Journal of the Operational Research Society (2017) 68, 998–1005

www.palgrave.com/journals

Open

Maximizing total job value on a single machine with job selection

Joonyup Eun[1], Chang Sup Sung[2] and Eun-Seok Kim[3]*

[1]*Department of Anesthesiology, School of Medicine, Vanderbilt University Medical Center, 1211 21st Avenue South, Nashville, TN 37212, USA;* [2]*Department of Industrial and System Engineering, Korea Advanced Institute of Science and Technology (KAIST), 291 Daehak-ro, Yuseong-gu, Daejeon, South Korea; and* [3]*Department of Management, Leadership and Organisations, Middlesex University, London, UK*

This paper describes a single-machine scheduling problem of maximizing total job value with a machine availability constraint. The value of each job decreases over time in a stepwise fashion. Several solution properties of the problem are developed. Based on the properties, a branch-and-bound algorithm and a heuristic algorithm are derived. These algorithms are evaluated in the computational study, and the results show that the heuristic algorithm provides effective solutions within short computation times.

Journal of the Operational Research Society (2017) **68(9),** 998–1005. doi:10.1057/s41274-017-0238-z;
published online 21 June 2017

Keywords: scheduling; job value; stepwise value function; branch-and-bound; heuristic

1. Introduction

It is quite often to have the situation where all the associated jobs are not completed due to a working hour limit or a production capacity limit. However, all the associated jobs are supposed to be completed in traditional scheduling problems with regular performance measures such as makespan, total tardiness and total completion time (Pinedo, 2012). In this research, a non-regular performance measure (i.e., total job value) is presented and maximized on the assumption that a machine does not have to complete all jobs.

The values of jobs in this paper deteriorate over time. It can be used to represent customer satisfaction that deteriorates with the increase in waiting time for the service (Bielen and Demoulin, 2007; van Riel *et al*, 2012; Borges *et al*, 2015).

The proposed situation can be illustrated by a repair service example for air conditioners. During hot summer, a mechanic is not able to accommodate all repair requests for the broken air conditioners within work hours. Based on the data collected, the repair firm knows how customer satisfaction decreases with the increase in waiting time. With this information, the repair firm may be interested in maximizing total satisfaction of the customers assigned to a mechanic.

This research is based on two broad branches of scheduling literature: job selection and job deterioration.

*Correspondence: Eun-Seok Kim, Department of Management, Leadership and Organisations, Middlesex University, London, UK.
E-mail: e.kim@mdx.ac.uk

A thorough review on job selection and scheduling literature is given in Slotnick (2011) and Shabtay *et al* (2013). Several topics are introduced to capture a characteristic of the literature. Job selection was mainly used to maximize total profit (or revenue) under a limited processing capacity (Lin and Ying, 2013, 2015; Reisi-Nafchi *et al*, 2015; Zhang *et al*, 2016). Some papers focused on minimizing the completion time of the last accepted job penalizing the value of rejected jobs (Bartal *et al*, 2000; Zhong *et al*, 2014; Ou *et al*, 2015). A few papers considered the trade-off between the cost to use an expensive machine and the service level. The service level increases as the processing time on the expensive machine increases. Therefore, jobs are selected to find the maximal trade-off (Thevenin *et al*, 2015, 2016).

As seen in the above-referenced work, job selection is employed to consider a production capacity limit or/and a time-related penalty. Likewise, this paper selects jobs to consider time-dependent job value with a machine availability constraint.

The scheduling literature describes the deterioration of jobs in two different ways. The first way is to define jobs whose processing times increase as their delays for processing increase (Voutsinas and Pappis, 2010). Most work in the area of deteriorating jobs focused on this description (Wang *et al*, 2011). However, like the repair service example for air conditioners, the deterioration of jobs may not increase the processing times of jobs. Therefore, another way to describe the deterioration of jobs appeared in the literature, that is, to define jobs whose values decrease over time. In Voutsinas and Pappis (2002, 2010), the values of jobs decrease in an

exponential fashion. In Janiak and Krysiak (2007), the values of jobs decrease in a stepwise fashion.

Following Janiak and Krysiak (2007), this paper uses a stepwise value function, taking advantage of its robustness in approximating any type of value functions. However, Janiak and Krysiak (2007) assume that all jobs should be completed, while this paper employs job selection with a machine availability constraint that may be in practice.

The scheduling problem is described in Section 2. Some solution properties of the problem are developed in Section 3. Based on the properties, a branch-and-bound algorithm and a heuristic algorithm are derived in Sections 4 and 5, respectively. The numerical experiments to evaluate the performance of the algorithms are presented in Section 6. Finally, some concluding remarks are made in Section 7.

