Scheduling with step-improving processing times

T.C. Edwin Cheng * YONG HE † HAN HOOGEVEEN ‡§ MIN JI ¶
GERHARD J. WOEGINGER †

Abstract

We consider the scheduling problem of minimizing the makespan on a single machine with step-improving job processing times around a common critical date. For this problem we give an NP-hardness proof, a fast pseudo-polynomial time algorithm, an FPTAS, and an on-line algorithm with best possible competitive ratio.

Keywords. Scheduling; knapsack problem; approximation scheme; competitive analysis.

1 Introduction

Recent years have shown a growing interest in the area of scheduling with time-dependent processing times; we refer the reader to the survey paper [1] by Cheng, Ding & Lin for more information on this area. In this short technical note, we will concentrate on the following single machine scheduling problem with time-dependent processing times: There are \( n \) independent jobs \( J_1, \ldots, J_n \) with a common critical date \( D \). All jobs are available for processing at time 0. The processing time of job \( J_j \) \((j = 1, \ldots, n)\) is specified by two integers \( a_j \) and \( b_j \) with \( 0 \leq b_j \leq a_j \). If job \( J_j \) is started at some time \( t < D \), then its processing time equals \( a_j \); if it is started at some time \( t \geq D \) then its processing time is \( a_j - b_j \). The goal is to find a non-preemptive schedule that minimizes the makespan, that is, the completion time of the last job.

In this note, we will derive a number of results for this scheduling problem. Most of our algorithmic results are based on the observation that the scheduling problem essentially boils down to a combination of two underlying hidden knapsack problems; see Section 2.

---

*E-mail: lgtcheng@polyu.edu.hk. Department of Logistics, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong SAR, China.

†E-mail: mathhey@zju.edu.cn. Department of Mathematics, State Key Lab of CAD & CG, Zhejiang University, Hangzhou 310027, China.

‡Corresponding author

§E-mail: slam@cs.uu.nl. Department of Computer Science, Utrecht University, P.O.Box 80089, 3508 TB Utrecht, The Netherlands.

¶E-mail: jimkeen@math.zju.edu.cn. Department of Mathematics, State Key Lab of CAD & CG, Zhejiang University, Hangzhou 310027, China.

‖E-mail: gwoegi@win.tue.nl. Department of Mathematics and Computer Science, TU Eindhoven, P.O. Box 513, 5600 MB Eindhoven, The Netherlands.
As a consequence, a number of standard results from the knapsack literature can be carried over directly to the scheduling problem. We thus get a pseudo-polynomial time algorithm and a fully polynomial time approximation scheme (FPTAS) for it. We also show that the scheduling problem is NP-hard in the ordinary sense; see Section 3. Finally, we construct an on-line algorithm with the best possible worst-case ratio 2 for a natural on-line version of this scheduling problem; see Section 4. Our results provide a complete picture of this scheduling problem.

2 The two underlying knapsack problems

This section translates the scheduling problem into two corresponding knapsack problems. For every job \(J_j\), we denote by \(c_j = a_j - b_j \geq 0\) the difference between \(a_j\) and \(b_j\). We let \(J = \{1, 2, \ldots, n\}\) denote the index set of all jobs. For an index set \(I \subseteq J\), we will write \(a(I)\) as a short-hand notation for \(\sum_{i \in I} a_i\), \(b(I)\) for \(\sum_{i \in I} b_i\), and \(c(I)\) for \(\sum_{i \in I} c_i\). Furthermore, we will assume

\[
D \leq a(J) \tag{1}
\]

Otherwise, the problem instance would be trivial: The optimal schedule processes all jobs before the critical date with a makespan of \(a(J)\). Next, let us consider an optimal schedule \(\sigma\) for any given instance. Let \(X \subseteq J\) denote the index set of the jobs with starting times smaller than \(D\) in \(\sigma\), and let \(Y = J - X\) denote the index set of the jobs with starting times greater than or equal to \(D\). Schedule \(\sigma\) belongs to one of the two possible scenarios \(a(X) \leq D - 1\) and \(a(X) \geq D\).

In the first scenario, the constraint \(a(X) \leq D - 1\) may be rewritten as \(a(J) - a(Y) \leq D - 1\). In this scenario, all jobs starting before the critical date \(D\) also complete before the critical date \(D\). Without loss of generality we may assume that the jobs in \(X\) are processed during the time interval \([0; a(X)]\), that the machine then stands idle during \([a(X); D]\), and that the remaining jobs in \(Y\) are executed contiguously from time \(D\) onwards. Because of (1), the corresponding makespan equals \(D + c(Y)\). The best schedule in the first scenario corresponds to the solution of the following problem:

\[
Z_1 := \min c(Y) \quad \text{subject to} \quad a(Y) \geq a(J) - D + 1; \quad Y \subseteq J. \tag{2}
\]

Note that the optimization problem in (2) is a knapsack problem subject to a covering constraint; see also Section 3 below.

