ELSEVIER

Operations Research Letters 30 (2002) 343–350

Operations Research Letters

www.elsevier.com/locate/dsw

A multiprocessor task scheduling model for berth allocation: heuristic and worst-case analysis

Yongpei Guan[a,1], Wen-Qiang Xiao[b], Raymond K. Cheung[b,*], Chung-Lun Li[b]

[a] *School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA*
[b] *Department of Industrial Engineering and Engineering Management, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong*

Received 2 March 2001; received in revised form 2 August 2001; accepted 10 January 2002

Abstract

We consider a scheduling problem in which the processors are arranged along a straight line, and each job requires simultaneous processing by multiple consecutive processors. We assume that the job sizes and processing times are agreeable. Our objective is to minimize the total weighted completion time of the jobs. This problem is motivated by the operation of berth allocation, which is to allocate vessels (jobs) to a berth with multiple quay cranes (processors), where a vessel may be processed by multiple consecutive cranes simultaneously. We develop a heuristic for the problem and perform worst-case analysis.

Keywords: Scheduling; Sequencing; Heuristic

1. Introduction

In this paper we consider a machine scheduling problem in which a job may require processing by several processors simultaneously. This type of problems has been called *multiprocessor task scheduling* in the literature (see [5,10] for recent surveys). These problems can be classified into two categories. The first one considers situations in which each job needs to be processed by a given fixed number of processors simultaneously, but the choices of processors are decision variables. The second category considers situations in which every job requires processing by certain dedicated processors. Our problem belongs to the first category. In addition, we assume that the processors are arranged along a straight line and each job has to be processed by "consecutive" processors. This problem is motivated by the operation of berth allocation in a container terminal, where vessels (jobs) are allocated to a berth with multiple quay cranes (processors). A berth may handle several vessels at the same time. A large vessel may be processed by multiple consecutive quay cranes simultaneously for container loading/unloading operation.

Research on berth allocation has appeared in the literature. This includes the scheduling of cranes along the berth [4], the evaluation of berthing policies via simulation [8], as well as the minimization of berthing costs [1,2], berth length required [12], and the makespan of vessel berthing time [11]. Our

* Corresponding author.
E-mail addresses: guanyp@isye.gatech.edu (Y. Guan), iexiao@ust.hk (W.-Q. Xiao), rcheung@ust.hk (R.K. Cheung), lichung@ust.hk (C.-L. Li).

[1] The author was a student at the Hong Kong University of Science and Technology while this work was being done.

0167-6377/02/$ - see front matter
PII: S0167-6377(02)00147-5

model is similar to the model in [11], but with the different objective of minimizing the total weighted completion time of jobs. Our problem is also a special form of the "general multiprocessor task scheduling" problem [3] in which several given alternatives can be used to process a job. Finally, our problem is similar to the two-dimensional packing problems that have been considered by many authors (see, for example, [6,7]), but we have a different objective function.

To define our problem mathematically, we consider a set of n jobs $\{J_1,\dots,J_n\}$ to be processed by m parallel processors $\{1,\dots,m\}$. Each job J_j $(j=1,\dots,n)$ has a given processing time p_j, a given weight w_j, and a given size s_j, where $p_j, w_j, s_j \in Z^+$ and $s_j \leqslant m$. Each job J_j has to be processed by s_j consecutive processors simultaneously, that is, at any time moment, if processors i and j $(i<j)$ are both processing the same job then processors $i+1, i+2,\dots,j-1$ must be processing that job as well. All jobs are available for processing at time 0, and preemption of jobs is not allowed. Our objective is to assign processors to jobs and to schedule the jobs such that the total weighted completion time $\sum_{j=1}^{n} w_j C_j$ is minimized, where C_j denotes the completion time of job J_j.

In the berth allocation problem, a job represents a vessel. The m parallel processors represent m cranes located along the berth, where the size of a job represents the number of cranes that will serve the vessel simultaneously. The objective of minimizing the total weighted completion time corresponds to the minimization of total weighted waiting time of the vessels. The weight of a job represents the importance of the job and is normally dependent on the vessel size. We consider two cases of job weights: (1) $w_j=\lambda s_j$ where λ is a constant; and (2) $w_j=\lambda s_j^{\rho}$ where $0 \leqslant \rho \leqslant 1$. The first case applies to the situation where the waiting cost of a vessel is linearly proportional to its size. The second case is a more general form for situations where the weight is dependent on vessel size but not necessarily linearly proportional to the size. Note that when $\rho=1$, we have the first special case. When $\rho=0$, w_j becomes a constant and the objective is to minimize the total waiting time of the vessels. We assume that the job processing times and job sizes are "agreeable," that is, for any $i,j=1,\dots,n$, if $p_i<p_j$ then $s_i \leqslant s_j$. This assumption is realistic since a longer vessel (i.e., larger s_j) normally has a larger width, and the cranes will take a longer time to finish the work.

