RAIRO-Oper. Res. 52 (2018) 55–60
https://doi.org/10.1051/ro/2017089

RAIRO Operations Research
www.rairo-ro.org

HIERARCHICAL OPTIMIZATION ON AN UNBOUNDED PARALLEL-BATCHING MACHINE[☆]

Cheng He[*] and Li Li

Abstract. This paper studies a hierarchical optimization problem on an unbounded parallel-batching machine, in which two objective functions are maximum costs, representing different purposes of two decision-makers. By a hierarchical optimization problem, we mean the problem of optimizing the secondary criterion under the constraint that the primary criterion is optimized. A parallel-batching machine is a machine that can handle several jobs in a batch in which all jobs start and complete respectively at the same time. We present an $O(n^4)$-time algorithm for this hierarchical scheduling problem.

Mathematics Subject Classification. 90B35, 90C29

Received March 1, 2017. Accepted November 30, 2017.

1. Introduction

In recent years, there has been an increasing interest in multicriteria scheduling problems because of their great application potential. For example, decision-makers usually pursue several performance criteria simultaneously in product quality evaluations. Among different types of the bicriteria scheduling problems, the *hierarchical optimization problems* play a basic role, in which we are asked to optimize the secondary criterion under the constraint that the primary criterion is optimized. The status of research in this direction can be consulted in [12, 13].

In this paper we study a new model of hierarchical optimization on a batching machine, in which each job J_j has a processing time p_j and two maximum cost functions $f_{\max}$ and $g_{\max}$. This is motivated by the situation that different decision-makers may have different cost functions for a job, representing different expected expenses. For example, the first maximum cost $f_{\max}$ may serve as the cost expected by the investment corporation (or the consumer), while the second maximum cost $g_{\max}$ may be the planned cost required by the project executor (or the producer). A similar scenario appears in the two-agent scheduling problems [1, 2, 3], in which each agent has his own objective.

For the batch scheduling, we concentrate on the unbounded parallel-batching (rather than serial-batching) model in which the jobs that are processed together form a batch with the same starting time and completion time, and the processing time of a batch is equal to the largest processing time of jobs in it. This model is motivated by the applications of burn-in operations for integrated circuit manufacturing and other areas. Following

☆ This work was supported by NSFC (11201121) and NSFSTDOHN (162300410221).

Keywords and phrases: Hierarchical optimization, batching machine, maximum cost.

School of Science, Henan University of Technology, Zhengzhou 450001, Henan, PR China.

* Corresponding author: hech202@163.com

the three-field notation scheme of Graham *et al.* [7], we denote the problem by $1|p\text{-}batch|Lex(f_{\max}, g_{\max})$, where "*p-batch*" refers to the parallel-batching and $Lex(\gamma_1, \gamma_2)$ represents minimizing γ_2 subject to the restriction that γ_1 is optimized ("Lex" stands for lexicographical optimization). Moreover, we study the unbounded model in which the number of jobs in each batch is unlimited. In this paper, we present an $O(n^4)$-time algorithm for this hierarchical optimization problem.

For detail developments of batch scheduling and multicriteria scheduling, we refer to surveys [4] and [11, 12], respectively. We only mention a few related results here. For the batch scheduling problem $1|p\text{-}batch|L_{\max}$, Brucker *et al.* [5] presented a dynamic programming algorithm that requires $O(n^2)$ time for minimizing the maximum lateness. As a byproduct, Brucker *et al.* [5] also pointed the feasibility (decision) problem $1|p\text{-}batch, f_{\max} \leq k|\cdot$ is solvable in $O(n^2 + n \log P)$ time, where $P = \sum_{1 \leq j \leq n} p_j$. So problem $1|p\text{-}batch|f_{\max}$ can be solved in polynomial time by binary search for the feasibility problem. Geng and Yuan [6] provided an improved algorithm which solves the problem in $O(n^4)$ time. In [8], an $O(n^3)$-time algorithm for this hierarchical scheduling problem with two maximum lateness $1|d_j, d'_j, p\text{-}batch|Lex(L_{\max}, L'_{\max})$ is given. The problems $1|p\text{-}batch|(L_{\max}, C_{\max})$ and $1|p\text{-}batch|(f_{\max}, C_{\max})$ have been solved in $O(n^3)$ time and $O(n^3 \log P)$ time, respectively [9, 10], Geng and Yuan [6] presented an improved $O(n^4)$-time algorithm for this problem $1|p\text{-}batch|(f_{\max}, C_{\max})$, where (γ_1, γ_2) refers to the simultaneous optimization in the sense of finding all Pareto optimal schedules for two criteria γ_1 and γ_1.

