Discrete Mathematics and Theoretical Computer Science **3**, 1999, 155–166

Analysis of an Approximation Algorithm for Scheduling Independent Parallel Tasks

Keqin Li[†]

Department of Mathematics and Computer Science, State University of New York, New Paltz, NY 12561-2499, USA

received April 14, 1998, *revised May 28, 1999*, *accepted May 30, 1999*.

In this paper, we consider the problem of scheduling independent parallel tasks in parallel systems with identical processors. The problem is NP-hard, since it includes the bin packing problem as a special case when all tasks have unit execution time. We propose and analyze a simple approximation algorithm called H_m, where m is a positive integer. Algorithm H_m has a moderate asymptotic worst-case performance ratio in the range $[1\frac{2}{3}..1\frac{13}{18}]$ for all $m \geq 6$; but the algorithm has a small asymptotic worst-case performance ratio in the range $[1+1/(r+1)..1+1/r]$, when task sizes do not exceed $1/r$ of the total available processors, where $r > 1$ is an integer. Furthermore, we show that if the task sizes are independent, identically distributed (i.i.d.) uniform random variables, and task execution times are i.i.d. random variables with finite mean and variance, then the average-case performance ratio of algorithm H_m is no larger than 1.2898680..., and for an exponential distribution of task sizes, it does not exceed 1.2898305.... As demonstrated by our analytical as well as numerical results, the average-case performance ratio improves significantly when tasks request for smaller numbers of processors.

Keywords: approximation algorithm, average-case performance ratio, parallel task scheduling, probabilistic analysis, worst-case performance ratio

1 Introduction

In this paper, we consider the problem of scheduling independent parallel tasks in parallel systems with identical processors. Assume that we are given a list of n tasks $L = (T_1, T_2, ..., T_n)$. Each task T_i is specified by its execution time τ_i, and its size ρ_i, i.e., T_i requires ρ_i processors to execute. There are M identical processors, and any ρ_i processors can be allocated to T_i. Once T_i starts to execute, it runs without interruption until it completes. Tasks in L are mutually independent, that is, there is no precedence constraint nor data communication among the n tasks. The problem addressed here is to find a nonpreemptive schedule of L such that its makespan (i.e., the total execution time of the n tasks) is minimized.

The problem is NP-hard, since it includes the bin packing problem as a special case when all tasks have unit execution time. Therefore, one practical way to solve this problem is to design and analyze approximation algorithms that produce near optimal solutions. Let $A(L)$ be the makespan of the schedule

[†]The author can be reached at email: li@mcs.newpaltz.edu, phone: (914) 257-3534, fax: (914) 257-3571.

generated by an algorithm A for L, and $\mathrm{OPT}(L)$ be the makespan of an optimal schedule of L. The quantity

$$R_A^\infty = \lim_{Z\to\infty}\left[\sup_{\mathrm{OPT}(L)=Z}\left(\frac{A(L)}{\mathrm{OPT}(L)}\right)\right]$$

is called the *asymptotic worst-case performance ratio* of algorithm A. If there exist two constants α and β such that for all L, $A(L) \le \alpha \cdot \mathrm{OPT}(L) + \beta\tau^*$, where $\tau^* = \max_{1\le i\le n}(\tau_i)$ is the longest execution time of the n tasks, then $R_A^\infty \le \alpha$, and α is called an *asymptotic worst-case performance bound* of algorithm A. Moreover, if for any small $\epsilon > 0$ and all large $Z > 0$, there exists a list L, such that $\mathrm{OPT}(L) \ge Z$, and $A(L) \ge (\alpha - \epsilon)\mathrm{OPT}(L)$, then the bound α is called tight, i.e., $R_A^\infty = \alpha$. When task sizes and execution times are random variables, both $A(L)$ and $\mathrm{OPT}(L)$ become random variables, and

$$\bar{R}_A^\infty = \lim_{n\to\infty}\left[\frac{E(A(L))}{E(\mathrm{OPT}(L))}\right]$$

is called the *asymptotic average-case performance ratio* of algorithm A, where $E(\cdot)$ stands for the expectation of a random variable. Of course, $\bar{R}_A^\infty$ depends on the probability distributions of task sizes and execution times. If there exists a constant γ such that for all L, $E(A(L)) \le \gamma \cdot E(\mathrm{OPT}(L))$ as $n \to \infty$, then $\bar{R}_A^\infty \le \gamma$, and γ is called an *asymptotic average-case performance bound* of algorithm A.

