Information Processing Letters 109 (2009) 1125–1128

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Optimal online-list batch scheduling

Jacob Jan Paulus [a], Deshi Ye [b,*], Guochuan Zhang [b]

[a] University of Twente, PO Box 217, 7500AE Enschede, The Netherlands
[b] College of Computer Science, Zhejiang University, Hangzhou 310027, China

ARTICLE INFO

Article history:
Received 22 February 2009
Received in revised form 29 June 2009
Accepted 15 July 2009
Available online 17 July 2009
Communicated by F.Y.L. Chin

Keywords:
Batch scheduling
Online algorithms
Competitive analysis

ABSTRACT

We consider the online-list batch scheduling problem. Jobs arrive one by one and have to be assigned upon arrival to a scheduled batch such that the makespan is minimized. Each batch can accommodate up to B jobs. We give a complete classification of the tractability of this online problem.

1. Introduction

In this paper we consider online-list scheduling on one batching machine. A set of jobs has to be scheduled on the batching machine which processes jobs parallel in batches. Each job j is characterized by its processing time p_j. The batching machine has capacity B, which gives the maximum number of jobs that can be scheduled in a single batch. The processing time of a batch must be larger than or equal to the maximum processing time of all jobs in the batch. The objective is to minimize the makespan, i.e. the completion time of the last batch. Note that the order of the batches is of no importance, it does not influence the objective function, only the processing times of the created batches are of interest. The above type of batching is referred to as parallel batching or *p-batch*, contrary to an *s-batch* which processes jobs sequential with a start-up time for each batch [3]. The model of parallel batching finds applications in, for example, scheduling burn-in ovens used for circuit board manufacturing [8].

In the *online-list* version of this problem jobs from a sequence σ are presented one by one to the scheduler. Upon arrival, the processing time of the job becomes known and the job has to be assigned immediately and irrevocably to a batch. The job is either included in a non-full existing batch (i.e. a batch with less than B jobs assigned to it) or put in a new batch (created for this job). The processing time of each batch has to be fixed upon its creation, and a job j can only be assigned to a batch with processing time at least p_j.

In the corresponding offline problem, the scheduler has all jobs available at $t = 0$, and an optimal offline schedule can be found by applying the algorithm known as FBLPT (Full Batch Longest Processing Time) [7]. This algorithm schedules the B jobs with largest processing time in the first batch, the next B jobs with largest processing time in a second batch, etc.

For a sequence of jobs σ, we denote the makespan of the optimal offline schedule by $C^*(\sigma)$ and the makespan of the online schedule created by an online Algorithm A by $C^A(\sigma)$. The performance of an online Algorithm A is measured by its competitive ratio defined as $\sup_\sigma\{C^A(\sigma)/C^*(\sigma)\}$. An online algorithm is called optimal if it has the smallest possible competitive ratio among all online algorithms.

In the literature a number of related problems have been studied. In [2] the *online-list* batching problem with

* Corresponding author.
E-mail address: yedeshi@zju.edu.cn (D. Ye).

0020-0190/$ – see front matter
doi:10.1016/j.ipl.2009.07.006

the objective to minimize the average flow time is studied and an optimal 4-competitive algorithm is given. The considered model allows only to schedule the next job in the last created batch or to create a new batch, and the capacity of the batching machine is unlimited. Much more work has been done on the *online-time* version of the batching problem to minimize the makespan, where jobs arrive at their release dates. For the case of unlimited batch capacity, optimal $(\sqrt{5}+1)/2$-competitive algorithms are independently given in [6] and [12] and generalized in [10]. The tractability of the online-time problem has not yet been resolved in case of bounded batch capacity. The best known online algorithm is 2-competitive for any capacity B [12,11]. Only for the case $B=2$ a better algorithm is presented in [11], which is 7/4-competitive. We refer to [9] for the more general problem with job families and a more extensive overview of the results on the online-time model.