The paper is organized as follows. In Section 2 we state some preliminaries. In Section 3, we present a strongly polynomial-time algorithm, an $O(n^4)$ algorithm for the problem. Section 4 gives a short summary. We shall follow the terminology and notation of [4].

2. Preliminaries

Suppose that we are given n jobs, denoted by $J_1, J_2, \ldots, J_n$. These jobs are to be scheduled on a single batching machine that is continuously available from time zero onwards and that can handle any number of jobs at the same time. Job J_j has a processing time p_j and its two cost functions f_j and g_j ($j = 1, \ldots, n$), respectively, where $f_j(t)$ and $g_j(t)$ denotes two costs incurred if the job is completed at time t, which are nondecreasing functions of t, for $j = 1, \ldots, n$. Given a schedule σ, we denote the completion time of job J_j in σ by $C_j(\sigma)$, $f_j(\sigma) = f_j(C_j(\sigma))$ and $g_j(\sigma) = g_j(C_j(\sigma))$ are defined as two costs of job J_j in σ. Furthermore, $f_{\max}(\sigma) = \max_{j=1}^{n} f_j(C_j(\sigma))$ and $g_{\max}(\sigma) = \max_{j=1}^{n} g_j(C_j(\sigma))$ are the maximum costs of the jobs in σ with respect to cost functions $f_j(\sigma)$ and $g_j(\sigma)$ respectively. Without loss of generality, we assume that the job parameters are integral.

For problems of minimizing a regular objective function without job release dates, we know that there must be an optimal solution in which the batches are processed contiguously from time zero onwards. Throughout the paper, we restrict our attention to the solutions with this property. Thus, a schedule σ can be denoted by a batch sequence $\sigma = (B_1, B_2, \ldots, B_r)$, where each batch B_l ($l = 1, \ldots, r$) is a set of jobs. The processing time of batch B_l is $p(B_l) = \max_{J_j \in B_l}\{p_j\}$ and its completion time is $C(B_l) = \sum_{q=1}^{l} p(B_q)$. Note that the completion time of job J_j in σ, for each $J_j \in B_l$ and $1 \leq l \leq r$, is $C_j(\sigma) = C(B_l)$. When there is no ambiguity, we abbreviate $C_j(\sigma)$ to C_j. This type of batching machine is called parallel-batching machine, denoted by "*p-batch*" in short. Besides, we only consider the unbounded model in which the number of jobs in each batch is unlimited.

In this paper, the criteria under consideration are two regular minimax objective functions: maximum costs $f_{\max}(\sigma)$ and $g_{\max}(\sigma)$. Our goal is solving the problem $1|p\text{-}batch|Lex(f_{\max}, g_{\max})$. Here, the objective $Lex(f_{\max}, g_{\max})$ stands for the hierarchical (lexicographical) optimization of minimizing $g_{\max}$ under the constraint that $f_{\max}$ is minimum, namely, the minimization of $g_{\max}$ is taken in the set of all optimal schedules of problem $1|p\text{-}batch|f_{\max}$.

3. Polynomial-time algorithm

Following Brucker *et al.* [5], a batch schedule $\sigma = (B_1, B_2, \ldots, B_r)$ is called an *SPT-batch schedule*, if for any two jobs J_i and J_j, $p_i \leq p_j$ implies $C_i(\sigma) \leq C_j(\sigma)$. Further, a schedule σ is called a *strict SPT-batch schedule* if

σ is an SPT-batch schedule and all jobs with identical processing time belong to a common batch in σ. Similar to Lemma 1 of Brucker *et al.* [5], we can easily obtain the following lemma.