2. Problem description

A set of jobs $J = \{1, \ldots, n\}$ is to be scheduled for processing on a single machine where only one job is allowed at a time and the time horizon of from time 0 to time T is considered, and T is less than the sum of processing times of all jobs. All jobs in J are independent, non-preemptive and available for processing at time zero. Each job $j \in J$ has its processing time $p_j > 0$ and its job value $V_j(C_j)$ which is given as a non-increasing stepwise function represented by the completion time of job j $C_j > 0$ and the same moments of change $e_t > 0$, $t = 1, \ldots, k-1$ for all jobs in the situation where at least one job in J decreases in its value. $V_j(C_j)$ is defined as follows:

$$V_j(C_j) = \begin{cases} V_{j1}, & 0 < C_j \le e_1 \\ V_{j2}, & e_1 < C_j \le e_2 \\ \quad\vdots & \\ V_{jk}, & e_{k-1} < C_j \end{cases}$$

where $V_{j1} \ge V_{j2} \ge \cdots \ge V_{jk} \ge 0$ and $0 < e_1 < e_2 < \cdots < e_{k-1}$.

The objective is to find a schedule π that maximizes the total value of jobs (total job value: TJV) completed within the limited machine available time (from time 0 to time T) under the assumption that the considered jobs do not need all to be processed.

Since the problem under consideration with constant job values that do not depend on the job completion times is equivalent to the problem $1 \,|\, d_j = d \,|\, \sum w_j U_j$ which is proved NP-hard by Karp (1972), the proposed problem with stepwise job values is also NP-hard.

3. Problem analysis

Denote by σ the assigned partial schedule and σ' the set of jobs not in σ but in J. Let h denote a time point t in the interval $(e_{t-1}, e_t]$ where $t = 1, 2, \ldots, k$, $e_0 = 0, e_k = T$, which satisfies the relation $e_{t-1} < \sum_{j\in\sigma} p_j \le e_t$ and Δ_t denote the difference of job values between two jobs $i, j \in J$ during the interval $(e_{t-1}, e_t]$.

Property 1 *Given that two unscheduled jobs $i \in J$ and $j \in J$ satisfy the following conditions simultaneously, job i precedes job j in the optimal schedule when the optimal schedule contains both jobs, while only job i is selected in the optimal schedule when the optimal schedule contains one job out of two jobs i and j:*

(a) $V_{it} \ge V_{jt}$, $t = h, h+1, \ldots, k$, and $i, j \in \sigma'$
(b) $\Delta_h \ge \Delta_{h+1} \ge \cdots \ge \Delta_k$
(c) $p_i \le p_j$, $i, j \in \sigma'$

Proof Consider the following notations:

π_u = partial schedule, $u = 1, 2, 3$,
t_{π_u} = sum of the job processing times of π_u,
W_{π_u} = TJV of π_u.

Case 1 (when schedules contains both i and j).
Given two feasible schedules as shown in Figure 1, say $(\sigma\pi_1 i\pi_2 j\pi_3)$ and $(\sigma\pi_1 j\pi_2 i\pi_3)$, TJV of $(\sigma\pi_1 i\pi_2 j\pi_3) = W_\sigma + W_{\pi_1} + V_i(t_\sigma + t_{\pi_1} + p_i) + W_{\pi_2} + V_j\ (t_\sigma + t_{\pi_1} + p_i + t_{\pi_2} + p_j) + W_{\pi_3}$ and TJV of $(\sigma\pi_1 j\pi_2 i\pi_3) = W'_\sigma + W'_{\pi_1} + V_j(t_\sigma + t_{\pi_1} + p_j) + W'_{\pi_2} + V_i(t_\sigma + t_{\pi_1} + p_j + t_{\pi_2} + p_i) + W'_{\pi_3}$. Since the start times of σ, π_1 and π_3 in $(\sigma\pi_1 i\pi_2 j\pi_3)$ are equivalent to the start times of σ, π_1 and π_3 in $(\sigma\pi_1 j\pi_2 i\pi_3)$, respectively, $W_\sigma = W'_\sigma$, $W_{\pi_1} = W'_{\pi_1}$ and $W_{\pi_3} = W'_{\pi_3}$. Since $t_\sigma + t_{\pi_1} + p_i \le t_\sigma + t_{\pi_1} + p_j$ ($\because p_i \le p_j$ (condition c) and each job's value in π_2 is non-increasing in its completion time, $W_{\pi_2} \ge W'_{\pi_2}$. Since $V_{it} \ge V_{jt}$ (condition a), $\Delta_h \ge \Delta_{h+1} \ge \cdots \ge \Delta_k$ (condition b) and $p_i \le p_j$ (condition c), $V_i(t_\sigma + t_{\pi_1} + p_i) + V_j(t_\sigma + t_{\pi_1} + p_i + t_{\pi_2} + p_j) \ge V_j(t_\sigma + t_{\pi_1} + p_j) + V_i(t_\sigma + t_{\pi_1} + p_j + t_{\pi_2} + p_i)$. By the above three results, TJV of $(\sigma\pi_1 i\pi_2 j\pi_3) \ge$ TJV of $(\sigma\pi_1 j\pi_2 i\pi_3)$.

