Short Note on Scheduling on a Single Machine with one Non-availability Period

N. Megow* José Verschae†

January 5, 2009

Abstract

We consider scheduling on a single machine with one non-availability period to minimize the weighted sum of completion times. We provide a preemptive algorithm with an approximation ratio arbitrarily close to the Golden Ratio, $(1+\sqrt{5})/2+\varepsilon$, which improves on a previously best known 2-approximation. The non-preemptive version of the same algorithm yields a $(2+\varepsilon)$-approximation.

1 Introduction

We consider the problem of scheduling a set of jobs, $J = \{1, 2, \ldots n\}$, with nonnegative processing times p_j and weights w_j, for each $j \in J$, on a single machine. Due to maintenance or shift breaks, the machine is not available during the time intervals $[s_i, t_i]$, for $i \in \mathbb{N}$. Let C_j denote the completion time of job $j \in J$. The objective is to schedule all jobs preemptively on the machine using only time periods in which the machine is available, minimizing the sum of weighted completion times, $\sum_{j \in J} w_j C_j$.

This problem is known to be strongly $\mathcal{NP}$-hard [8]. If jobs are not allowed to be preempted an easy reduction from Partition shows that the problem with two or more non-available periods is not approximable, unless $\mathcal{P} = \mathcal{NP}$. The reduction is similar to the one in [9] for scheduling with fixed jobs. On the other hand, if all jobs have equal weight, a simple interchange argument shows that processing jobs preemptively in non-increasing order of processing times is optimal as it is in the setting with continuous machine availability [7]. However, in the problem setting with general job weights, natural greedy strategies perform arbitrarily bad. So far, no constant approximation algorithm has been found and therefore research focusses on special problem settings; a survey on scheduling with limited machine availability can be found in [6]. Particular interest has been devoted to the special case of scheduling on a machine that has exactly one non-available period $[s, t]$. In that case, the problem is weakly $\mathcal{NP}$-hard [4, 1, 5] in both, the preemptive and non-preemptive setting. The

*Max-Planck-Institut für Informatik, Saarbrücken, Germany, nmegow@mpi-inf.mpg.de.
Partially supported by the DFG Research Center Matheon in Berlin.
†Department of Mathematics, Technische Universität Berlin, Germany, verschae@math.tu-berlin.de.

currently best known results for both problems are 2-approximations [8, 3]; for a recent overview on previous results we refer to the same two papers.

In this note, we give a preemptive polynomial time algorithm for scheduling with a single non-available period that achieves an approximation ratio arbitrarily close to the Golden Ratio, namely $(1+\sqrt{5})/2+\varepsilon \approx 1.618+\varepsilon$ for any fixed $\varepsilon > 0$. More precisely, our algorithm yields this performance guarantee even in the more restricted resumable scheduling model, in which a job can be interrupted only by a machine breakdown and must resume processing immediately after this unavailable period. However, for simplicity we will generally talk about preemption. As a side result, the non-preemptive version of the same algorithm yields an approximation factor of $2+\varepsilon$.

2 The algorithm

The classical *Smith rule* [7] runs jobs in non-increasing order of their ratios of weight over processing time. It is the optimal scheduling algorithm for minimizing the weighted sum of completion times if the machine is continuously available. It has been observed in earlier works that in case that a machine becomes unavailable, it may perform "arbitrarily bad".

Example 1. *Consider two jobs with $p_1 = 1, w_1 = 1+\varepsilon$ and $p_2 = s, w_2 = s$ and a non-availability period $[s, s^2]$. The ratio between the solution of* Smith's rule *and an optimal solution is given by s which is unbounded.*

However, the performance of *Smith's rule* can be expressed dependent on the parameter $\delta := t/s$. In the example above we have a worst case $\delta = s$. Before showing such an upper bound on the performance guarantee, we introduce two lower bounds on the value $\textsc{Opt}$ of an optimal solution. For a job set $A \subseteq J$ let $S(A)$ denote the weighted sum of completion times for scheduling all jobs in A by *Smith's rule* starting at time 0 on a continuously available machine. Moreover, let J_1^* and J_2^* denote the sets of jobs that complete in the optimum schedule before and after the break, respectively. Clearly, within each set the jobs are scheduled in *Smith's* order. Thus, a lower bound on the optimal value $\textsc{Opt}$ is given by

$$\textsc{Opt} \ \geq\ S(J) \ +\ (t-s)\sum_{j\in J_2^*} w_j \,.$$

The second lower bound we will use is $\textsc{Opt} \ \geq\ t\sum_{j\in J_2^*} w_j$.