Lemma 3.1. *For problem $1|p\text{-}batch|Lex(f_{\max}, g_{\max})$, there exists an optimal strict SPT-batch schedule.*

Proof. Consider an optimal schedule $\sigma = (B_1, \dots, B_l, \dots, B_q, \dots, B_r)$, where $1 \le l < q \le r$, with $J_k \in B_l$, $J_j \in B_q$, and $p_k \ge p_j$. Consider now the schedule $\sigma' = (B_1, \dots, B_l \bigcup \{J_j\}, \dots, B_q \backslash \{J_j\}, \dots, B_r)$ that is obtained from σ by moving job J_j to batch B_l from B_q. Since $p_k \ge p_j$, we have $p(B_l \bigcup \{J_j\}) = p(B_l)$, $p(B_q \backslash \{J_j\}) \le p(B_q)$. Thus $f_i(\sigma) \ge f_i(\sigma')$ and $g_i(\sigma) \ge g_i(\sigma')$ $(i = 1, \dots, n)$, *i.e.*, $f_{\max}(\sigma') \le f_{\max}(\sigma)$ and $g_{\max}(\sigma') \le g_{\max}(\sigma)$. Hence, the new schedule σ' is also optimal. A finite number of repetitions of this procedure yields an optimal schedule of the required form. □

Lemma 3.1 shows that we may restrict our attention to the strict SPT-batch schedules. To simplify the model, we re-index the jobs according to the SPT rule so that $p_1 \le p_2 \le \cdots \le p_n$, which takes $O(n \log n)$ time. Assume that the n jobs have m different processing times $p^{(1)}, p^{(2)}, \dots, p^{(m)}$ so that $p^{(1)} < p^{(2)} < \cdots < p^{(m)}$. Let $\mathcal{J}^{(i)} = \{J_j : p_j = p^{(i)}\}$ and $n_i = |\mathcal{J}^{(i)}|$ denotes the number of jobs in job set $\mathcal{J}^{(i)}$ $(i = 1, \dots, m)$. By Lemma 3.1, we may regard each $\mathcal{J}^{(i)}$ as a merged job with processing times $p^{(i)}$ and cost function $f^{(i)}(t) = \max\{f_j(t) : J_j \in \mathcal{J}^{(i)}\}$ and $g^{(i)}(t) = \max\{g_j(t) : J_j \in \mathcal{J}^{(i)}\}$ for $t \ge 0$ without affecting the costs $f_{\max}$ and $g_{\max}$. Note that, $f^{(i)}(t)$ and $g^{(i)}(t)$ can be calculated in $O(n_i)$ time for each given i and t. Introducing of the merged jobs $\mathcal{J}^{(i)}$ simplifies our discussions and representations. Furthermore, we need not calculate $f^{(i)}(t)$ and $g^{(i)}(t)$ for all values of t in our algorithm, which guarantees the polynomial-time complexity. Hence we assume that our discussions are based on introducing of the merged jobs in the following.

Lemma 3.2. *[6] An optimal strict SPT-batch schedule for the problem $1|p\text{-}batch|f$ can be obtained in $O(n^4)$ time, where $f \in \{f_{\max}, g_{\max}\}$.*

Let σ^* and π^* be the optimal schedules of $1|p\text{-}batch|f_{\max}$ and $1|p\text{-}batch|g_{\max}$, respectively. Let $f^* := f_{\max}(\sigma^*)$ and $\overline{g} := g_{\max}(\sigma^*)$ and $\underline{g} := g_{\max}(\pi^*)$. Then problem $1|p\text{-}batch|Lex(f_{\max}, g_{\max})$ is equivalent to problem $1|p\text{-}batch, f_{\max} \le f^*|g_{\max}$ and $\underline{g} \le g_{\max} \le \overline{g}$. Assume that there is a $\mathcal{DP}(g)$ that may solve the problem $1|p\text{-}batch, f_{\max} \le f^*, g_{\max} \le g|\cdot$. Then $1|p\text{-}batch, f_{\max} \le f^*|g_{\max}$ can be solved by solving a series of the feasibility problems $1|p\text{-}batch, f_{\max} \le f^*, g_{\max} \le g|\cdot$ for the decreasing g, where $\underline{g} \le g \le \overline{g}$. The iteration procedure as follows.