Lee and Cai [9] have shown that the multiprocessor scheduling problem $P2|size_j|\sum C_j$ is NP-hard. Their model is the same as ours, except that they consider only 2 processors and with no restrictions on job processing times and job sizes, while we consider m processors with agreeable job processing times and job sizes. In their construction of the NP-hardness proof, the job processing times and job sizes are agreeable. This implies that when $\rho=0$ in our general case, the problem is NP-hard even when there are only two processors. In fact, a straightforward modification of their NP-hardness proof can show that our first case is NP-hard as well. In other words, the existence of efficient algorithms for solving either of the two cases is highly unlikely. In the following sections, we develop an efficient heuristic solution procedure and perform worst-case analysis.

2. Heuristic and lower bound

In this section we consider the special case of $w_j=\lambda s_j$ for $j=1,\dots,n$. We present a heuristic procedure for solving this problem. In this heuristic, we first arrange the jobs in increasing order of p_j and s_j and divide the jobs into groups. Then we assign the groups of jobs to the processors in a "zig-zagging" greedy fashion. Let t be a group index and G_t be the set of jobs in tth group. The heuristic is described as follows.

Heuristic H.

Step 0: Sort and renumber the jobs such that $p_1 \leqslant p_2 \leqslant \cdots \leqslant p_n$ and $s_1 \leqslant s_2 \leqslant \cdots \leqslant s_n$ (note: p_j's and s_j's are agreeable). Set $t \leftarrow 1$.

Step 1: Let $\{J_\ell, J_{\ell+1},\dots,J_n\}$ be the set of unscheduled jobs. Let

$$u=\max\left\{q \,\middle|\, \sum_{j=\ell}^{q} s_j \leqslant m \quad \text{and } q \leqslant n\right\}.$$

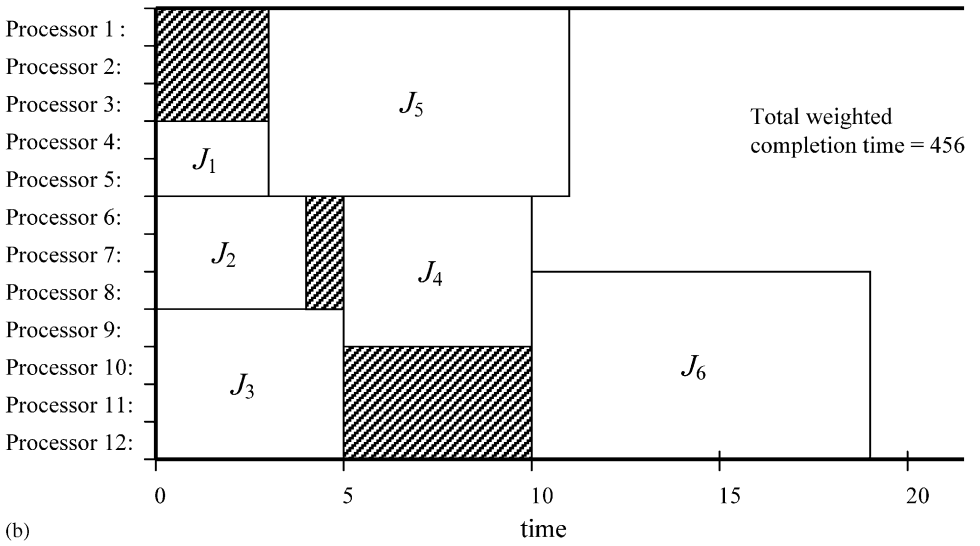
Set $G_t \leftarrow \{J_\ell, J_{\ell+1},\dots,J_u\}$.

Step 2: For $r=\ell,\ell+1,\dots,u$:

(a) if t is odd, then assign J_r to processors $m-\sum_{j=r}^{u} s_j+1, m-\sum_{j=r}^{u} s_j+2,\dots, m-\sum_{j=r+1}^{u} s_j$;
(b) if t is even, then assign J_r to processors $\sum_{j=r+1}^{u} s_j+1, \sum_{j=r+1}^{u} s_j+2,\dots, \sum_{j=r}^{u} s_j$.