We notice that our scheduling problem defined above looks similar to but is quite different from the two dimensional rectangle packing problem [1, 2, 7, 9], where each task T_i is treated as a rectangle with width ρ_i and height τ_i. The rectangle packing model implies that processors should be allocated in contiguous groups. That is, the M processors have indices 1, 2, 3, ..., M, and task T_i must be allocate ρ_i processors with indices $j, j+1, ..., j+\rho_i-1$ for some j. Such a scheduling problem arises in parallel systems like linear arrays. The rectangle packing problem has been extensively studied, where a complicated algorithm with asymptotic worst-case performance ratio as low as 1.25 has been found [1]. However, contiguous processor allocation is not required in our model, where any ρ_i processors can be allocated to T_i. Our problem could be regarded as a resource constraint scheduling problem [3, 6], where the resource is a set of processors. It has applications in parallel computing systems such as symmetric shared memory multiprocessors and distributed computing systems such as bus-connected networks of workstations. In these systems, a processor allocation mechanism is independent of the topology of an interconnection network. Another related problem is scheduling malleable tasks which has also been investigated in the literature [8, 10]. In that problem, each task requests for several possible numbers of processors, i.e., a task has adjustable size, and for each size, an execution time is also specified. The problem has several variations depending on different ways in which the execution time of a task changes with the number of processors allocated to it, and the performance measures to be optimized (e.g., makespan, average flow time).

The problem we consider here is to schedule nonmalleable tasks with noncontiguous processor allocation. Even though the complicated algorithm in [1] for rectangle packing can also be applied to solve our problem, we propose and analyze a simple approximation algorithm called H_m, where m is a positive integer. Algorithm H_m has a moderate asymptotic worst-case performance ratio ($1.666... \le R_{\mathrm{H}_m}^\infty \le 1.722...$ for all $m \ge 6$); but the algorithm has a small asymptotic worst-case performance ratio $1 + 1/(r+1) \le R_{\mathrm{H}_m}^\infty \le 1 + 1/r$, when task sizes do not exceed M/r, where $r > 1$ is an integer. We notice that the capability to deal with small tasks is important in real applications since many task sizes are relatively small as compared with the system size so that a large scale parallel system can be shared by many users simultaneously. However, it is not clear whether the algorithm in [1] has such capability. Furthermore, the simplicity of our algorithm allows us to conduct average-case performance analysis. In particular, we show that if the numbers of processors requested by the tasks are independent, identically distributed (i.i.d.) random variables uniformly distributed in the range $[1..M]$, and task execution times are i.i.d. random variables with finite mean and variance, then $\bar{R}^{\infty}_{\mathrm{H}_m} \leq 1.2898680...$, i.e., $E(\mathrm{H}_m(L))/E(\mathrm{OPT}(L))$ is asymptotically bounded from above by 1.2898680... as $n \to \infty$. For an exponential distribution of task sizes, we have $\bar{R}^{\infty}_{\mathrm{H}_m} \leq 1.2898305....$ As demonstrated by our analytical as well as numerical results, the average-case performance ratio improves significantly when tasks request for smaller numbers of processors. We notice that there is lack of such results on probabilistic algorithm analysis, especially in multi-dimensional cases [4]. The average-case performance of the algorithm in [1] is unknown.

The rest of the paper is organized as follows. We present algorithm H_m in Section 2. The worst-case performance of the algorithm is analyzed in Section 3, and its average-case performance is studied in Section 4. Finally we give a summary in Section 5.

2 The Approximation Algorithm H_m

Our algorithm H_m for scheduling independent parallel tasks divides L into m sublists L_1, L_2, ..., L_m according to task sizes (i.e., numbers of processors requested by tasks), where $m \geq 1$ is a positive integer. For $1 \leq k \leq m-1$, define $L_k = \{T_i \in L \mid M/(k+1) < \rho_i \leq M/k\}$, i.e., L_k contains all tasks in L that have sizes in the interval $I_k = (M/(k+1), M/k]$. Define $L_m = \{T_i \in L \mid 0 < \rho_i \leq M/m\}$, i.e., L_m contains all tasks whose sizes are in the range $I_m = (0, M/m]$.

Algorithm H_m produces schedules of the L_k's sequentially and separately. To process tasks in L_k, where $1 \leq k \leq m-1$, the M processors are partitioned into k groups, $G_1, G_2, ..., G_k$, each contains M/k processors. Each group G_j of processors is treated as a unit, and is assigned to a task in L_k. Such an allocation can be implemented using, for example, the list scheduling algorithm [4]. Suppose $L_k = (T^k_1, T^k_2, ..., T^k_{n_k})$, where n_k is the number of tasks in L_k. Initially, group G_j is assigned to T^k_j, where $1 \leq j \leq k$, and $T^k_1, T^k_2, ..., T^k_k$ are removed from L_k. Upon the completion of a task T^k_j, the first unscheduled task in L_k, i.e., T^k_{k+1}, is removed from L_k and scheduled to execute on G_j. This process repeats until all tasks in L_k are finished. Then algorithm H_m begins the scheduling of next sublist L_{k+1}.

For L_m, there is no need to divide the M processors. The list scheduling algorithm is again employed here. Let $L_m = (T^m_1, T^m_2, ..., T^m_{n_m})$. Initially, as many tasks in L_m are scheduled as possible, i.e., tasks $T^m_1, T^m_2, ..., T^m_s$ start their execution, where s is defined in such a way that the total size of $T^m_1, T^m_2, ..., T^m_s$ is no larger than M, but the total size of $T^m_1, T^m_2, ..., T^m_{s+1}$ exceeds M. When a task finishes, the next task in L_m begins its execution, provided that there are enough idle processors. Notice that it is the scheduling of tasks in L_m that takes advantage of noncontiguous processor allocation.

