On-line minimization of makespan for single batching machine scheduling problems

Ridouard Frédéric[1], Richard Pascal[1], and Martineau Patrick[2]

[1] LISI laboratory, France
e-mail: {frederic.ridouard, pascal.richard}@ensma.fr
[2] Polytech'Tours - Dpt of Computer Science, France
e-mail: pmartineau@univ-tours.fr

Keywords. single parallel batching machines, on-line scheduling, makespan, competitive analysis.

1 Introduction

A *batching machine* (or batch processing machine) is a machine that can process up to b jobs simultaneously. The jobs that are processed together form a batch. Two kinds of batching machines can be defined: in a *serial batching machine* the processing time of a batch is equal to the sum of processing times of jobs belonging to it; in a *parallel batching machine*, the processing time of a batch is the maximum of the processing times of jobs belonging to it. Next, we only focus on a parallel batching machine. In particular, such machines are used in semi-conductor and pharmaceutical industries.
Each instance has n jobs and each job J_i ($i \in \{1, \ldots n\}$) has a processing time p_i and a release date r_i. A job cannot start before its release date and the completion time C_i of each job J_i in a batch is equals to the completion time of the batch itself. Preemption is not allowed. A scheduling algorithm is said to be *conservative* if it is not allowed to postpone the start of an available job. We consider the scheduling problem in the on-line setting where jobs are released over time: characteristics of a job is known when it arrives in the system and the number of jobs is known when the last one arrives. Finally, it can be established that $b \geq n$, we deal with unbounded batch sizes; the batch size is said bounded otherwise.
The objective is to minimize the makespan of the schedule, that is the completion time of the last scheduled batch.
To study on-line algorithms, we shall use competitive analysis. This approach compares on-line algorithms to an optimal *clairvoyant* algorithm: *the adversary.* A good adversary defines instances of problems so that the on-line algorithm achieves its worst-case performance. An algorithm that minimizes a measure of performance is c-competitive if the value obtained by the on-line algorithm is less than or equal to c times the optimal value obtained by the adversary. We also say that c is the performance guarantee of the on-line algorithm. An algorithm is said *competitive* if there exists a constant c so that it is c-competitive. More formally, given an on-line algorithm A. Let I be an instance. Then, $\sigma_A(I)$ is the makespan obtained by A and $\sigma^*(I)$ is the makespan obtained by the optimal clairvoyant algorithm, then A is c-competitive if there exists a constant c so that $\sigma_A(I) \leq c\sigma^*(I)$. The competitive ratio c_A of the algorithm A is the worst-case ratio while considering any instance

I: $c_A = \sup_{any I} \frac{\sigma_A(I)}{\sigma^*(I)}$. The competitive ratio of an algorithm A is greater than or equal to 1. If $c_A = 1$, then A is an optimal algorithm.
The scheduling of batch processing machines has not been studied until recently by researchers in deterministic scheduling. For the off-line problem, Lee and Uzsoy (1999) provide an polynomial algorithm to minimize the makespan for unbounded batch sizes. When $b < n$, then the problem has been proved NP-hard by Liu and Yu (2000). In the on-line setting, these authors also proposed a simple greedy algorithm leading to a performance guarantee of 2 for the general bounded problem. But for non-conservative algorithms, Zhang and al (2001) establish a general lower bound of $1+\alpha$. In the remainder, we note $\alpha = \frac{-1+\sqrt{5}}{2}$. This result holds for bounded and unbounded batch sizes. Furthermore, when the size of the batch is unbounded, Zhang and al (2001) establish the algorithm H^∞ with a tight performance guarantee of $1+\alpha$.
In the following section, we present the results when the processing times of any task are equals. Section 3 contains outcomes in the general unbounded case.

