RAIRO-Oper. Res. 47 (2013) 1–8
DOI: 10.1051/ro/2012023

RAIRO Operations Research
www.rairo-ro.org

AN IMPROVED ALGORITHM FOR A BICRITERIA BATCHING SCHEDULING PROBLEM *

Cheng He[1], Xiumei Wang[2], Yixun Lin[2] and Yundong Mu[1]

Abstract. This note is concerned with the bicriteria scheduling problem on a series-batching machine to minimize maximum cost and makespan. An $O(n^5)$ algorithm has been established previously. Here is an improved algorithm which solves the problem in $O(n^3)$ time.

Keywords. Multicriteria scheduling, batching machine, maximum Cost, pareto optimal solutions.

Mathematics Subject Classification. 68Q25, 90B35, 90B50.

1. Introduction

This note studies the bicriteria problem of scheduling n jobs on a series-batching machine to minimize maximum cost and makespan simultaneously. A series-batching machine is a machine that can handle up to b jobs in a batch and jobs in a batch start and complete respectively at the same time, where the processing time of a batch is equal to the sum of the processing times of jobs in the batch. When a new batch starts, a constant setup time s occurs. For the series-batching scheduling problems, some work has been known in the literature. For example, Albers and Brucker [1] presented an $O(n)$ algorithm to the problem of minimizing weighted completion time, and Webster *et al.* [9] presented an $O(n^2)$-time algorithm to the problem of minimizing maximum lateness. On the other hand, for

Received March 10, 2012. Accepted November 26, 2012.

* *This work was supported by NSFC (grant no.11201121, 11001265, 11101383) and NSFHN (grant no. 112300410078).*

[1] School of Science, Henan University of Technology, Zhengzhou, Henan 450001, China. hech202@163.com

[2] Department of Mathematics, Zhengzhou University, Zhengzhou, Henan 450001, China

Article published by EDP Sciences

© EDP Sciences, ROADEF, SMAI 2013

https://doi.org/10.1051/ro/2012023 Published online by Cambridge University Press

a bicriteria scheduling problem, the simultaneous optimization is to determine all Pareto optimal solutions for two objectives. In this respect, Hoogeveen [8] showed that the bicriteria problem of minimizing two maximum cost criteria is solvable in $O(n^4)$ time. Our motivation is to generalize this result to the context of batching scheduling problems. In our previous work [5, 6] we investigated bicriteria problems of certain combinations of criteria. In particular, we proposed an $O(n^5)$-time algorithm for the bicriteria scheduling problem of minimizing maximum cost and makespan ([4]). In this note we further improve this algorithm to have complexity $O(n^3)$ by a more elaborate analysis.

The rest of the note is organized as follows. In Section 1 we describe the problem studied. Section 2 is dedicated to an improved result, an $O(n^3)$ algorithm. We shall follow the terminology and notation of Brucker [2].

2. Problem formulation

We first formulate the problem as follows. Suppose that there are n jobs J_1, $J_2, \ldots, J_n$ being scheduled on a batching machine. Each job J_j has a processing time p_j and a cost function $f_j(t)$ $(j = 1, \ldots, n)$. A schedule σ is a sequence of batches $\sigma = (B_1, B_2, \ldots, B_l)$, where each batch B_k $(k = 1, \ldots, l)$ is a set of jobs. Before the processing of each batch, there is a setup time s (a given positive constant). The processing time of batch B_k is $p(B_k) = \sum_{J_j \in B_k} p_j$ and its completion time is $C(B_k) = \sum_{q=1}^{k} p(B_q) + ks$. The completion time of job J_j in σ, for each $J_j \in B_k$ and $1 \leq k \leq l$, is $C_j(\sigma) = C(B_k)$. This type of batching machine is called series-batching machine, denoted by "*s-batch*" in short. Besides, we only consider the unbounded model in which the number of jobs in each batch is unlimited, denoted by "$b \geq n$" in short.

For a schedule σ, $f_j(\sigma) = f_j(C_j(\sigma))$ is defined as the cost of job J_j and $f_{\max}(\sigma) = \max_{j=1}^{n} f_j(\sigma)$ is the maximum cost of σ. Without loss of generality, we may assume that the processing times and the costs are integral. Additionally, we assume that the cost function $f_j(t)$ is strictly increasing with respect to the completion time $t = C_j$ of job J_j $(j = 1, \ldots, n)$.