Case 2 (when schedules contain only one job out of i and j).
Given two feasible schedules, say $(\sigma\pi_1 i\pi_2)$ and $(\sigma\pi_1 j\pi_2)$, TJV of $(\sigma\pi_1 i\pi_2) = W_\sigma + W_{\pi_1} + V_i\ (t_\sigma + t_{\pi_1} + p_i) + W_{\pi_2}$ sand TJV of $(\sigma\pi_1 j\pi_2) = W'_\sigma W'_{\pi_1} + V_j(t_\sigma + t_{\pi_1} + p_j) + W'_{\pi_2}$. Since the start times of σ and π_1 in $(\sigma\pi_1 i\pi_2)$ are equivalent to the start times of σ and π_1 in $(\sigma\pi_1 j\pi_2)$, respectively, $W_\sigma = W'_\sigma$ and $W_{\pi_1} = W'_{\pi_1}$. Since $t_\sigma + t_{\pi_1} + p_i \le t_\sigma + t_{\pi_1} + p_j$ ($\because p_i \le p_j$ (condition c) and each job's value in π_2 is non-increasing in its completion time, $W_{\pi_2} \ge W'_{\pi_2}$. Since $V_{it} \ge V_{jt}$ (condition a) and $p_i \le p_j$ (condition c), $V_i(t_\sigma + t_{\pi_1} + p_i) \ge V_j(t_\sigma + t_{\pi_1} + p_j)$. By the above three results, TJV of $(\sigma\pi_1 i\pi_2) \ge$ TJV of $(\sigma\pi_1 j\pi_2)$. □

Let Δ_{C_i} denote the difference of job values between two jobs $i, j \in J$ at the completion time of job i in a given schedule and s denote the difference of job values between two jobs $i, j \in J$ at the completion time of job j in a given schedule. The

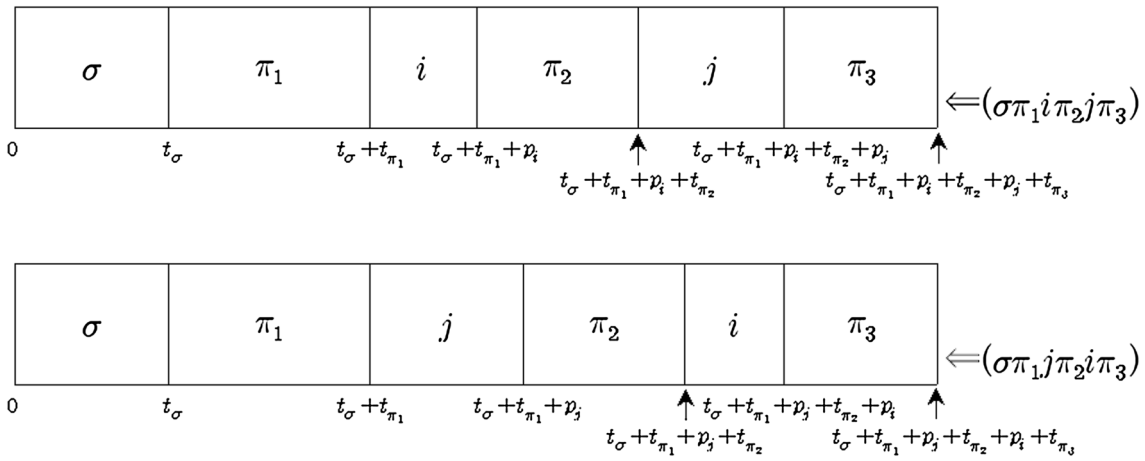

Figure 1 Structure of two schedules $(\sigma\pi_1 i\pi_2 j\pi_3)$ and $(\sigma\pi_1 j\pi_2 i\pi_3)$.

following properties (from Property 2 to Property 5) can be proved in similar way to the proof of Property 1.

Property 2 *Given that the following conditions hold in* $(\pi_1 i\pi_2 j\pi_3)$ *where* π_1, π_2 *and* π_3 *are partial schedules and* $i, j \in J$, *TJV of* $(\pi_1 j\pi_2 i\pi_3)$ *is greater than or equal to TJV of* $(\pi_1 i\pi_2 j\pi_3)$:

(a) $V_j(C_i) \geq V_i(C_i)$ and $V_j(C_j) \geq V_i(C_j)$
(b) $\Delta_{C_i} \geq \Delta_{C_j}$
(c) $p_i \geq p_j$

Property 3 *Given that the following conditions hold in the given schedule* $(\pi_1 i\pi_2 j\pi_3)$ *where* π_1, π_2 *and* π_3 *are partial schedules and* $i, j \in J$, *TJV of* $(\pi_1 j\pi_2 i\pi_3)$ *is greater than or equal to TJV of* $(\pi_1 i\pi_2 j\pi_3)$:

(a) $V_i(C_i) \geq V_j(C_i)$ and $V_i(C_j) \geq V_j(C_j)$
(b) $\Delta_{C_i} \leq \Delta_{C_j}$
(c) $p_i \leq p_j$

Property 4 *Given that the following conditions hold in the given schedule* $(\pi_1 ij\pi_2)$ *where* π_1 *and* π_2 *are partial schedules and* $i, j \in J$, *TJV of* $(\pi_1 ji\pi_2)$ *is greater than or equal to TJV of* $(\pi_1 ij\pi_2)$:

(a) $V_j(C_i) > V_i(C_i)$ and $V_j(C_j) \geq V_i(C_j)$
(b) $\Delta_{C_i} > \Delta_{C_j}$
(c) $p_i \leq p_j$
(d) *job* j *in* $(\pi_1 ji\pi_2)$ *and job* i *in* $(\pi_1 ij\pi_2)$ *are completed during the same interval* $(e_{t-1}, e_t]$ *where* $t=1, 2, \ldots, k$, $e_0 = 0, e_k = T$.