3 Combinatorial Analysis

In this section, we analyze the worst-case performance of algorithm H_m. Let $\mathrm{H}_m(L)$ be the makespan of the schedule produced by algorithm H_m for L, and $\mathrm{OPT}(L)$ be the makespan of an optimal schedule of L. First, we show the following result.

Theorem 1. *For any list L of n tasks and $m \geq 3$, we have*

$$\mathrm{H}_m(L) \leq 1\frac{13}{18}\mathrm{OPT}(L) + (m-1)\tau^*,$$

where τ^* is the longest execution time of the n tasks. Furthermore, for $m \geq 6$ and any large $Z > 0$, there exists L, such that $\mathrm{OPT}(L) \geq Z$, and $\mathrm{H}_m(L)/\mathrm{OPT}(L) = 1\frac{2}{3}$. Therefore, for all $m \geq 6$, we have $1.666... \leq R^{\infty}_{\mathrm{H}_m} \leq 1.722....$

Proof. Assume that all tasks in L are executed in the time interval $[0, \mathrm{H}_m(L)]$, and that tasks in L_k are scheduled in $[t_k, t_{k+1}]$, where $1 \leq k \leq m$, i.e., the first task in L_k starts at time t_k, and the last completion time of the tasks in L_k is t_{k+1}. Let s_k be the starting time of the last task $T^k_{n_k}$ in L_k. Define $z_1 = t_2 - t_1$, and for all $2 \leq k \leq m$, define $z_k = s_k - t_k$, and $r_k = t_{k+1} - s_k$ be the remaining execution time of L_k once $T^k_{n_k}$ starts. Clearly, $r_k \leq \tau^*$, and

$$\begin{aligned} \mathrm{H}_m(L) &= z_1 + z_2 + z_3 + \cdots + z_m + r_2 + r_3 + \cdots + r_m \\ &\leq z_1 + z_2 + z_3 + \cdots + z_m + (m-1)\tau^*. \end{aligned} \tag{1}$$

We give several lower bounds for $\mathrm{OPT}(L)$. First, tasks in L_1 have large sizes such that no two of them can execute in parallel. Therefore, we have

$$\mathrm{OPT}(L) \geq z_1. \tag{2}$$

Second, there can be at most two tasks from L_2 that can execute at the same time, and there can be at most one task from L_2 that can execute simultaneously with a task in L_1. Thus,

$$\mathrm{OPT}(L) \geq z_1 + \frac{2z_2 - z_1}{2} = \frac{z_1}{2} + z_2. \tag{3}$$

Third, define the area of a task T_i to be $\rho_i\tau_i$, and the total area of L_k to be $A_k = \sum_{T_i \in L_k} \rho_i\tau_i$ for $1 \leq k \leq m$, and $A = A_1 + A_2 + \cdots + A_m$. For convenience, we assume that processor requirements are normalized such that $0 < \rho_i \leq 1$ for all $1 \leq i \leq n$. Clearly, $\mathrm{OPT}(L) \geq A$. We give a lower bound for A as follows. For $1 \leq k \leq m-1$, we notice that all the k groups G_1, G_2, ..., G_k are busy until $T^k_{n_k}$ starts its execution. That is, during time interval $[t_k, s_k]$, at least $k/(k+1)$ of the processors are busy, i.e., we have $A_k > (k/(k+1))z_k$. For L_m, we note that during time interval $[t_m, s_m]$, the percentage of busy processors is at least $(m-1)/m$, i.e., $A_m > ((m-1)/m)z_m$; otherwise, some tasks should start earlier. Hence,

$$\mathrm{OPT}(L) \geq \frac{1}{2}z_1 + \frac{2}{3}z_2 + \frac{3}{4}z_3 + \cdots + \frac{m-1}{m}z_{m-1} + \frac{m-1}{m}z_m. \tag{4}$$

Now let us consider the ratio

$$Q = \frac{z_1 + z_2 + z_3 + \cdots + z_{m-1} + z_m}{\max\left(z_1,\ \frac{1}{2}z_1 + z_2,\ \frac{1}{2}z_1 + \frac{2}{3}z_2 + \cdots + \frac{m-1}{m}z_{m-1} + \frac{m-1}{m}z_m\right)}. \tag{5}$$

Let $x = z_3 + \cdots + z_m$. Apparently,

$$Q \leq \frac{z_1 + z_2 + x}{\max\left(z_1,\ \frac{1}{2}z_1 + z_2,\ \frac{1}{2}z_1 + \frac{2}{3}z_2 + \frac{3}{4}x\right)}. \tag{6}$$

We give the proof of the following result in the appendix.

$$\frac{z_1 + z_2 + x}{\max\left(z_1,\ \frac{1}{2}z_1 + z_2,\ \frac{1}{2}z_1 + \frac{2}{3}z_2 + \frac{3}{4}x\right)} \leq 1\frac{13}{18}, \quad \text{where } z_1, z_2, x \geq 0. \tag{7}$$

Combining Equations (1)–(7), we get $\mathrm{H}_m(L) \le 1\frac{13}{18}\mathrm{OPT}(L) + (m-1)\tau^*$.