Lemma 1. *Scheduling jobs by Smith's rule yields an approximation ratio of δ for preemptive scheduling on a single machine with one non-availability period.*

Proof. Let J_1 and J_2 denote the sets of jobs completing before and after the break, respectively, in the schedule obtained by *Smith's rule*. At any completion time C_j for $j \in J$ in this schedule, we have finished at least as much weight as any other algorithm by that time. Let $x \in J_2$ be the first job completing after the break, i.e. $C_x := \min\{C_j \,|\, C_j > t \text{ for } j \in J\}$. Then it holds for the remaining weight

$$\sum_{j\in J_2\setminus\{x\}} w_j \ \leq\ \sum_{j: C_j^* > C_x} w_j \ \leq\ \sum_{j\in J_2^*} w_j \,,$$

where C_j^* denotes the completion time of job $j \in J$ in an optimal schedule.

The value of the schedule obtained by the algorithm is then

$$\begin{aligned}\sum_{j\in J} w_j C_j &= S(J) \;+\; (t-s)\sum_{j\in J_2} w_j \qquad (1)\\ &\le S(J) \;+\; (t-s)\sum_{j\in J_2^*} w_j \;+\; (t-s)w_x\\ &\le \textsc{Opt} \;+ (\delta-1)\, s\, w_x\,.\end{aligned}$$

Suppose that jobs are indexed in their oder of scheduling by *Smith's rule*. Then $\textsc{Opt} \ge S(J) \ge \sum_{j\in J} w_j \sum_{k\le j} p_k \ge w_x \sum_{k\le x} p_k$. Since x is the first job that completes after the break, we have $\sum_{k\le x} p_k \ge s$, and therefore, $s\, w_x \le \textsc{Opt}$. □

A simple adaptation of the proof gives immediately a slightly worse performance guarantee for non-preemptive scheduling in *Smith's* order.

Corollary 1. *Scheduling jobs non-preemptively in the order of Smith's rule yields an approximation ratio of $1+\delta$ on a single machine with one non-availability period.*

Proof. In addition to the value of the preemptive algorithm given in equality (1), we have to add the postponement of all jobs in set J_2 by at most p_x, that is, $p_x \sum_{j\in J_2} w_j$ which is a lower bound on the optimal value $\textsc{Opt} \ge S(J) \ge \sum_{j\in J} p_j \sum_{k\ge j} w_k$. □

While the schedule obtained by *Smith's rule* is close to optimal if there is a very small break, its performance deteriorates with increasing parameter δ. If the proportional break length is very large, i.e. δ is very large, then intuitively we have a knapsack problem and want to complete as much weight as possible before the break. In other words, we want to find a schedule that minimizes the total weight completed after the break. This corresponds to the deadline based scheduling problem $1||\sum w_j U_j$ with the common deadline $d_j = s$ for all $j \in J$. Here U_j denotes a binary variable indicating if job j has completed by its due date. Gens and Levner [2] provided a fully polynomial time approximation scheme (FPTAS) for this problem.

Lemma 2. *For any $\varepsilon > 0$, there is a polynomial time algorithm solving the scheduling problem with one non-available period with an approximation ratio*

$$1 + \frac{1}{\delta} + \varepsilon\,.$$

This is true even if jobs must not be preempted.

Proof. We solve the problem $1|d=s|\sum w_j U_j$ using the FPTAS in [2]. Let $J_1 = \{j \in J \,|\, U_j = 0\}$ and $J_2 = J \setminus J_1$. We schedule the jobs in J_1 by *Smith's rule* before the break and the jobs in J_2 accordingly after t.