Property 5 *Given that the following conditions hold in the given schedule* $(\pi_1 ij\pi_2)$ *where* π_1 *and* π_2 *are partial schedules and* $i, j \in J$, *TJV of* $(\pi_1 ji\pi_2)$ *is greater than or equal to TJV of* $(\pi_1 ij\pi_2)$:

(a) $V_i(C_i) > V_j(C_i)$ and $V_i(C_j) \geq V_j(C_j)$
(b) $\Delta_{C_i} < \Delta_{C_j}$
(c) $p_i \leq p_j$
(d) *job* j *in* $(\pi_1 ji\pi_2)$ *and job* i *in* $(\pi_1 ij\pi_2)$ *are completed during the same interval* $(e_{t-1}, e_t]$ *where* $t=1, 2, \ldots, k$, $e_0 = 0, e_k = T$.

This analysis implies that when two jobs i and j in a schedule satisfy a property out of 4 properties (from Property 2 to Property 5), the schedule can increase TJV (or remain the same) interchanging the positions of i and j.

4. Branch-and-bound

This section derives a branch-and-bound algorithm, referring to Baker (1974) and Pinedo (2012).

4.1. Upper bound

Let P_σ^K represent a subproblem at level K where σ specifies the assigned partial schedule in a branching tree and K specifies the number of jobs in the partial schedule σ. P_σ^K is the same as the original problem P^0 except with the first K positions assigned in the partial schedule σ. Let s denote the last job of the partial schedule σ, C_s denote the completion time of the last job of the partial schedule σ, g denote a time point t in the interval $(e_{t-1}, e_t]$ where $t = 1, 2, \ldots, k, e_0 = 0$, $e_k = T$, which satisfies the relation $e_{t-1} < C_s \leq e_t$, and σi denote the partial schedule in which the partial schedule σ is immediately succeeded by job $i \in J$.

In order to find an upper bound at the subproblem P_σ^K, the algorithm modifies the values of all jobs in set σ' first. For each job $j \in \sigma'$, all values in the interval $(e_{t-1}, e_t]$ where $t = g+1, g+2, \ldots, k, e_k = T$ are converted to V_{jg} which is the value of job j in the interval $(e_{g-1}, e_g]$.

Then, a knapsack problem to maximize TJV with all jobs, having the modified job values, in set σ' and the available time (capacity) constraint given as $T - C_s$ is derived. This problem is denoted by Problem ①. To solve Problem ①, Dantzig's upper bound in Dantzig (1957) and Martello *et al* (2000) is adopted as follows:

Let n' denote the number of jobs in set σ' and $[j]$ denote the job at jth position in a arbitrary sequence.

1. Arrange the jobs in set σ' as $V_{[j]g}/p_{[j]} \geq V_{[j+1]g}/p_{[j+1]}$ where $j = 1, \ldots, n' - 1$.
2. Find l which is the greatest integer such that $\sum_{j=1}^{l} p_{[j]} \leq T - C_s$.
3. Calculate an upper bound of Problem ①.
   Upper bound of Problem ① $= \sum_{j=1}^{l} V_{[j]g} + [(T - C_s - \sum_{j=1}^{l} p_{[j]} \times V_{[l+1]g}/p_{[l+1]})]$.

Thus, the upper bound at the subproblem P_σ^K is obtained as follows:

$$\mathrm{UB}_1 = \sum_{j \in \sigma} V_j(C_j) + \text{Dantzig upper bound of Problem ①.}$$

For the second upper bound at the subproblem P_σ^K, the values of all jobs in set σ' are also changed but in a different way. For each job $j \in \sigma'$, the job value in each interval $(e_{t-1}, e_t]$ where $t = g, g+1, \ldots, k, e_k = T$ is converted to the maximum value, in each interval $(e_{t-1}, e_t]$ where $t = g, g+1, \ldots, k$, of all jobs in set σ'.

After the changes in the job values, the single-machine scheduling problem that maximizes the TJV with all jobs, having the same job values, in set σ' and machine available time $(T - C_s)$ is derived. This problem is denoted by Problem ②.

Lemma 1 *The optimal solution of Problem ② is obtained in SPT sequence.*

Proof Suppose that π^* is the optimal schedule that contains job u and job v which is the successor of u, and does not contain job w, where $p_u > p_v > p_w$.

Case 1 (Comparison between u and v).
Set $k = u$ and $l = v$, then follows the proof of Janiak and Krysiak (2007)'s Property 1.

Case 2 (Comparison between u and w).
Since $p_u > p_w$, the completion times of the jobs processed after u will decrease after replacing job u with job w. Since $V_j(C_j)$ is a non-increasing function, the values of the jobs processed after job u increases (or remains the same) after the replacement. Moreover, the completion times of the jobs processed before job u remain the same. Therefore, replacing job u with job w can increase the TJV of π^*. This contradicts the assumption.