3.1. Iteration procedure ($\mathcal{DP}(g)$)

Step 1: Let σ^* be defined as above.
Step 2: If $g_{\max}(\sigma^*) = \underline{g}$, then σ^* is an optimal schedule of $1|p\text{-}batch|Lex(f_{\max}, g_{\max})$ and stop. Otherwise, let $g := g_{\max}(\sigma^*) - 1$.
Step 3: Solving the problem $1|p\text{-}batch, f_{\max} \le f^*, g_{\max} \le g|\cdot$ by $\mathcal{DP}(g)$. If the problem is infeasible, then σ^* is an optimal schedule of $1|p\text{-}batch|Lex(f_{\max}, g_{\max})$ and stop. Otherwise, let σ^* is the schedule obtained by performing $\mathcal{DP}(g)$, go back to Step 2.

In fact, we solve problem $1|p\text{-}batch, f_{\max} \le f^*, g_{\max} \le g|\cdot$ by solving problem $1|p\text{-}batch, f_{\max} \le f^*, g_{\max} \le g|C_{\max}$, and if the optimal value $C_{\max} < +\infty$, then problem $1|p\text{-}batch, f_{\max} \le f^*, g_{\max} \le g|\cdot$ is feasible; otherwise infeasible. The problem $1|p\text{-}batch, f_{\max} \le f^*, g_{\max} \le g|C_{\max}$ can be solved by the following dynamic programming algorithm.

$\mathcal{DP}(g)$: Let $C(j, g)$ be the minimum makespan for all feasible SPT-batch schedules, in respect to the jobs $J^{(1)}, J^{(2)}, \dots, J^{(j)}$, of $1|p\text{-}batch, f_{\max} \le f^*, g_{\max} \le g|C_{\max}$. Besides, we use $\{J^{(k+1)}, J^{(k+2)}, \dots, J^{(j)}\}$ to denote the last batch in the schedule. Thus the last batch $\{J^{(k+1)}, J^{(k+2)}, \dots, J^{(j)}\}$ has completion time $C(k, g) + p^{(j)}$. The initialization is $C(0, g) = 0$ and the recursion relation for $j = 1, \dots, m$ is

$$C(j,g) = \min\{C(k,g) + p^{(j)} :\ 0 \le k < j, \max_{k+1 \le i \le j}\{f^{(i)}(C(k,g) + p^{(j)})\} \le f^*, \max_{k+1 \le i \le j}\{g^{(i)}(C(k,g) + p^{(j)})\} \le g\}. \tag{3.1}$$

Hence $C(j,g)$ is the minimum value of $C(k,g)+p^{(j)}$ from among these feasible schedules. Finally, the optimal value is equal to $C(m,g)$ and the corresponding optimal schedule can be found by backtracking. In more detail, let $k_j(g)$ be the value of k attaining the minimum of $C(k,g)+p^{(j)}$ in (1), *i.e.*, the optimal decision of stage j (if there is no k so that (1) holds, then let $C(j,g)=+\infty$ and $k_j(g)=+\infty$ and the problem is infeasible). Then the optimal schedule σ is obtained by taking $\{J^{(k_m(g)+1)},\dots,J^{(m)}\}$ as the last batch, and $\{J^{(k_j(g)+1)},\dots,J^{(j)}\}$ where $j=k_m(g)$ as the second last batch, and so on.

For the solutions of equation (3.1), we have the following properties:

Property 3.3. *Let $t_k(j,g)=C(k,g)+p^{(j)}$ for $0\le k<j$ and $\underline{g}\le g\le \overline{g}$. Then*

(a) $C(j-1,g)<C(j,g)$for $j=2,3,\dots,m$.
(b) $k_j(g)=\min\{k:\ 0\le k<j,\ \max_{k+1\le i\le j}\{f^{(i)}(t_k(j,g))\}\le f^*,\ \max_{k+1\le i\le j}\{g^{(i)}(t_k(j,g))\}\le g\}$.
(c) $k_{j-1}(g)\le k_j(g)$.
(d) If $g'<g$, then $C(j,g')\ge C(j,g)$.
(e) If $g'<g$ and $k_j(g')$ is defined, then $k_j(g')\ge k_j(g)$.

