Discrete Applied Mathematics 6 (1983) 173-191
North-Holland

BIN PACKING AND MULTIPROCESSOR SCHEDULING PROBLEMS WITH SIDE CONSTRAINT ON JOB TYPES

I. MORIHARA
Yokosuka Electrical Communication Laboratory, N.T.T., Yokosuka-shi, Japan

T. IBARAKI* and T. HASEGAWA
Department of Applied Mathematics and Physics, Faculty of Engineering, Kyoto University, Kyoto, Japan

Received 6 August 1980
Revised 3 May 1982

This paper deals with the bin packing problem and the multiprocessor scheduling problem both with an additional constraint specifying the maximum number of jobs in each type to be processed on a processor. Since these problems are NP-complete, various approximation algorithms are proposed by generalizing those algorithms known for the ordinary bin packing and multiprocessor scheduling problems. The worst-case performance of the proposed algorithms are analyzed, and some computational results are reported to indicate their average case behavior.

1. Introduction

This paper studies the multiprocessor scheduling problem and the bin packing problem, which have a side constraint concerning the number of jobs of the same type to be processed on each processor. This problem arises in various production situations, typically in the following problem setting.

Let n types of machines $M_1, M_2, \ldots, M_n$ be used to produce n types of goods $I_1, I_2, \ldots, I_n$ respectively. A machine M_i produces a unit of good I_i a day, requiring a_i workers. The number of available machines of type M_i is m_i for $i=1,2,\ldots,n$. Our goal is to produce b_i units of I_i within a certain time span. Then, the following two problems can be considered.

Problem 1. Assuming that the number of workers usable in a day is at most P, minimize the number of days needed to produce all the required goods.

Problem 2. Assuming that the number of days spent to produce all the required goods is limited to T, minimize the maximum number of workers needed a day.

All the numbers T, P, a_i, b_i and m_i are assumed to be positive integers. The con-

*Currently with Dept. of Information and Computer Sciences, Toyohashi Univ. of Technology, Toyohashi, Japan.

ditions $a_i \leq P$ in Problem 1 and $\lceil b_i/m_i \rceil \leq T$ in Problem 2 must hold for $i=1,2,\ldots,n$ in order that the problems are feasible.[1]

If we remove from Problems 1 and 2 the constraint that at most m_i machines can be used in a day (we call this the *machine constraint*), we obtain the standard bin packing problem and multiprocessor scheduling problem, respectively. These problems are known to be NP-complete, implying that it is most unlikely to have efficient algorithms to obtain exact optimal solutions [3, 4, 5, 7, 8]. Hence we consider approximation algorithms based on heuristics for Problems 1 and 2 in the subsequent discussion.

Heuristics such as *first fit* (FF), *best fit* (BF), *first fit decreasing* (FFD) and *best fit decreasing* (BFD) are known [6] for the bin packing problem. Also heuristics *largest processing time* (LPT) and *multifit* (MF) have been studied [1, 4] for the multiprocessor scheduling problem.

In this paper, we extend these approximation algorithms to our problems by taking into account the machine constraint. We investigate their worst case behavior theoretically and also their average case behavior by computational experiment.

In Sections 2–6, approximation algorithms (some of which are direct adaptations of the previously known algorithms, and others are new) are introduced and their worst case behavior is examined. For Problem 1, the approximate values obtained by each algorithm don't exceed the optimal values by more than 100%. For Problem 2, the approximate values obtained by LPT algorithm don't exceed the optimal values by more than 33.3%. It is also shown that these are the best possible bounds.

Finally in Section 7, the average case behavior of these approximation algorithms is examined by the computational experiment. Test problems are randomly generated. In all cases, if we adopt an appropriate approximation algorithm, we can obtain approximate solutions whose average errors from the optimal values are within about 16%.

2. Approximation algorithms for Problem 1

If $b_i = m_i = 1$ holds for all $i=1,2,\ldots,n$ in Problem 1 (i.e., the machine constraint vanishes), it is the so-called bin packing problem.

Bin Packing Problem. Given a list $A=\{a_1, a_2, \ldots, a_n\}$ of positive integers representing the sizes of n elements and an arbitrary number of bins with capacity P, place all the n elements into a minimum number of bins such that the level of each bin does not exceed P, where the level of a bin is the total size of the elements placed in it.