In the context of bicriteria scheduling, we always use a composition objective function $F(f(\sigma), g(\sigma))$ to combine two performance criteria $f(\sigma)$ and $g(\sigma)$, where F is an unknown function that is nondecreasing in both arguments. In this note, the criteria $f(\sigma)$ and $g(\sigma)$ are maximum cost $f_{\max}(\sigma)$ and makespan $C_{\max}(\sigma)$. Following the traditional three-field notation of Graham *et al.* [3], this model is denoted by $1|s\text{-}batch, b \geq n|F(f_{\max}, C_{\max})$.

A feasible schedule σ is Pareto optimal, or nondominated, with respect to the performance criteria $f_{\max}$ and $C_{\max}$ if there is no feasible schedule π such that both $f_{\max}(\pi) \leq f_{\max}(\sigma)$ and $C_{\max}(\pi) \leq C_{\max}(\sigma)$, where at least one of the inequalities is strict. By solving a Pareto optimization problem in polynomial time, we mean that we are able to identify all Pareto optimal schedules in polynomial time. Moreover, with each schedule σ we associate a point $(f_{\max}(\sigma), C_{\max}(\sigma))$ in R^2 to

https://doi.org/10.1051/ro/2012023 Published online by Cambridge University Press

represent the status of two criteria, which is called a *Pareto optimal point* if σ is a Pareto optimal schedule.

The following result provides a general approach for finding Pareto optimal schedules ([7]): Let y be the optimal value of the problem $\alpha|f \leq \widehat{x}|g$, and let x be the optimal value of the problem $\alpha|g \leq y|f$. Then (x, y) is a Pareto optimal point for $\alpha||F(f, g)$.

3. Pareto optimality algorithm

The algorithmic framework of this note is the same as that of [4], only we revise the algorithm procedure into a more compact form. Especially, the main improvement is embodied in the complexity analysis by using efficient data structures.

As in [4], the makespan of schedule σ is mainly dependent on the number l of batches. So we have the following.

Proposition 3.1. *For any schedule σ with l batches, $C_{\max}(\sigma) = ls + \sum_{1\leq j\leq n} p_j$.*

Therefore, in order to find all Pareto optimal schedules of the problem $1|s\text{-}batch, b \geq n|\ F(f_{\max}, C_{\max})$, it suffices to minimize $f_{\max}(\sigma)$ for every fixed l, $1 \leq l \leq n$. For each given l, the algorithmic process of minimizing $f_{\max}(\sigma)$ of l batches is called a *Stage l*. In each stage, we can compute the sums

$$C_{\max}^{(l)} = ls + \sum_{j=1}^{n} p_j \quad (1 \leq l \leq n)$$

in a preprocessing step, which takes $O(n)$ time.

Moreover, by the classical Lawler algorithm for problem $1||f_{\max}$ (see [2]), as long as $f_{\max}(\sigma)$ is unchanged, every job is put as late as possible. This gives rise to the following.

Definition 3.2. A feasible schedule $\sigma = (B_1, \ldots, B_i, \ldots, B_k, \ldots, B_l)$ is called *f-tight* if for every job $J_j \in B_i$ and a subsequent batch B_k, $f_j(C(B_k)) > f_{\max}(\sigma)$.

Proposition 3.3 ([4])**.** *For each Pareto optimal point of the problem $1|s\text{-}batch, b \geq n|F(f_{\max}, C_{\max})$, there exists an f-tight schedule that attains this point.*

By this property, we may confine ourselves to consider the feasible schedules which are f-tight.

Now we modify the algorithm of [4] at each Stage l ($1 \leq l \leq n$). Suppose that σ^* is the last Pareto optimal schedule of the problem with at most $l-1$ batches.

Initial solution procedure (stage l)

Step 0. Set $\mathbf{J} := \{J_1, J_2, \ldots, J_n\}$, $k := l$, $t := C_{\max}^{(l)}$, and $f = f_{\max}(\sigma^*) - 1$.

