Theoretical Computer Science 646 (2016) 40–48

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Semi-online scheduling on a single machine with unexpected breakdown

Imed Kacem [a], Hans Kellerer [b,*]

[a] LCOMS, Université de Lorraine, France
[b] Institut für Statistik und Operations Research, University of Graz, Austria

ARTICLE INFO

Article history:
Received 21 January 2016
Accepted 13 July 2016
Available online 29 July 2016
Communicated by T. Erlebach

Keywords:
Semi-online scheduling
Unexpected breakdown
Competitive analysis

ABSTRACT

In this paper, we consider a single machine scheduling problem where a breakdown period occurs in an online way. Two main objective functions are studied: the makespan and the maximum lateness. We propose two approximation algorithms for the makespan minimization for solving two variants of the problem: with different release dates or without release dates. We show that the competitive ratio of the two algorithms is 3/2 and that this bound is the best possible for the makespan minimization. For the maximum lateness minimization we propose a $\left(1+\sqrt{2}/2\right) \approx 1.70$-approximation algorithm capable to solve the problem with delivery times but no release dates. This ratio is tight for the proposed algorithm and allows us to establish a precise window for the best possible ratio, which belongs to $[3/2, 1+\sqrt{2}/2] \approx [1.50, 1.70]$.

1. Introduction

Scheduling with non-availability constraints has received considerable attention in research in the last decade. Most research focuses on the case that the non-availability period is known in advance. But this assumption is not practical in many cases. Unexpected machine breakdown is a major issue in manufacturing or parallel and distributed computing. This is a motivation to consider the following semi-online scheduling scenario.

We are given a single machine scheduling problem with a non-availability interval during which the machine cannot perform the processing of any job. A non-availability interval shall be due to some unexpected breakdown of the machine and will also be called *breakdown period*. Consequently, the starting time and the length of the interval is not known before a sequence of the jobs has been determined and this sequence cannot be changed after the breakdown of the machine. Our model is non-resumable, i.e. the job which has been interrupted by the breakdown period has to be restarted after the machine is recovered later.

We are addressing three problems under an online arrival of a non-availability period (or breakdown). In the first problem, jobs have no release times and delivery times. In the second problem, jobs have release times and in the third problem jobs have delivery times. In the first two problems our objective is to minimize the makespan, where for the third problem the objective is to minimize the maximum lateness. Without unexpected breakdown the corresponding problems are denoted in the three-field notation as $1|\cdot|C_{\max}$, as $1|r_j|C_{\max}$ and as $1|\cdot|L_{\max}$, respectively. There are no results on these three semi-online scheduling problems with the online machine unavailability setting. For each of the three problems we

* Corresponding author.
E-mail addresses: imed.kacem@univ-lorraine.fr (I. Kacem), hans.kellerer@uni-graz.at (H. Kellerer).

http://dx.doi.org/10.1016/j.tcs.2016.07.014
0304-3975/

will give an approximation algorithm and analyze its competitive behavior. We will compare the competitive ratio of each of the algorithms to the best possible performance ratio.

Without breakdown periods, problems $1|\cdot|C_{\max}$ and $1|r_j|C_{\max}$ can be solved easily by avoiding unnecessary idle times and problem $1|\cdot|L_{\max}$ is solved by sorting the jobs in Jackson's order, i.e., jobs sorted by non-increasing delivery times.

The corresponding offline problems with a non-availability interval are denoted by $1,h_1|\cdot|C_{max}$, $1,h_1|r_j|C_{max}$ and $1,h_1|\cdot|L_{max}$, respectively. A fully polynomial time approximation scheme (FPTAS) for $1,h_1|\cdot|C_{max}$ is delivered by any FPTAS for the partition problem. Kacem and Haouari [5] gave an FPTAS for $1,h_1|r_j|C_{max}$. Lee [7] studied Jackson's rule and proved that its deviation to the optimal value is less or equal to the maximum of the processing times for $1,h_1|\cdot|L_{max}$. Kacem [3], proposed a strongly FPTAS for this problem, which has been recently improved by Kacem et al. [4]. Other criteria, like sum of completion times, have been treated in e.g., [1] and [6]. Surveys on scheduling with non-availability intervals can be found in [8] or [9].