(a)

j	1	2	3	4	5	6
s_j	2	3	4	4	5	5
w_j	4	6	8	8	10	10
p_j	3	4	5	5	8	9



(b)

Fig. 1. An example for Heuristic H: (a) given job data, (b) heuristic solution.

Schedule J_r behind the existing scheduled jobs on these processors, and make it start as early as possible.

Step 3: Set $t \leftarrow t+1$. If there is no more unscheduled job, then stop, else go to Step 1.

We now demonstrate Heuristic H by using an example with 12 processors, 6 jobs, and parameters as shown in Fig. 1(a). Note that these jobs are already numbered in such a way that $p_1 \leqslant p_2 \leqslant \cdots \leqslant p_6$ and $s_1 \leqslant s_2 \leqslant \cdots \leqslant s_6$. In this example, $\lambda = w_j/s_j = 2$ for $j = 1,\ldots,6$. At the beginning of the first iteration, the set of unscheduled jobs is $\{J_1,J_2,J_3,J_4,J_5,J_6\}$. Step 1 of Heuristic H determines that $u = 3$ and $G_1 = \{J_1,J_2,J_3\}$. According to Step 2(a), J_1 is assigned to processors 4 and 5; J_2 is assigned to processors 6–8; and J_3 is assigned to processors 9–12. These three jobs will start at time 0. Now, the set of unscheduled jobs is $\{J_4,J_5,J_6\}$. Step 1 determines that $u = 5$ and $G_2 = \{J_4,J_5\}$. According to Step 2(b), J_4 is assigned to processors 6–9 starting at time 5; while J_5 is assigned to processors 1–5 starting at time 3. Next, the set of unscheduled jobs becomes $\{J_6\}$. We get $G_3 = \{J_6\}$ and J_6 is assigned to processors 8–12. The final schedule is depicted in Fig. 1(b). The total weighted completion time of this schedule is $\sum_{j=1}^{6} w_j C_j = (4)(3) + (6)(4) + (8)(5) + (8)(10) + (10)(11) + (10)(19) = 456$.

If the number of processors, m, is fixed, then the running time of this heuristic is dominated by Step 0 and the complexity is $O(n \log n)$. If m is not fixed, then we need to keep track of the information of each processor and the complexity of the heuristic is $O(m + n \log n)$.

Next, we consider a lower bound of the optimal solution value of the problem. Given any instance of our problem, we may construct a corresponding relaxed problem as follows. For every job J_j, we replace it by s_j identical jobs $\{J_{j1},J_{j2},\ldots,J_{js_j}\}$ each of unit size, weight $w_j/s_j = \lambda$, and processing time p_j (see Fig. 2). This relaxed problem has a total of $N = \sum_{j=1}^{n} s_j$ jobs. Denote these new jobs by $\{J_1^{\mathrm{R}},J_2^{\mathrm{R}},\ldots,J_N^{\mathrm{R}}\}$, where the first s_1 jobs in this set come from J_1, the next s_2 jobs

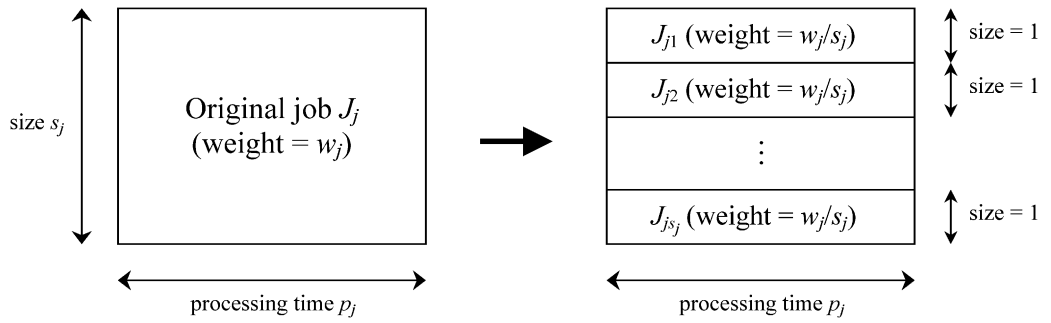


Fig. 2. Construction of the relaxed problem.