To show the lower bound for $R^{\infty}_{\mathrm{H}_m}$, let us consider a list L of tasks which contains n tasks of size $\frac{1}{2}+\epsilon$, n tasks of size $\frac{1}{3}+\epsilon$, and n tasks of size $\frac{1}{6}-2\epsilon$, where n is a multiple of 6, and $\epsilon > 0$ is a very small quantity. All tasks have unit execution time. Clearly, $\mathrm{OPT}(L) = n$. Algorithm H_m divides L into L_1, L_2, and L_6, and $\mathrm{H}_m(L) = n + n/2 + n/6 = 1\frac{2}{3}n$. Thus, we can choose n a sufficiently large number while keeping $\mathrm{H}_m(L)/\mathrm{OPT}(L) = 1\frac{2}{3}$. This completes the proof of the theorem. ■

For tasks with small sizes, algorithm H_m exhibits much better performance due to increasing processor utilization, as claimed by the following theorem.

Theorem 2. *For any list L of n tasks, such that $\rho_i \le M/r$ for all i, where $r > 1$ is an integer, we have*

$$\mathrm{H}_m(L) \le \left(1 + \frac{1}{r}\right)\mathrm{OPT}(L) + (m-r+1)\tau^*,$$

where τ^* is the longest execution time of the n tasks. Furthermore, for $m \ge r+1$ and any large $Z > 0$, there exists L, such that $\mathrm{OPT}(L) \ge Z$, and $\mathrm{H}_m(L)/\mathrm{OPT}(L) = 1 + 1/(r+1)$. Therefore, we have $1 + 1/(r+1) \le R^{\infty}_{\mathrm{H}_m} \le 1 + 1/r$.

Proof. The proof is similar to that of Theorem 1. Since $r \ge 2$, and sublists L_1, L_2, ..., L_{r-1} are empty, Equation (1) becomes

$$\mathrm{H}_m(L) \le z_r + z_{r+1} + \cdots + z_m + (m-r+1)\tau^*.$$

By using the area lower bound for $\mathrm{OPT}(L)$, we have

$$\begin{aligned} \mathrm{OPT}(L) &\ge \left(\frac{r}{r+1}\right)z_r + \left(\frac{r+1}{r+2}\right)z_{r+1} + \cdots + \left(\frac{m-1}{m}\right)z_{m-1} + \left(\frac{m-1}{m}\right)z_m \\ &\ge \frac{r}{r+1}(z_r + z_{r+1} + \cdots + z_m). \end{aligned}$$

The above two inequalities give the asymptotic worst-case performance bound $1 + 1/r$ in the theorem. To show the lower bound $1 + 1/(r+1)$ for $R^{\infty}_{\mathrm{H}_m}$, let us consider a list L which contains nr tasks of size $1/(r+1)+\epsilon$, and n tasks of size $1/(r+1) - r\epsilon$, where n is a multiple of $r+1$, and ϵ is a sufficiently small value. All tasks have unit execution time. Clearly, $\mathrm{OPT}(L) = n$. Algorithm H_m divides L into L_r and L_{r+1}, and $\mathrm{H}_m(L) = n + n/(r+1)$. Thus, we can choose n sufficiently large while keeping $\mathrm{H}_m(L)/\mathrm{OPT}(L) = 1 + 1/(r+1)$. ■

When $r \ge 5$, algorithm H_m has better asymptotic worst-case performance ratio than the algorithm in [1].

4 Probabilistic Analysis

Now let us consider the average-case performance of algorithm H_m. For convenience, we assume the task sizes are normalized such that $0 < \rho_i \le 1$, and that the ρ_i's are independent, identically distributed (i.i.d.) random variables with a common probability density function $f(x)$ in the range $(0, 1]$. Our assumption on the task execution times is quite general, i.e., the τ_i's are i.i.d. random variables with mean μ and variance

σ^2, where μ and σ are any finite numbers independent of n. The probability distributions of task sizes and execution times are independent of each other.

Theorem 3. *We have the following asymptotic average-case performance bound for algorithm* H_m*:*

$$\bar{R}^{\infty}_{\mathrm{H}_m} = \lim_{n\to\infty}\left[\frac{E(\mathrm{H}_m(L))}{E(\mathrm{OPT}(L))}\right] \le \frac{\displaystyle\sum_{k=1}^{m-1}\frac{1}{k}\int_{\frac{1}{k+1}}^{\frac{1}{k}} f(x)dx + \frac{m}{m-1}\int_0^{\frac{1}{m}} xf(x)dx}{\max\left(\displaystyle\int_0^1 xf(x)dx,\ \int_{\frac{1}{2}}^1 f(x)dx\right)}.$$

