



On-line scheduling with forbidden zones

K Khammuang¹, A Abdekhodae² and A Wirth^{1*}

¹The University of Melbourne, Victoria, Australia; and ²Commonwealth Scientific and Industrial Research Organisation, Australia

In various manufacturing and computing contexts there may be a certain period in each time interval, during which processing may continue but may not be initiated. We examine the problem of on-line scheduling in the presence of such forbidden zones, whose complements are starting time windows. We show that no on-line algorithm is better than $\frac{9}{7}$ -competitive, when minimizing the number of intervals used (essentially the makespan), whereas list scheduling is shown to be 2-competitive. We also investigate adaptations of the first fit, next fit and harmonic bin packing algorithms and test all four empirically.

Journal of the Operational Research Society (2007) **58**, 80–90. doi:10.1057/palgrave.jors.2602124

Published online 11 January 2006

Keywords: scheduling; on-line algorithms; competitive analysis

Introduction

On-line approaches have become increasingly important in the field of scheduling. We frequently encounter problems where only partial information is provided but fast and effective solutions are required. Such problems are particularly common in computing and manufacturing environments. In computer science, there are many instances when tasks are to be scheduled on servers yet complete information such as their arrival times, lengths or due-dates are not available in advance. In manufacturing systems, there are also instances when accurate information about processes is not available or optimization methods are so complex as to make it undesirable for decision makers to implement them. Furthermore, in some situations it may be necessary to instantly inform customers placing orders of the completion time of their jobs without having any information about future incoming orders. In such cases, it may be preferable to have some simple rules to assist a decision maker, for example a shopfloor manager, to decide and prioritize a set of tasks virtually instantly. Dispatch rules are frequently mentioned in operations management literature, these provide fast, simple and effective solution methods for many problems. The problem outlined below is one example of a situation where on-line methods are highly useful.

The motivation for considering scheduling with starting time windows, whose complements are *forbidden zones*, arose from work by Abdekhodae on a supply chain problem. In that problem a certain activity (ship berthing) was not allowed to commence inside particular time intervals.

More generally, in some scheduling problems processing may continue but may not be initiated during certain intervals. For example, this will occur if a particular service, such as an operator's attendance is required at the initiation and the completion of certain tasks but not throughout the whole process. The objective is to carry out all tasks in minimum time, given that the operator's attendance is limited to certain time periods.

We begin by recalling some definitions and results from an earlier complementary paper (Abdekhodae and Ernst, 2004a), which studied the off-line version of this problem.

Denote the n jobs to be scheduled by $J_1 \dots J_n$. Let p_i be the processing time of J_i . Furthermore, we shall assume from now on that $p_i \leq 1 \forall i$. We partition time into a set of abutting intervals $I = \{I_1, I_2, \dots, I_s\}$ with $I_1 = [0, 2]$, $I_2 = [2, 4]$, $I_s = [2s-2, 2s]$. Each I_j includes a corresponding *forbidden zone* $F_j \subset I_j \forall j$, where $F_1 = (1, 2)$, $F_2 = (3, 4) \dots F_s = (2s-1, 2s]$. We call the intervals $I_j \setminus F_j$ the *allowed zones*. Below, when we refer to *intervals* we shall mean the sets I_j .

Forbidden zones represent time intervals during which a job cannot be started, but can be processed (Figure 1). Furthermore, if a job is completed before the end of a forbidden zone, it will be released at the start of the next interval. Thus, the latest time a job may commence in I_j is at $t = 2j-1$.

The objective is to sequence the jobs so that the number of intervals used is minimized. (If the last forbidden zone contains a job, then this is equivalent to minimizing the maximum completion time or makespan.)

Abdekhodae and Ernst (2004a) showed, for example, that the decision version of the (off-line) problem when all intervals are of equal length ($|I_i| = I, \forall i$) and all forbidden zones are equal ($|F_i| = F, \forall i$), is NP-complete in the weak

*Correspondence: A Wirth, Department of Mechanical and Manufacturing Engineering, The University of Melbourne, Victoria 3010, Australia. E-mail: wirtha@unimelb.edu.au

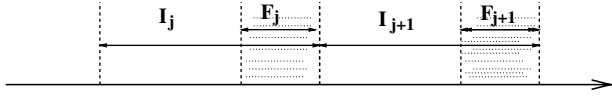


Figure 1 Intervals and forbidden zones.

sense. Abdekhodae and Ernst's (2004b) empirical results suggest that list scheduling and other simple heuristics perform best when F/I is close to 0 or 1, and that the most difficult cases occurred when F/I is close to 0.5.

There is a close link between our problem and that of *bin packing*. This becomes particularly evident once we consider integer programming formulations of the two problems.

We first recall the bin packing problem. Suppose we have n items with sizes p_1, \dots, p_n . We assume that $p_i \leq 1 \forall i$. Let $x_{ij} = 1$ if item i is allocated to bin j and $x_{ij} = 0$ otherwise. Also let $z_j = 1$ if bin j is used and $z_j = 0$ otherwise. Then our problem is

$$\min \sum_{j=1}^n z_j \quad (1)$$

subject to

$$\sum_{j=1}^n x_{ij} = 1, \quad \forall i \quad (2)$$

$$x_{ij} \leq z_j, \quad \forall i, j \quad (3)$$

$$\sum_{i=1}^n p_i x_{ij} \leq 1, \quad \forall j \quad (4)$$

$$x_{ij}, z_j \in \{0, 1\} \quad (5)$$

The objective function (1) that we wish to minimize is the number of bins used. Constraint (2) ensures that each item is allocated to a bin. Constraint (3) indicates that an item can be assigned to a bin only if that bin is used. Constraint (4) specifies the size constraint, namely that the sum of the sizes of items allocated to a unit-sized bin cannot exceed one.

We now modify the above model to handle the forbidden zones problem. Let $x_{ij} = 1$ if job i starts and ends in $I_j \setminus F_j$ and $x_{ij} = 0$ otherwise. Also let $z_j = 1$ if I_j is used and $z_j = 0$ otherwise. We introduce new variables to handle jobs finishing in (the closure of) forbidden zones. Let $y_{ij} = 1$ if job i starts in I_j and ends in \bar{F}_j and $y_{ij} = 0$ otherwise. Then the forbidden zones problem can be formulated as follows:

$$\min \sum_{j=1}^n z_j \quad (6)$$

subject to

$$\sum_{j=1}^n (x_{ij} + y_{ij}) = 1, \quad \forall i \quad (7)$$

$$x_{ij} + y_{ij} \leq z_j, \quad \forall i, j \quad (8)$$

$$\sum_{i=1}^n p_i x_{ij} \leq 1, \quad \forall j \quad (9)$$

$$\sum_{i=1}^n y_{ij} \leq 1, \quad \forall j \quad (10)$$

$$x_{ij}, y_{ij}, z_j \in \{0, 1\} \quad (11)$$

Constraint (10) ensures that at most one job can be processed in a forbidden zone. We now recall some definitions related to *on-line scheduling*. We shall consider the *one-by-one* (Sgall, 1998) or *on-line list* (Pruhs et al, 2004) version of on-line job scheduling. That is, we assume that all jobs are available at time 0 but they are presented to the scheduler one-by-one, in a list. The scheduler obtains information about the next job on the list and has to irrevocably schedule the job, that is assign the job to a time slot, based on this information, prior to being shown the next job on the list.

We also recall the definition of a c -competitive on-line algorithm. An on-line minimization algorithm A is said to be c -competitive if $A(I)/OPT(I) \leq c$ for all job instances I , where $A(I)$ and $OPT(I)$ are the objective function values for algorithm A and the optimal solution, respectively.

The corresponding *asymptotic performance ratio* $R_A^\infty(\alpha)$ is defined as follows. Suppose that $p_{max} = \alpha$, $0 < \alpha \leq 1$ then $R_A^\infty(\alpha) = \inf\{r \geq 1: \text{for some } N > 0, A(I)/OPT(I) \leq r, \text{ for all } I \text{ with } OPT(I) > N\}$.