come from J_2, and so on. Each of these jobs will be processed by only one processor. Since all jobs in this relaxed problem have the same weight λ, the relaxed problem becomes a traditional parallel machine minimum total completion time problem (i.e., $Pm\|\sum C_j$), which can be solved optimally by the shortest processing time first (SPT) rule [13]. The SPT rule schedules jobs in nondecreasing order of processing times and assigns each job to the earliest available processor. When the SPT rule is applied to the relaxed problem, the resulting solution is to assign jobs $J_1^R, J_2^R, \ldots, J_m^R$ to the 1st position of the processors $1, 2, \ldots, m$, respectively, assign jobs $J_{m+1}^R, J_{m+2}^R, \ldots, J_{2m}^R$ to the 2nd position of processors $1, 2, \ldots, m$, respectively, and so on. Hence, the total weighted completion time of the SPT schedule of the relaxed problem (i.e., the total completion time times λ) is a lower bound on the optimal solution value of the original problem.

3. Worst-case analysis for the case with proportional job weights

In this section we show that the relative error of the solution obtained by Heuristic H must be no more than 100% for the case where $w_j = \lambda s_j$ for $j = 1, \ldots, n$. Let Z^H denote the total weighted completion time of the schedule obtained by Heuristic H. Let Z^* denote the total weighted completion time of the optimal schedule. Let L denote the total weighted completion time of the optimal solution of the relaxed problem. It is clear that L is a lower bound of Z^*, i.e., $L \leqslant Z^*$. Let S_j^H and C_j^H be the start time and completion time, respectively, of J_j in the heuristic solution, for $j = 1, \ldots, n$. Let S_{jh}^R and C_{jh}^R be the start time and completion time, respectively, of J_{jh} in the SPT schedule of the relaxed problem, for $j = 1, \ldots, n$ and $h = 1, \ldots, s_j$. Clearly, for any $1 \leqslant j < k \leqslant n$,

$$S_{j1}^R \leqslant S_{j2}^R \leqslant \cdots \leqslant S_{js_j}^R \leqslant S_{k1}^R \leqslant S_{k2}^R \leqslant \cdots \leqslant S_{ks_k}^R \quad (1)$$

and

$$C_{j1}^R \leqslant C_{j2}^R \leqslant \cdots \leqslant C_{js_j}^R \leqslant C_{k1}^R \leqslant C_{k2}^R \leqslant \cdots \leqslant C_{ks_k}^R (2)$$

For any $j = 1, \ldots, n$, if $S_j^H > 0$, then there must exist some job J_i such that $C_i^H = S_j^H$ and that J_i is assigned to at least one of the processors that process J_j. This is because in Step 2 of Heuristic H, we always make the job start as early as possible. We call this job J_i a "predecessor" of J_j. Note that J_i must be assigned to the group prior to that of J_j. Note also that a job may have multiple predecessors.

Lemma 1. *Suppose J_i is a predecessor of J_j in the solution obtained from Heuristic H. Let $G_t = \{J_\ell, \ldots, J_i, \ldots, J_u\}$ and $G_{t+1} = \{J_{u+1}, \ldots, J_j, \ldots, J_v\}$. Then $\sum_{k=i}^{v} s_k > m$.*

Proof. *Case* 1: t is odd. In Step 2(a), jobs $J_i, J_{i+1}, \ldots, J_u$ are assigned to processors $y, y+1, \ldots, m$ for some y. In the next iteration, we will arrive at Step 2(b) where jobs $J_{u+1}, J_{u+2}, \ldots, J_v$ will get assigned. In particular, jobs $J_j, J_{j+1}, \ldots, J_v$ will get assigned to processors $\bar{y}, \bar{y}-1, \ldots, 1$, for some $\bar{y}$. Since J_i is a predecessor of J_j, jobs J_i and J_j must get assigned to at least one processor in common. This implies that $\bar{y} \geqslant y$ (see Fig. 3(a)). Therefore, we have

$$\sum_{k=i}^{v} s_k \geqslant \sum_{k=i}^{u} s_k + \sum_{k=j}^{v} s_k = (m - y + 1) + \bar{y} \geqslant m + 1 > m.$$