(Note that the bound only depends on $f(x)$*.)*

Proof. It is clear that the mean task size is

$$\bar{\rho} = \int_0^1 xf(x)dx.$$

Since a task size falls into I_k with probability $\int_{I_k} f(x)dx$, where $I_k = (1/(k+1), 1/k]$ for all $1 \le k \le m-1$, and $I_m = (0, 1/m]$, the expected number of tasks in L_k is

$$E(n_k) = \left[\int_{I_k} f(x)dx\right]n,$$

for all $1 \le k \le m$. Also, the expected size of tasks in L_k is

$$\bar{\rho}_k = \frac{\displaystyle\int_{I_k} xf(x)dx}{\displaystyle\int_{I_k} f(x)dx}.$$

Since the area of a task T_i has expectation $\bar{\rho}\mu$, and tasks in L_1 have to be executed sequentially, we have

$$E(\mathrm{OPT}(L)) \ge \max\,(n\bar{\rho}\mu, E(n_1)\mu) = Dn\mu, \tag{8}$$

where

$$D = \max\,(\bar{\rho}, E(n_1)/n) = \max\left(\int_0^1 xf(x)dx,\ \int_{\frac{1}{2}}^1 f(x)dx\right).$$

Let H_k be the makespan of the schedule for L_k. Then, $E(\mathrm{H}_m(L)) = E(H_1) + E(H_2) + \cdots + E(H_m)$. Clearly,

$$E(H_1) = E(n_1)\mu = \left[\int_{\frac{1}{2}}^1 f(x)dx\right]n\mu. \tag{9}$$

For $2 \le k \le m-1$, we have $E(H_k) = E(z_k) + E(r_k)$, and

$$E(z_k) \le \frac{E(n_k)}{k}\mu.$$

Furthermore, r_k is no more than the maximum of k random execution times. It is well known from order statistics [5] that the mean of the maximum of q i.i.d. random variables $X_1, X_2, ..., X_q$ with mean μ and variance σ^2 is

$$E(\max(X_1, X_2, ..., X_p)) \leq \mu + \frac{q-1}{\sqrt{2q-1}}\,\sigma.$$

Therefore,

$$\begin{aligned} E(H_k) &\leq \frac{E(n_k)}{k}\mu + \mu + \frac{k-1}{\sqrt{2k-1}}\,\sigma \\ &= \left[\frac{1}{k}\left(\int_{I_k} f(x)dx\right)n + 1\right]\mu + \frac{k-1}{\sqrt{2k-1}}\,\sigma. \end{aligned} \tag{10}$$

Finally, we consider H_m. Since

$$E(A_m) = E(n_m)\bar{\rho}_m\mu = \left[\int_{I_m} xf(x)dx\right]n\mu,$$

and the processor utilization is at least $1 - 1/m$ in the time interval $[t_m, s_m]$, we get

$$E(H_m) \leq \frac{E(A_m)}{1 - \frac{1}{m}} + E(r_m) = \frac{m}{m-1}\left[\int_{I_m} xf(x)dx\right]n\mu + E(r_m).$$

For $E(r_m)$, the main difficulty is that when task $T^m_{n_m}$ starts execution, the number of active tasks still in execution is unknown, which could be as large as n_m, the total number of tasks in L_m. Since $E(n_m)$ could be $\Theta(n)$, we use the following quite loose upper bound for $E(r_m)$, that is, r_m is no more than the maximum of n random execution times,

$$E(r_m) \leq \mu + \frac{n-1}{\sqrt{2n-1}}\,\sigma < \mu + \sqrt{\frac{n}{2}}\,\sigma.$$

Therefore, we get

$$E(H_m) \leq \left(\frac{m}{m-1}\left[\int_{I_m} xf(x)dx\right]n + 1\right)\mu + \sqrt{\frac{n}{2}}\,\sigma. \tag{11}$$

Combining Equations (9)–(11), we obtain

$$\begin{aligned} E(\mathrm{H}_m(L)) \leq\ & \left(\int_{\frac{1}{2}}^{1} f(x)dx + \sum_{k=2}^{m-1}\frac{1}{k}\int_{\frac{1}{k+1}}^{\frac{1}{k}} f(x)dx \right. \\ & \left. + \frac{m}{m-1}\int_0^{\frac{1}{m}} xf(x)dx + \frac{m-1}{n}\right)n\mu + \left(\sum_{k=2}^{m-1}\frac{k-1}{\sqrt{2k-1}} + \sqrt{\frac{n}{2}}\right)\sigma. \end{aligned}$$

Notice that

$$\frac{k-1}{\sqrt{2k-1}} < \sqrt{\frac{k}{2}}$$

for all $k \geq 1$. Consequently,

$$\sum_{k=2}^{m-1} \frac{k-1}{\sqrt{2k-1}} < \sum_{k=2}^{m-1} \sqrt{\frac{k}{2}} < \int_2^m \sqrt{\frac{x}{2}} dx < \frac{\sqrt{2}}{3} m^{1.5}.$$

The above calculations give rises to

$$E(\mathrm{H}_m(L)) \leq \left(\sum_{k=1}^{m-1} \frac{1}{k} \int_{\frac{1}{k+1}}^{\frac{1}{k}} f(x)dx + \frac{m}{m-1} \int_0^{\frac{1}{m}} xf(x)dx + \frac{m-1}{n} \right) n\mu + \left(\frac{\sqrt{2}}{3} m^{1.5} + \sqrt{\frac{n}{2}} \right) \sigma. \tag{12}$$