Step 1. Set $B_k := \{J_j \in \mathbf{J} | f_j(t) \leq f\}$. If $B_k = \emptyset$, then go to Step 3, else set $k := k-1$, $\mathbf{J} := \mathbf{J} \setminus B_k$ and $t := t - p(B_k) - s$.

https://doi.org/10.1051/ro/2012023 Published online by Cambridge University Press

Step 2. If $k > 0$, go back to Step 1. If $k = 0$ and $\mathbf{J} = \emptyset$, then return the f-tight feasible schedule $\sigma_l := (B_1, B_2, \ldots, B_l)$. Otherwise $k = 0$ and $\mathbf{J} \neq \emptyset$, then Stage l terminates (there is no feasible schedule with l batches), go to Stage $l + 1$.

Step 3. If $k > 0$ and $B_k = \emptyset$, then the whole algorithm terminates (there is no other Pareto optimal schedule with l and more batches).

If $k = 0$ and $\mathbf{J} \neq \emptyset$ in Step 2 or $k > 0$ and $B_k = \emptyset$ in Step 3, we say that the decision problem of $f_{\max} \leq f$ is infeasible.

Proposition 3.4. *Unless the decision problem* $1|s\text{-}batch, b \geq n, C_{\max} = C^{(l)}_{\max}|f_{\max} \leq f$ *is infeasible, the Initial Solution Procedure produces an* f*-tight feasible schedule of Stage* l *in* $O(ln)$ *time.*

The proof is similar to that of Proposition 3.4 in [4] and so omitted here.

We next consider solving the optimization problem $1|s\text{-}batch, b \geq n, C_{\max} = C^{(l)}_{\max}|f_{\max}$ of Stage l. The following data structure is useful in evaluating the running time of the algorithm.

Definition 3.5. In a schedule $\sigma = (B_1, B_2, \ldots, B_l)$, if a job $J_j \in B_i$ $(1 \leq i \leq l)$, then we say that J_j has *order label* $L_j(\sigma) = i$. Moreover, we denote $\lambda(\sigma) = \sum_{1 \leq j \leq n} L_j(\sigma)$.

The algorithm of Stage l will be iterated from a feasible schedule to another. The prominent improvement of this note is based on the relation of iterated solutions represented in the following proposition.

Proposition 3.6. *Let* $\sigma = (B_1, B_2, \ldots, B_l)$ *be an initial schedule of* $f = f_{\max}(\sigma)$ *obtained by the Initial Solution Procedure and let* $\sigma' = (B'_1, B'_2, \ldots, B'_l)$ *be another* f*-tight feasible schedule of* $f' \leq f - 1$*. Then the following relations are satisfied:*

(1) $B'_i \subset B_i \cup B_{i+1}$ $(1 \leq i \leq l)$*;*
(2) $\lambda(\sigma') < \lambda(\sigma)$*.*

Proof. To obtain (1), we first show that $\cup_{k \leq i \leq l} B'_i \subseteq \cup_{k \leq i \leq l} B_i$ for $1 \leq k \leq l$. In fact, σ is obtained by the Initial Solution Procedure with threshold f. Similarly, σ' can be regarded to be obtained by the Initial Solution Procedure with threshold $f' \leq f - 1$. We compare the procedures for producing σ and σ'. Suppose that $t_k = C(B_k)$ and $t'_k = C(B'_k)$. Then $B'_k = \{J_j \in \mathbf{J}' | f_j(t'_k) \leq f'\}$ and $B_k = \{J_j \in \mathbf{J} | f_j(t_k) \leq f_{\max}(\sigma)\}$. We show the above-mentioned inclusion relation by induction on k backwards. For $k = l$, $t'_l = t_l = C^{(l)}_{\max}$. If $J_j \in B'_l$, then $f_j(t_l) = f_j(t'_l) \leq f' < f_{\max}(\sigma)$ and so $J_j \in B_l$, *i.e.*, $B'_l \subseteq B_l$. Assume that $\cup_{k+1 \leq i \leq l} B'_i \subseteq \cup_{k+1 \leq i \leq l} B_i$. Then $t'_k \geq t_k$, and $f_j(t_k) \leq f_j(t'_k) \leq f' < f_{\max}(\sigma)$ for any job $J_j \in B'_k$ and so J_j belongs to B_k or it has been deleted from $\mathbf{J}$ (that is, $B'_k \subseteq \cup_{k \leq i \leq l} B_i$). Therefore $\cup_{k \leq i \leq l} B'_i \subseteq \cup_{k \leq i \leq l} B_i$ by the induction hypothesis.