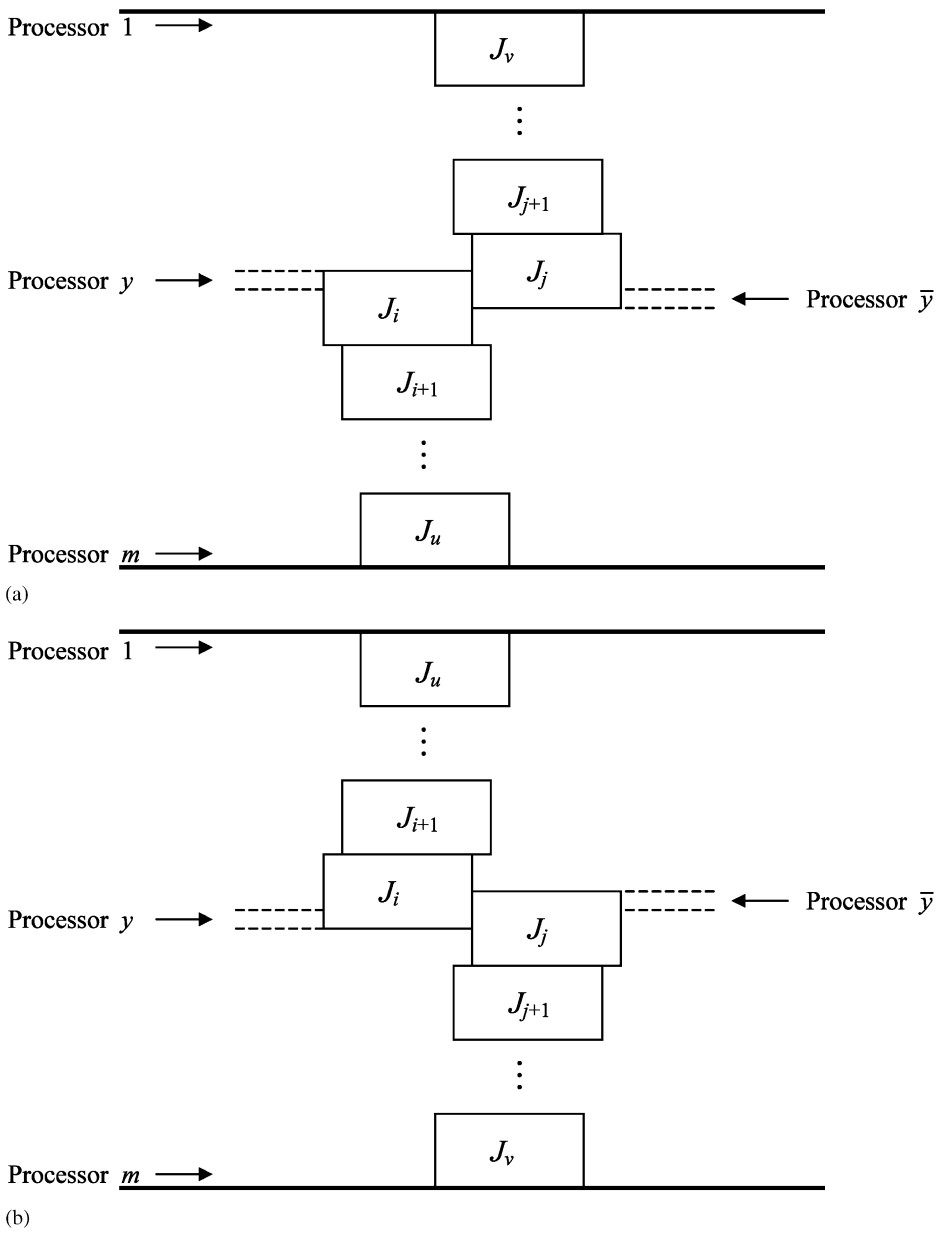


Fig. 3. Proof of Lemma 1.

Case 2: t is even. As shown in Fig. 3(b), suppose jobs $J_i, J_{i+1}, \ldots, J_u$ are assigned to processors $y, y-1, \ldots, 1$ and jobs $J_j, J_{j+1}, \ldots, J_v$ are assigned to processors $\bar{y}, \bar{y}+1, \ldots, m$. Then $\bar{y} \leqslant y$. Following the same argument as in Case 1, we have $\sum_{k=i}^{v} s_k > m$. □

Lemma 2. *In the solution obtained from Heuristic H, if J_i is a predecessor of J_j and J_j is a predecessor of J_k, then $C_{i1}^{R} \leqslant S_{k1}^{R}$.*