We shall, in particular, consider the case when $\alpha = 1/m$ for some positive integer m .

A competitive ratio lower bound

There are many results about lower bounds and the performance of various on-line algorithms for bin packing (eg Coffman et al, 1997). In general terms, even small improvements in these results are often achieved at the cost of substantial increases in the complexity of the proofs. The current best lower bound on the competitive ratio is 1.54014 (van Vliet, 1992). The competitive ratios for the first fit and next fit bin packing algorithms are 1.7 (Garey et al, 1972) and 2 (Johnson, 1974), respectively. We shall present one lower bound result for the forbidden zones problem after proving the following proposition that points towards the relation between the bin packing and forbidden zones problems.

Rearrange the jobs in non-increasing order of their processing times and denote this new list by $J'_1 \dots J'_n$. Let p'_i be the processing time of J'_i . Thus $p'_1 \geq p'_2 \geq \dots \geq p'_n$. We have the following relationship between the optimal solutions of the bin packing and the forbidden zones problems.

Proposition 1 *Suppose the optimal solution to the bin packing problem for the $n-i$ smallest jobs is i bins. Then an optimal schedule for the forbidden zones problem is i intervals.*

Proof Firstly we show that i intervals suffice for the forbidden zones problem. Apply the optimal bin packing solution to placing jobs $J'_{i+1} \dots J'_n$ in i allowed zones. Now add the remaining i longest jobs $J'_1 \dots J'_i$ to the i forbidden zones. On the other hand, if the optimal solution to the forbidden zones problem requires fewer than i intervals then by swapping any jobs that finish in the forbidden zones with shorter ones wholly in the allowed zones we obtain a solution to the bin packing problem for the $n-i$ shortest jobs requiring fewer than i intervals, a contradiction. \square

Some lower bound proofs for on-line bin packing rely on using jobs with excruciatingly small processing times. Our next proposition uses an argument adapted from a proof of the result, that no on-line algorithm for bin packing is better than $\frac{4}{3}$ -competitive (Liang, 1980). It does not require jobs of duration less than $\frac{1}{3}$.

Proposition 2 *No on-line algorithm for the forbidden zones problem is better than $\frac{9}{7}$ -competitive.*

Proof Assume our algorithm A is c -competitive. Consider a sequence of n jobs each of length $\frac{1}{2} - \varepsilon$. Provided that $\varepsilon < \frac{1}{6}$, and that n is divisible by 3, A requires at most $a \leq cn/3$ intervals, since the optimal solution requires $n/3$ intervals. Suppose we are then presented with $2n$ jobs of length $\frac{1}{2} + \varepsilon$. Suppose that A uses an additional b intervals for these jobs. Each of these b intervals can accommodate at most two of these new jobs, whereas each of the first a intervals could accommodate up to three of the original n jobs. Thus, $3a + 2b \geq 3n$. Also $a + b \leq cn$ since the optimal solution requires n intervals. Hence $(3-2c)n \leq a \leq cn/3$. So $c \geq \frac{9}{7}$. \square

There is a strong connection between bin packing and forbidden zones algorithms. Suppose B is a bin packing algorithm that never allows, during its execution, a gap of size larger than $1/m$ in a bin, other than possibly the last open bin. We can generate a forbidden zones version B' of B in the following way. If $p_i < 1/m$ ignore the forbidden zones and place the job in the appropriate allowed zone according to B . Otherwise, if $p_i \geq 1/m$ place the job in the first available allowed zone. If this is not possible place it in the first available forbidden zone. If that is not possible, generate a new interval and place the job in the allowed zone of the new interval.

Lemma 3 *If a bin packing algorithm B never allows a gap of $1/m$ in a bin then $R_B^\infty(1/m) + (1/m) \leq R_{B'}^\infty(1/m) \leq (m+1)/(m-1)$.*

Proof Consider the worst case instance for B . No jobs can be larger than $1/m$. Replace any job of size $1/m$, with a job of size $(1/m) - \varepsilon$. Thus B' would not place any jobs in the forbidden zones.

Suppose for this instance, an optimal solution for the bin packing problem uses n^* bins and so B uses $n \geq (R_B^\infty - \varepsilon)n^*$ bins. We add to this instance n^* jobs of size $(1/m) - \varepsilon$. Since B' cannot schedule these jobs in the forbidden zones, they must be placed in the allowed zones resulting in an increase of (n^*/m) intervals. However, based on the optimal solution for the bin packing problem these jobs can be put into the empty forbidden zones of the n^* intervals. That is, there is no increase in the number of intervals. Thus we have

$$B'(I) = n + \frac{1}{m}n^*$$

$$OPT(I) \leq n^*$$

Therefore,

$$\frac{B'(I)}{OPT(I)} \geq \frac{n + (1/m)n^*}{n^*}$$

$$\geq \frac{(R_B^\infty(1/m) + (1/m))n^* - \varepsilon n^*}{n^*}$$

$$\geq R_B^\infty\left(\frac{1}{m}\right) + \frac{1}{m} - \varepsilon$$

On the other hand, at worst, B' will leave all forbidden zones empty. The gap in any allowed zone, except possibly the last one, must be less than $1/m$. So all allowed zones, except possibly the last, in the B' solution are at least $1 - (1/m)$ full. On the other hand, in the optimal solution each forbidden zone is at most $1/m$ full. The result now follows. \square

Bounds for four on-line algorithms

In this section, we propose four on-line algorithms, including adaptations of three well-known bin packing algorithms, namely first fit, next fit and harmonic.

List scheduling (LS)

List scheduling (LS) sequences the next job on the list in the first feasible position. Thus, provided the previous job has completed by the end of the current allowed zone, we schedule the next job immediately after it. Otherwise, we start the next job at the beginning of the next available zone. (If the previous job finishes at the end point of an allowed zone we start the next job at the beginning of the abutting forbidden zone.)

Proposition 4 $R_{LS}^\infty(1/m) = 1 + (1/m)$

Proof Consider the concatenation of n sequences, each of $m + 1$ jobs consisting of m jobs each of length $(1/m)$ followed by a job of length ε . LS requires n intervals, whereas the optimal solution requires only $nm/(m + 1) + 1$ intervals, for sufficiently small ε .

On the other hand, for LS , each allowed zone except possibly the last one, is full. So, LS is at worst $(1 + (1/m))$ -competitive. \square

Half Next Fit algorithm (HNF)

This algorithm which is adapted from the next fit (NF) algorithm for the bin packing problem operates as follows.

HNF always considers only one *active* allowed zone. Whenever a job arrives, if its size is smaller than the gap in the active allowed zone, the job is placed into that zone. Otherwise, if the size of the job is greater than or equal to $\frac{1}{2}$ we place the job into the first available forbidden zone. If there is no available forbidden zone, close the current interval, open a new one and place the job in the allowed zone of that interval. If the size of the job is less than $\frac{1}{2}$ we close the current active allowed zone and open a new one and place the job in that zone.

Based on Lemma 3, we know that the competitive ratio is at least 2.5 since $R_{NF}^\infty(\frac{1}{2}) = 2$ (Coffman *et al*, 1997). However, the following proposition shows that, in fact, the asymptotic performance ratio of HNF is 3.

Proposition 5 $R_{NF}^\infty = 3$.