Proof. By definition, $C(j,g)$ is the minimum makespan of the first j jobs while $C(j-1,g)$ is the minimum makespan of the first $j-1$ jobs and $p^{(j-1)}<p^{(j)}$, so (a) is clear.

For (b), it follows from (a).

For (c), if $k_j(g)=j-1$, then $k_j(g)=j-1>k_{j-1}(g)$. Otherwise $k_j(g)<j-1$, $k_j(g)=\min\{k:0\le k<j-1,\ \max_{k+1\le i\le j}\{f^{(i)}(t_k(j,g))\}\le f^*,\max_{k+1\le i\le j}\{g^{(i)}(t_k(j,g))\}\le g\}\ge\min\{k:0\le k<j-1,\ \max_{k+1\le i<j}\{f^{(i)}(t_k(j-1,g))\}\le f^*,\ \max_{k+1\le i<j}\{g^{(i)}(t_k(j-1,g))\}\le g\}=k_{j-1}(g)$ (since $p^{(j)}>p^{(j-1)}$ and $t_k(j,g)\ge t_k(j-1,g)$).

For (d), we compare the two solutions obtained by $\mathcal{DP}(g)$ for the jobs $J^{(1)},J^{(2)},\dots,J^{(j)}$ with respect to g and g', respectively. By induction, we assume that $C(k,g)\le C(k,g')$ for $k<j$. Then if $k_j(g')=k'<j$, we have $C(j,g')=C(k',g')+p^{(j)}\ge C(k',g)+p^{(j)}$, *i.e.*, $t_{k'}(j,g')\ge t_{k'}(j,g)$ and

$$\max_{k'+1\le i\le j}\{f^{(i)}(t_{k'}(j,g))\}\le \max_{k'+1\le i\le j}\{f^{(i)}(t_{k'}(j,g'))\}\le f^*,$$
$$\max_{k'+1\le i\le j}\{g^{(i)}(t_{k'}(j,g))\}\le \max_{k'+1\le i\le j}\{g^{(i)}(t_{k'}(j,g'))\}\le g'<g.$$

Hence, $C(j,g)=\min\{t_k(j,g):\ 0\le k<j,\ \max_{k+1\le i\le j}\{f^{(i)}(t_k(j,g))\}\le f^*,\ \max_{k+1\le i\le j}\{g^{(i)}(t_k(j,g))\}\le g\}\le t_{k'}(j,g)\le t_{k'}(j,g')=C(j,g')$

For (e), we have $t_k(j,g)=C(k,g)+p^{(j)}\le C(k,g')+p^{(j)}=t_k(j,g')$ by (d). Therefore, if $f^{(i)}(t_k(j,g'))\le f^*$, $g^{(i)}(t_k(j,g'))\le g'$, then $f^{(i)}(t_k(j,g))\le f^{(i)}(t_k(j,g'))\le f^*$, $g^{(i)}(t_k(j,g))\le g^{(i)}(t_k(j,g'))\le g'<g$. It follows that $k_j(g')=\min\{k:0\le k<j,\max_{k+1\le i\le j}\{f^{(i)}(t_k(j,g'))\}\le f^*,\ \max_{k+1\le i\le j}\{g^{(i)}(t_k(j,g'))\}\le g'\}\ge\min\{k:0\le k<j,\max_{k+1\le i\le j}\{f^{(i)}(t_k(j,g))\}\le f^*,\ \max_{k+1\le i\le j}\{g^{(i)}(t_k(j,g))\}\le g\}=k_j(g)$. The proof is completed. □

Theorem 3.4. *If all jobs are indexed according to the SPT rule and merged in advance, then the running time of $\mathcal{DP}(g)$, for each given g, based on equation (3.1) is $O(n^2)$.*