Then Problem 1 can be described as the bin packing problem with an additional

[1] For a real number A, $\lceil A \rceil$ ($\lfloor A \rfloor$) denotes the least (largest) integer value not less (not more) than A.

constraint: Given three lists $A=\{a_1,a_2,\dots,a_n\}$, $B=\{b_1,b_2,\dots,b_n\}$ and $M=\{m_1,m_2,\dots,m_n\}$ of positive integers, indicating that there are b_i number of elements A_i with size a_i, and an arbitrary number of bins with capacity P, denoted $\mathrm{BIN}_1,\mathrm{BIN}_2,\dots,$ place all A_i's $(i=1,2,\dots,n)$ into a minimum number of bins such that the level of each bin does not exceed P and the number of A_i's placed in a bin does not exceed m_i.

In the following we propose some algorithms for Problem 1 by generalizing the known approximation algorithms for the bin packing problem.

First Fit (FF) Algorithm. Each A_i (recall that there are b_i A_i's) is placed one at a time into BIN_j with the least j among those having the level $\le P-a_i$ and having at most m_i-1 A_i's. This placement is executed in the order of $i=1,2,\dots,n$.

Best Fit (BF) Algorithm. Each A_i is placed one at a time into BIN_j with the maximum level among those having the level $\le P-a_i$ and having at most m_i-1 A_i's (the least j is selected if the tie occurs). This placement is executed in the order of $i=1,2,\dots,n$.

FDA Algorithm (BDA Algorithm). Arrange indices i in the nonincreasing order of a_i and apply FF (BF) algorithm in the resulting order of i.

FDC Algorithm (BDC Algorithm). The parameters

$$c_i=\lceil b_i/m_i\rceil \quad \text{for } i=1,2,\dots,n \tag{1}$$

represent the number of bins needed to place b_i A_i's only. Each c_i may represent the strength of the machine constraint. Thus arrange indices i in the nonincreasing order of c_i (if $c_i=c_j$ and $a_i>a_j$ $(i\ne j)$, let i be ahead of j) and apply FF (BF) algorithm in the resulting order of i.

FFC Algorithm. Note that $T_{\mathrm{LOW}}=\max\{\lceil \sum_i a_ib_i/P\rceil,\ \max_i(c_i)\}$ is a lower bound of the number of the required bins. Partition the elements into three sets as follows.

$$S_1=\{A_i \mid a_i>\tfrac{1}{2}P\},$$

$$S_2=\{A_i \mid a_i\le\tfrac{1}{2}P, c_i\ge\tfrac{1}{2}T_{\mathrm{LOW}}\},$$

$$S_3=\{A_i \mid a_i\le\tfrac{1}{2}P, c_i<\tfrac{1}{2}T_{\mathrm{LOW}}\}.$$

Then execute the following steps.

Step 1. Place all the elements $A_i\in S_1$ (there are b_i A_i's) into the bins according to FDA algorithm.

Step 2. For each $A_i\in S_2$, partition b_i A_i's into C groups, where $C=\max_i(c_i)$, such that the number of A_i's in group j, denoted by b_{ij}, satisfies

$$b_{ij}=\begin{cases} m_i, & 1\le j\le c_i-1\\ b_i-m_i(c_i-1), & j=c_i\\ 0, & j>c_i.\end{cases}$$

If S_2 contains elements $A_{i_1}, A_{i_2}, \ldots, A_{i_k}$, place $b_{i_l j}$ of A_{i_l} into bins in the order of $l=1,2,\ldots,k$ according to FDA algorithm, and repeat it for $j=1,2,\ldots,C$.

Step 3. In the same way as Step 2, place all elements $A_i \in S_3$ into bins.

3. Approximation algorithms for Problem 2

Recall the following problem.

Multiprocessor Scheduling Problem. Given n nonpreemptive (i.e., cannot be divided) independent jobs and T identical processors, minimize the total timespan required to process all the jobs.

Problem 2 can be considered as the multiprocessor scheduling problem with an additional constraint. We state it in the context of the bin packing problem. Given three lists $A=\{a_1,a_2,\ldots,a_n\}$, $B=\{b_1,b_2,\ldots,b_n\}$ and $M=\{m_1,m_2,\ldots,m_n\}$ of positive integers, and T bins $\mathrm{BIN}_1, \mathrm{BIN}_2, \ldots, \mathrm{BIN}_T$, place b_i A_i's $(i=1,2,\ldots,n)$ into these bins under the restriction that the number of A_i's placed in a bin is at most m_i for $i=1,2,\ldots,n$, so that the maximum level of the bins is minimized.