The algorithms designed in this paper use what is called the "doubling" strategy. The idea behind this strategy is to use geometrically increasing batch processing times to approximate the optimal offline solution. However, as mentioned in [4], the increase is not always done by a factor of 2. An example of this principle is found in online algorithms for the problem of *searching a line in the plane* [1]. A short overview of other online problems solved with the "doubling" strategy is given in [4].

If the batch capacity is infinite, the online-list batch scheduling problem is the same as the *online bidding problem*, in which an optimal 4-competitive algorithm by "doubling" strategy and an optimal e-competitive randomized algorithm are given in [5]. The online bidding problem is stated as follows: An online player submits *bids* b_i until it submits a bid larger than or equal to a threshold $T \geqslant 1$. The online player pays the sum of all submitted bids. It is not difficult to see that the two problems are equivalent. The *online scheduler* determines a sequence of batch lengths $b_1 < b_2 < \cdots < b_{k-1} < b_k$ such that $b_{k-1} < p_{\max} \leqslant b_k$, and has makespan $\sum_{i=1}^{k} b_i$. Since the batch capacity is unlimited, no reasonable algorithm creates a batch for an arriving job that can be included in an existing batch. The *online bidder* determines a sequence of bids $b_1 < b_2 < \cdots < b_{k-1} < b_k$ such that $b_{k-1} < T \leqslant b_k$, and pays $\sum_{i=1}^{k} b_i$. Again, no reasonable bidder submits a bid smaller than the previous bid. The offline costs are $p_{\max}$ and T for the scheduling and bidding problem, respectively.

If the batch capacity is bounded the problem becomes different. In the next section we will give a full picture for the bounded case.

2. Bounded capacity

Consider online-list batch scheduling with a fixed bounded capacity B for each batch. To obtain an optimal algorithm, we have to use a different growth rate in batch lengths (different for each capacity B). Concretely we propose the following online strategy. If $B \leqslant 3$ we schedule the jobs greedily. If $B \geqslant 4$, we use a growth rate of z_B (to be defined below) in batch lengths instead of 2.

Table 1
Values of z_B and ρ_B.

B	2	3	4	5	6	7	8	∞
z_B	1	1	1.5214	1.7614	1.8768	1.9349	1.9651	2
ρ_B	2	3	3.6107	3.8344	3.9254	3.9651	3.9833	4

Algorithm A^B

If $B \leqslant 3$, then always schedule a job j with processing time p_j in a non-full batch of length at least p_j. If such a batch does not exist, create a batch with length p_j at the end of the current schedule.

If $B \geqslant 4$, then schedule a job j with processing time $p_j \in (z_B^{i-1}, z_B^i]$ in a non-full batch of length z_B^i. If there is no such a batch, create one at the end of the current schedule.

We choose z_B such that

$$z_B = \operatorname*{argmin}_{x \geqslant 1} \left\{ x + 1 + \frac{1}{x} + \frac{1}{x^2} + \cdots + \frac{1}{x^{B-2}} \right\}, \tag{1}$$

and show that the competitive ratio of Algorithm A^B is

$$\rho_B = \min_{x \geqslant 1} \left\{ x + 1 + \frac{1}{x} + \frac{1}{x^2} + \cdots + \frac{1}{x^{B-2}} \right\}.$$

Before we determine the competitive ratio of A^B, we point out that z_B and ρ_B are unique for a given B, i.e. the function in (1) is convex for $x \geqslant 1$. To indicate what kind of growth rates and competitive ratios we are dealing with, we display in Table 1 the values of z_B and ρ_B for some specific values of B.

Theorem 1. *For online-list batch scheduling with capacity B, Algorithm A^B is ρ_B-competitive.*

Proof. For $B \leqslant 3$, we know that each batch in the online schedule contains at least one job with processing time equal to the length of the batch. So, by a load argument the offline makespan cannot be less than $\frac{1}{B}$ times the online makespan. Thus, Algorithm A^B is B-competitive.