By the results of Case 1 and Case 2, without loss of generality, SPT rule give the optimal solution of Problem ②. □

The second upper bound for subproblem P_σ^K is obtained as follows:

$$\mathrm{UB}_2 = \sum_{j \in \sigma} V_j(C_j) + \text{TJV of the SPT sequence in Problem ②.}$$

While UB_2 gives a tighter bound when the difference of job values within each $(e_{t-1}, e_t]$ is small, UB_1 gives a tighter bound when the decreasing rate of job values is small. The upper bound of subproblem P_σ^K is defined as:

$$\mathrm{UB} = \min\{\mathrm{UB}_1, \mathrm{UB}_2\}.$$

4.2. Branching

A subproblem (P^0 or P_σ^K) is partitioned into one or more subproblems (P_i^1 or $P_{\sigma i}^{K+1}$) that are defined by sequencing job i in set σ' right after the partial schedule σ if the partial schedule σi satisfies the available time constraint. If no job in set σ' satisfies the available time constraint at the subproblem P_σ^K, P_σ^K is a ending node and the partial schedule σ is called a trial solution which updates LB (lower bound).

To select a subproblem for branching, the depth-first rule is adopted to select the subproblem with the largest K and, in the case of tie, the best-first rule is adopted to select the subproblem with the largest UB. Moreover, properties (from Property 1 to Property 5) will be used as branching rules.

4.3. Bounding

With the original problem P^0, the initial LB is computed by a heuristic which will be explained in Section 5. The algorithm calculates the UB of the subproblem $P_{\sigma i}^{K+1}$ at the subproblem P_σ^K. If the UB is greater than the current LB, the subproblem $P_{\sigma i}^{K+1}$ will be branched from the subproblem P_σ^K. Otherwise, the subproblem $P_{\sigma i}^{K+1}$ will be fathomed. Moreover, when the algorithm finds a new LB, the subproblems whose UBs are not greater than the new LB are also fathomed.

4.4. Overall procedure of branch-and-bound algorithm

Step 1 Obtain the initial LB and trial solution by the heuristic (see Section 5). Place the original problem P^0 on active list and go to Step 2

Step 2 If active list $= \emptyset$, go to Step 7. Otherwise, remove the first P_σ^K from active list. If $\exists i \in \sigma'$ such that $\sum_{j \in \sigma} p_j + p_i \leq T$, then go to Step 3. Otherwise, update trial solution such that trial solution $= \sigma$ and LB and then go to Step 6

Step 3 Check the conditions of Property 1 for all pairs of jobs included in σ'. If the conditions hold, the jobs which satisfy the conditions of job j in Property 1 are eliminated from σ'. Calculate UB of $P_{\sigma i}^{K+1}$ for each job $i \in \sigma'$ such that $\sum_{j\in\sigma} p_j + p_i \leq T$. If UB of $P_{\sigma i}^{K+1}$ is less than or equal to the current LB, $P_{\sigma i}^{K+1}$ is fathomed and go to Step 2. If the number of jobs in σ is zero, go to Step 5. Otherwise, set $q = 1$ and go to Step 4

Step 4 Check the conditions of from Property 2 to Property 5 with qth scheduled job and job i in σi. If the conditions of any one of those properties hold, $P_{\sigma i}^{K+1}$ is fathomed and go to Step 2. If q is the number jobs in σ, go to Step 7. Otherwise, set $q = q + 1$ and repeat Step 4

Step 5 Place $P_{\sigma i}^{K+1}$ on active list and rank subproblems by the criteria which are the largest K and, in the case of tie, the largest UB, and go to Step 2

Step 6 Eliminate the subproblems whose UB $\leq$ LB from active list and go to Step 2

Step 7 Terminate the algorithm. The lastly updated trial solution is optimal

5. Heuristic

The proposed heuristic algorithm consists of two parts. One part is job selection and allocation mechanism. The other one is the interchange mechanism of job positions.

5.1. Job selection and allocation

Let A denote the set of jobs which can be assigned to the partial schedule σ at a dispatching point (a completion time of the partial schedule σ). The algorithm selects and assigns a job which has the largest job value divided by its processing time in the situation where each job in set A is processed right after the partial schedule σ. Then the algorithm can give the best solution from the dispatching point to the completion time of the selected job. Moreover, any two jobs in the partial schedule σ which is made by the above job selection mechanism do not satisfy the conditions of Property 1 and Property 2, because these properties deal with the specific situation such that the jobs having smaller processing times and larger job values precede any other jobs having larger processing times and smaller job values in the optimal solution.

5.2. Interchange of job positions

Given the partial schedule σ, the algorithm can increase the associated TJV by Property 3, Property 4 and Property 5. If a new job is assigned to the partial schedule σ, the conditions of those properties are checked with the new assigned job and any other one job existing in the partial schedule σ until all pair jobs in the partial schedule σ do not satisfy the conditions of all those properties. If the existing job moves backward in its position by interchanging mechanism, the algorithm should check the conditions of Property 2 with the backward-position-changed job and the jobs which are scheduled between the previous and the new position of the backward-position-changed job, by the fact that the new position of the backward-position-changed job was not assigned by job selection mechanism.