Proof We first show that $R_{HNF}^\infty \geq 3$. Suppose a job of size $\frac{1}{2} - \varepsilon$ followed by a job of size 3ε arrives. Repeat this sequence n times. Note that since all jobs in this instance are smaller than $\frac{1}{2}$, HNF cannot place a job in a forbidden zone. HNF would put a job of size $\frac{1}{2} - \varepsilon$ and a job of size 3ε in an allowed zone. The gap in the allowed zone is smaller than $\frac{1}{2} - \varepsilon$. Therefore the current active allowed zone is then closed and the next job is placed in the new active allowed zone. HNF would need n allowed zones for this instance. Whereas in an optimal solution, 3 jobs of size $\frac{1}{2} - \varepsilon$ can be scheduled in an interval (that is, an allowed zone and a forbidden zone). For n jobs of size $\frac{1}{2} - \varepsilon$, $\frac{n}{3}$ intervals are needed. Also for n jobs of size 3ε , and sufficiently small ε the optimal solution would place them in one allowed zone. Therefore we have

$$\begin{aligned} HNF(I) &= n \\ OPT(I) &\leq \frac{n}{3} + 1 \\ \frac{HNF(I)}{OPT(I)} &\geq \frac{n}{(n/3) + 1} \rightarrow 3 \text{ if } n \rightarrow \infty \end{aligned}$$

Now we show that $R_{HNF}^\infty \leq 3$. Let $s(a_i)$ be the sum of the processing times of the jobs in the i th allowed zone and $s(f_i)$ be the processing time of the job in the i th forbidden zone. It follows from the definition of HNF that $s(a_j) > 0 \Rightarrow s(a_i) + s(f_i) \geq \frac{1}{2} \forall i < j$. So, if $HNF(I) = n$ then we have at least $n - 1$ groups of jobs each of size at least $\frac{1}{2}$. Clearly, $OPT(I) \geq (n - 1)/3$. \square

Proposition 6 $(m^2 + m - 1)/(m(m - 1)) \leq R_{HNF}^\infty(1/m) \leq (m^2 + m)/(m(m - 1))$ and $R_{HNF}^\infty(1/m) \sim (m + 1)/(m - 1)$ as $m \rightarrow \infty$.

Proof The proof follows from Lemma 3 and the fact that $R_{NF}^\infty(1/m) = m/(m - 1)$ (Coffman *et al*, 1997). \square

Proposition 7 $R_{HNF}^\infty(\frac{1}{2}) = 3$.

Proof By Proposition 6 we have $R_{HNF}^\infty(\frac{1}{2}) \leq (2^2 + 2)/(2(2 - 1)) = 3$. In the proof of Proposition 5 no job is longer than $\frac{1}{2}$, so $R_{HNF}^\infty(\frac{1}{2}) \geq 3$. \square

Suppose now that we modify HNF so that our new algorithm, bNF , say, always considers only one *active* allowed zone, and whenever a job arrives, and if its size is smaller than the gap in the active allowed zone, the job is placed into that zone, as for HNF . Otherwise, if the size of the job is greater than or equal to b (where $b < 1$) we place the job into the first available forbidden zone. If there is no available forbidden zone, close the current interval, open a new one and place the job in the allowed zone of that interval. If the size of the job is less than b we close the current active allowed zone and open a new one and place the job in that zone.

Proposition 8 $2 + \lfloor b + \frac{1}{2} \rfloor \leq R_{bNF}^\infty \leq 4$.

Proof We first show that $2 + \lfloor b + \frac{1}{2} \rfloor \leq R_{bNF}^\infty$. Assume first that $b > \frac{1}{2}$. Consider a sequence of $5n$ jobs consisting of n quintuplets of the following size jobs: $b - \varepsilon$ followed by 4ε , $1 - b - \varepsilon$, 4ε and $b - \varepsilon$. Note that since all jobs in this instance are smaller than b , bNF cannot place a job in a forbidden zone. Clearly $bNF(I) = 3n$ and $OPT(I) = n + 1$. Now suppose that $b < \frac{1}{2}$. Consider a sequence of $3n$ jobs consisting of n triplets of jobs of size $b - \varepsilon$, $1 - 2b - \varepsilon$ and 4ε . Clearly, $bNF(I) = n$, since no forbidden zones are used. Now $OPT(I) = (n/2) + 1$, so $2 + \lfloor b + \frac{1}{2} \rfloor \leq R_{bNF}^\infty$.

Now we show that $R_{bNF}^\infty \leq 4$. Suppose that $bNF(I) = n$, then $s(a_n) > 0$ and hence $s(a_i) + s(a_{i+1}) > 1$ for $i = 1 \dots n - 1$. So $\sum_{i=1}^n s(a_i) > (n - 1)/2$ and $OPT(I) \geq (n - 1)/4$ proving the result. \square

Half First Fit algorithm (HFF)

This heuristic, half first fit algorithm (HFF), makes use of the first fit (FF) algorithm for the bin packing problem. Its procedure is as follows.

Whenever a job arrives, if the size of the job is smaller than or equal to a gap in one of the existing allowed zones then we schedule the job into the first such gap. Otherwise, if the size of the job is greater than or equal to $\frac{1}{2}$ we place the job into the first available forbidden zone. If there is no available forbidden zone we open a new interval and place the job in the allowed zone of that interval. If the size of the job is less than $\frac{1}{2}$ we open a new allowed zone and place the job in the new zone. The motivation for this is to try to place the longest jobs in the forbidden zones.

Proposition 9 $\frac{5}{2} \leq R_{HFF}^\infty \leq \frac{27}{10}$.

Proof Firstly we show that $\frac{5}{2} \leq R_{HFF}^\infty$. By a result of Johnson *et al* (1974), $R_{FF}^\infty(.4) = \frac{3}{2}$. It follows that, for any $\delta > 0$, there is an instance I for the bin packing problem, with all items of size $< \frac{1}{2}$, such that $FF(I)/OPT(I) \geq \frac{3}{2} - \delta$. Suppose that $OPT(I)$ uses n bins, thus $FF(I)$ would require at least $(\frac{3}{2} - \delta)n$ bins. If we regard these bins as allowed zones in the forbidden zones problem, the result still holds since all forbidden zones are empty. Now we create a new instance, I' as follows. A job of size ϵ is presented followed by a job of size $1 - \epsilon$. This arrangement is repeated n times resulting in $2n$ jobs to be scheduled. Then a set of jobs in I is presented. HFF would require n intervals for the first $2n$ jobs and $(\frac{3}{2} - \delta)n$ intervals for the jobs from I . On the other hand, the optimal solution would utilize only $n + 1$ intervals since the n jobs of size $1 - \epsilon$ can be put into forbidden zones and one interval for jobs of size ϵ suffices. Thus $\frac{5}{2} \leq R_{HFF}^\infty$.

We now show that $R_{HFF}^\infty \leq 2.7$ by slightly adapting the proof by Johnson *et al* (1974) showing $R_{FF}^\infty = 1.7$. Consider the following weighting function:

$$w(x) = \begin{cases} \frac{6}{5}x & \text{if } 0 \leq x \leq \frac{1}{6} \\ \frac{9}{5}x - \frac{1}{10} & \text{if } \frac{1}{6} \leq x \leq \frac{1}{3} \\ \frac{6}{5}x + \frac{1}{10} & \text{if } \frac{1}{3} \leq x \leq \frac{1}{2} \\ 1 & \text{if } x > \frac{1}{2} \end{cases}$$

Let $L = (a_1, \dots, a_n)$ be an input instance and $W(L) = \sum_{i=1}^n w(a_i)$ be its corresponding weighting function. Also define $W(I) = \sum_{j=1}^t w(a_{ij})$, where $a_{ij}, j = 1, \dots, t$ are jobs scheduled to an interval I . We make use of the following two results by Johnson *et al* (1974), for the bin packing problem:

1. For any bin B , $W(B) \leq \frac{17}{10}$ and therefore $W(L) \leq \frac{17}{10}OPT(L)$.
2. If FF uses k bins $B_1 \dots B_k$ then $\sum_{j=1}^k W(B_j) \geq k - 2$, and thus $FF(L) \leq W(L) + 2$.

We now apply the above two results to the forbidden zones problem. For any interval I , consider the jobs entirely within the allowed zone. If we regard the allowed zone as a bin B we have $W(B) \leq \frac{17}{10}$. Any job in the forbidden zone has a weight no greater than 1. Thus, for any interval I , we have $W(I) \leq \frac{27}{10}$. So, $W(L) \leq \frac{27}{10}OPT(L)$.