2 The problem $1|p-batch, r_i, p_i = p, b = \infty|C_{max}$

We now consider the special case where the processing of tasks are equals. In Richard et al (2003), we presented the αH algorithm defined as follows: at any time, let $U(t)$ be the set of available unscheduled jobs at time t; when the machine is idle and some unscheduled jobs are available, let J_i be an available job, then the next batch is not scheduled before $r_i + \alpha p_i$ and then schedule available unscheduled jobs as many as possible as a batch. Richard and al (2003) prove that αH is a best possible deterministic algorithm ($(1+\alpha)$-competitive).
We next propose another best possible algorithm called $\alpha H2$, that is a slight modification of αH: at any time t when the machine is idle and some unscheduled jobs are available, let J_i be the available job such as $r_i + \alpha p_i = \min_{J_j \in U(t)}(r_j + \alpha p_j)$, then the next batch schedules jobs as many as possible at a time t' after $r_i + \alpha p_i$. A slightly modification of the proof presented for αH in Richard and al (2003) allows to prove that $\alpha H2$ is $(1+\alpha)$-competitive.
To summarize, at any time, αH waits the time $\max_{J_j \in U(t)}(r_j + \alpha p_j)$ and schedules the set of available unscheduled jobs as many as possible. $\alpha H2$ waits $\min_{J_j \in U(t)}(r_j + \alpha p_j)$ and schedules the jobs of $U(t)$ as a batch as many as possible at a time t. Now if at any time, an on-line algorithm A waits a time a such that $a \in [\min_{J_j \in U(t)}(r_j + \alpha p_j), \max_{J_j \in U(t)}(r_j + \alpha p_j)]$ then A is still a best possible algorithm for the problem $1|p-batch, r_i, p_i = p, b = \infty|C_{max}$.

3 The problem $1|p-batch, r_i, b = \infty|C_{max}$

In this section, we deal with the case where the capacity of the machines b is sufficiently large to process all jobs simultaneously in a single batch. But, jobs have non-equal processing times. Firstly, we prove that αH and $\alpha H2$ are 2-competitive. Secondly, we present a best possible algorithm, called αH^∞ that inserts less idle times than H^∞ (Zhang et al (2001)).

3.1 The competitive ratio of αH and $\alpha H2$

The competitive ratios of αH and $\alpha H2$ are not better than 2. We use an adversary argument:

- For αH, we just need to study the instance I such that $J_1 = (r_1 = 0, p_1 = p), J_2 = (r_2 = \alpha p, p_2 = 1), J_3 = (r_3 = \alpha p + \alpha, p_3 = 1), \ldots, J_{\lfloor \alpha p \rfloor - 1} = (r_{\lfloor \alpha p \rfloor - 1} = \alpha p + (\lfloor \alpha p \rfloor - 1)\alpha,\ p_{\lfloor \alpha p \rfloor - 1} = 1)$ and $J_{\lfloor \alpha p \rfloor} = (r_{\lfloor \alpha p \rfloor} = p,\ p_{\lfloor \alpha p \rfloor} = 1)$, where p is a very large integer. αH schedules J_1 as the first batch and $\{J_2, \ldots, J_{\lfloor \alpha p \rfloor}\}$ as second batch. The optimal algorithm of Lee and Uzsoy (1999) schedules $\{J_1, \ldots, J_{\lfloor \alpha p \rfloor - 1}\}$ as the first batch and $J_{\lfloor \alpha p \rfloor}$ as second batch. Consequently, after some arithmetical calculations, $\sigma^* = 1 + p$ and $\sigma_{\alpha H} = \alpha p + \lfloor \alpha p \rfloor \alpha + p + 1$ and $c_{\alpha H} \geq 2$.
- For $\alpha H2$, if we study the instance, $J_1 = (r_1 = 0, p_1 = 1), J_2 = (r_2 = 0, p_2 = p), J_3 = (r_3 = \alpha + \epsilon, p_3 = p)$, where p is still a very large integer, then $\sigma^* = \alpha + \epsilon + p$, $\sigma_{\alpha H2} = \alpha + 2p$ and $c_{\alpha H2} \geq 2$.

To conclude, αH and $\alpha H2$ are no better than 2-competitive for the problem $1|p-batch, r_i, b = \infty|C_{max}$.

3.2 The αH^∞ algorithm

We will give a description of the algorithm αH^∞. Note that $U(t)$ is the set of available unscheduled jobs at the time t.

Algorithm αH^∞**:**
STEP **0**. Set $t = 0$.
STEP **1**. Find job $J_k \in U(t)$ such that $p_k = \max\{p_j \mid J_j \in U(t)\}$.
Let $\gamma = r_k + \alpha p_k$ and $s = \max\{t, \gamma\}$.
STEP **2**. In the time interval *[t,s]*, whenever a new job J_h arrives, at the time *t'* and if $p_h > p_k$ then $k = h$, $\gamma = r_h + \alpha p_h$, $t = t'$, $s = \max\{t, \gamma\}$.
To conclude in any case, $U(t) = U(t) \cup \{J_h\}$.
STEP **3**. At time s, we schedule in a single batch, the set of available unscheduled jobs, $U(s)$. If some new jobs arrive during the execution of the batch then let $t = s + p_k$ else let t be the arrival time of such a job. Go to *STEP 1*.