Consider $B \geqslant 4$. Let σ be a worst-case instance for Algorithm A^B. By normalizing the job lengths let z_B^1 be the smallest online batch and n such that z_B^n is the largest online batch. Thus the online schedule consists of batches with lengths in $\{z_B^1, z_B^2, \ldots, z_B^n\}$. Note that for each i there is at most one non-full batch of length z_B^i. In the following we derive three properties which we may assume for worst-case instance σ:

(1) Each job j scheduled in a batch of length z_B^i has length $p_j = z_B^{i-1} + \epsilon$, where $\epsilon > 0$ is arbitrary small. Decreasing the job lengths in a batch of length z_B^i to $z_B^{i-1} + \epsilon$ does not affect the online makespan and may decreases the offline makespan. All what follows is subject to the small value ϵ in the construction of σ^{adv}, but by choosing ϵ appropriately small it does not affect the outcome. So, we choose ϵ small enough and leave it from the remainder of the analysis.

(2) For each batch length z_B^i, there is at most one batch. If the worst case instance has more than B jobs in batches of length z_B^i, then B of these jobs are together in a batch in both the online and offline schedule. Due to property (1), removing these B jobs causes a decrease of z_B^i in the online makespan and a decrease of z_B^{i-1} in the optimal offline makespan. Let $\tilde{\sigma}$ be the instance resulting by removal of these B jobs from σ. Since σ is a worst-case instance we have

$$\frac{C^A(\sigma)}{C^*(\sigma)} \geqslant \frac{C^A(\tilde{\sigma})}{C^*(\tilde{\sigma})} = \frac{C^A(\sigma) - z_B^i}{C^*(\sigma) - z_B^{i-1}}.$$

This implies that $z_B \cdot C^*(\sigma) \geqslant C^A(\sigma)$, and that the algorithm has competitive ratio of at most $z_B < \rho_B$. So, we only have to consider instances which result in an online schedule with for each batch length z_B^i at most one batch.

(3) Each batch consists of only one job. If the only batch of length z_B^i contains k jobs with $2 \leqslant k \leqslant B$, then we can remove $k-1$ of these jobs without decreasing the makespan.

By the above properties of σ, we get that the cumulative length of the B largest batches in the online schedule is at most $z_B^n + z_B^{n-1} + \cdots + z_B^{n-B+1}$. By (1) this is equal to $\rho_B \cdot z_B^{n-1}$, that is ρ_B times the largest offline batch. As a result the B largest batches in the online schedule have a cumulative length of at most ρ_B times the largest batch in the optimal offline schedule. By the repetition of this argument the next B largest batches in the online schedule have a cumulative length of at most ρ_B times the second largest batch in the optimal offline schedule, etc. Thus, Algorithm A^B is ρ_B-competitive. □

It remains to show the upper bound is best possible. To this end, we consider a special job sequence:

Definition. The infinite job sequence σ^{adv} has $p_1 = 1$ and each following job has length equal to the last created batch by the online algorithm plus a small amount $\epsilon > 0$. The subsequence σ_k^{adv} is given by the first k jobs of sequence σ^{adv}. The next theorem, which is the main contribution of this paper, uses the structure of the FBLPT solution to prove that Algorithm A^B is optimal. It also implies the infinite capacity case by letting B go to infinity.

Theorem 2. *For online-list batch scheduling with capacity B, no online algorithm is $(\rho_B - \delta)$-competitive, for any $\delta > 0$ and B.*

Proof. Suppose there exists a $(\rho_B - \delta)$-competitive algorithm A, which is presented with job sequence σ^{adv}. Recall that due to the construction of σ^{adv} each job has its own batch in the online schedule, regardless of the online algorithm used. For simplicity we denote the optimal offline makespan by C_k^* and the online makespan of Algorithm A by C_k^A for the subsequence σ_k^{adv}. All what follows is subject to the small value ϵ in the construction of σ^{adv}, but by choosing ϵ appropriately small it does not affect the outcome. So, we choose ϵ small enough and ignore it from the remainder of the analysis.