Now consider a schedule S obtained by employing HFF . We create a new schedule S' by removing all jobs that are scheduled to forbidden zones in S . Note that if we consider only allowed zones, S' which uses s , say, allowed zones (hence intervals) can be regarded a set of s bins, B , packed by FF . Since $W(B) \geq s - 2$ thus $W(S') \geq s - 2$. So, $HFF(S) = s \leq W(S') + 2 \leq W(S) + 2 \leq \frac{27}{10}OPT(S) + 2$, proving the result. \square

Half Harmonic M algorithm (HH_M)

Lee and Lee (1985) proposed *Harmonic M*, an improvement on FF , which keeps M bins open at any one time. They used the following sequence, $k_1 = 1$ and $k_{i+1} = k_i(k_i + 1)$ for $i \geq 1$, proposed earlier by Liang (1980). They showed that $R_{H_M}^\infty$ is monotonically decreasing with M and that $\lim_{M \rightarrow \infty} R_{H_M}^\infty = \sum_{i=1}^\infty \frac{1}{k_i} = 1.6910\dots$

The algorithm divides the unit interval $(0, 1]$ into M subintervals $I_j = (1/(j+1), 1/j]$ ($1 \leq j < M$) and $I_M = (0, 1/M]$. The corresponding version for the forbidden zones problem, HH_M , always considers M allowed zones at a time. A job is called an I_j -job if its size belongs to interval I_j . An allowed zone is called an I_j allowed zone if it is designated to process only I_j jobs. At any one time, there are M allowed zones available, one for each I_j ($1 \leq j \leq M$). Whenever a job arrives, if it is an I_j -job and its size is smaller than the gap in the active I_j allowed zone, the job is placed in that zone. Otherwise, if the size of the job is greater than $\frac{1}{2}$ we place the job in the first available forbidden zone. If there is no available forbidden zone we open a new interval make its allowed zone of type I_1 and place the job in the allowed zone of that interval. If the size of the job is less than or equal to $\frac{1}{2}$ we open a new I_j allowed zone, for the appropriate $2 \leq j \leq M$ and place the job in the new allowed zone.

Below, we prove, by a series of lemmas, using arguments similar to those by Lee and Lee (1985), that $\lim_{M \rightarrow \infty} R_{HH_M}^\infty = 1.9231177\dots$

To facilitate our analysis, we shall refer to a job with size larger (not larger) than $\frac{1}{2}$ as a *large (small)* job.

Lemma 10 Consider any instance J of jobs presented in a particular order. Let J' be the same set of jobs in the same order save that the large jobs are now presented first. Then $HH_M(J') \geq HH_M(J)$.

Proof The I_2, \dots, I_M allowed zones for both J and J' are identical. The number of I_1 intervals for J is less than or equal to that of J' since in J some forbidden zones in $I_2 \dots I_M$ may be non-empty, whereas for J' all I_2, \dots, I_M forbidden zones are empty. \square

Lemma 11 For each forbidden zones problem, with an optimal solution consisting of n intervals, there exists an optimal solution containing the n largest jobs in the forbidden zones.

Proof Consider an optimal solution and swap the n jobs in the forbidden zones (considering an empty forbidden zone as a job of size 0) with the n largest jobs. The new solution is at least as good as the optimal one. \square

Based on Lemma 10 which states that the worst case behaviour of HH_M occurs when all the large jobs are presented prior to the arrival of small jobs, we adapt the result of Lee and Lee (1985) slightly and follow their notation. Let n_k be the number of I_k jobs in an input instance L and S_M be the total size of I_M jobs. Also, let m_k denote the number of I_k zones used (which means both allowed and forbidden zones for $k=1$ and allowed zones for $1 < k \leq M$). Thus, we have

$$m_1 = \left\lceil \frac{n_1}{2} \right\rceil$$

and for $2 \leq k < M$

$$m_k = \left\lceil \frac{n_k}{k} \right\rceil$$

and it is easy to see that

$$m_M < \left\lceil \frac{MS_M}{(M-1)} \right\rceil + 1$$

since an I_M allowed zone (except possibly the last one) can, at worst, have a gap of size $(1/M) - \varepsilon$.

It is clear that we can obtain a bound on the number of intervals used by HH_M . Thus we have

$$\begin{aligned} HH_M(L) &\leq \left\lceil \frac{n_1}{2} \right\rceil + \left\lceil \frac{n_2}{2} \right\rceil + \cdots + \left\lceil \frac{n_M - 1}{M - 1} \right\rceil + m_M \\ &\leq \frac{n_1}{2} + \frac{n_2}{2} + \cdots + \frac{n_M - 1}{M - 1} + \frac{MS_M}{M - 1} + M + 1 \end{aligned}$$

Since we consider the asymptotic performance of the algorithm, the last term, $M+1$, can be ignored. As for $Harmonic_M$ for bin packing, each I_k -job contributes to $1/k$ interval for $1 < k < M$, since in an I_k -interval, there can be at most k I_k -jobs scheduled in its allowed zone and its associated forbidden zone is left empty. This is because, as proposed in Lemma 10, we only consider the case where large jobs (if there is any) are presented before all the small jobs resulting in no large jobs scheduled in the forbidden zones of the I_k -intervals (for $k > 1$). This is also true for I_M , that is the I_M -jobs contribute at most to $MS_M/(M-1)$ intervals. However, our I_1 -jobs are slightly different from that of $Harmonic_M$ for bin packing. Since all the large jobs arrive first, they can be placed in the allowed and forbidden zones of their I_1 -intervals. Hence an I_1 -job contributes to $\frac{1}{2}$ of such intervals. Based on these differences, we modify slightly the *fraction function* g for $harmonic_M$ algorithm, defined

earlier by Lee and Lee (1985). Thus

$$g(x) = \begin{cases} \frac{1}{2} & \text{if } x \in I_1 \\ \frac{1}{k} & \text{if } x \in I_k \text{ and } 1 < k < M \\ \frac{Mx}{M-1} & \text{if } x \in I_M \end{cases}$$

Let $L = (a_1, a_2, \dots, a_n)$ be a sequence of jobs that needs to be scheduled. Suppose that an optimal solution for the forbidden zone problem uses L^* intervals. Let the list of jobs in the i th interval be $(y_{i1}, y_{i2}, \dots, y_{it_i})$ with $y_{i1} \geq y_{i2} \geq \dots \geq y_{it_i} > 0$. Since, in the optimal solution, the sum of the job sizes in an interval (except for possibly the last one) must be larger than 1 and less than or equal to 2 and only one job is allowed to be processed (not initiated) in a forbidden zone, we have $1 < \sum_{j=1}^{t_i} y_{ij} \leq 2$ and $\sum_{j=2}^{t_i} y_{ij} \leq 1$. Also let $G_{i,M}$ denote $\sum_{j=1}^{t_i} g(y_{ij})$.

Let $S = \{(y_1, y_2, \dots, y_t) \mid y_i > 0, 1 \leq i \leq t, 1 < \sum_{j=1}^t y_j \leq 2 \text{ and } \sum_{j=2}^t y_j \leq 1\}$ and let $\bar{G}_M = \sup_S \sum_{i=1}^t g(y_i)$, where the supremum is taken over all sets $\{y_i\}$ satisfying the above conditions. We have

$$\begin{aligned} HH_M(L) &< \sum_{i=1}^n g(a_i) + M + 1 \\ &= \sum_{i=1}^{L^*} G_{i,M} + M + 1 \\ &\leq (\bar{G}_M)L^* + M + 1 \end{aligned}$$

Since we seek the competitiveness of HH_M , it is apparent that

$$R_{HH_M}^\infty \leq \bar{G}_M$$

Thus we now investigate an upper bound for \bar{G}_M .