Using Equations (8) and (12), we obtain

$$\begin{aligned} \frac{E(\mathrm{H}_m(L))}{E(\mathrm{OPT}(L))} &\leq \frac{1}{D} \left(\sum_{k=1}^{m-1} \frac{1}{k} \int_{\frac{1}{k+1}}^{\frac{1}{k}} f(x)dx + \frac{m}{m-1} \int_0^{\frac{1}{m}} xf(x)dx + \frac{m-1}{n} \right) \\ &\quad + \frac{1}{D} \left(\frac{\sqrt{2}}{3} \cdot \frac{m^{1.5}}{n} + \sqrt{\frac{1}{2n}} \right) \frac{\sigma}{\mu} \\ &= \frac{1}{D} \left(\sum_{k=1}^{m-1} \frac{1}{k} \int_{\frac{1}{k+1}}^{\frac{1}{k}} f(x)dx + \frac{m}{m-1} \int_0^{\frac{1}{m}} xf(x)dx \right) + O\left(\frac{m^{1.5}}{n} + \frac{1}{\sqrt{n}} \right). \end{aligned}$$

It is clear that as $n \to \infty$, we get the asymptotic average-case performance bound in the theorem. ■

As an example, let us consider the uniform distributions, that is, the ρ_i's are i.i.d. random variables uniformly distributed in the range $(0, 1/r]$, where $r \geq 1$ is a positive integer. That is, $f(x) = r$ for $0 < x \leq 1/r$. (Notice that when M is sufficiently large, a discrete uniform distribution on $\{1, 2, ..., M/r\}$ can be treated as a continuous uniform distribution on $(0, 1/r]$.)

Theorem 4. *If the ρ_i's be i.i.d. random variables uniformly distributed in the range $(0, 1/r]$, we have $\bar{R}^{\infty}_{\mathrm{H}_m} \leq B_r$, that is,*

$$E(\mathrm{H}_m(L)) \leq B_r E(\mathrm{OPT}(L)),$$

as $n \to \infty$, where

$$B_r = 2r^2 \left[\frac{\pi^2}{6} - \frac{1}{r} - \left(1 + \frac{1}{2^2} + \cdots + \frac{1}{(r-1)^2} \right) \right],$$

as $m \to \infty$.

Proof. We examine the numerator and denominator of the bound in Theorem 3. The denominator is simply

$$D = \max \left(\int_0^1 xf(x)dx, \int_{\frac{1}{2}}^1 f(x)dx \right) = \frac{1}{2r},$$

and the numerator is

$$N = r \left(\sum_{k=r}^{m-1} \frac{1}{k^2(k+1)} + \frac{1}{2m(m-1)} \right).$$

Since

$$\frac{1}{k^2(k+1)} = \frac{1}{k^2} - \frac{1}{k} + \frac{1}{k+1},$$

we have

$$\sum_{k=r}^{m-1} \frac{1}{k^2(k+1)} = \left(\sum_{k=r}^{m-1} \frac{1}{k^2}\right) - \frac{1}{r} + \frac{1}{m}.$$

Note that

$$\sum_{k=r}^{m-1} \frac{1}{k^2} = \sum_{k=1}^{\infty} \frac{1}{k^2} - \sum_{k=1}^{r-1} \frac{1}{k^2} - \sum_{k=m}^{\infty} \frac{1}{k^2} \le \frac{\pi^2}{6} - \sum_{k=1}^{r-1} \frac{1}{k^2} - \frac{1}{m}.$$

Thus, we have

$$\sum_{k=r}^{m-1} \frac{1}{k^2(k+1)} \le \frac{\pi^2}{6} - \frac{1}{r} - \left(1 + \frac{1}{2^2} + \cdots + \frac{1}{(r-1)^2}\right),$$

and

$$N \le r\left(\frac{\pi^2}{6} - \frac{1}{r} - \left(1 + \frac{1}{2^2} + \cdots + \frac{1}{(r-1)^2}\right) + \frac{1}{2m(m-1)}\right).$$

By choosing m sufficiently large, the average-case performance bound N/D can be made arbitrarily close to B_r. ■

When $r = 1$, the asymptotic average-case performance bound given in Theorem 4 is $B_1 = \pi^2/3 - 2 = 1.2898680...$. To show the quality of the average-case performance bound B_r in Theorem 4, we give the following numerical data.

$$\begin{aligned}
B_1 &= 1.2898680... \\
B_2 &= 1.1594720... \\
B_3 &= 1.1088121... \\
B_4 &= 1.0823327... \\
B_5 &= 1.0661449... \\
B_6 &= 1.0552487... \\
B_7 &= 1.0474219... \\
B_8 &= 1.0415306... \\
B_9 &= 1.0369372... \\
B_{10} &= 1.0332559...
\end{aligned}$$

It is clear that $B_r < 1 + 1/r$ for all $r > 1$, i.e., B_r is less than the asymptotic worst-case performance bound in Theorem 2.