In the second scenario with \(a(X) \geq D\), we may assume that the jobs in \(X\) are processed during the time interval \([0; a(X)]\), and that the remaining jobs in \(Y\) are processed during \([a(X); a(X) + c(Y)]\). The corresponding makespan equals

\[
a(X) + c(Y) = b(X) + c(X) + c(Y) = c(J) + b(X).
\]

Hence, the best schedule under the second scenario corresponds to the optimal solution of

\[
Z_2 := \min b(X) \quad \text{subject to} \quad a(X) \geq D; \quad X \subseteq J. \tag{3}
\]

Note that (3) again is a knapsack problem subject to a covering constraint. The optimal makespan for the scheduling problem equals \(\min\{D + Z_1, c(J) + Z_2\}\), that is, the better makespan found under the two scenarios.
3 Results on the off-line version

This section deduces a number of results from the knapsack characterization. We first prove
the ordinary NP-hardness of the problem. For this we use a reduction from PARTITION: We
are given $n$ positive integers $p_1, \ldots, p_n$ with $\sum_{j=1}^{n} p_j = 2P$ and we are asked whether there exists
a set $I \subseteq \{1, \ldots, n\}$ with $\sum_{j \in I} p_j = P$. We construct the following instance of the scheduling
problem: There are $n$ jobs, where job $J_j$ is specified by $a_j = 2p_j$ and $b_j = p_j$, and the critical
date is $D = 2P$. It is easily verified that the answer to PARTITION is YES if and only if
the optimal makespan in the scheduling instance equals $3P$. As a consequence, we find the
following theorem.

**Theorem 1** Makespan minimization for jobs with step-improving processing times and a com-
mon critical date on a single machine is NP-hard in the ordinary sense. ■

Recall that the knapsack problem subject to a covering constraint has as its input $n$ items
with weights $w_1, \ldots, w_n$ and profits $p_1, \ldots, p_n$ and a bound $P$. The goal is to find a subset of
the items that has total profit at least $P$ and that has the smallest possible weight. The knapsack
problem can be solved in pseudo-polynomial time by dynamic programming with running time
$O(n \sum_{j=1}^{n} w_j)$ or $O(n \sum_{j=1}^{n} p_j)$; see for instance Kellerer, Pferschy & Pisinger [2]. For our
knapsack problems in (2) and (3), this translates into a time complexity of $O(n \sum_{j=1}^{n} a_j)$.

**Theorem 2** Makespan minimization for jobs with step-improving processing times and a com-
mon critical date on a single machine can be solved in $O(n \sum_{j=1}^{n} a_j)$ time. ■

The knapsack problem subject to a covering constraint also possesses a fully polynomial time
approximation scheme (FPTAS); see for instance Kellerer, Pferschy & Pisinger [2]. This means
that for any $\varepsilon > 0$, there is an approximation algorithm that yields a feasible solution with
total weight at most $1 + \varepsilon$ times the optimal weight. The running time of this approximation
algorithm is polynomially bounded in the input size and in $1/\varepsilon$. This yields an FPTAS for
our knapsack problems in (2) and (3). In (2), the FPTAS gives us an approximation $Y^A$ of
the optimal solution $Y^*$ such that $c(Y^A) \leq (1 + \varepsilon)c(Y^*)$. Consequently, the corresponding
approximate makespan $D + c(Y^A)$ is at most $1 + \varepsilon$ times the optimal makespan $D + c(Y^*)$.
And in (3), the FPTAS gives us an approximation $X^A$ of the optimal solution $X^*$ with $b(X^A) \leq
(1 + \varepsilon)b(X^*)$. Consequently, the corresponding approximate makespan $c(J) + b(X^A)$ is at most
$1 + \varepsilon$ times the optimal makespan $c(J) + b(X^*)$. We summarize these observations in the
following theorem.