Proof. Suppose $J_j \in G_{t+1} = \{J_{u+1}, J_{u+2}, \ldots, J_v\}$. We know that $i < u+1 \leqslant j \leqslant v$. By Lemma 1, the total

size of the jobs $J_i, J_{i+1}, \ldots, J_v$ is greater than m. Thus, these $v-i+1$ jobs will result in more than m unit-sized jobs in the relaxed problem, namely, $J_a^R, J_{a+1}^R, \ldots, J_b^R$, where $b-a+1 > m$. Hence, in the SPT solution of the relaxed problem, there exist two jobs $J_{a'}^R$ and $J_{b'}^R$ with $a \leqslant a' < b' \leqslant b$ where $J_{a'}^R$ is processed immediately before $J_{b'}^R$ by the same processor. Thus, the completion time of $J_{a'}^R$ is equal to the start time of $J_{b'}^R$, implying that the completion time of J_a^R is no greater than the start time of J_b^R, that is

$$C_{i1}^R \leqslant S_{vs_v}^R. \tag{3}$$

Since J_j is a predecessor of J_k, we have $v < k$. From (1), we have

$$S_{vs_v}^R \leqslant S_{k1}^R. \tag{4}$$

Combining (3) and (4) gives us the desired result. □

Lemma 3. *$C_j^H \leqslant 2C_{j1}^R$ for $j = 1, \ldots, n$.*

Proof. We consider a particular job J_j and denote $J_j \equiv J_{\pi_0}$. Let job J_{π_1} be a predecessor of J_{π_0}, job J_{π_2} be a predecessor of J_{π_1}, and so on. Suppose J_{π_α} has a zero start time. Then the job sequence $(J_{\pi_\alpha}, J_{\pi_{\alpha-1}}, \ldots, J_{\pi_1}, J_{\pi_0})$ forms a "critical path" for J_j, and

$$C_j^H = \sum_{i=0}^{\alpha} p_{\pi_i}.$$

Let $\beta = \lfloor \alpha/2 \rfloor$. Since $p_{\pi_i} \leqslant p_{\pi_{i-1}}$ $(i = 1, 2, \ldots, \alpha)$, we have

$$C_j^H \leqslant 2\sum_{i=0}^{\beta} p_{\pi_{2i}}. \tag{5}$$

By Lemma 2,

$$C_{\pi_{2i},1}^R \leqslant S_{\pi_{2i-2},1}^R$$

for $i = 1, 2, \ldots, \beta$. This implies

$$\sum_{i=1}^{\beta} C_{\pi_{2i},1}^R \leqslant \sum_{i=1}^{\beta} S_{\pi_{2i-2},1}^R$$

or equivalently,

$$\sum_{i=1}^{\beta-1} [C_{\pi_{2i},1}^R - S_{\pi_{2i},1}^R] + C_{\pi_{2\beta},1}^R \leqslant S_{\pi_0,1}^R.$$

Note that $C_{\pi_{2i},1}^R - S_{\pi_{2i},1}^R = p_{\pi_{2i}}$ $(i = 1, \ldots, \beta - 1)$ and $C_{\pi_{2\beta},1}^R \geqslant p_{\pi_{2\beta}}$. Thus,

$$\sum_{i=1}^{\beta-1} p_{\pi_{2i}} + p_{\pi_{2\beta}} \leqslant C_{\pi_0,1}^R - p_{\pi_0}$$

or equivalently,

$$\sum_{i=0}^{\beta} p_{\pi_{2i}} \leqslant C_{\pi_0,1}^R.$$

Hence, from (5), we have $C_j^H \leqslant 2C_{\pi_0,1}^R = 2C_{j1}^R$. □

Theorem 4. *If $w_j = \lambda s_j$ for $j = 1, \ldots, n$, then $Z^H/Z^* \leqslant 2$, and this bound is asymptotically tight as m tends to infinity.*

Proof. By Lemma 3 and inequality (2), we have

$$Z^H = \sum_{j=1}^{n} w_j C_j^H = \lambda \sum_{j=1}^{n} s_j C_j^H$$

$$\leqslant 2\lambda \sum_{j=1}^{n} s_j C_{j1}^R \leqslant 2\lambda \sum_{j=1}^{n} \sum_{k=1}^{s_j} C_{jk}^R = 2L \leqslant 2Z^*.$$