Largest processing time (LPT) algorithm and the multifit algorithm known as approximation algorithms for the multiprocessor scheduling problem can be generalized as follows.

LPT Algorithm. Arrange the indices i in the nonincreasing order of a_i. Each A_i (recall that there are b_i A_i's) is placed one at a time into BIN_j with the least j among those currently having the minimum level and having at most $m_i - 1$ A_i's. This placement is executed in the order of $i=1,2,\ldots,n$.

Multifit Algorithms. Tentatively we give a capacity P to the bins and apply a bin packing algorithm of Problem 1. Let $T_X(P)$ denote the number of necessary bins when Algorithm X (such as FF, FDA and FDC) is applied to a given P. If $T_X(P) \le T$, reduce the value P; otherwise increase the value P. Then repeat the same procedure. After testing an appropriate number of P's in the above manner, the smallest P satisfying the constraint is output as an approximate value. The search of P is usually done by binary search method.

We call the above multifit algorithms MFFF, MFFDA and MFFDC depending upon $X=\mathrm{FF}$, FDA and FDC respectively. It is shown in [8] that a feasible value P can be found by MFFDC algorithm between P_{LOW} and $2P_{\mathrm{LOW}}$, where

$$P_{\mathrm{LOW}} = \max\left\{ \left\lceil \sum_i a_i b_i / T \right\rceil, \max_i (a_i) \right\}.$$

For MFFF and MFFDA algorithms, the same lower bound and a trivial upper bound $T \times P_{\mathrm{LOW}} - T + 1$ are used [8]. The P's are then searched in these intervals by binary search.

Finally, it is also possible to consider a multifit algorithm for Problem 1 in the dual manner by repeatedly using the LPT algorithm for Problem 2. The resulting algorithm is called MFLPT. A feasible value of T can be obtained in the interval $[T_{\mathrm{LOW}}, 2T_{\mathrm{LOW}}]$, where

$$T_{\mathrm{LOW}} = \max\left\{ \left\lceil \sum_i a_i b_i / P \right\rceil, \max_i (c_i) \right\} \qquad \text{(see [8]).}$$

4. Worst case behavior of the algorithms for Problem 1

The worst case behavior of approximation algorithms for the bin packing problem has been analyzed by many researchers. The known worst case bounds for FF, BF, BDA, FDC and BDC algorithms against the optimum value T_{OPT} are as follows:

$$T_{\mathrm{FF}}\ (T_{\mathrm{BF}}) \le \tfrac{17}{10} T_{\mathrm{OPT}} + 2 \quad \text{and} \quad T_{\mathrm{FDA}}\ (T_{\mathrm{BDA}}, T_{\mathrm{FDC}}, T_{\mathrm{BDC}}) \le \tfrac{11}{9} T_{\mathrm{OPT}} + 4,$$

where T_X denotes the number of necessary bins by algorithm X.

However, if the machine constraint is imposed, these bounds ae no longer valid as shown in the following examples.

Example 1. Let $P \ge 1$ be a given integer and let $A = \{1, 1\}$, $B = \{P(P-1), P\}$, $M = \{P, 1\}$. Then $T_{\mathrm{OPT}} = P$ and $T_{\mathrm{FF}} = T_{\mathrm{BF}} = T_{\mathrm{FDA}} = T_{\mathrm{BDA}} = 2P - 1$ hold as illustrated in Fig. 1.

Example 2. (i) $P \le 2$. Let $A = \{1, 1\}$, $B = \{P^2, P\}$ and $M = \{P, 1\}$. Then $T_{\mathrm{OPT}} = P + 1$ and $T_{\mathrm{FDC}} = T_{\mathrm{BDC}} = 2P$ hold as easily proved.

(ii) $P \ge 3$. Let $A = \{1, 1\}$, $B = \{(P-1)^2, P-1\}$ and $M = \{P, 1\}$ (i.e., $c_1 = c_2$). Then $T_{\mathrm{OPT}} = P - 1$ and $T_{\mathrm{FDC}} = T_{\mathrm{BDC}} = 2P - 3$ hold as illustrated in Fig. 2.