The optimal offline and online makespan can be expressed, respectively, by

$$\begin{aligned} C_k^* &= p_k + p_{k-B} + p_{k-2B} + \cdots, \\ C_k^A &= p_{k+1} + p_k + \cdots + p_2 \\ &= C_{k+1}^* + C_k^* + \cdots + C_{k-B+2}^* - C_1^*. \end{aligned}$$

Let $\gamma_k = C_{k+1}^*/C_k^*$, be the ratio between the value of two subsequent optimal offline makespans. Obviously the optimal offline makespan increases in k, thus $\gamma_k \geqslant 1$. By Algorithm A being $(\rho_B - \delta)$-competitive, we have

$$\begin{aligned} \frac{C_k^A}{C_k^*} &= \frac{C_{k+1}^* + C_k^* + \cdots + C_{k-B+2}^* - C_1^*}{C_k^*} \\ &= \gamma_k + 1 + \frac{1}{\gamma_{k-1}} + \frac{1}{\gamma_{k-1}\gamma_{k-2}} + \cdots \\ &\quad + \frac{1}{\gamma_{k-1}\gamma_{k-2}\cdots\gamma_{k-B+2}} - \frac{C_1^*}{C_k^*} \\ &\leqslant \rho_B - \delta. \end{aligned}$$

We assume k to be large enough such that $\frac{C_1^*}{C_k^*} \leqslant \frac{\delta}{2}$, thus

$$\gamma_k + 1 + \frac{1}{\gamma_{k-1}} + \frac{1}{\gamma_{k-1}\gamma_{k-2}} + \cdots + \frac{1}{\gamma_{k-1}\gamma_{k-2}\cdots\gamma_{k-B+2}} \leqslant \rho_B - \frac{\delta}{2}. \tag{2}$$

In the remainder of this proof we show that (2) and $\gamma_k \geqslant 1$ are contradicting. To obtain this contradiction, we introduce $\tilde{\gamma}_k := \max\{\gamma_{k-1}, \ldots, \gamma_{k-B+2}\}$ and show that $\gamma_k < \tilde{\gamma}_k$ and $\tilde{\gamma}_k$ decreases to below 1.

By (2) and the definition of $\tilde{\gamma}_k$ we have

$$\begin{aligned} \rho_B - \frac{\delta}{2} &\geqslant \gamma_k + 1 + \frac{1}{\gamma_{k-1}} + \frac{1}{\gamma_{k-1}\gamma_{k-2}} + \cdots \\ &\quad + \frac{1}{\gamma_{k-1}\gamma_{k-2}\cdots\gamma_{k-B+2}} \\ &\geqslant \gamma_k + 1 + \frac{1}{\tilde{\gamma}_k} + \frac{1}{\tilde{\gamma}_k^2} + \cdots + \frac{1}{\tilde{\gamma}_k^{B-2}}. \end{aligned} \tag{3}$$

Since z_B minimizes $x + 1 + \frac{1}{x} + \frac{1}{x^2} + \cdots + \frac{1}{x^{B-2}}$ and the minimum is ρ_B, we have $\gamma_k < \tilde{\gamma}_k$. This can be seen by assuming $\gamma_k \geqslant \tilde{\gamma}_k$. The value of γ_k can then be decreased to $\tilde{\gamma}_k$ without violating (3). So, this would yield a better minimum for $x + 1 + \frac{1}{x} + \frac{1}{x^2} + \cdots + \frac{1}{x^{B-2}}$ than z_B does.

As a consequence of $\gamma_k < \tilde{\gamma}_k$ for all k, we have $\tilde{\gamma}_k < \max\{\tilde{\gamma}_{k-1}, \ldots, \tilde{\gamma}_{k-B+2}\}$. Now assume that $\tilde{\gamma}_k$ converges to some y. Then Eq. (2) holds when all γ_i's are substituted by y, implying that y gives a better minimum in (1) than z_B does. Thus, $\tilde{\gamma}_k$ cannot converge.