As in Lee and Lee (1985) for bin packing, we can also state a relationship between the size of a job and its fraction function. That is, we establish a connection between a job processing time and the number of intervals used. For $x \in I_k$ where $1 < k < M$, $g(x) = 1/k$ and $1/(k+1) < x \leq 1/k$. Thus $g(x)/x < (k+1)/k$. That is,

$$\frac{g(x)}{x} < \frac{k+1}{k}, \quad \text{if } x \leq \frac{1}{k} \text{ for } 1 < k < M$$

and obviously we have,

$$g(x) < x, \quad \text{if } x \leq 1$$

and

$$\frac{g(x)}{x} = \frac{M}{(M-1)}, \quad \text{if } x \in I_M$$

We define two sequences. Our first sequence, L , thus is

$$\begin{aligned} l_1 &= 1 \\ l_2 &= 2 \\ l_3 &= 2 \\ l_4 &= 3 \\ l_{j+1} &= l_j(l_j + 1), \quad \text{for } j > 3 \end{aligned}$$

For the other one, we make use of the well-known bin packing sequence mentioned earlier. Namely

$$\begin{aligned} k_1 &= 1 \\ k_{j+1} &= k_j(k_j + 1) \quad \text{for } j > 1 \end{aligned}$$

Note that these two sequences satisfy the following inequalities

$$\sum_{i=2}^S \left(\frac{1}{l_i + 1} + \varepsilon \right) < 1$$

since, for small enough S and $\varepsilon > 0$

$$\begin{aligned} \sum_{i=2}^S \left(\frac{1}{(l_i + 1)} + \varepsilon \right) &= \left(\frac{1}{3} + \varepsilon \right) + \left(\frac{1}{3} + \varepsilon \right) \\ &\quad + \sum_{i=4}^S \left(\frac{1}{l_i + 1} + \varepsilon \right) \\ &= \left(\frac{2}{3} + 2\varepsilon \right) + \sum_{i=4}^S \left(\frac{1}{l_i} - \frac{1}{l_i + 1} + \varepsilon \right) \\ &= \left(\frac{2}{3} + 2\varepsilon \right) + \frac{1}{3} = \frac{1}{l_{S+1}} + (S - 3)\varepsilon \\ &< 1 \end{aligned}$$

for any S and small enough $\varepsilon > 0$ and, for the latter sequence.

$$\sum_{i=1}^S \left(\frac{1}{k_i + 1} + \varepsilon \right) < 1$$

for any S and small enough $\varepsilon > 0$. The proof of this result is similar to that of the first sequence.

$$\begin{aligned} \sum_{i=1}^S \left(\frac{1}{k_i + 1} + \varepsilon \right) &= \sum_{i=1}^S \left(\frac{1}{k_i} - \frac{1}{k_{i+1}} + \varepsilon \right) \\ &= 1 - \frac{1}{k_{S+1}} + S\varepsilon \\ &< 1 \end{aligned}$$

Since our objective is to minimize the number of intervals used and only jobs of type I_1 can be processed in forbidden zones, we need to consider two cases: (i) when there are no large jobs in the input instance, (ii) otherwise. Note that here $i > 2, j > 2, l_j < M \leq l_{j+1}$ and $k_i < M \leq k_{i+1}$.

Case I: $m_1 = 0$. We start by considering the first case, that is, where $n_1 = m_1 = 0$. Here we have $y_1 \notin I_1$. Thus, $y_i \leq \frac{1}{2} \forall i =$

$1 \dots t$. Then we have $g(y_i) < \frac{3}{2}y_i$. Therefore, $G_M \leq \sum_{i=1}^t g \times (y_i) < \frac{3}{2} \sum_{i=1}^t y_i \leq \frac{3}{2} \cdot \frac{3}{2}$ as $\sum_{i=1}^t y_i \leq \frac{3}{2}$ (the largest job that can be scheduled to a forbidden zone is of size at most $\frac{1}{2}$). Thus, in the case where there is no large job in the input instance, we have

$$G_M < \frac{9}{4} = 2.25$$

However, the above bound can be improved since $g(y_i) < \frac{3}{2}y_i$ can be tightened. That is, if $y_i \leq \frac{1}{3}$ then $g(y_i) < \frac{4}{3}y_i$ and so on. Since $\sum_{i=2}^t y_i \leq 1$, there can be at most two jobs of type I_2 and we wish to obtain as large $g(y_i)$ as possible for a particular y_i . So we may assume that $y_2, y_3 \in I_2$. Thus we have $\sum_{i=4}^t y_i \leq \frac{1}{3} - 2\varepsilon < \frac{1}{3}$ for some $\varepsilon > 0$. For $g(y_4)$ to be as large as possible, we assume $y_4 \in I_3$. This makes $\sum_{i=5}^t y_i \leq \frac{1}{12} - 3\varepsilon < \frac{1}{12}$. That is, $g(y_i) < \frac{13}{12}y_i$ for $i \geq 5$. If we continue this process, eventually we would have two mutually exclusive cases.

Subcase 1.1: Suppose that $y_1 \in I_2, y_2 \in I_2, y_3 \in I_3, y_4 \in I_4, \dots, y_{u-1} \in I_{u-1}$ and $y_u \notin I_u$, where $u < j$. Since $\sum_{p=2}^t y_p \leq 1, \sum_{p=2}^{u-1} y_p > \sum_{p=2}^{u-1} 1/(l_p + 1)$ and $\sum_{s=2}^u 1/l_p + 1 = 1 - (1/(l_s + 1))$, thus we have $\sum_{s=u}^t y_s < 1 - \sum_{p=2}^{u-1} 1/(l_p + 1) = 1/l_u$. Additionally, as $y_u \notin I_u$, so $y_s \leq 1/(l_u + 1)$ for all $s \geq u$. Thus we obtain a bound on G_M for this case as follows:

$$\begin{aligned} G_M &= \frac{1}{2} + \sum_{i=2}^{u-1} \frac{1}{l_i} + \sum_{s=u}^t \frac{(l_u + 2)y_s}{(l_u + 1)} \\ &< \frac{1}{2} + \sum_{i=2}^{u-1} \frac{1}{l_i} + \frac{(l_u + 2)}{(l_u + 1)} \cdot \frac{1}{l_u} \\ &= \frac{1}{2} + \sum_{i=2}^{u-1} \frac{1}{l_i} + \frac{1}{l_u} + \frac{1}{l_u + 1} \\ &= \frac{1}{2} + \sum_{s=2}^{u+1} \frac{1}{l_s} \\ &\leq \frac{1}{2} + \sum_{s=2}^j \frac{1}{l_s} \end{aligned}$$

Subcase 1.2: In this subcase, we have $y_1 \in I_2, y_2 \in I_2, y_3 \in I_3, y_4 \in I_4, \dots, y_j \in I_j$. That is, a job of type I_p , where $2 \leq p \leq j$ is put in an I_p allowed zone. Thus all smaller jobs are placed in an I_M allowed zone. Because $\sum_{p=2}^t y_p \leq 1$ and $\sum_{p=2}^j y_p > \sum_{p=2}^j 1/(l_p + 1) = 1 - (1/(l_{j+1}))$ thus $\sum_{s=j+1}^t y_s < 1/(l_{j+1})$. From the fact that $M \leq l_{j+1}$, it follows that $y_s \in I_M, \forall s \geq j + 1$. Thus we have the following bound:

$$\begin{aligned} G_M &= g(y_1) + \sum_{p=2}^j g(y_p) + \frac{M \sum_{s=j+1}^t y_s}{M - 1} \\ &= \frac{1}{2} + \sum_{p=2}^j \frac{1}{l_p} + \frac{M \sum_{s=j+1}^t y_s}{M - 1} \end{aligned}$$

In both subcases, it can be observed that $G_M < \frac{1}{2} + \sum_{p=2}^j (1/l_p) + M/((M - 1)l_{j+1})$.

Case 2: $m_1 > 0$. This case is slightly more complicated. We consider three subcases where the last two deploy different characteristics of the size of y_i according to the sequence K .