Though closed form solutions are not available, the average-case performance bounds of algorithm H_m could be calculated using Theorem 3 numerically for arbitrary probability distribution of task sizes. For instance, let us consider a truncated exponential distribution, i.e.,

$$f(x) = \frac{\lambda e^{-\lambda x}}{1 - e^{-\lambda}}, \qquad 0 < x \le 1.$$

Theorem 5. *If the ρ_i's be i.i.d. random variables exponentially distributed in the range $(0, 1]$, we have $\bar{R}^{\infty}_{\mathrm{H}_m} \le B_\lambda$, that is,*

$$E(\mathrm{H}_m(L)) \le B_\lambda E(\mathrm{OPT}(L)),$$

as $n \to \infty$, where

$$B_\lambda = \frac{\displaystyle\sum_{k=1}^{\infty} \frac{1}{k} \cdot \frac{e^{-\lambda/(k+1)} - e^{-\lambda/k}}{1 - e^{-\lambda}}}{\displaystyle\frac{1}{\lambda} - \frac{e^{-\lambda}}{1 - e^{-\lambda}}},$$

as $m \to \infty$.

Proof. It can be easily verified by straightforward calculation that the numerator and denominator in Theorem 3 are

$$N = \sum_{k=1}^{m-1} \frac{1}{k} \cdot \frac{e^{-\lambda/(k+1)} - e^{-\lambda/k}}{1 - e^{-\lambda}} + \frac{m}{m-1} \cdot \frac{1}{1 - e^{-\lambda}} \left(\frac{1 - e^{-\lambda/m}}{\lambda} - \frac{e^{-\lambda/m}}{m} \right),$$

and

$$D = \max\left(\frac{1}{\lambda} - \frac{e^{-\lambda}}{1 - e^{-\lambda}}, \ \frac{e^{-\lambda/2} - e^{-\lambda}}{1 - e^{-\lambda}} \right) = \frac{1}{\lambda} - \frac{e^{-\lambda}}{1 - e^{-\lambda}},$$

respectively. By letting $m \to \infty$, the average-case performance bound N/D can be made arbitrarily close to B_λ. ∎

To show the average-case performance bound B_λ in Theorem 5, we let $m = 1024$, and choose λ in such a way that the mean task size

$$\bar{\rho} = \frac{1}{\lambda} - \frac{e^{-\lambda}}{1 - e^{-\lambda}}$$

takes the values $1/(2r)$ for $r = 1, 2, ..., 10$, so that a comparison can be made between performance bounds of H_m under the uniform and the exponential distributions.

$$\begin{aligned}
\lambda &= 0.0001813..., & B_\lambda &= 1.2898305... \\
\lambda &= 3.5935119..., & B_\lambda &= 1.2731064... \\
\lambda &= 5.9030000..., & B_\lambda &= 1.2145755... \\
\lambda &= 7.9781077..., & B_\lambda &= 1.1640016... \\
\lambda &= 9.9954411..., & B_\lambda &= 1.1273400... \\
\lambda &= 11.9991145..., & B_\lambda &= 1.1021181... \\
\lambda &= 13.9998370..., & B_\lambda &= 1.0846745... \\
\lambda &= 15.9999711..., & B_\lambda &= 1.0722076... \\
\lambda &= 17.9999950..., & B_\lambda &= 1.0629307... \\
\lambda &= 19.9999991..., & B_\lambda &= 1.0557653...
\end{aligned}$$

As shown in the above list, B_λ is slightly smaller than $\pi^2/3 - 2$, when $\bar{\rho} = 0.5$, i.e, $r = 1$, due to distribution imbalance in $(0, 1]$. However, B_λ is larger than B_r for all $r > 1$, because $E(n_1)$ is never null.

5 Conclusions

We have studied the problem of scheduling independent nonmalleable parallel tasks in parallel systems with identical processors. We proposed a simple approximation algorithm called H_m, and performed combinatorial analysis for its worst-case performance and probabilistic analysis for its average-case performance. In particular, we proved the following results. (1) The asymptotic worst-case performance ratio $R_{\mathrm{H}_m}^{\infty}$ is in the range $[1\frac{2}{3}..1\frac{13}{18}]$. (2) If the numbers of processors requested by the tasks are uniformly distributed i.i.d. random variables and task execution times are i.i.d. random variables with finite mean and variance, then the average-case performance ratio is $\bar{R}_{\mathrm{H}_m}^{\infty} \leq 1.2898680....$ In other words, less than 22.5% of the allocated computing power is wasted. (3) Both the worst- and average-case performance ratios improve significantly when tasks request for smaller numbers of processors. (4) Results similar to (2)–(3) also hold for the truncated exponential distribution of task sizes.

Acknowledgements

The author wishes to express his gratitude to the editor and two anonymous referees for their comments on improving the paper. This research was supported in part by National Aeronautics and Space Administration and the Research Foundation of State University of New York through the NASA/University Joint Venture in Space Science Program under Grant NAG8-1313.