By definition, the total weight completing after the break is bounded from above: $\sum_{j\in J_2} w_j \le (1+\varepsilon)\sum_{j\in J_2^*} w_j$. Thus, the algorithm yields a solution that is bounded by

$$\begin{aligned}
\sum_{j\in J} w_j C_j \quad &\leq \quad S(J_1) \; + \; S(J_2) \; + t \sum_{j \in J_2} w_j \\
&\leq \quad S(J_1) \; + \; S(J_2) \; + \; (1+\varepsilon)\, t \sum_{j\in J_2^*} w_j \\
&= \quad S(J_1) \; + \; S(J_2) \; + \; (t-s) \sum_{j\in J_2^*} w_j \; + \; \left(\frac{1}{\delta} + \varepsilon\right) t \sum_{j\in J_2^*} w_j \, .
\end{aligned}$$

Clearly, $S(J_1) + S(J_2) \leq S(J)$. Then the approximation guarantee follows directly from applying the two lower bounds above. □

The following example shows that this analysis is (almost) tight.

Example 2. *Consider two jobs such that $p_1 = s, w_1 = 1$ and $p_2 = \varepsilon, w_2 = 1$. An optimal algorithm for the deadline problem ignores the scheduling aspect ($\min \sum w_j C_j$) and schedules job 2 after the break whereas an optimal algorithm schedules it before the break. The ratio between both solution values tends to $1 + 1/\delta$ if ε goes to 0.*

Obviously, for small values of δ *Smith's rule* is preferably while for large δ the deadline focussed approach performs well. Combining these two algorithms yields the main result.

Theorem 1. *For any $\varepsilon > 0$, there is a polynomial time algorithm solving the preemptive scheduling problem on a single machine with one non-availability period that achieves an approximation ratio*

$$\frac{1+\sqrt{5}}{2} + \varepsilon \;\approx\; 1.618 + \varepsilon \, .$$

Proof. The algorithm runs either *Smith's rule* or the FPTAS with a parameter ε' whatever gives the minimum value. Given an $\varepsilon > 0$, set $\varepsilon' = 2\varepsilon(\sqrt{5}+\varepsilon)/(1+\sqrt{5}+2\varepsilon)$. Then the minimum of the performance guarantees proven in Lemma 1 and 2 gives in the worst case the claimed approximation ratio. □

Corollary 1 and Theorem 2 immediately imply a performance guarantee arbitrarily close to 2 for the non-preemptive version of the same procedure.

Corollary 2. *For any $\varepsilon > 0$, there is a polynomial time algorithm solving the non-preemptive scheduling problem on a single machine with one non-availability period that achieves an approximation ratio $2 + \varepsilon$.*

Acknowledgements

We thank José R. Correa for helpful discussions at the *Centro de Modelamiento Matematico* (CMM) in Santiago, Chile. We are grateful to CMM for supporting the research visit of the first author. The second author was supported by CONICYT (Chile) through grant Anillo en Redes ACT08.

References

[1] I. Adiri, J. Bruno, E. Frostig, and A. H. G. Rinnooy Kan. Single machine flow-time scheduling with a single breakdown. *Acta Informatica*, 26(7):679–696, 1989.

[2] G. Gens and E. Levner. Fast approximation algorithm for job sequencing with deadlines. *Discrete Applied Mathematics*, 3:313–318, 1981.

[3] I. Kacem. Approximation algorithm for the weighted flow-time minimization on a single machine with a fixed non-availability interval. *Computers & Industrial Engineering*, 54(3):401–410, 2008.

[4] C.-Y. Lee. Machine scheduling with an availability constraint. *Journal of Global Optimimization*, 9:395–416, 1996.

[5] C.-Y. Lee and S. D. Liman. Single machine flow-time scheduling with scheduled maintenance. *Acta Informatica*, 29(4):375–382, 1992.

[6] G. Schmidt. Scheduling with limited machine availability. *European Journal of Operational Research*, 121(1):1–15, 2000.

[7] W. E. Smith. Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3:59–66, 1956.

[8] G. Wang, H. Sun, and C. Chu. Preemptive scheduling with availability constraints to minimize total weighted completion times. *Annals of Operations Research*, 133:183–192, 2005.

[9] J. Yuan, Y. Lin, C. Ng, and T. Cheng. Approximability of single machine scheduling with fixed jobs to minimize total completion time. *European Journal of Operational Research*, 178(1):46–56, 2007.