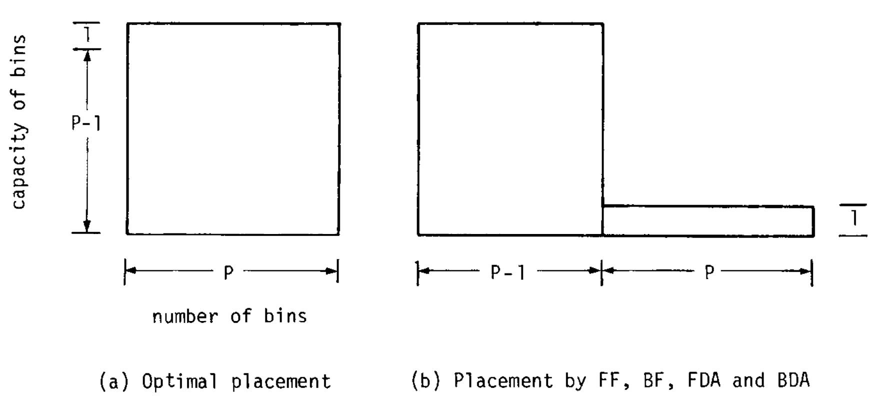

Fig. 1. Worst case example of FF, BF, FDA and BDA algorithms for Problem 1.

Furthermore, we show by the following theorem that these examples exhibit the worst cases for the proposed algorithms.

Theorem 1. *For any instance of Problem* 1,

$$T_{\mathrm{FF}}\ (T_{\mathrm{BF}}, T_{\mathrm{FDA}}, T_{\mathrm{BDA}}) \leq \left(2-\frac{1}{P}\right) T_{\mathrm{OPT}}$$

holds, where P is the capacity of bins. Furthermore these bounds are best possible.

Proof. We consider only algorithm FF since others can be similarly treated. If $a_i > \frac{1}{2}P$ for all $i = 1, 2, \ldots, n$, $T_{\mathrm{FF}} = T_{\mathrm{OPT}}$ is obvious. Otherwise, let h be the maximum index of the bin which contains an element A_k with the size $a_k \leq \frac{1}{2}P$ and let A_r be the element with the minimum size among those satisfying $a_i > \frac{1}{2}P$ and placed in BIN_j with $h < j \leq T_{\mathrm{FF}}$ by algorithm FF. Then each BIN_j with $h < j \leq T_{\mathrm{FF}}$ contains exactly one element, and

$$(\text{the level of } \mathrm{BIN}_l) \geq P - a_r + 1 \tag{2}$$

holds for $1 \leq l \leq h$. Since the number of A_i with the size $a_i > \frac{1}{2}P$ obviously must not be more than T_{OPT}, we have

$$h \geq T_{\mathrm{FF}} - T_{\mathrm{OPT}}. \tag{3}$$

Now let the number of bins BIN_j with $j < h$ which have m_k A_k's be $\bar{c}_k$, where A_k satisfies $a_k \leq \frac{1}{2}P$ and is placed in BIN_h according to algorithm FF. Each BIN_l with $1 \leq l \leq h-1$, which does not contain m_k A_k's, satisfies

$$(\text{the level of } \mathrm{BIN}_l) \geq P - a_k + 1. \tag{4}$$

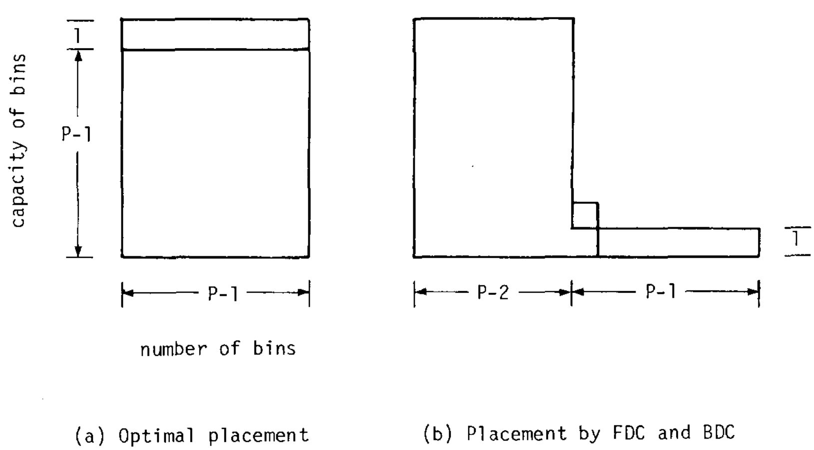

Fig. 2. Worst case example of FDC and BDC algorithms for Problem 1.