By the above inclusion relation it follows that $C(B_i) \leq C(B'_i)(1 \leq i \leq l)$. Furthermore, we can show that $C(B'_i) < C(B_{i+1})(1 \leq i < l)$. Suppose not. Then $t_{i+1} = C(B_{i+1}) \leq C(B'_i) = t'_i$ for some i. Thus $\cup_{i \leq h \leq l} B'_h \subseteq \cup_{i+1 \leq h \leq l} B_h$, and $\cup_{1 \leq h \leq i} B_h \subseteq \cup_{1 \leq h \leq i-1} B'_h$. In the procedure of producing σ', there are $i-1$ batches

https://doi.org/10.1051/ro/2012023 Published online by Cambridge University Press

before the time t'_{i-1}. Therefore, in the procedure of producing σ (note that $f \geq f'+1$), there are at most $i-1$ batches before the time t_i. This contradicts that σ has l batches. So (1) follows by $C(B_i) \leq C(B'_i) < C(B_{i+1})$ $(1 \leq i \leq l)$.

By (1), we have $L_j(\sigma') \leq L_j(\sigma)$ for every job J_j $(1 \leq j \leq n)$. Thus $\lambda(\sigma') \leq \lambda(\sigma)$. If this inequality holds with equality, then every job has the same order label, thus $\sigma = \sigma'$. This contradicts that $f_{\max}(\sigma') < f_{\max}(\sigma)$. Consequently, (2) holds. The proof is completed. □

Let $\sigma^0 = (B_1, B_2, \ldots, B_l)$ be the f-tight feasible schedule of Stage l and $f = f_{\max}(\sigma^0) - 1$. We have the following procedure for producing iterated solutions.

Iteration procedure (stage l)

Step 0. Set $\mathbf{J} := \{J_1, J_2, \ldots, J_n\}$, $k := l$, $t := C^{(l)}_{\max}$, and $f = f_{\max}(\sigma^0) - 1$.

Step 1. Set $B'_k := \{J_j \in \mathbf{J} \cap (B_k \cup B_{k+1}) | f_j(t) \leq f\}$. If $B'_k = \emptyset$, then go to Step 3, else set $k := k-1$, $\mathbf{J} := \mathbf{J} \setminus B'_k$ and $t := t - p(B'_k) - s$.

Step 2. If $k > 0$, go back to Step 1. If $k = 0$ and $\mathbf{J} = \emptyset$, then return the f-tight feasible schedule $\sigma' := (B'_1, B'_2, \ldots, B'_l)$, which is an improved schedule of σ^0. Otherwise $k = 0$ and $\mathbf{J} \neq \emptyset$, then Stage l terminates (there is no more feasible schedule with l batches), return the optimal schedule σ^0 of Stage l, and go to Stage $l+1$.

Step 3. If $k > 0$ and $B'_k = \emptyset$, then the whole algorithm terminates (there is no other Pareto optimal schedule with l and more batches), return the last Pareto optimal schedule σ^0.

The crucial point of the complexity improvement lies in the following.

Proposition 3.7. *The Iteration Procedure correctly gives three feasible conclusions:*

(1) an improved schedule σ' is obtained;
(2) σ^0 is an optimal schedule of Stage l;
(3) σ^0 is the last Pareto optimal schedule.

Moreover, the running time of the procedure is $O(n)$.

Proof. The correctness of the batching is based on Proposition 3.6. There are three outcomes as follows. (1) If the procedure stops at Step 2 with $k = 0$ and $\mathbf{J} = \emptyset$, then all jobs are exactly assigned into l batches and we get a schedule σ' with $f_{\max}(\sigma') \leq f_{\max}(\sigma^0) - 1$, which is an improved schedule. (2) If the procedure stops at Step 2 with $k = 0$ and $\mathbf{J} \neq \emptyset$, then n jobs cannot be assigned into l batches to get a better schedule of σ^0. So σ^0 is an optimal schedule of Stage l. (3) If the procedure stops at Step 3 with $k > 0$ and $B'_k = \emptyset$, then any schedule with at least l batches cannot has $f_{\max}(\sigma) < f_{\max}(\sigma^0)$, thus σ^0 is the last Pareto optimal schedule.