Proof. By Property 3.3(c), $0=k_1(g)\le k_2(g)\le\cdots\le k_m(g)\le m-1<n$. For each j with $1<j\le m$, we determine $k_j(g)$ and $C(j,g)$ by deciding if $f^{(i)}(t_k(j,g))\le f^*$ and $g^{(i)}(t_k(j,g))\le g$ for $k=k_{j-1}(g),k_{j-1}(g)+1,\dots k_j(g)$ and $k+1\le i\le j$ by Property 3.3(b-c), and deciding if $f^{(i)}(t_k(j,g))\le f^*$ and $g^{(i)}(t_k(j,g))\le g$ need $O(n_{k+1}+n_{k+2}+\cdots+n_j)=O(n)$ time for given k and all $k+1\le i\le j$. so determining $k_j(g)$ and $C(j,g)$ take $O(n(k_j(g)-k_{j-1}(g)+1))$ time for given j. Hence, in $O(nm)=O(n^2)$ time, we determine all $k_j(g)$ and $C(j,g)$, $1\le j\le m$. It follows that the running time of $\mathcal{DP}(g)$ is $O(n^2)$. □

Proposition 3.5. *The number of steps of the Iteration Procedure ($\mathcal{DP}(g)$) is at most $O(n^2)$.*

Proof. By Theorem 3.4, $\mathcal{DP}(g)$ solves problem $1|p\text{-}batch, f_{\max} \le f^*, g_{\max} \le g|C_{\max}$ in $O(n^2)$ time. So $\mathcal{DP}(g)$ also solves problem $1|p\text{-}batch, f_{\max} \le f^*, g_{\max} \le g|\cdot$ in $O(n^2)$ time. Further, we can solve the problem $1|p\text{-}batch|Lex(f_{\max}, g_{\max})$ by iteration procedure ($\mathcal{DP}(g)$).

We compare two successive solutions g^i and g^{i+1} with $g^{i+1} < g^i$. The corresponding schedules are σ^i and σ^{i+1} respectively. Suppose that $\sigma^i = (B_1, B_2, \ldots, B_r)$. The last job in a batch is called the *boundary job* of this batch. Let J_{k_l} be the boundary job of batch B_l. Then $k_1 < k_2 < \cdots < k_{r-1} < k_r = m$. We define a *weight* of schedule σ^i as $\lambda(\sigma^i) = k_1 + k_2 + \cdots + k_{r-1}$. On the other hand, suppose that $\sigma^{i+1} = (B'_1, B'_2, \ldots, B'_{r'})$ and the indices of the boundary jobs are $k'_1, k'_2, \ldots, k'_{r'-1}, m$. Then the weight of σ^{i+1} is $\lambda(\sigma^{i+1}) = k'_1 + k'_2 + \cdots + k'_{r'-1}$. We have the following claim.

Claim $\lambda(\sigma^i) < \lambda(\sigma^{i+1})$.

In fact, for the last batches of σ^i and σ^{i+1}, by Property 3.3(e), we have $k_{r-1} = k_m(g^i) \le k_m(g^{i+1}) = k'_{r'-1}$. Furthermore, let $j = k_{r-1}$ and $j' = k'_{r'-1}$. Then $j \le j'$. By (c) and (e) of Property 3.3, we have $k_{r-2} = k_j(g^i) \le k_{j'}(g^i) \le k_{j'}(g^{i+1}) = k'_{r'-2}$. Assume that $r_0 = \min\{r, r'\}$. Then by the same argument, we can show that $k_{r-h} \le k'_{r'-h}$ for $1 \le h \le r_0 - 1$. If $r_0 = \min\{r, r'\} = r' \ne r$, *i.e.*, $r' < r$, then $k_{r-r_0+1} \le k'_1$, where $r - r_0 + 1 \ge 2$. Similarly, let $l = k_{r-r_0+1}$ and $l' = k'_1$. Then $l \le l'$. By (c) and (e) of Property 3.3 again, we have $1 \le k_{r-r_0} = k_l(g^i) \le k_{l'}(g^i) \le k_{l'}(g^{i+1}) = 0$, a contradiction. Therefore $r' \ge r$ and $k_{r-h} \le k'_{r'-h}$ for $1 \le h \le r-1$. Hence $\lambda(\sigma^i) \le \lambda(\sigma^{i+1})$. However, these two schedules are different, and so they cannot have the same set of boundary jobs. Thus the claim follows.