By $c_k \le T_{\mathrm{OPT}}$ and $c_k \ge \bar{c}_k + 1$ (see (1) for the definition of c_k), we also obtain

$$\bar{c}_k \le T_{\mathrm{OPT}} - 1. \tag{5}$$

Then the following inequality holds.

Case (i): $h = T_{\mathrm{FF}}$. Then we have

$$\begin{aligned} P\ T_{\mathrm{OPT}} &\ge (P - a_k + 1)(T_{\mathrm{FF}} - \bar{c}_k - 1) + m_k a_k \bar{c}_k + a_k && \text{(by (4))} \\ &\ge (P - a_k + 1)T_{\mathrm{FF}} - (P - 2a_k + 1)(\bar{c}_k + 1) && \text{(by } m_k \ge 1\text{)} \\ &\ge (P - a_k + 1)T_{\mathrm{FF}} - (P - 2a_k + 1)T_{\mathrm{OPT}} && \text{(by (5)).} \end{aligned}$$

This is equal to

$$T_{\mathrm{FF}} \le \left(2 - \frac{1}{P - a_k + 1}\right) T_{\mathrm{OPT}},$$

implying by $a_k \le \frac{1}{2}P$ (i.e. $(P - a_k + 1) > 0$) that

$$T_{\mathrm{FF}} \le \left(2 - \frac{1}{P}\right) T_{\mathrm{OPT}}.$$

Case (ii): $h \le T_{\mathrm{FF}} - 1$ and $a_k \ge P - a_r + 1$.

$$\begin{aligned} PT_{\mathrm{OPT}} &\ge a_r(T_{\mathrm{FF}} - h) + (P - a_k + 1)(h - \bar{c}_k - 1) + m_k a_k \bar{c}_k + a_k \\ &\ge a_r(T_{\mathrm{FF}} - h) + (P - a_k + 1)(h - \bar{c}_k - 1) + a_k(\bar{c}_k + 1) \\ &= a_r T_{\mathrm{FF}} + (P - a_r - a_k + 1)h - (P - 2a_k + 1)(\bar{c}_k + 1). \end{aligned}$$

By $P - a_r - a_k + 1 \le 0$, $h \le T_{\mathrm{FF}} - 1$, $a_k \le \frac{1}{2}P$ and $\bar{c}_k + 1 \le T_{\mathrm{OPT}}$, we obtain

$$PT_{\mathrm{OPT}} \ge a_r T_{\mathrm{FF}} + (P - a_r - a_k + 1)(T_{\mathrm{FF}} - 1) - (P - 2a_k + 1)T_{\mathrm{OPT}},$$

which is equal to

$$T_{\mathrm{FF}} \le \left(2 - \frac{1}{P - a_k + 1}\right) T_{\mathrm{OPT}} + \left(1 - \frac{a_r}{P - a_k + 1}\right).$$

This implies

$$T_{\mathrm{FF}} \le \left(2 - \frac{1}{P}\right) T_{\mathrm{OPT}}.$$

Case (iii): $h \le T_{\mathrm{FF}} - 1$ and $a_k \le P - a_r + 1$. By using relations (2) and (4), we have

$$\begin{aligned} PT_{\mathrm{OPT}} &\ge a_r(T_{\mathrm{FF}} - h) + (P - a_k + 1)(h - \bar{c}_k - 1) + (P - a_r + 1)(\bar{c}_k + 1) \\ &= a_r T_{\mathrm{FF}} + (P - a_r - a_k + 1)h - (a_r - a_k)(\bar{c}_k + 1) \\ &\ge a_r T_{\mathrm{FF}} + (P - a_r - a_k + 1)(T_{\mathrm{FF}} - T_{\mathrm{OPT}}) - (a_r - a_k)T_{\mathrm{OPT}} \end{aligned}$$

(by (3) and (5)).

Then,

$$T_{FF} \leq \left(2 - \frac{1}{P - a_k + 1}\right) T_{OPT},$$

implying

$$T_{FF} \leq \left(2 - \frac{1}{P}\right) T_{OPT}.$$

The second half of the theorem statement is obvious from Example 1. □

Theorem 2. *For any instance of Problem* 1,

$$T_{FDC}\ (T_{BDC}) \leq \begin{cases} \left(2 - \frac{1}{P-1}\right) T_{OPT} & (P \geq 3), \\ \left(2 - \frac{2}{P+1}\right) T_{OPT} & (P \leq 2) \end{cases}$$

holds, where P is the capacity of bins. Furthermore, these bounds are best possible.

