are h disjoint periods of the machine non-availability. These periods are described by time intervals $[W_{i,1}, W_{i,2}]$, where $t_0 < W_{1,1}$ and $W_{i,1} < W_{i,2}$ for $1 \leq i \leq h$. Since in real-world applications the number of non-availability periods is usually small, we will assume that $h < n$. Moreover, since any such non-availability period can interrupt the processing of any job, we will assume that the jobs are *resumable*, i.e. if a job has been interrupted by the start time of a non-availability period, then this job does not need to be restarted and can be completed after the machine becomes available again. The applied criterion of schedule optimality is the maximum completion time $C_{\max} = \max_{1 \leq j \leq n} \{C_{[j]}\}$, where $C_{[j]}$ denotes the completion time of the j -th job in schedule. For simplicity of presentation, the above problem will be denoted in short by $RDP(h)$.

The problem $RDP(h)$, introduced by Wu and Lee [7] for $h = 1$, is a combination of time-dependent scheduling with scheduling on a machine with non-availability periods. Time-dependent scheduling has numerous practical applications, e.g. in financial management and modelling service operations. Scheduling on machines with non-availability periods is applied in manufacturing environments in which machines have conservation or maintenance breaks, caused by production or service reasons. We refer the reader to reviews by Cheng et al. [1] and Lee [5] for more details on time-dependent scheduling and scheduling on machines with non-availability periods, respectively.

In this talk, first we extend some of the results obtained by Gawiejnowicz [2] and Ji et al. [4], where a counterpart of the problem $RDP(1)$ with *nonresumable* jobs has been considered. In particular, we prove the ordinary \mathcal{NP} -hardness of the $RDP(1)$ problem and thus we give an answer to the open problem stated in [4]. Next, we present a pseudopolynomial dynamic programming algorithm, DP , for the problem $RDP(h)$ in the case when h is a fixed number. After that we show that the algorithm DP allows us to construct an FPTAS for the $RDP(1)$ problem. Finally, we prove that for the problem $RDP(h)$ with $h \geq 2$ there does not exist a polynomial-time approximation algorithm with a constant worst-case ratio, unless $\mathcal{P} = \mathcal{NP}$.

2 Results

Let a machine, starting from time t_0 , execute n jobs without idle times, let the processing times of the jobs be in the form of $p_j = \alpha_j S_j$ for $1 \leq j \leq n$, and let $[W_{h,1}, W_{h,2}], h \geq 1$, be the last interval

such that the machine executes some jobs after $W_{h,2}$. Let $W_{0,1} = 0$, $W_{0,2} = t_0$ and $k_0 = 0$. Then

$$C_{[n]} = \sum_{i=0}^h (W_{i,2} - W_{i,1}) \prod_{j=k_i+1}^n (1 + \alpha_{[j]}). \tag{1}$$

where k_i is the index of the job which was interrupted by the start time of the non-availability period $[W_{i,1}, W_{i,2}]$, $0 \leq i \leq h$.

Throughout the talk, the job which has been started before and completed not earlier than the start time of a non-availability period will be called a *critical job*, i.e. if for some $1 \leq j \leq n$ and $1 \leq i \leq h$ there hold inequalities $S_{[j]} < W_{i,1}$ and $C_{[j]} \geq W_{i,1}$, then $J_{[j]}$ is a critical job.

First we formulate the following two properties of an optimal schedule of the problem $RDP(h)$.

Property 1 *For the problem $RDP(h)$ there exists an optimal schedule such that jobs start in nondecreasing order of their processing times inside time interval $[W_{i-1,2}, W_{i,1})$ for $1 \leq i \leq h$.*

Property 2 *For the $RDP(1)$ problem there exists an optimal schedule such that the job J_{\max} with deterioration rate $\alpha_{\max} = \max_{1 \leq j \leq n} \{\alpha_j\}$ is the critical job.*

Now we pass to the presentation of our results. We start with the proof of \mathcal{NP} -hardness of the problem $RDP(1)$. This proof is based on Property 2. The main idea is as follows. Let $t_0 = 1$ and let J_{\max} be the critical job. We can separate the set of all jobs except J_{\max} into two sets, N_1 and N_2 . Set N_1 contains the jobs which are executed in an optimal schedule before the critical job and set N_2 contains the remaining jobs. The completion time of all jobs from N_1 does not depend on permutation of these jobs and is equal to $C' = \prod_{J_j \in N_1} (1 + \alpha_{[j]})$. Moreover, C' has to be smaller than W_{11} . We can rewrite (1) as $C_{[n]} = (1 + \alpha_{\max}) \prod_{J_j \in N_1} (1 + \alpha_j) + (W_{i,2} - W_{i,1}) \prod_{J_j \in N_2} (1 + \alpha_j)$. Since the first term is a constant for any given instance of the problem $RDP(1)$, $C_{[n]}$ reaches minimum if and only if C' reaches maximum. This reasoning implies that the problem $RDP(1)$ is polynomially equivalent to the Subset Product problem [3] which in the decision version is \mathcal{NP} -complete.

Now we present the dynamic programming algorithm DP for solving the $RDP(1)$ problem. The algorithm DP goes through n phases. The k th phase, $1 \leq k \leq n$, produces a set \mathcal{S}_k of states. Any state in \mathcal{S}_k is a vector $S = [s_1, s_2]$. The vector S encodes a partial schedule for the first k jobs under the assumption that J_{\max} , by Property 2, is the critical job. The component s_1 of S is the completion time of jobs which have been completed before time $W_{1,1}$ and the component s_2 is the additional total processing time of jobs which start after time $W_{1,2}$.