5.3. Overall procedure of heuristic algorithm

Step 1 Put all the jobs not included in σ into σ' and put all $i \in \sigma'$ such that $\sum_{j\in\sigma} p_j + p_i \leq T$ into A. If the number of jobs in A is greater than zero, select $q \in \sigma'$ such that $q = \arg\max_{i\in A} V_{it}/p_i$, where the time point t in the interval $(e_{t-1}, e_t]$, $t = 1, 2, \ldots, k$, $e_0 = 0$, $e_k = T$ satisfies the relation $e_{t-1} < \sum_{j\in\sigma} p_j + p_i \leq e_t$, and assign job q to the last position in σ. Otherwise, go to Step 9. If the number of jobs in σ is 1, repeat Step 1. Set $v =$ the number of jobs in σ and $u = v - 1$. Let $[j]_\sigma$ denote the job at j th position in σ. Add $[v]_\sigma$ to List1 and go to Step 2

Step 2 Check $[u]_\sigma$ and $[v]_\sigma$ whether they satisfy the conditions of Property 3, Property 4 and Property 5, respectively. If the conditions of any one of those properties hold, go to Step 3. Otherwise, go to Step 4

Step 3 Add $[u]_\sigma$ to List2 and let the position of $[u+1]_\sigma$ be a breakpoint. If $[u]_\sigma$ is on a breakpoint or the first scheduled job, remove the job index of $[v]_\sigma$ from List1 or List2, interchange the positions of $[u]_\sigma$ and $[v]_\sigma$, and go to Step 6. Otherwise, interchange the positions of $[u]_\sigma$ and $[v]_\sigma$, and go to Step 5

Step 4 If $[u]_\sigma$ is on a breakpoint or the first scheduled job, remove $[v]_\sigma$ from List1 or List2 and go to Step 6. Otherwise, set $u = u - 1$ and go to Step 2

Step 5 If List1 $= \emptyset$, go to Step 6. Otherwise, find any one job z in List1. Set $u =$ the number of jobs scheduled before job z and $v = u + 1$. Go to Step 2

Step 6 If List2 $= \emptyset$, go to Step 1. Otherwise, find any one job y in List2. Set $u =$ the number of jobs scheduled before job y and $v = u + 1$. Go to Step 7

Step 7 Check $[u]_\sigma$ and $[v]_\sigma$ whether they satisfy the conditions of from Property 2 to Property 5. If the conditions of any one of those properties hold, go to Step 3. Otherwise, go to Step 8

Step 8 If $[u]_\sigma$ is on a breakpoint or the first scheduled job, remove $[v]_\sigma$ from List1 or List2 and go to Step 6. Otherwise, set $u = u - 1$ and go to Step 7

Step 9 Terminate the algorithm. The schedule σ is the solution

Note that the interchanging mechanism requires $O(n \log n)$ time and the job selection mechanism selects less than n jobs. Thus, the time complexity of this algorithm is $O(n^2 \log n)$.

6. Computational study

For the numerical experiments, the algorithms are coded in C++ language and tested on a personal computer with a 2.40 GHz Intel Core i7-3630QM processor (8 GB RAM) and evaluated by use of several impact factors including the number of jobs (n) and the number of intervals (k). In addition, to determine whether the ranges of p_j and V_{jt} for $t = 1, \ldots, k$ have any influence on the performance of the algorithms, three different range sets are established and shown in Table 1.

Instances were generated referring to the data generation scheme used in Hall and Posner (2001) and Kim *et al* (2009).

Table 1 Three different range sets for generating p_j and V_{jt}

	p_j	V_{jt}
Set 1	[1, 50]	[0, 100]
Set 2	[1, 100]	[0, 50]
Set 3	[1, 100]	[0, 100]

All instances used in this study are available online at https://www.dropbox.com/s/mjv07qzkttxtfn0/Instances.zip?dl=0. Each p_j takes an integer selected randomly from a discrete uniform distribution. k integer values are generated and used for V_{jt} where $V_{i1} \geq \cdots \geq V_{ik}$ which is checked whether it satisfies the assumption that the value of at least one job in J decreases at each moment of change. T takes an integer selected randomly from a discrete uniform distribution under the range $\left[\sum_{j\in J} p_j \Big/ 2, \sum_{j\in J} p_j - 1\right]$. For generating the moments of change, $(k-1)$ integers are selected without duplicating number under the range $[1, T]$ and used for e_t where $e_1 < \cdots < e_{k-1}$.

The algorithms were tested for 60 problem cases of 10 instances each. In Table 2, the third, sixth and ninth columns show the average run times of the branch-and-bound (B-&-B) algorithm to give the optimal solution. As k increases, the computation of the B-&-B algorithm tends to take more time. The procedure of computing UB, in which the number of job values modified for computing UB depends on k, accounts for this tendency.