The claim says that the weight $\lambda(\sigma^i)$ is strictly increasing during the iteration procedure. However, the weight $\lambda(\sigma)$ has an upper bound $1 + 2 + \cdots + (m-1) = \frac{1}{2}m(m-1) \le O(n^2)$ for any schedule σ. Hence the number of steps is at most $O(n^2)$, proving the result. □

To summarize, we obtain the following theorem.

Theorem 3.6. *Iteration procedure ($\mathcal{DP}(g)$) solves the hierarchical optimization problem $1|p\text{-}batch|Lex(f_{\max}, g_{\max})$ in $O(n^4)$ time.*

Proof. The correctness of iteration procedure ($\mathcal{DP}(g)$) is due to the analysis in proof of Proposition 3.5. Let us see the running time of the algorithm. Step 1 takes $O(n^4)$ time by Lemma 3.2. From Theorem 3.4 and Proposition 3.5, the number of steps of the iteration procedure ($\mathcal{DP}(g)$) in Step 2-3 is $O(n^2)$, and in each step, the $\mathcal{DP}(g)$ of (1) takes $O(n^2)$ time. Besides, all jobs are sorted and merged in advance in $O(n \log n)$ time. Therefore the overall complexity is $O(n^4)$. Thus the result is proved. □

4. Concluding remarks

In the foregoing discussion, we investigate hierarchical optimization problem with two maximum cost functions. Moreover, for the simultaneous optimization problem, in the sense of finding all Pareto optimal schedules, with two objective functions maximum cost and makespan, [6, 10] presented respectively a polynomial-time algorithm. One of our future work would be simultaneous optimization problem with two maximum cost functions on an an unbounded parallel-batching machine.

References

[1] A. Agnetis, P.B. Mirchani, D. Pacciarelli and A. Pacifici, Scheduling problems with two competing agents. *Oper. Res.* **52** (2004) 229–242.

[2] A. Agnetis, J.-C. Billaut, S. Gawiejnowicz, D. Pacciarelli and A. Soukhal, Multiagent Scheduling - Models and Algorithms Problems. Springer-Verlag, Berlin (2014).

[3] K.R. Baker and J.C. Smith, A multiple-criterion model for machine scheduling. *J. Sched.* **6** (2003) 7–16.

[4] P. Brucker, Scheduling Algorithms, 4th edn. Springer-Verlag, Berlin (2004).

[5] P. Brucker, A. Gladky, H. Hoogeveen, M.Y. Kovalyov, C.N. Potts, T. Tautenhahn *et al.*, Scheduling a batching machine. *J. Sched.* **1** (1998) 31–54.

[6] Z.C. Geng and J.J. Yuan, A note on unbounded parallel-batch scheduling. *Inf. Process. Lett.* **115** (2015) 969–974.

[7] 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. *Ann. Discret. Math.* **5** (1979) 287–326.

[8] C. He and H. Lin, Hierarchical optimization with double due dates on an unbounded parallel-batching machine to minimize maximum lateness. *4OR Q. J. Oper. Res.* **14** (2016) 153–164.

[9] C. He, Y.X. Lin and J.J. Yuan, Bicriteria scheduling on a batching machine to minimize maximum lateness and makespan. *Theor. Comput. Sci.* **381** (2007) 234–240.

[10] C. He, H. Lin, J.J. Yuan and Y.D. Mu, Batching machine scheduling with bicriteria: maximum cost and makespan. *Asia-Pac. J. Oper. Res.* **31** (2014) 1–10.

[11] J.A. Hoogeveen, Single-machine Scheduling to minimize a function of two or three maximum cost criteria. *J. Algorithms* **21** (1996) 415–433.

[12] H. Hoogeveen, Multicriteria scheduling. *Eur. J. Oper. Res.* **167** (2005) 592–623.

[13] V. Tkindt and J.-C. Billaut, Multicriteria Scheduling: Theory, Models and Algorithms (2nd edn). Springer-Verlag, Berlin (2006).