Proof. The proof is similar to that of Theorem 1, but more involved. The details are given in [8]. □

Theorem 1 can also be applied to FFC algorithm showing $T_{FFC} \leq (2 - 1/P)T_{OPT}$. But this bound does not seem to be best possible. We have not been able to find an example satisfying $T_{FFC} \geq \frac{7}{4} T_{OPT}$. Also, a bound for MFLPT algorithm, $T_{MFLPT} \leq 2T_{OPT} - 1$ is known [8]. In this case again, we have not found an example satisfying $T_{MFLPT} \geq \frac{4}{3} T_{OPT}$.

5. Worst case behavior of the LPT algorithm for Problem 2

The LPT algorithm for Problem 2 with $b_i = m_i = 1$ (the multiprocessor scheduling problem) is known to have the best possible bound $P_{LPT}/P_{OPT} \leq \frac{4}{3} - 1/3T$, where T is the number of bins [4]. This can be extended to general Problem 2.

Theorem 3. *For any problem instance of Problem* 2, *we obtain*

$$\frac{P_{LPT}}{P_{OPT}} \leq \frac{4}{3} - \frac{1}{3T},$$

where P_{OPT} is the maximum level of the bins obtained by an optimal placement, P_{LPT} is the one obtained by LPT *algorithm, and T is the number of bins. This bound is best possible.*

We give some lemmas before proving this theorem.

Lemma 1. *Without loss of generality, we can assume $m_i = 1$ for $i = 1, 2, \ldots, n$ in proving Theorem 3.*

Proof. If $m_i \geq 2$, decompose the b_i A_i's into m_i groups, $A_{i1}, A_{i2}, \ldots, A_{im_i}$, and define b_{ij} and m_{ij} by

$$\begin{aligned} &b_{i1} = b_{i2} = \cdots = b_{ik} = \lceil b_i/m_i \rceil \quad (\leq T), \\ &b_{i,k+1} = \cdots = b_{im_i} = \lfloor b_i/m_i \rfloor, \\ &m_{i1} = m_{i2} = \cdots = m_{im_i} = 1, \end{aligned} \tag{6}$$

where $k = b_i - (\lfloor b_i/m_i \rfloor) m_i$. Obviously $\sum_{j=1}^{m_i} b_{ij} = b_i$ and $\sum_{j=1}^{m_i} m_{ij} = m_i$.

Now, if m_i A_i's are placed in a bin by LPT algorithm in the original setting, we consider that exactly one of them belongs to each of the m_i groups. By the nature of LPT algorithms, it is not difficult to see that such placement also results when LPT algorithm is applied to the newly grouped list. The same argument also applied to the case in which less than m_i A_i's are placed in a bin.

Next it is easy to prove that the original and modified problems have the same P_{OPT}. Therefore, the ratio $P_{\text{LPT}}/P_{\text{OPT}}$ obtained for a general problem does not exceed the maximum of $P_{\text{LPT}}/P_{\text{OPT}}$ obtained for problems with restriction $m_i = 1$ $(i = 1, 2, \ldots, n)$. □

By this lemma, we assume $m_i = 1$ for all $i = 1, 2, \ldots, n$ in the subsequent discussion.

Let $P_{j,i}$ denote the level of BIN_j when the placement of all A_i's has been just completed by the LPT algorithm.

Lemma 2. *For any two bins,* BIN_r *and* BIN_s $(r < s)$, *we have*

$$|P_{s,i} - P_{r,i}| \leq \max[|P_{s,i-1} - P_{r,i-1}|, a_i].$$

Proof. Assume without loss of generality that $P_{s,i-1} < P_{r,i-1}$. Then, A_i is placed either in BIN_r only or in both BIN_s and BIN_r. The above inequality is an immediate consequence of this observation. □

Lemma 3.

$$\begin{aligned} &\max\{P_{j,i} \mid j = 1, 2, \ldots, T\} - \min\{P_{j,i} \mid j = 1, 2, \ldots, T\} \\ &\quad \leq \max[\max\{P_{j,i-1} \mid j = 1, 2, \ldots, T\} - \min\{P_{j,i-1} \mid j = 1, 2, \ldots, T\}, a_i]. \end{aligned}$$