The fifth, eighth and eleventh columns of Table 2 show the average Gap between the TJV corresponding to the optimal solution of the B-&-B algorithm (TJV of B-&-B) and the TJV corresponding to the heuristic solution (TJV of heuristic):

$$\text{Gap } (\%) = \frac{\text{TJV of B}-\&-\text{B} - \text{TJV of heuristic}}{\text{TJV of B-\&-B}} \times 100.$$

Table 2 Performances of the B-&-B and the heuristic

n	k	$p_j \in [1, 50]$, $V_{jt} \in [0, 100]$			$p_j \in [1, 100]$, $V_{jt} \in [0, 50]$			$p_j \in [1, 100]$, $V_{jt} \in [0, 100]$		
		B-&-B average run time (s)	*Heuristic average run time (s)*	*Average Gap (%)*	*B-&-B average run time (s)*	*Heuristic average run time (s)*	*Average Gap (%)*	*B-&-B average run time (s)*	*Heuristic average run time (s)*	*Average Gap (%)*
5	10	0.00	0.00	1.48	0.00	0.00	2.31	0.00	0.00	1.78
5	20	0.00	0.00	1.32	0.00	0.00	0.98	0.00	0.00	2.25
5	30	0.00	0.00	1.09	0.00	0.00	0.37	0.00	0.00	1.03
5	40	0.00	0.00	0.31	0.00	0.00	1.37	0.00	0.00	2.08
10	10	0.03	0.00	2.82	0.02	0.00	2.77	0.02	0.00	3.74
10	20	0.04	0.00	1.57	0.03	0.00	3.28	0.03	0.00	2.20
10	30	0.04	0.00	2.49	0.03	0.00	2.47	0.04	0.00	3.31
10	40	0.04	0.00	1.08	0.03	0.00	0.90	0.04	0.00	1.29
15	10	2.60	0.00	4.85	2.54	0.00	3.63	4.00	0.00	3.76
15	20	3.05	0.00	2.29	5.27	0.00	3.37	2.95	0.00	2.66
15	30	7.49	0.00	2.71	3.68	0.00	2.31	4.73	0.00	2.40
15	40	5.88	0.00	2.03	5.73	0.00	2.66	5.32	0.00	1.86
20	10	277.61	0.00	5.20	133.24	0.00	5.19	288.63	0.00	5.89
20	20	489.04	0.00	4.04	331.15	0.00	3.43	368.34	0.00	4.02
20	30	947.26	0.00	3.37	430.89	0.00	2.34	309.22	0.00	2.98
20	40	675.44	0.00	2.46	454.51	0.00	2.38	531.23	0.00	3.21
23	10	2676.92	0.00	4.46	1204.90	0.00	4.36	1447.39	0.00	5.33
23	20	4243.53	0.00	3.42	3751.48	0.00	3.09	2769.74	0.00	4.28
23	30	4216.37	0.00	2.79	3117.73	0.00	2.58	6037.83	0.00	3.03
23	40	7928.35	0.00	2.35	6205.54	0.00	3.00	10147.17	0.00	1.99

Table 3 Average number of interchanging job positions by each property in the heuristic

n	k	*Average number of interchanges by Property 2*	*Average number of interchanges by Property 3*	*Average number of interchanges by Property 4*	*Average number of interchanges by Property 5*
5	10	0.00	0.27	0.07	0.13
5	20	0.00	0.07	0.03	0.07
5	30	0.00	0.17	0.00	0.13
5	40	0.00	0.03	0.00	0.03
10	10	0.00	0.60	0.23	0.23
10	20	0.03	0.53	0.17	0.53
10	30	0.00	0.63	0.20	0.63
10	40	0.00	0.37	0.07	0.37
15	10	0.10	2.27	0.33	0.73
15	20	0.03	1.77	0.30	0.87
15	30	0.00	1.27	0.33	0.70
15	40	0.00	0.80	0.33	1.00
20	10	0.10	5.83	0.17	0.93
20	20	0.03	2.97	0.77	1.50
20	30	0.03	2.07	0.80	1.40
20	40	0.10	2.07	0.53	1.47
23	10	0.03	7.60	0.47	1.07
23	20	0.13	3.50	0.43	1.40
23	30	0.00	2.63	0.93	1.07
23	40	0.03	3.03	0.73	2.07

Minimum average Gap is 0.31% and maximum average Gap is 5.89%. The heuristic algorithm gives quite a good solution, whereas the run time of the heuristic algorithm is short. The average Gap shows a trend such that as k increases, the average Gap gets smaller because, with large k, the differences of job values in each interval $(e_{t-1}, e_t]$ are relatively smaller, and thus, the penalty for the non-optimal schedule is small.

In addition, Table 2 shows that there are no distinct trends between three different range sets for p_j and V_{jt}. The performances of the B-&-B and the heuristic algorithms are indifferent to the ranges for p_j and V_{jt}.

Table 3 shows the average number of interchanging job positions by each property in the heuristic algorithm. The number of interchanges by Property 2 is quite small because the job selection mechanism guarantees that any two jobs in the partial schedule σ do not satisfy the conditions of Property 2, and thus, Property 2 is applicable only to the jobs which is moved its position backward by the interchanging mechanism. Property 3 plays an important role in the interchanging mechanism because Property 3 argues that the jobs which have larger processing times and lower job values can precede other job which has smaller processing times and higher job values in the optimal solution, which is the opposite argument of the job selection mechanism. In the same reason, Property 5 is used more frequently than Property 4. Thus, it is concluded that the job selection mechanism and properties complement each other to give a better solution. Since Property 4 and Property 5 are applicable only to the adjacent pair jobs in the partial schedule σ, the number of interchanging job positions by those properties is smaller than the number of interchanging job positions by Property 3.