To prove that this bound is asymptotically tight, we consider the example with 2 jobs, m processors, $w_1 = s_1 = 1$, $w_2 = s_2 = m$, and $p_1 = p_2 = 1$. Clearly, in this example the job processing times and job sizes are agreeable. Heuristic H will first schedule J_1 and then J_2. The resulting schedule is shown in Fig. 4(a). The total weighted completion time of this schedule is $Z^H = (1)(1) + (m)(2) = 2m + 1$. The optimal solution is depicted in Fig. 4(b), with a total weighted completion time of $Z^* = (m)(1) + (1)(2) = m + 2$. Therefore, $Z^H/Z^* = (2m + 1)/(m + 2) \to 2$ as $m \to \infty$. □

4. The general case

The case with proportional weights (i.e., $w_j = \lambda s_j$, $\forall j$) represents the extreme situation where the importance of a job/vessel is linearly related to the vessel length. The case with equal weights (i.e., $w_j = \lambda$, $\forall j$) represents another extreme situation where all jobs/vessels are equally important. A more

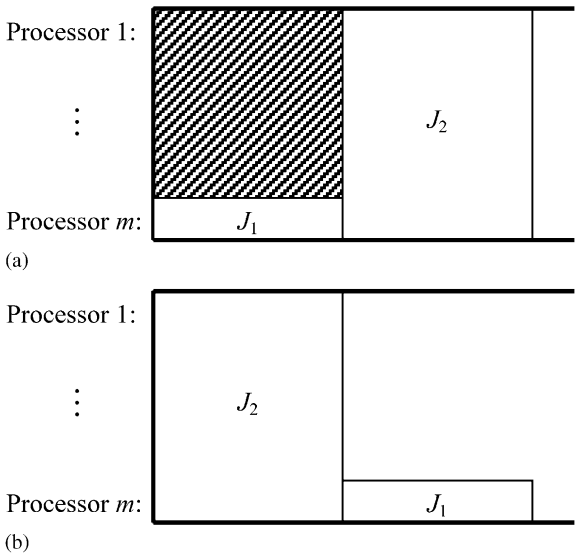


Fig. 4. Worst-case example for the proportional weight case: (a) heuristic solution, (b) optimal solution.

practical situation may lie between these two extreme situations, namely $w_i = \lambda s_i^\rho$, where $0 < \rho < 1$. In the following we show that for the general case of $0 \leqslant \rho \leqslant 1$, Heuristic H still provides a solution with no more than 100% error for any number of processors. In order to prove this result, we first discuss an important property of the parallel machine minimum total weighted completion problem (i.e., $Pm\|\sum w_j C_j$). We consider the Extended Shortest Processing Time first (ESPT) rule defined as follows. The rule schedules jobs in order of nondecreasing processing times, breaks ties by selecting the jobs with the larger weights first, and assigns each job to the earliest available processor.

Lemma 5. *If* $p_1 \leqslant p_2 \leqslant \cdots \leqslant p_N$ *and* $w_1 \geqslant w_2 \geqslant \cdots \geqslant w_N$, *then the ESPT rule is optimal for* $Pm\|\sum w_j C_j$.

Proof. Note that if $p_1 \leqslant p_2 \leqslant \cdots \leqslant p_N$ and $w_1 \geqslant w_2 \geqslant \cdots \geqslant w_N$, then the ESPT rule will schedule jobs in increasing order of job indices, except for those identical jobs (i.e., jobs with the same processing time and weight). Suppose, to the contrary, that the schedule obtained from the ESPT rule is not optimal. Then in the optimal schedule, there exist jobs $J_j, J_{j'}$ such that $j < j'$ and J_j is not identical to $J_{j'}$ but the start time of J_j is greater than that of $J_{j'}$. Let i and i' be the processors that process J_j and $J_{j'}$, respectively. (Note that i and i' may be the same processor.) Let Γ_i be the set of all jobs scheduled behind J_j on processor i, and $\Gamma_{i'}$ be the set of all jobs scheduled behind $J_{j'}$ on processor i'. Let W_i and $W_{i'}$ be the total weight of the jobs in Γ_i and $\Gamma_{i'}$, respectively. Note that either $p_j \leqslant p_{j'}$ and $w_j > w_{j'}$, or $p_j < p_{j'}$ and $w_j \geqslant w_{j'}$.

Case 1: If $W_i < W_{i'}$, then interchanging jobs J_j and $J_{j'}$ will improve the total weighted completion time of the schedule. (Note that in this case, i may or may not be the same as i'.)