Proof. Obvious from Lemma 2. □

Lemma 4. *If* $\max\{P_{j,n} \mid j = 1, 2, \ldots, T\} - \min\{P_{j,n} \mid j = 1, 2, \ldots, T\} \leq \frac{1}{3} P_{\text{OPT}}$, *then*

$$\frac{P_{\text{LPT}}}{P_{\text{OPT}}} \leq \frac{4}{3} - \frac{1}{3T}.$$

Proof. Since $P_{\mathrm{LPT}} = \max\{P_{j,n} \mid j = 1, 2, \ldots, T\}$, we obtain

$$P_{\mathrm{LPT}} + (T-1)(P_{\mathrm{LPT}} - \tfrac{1}{3}P_{\mathrm{OPT}}) \le P_{\mathrm{LPT}} + (T-1)\min\{P_{j,n} \mid j = 1, 2, \ldots, T\}$$

$$\le \sum_{i=1}^{n} a_i b_i \le TP_{\mathrm{OPT}}.$$

This implies

$$\frac{P_{\mathrm{LPT}}}{P_{\mathrm{OPT}}} \le \frac{4}{3} - \frac{1}{3T}. \qquad \square$$

Lemma 5. *Consider the time when we have placed A_i's satisfying $a_i > \frac{1}{3}P_{\mathrm{OPT}}$ according to the* LPT *rule. For simplicity assume that $P_{j,i}$ satisfy $P_{1,i} \le P_{2,i} \le \cdots \le P_{T,i}$ by rearranging bins if necessary. Then, for any $h (1 \le h \le T)$, $\sum_{j=h}^{T} P_{j,i} \le \sum_{j=h}^{T} P'_{j,i}$ holds for the levels $P'_{j,i}$ obtained by any placement of A_1's, A_2's, ..., A_i's, where $P'_{j,i}$ are also arranged in the order of $P'_{1,i} \le P'_{2,i} \le \cdots \le P'_{T,i}$. In particular, this implies $P_{j,i} \le P_{\mathrm{OPT}}$.*

Proof. Let $\bar{P}_{j,i}$ denote the level of BIN_j for the placement of the same set of elements, which minimizes $\sum_{j=h}^{T} \bar{P}_{j,i}$, where $\bar{P}_{1,i} \le \bar{P}_{2,i} \le \cdots \le \bar{P}_{T,i} \le \bar{P}_{\mathrm{OPT}}$ is assumed. The number of elements placed in a bin is no more than two, because each element has the size greater than $\frac{1}{3}P_{\mathrm{OPT}}$. Then, for notational simplicity, we consider that exactly two elements are placed in each bin, by assigning fictitious elements of size zero to the bins containing less than two elements. The lemma is proved by showing the equality $\sum_{j=h}^{T} \bar{P}_{j,i} = \sum_{j=h}^{T} P_{j,i}$ by induction on T.

For $T = 1$, this equality is obvious.

Assuming $\sum_{j=h}^{T} \bar{P}_{j,i} = \sum_{j=h}^{T} P_{j,i}$ for $T \le k$, we show $\sum_{j=h}^{k+1} \bar{P}_{j,i} = \sum_{j=h}^{k+1} P_{j,i}$ for $T = k+1$.

Now assume $\sum_{j=h}^{k+1} \bar{P}_{j,i} < \sum_{j=h}^{k+1} P_{j,i}$, i.e. there exists a pair r and s with $1 \le r < s \le k+1$ such that the elements in BIN_r and BIN_s assigned by optimal placement are not consistent with the LPT rule (otherwise the optimal placement must be equal to the LPT placement). Let elements A_d and A_e be placed in BIN_r, and elements A_f and A_g be placed in BIN_s, where $d > e$ and $f > g$, i.e., $a_d \le a_e$ and $a_f \le a_g$ (see Fig. 3). Thus

$$\bar{P}_{r,i} = a_d + a_e \quad \text{and} \quad \bar{P}_{s,i} = a_f + a_g. \tag{7}$$

Furthermore,

$$\text{either} \quad a_d \le a_e < a_f \le a_g \quad \text{or} \quad a_d < a_f \le a_e < a_g \wedge e \ne f \tag{8}$$

holds, since in all other cases the elements in BIN_r and BIN_s are consistent with the LPT rule, as easily checked. (For example, $a_d < a_f < a_e < a_g$ is not consistent with the LPT rule; A_f must be placed in BIN_r by the LPT rule because BIN_s has a higher level than BIN_r when A_f is placed.) We consider the following six cases separately.