The sets $\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_k$ are constructed iteratively. An initial set \mathcal{S}_0 contains the only state $[t_0, \Delta]$, where $\Delta = W_{1,2} - W_{1,1}$ is the length of the non-availability period. The set \mathcal{S}_k is obtained from the set \mathcal{S}_{k-1} via two mappings, F_1 and F_2 , which translate the states of the "old" state space \mathcal{S}_{k-1} into the states of the "new" state space \mathcal{S}_k . More precisely, $\mathcal{S}_k = \{F(\alpha_k, S) : S \in \mathcal{S}_{k-1}, F \in \{F_1, F_2\}\}$, where $1 \leq k \leq n$. Intuitively speaking, mapping F_1 "puts" $J_{[k]}$ after time $W_{1,2}$ and mapping F_2 "puts" $J_{[k]}$ before time $W_{1,1}$, if it is possible for the given state. The form of the criterion function $G(S)$ for the problem $RDP(1)$ is given in (1). The algorithm DP is as follows.

Algorithm DP for the problem $RDP(1)$

```

 $\mathcal{S}_0 \leftarrow \{[t_0, \Delta]\};$ 
for  $k \leftarrow 1$  to  $n - 1$  do
     $\mathcal{S}_k \leftarrow \emptyset;$ 
    for each  $S \in \mathcal{S}_{k-1}$  do
         $F_1(\alpha_k, s_1, s_2) \leftarrow [s_1, s_2(1 + \alpha_k)];$ 

```

```

if  $s_1(1 + \alpha_k) \leq W_{1,1}$  then  $F_2(\alpha_k, s_1, s_2) \leftarrow [s_1(1 + \alpha_k), s_2]$ ;
 $\mathcal{S}_k \leftarrow \mathcal{S}_k \cup F_1(\alpha_k, s_1, s_2) \cup F_2(\alpha_k, s_1, s_2)$ ;
end
end
for each  $S \in \mathcal{S}_n$  do
   $G(S) \leftarrow \tau + s_2$ ;
return  $\min\{G(S) : S \in \mathcal{S}_n\}$ 

```

Theorem 1 *For the problem $RDP(1)$ there exists an FPTAS.*

Proof. The mappings F_1 and F_2 are vectors of polynomials with nonnegative coefficients, and the polynomial functions in F_1 and F_2 that yield the first coordinates are polynomials which do not depend on the second coordinate s_2 . Moreover, all polynomials linearly depend on s_1 and s_2 . The inequality inside operator **if** may be checked in polynomial time, does not depend on s_2 and has the positive coefficient of s_1 . Goal function G is a polynomial function with nonnegative coefficients. Moreover, the length of the binary encoding of every value obtained by the algorithm DP is polynomially bounded in the input size, since the length does not exceed $\log((W_{1,2} - W_{1,1}) \prod_{j=1}^n (1 + \alpha_j)) \leq n \log \max\{1 + a_{\max}, W_{1,2}\}$. Therefore, the algorithm DP satisfies conditions of Lemma 7.1 and Theorem 3.5 from the paper Woeginger [6], and the problem $RDP(1)$ is a *DP-benevolent* problem. Hence, for the problem $RDP(1)$ there exists an FPTAS. ■

Notice that Property 1 also provides a dynamic programming algorithm for the problem $RDP(h)$ with $h = \text{const}$. However, in this case the problem $RDP(h)$ is not a DP-benevolent problem. Moreover, using a simple gap reduction technique we can prove the following result.

Theorem 2 *For the problem $RDP(h)$ with $h \geq 2$, there does not exist a polynomial-time approximation algorithm with a constant worst-case ratio $r > 1$, unless $\mathcal{P} = \mathcal{NP}$.*

Acknowledgement. The research of A. Kononov was supported by RFBR grants 05-01-00075-a and 06-01-00960-a.

References

- [1] T-C.E. Cheng, Q. Ding and B.M-T. Lin (2004), A concise survey of scheduling with time-dependent scheduling times, *European Journal of Operational Research* **152**, 1 – 13.
- [2] S. Gawiejnowicz (2007), Scheduling deteriorating jobs subject to machine or job availability constraints, *European Journal of Operational Research* **180**, 472 – 478.
- [3] M.R. Garey and D.S. Johnson (1979), *Computers and Intractability. A Guide to the Theory of \mathcal{NP} -completeness*, W. H. Freeman, San Francisco, CA.
- [4] M. Ji, Y. He and T-C.E. Cheng (2006), Scheduling linear deteriorating jobs with an availability constraint on a single machine, *Theoretical Computer Science* **362**, 115 – 126.
- [5] C-Y. Lee (2004), Machine scheduling with availability constraints, Chapter 22 in J.Y-T. Leung (ed.), *Handbook of Scheduling*, Chapman and Hall/CRC, Boca Raton.
- [6] G. Woeginger (2000), When does a dynamic programming formulation guarantee the existence of an FPTAS ?, *INFORMS Journal on Computing* **12**, 57 – 73.
- [7] C-C. Wu and W-C. Lee (2003), Scheduling linear deteriorating jobs to minimize makespan with an availability constraint on a single machine, *Information Processing Letters* **87**, 89 – 93.