Subcase 2.1: In this subcase, we have only one large job in an interval of the optimal solution. Based on lemma 11, this large job must be put into the forbidden zone. Hence we consider $y_1 \in I_1$ and $y_2 \notin I_1$. Thus, $y_i \leq \frac{1}{2} \forall i = 2 \dots t$. So $g(y_1) = \frac{1}{2}$ and $g(y_i) < \frac{3}{2} y_i \forall i = 2 \dots t$. Therefore, we have $G_M = \sum_{j=1}^t g(y_j) < g(y_1) + \frac{3}{2} \sum_{i=2}^t y_i \leq \frac{1}{2} + \frac{3}{2}$, since $g(y_1) = \frac{1}{2}$ and $\sum_{i=2}^t y_i \leq 1$. So we have, in this subcase,

$$G_M < \frac{1}{2} + \frac{3}{2} = 2$$

However, this bound is not tight. This bound can be tightened by using a similar argument to that in subcases 1.1 and 1.2 mentioned above. We modify slightly the input instances in both subcases by replacing $y_1 \in I_2$ with $y_1 \in I_1$ (since there must be one large job in the instance). Nevertheless, the results of both subcases still hold for this subcase as well.

Subcase 2.2: Here and in the next subcase, instead of using the sequence L we make use of the sequence K and consider the situation for which, in the optimal solution, there are at least two large jobs. That is, $y_1 \in I_1$ and $y_2 \in I_1, y_3 \in I_{k_2}, y_4 \in I_{k_3}, \dots, y_v \in I_{k_{v-1}}$ and $y_{v+1} \notin I_{k_v}$ for all $v \leq i$. Then we have $\sum_{s=v+1}^i y_s < 1 - \sum_{p=2}^v 1/(k_{p-1} + 1) = 1 - \sum_{q=1}^{v-1} 1/(k_q + 1) = 1/k_v$, since $\sum_{p=2}^v y_p \leq 1$ and $\sum_{p=2}^{v-1} y_p > \sum_{p=2}^{v-1} 1/(k_{p-1} + 1)$. Also since $y_{v+1} \notin I_{k_v}$, thus $y_s \leq 1/(k_v + 1) \forall s \geq v + 1$. Thus we have,

$$\begin{aligned} G_M &= g(y_1) + \sum_{p=2}^i g(y_p) \\ &\leq g(y_1) + g(y_2) + \sum_{p=3}^v \frac{1}{k_{(p-1)}} + \sum_{s=v+1}^i \frac{y_s(k_v + 2)}{(k_v + 1)} \\ &< g(y_1) + g(y_2) + \sum_{q=2}^{v-1} \frac{1}{k_q} + \frac{(k_v + 2)}{(k_v + 1)} \cdot \frac{1}{k_v} \\ &= \frac{1}{2} + \frac{1}{2} + \sum_{q=2}^{v-1} \frac{1}{k_q} + \frac{1}{k_v} + \frac{1}{k_v(k_v + 1)} \\ &= 1 + \sum_{q=2}^{v+1} \frac{1}{k_q} \\ &\leq 1 + \sum_{q=2}^{i+1} \frac{1}{k_q} \end{aligned}$$

Subcase 2.3: We examine the case where $y_1 \in I_{k_1}$ and $y_2 \in I_{k_1}, y_3 \in I_{k_2}, y_4 \in I_{k_3}, \dots, y_{i+1} \in I_{k_i}$. That is, in an interval of the optimal solution, each job (y_p where $2 \leq p \leq i + 1$) is scheduled to an allowed zone. Therefore, all other smaller jobs (y_q , where $q > i + 1$) are classified to be of type I_M . That is, $\sum_{p=i+2}^t y_p < 1/(k_{i+1})$ since $\sum_{p=2}^i y_p \leq 1$ and $\sum_{s=2}^{i+1} y_s > \sum_{s=2}^{i+1} 1/(k_{s-1} + 1) = \sum_{q=1}^i 1/(k_q + 1)$. And as

$M \leq k_{i+1}$ so $y_s \in I_M$ for $s > i + 1$. Consequently, we have

$$\begin{aligned} G_M &= g(y_1) + \sum_{p=2}^i g(y_p) \\ &= g(y_1) + g(y_2) + \sum_{p=3}^{i+1} \frac{1}{k_{(p-1)}} + \frac{M(\sum_{s=i+2}^t y_s)}{M-1} \\ &= g(y_1) + g(y_2) + \sum_{q=2}^i \frac{1}{k_q} + \frac{M(\sum_{s=i+2}^t y_s)}{M-1} \end{aligned}$$

Note that the structure of the proofs for subcases 2.2 and 2.3 is similar to those of subcases 1.1 and 1.2 except for the different sequences used. It is not difficult to see that, for subcases 2.2 and 2.3, $G_M < 1 + \sum_{q=2}^i (1/k_q) + M/(M-1)k_{i+1}$. We note that this result is the same as that of the *Harmonic_M* algorithm for bin packing.

After considering all cases, we can conclude that the tightest upper bound for \bar{G}_M is

$$\max\left(\frac{1}{2} + \sum_{p=2}^i \frac{1}{l_p} + \frac{M}{(M-1)l_{j+1}}, 1 + \sum_{q=2}^i \frac{1}{k_q} + \frac{M}{(M-1)k_{i+1}}\right)$$

for a particular M . If $M \geq 3$, $M \leq l_{j+1}$ and $M \leq k_{i+1}$ so $j \geq i + 1, j \geq i + 1, j \geq 3$ and $i \geq 2$, we have

$$\begin{aligned} \frac{1}{2} + \sum_{p=2}^j \frac{1}{l_p} + \frac{M}{(M-1)l_{j+1}} &\geq \frac{1}{2} + \sum_{p=2}^{i+1} \frac{1}{l_p} + \frac{M}{(M-1)l_{i+2}} \\ &= \frac{1}{2} + \sum_{p=2}^{i+2} \frac{1}{l_p} + \frac{1}{(M-1)l_{i+2}} \\ &= \frac{1}{2} + \frac{1}{2} + \sum_{p=3}^{i+2} \frac{1}{l_p} + \frac{1}{(M-1)l_{i+2}} \\ &\geq 1 + \sum_{q=2}^{i+1} \frac{1}{k_q} + \frac{1}{(M-1)k_{i+1}} \\ &= 1 + \sum_{q=2}^i \frac{1}{k_q} + \frac{M}{(M-1)k_{i+1}} \end{aligned}$$

since $l_{i+1} \leq k_i, \forall i \geq 2$. Therefore, if $M \geq 3$, we have

$$\frac{1}{2} + \sum_{p=2}^j \frac{1}{l_p} + \frac{M}{(M-1)l_{j+1}} \geq 1 + \sum_{q=2}^i \frac{1}{k_q} + \frac{M}{(M-1)k_{i+1}}$$

The upper bounds for \bar{G}_M for different M are shown in Table 1. Let $U_\infty = \frac{3}{2} + \frac{1}{3} + \frac{1}{12} + \frac{1}{156} + \frac{1}{24492} + \dots = 1.923117753 \dots$

Thus we have the following lemma.

Lemma 12 For $M \geq 3$ and $l_j < M \leq l_{j+1}$, $R_{HH_M}^\infty \leq \bar{G}_M < \frac{1}{2} + \sum_{p=2}^j (1/l_p) + M((M-1)l_{j+1})$ and $\lim_{M \rightarrow \infty} R_{HH_M}^\infty \leq U_\infty = 1.923117753 \dots$

Now we obtain a lower bound on the competitive ratio for HH_M .

