Scheduling jobs with release dates on a single machine subject to an online unavailability period

Rima Adjal[1], Pierre Laroche[1], Christophe Rapine[1]

Université de lorraine, laboratoire LCOMS, Metz, France
{rima.adjal, pierre.laroche, christophe.rapine}@univ-lorraine.fr

Keywords : *Online scheduling; Competitive algorithm; Unavailability period; Release dates*

1 Introduction

In this paper, we study a scheduling problem where an unavailability period occurs online. In the literature, most researches assume that machine unavailability periods are known in advance. However, this assumption is not verified in many cases, for instance if a machine experiences a sudden breakdown. In our problem, we have a set of n jobs to schedule on a single resource. Each job is associated with a release date and a processing time. All the characteristics of the jobs are known by the scheduler. We have to determine a sequence of the jobs minimizing the makespan, *i.e*, the completion time of the last job. This problem is challenging because an unavailability period may occur on the fly. Hence, the scheduler is unaware of when the unavailability period will start and how long it will last. Moreover, the sequence decided by the scheduler can not be changed later and the jobs are non-resumable. This means that if the processing of a job is interrupted by the occurrence of the unavailability period, the job must restart from the beginning once the machine is available.

When all the information is known in advance, [3] demonstrated that minimizing the makespan is NP-hard even with no releases dates. In semi-online setting, where some information is not known in advance, two categories of research papers can be distinguished. The first [4, 1] considers that the unavailability periods are known in advance but jobs arrive online. The second category of articles [2, 5] assumes that the jobs are known in advance, but the unavailability period is an unexpected breakdown. Our problem falls into the second category and was addressed in [2], where the authors consider the same problem but assume a null duration for the unavailability period.

In our contribution, we extend the study of [2] by considering that the unavailability period can have any duration. This assumption is more realistic, as in practical scenarios, when a breakdown occurs, it generally requires a certain amount of time to restore the machine. Specifically, we study two semi-online scheduling problems. In the first problem, the starting time and the duration of the unavailability period are not known in advance. In contrast, in the second problem, only the starting time of the unavailability period is not known in advance but the duration is part of the instance. Indeed, when the machine is subject to a routine repair, *e.g.* the replacement of a defective component, the duration of the unavailability period may be predictable.

2 Best possible competitive algorithms

For this semi-online problem, we are interested in finding competitive algorithms. An algorithm is said to be α-competitive if for any instance it delivers a solution whose value is within a factor α of an optimal offline algorithm. If no semi-online algorithm can achieve a competitive ratio smaller than γ, then γ is a lower bound for the competitive ratio of this problem. An

online algorithm whose competitive ratio matches the lower bound for the problem is called best possible.

In the case where the starting time and the duration of the unavailability period are not known in advance, we provide an instance such that the competitive ratio of any sequence for the jobs is greater than 2. Then, we prove that any list scheduling algorithm achieves the best possible competitive ratio for the problem. Recall that the principle of the list scheduling algorithms is to greedily allocate the jobs to the machine. Specifically, based on a priority list of the jobs, each time the machine becomes available, the algorithm schedules the first available job in the list.

Lemma 1 (Online duration of the unavailability period) *If the duration and the starting time of the unavailability period are not known in advance, any list scheduling algorithm has a competitive ratio of 2, and this is the best possible competitive ratio.*

In [2], the authors proposed a $\frac{3}{2}-$competitive algorithm when the duration of the unavailability period is null. This result indicates that the competitive ratio may be less than 2 if the duration of the unavailability period is known in advance. Intuitively, if the duration can be large, one must choose a sequence which is optimal without unavailability periods. Otherwise, the competitive ratio can be arbitrarily large. In contrast, if we know in advance that the duration is short, we have room for choosing a sequence which may not be optimal without unavailability, but performing better if an unavailability period occurs. This motives us to consider the problem where the duration of the unavailability period is part of instance, but can take any value.

In this case, we prove that the best possible ratio that an algorithm can achieve is the golden ratio, $\phi = \frac{1+\sqrt{5}}{2}$. We propose an algorithm, denoted LLS, which is structured in two steps. In the first step, we only schedule a set composed of the largest jobs. This set can be either empty, or include only the largest job, or include the two largest jobs, depending on the characteristics of the instance. These jobs are scheduled using any list scheduling algorithm. In the second step, we schedule the remaining jobs using again any list scheduling algorithm. The schedule determined in the first step for the set of large jobs is fixed.

Lemma 2 (Offline duration of the unavailability period) *When the duration of the unavailability period is known in advance, LLS algorithm achieves the best possible competitive ratio of* $\phi = \frac{1+\sqrt{5}}{2}$.

3 Perspectives

In [5], the authors studied the same semi-online problem, but with delivery times rather than release dates. For this problem, they proposed a ϕ-competitive algorithm. In future work, it would be interesting to consider both release dates and delivery times for jobs.

References

[1] Qiang Gao, Ganggang Li, and Xiwen Lu. Online and semi-online scheduling to minimize makespan on single machine with an availability constraint. *Discrete Mathematics, Algorithms and Applications*, 7(03):1550021, 2015.

[2] Imed Kacem and Hans Kellerer. Semi-online scheduling on a single machine with unexpected breakdown. *Theoretical Computer Science*, 646:40–48, 2016.

[3] Chung-Yee Lee. Machine scheduling with an availability constraint. *Journal of global optimization*, 9(3-4):395–416, 1996.

[4] Zhiyi Tan and Yong He. Optimal online algorithm for scheduling on two identical machines with machine availability constraints. *Information Processing Letters*, 83(6):323–329, 2002.

[5] Ji Tian, Yan Zhou, and Ruyan Fu. An improved semi-online algorithm for scheduling on a single machine with unexpected breakdown. *Journal of Combinatorial Optimization*, 40(1):170–180, 2020.