**Theorem 3** Makespan minimization for jobs with step-improving processing times and a com-
mon critical date on a single machine possesses an FPTAS. ■

If we apply other fast knapsack approximation algorithms to problems (2) and (3), we will
get corresponding approximation algorithms with corresponding approximation guarantees for
our scheduling problem in a straightforward way.
4 Results on the on-line version

In the on-line version of our scheduling problem, the jobs \( J_1, \ldots, J_n \) are revealed one by one. As soon as the on-line algorithm learns the values \( a_j \) and \( b_j \) for job \( J_j \), it must assign the job to an appropriate time interval; this decision is irrevocable and must not depend on later arriving jobs. We consider an extremely simple on-line algorithm \( \text{ON} \) that schedules the jobs in their given ordering \( J_1, \ldots, J_n \) and without introducing unnecessary idle time: Algorithm \( \text{ON} \) schedules every new job \( J_j \) after all the jobs \( J_1, \ldots, J_{j-1} \), so that the completion time of \( J_j \) becomes as small as possible.

For analyzing algorithm \( \text{ON} \), let \( k \) be the unique index with \( \sum_{j=1}^{k-1} a_j < D \leq \sum_{j=1}^{k} a_j \); this index \( k \) exists because of (1). Define \( X' = \{1, \ldots, k-1\} \) and \( Y' = \{k+1, \ldots, n\} \). Clearly, algorithm \( \text{ON} \) schedules the jobs \( J_j \) with \( j \in X' \) before \( D \) during the interval \( [0; a(X')] \), and it schedules the jobs \( J_j \) with \( j \in Y' \) after \( D \). For the pivotal job \( J_k \) there are two possibilities: Either it is executed during the interval \( [D; D+a_k-b_k] \) or during the interval \( [a(X'); a(X')+a_k] \). Algorithm \( \text{ON} \) chooses the option that minimizes the completion time of \( J_k \). If the first option is chosen, then \( D-b_k \leq a(X') \) holds, and the resulting makespan is

\[
C_{\text{ON}}^{\max} = D + c(Y' \cup \{k\}) \leq D + c(J). \tag{4}
\]

If the second option is chosen, then \( a(X') \leq D - b_k \) holds, and the resulting makespan equals

\[
C_{\text{ON}}^{\max} = a(X') + a_k + c(Y') \leq (D - b_k) + a_k + c(Y') \leq D + c(J). \tag{5}
\]

In either case we have \( C_{\text{ON}}^{\max} \leq D + c(J) \). Since \( D \) and \( c(J) \) both are trivial lower bounds on the optimal makespan, we arrive at the following theorem.

**Theorem 4** There exists an on-line algorithm for scheduling jobs with step-improving processing times and a common critical date on a single machine that always produces a schedule whose makespan is at most twice the optimal off-line makespan. \( \blacksquare \)

Finally, let us show that the ratio 2 in the statement of Theorem 4 is best possible for the on-line version. Suppose for the sake of contradiction that there exists an on-line algorithm \( A \) that always yields a makespan that is at most \( 2 - \varepsilon \) times the optimal off-line makespan for some \( \varepsilon \) with \( 0 < \varepsilon \leq 1 \). We confront \( A \) with the following instance with \( D \geq 2 \): The first job \( J_1 \) has \( a_1 = D \) and \( b_1 = D - \varepsilon \). Algorithm \( A \) either assigns \( J_1 \) to an interval \( [x; x+D] \) with \( x < D \) or to an interval \( [x; x+\varepsilon] \) with \( x \geq D \).

- In the first case, job \( J_2 \) arrives with \( a_2 = D \) and \( b_2 = 0 \). The optimal off-line makespan is \( D + \varepsilon \), whereas the on-line makespan is at least \( 2D + x \). Hence, the ratio is larger than \( 2 - \varepsilon \).

- In the second case, job \( J_2 \) arrives with \( a_2 = x + \varepsilon \) and \( b_2 = 0 \). The optimal off-line makespan equals \( x + 2\varepsilon \), and the on-line makespan is at least \( 2x + 2\varepsilon \). Since \( x \geq D \geq 2 \), the ratio is again larger than \( 2 - \varepsilon \).

In either case we get a contradiction. Hence, the ratio 2 is indeed best possible.
Acknowledgements

This work was partially conducted during a workshop on ‘Optimization with incomplete information’, which was held at Schloss Dagstuhl, Germany, January 16-21. Han Hoogeveen and Gerhard Woeginger are grateful to the organizers of the workshop and to the staff of Dagstuhl for providing a stimulating atmosphere.

T.C. Edwin Cheng was supported in part by The Hong Kong Polytechnic University under a grant from the Area of Strategic Development in China Business Services. Yong He acknowledges support by TRAPOYT, China, and the NSFC of China (10271110, 60021201). Han Hoogeveen and Gerhard Woeginger were partially supported by BSIK grant 03018 (BRICKS: Basic Research in Informatics for Creating the Knowledge Society). Gerhard Woeginger was partially supported by the Netherlands Organisation for Scientific Research, grant 639.033.403.

References