Table 1 Upper bounds on the competitive ratio HH_M for different values of M

M	Upper bound for \bar{G}_M
3	2.00
4	1.94444...
5	1.93750...
6	1.93333...
7	1.93055...
10	1.92592...
13	1.92361...
157	1.92312...
∞	1.9231177

Lemma 13 $\lim_{M \rightarrow \infty} R_{HH_M}^\infty \geq U_\infty = 1.923117753 \dots$

Proof In order to prove the claim, we make use of the sequence $\{l_j\}$. Suppose that $M = l_i + 1$ where $i \rightarrow \infty$ and n is a multiple of l_i (hence, divisible by 2). Consider the following input instance, in jobs which consist of n jobs of size a where $a \in I_1$ and n jobs of size $1/(l_j + 1) + \epsilon$ for each j , where $2 \leq j \leq i$. Note that the jobs are presented to the scheduler in non-increasing order, that is, n jobs of size a are presented first followed by n jobs of size $1/(l_2 + 1) + \epsilon$ and so on. For this instance, HH_M would use $\lceil n/l_j \rceil$ allowed zones for n jobs of size $1/(l_j + 1) + \epsilon$ where $2 \leq j \leq i$, since, for these jobs, HH_M would leave all their corresponding forbidden zones empty. Thus HH_M would need $\lceil n/l_j \rceil$ intervals for n jobs of size $1/(l_j + 1) + \epsilon$. Additionally, HH_M needs $n/2$ allowed zones and $n/2$ forbidden zones for n jobs of size a . Thus we have

$$\begin{aligned}
 HH_M(L) &= \frac{n}{2} + \sum_{j=2}^i \left\lceil \frac{n}{l_j} \right\rceil \\
 &= \frac{n}{2} + \sum_{j=2}^i \frac{n}{l_j}
 \end{aligned}$$

since, by our assumption, $\lceil n/l_j \rceil = n/l_j$.

Whereas the optimal solution uses only n intervals. Thus we have

$$\begin{aligned}
 \frac{HH_M(L)}{OPT(L)} &= \frac{1}{n} \left[\frac{n}{2} + n \left(\frac{1}{2} + \frac{1}{2} + \frac{1}{3} + \frac{1}{12} + \frac{1}{156} + \dots \right) \right] \\
 &= 1.9231177
 \end{aligned}$$

Therefore, $R_{HH_M}^\infty \geq 1.9231177$.

Note that, in this proof, the input instance contains n large jobs. However, the result is still valid when there are no large jobs in the instance. This can be shown by simply replacing n jobs of size a with n jobs of size b where $b \in I_2$ in the above instance. \square

We have thus proved

Proposition 14 $\lim_{M \rightarrow \infty} R_{HH_M}^\infty = U_\infty = 1.9231177 \dots$

Simulation study

In this section, we present an experimental study of the four algorithms. Since this problem is NP-hard we did not attempt to find optimal off-line solutions but rather measured the performance of each algorithm by calculating the ratio of the on-line solution/lower bound. This ratio is recalculated every time a new job is considered. A lower bound for the optimal makespan is j intervals, where $j = \min\{k : \sum_{i=k+1}^n p'_i \leq k\}$. Note that for HH_M we investigated the performance where M is set to 4,7 and 10.

We study the performance of the scheduling algorithms on job sequences which are generated by the following distributions: bounded normal, bounded exponential, triangular and uniform. We also consider the cases where jobs are uniformly distributed and arrive in either non-decreasing or non-increasing order of processing time. The instances consist of 500 jobs and jobs are bounded between either $[.1, 1]$ (in 10 steps) or $[.01, 1]$ (in 100 steps). Each simulation was performed 10 times and the mean performance is presented here.

The results show that the performance of the algorithms do not appear to be significantly affected by the particular probability distribution used. However, it seems to be influenced by the number of steps. When jobs are bounded exponentially distributed, that is when there are more small

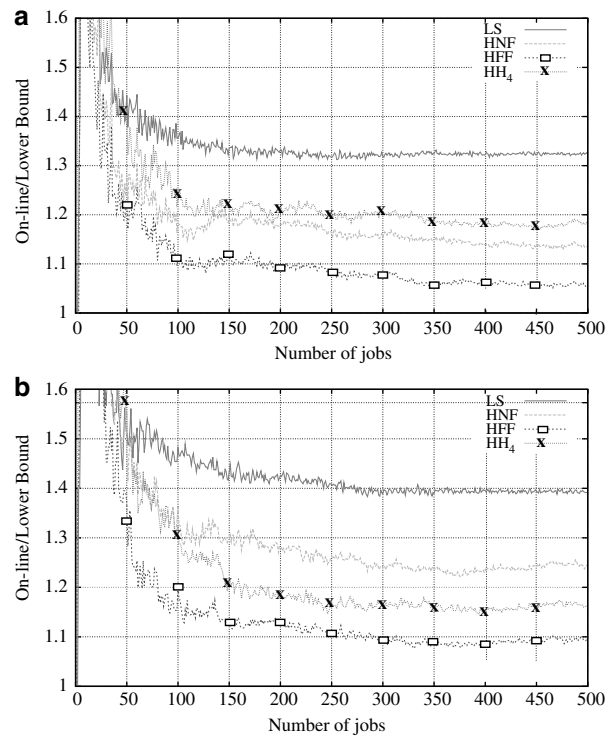


Figure 2 Graphs of the ratios of LS , HFF , HNF and HH_M when jobs are generated by the bounded exponential distribution. Bounded exponentially distributed and (a) 10 steps, (b) 100 steps.

jobs than large jobs, *HFF* outperforms *HNF* and *HH_M* significantly and *LS* performs worst. This is because *LS* puts small jobs in forbidden zones. Whereas *HFF*, *HNF*, *HH_M* leave forbidden zones for larger jobs. When the number of

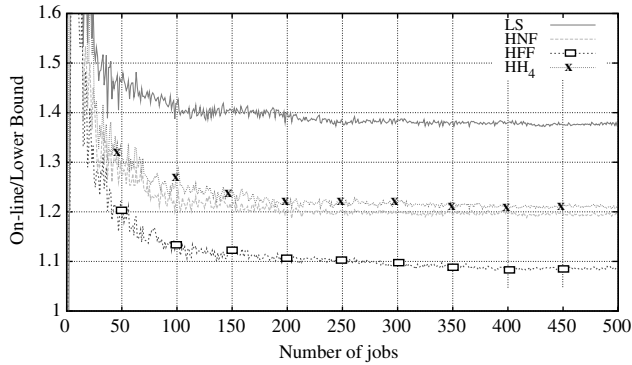


Figure 3 Graphs of the ratios of *LS*, *HFF*, *HNF* and *HH_M* when jobs are generated by the uniform distribution in 100 steps.

steps is ten, *HNF* is better than *HH_M*. However, when we set the number of steps to 100, *HH_M* outperforms *HNF*. The results of the simulations are shown in Figure 2a and b. Note that *HH₄* is used to exhibit the performance of *HH_M* since, for all distributions studied, *HH₄* performed better or similarly to *HH₇* and *HH₁₀*.

For the other three distributions, namely bounded normal, uniform and triangular, *HFF* still performs better than *HNF* and *HH_M*, and *LS* is the worst. With these distributions, the numbers of small and large jobs are similar. *HFF*, *HNF* and *HH_M* would put only larger jobs in forbidden zones, whereas *LS* may put some smaller jobs in such zones. *HFF* outperforms *HNF* and *HH_M* because it keeps several active allowed zones for future jobs whereas *HNF* has only one active allowed zone at a time and *HH_M* can only schedule *i* jobs to an *I_i* allowed zone (hence, the smaller *i* create a larger gap in an allowed zone). With all three distributions, when the number of steps is set to 100 *HNF* outperforms *HH_M*. However, in the case of 10 steps, *HH_M* performs better than *HNF* when the job sizes are