References

[1] B.S. Baker, D.J. Brown, and H.P. Katseff, "A 5/4 algorithm for two-dimensional packing," *Journal of Algorithms* **2**, 348-368, 1981.

[2] B.S. Baker and J.S. Schwarz, "Shelf algorithms for two-dimensional packing problems," *SIAM Journal on Computing* **12**, 508-525, 1983.

[3] J. Blazewicz, J.K. Lenstra, and A.H.G. Rinnooy Kan, "Scheduling subject to resource constraints: classification and complexity," *Discrete Applied Mathematics* **5**, 11-24, 1983.

[4] E.G. Coffman, Jr., D.S. Johnson, P.W. Shor, and G.S. Lueker, "Probabilistic analysis of packing and related partitioning problems," in *Probability and Algorithms*, 87-107, National Research Council, 1992.

[5] H.A. David, *Order Statistics*, John Wiley & Sons, 1970.

[6] M.R. Garey and R.L. Graham, "Bound for multiprocessor scheduling with resource constraints," *SIAM Journal on Computing* **4**, 187-200, 1975.

[7] I. Golan, "Performance bounds for orthogonal oriented two-dimensional packing algorithms," *SIAM Journal on Computing* **10**, 571-582, 1981.

[8] R. Krishnamurti and E. Ma, "An approximation algorithm for scheduling tasks on varying partition sizes in partitionable multiprocessor systems," Technical Report RC 15900, IBM Research Division, 1990.

[9] D.K.D.B. Sleator, "A 2.5 times optimal algorithm for bin packing in two dimensions," *Information Processing Letters* **10**, 37-40, 1980.

[10] J. Turek, J.L. Wolf, K.R. Pattipati, and P.S. Yu, "Scheduling parallelizable tasks: putting it all on the shelf," *Proc. International Conference on Measurement and Modeling of Computer Systems*, 225-236, 1992.

Appendix. Proof of Equation (7)

The following fact is used in the proof. If

$$g(x) = \frac{ax+b}{cx+d},$$

where $a, b, c, d, x > 0$, then $g(x)$ is an increasing (decreasing) function of x if $ad - bc > 0$ (< 0). We consider three cases.

Case 1. z_1 is the maximum. Since $z_1 \geq \frac{1}{2}z_1 + z_2$, we have $z_2 \leq \frac{1}{2}z_1$. Since $z_1 \geq \frac{1}{2}z_1 + \frac{2}{3}z_2 + \frac{3}{4}x$, we have $x \leq \frac{4}{3}(\frac{1}{2}z_1 - \frac{2}{3}z_2)$. Hence,

$$\begin{aligned} Q &= 1 + \frac{z_2}{z_1} + \frac{x}{z_1} \\ &= 1 + \frac{z_2}{z_1} + \frac{4}{3}\left(\frac{1}{2} - \frac{2}{3} \cdot \frac{z_2}{z_1}\right) \\ &= 1 + \frac{2}{3} + \frac{1}{9} \cdot \frac{z_2}{z_1} \\ &\leq 1 + \frac{2}{3} + \frac{1}{9} \cdot \frac{1}{2} \\ &= 1\frac{13}{18}. \end{aligned}$$

Case 2. $\frac{1}{2}z_1 + z_2$ is the maximum. Since $z_1 \leq \frac{1}{2}z_1 + z_2$, we have $z_1 \leq 2z_2$. Since $\frac{1}{2}z_1 + z_2 \geq \frac{1}{2}z_1 + \frac{2}{3}z_2 + \frac{3}{4}x$, we get $x \leq \frac{4}{9}z_2$. Now,

$$Q = \frac{z_1 + z_2 + x}{\frac{1}{2}z_1 + z_2} \leq \frac{z_1 + \frac{13}{9}z_2}{\frac{1}{2}z_1 + z_2},$$

which is an increasing function of z_1. Hence Q takes its maximum value $1\frac{13}{18}$ when $z_1 = 2z_2$.

Case 3. $\frac{1}{2}z_1 + \frac{2}{3}z_2 + \frac{3}{4}x$ is the maximum. Since $\frac{1}{2}z_1 + z_2 \leq \frac{1}{2}z_1 + \frac{2}{3}z_2 + \frac{3}{4}x$, we get $x \geq \frac{4}{9}z_2$. Note that

$$Q = \frac{x + z_1 + z_2}{\frac{3}{4}x + \frac{1}{2}z_1 + \frac{2}{3}z_2}$$

is a decreasing function of x. Thus, Q gets its maximum value when $x = \frac{4}{9}z_2$, i.e.,

$$Q \leq \frac{z_1 + \frac{13}{9}z_2}{\frac{1}{2}z_1 + z_2},$$

which is an increasing function of z_1. Since, $z_1 \leq \frac{1}{2}z_1 + \frac{2}{3}z_2 + \frac{3}{4}x$, and $x = \frac{4}{9}z_2$, we have $z_1 \leq 2z_2$. Hence Q reaches its maximum value $1\frac{13}{18}$ when $z_1 = 2z_2$.