Case 2: If $W_i \geqslant W_{i'}$, then interchanging job subsets $\Gamma_i \cup \{J_j\}$ and $\Gamma_{i'} \cup \{J_{j'}\}$ will improve the total weighted completion time of the schedule. (Note that in this case, $i \neq i'$.) Thus, both cases contradict the optimality of the schedule. Therefore, the schedule obtained from the ESPT rule must be optimal. □

Theorem 6. *If* $w_j = \lambda s_j^\rho$ *for* $j = 1, \ldots, n$ *where* $0 \leqslant \rho \leqslant 1$, *then* $Z^{\mathrm{H}}/Z^* \leqslant 2$.

Proof. We consider a relaxation of the problem as follows. For every job J_j, we replace it by s_j identical jobs each of unit size with weight $w'_j = \lambda s_j^{\rho-1}$, and processing time p_j as shown in Fig. 2. This relaxed problem is a parallel machine minimum total *weighted* completion time problem. Let L' denote the total weighted completion time of the optimal solution of this relaxed problem. Clearly, $L' \leqslant Z^*$. Note that since $s_1 \leqslant s_2 \leqslant \cdots \leqslant s_n$, we have the property that $w'_1 \geqslant w'_2 \geqslant \cdots \geqslant w'_n$. By Lemma 5, an optimal solution to this relaxed problem can be obtained by the ESPT rule. Note that the schedule of the relaxed problem obtained by the ESPT rule is actually an SPT schedule, and hence, Lemma 3 and inequality (2) remain valid for this case. Therefore, similar to the proof of Theorem 4, we have $Z^{\mathrm{H}} = \sum_{j=1}^n \lambda s_j^\rho C_j^{\mathrm{H}} \leqslant 2\sum_{j=1}^n \lambda s_j^\rho C_{j1}^{\mathrm{R}} = 2\sum_{j=1}^n w'_j s_j C_{j1}^{\mathrm{R}} \leqslant 2\sum_{j=1}^n w'_j \sum_{k=1}^{s_j} C_{jk}^{\mathrm{R}} = 2L' \leqslant 2Z^*$. □

Theorem 6 states that the relative error of the solution obtained by Heuristic H is at most 100% for the general case. However, it remains an open question of whether this constant error bound is improvable when $\rho \neq 1$.

Acknowledgements

This research was supported in part by Grant HKUST6205/99E from the Research Grants Council of Hong Kong. The authors would like to thank an anonymous referee for his/her valuable comments.

References

[1] G.G. Brown, K.J. Cormican, S. Lawphongpanich, D.B. Widdis, Optimizing submarine berthing with a persistence incentive, Naval Res. Logist. 44 (1997) 301–318.
[2] C.-Y. Chen, T.-W. Hsieh, A time-space network model for the berth allocation problem, First International Conference of Maritime Engineering and Ports, Genoa, Italy, September 28–30, 1998.
[3] J. Chen, C.-Y. Lee, General multiprocessor task scheduling, Naval Res. Logist. 46 (1999) 57–74.
[4] C. Daganzo, The crane scheduling problem, Transportation Res. B 23B (1989) 159–175.
[5] M. Drozdowski, Scheduling multiprocessor tasks—an overview, European J. Oper. Res. 94 (1996) 215–230.
[6] S.P. Fekete, E. Köhler, J. Teich, Higher-dimensional packing with order constraints, Proceedings of the 7th International Workshop on Algorithms and Data Structures, Lecture Notes in Computer Science, Vol. 2125, Springer, Berlin, 2001, pp. 300–312.
[7] E. Hadjiconstantinou, N. Christofides, An exact algorithm for general, orthogonal, two-dimensional knapsack problems, European J. Oper. Res. 83 (1995) 39–56.
[8] K.K. Lai, K. Shih, A study of container berth allocation, J. Adv. Transportation 26 (1992) 45–60.
[9] C.-Y. Lee, X. Cai, Scheduling one and two-processor tasks on two parallel processors, IIE Trans. 31 (1999) 445–455.
[10] C.-Y. Lee, L. Lei, M. Pinedo, Current trends in deterministic scheduling, Ann. Oper. Res. 70 (1997) 1–41.
[11] C.-L. Li, X. Cai, C.-Y. Lee, Scheduling with multiple-job-on-one-processor pattern, IIE Trans. 30 (1998) 433–445.
[12] A. Lim, The berth planning problem, Oper. Res. Lett. 22 (1998) 105–110.
[13] M. Pinedo, Scheduling: Theory, Algorithms, and Systems, 2nd Edition, Prentice-Hall, Upper Saddle River, NJ, 2002.