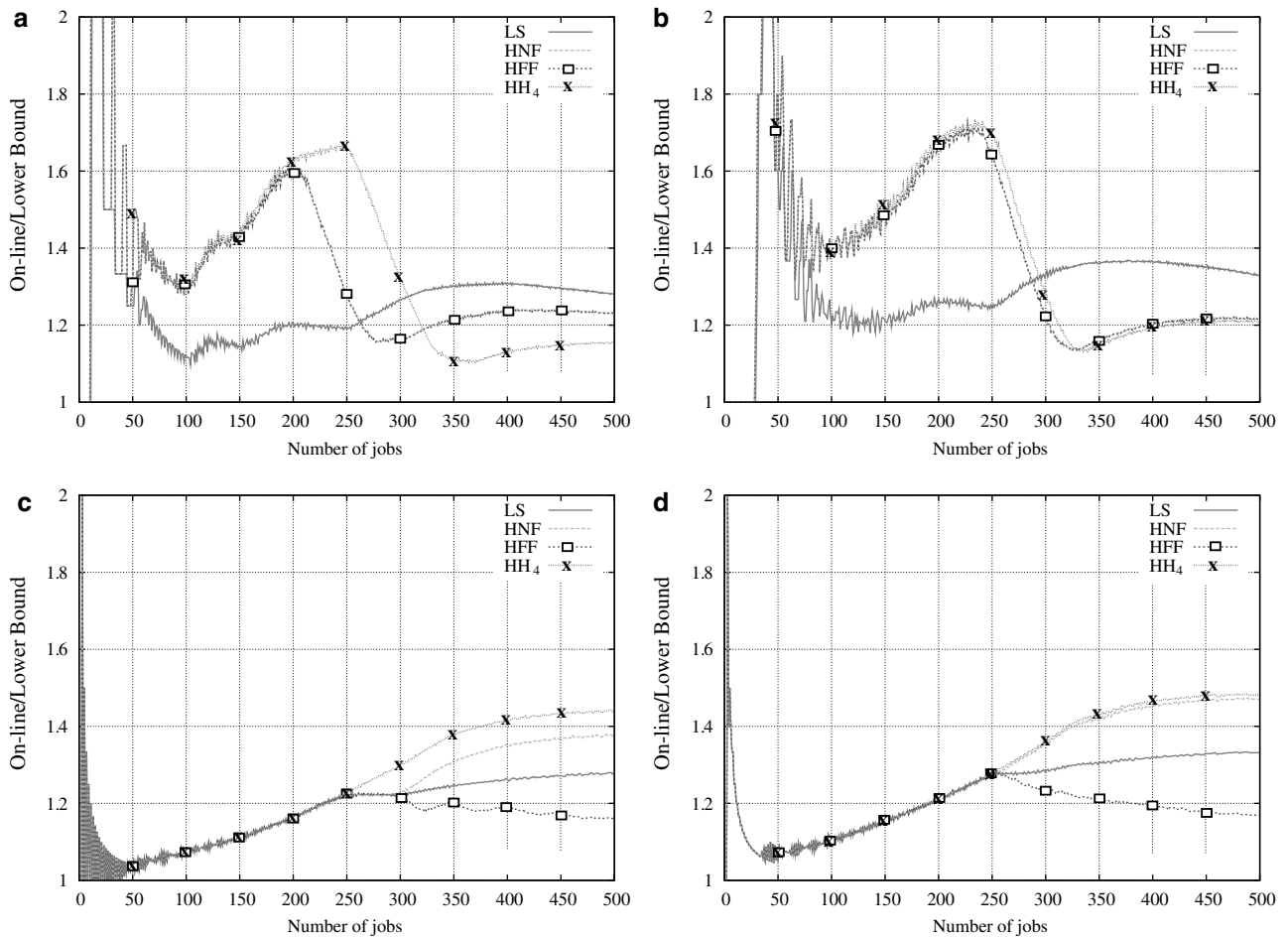


Figure 4 Graphs of the ratios of *LS*, *HFF* and *HNF* when jobs are generated by the uniform distribution and arrive in non-decreasing or non-increasing order. Non-decreasing order and (a) 10 steps, (b) 100 steps. Non-increasing order and (c) 10 steps, (d) 100 steps.

normally and uniformly distributed and both perform similarly when job sizes are created by a triangular distribution. Figure 3 depicts the performance of four algorithms when the job sizes are uniformly distributed with 100 steps.

When jobs sizes are uniformly distributed and jobs arrive in non-decreasing order, *HNF* and *HFF* perform identically, because the non-decreasing property of the arriving job sizes means that *HFF* cannot put an incoming job in a previous allowed zone. Initially *HNF*, *HFF* and HH_M are worse than *LS*, since smaller jobs fill in the allowed zones leaving forbidden zones empty. Once jobs which are not less than $\frac{1}{2}$ arrive, the empty forbidden zones are then filled by these jobs resulting in a better performance for *HNF*, *HFF* and HH_M . At this point, *HNF*, *HFF* and HH_M are both better than *LS*. However, once all the empty forbidden zones are filled, the ratios for *HFF*, *HNF* and HH_M begin increasing again but they are still better than *LS*.

Some interesting results can also be found when job sizes are uniformly distributed and sequenced in non-increasing order. These results are shown in Figure 4c and d. Here, at the beginning, all four algorithms perform identically. Once smaller jobs arrive, *HFF* outperforms *LS* and *HNF*, whereas HH_M is the worst. The reason is that *HFF* has some gaps in allowed zones for smaller jobs, whereas *HNF* cannot use such gaps since it considers only one allowed zone at a time and HH_M leaves too many gaps in allowed zones since the smaller jobs cannot be scheduled to the allowed zones reserved for larger jobs.

Conclusions and future work

In this paper, we discussed the one-by-one on-line scheduling problem for a single machine with forbidden zones. We provided a lower bound for the competitive ratio for any on-line algorithm and discussed the relationship between this problem and bin packing. Four on-line algorithms were proposed, for which we derived various asymptotic performance ratio results. An experimental study of these algorithms revealed that their performance was not significantly affected by the probability distribution used. However, their comparative mean performance (as empiri-

cally determined) differed from that established by the earlier worst-case analysis.

There are various ways to extend the above results. Can the lower bound result be improved? Is there an algorithm with a lower asymptotic performance ratio than HH_M ? What if $F \neq \frac{1}{2}I$, or if $F_i \neq F$ and $I_i \neq I$. How is the problem affected if we allow p_i to exceed 1?

References

- Abdekhodae A and Ernst AT (2004a). *Scheduling jobs with forbidden zones*. Technical report, Commonwealth Scientific and Industrial Research Organisation, Australia.
- Abdekhodae A and Ernst AT (2004b). Scheduling jobs with forbidden zones in the context of coal supply chain. In: Van Wassenhove N *et al* (eds). *Proceeding of European Operations Management Association 2004 Operations Management as a Change Agent*. INSEAD, Fontainebleau, pp 669–676.
- Coffman Jr EG, Garey MR and Johnson DS (1997). Approximation algorithms for bin packing: a survey. In: Hochbaum DS (ed). *Approximation Algorithms for NP-hard Problem*. PWS publishing company, Boston, pp 46–93.
- Garey MR, Graham RL and Ullman JD (1972). Worst-case analysis of memory allocation algorithms. In: *Proceedings of the Fourth Annual ACM Symposium on Theory of Computing*. ACM, Denver, pp 143–150.
- Johnson DS *et al.* (1974). Worst case performance bounds for simple one-dimensional packing algorithms. *SIAM J Comput* **3**: 299–325.
- Johnson DS (1974). Fast algorithms for bin packing. *J Comput Syst Sci* **8**: 272–314.
- Lee CC and Lee DT (1985). A simple on-line bin-packing algorithm. *J Assoc Comput Mach* **32**: 562–572.
- Liang FM (1980). A lower bound for on-line bin packing. *Inform Process Lett* **10**: 76–79.
- Pruhs K, Sgall J and Torng E (2004). Online scheduling. In: Leung J (ed). *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. CRC Press, Boca Raton, pp 15-1–15-41.
- Sgall J (1998). On-line scheduling: a survey. In: Fiat A and Woeginger G (eds). *Online Algorithms: The State of Art, Lecture Notes in Computer Science 1442*. Springer-Verlag, Berlin, pp 196–231.
- van Vliet A (1992). An improved lower bound for online bin packing algorithms. *Inform Process Lett* **43**: 277–284.

*Received February 2005;
accepted September 2005 after one revision*