Int. J. Production Economics 156 (2014) 31–38

Contents lists available at ScienceDirect

Int. J. Production Economics

journal homepage: www.elsevier.com/locate/ijpe

Online bounded-batch scheduling to minimize total weighted completion time on parallel machines ☆

Ran Ma [a,b], Long Wan [c], Lijun Wei [c], Jinjiang Yuan [a,*]

[a] School of Mathematics and Statistics, Zhengzhou University, Zhengzhou 450001, China
[b] School of Mathematics and Information Science, Henan Polytechnic University, Jiaozuo 454000, China
[c] School of Information Technology, Jiangxi University of Finance and Economics, Nanchang 330013, China

ARTICLE INFO

Article history:
Received 5 June 2013
Accepted 16 May 2014
Available online 27 May 2014

Keywords:
Scheduling
Batch processing
Online algorithms
Performance guarantee

ABSTRACT

We consider the online bounded-batch scheduling to minimize total weighted completion time on parallel machines. In the problem, a set of n independent jobs arriving online over time has to be scheduled on m given machines, where the information of each job including its processing time and weight is not known in advance. Each machine can process up to b ($b < n$) jobs simultaneously as a batch. The processing time of a batch is the time required for processing the longest job in the batch. We present $4(1+\epsilon)$−competitive online algorithms on uniform machines when m is fixed and on identical machines when m is a part of input, respectively. Experimentation results show that the algorithm for identical machines (which covers the setting of uniform machines) is efficient.

1. Introduction and preliminaries

The online scheduling problem studied in this paper can be stated as follows. There are n jobs $J_1, J_2, \ldots, J_n$ arriving over time to be assigned into batches and scheduled without preemption on m parallel machines to minimize total weighted completion time. The information of each job J_j, including its processing time p_j and its weight w_j, will be known at its release date r_j. In a batch processing system, b jobs can be processed simultaneously as a batch, where b is called the batch capacity. If $b < n$ then the batch capacity b is bounded and unbounded otherwise. In this paper we only take the bounded capacity into consideration. A batch B is called full if $|B| = b$ and unfull if $|B| \le b-1$. The processing time of a batch is defined to be the maximum processing time of all jobs in the batch, and the weight of a batch is the sum of the weights of all jobs in the batch. Let $p(B)$ and $W(B)$ denote the processing time and weight of a batch B, respectively. Then we have $p(B) = \max\{p_j : J_j \in B\}$ and $W(B) = \sum_{J_j \in B} w_j$. The goal is to find a schedule to minimize the total weighted completion time.

In this paper, we deal with the above online scheduling problems on m uniform machines when m is a fixed number and m identical machines when m is a part of input, respectively. Then the problems we consider can be written in the three-field notation of Graham et al. (1979) as $Qm|\text{online}, r_j, \text{p}-\text{batch}, b < n|\sum w_j C_j$ and $P|\text{online}, r_j, \text{p}-\text{batch}, b < n|\sum w_j C_j$.

For the off-line version, research for the bounded-batch scheduling to minimize the total (weighted) completion time is concentrated on the setting of single machine. For problem $1|\text{p}-\text{batch}, b < n|\sum C_j$, Chandru et al. (1993) presented heuristics and branch-and-bound algorithms. Chandru (1993) presented an $O(k^3 b^{k+1})$−time optimal algorithm, where k is the number of processing times of the jobs. Hochbaum and Landy (1997) presented an optimal algorithm with time complexity $O(k^2 3^k)$. Brucker et al. (1998) presented an $O(n^{b(b-1)})$−time optimal algorithm and an $O(b^2 k^2 2^k)$−time optimal algorithm. For problem $1|\text{p}-\text{batch}, b < n|\sum w_j C_j$, Uzsoy and Yang (1997) presented heuristics and branch-and-bound algorithms. For problem $Pm|\text{p}-\text{batch}, b < n|\sum w_j C_j$, Li et al. (2006) presented a polynomial-time approximation scheme.

However, for the online version, the lack of knowledge of the future does not generally guarantee the optimality of the schedule generated by an online algorithm. To obtain a good online algorithm, a class of techniques is to convert an off-line scheduling algorithm into an online algorithm for the scheduling problem considered. To the best of our knowledge, such techniques are earliest used by Blum et al. (1994), Shmoys et al. (1995), and Hall et al. (1997).

For minimizing the total weighted completion time for an online scheduling, a general online framework, called Greedy-Interval, was presented by Hall et al. (1997). Greedy-Interval uses as a subroutine a dual ρ−approximation off-line algorithm for the following problem MSWP to obtain a 4ρ−competitive online algorithm.

☆Foundation item: Supported by NSFC (11271338), NSFC (11171313), and NSFC (11301528).