https://doi.org/10.1051/ro/2012023 Published online by Cambridge University Press

As for the running time, we see that the procedure has $l+1$ rounds. Let $b_k = |B_k|$ be the number of jobs in batch B_k. In the k-th round, the number of values $f_j(t)$ is at most $b_k + b_{k+1}$ and each value is compared with f. Therefore the overall running time is bounded by $2(b_{l+1} + b_l + b_l + b_{l-1} + b_{l-1} + b_{l-2} + \ldots + b_2 + b_1 + b_1 + b_0) = 4\Sigma_{1\le i\le l} b_i = 4n = O(n)$. This completes the proof. □

With the above preparations, we can state the main algorithm as follows.

Pareto optimality algorithm

Step 0.(Initiation)**:** Let $\sigma_1^* = (B_1)$, where $B_1 = \{J_1, J_2, \ldots, J_n\}$, be the schedule of one batch containing all jobs. Write $f_{\max}^{(1)} = f_{\max}(\sigma_1^*) = \max_{1\le j\le n}\{f_j(C_{\max}^{(1)})\}$. Then $(f_{\max}^{(1)},\ C_{\max}^{(1)})$ is the first Pareto optimal point. Set $m := 1$, $l = 1$, $i_1 := 1$.

Step 1. (Decision Step)**:** If $l = n$, then go to Step 3; otherwise let $l := l + 1$. To decide whether there exists a Pareto optimal schedule of l batches, let $f = f_{\max}^{(i_m)} - 1$ and run the Initial Solution Procedure. If the procedure stops at Step 3 with the conclusion that there is no other Pareto optimal schedule with l and more batches, then go to Step 3. If the procedure stops at Step 2 with the conclusion that there is no feasible schedule with l, then go back to the beginning of this step. Otherwise a feasible schedule is obtained, which is an initial schedule of Stage l.

Step 2. (Stage l)**:** With the initial schedule $\sigma^0 = (B_1, B_2, \ldots, B_l)$ obtained previously, we carry out the Iteration Procedure. There are three outcomes:

(1) If the procedure returns an improved schedule σ', then set $\sigma^0 := \sigma'$ and go back to the beginning of Step 2.
(2) If the procedure stops at Step 2 with an optimal schedule σ^0 of Stage l, then set $\sigma_l^* = \sigma^0$ and set $m := m + 1$ and $i_m := l$. Write $f_{\max}^{(i_m)} = f_{\max}(\sigma_{i_m}^*)$. Then $(f_{\max}^{(i_m)},\ C_{\max}^{(i_m)})$ is the m-th Pareto optimal point. Go back to Step 1.
(3) If the procedure stops at Step 3 with the last Pareto optimal schedule σ^0, then set $\sigma_l^* = \sigma^0$ and set $m := m + 1$ and $i_m := l$. Write $f_{\max}^{(i_m)} = f_{\max}(\sigma_{i_m}^*)$. Then $(f_{\max}^{(i_m)},\ C_{\max}^{(i_m)})$ is the last Pareto optimal point.

Step 3. (Termination)**:** Return all Pareto optimal schedules $\sigma_{i_1}^*, \sigma_{i_2}^*, \ldots, \sigma_{i_m}^*$ obtained.

Finally, we come to the conclusion of this paper.

Theorem 3.8. *The Pareto Optimality Algorithm produces all Pareto optimal points of problem* $1|s\text{-}batch, b \ge n|F(f_{\max}, C_{\max})$ *in* $O(n^3)$ *time.*

Proof. When the algorithm is carried out stage by stage, the number l of batches is increasing and so $C_{\max}^{(i_1)} < C_{\max}^{(i_2)} < \ldots < C_{\max}^{(i_m)}$. By the Initial Solution Procedure and the Iteration Procedure, we see that $f_{\max}^{(i_1)} > f_{\max}^{(i_2)} > \ldots > f_{\max}^{(i_m)}$. By Proposition 3.7, the algorithm terminates when no further Pareto optimal schedules can

https://doi.org/10.1051/ro/2012023 Published online by Cambridge University Press

be found. Hence the schedules obtained by the algorithm are all Pareto optimal schedules.