An analysis of a scheduling problem with breakdown period has been done in two special cases. Tan and He [10] studied the makespan minimization on two identical machines, with each machine having a single unavailability period that does not overlap with the unavailability period of the other machine. Huo et al. [2] studied the total weighted completion time minimization problem on a single machine with breakdown period under the additional assumption that the weight of each job is proportional to its processing time.

Our paper is organized as follows. In Section 2, we will give an exact problem definition and introduce further notations. Section 3 contains two algorithms with competitive ratio 3/2 for the makespan minimization problems and gives a lower bound of 3/2 for both problems, showing that the results in this section are best possible. In Section 4, we analyze the competitive ratio of the problem with delivery times. Finally, Section 5 contains some conclusions and open problems.

2. Problem definition and notations

The following semi-online scenario is considered: we are given a single machine and a set $N=\{1,\dots,n\}$ of n independent jobs j with processing times p_j, $j=1,\dots,n$. Let $P=\sum_{j=1}^{n}p_j$ denote the sum of processing times. All input data are assumed to be integers. Each job j may have release times r_j (heads) and delivery times q_j (tails), $j=1,\dots,n$. All these data are known in the beginning. Then, a sequence of jobs S has to be determined. After S is given, a machine non-availability (breakdown period) interval may occur which occupies a time interval $I=[T,U]$. Its starting time T and length $\triangle=U-T$ are not known beforehand.

The job that is affected by the breakdown period is called the *crossover* job. We assume a non-resumable scenario, i.e., the crossover job that cannot be completed by time T is restarted from scratch at time U. If not stated otherwise, let ℓ denote the job with the largest processing time and c the crossover job, respectively.

We investigate three single machine scheduling problems with this scenario. Problem $P1$ denotes the problem without release times neither delivery times, problem $P2$ denotes the problem with release times and problem $P3$ denotes the problem with delivery times. Let C_j denote the completion time of job j for a given schedule. Then, for problems $P1$ and $P2$ the objective is to minimize the makespan $C_{\max}=\max_{j=1,\dots,n}\{C_j\}$ and for $P3$ the objective is to minimize the maximum lateness $L_{\max}=\max_{j=1,\dots,n}\{C_j+q_j\}$.

Given a sequence of jobs S we denote by $S(I)$ the schedule obtained by sequence S and breakdown period I. Note that S^{∞} corresponds to the schedule without breakdown. The completion time of job j under schedule $S(I)$ is given by $C_j(S(I))$, the starting time of job j under schedule $S(I)$ is given by $s_j(S(I))$, the makespan of schedule $S(I)$ for breakdown period I is denoted as $C_{\max}(S(I))$, and the maximum lateness is denoted as $L_{\max}(S(I))$, respectively. We will often write simply C_j instead of $C_j(S(I))$, s_j instead of $s_j(S(I))$, $C_{\max}$ instead of $C_{\max}(S(I))$ and $L_{\max}$ instead of $L_{\max}(S(I))$ if it is clear from the context.

The value of a heuristic schedule for breakdown period I is written as $C^H(I)$. It is compared to the value of the optimal offline algorithm $C^*(I)$ where also the breakdown period is known in advance. If it is clear from the context, we write C^H instead of $C^H(I)$ and C^* instead of $C^*(I)$. We say that a heuristic has *competitive ratio* α if $C^H(I)/C^*(I)\leq\alpha$ holds for all possible job data and breakdown periods.

3. Minimizing the makespan

In this section we will present a 3/2-competitive algorithm for problem $P1$ without release times and a 3/2-competitive algorithm for problem $P2$ with release times. Indeed, the algorithm for $P2$ also works for $P1$, but for the sake of completeness we describe also the algorithm for $P1$ since it has linear running time and is very easy to analyze. Moreover, we will show that both algorithms are best possible.

Algorithm $A1$ for problem P1 is very simple. It puts job ℓ on the first position and assigns the remaining jobs in an arbitrary sequence.

Theorem 1. *Algorithm $A1$ is a 3/2-competitive algorithm for problem $P1$.*

Proof. Without loss of generality the breakdown period starts before time P, since otherwise algorithm $A1$ is optimal. The makespan of Algorithm $A1$ is bounded from above by adding the processing time of the crossover job to the total processing