Given an instance, we assume that αH^∞ generates m batches in total. We index these batches in non-decreasing order of their completion times. The k^{th} batch is noted B_k and let s_k its starting time. For convenience, in batch k, $J_{(k)}$ denotes the longest job of B_k (determinated by *Step 1*). Let $p_{(k)}$ and $r_{(k)}$ be the processing time and the arrival time of $J_{(k)}$. Note that a batch B_k starts execution either at time $r_{(k)} + \alpha p_{(k)}$ or immediately after the execution of the batch B_{k-1} is finished. If it starts at time $r_{(k)} + \alpha p_{(k)}$, it is a *regular* batch else a *delayed* batch. Furthermore, by definition of αH^∞, $p_{(k)}$ is the processing time of batch B_k.
Now, we present the principle of the proof for the $(1 + \alpha)$-competitivity of αH^∞. In Ridouard (2003), a whole demonstration is established.

Theorem αH^{∞} is $(1+\alpha)$-competitive.
Proof:

- For σ^*, we just consider that $J_{(m)}$ must be processed by the optimal algorithm; therefore $\sigma^* \geq r_{(m)} + p_{(m)}$. Futhermore, we can prove that $r_{(m)}$ occurs between the start and the completion of B_{m-1}.
- To calculate $\sigma_{\alpha H^{\infty}}$, we must determine the makespan achieved by αH^{∞}. We recall that the makespan is the completion time of B_m. We have only consider in the schedule, the last sequence of batches without idle-time. Let $B_k, \ldots, B_m$ be such a sequence. B_k is regular, thus its starting time is $s_k = r_{(k)} + \alpha p_{(k)}$. Therefore we conclude calculating $\sigma_{\alpha H^{\infty}}$ adding the processing times of B_k and of the batches after B_k (because there are delayed). Hence we have $\sigma_{\alpha H^{\infty}} = r_{(k)} + \alpha p_{(k)} + \sum_{i=k}^{m} p_{(i)}$.
- We can study all possible inequalities between $p_{(k)}$, $p_{(m-1)}$ and $p_{(m)}$ to obtain the expected competitive ratio.

To conclude, αH^{∞} is a best possible algorithm for the problem $1|p-batch, r_i, b = \infty|C_{max}$ and it is easy to show to αH^{∞} introduce less idle-times than H^{∞}.

4 Conclusion and perspectives

An optimal polynomial off-line algorithm is known for the $1|p-batch, r_i, b = \infty|C_{max}$ problem. We have studied the on-line scheduling problem of a single batching machine to minimize the makespan. For this problem, with bounded or unbounded batch sizes, the lower bound of the competitive ratio of on-line deterministic algorithm is $1+\alpha$. But conservative algorithms cannot be better than 2-competitive. Now, for the problem $1|p-batch, r_i, b = \infty|C_{max}$, if for each instance we have equal processing times then we propose several best possible on-line algorithms such as αH or $\alpha H2$. For non-equal processing times, we have proposed an on-line algorithm αH^{∞} that inserts less idle-time than H^{∞} but with the same performance guarantee.
Works are remaining for the general bounded problem since the best known algorithm is 2-competitive whereas the best known lower bound is equal to $1+\alpha$. In Zhang et al (2001) is proposed an algorithm and they only *conjecture* that it is $1+\alpha$ competitive.

References

LEE, C.Y. and UZSOY, R. (1999): Minimizing makespan on a single batch processing machine with dynamic job arrivals. *In INT. J. PROD. RES., VOL 37, No. 1*, 219–236

LIU, Z. and YU, W. (2000): Scheduling one batch processor subject to job release dates. *Discrete Applied Mathematics*, *105*, 129–136

RICHARD, P. and RIDOUARD, F. and MARTINEAU, P. (2003): On-line scheduling on a single batching machine to minimize the makespan. *Proc. Industrial Engineering and Production Management, (IEPM'03), Porto, 2003.*

RIDOUARD, F.(2003): Real-Time scheduling: a single batching machine. *Master thesis, University of Poitiers*, (in french).

ZHANG, G. and CAI, X. and WONG, C.K. (2001) : On-line algorithms for minimizing makespan on batch processing machines