We next consider the running time of the algorithm. Similar to Proposition 3.6, we have the following claim.

Claim. Suppose that σ is the feasible schedule of Stage $l-1$ obtained by the Initial Solution Procedure and σ' is the Pareto optimal schedule of Stage l. Then $\lambda(\sigma) < \lambda(\sigma')$ (if the number of batches of σ' is larger than l, the conclusion is still established).

In fact, without loss of generality we may assume that $\sigma = (B_1, B_2, \ldots, B_{l-1})$ and $\sigma' = (B'_1, B'_2, \ldots, B'_l)$. Note that in the procedure producing σ', the threshold is $f' < f = f_{\max}(\sigma)$. If $J_j \in B'_l$, then $f_j(C^{(l-1)}_{\max}) < f_j(C^{(l)}_{\max}) \leq f' < f$, thus $J_j \in B_{l-1}$. So we have $B'_l \subseteq B_{l-1}$. Furthermore, by comparing each batch backwards (as in Proposition 3.6), we can see that $\cup_{k\leq i\leq l} B'_i \subseteq \cup_{k-1\leq i\leq l-1} B_i$ for $1 \leq k \leq l$. By changing the schedule from σ to σ', the order label L_j of each job is either unchanged or increasing by one. Hence $L_j(\sigma) \leq L_j(\sigma')$. However, there must be some jobs, say those in B'_l, whose order labels are strictly increasing. Therefore $\lambda(\sigma) < \lambda(\sigma')$, proving the claim.

By Proposition 3.6, during each stage the value of $\lambda(\sigma)$ is strictly decreasing. By the above claim, when the stage number l changes in the reverse order, the value of $\lambda(\sigma)$ is also strictly decreasing. Hence all values of $\lambda(\sigma)$ in our algorithm are different and its maximum value is given by $\lambda(\sigma) = \sum_{1\leq j\leq n} L_j(\sigma) \leq 1+2+\ldots+n = \frac{1}{2}n(n-1)$. Therefore the total number of the schedules generated by the algorithm is $O(n^2)$. By Proposition 3.7, the computation of generating each schedule in the Iteration Procedure takes $O(n)$ time. So the total running time of all iteration procedures in the algorithm is $O(n^3)$. On the other hand, by Proposition 3.4, the computation of generating each initial schedule in the Initial Solution Procedure takes $O(ln)$ time. So the total running time of all Initial Solution Procedures in the algorithm is also $O(n^3)$, as $\sum_{1\leq l\leq n} ln = (1+2+\ldots+\mathrm{n})n = \frac{1}{2}n^2(n-1)$. Hence the overall complexity of the algorithm is $O(n^3)$, completing the proof. □

References

[1] S. Albers and P. Brucker, The complexity of one-machine batching problems. *Discrete Appl. Math.* **47** (1993) 87–107.

[2] P. Brucker, Scheduling Algorithms (third edition). Springer, Berlin (2001).

[3] R.L. Graham, E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annal. Discr. Math.* **5** (1979) 287–326.

[4] C. He, H. Lin, Y.X. Lin and J. Tian, Bicriteria scheduling on a series -batching machine to minimize maximum cost and makespan. *CEJOR Cent. Eur. J. Oper. Res.* **21** (2013) 177–186.

[5] C. He, Y.X. Lin and J.J. Yuan, Bicriteria scheduling of minimizing maximum lateness and makespan on a serial-batching machine. *Found. Comput. Decision Sci.* **33** (2008) 369–376.

https://doi.org/10.1051/ro/2012023 Published online by Cambridge University Press

[6] C. He, Y.X. Lin and J.J. Yuan, A DP algorighm for minimizing makespan and total completion time on a series-batching machine. *Informat. Process. Lett.* **109** (2009) 603–607.
[7] H. Hoogeveen, Multicriteria scheduling. *European J. Oper. Res.* **167** (2005) 592–623.
[8] J.A. Hoogeveen, Single-machine scheduling to minimize a function of two or three maximum cost criteria. *J. Algorithms* **21** (1996) 415–433.
[9] S. Webster and K.R. Baker, Scheduling groups of jobs on a single machine. *Oper. Res.* **43** (1995) 692–703.

https://doi.org/10.1051/ro/2012023 Published online by Cambridge University Press