By the above we have that the value $\tilde{\gamma}_k$ is an upper bound on γ_k and decreases below any fixed value. Thus, eventually $\gamma_k < \tilde{\gamma}_k \leqslant 1$, contradicting $C_{k+1}^* \geqslant C_k^*$. □

By Theorems 1 and 2, we obtain the optimality of online Algorithm A^B. From the proof of Theorem 2 we see

that any optimal online algorithm presented with σ^{adv} must behave like Algorithm A^B as k grows large. No matter which optimal algorithm is used, the upper bound $\tilde{\gamma}_k$ must converge to z_B. In order to let $\tilde{\gamma}_k$ converge to z_B the value γ_k must converge to z_B. Therefore, as k grows large the batch sizes grow with rate z_B.

3. Concluding remarks

This paper presents an optimal online algorithm for online-list batch scheduling with any batch capacity B. For $B \leqslant 3$ this algorithm is a greedy type algorithm, i.e. each batch has the same length as the first job scheduled in it. As B goes to infinity the growth rate z_B in the online algorithm goes to 2 and its competitive ratio ρ_B to 4. Therefore, the known results for the unlimited capacity case are implied by the new results for the bounded capacity case.

For the online bidding problem there exists an optimal e-competitive randomized online algorithm [5]. This algorithm starts by drawing a random variable ξ uniformly from $[0, 1)$ and then submits bids $b_i = e^{i+\xi}$. This result gives immediately the optimal scheduling strategy for the online batch scheduling with unlimited capacity. It is an interesting question whether such a nice randomization of the algorithm presented here for the bounded capacity case, results in an optimal randomized online algorithm.

Acknowledgements

We thank Johann Hurink for helpful remarks. We are also grateful to anonymous referees and editors for many helpful suggestions. Part of this research was done while the first author visited Zhejiang University, Hangzhou. He is grateful for the hospitality received.

Part of this research has been funded by the Dutch BSIK/BRICKS project, NSFC (60573020), NSFC (10601048) and Chinese 973 project (2007CB310900).

References

[1] R.A. Baeza-Yates, J.C. Culberson, G.J.E. Rawlins, Searching in the plane, Information and Computation 106 (2) (1993) 234–252.
[2] W.W. Bein, L. Epstein, L.L. Larmore, J. Noga, Optimally competitive list batching, in: Algorithm Theory – SWAT2004, in: Lecture Notes in Computer Science, vol. 3111, 2004, pp. 77–89.
[3] P. Brucker, Scheduling Algorithms, fourth edition, Springer-Verlag, 2004.
[4] M. Chrobak, C. Kenyon, Competitiveness via doubling, SIGACT News 37 (4) (2006) 115–126.
[5] M. Chrobak, C. Kenyon, J. Noga, N.E. Young, Incremental medians via online bidding, Algorithmica 50 (4) (2008) 455–478.
[6] X. Deng, C.K. Poon, Y. Zhang, Approximation algorithms in batch processing, Journal of Combinatorial Optimization 7 (3) (2003) 247–257.
[7] C.Y. Lee, R. Uzsoy, Minimizing makespan on a single batch processing machine with dynamic job arrivals, International Journal on Production Research 37 (1) (1999) 219–236.
[8] C.Y. Lee, R. Uzsoy, L.A. Martin-Vega, Efficient algorithms for scheduling semiconductor burn-in operations, Operations Research 40 (4) (1992) 764–775.
[9] Q. Nong, J. Yuan, R. Fu, L. Lin, J. Tian, The single-machine parallel-batch on-line scheduling problem with family jobs to minimize makespan, International Journal of Production Economics 111 (2) (2008) 435–440.
[10] C.K. Poon, W. Yu, A flexible on-line scheduling algorithm for batch machine with infinite capacity, Annals of Operations Research 133 (1) (2005) 175–181.
[11] C.K. Poon, W. Yu, On-line scheduling algorithms for a batch machine with finite capacity, Journal of Combinatorial Optimization 9 (2) (2005) 167–186.
[12] G. Zhang, X. Cai, C.K. Wong, On-line algorithms for minimizing makespan on batch processing machines, Naval Research Logistics 48 (3) (2001) 241–258.