* Corresponding author. Tel.: +86 13838147625.
E-mail addresses: sungirlmr@hpu.edu.cn (R. Ma), cocu3328@163.com (L. Wan), lijunwei522@163.com (L. Wei), yuanjj@zzu.edu.cn (J. Yuan).

http://dx.doi.org/10.1016/j.ijpe.2014.05.012
0925-5273/

The maximum scheduled weight problem (*MSWP*): Given a certain scheduling environment, a deadline D, a set of jobs available at time 0, and a weight for each job, construct a feasible schedule that maximizes the total weight of jobs completed by D.

A dual $\rho-$approximation algorithm for MSWP is a polynomial-time algorithm that always delivers a schedule of length at most ρD and whose total weight is at least the optimal weight for the deadline D. Then the algorithm Greedy-Interval can be stated as follows.

Greedy-Interval: Partition the time horizon of possible completion time at geometrically increasing points. Let $\tau_i = 2^i$, $i = 0, 1, \ldots$, be in the time horizon. Greedy-Interval constructs the schedule iteratively. At iteration $i = 1, 2, \ldots$, we wait until time τ_{i-1}, and then focus on the set of jobs that have been released by this time but not yet scheduled. These jobs are scheduled to run from time $\rho\tau_{i-1}$ to $\rho\tau_i$ by invoking the dual $\rho-$approximation (off-line) algorithm with deadline $D = \tau_{i-1}$. □

The following result, which reveals the power of Greedy-Interval, was established in Hall et al. (1997).

Lemma 1.1. *Any dual $\rho-$approximation off-line algorithm for MSWP is efficiently converted in the aforementioned way by Greedy-Interval into a $4\rho-$*competitive *online algorithm for minimizing the total weighted completion time.* □

The above general online framework has received much attention in the recent literature. It is generalized widely to obtain better online algorithms not only for scheduling problems but also for other combinatorial optimization problems. For instance, Chakrabarti et al. (1996) extended the online techniques to a number of different scheduling models and even extended the technique to yield bicriteria scheduling algorithms within a small factor of both optimal schedule length and average weighted completion time. The deterministic strategy for L-OlDarp in the online traveling repairman problem together with the proof of performance presented by Krumke () was also an adaption of the algorithm greedy-interval. Prabhakaran (2001) proposed the algorithms for scheduling multimedia information delivery over wireless channels by taking advantage of the dual $\rho-$approximation algorithm for MSWP. In addition, Garg et al. (2007) gave online algorithms for minimizing the weighted completion time in order scheduling model from the above online technique.

We now pay attention to the following two papers most related to our research.

By using the general online framework Greedy-Interval, Chen et al. (2004) provided a $4(1+\epsilon)-$competitive online algorithm for problem $1|\text{online}, r_j,\ \text{p}-\text{batch}, b < n|\sum w_j C_j$. They first reduced the problem MSWP to a certain problem on interval graphs. Then they provided a dynamic programming algorithm with the time complexity $O(n^4 D)$. With the pseudo-polynomial algorithm, they derived for MSWP a dual FPTAS by applying the rounding and scaling techniques.

By using the general online framework Greedy-Interval again, Zhang and Bai (2007) gave a $4(2-1/b+\epsilon)-$competitive algorithm for problem $Pm|\text{online}, r_j,\ \text{p}-\text{batch}, b < n|\sum w_j C_j$ by a technique named Duplicating. In their algorithm, they first duplicated b identical machines for each machine. Then they determined the jobs to be scheduled on the mb machines by the algorithm for MSWP of problem $Pm|\text{online}, r_j|\sum w_j C_j$ (Hall et al. (1997)). Finally they put all jobs scheduled on each b machines into batches according to FBLPT (Full Batch Largest Processing Time) rule and processed each resulting batch on the corresponding machine which yielded the b machines.

In this paper, motivated by the general online framework Greedy-Interval, we present a dual FPTAS for the corresponding MSWP of problem $Qm|\text{online}, r_j, \text{p}-\text{batch}, b < n|\sum w_j C_j$ and a dual PTAS for the corresponding MSWP of problem $P|\text{online}, r_j,\ \text{p}-\text{batch}, b < n|\sum w_j C_j$. From Lemma 1.1, both of the online algorithms are $4(1+\epsilon)-$competitive, which improve the result provided by Zhang and Bai (2007) from the worst-case prospective. As a consequence, the dual FPTAS for the corresponding MSWP for m uniform machines also yields a dual FPTAS for the corresponding MSWP of problem $1|\text{online}, r_j,\ \text{p}-\text{batch}, b < n|\sum w_j C_j$. Moreover, we improve the time complexity provided in Chen et al. (2004) by providing a direct dynamic programming algorithm with the time complexity $O(n^2 \log n + n^2 D)$ for the corresponding MSWP. From Lemma 1.1, this leads to a more efficient (in the aspect of time complexity) $(4+\epsilon)-$competitive online algorithm for problem $1|\text{online}, r_j,\ \text{p}-\text{batch}, b < n|\sum w_j C_j$.