7. Conclusion

This paper considers a single-machine scheduling problem in which the value of each job decreases over time in a stepwise fashion, and jobs need to be selected for processing due to a machine availability constraint. The problem can be applied to making a service sequence that maximizes total customer satisfaction with limited working hours. Solution properties of the problem are developed. Those properties are used for the branch-and-bound algorithm and the heuristic algorithm to efficiently explore the solution space without debasing the quality of solutions. The computational study shows that the heuristic algorithm is efficient and effective comparing with the performance of the branch-and-bound algorithm. Therefore, the heuristic algorithm may be applied to large-size problems.

As further study, it would be interesting to consider a multiple machine scheduling problem with machine availability constraints. In addition, considering resumable and non-resumable cases under multiperiod limited machine availability is also an interesting issue.

Acknowledgements—The authors sincerely thank the editors and two anonymous reviewers for their insightful comments and suggestions to improve the quality of this paper.

References

Baker KR (1974). *Introduction to Sequencing and scheduling*. Wiley: New York.

Bartal Y, Leonardi S, Marchetti-Spaccamela A, Sgall J and Stougie L (2000). Multiprocessor scheduling with rejection. *SIAM Journal on Discrete Mathematics* **13**(1):64–78.

Bielen F and Demoulin N (2007). Waiting time influence on the satisfaction-loyalty relationship in services. *Managing Service Quality: An International Journal* **17**(2):174–193.

Borges A, Herter MM and Cheat JC (2015). It was not that long!: the effects of the in-store TV screen content and consumers emotions on consumer waiting perception. *Journal of Retailing and Consumer Services* **22**:96–106.

Dantzig GB (1957). Discrete-variable extremum problems. *Operations Research* **5**(2):266–277.

Hall NG and Posner ME (2001). Generating experimental data for computational testing with machine scheduling applications. *Operations Research* **49**(6):854–865.

Janiak A and Krysiak T (2007). Single processor scheduling with job values depending on their completion times. *Journal of Scheduling* **10**(2):129–138.

Karp RM (1972). Reducibility among combinatorial problems. In Miller RE and Thatcher JW (Eds). *Complexity of Computer Computations*. Plenum Press: New York, pp 85–103.

Kim ES, Sung CS and Lee IS (2009). Scheduling of parallel machines to minimize total completion time subject to s-precedence constraints. *Computers & Operations Research* **36**(3): 698–710.

Lin SW and Ying KC (2013). Increasing the total net revenue for single machine order acceptance and scheduling problems using an artificial bee colony algorithm. *Journal of the Operational Research Society* **64**(2):293–311.

Lin SW and Ying KC (2015). Order acceptance and scheduling to maximize total net revenue in permutation flowshops with weighted tardiness. *Applied Soft Computing* **30**:462–474.

Martello S, Pisinger D and Toth P (2000). New trends in exact algorithms for the 0–1 knapsack problem. *European Journal of Operational Research* **123**(2):325–332.

Ou J, Zhong X and Wang G (2015) An improved heuristic for parallel machine scheduling with rejection. *European Journal of Operational Research* **241**(3):653–661.

Pinedo M (2012). *Scheduling: Theory, Algorithms, and Systems*, fourth ed. Prentice Hall: New York, NJ.

Reisi-Nafchi M and Moslehi G (2015) Integrating two-agent scheduling and order acceptance problems to maximise total revenue by bounding each agent penalty function. *International Journal of Services and Operations Management* **20**(3):358–384.

Shabtay D, Gasper N and Kaspi M (2013) A survey on offline scheduling with rejection. *Journal of Scheduling* **16**(1):3–28.

Slotnick SA (2011). Order acceptance and scheduling: a taxonomy and review. *European Journal of Operational Research* **212**(1):1–11.

Thevenin S, Zufferey N and Widmer M (2015). Metaheuristics for a scheduling problem with rejection and tardiness penalties. *Journal of Scheduling* **18**(1):89–105.

Thevenin S, Zufferey N and Widmer M (2016). Order acceptance and scheduling with earliness and tardiness penalties. *Journal of Heuristics* **22**(6):849–890.

van Riel ACR, Semejin J, Ribbink D and Bomert-Peters Y (2012). Waiting for service at the checkout: negative emotional responses, store image and overall satisfaction. *Journal of Service Management* **23**(2):144–169.

Voutsinas TG and Pappis CP (2002). Scheduling jobs with values exponentially deteriorating over time. *International Journal of Production Economics* **79**(3):163–169.

Voutsinas TG and Pappis CP (2010). A branch and bound algorithm for single machine scheduling with deteriorating values of jobs. *Mathematical and Computer Modelling* **52**(1):55–61.

Wang JB, Wang JJ and Ji P (2011). Scheduling jobs with chain precedence constraints and deteriorating jobs. *Journal of the Operational Research Society* **62**(9):1765–1770.

Zhang X, Zhou S, de Koster R and van de Velde S (2016). Increasing the revenue of self-storage warehouses by optimizing order scheduling. *European Journal of Operational Research* **252**(1):69–78.

Zhong X, Ou J and Wang G (2014). Order acceptance and scheduling with machine availability constraints. *European Journal of Operational Research* **232**(3):435–441.

Received 26 October 2015;
accepted 18 April 2017