This paper is organized as follows. In Section 2, a dual FPTAS is presented for the corresponding MSWP of problem $Qm|\text{online}, r_j,\ \text{p}-\text{batch}, b < n|\sum w_j C_j$. In Section 3, a dual PTAS is presented for the corresponding MSWP of problem $P|\text{online}, r_j,\ \text{p}-\text{batch}, b < n|\sum w_j C_j$. In Section 4, we provide a brief computational study of the algorithms presented, under randomly chosen instances. Conclusions are presented in Section 5.

2. A dual FPTAS for MSWP(Qm-batch)

Let MSWP(Qm-batch) denote the problem MSWP restricted on m uniform batch processing machines with a batch capacity $b < n$ when m is fixed. For convenience, we renumber the n jobs such that $p_1 \le p_2 \le \cdots \le p_n$. For each batch B, we use $l(B)$ and $u(B)$ to denote the minimum and maximum job indices in batch B, respectively. Then $l(B) = \min\{j : J_j \in B\}$ and $u(B) = \max\{j : J_j \in B\}$. The processing time of batch B is given by $p(B) = p_{u(B)}$ and the weight of batch B is given by $W(B) = \sum_{j \in B} w_j$.

For each pair of positive integers l and u with $1 \le l \le u \le n$, we can form a batch $B_{l,u}$ by including up to b jobs of the maximum weights among all jobs of indices between l and u. Let $W_{l,u}$ denote the total weight of jobs in the batch $B_{l,u}$. Note that the total number of such batches $B_{l,u}$, $1 \le l \le u \le n$, is $O(n^2)$. We claim that the batches $B_{l,u}$ and the values $W_{l,u}$, $1 \le l \le u \le n$, can be determined in $O(n^2 \log n)$ time.

To prove the claim, we consider a given l with $1 \le l \le n$. The first batch $B_{l,l} = \{l\}$ and the first value $W_{l,l} = w_l$ can be determined in $O(1)$, and so, in $O(\log n)$ time. Then we generate the other batches $B_{l,u}$ and the other values $W_{l,u}$, $l+1 \le u \le n$, one by one. For each generated batch $B_{l,u}$, we maintain a linked list $\mathcal{L}_{l,u}$ in which the jobs in $B_{l,u}$ are sorted in the nondecreasing order of their weights. The first list $\mathcal{L}_{l,l}$ is trivially determined in $O(\log n)$ time. In each iteration, when $B_{l,k}$, $W_{l,k}$ and $\mathcal{L}_{l,k}$ have been determined for some k with $l \le k \le n-1$, we insert job J_{k+1} to the list $\mathcal{L}_{l,k}$ to form a list $\mathcal{L}'_{l,k}$ in which the jobs keep the nondecreasing order of their weights. Since $|B_{l,k}| \le b < n$, the insertion can be implemented in $O(\log n)$ time by the binary search. Let J_i be the first job in list $\mathcal{L}'_{l,k}$. If $|B_{l,k}| < b$, we just set $B_{l,k+1} = B_{l,k} \cup \{J_{k+1}\}$, $W_{l,k+1} = W_{l,k} + w_{k+1}$ and $\mathcal{L}_{l,k+1} = \mathcal{L}'_{l,k}$. If $|B_{l,k}| = b$, we then set $B_{l,k+1} = (B_{l,k} \cup \{J_{k+1}\}) \backslash \{J_i\}$ and $W_{l,k+1} = W_{l,k} + w_{k+1} - w_i$, and the next list $\mathcal{L}_{l,k+1}$ can be obtained from $\mathcal{L}'_{l,k}$ by deleting the first job J_i. Consequently, $B_{l,k+1}$, $W_{l,k+1}$ and $\mathcal{L}_{l,k+1}$ can be obtained from $B_{l,k}$, $W_{l,k}$ and $\mathcal{L}_{l,k}$ in $O(\log n)$ time. The claim follows.

We now study a more general problem, denoted by $Q(n, D_1, D_2, \ldots, D_m)$, in which the jobs scheduled on machine i have a common deadline $D_i \ge 0$. This means that we have the following Condition A for a feasible schedule.

Condition A. The sum of the real processing times of the batches scheduled on each machine i must be upper bounded by D_i.

Assume that the speed of machine i is given by $s_i > 0$ for $1 \le i \le m$. When a batch B with the original processing time $p(B)$ is