(I) $r<h-1$, $h<s$. (II) $r=h-1$, $h<s$.
(III) $r<h-1$, $s=h$. (IV) $r=h-1$, $s=h$.
(V) $r<s\le h-1$. (VI) $h\le r<s$.

Case (I). Consider the new placement by switching A_d and A_f, i.e. $\tilde{P}_{r,i}=a_e+a_f$, $\tilde{P}_{s,i}=a_d+a_g$, where $\tilde{P}_{j,i}$ denotes the level of BIN_j by this new placement (the index j of $\bar{P}_{j,i}$ and $\tilde{P}_{j,i}$ refers to the same bin; $\tilde{P}_{j,i}$ are not rearranged in the nondecreasing order of $\tilde{P}_{j,i}$). Obviously $\bar{P}_{r,i}<\tilde{P}_{r,i}<\bar{P}_{s,i}$ and $\bar{P}_{r,i}<\tilde{P}_{s,i}<\bar{P}_{s,i}$ hold by (7) and (8) (and $\bar{P}_{j,i}=\tilde{P}_{j,i}$ for $j\ne r,s$ hold).

The following four cases are possible.

(I.1) $\tilde{P}_{r,i}\le\tilde{P}_{h-1,i}\le\tilde{P}_{s,i}$, $\tilde{P}_{h-1,i}\le\tilde{P}_{r,i}\le\tilde{P}_{s,i}\le\tilde{P}_{h,i}$ or $\tilde{P}_{h-1,i}\le\tilde{P}_{r,i}\le\tilde{P}_{h,i}\le\tilde{P}_{s,i}$.
(I.2) $\tilde{P}_{s,i}\le\tilde{P}_{h-1,i}\le\tilde{P}_{r,i}$, $\tilde{P}_{h-1,i}\le\tilde{P}_{s,i}\le\tilde{P}_{r,i}\le\tilde{P}_{h,i}$ or $\tilde{P}_{h-1,i}\le\tilde{P}_{s,i}\le\tilde{P}_{h,i}\le\tilde{P}_{r,i}$.
(I.3) $\tilde{P}_{r,i}\le\tilde{P}_{s,i}\le\tilde{P}_{h-1,i}$ or $\tilde{P}_{s,i}\le\tilde{P}_{r,i}\le\tilde{P}_{h-1,i}$.
(I.4) $\tilde{P}_{h,i}\le\tilde{P}_{r,i}\le\tilde{P}_{s,i}$ or $\tilde{P}_{h,i}\le\tilde{P}_{s,i}\le\tilde{P}_{r,i}$.

Now rearrange the bins in the nonincreasing order of $\tilde{P}_{j,i}$, and denote the resulting order by $\hat{P}_{1,i}\le\hat{P}_{2,i}\le\cdots\le\hat{P}_{k+1,i}$.

Case (I.1). By $\bar{P}_{s,i}>\tilde{P}_{s,i}$, we have

$$\sum_{j=h}^{k+1}\bar{P}_{j,i}>\sum_{j=h}^{k+1}\bar{P}_{j,i}-(\bar{P}_{s,i}-\tilde{P}_{s,i})=\sum_{j=h}^{k+1}\hat{P}_{j,i}.$$

But this contradicts the assumption that the initial placement minimizes $\sum_{j=h}^{k+1}\bar{P}_{j,i}$.

Case (I.2). By $\bar{P}_{s,i}>\tilde{P}_{r,i}$, we obtain a contradiction in a manner similar to Case (I.1).

Case (I.3). By the assumption $h<s$, we have $\bar{P}_{h-1,i}\le\bar{P}_{h,i}\le\bar{P}_{s,i}$. If $\bar{P}_{h-1,i}<\bar{P}_{s,i}$, it holds that

$$\sum_{j=h}^{k+1}\bar{P}_{j,i}>\sum_{j=h}^{k+1}\bar{P}_{j,i}-(\bar{P}_{s,i}-\bar{P}_{h-1,i})=\sum_{\substack{j=h-1\\ j\ne s}}^{k+1}\tilde{P}_{j,i}=\sum_{j=h}^{k+1}\hat{P}_{j,i}. \tag{9}$$

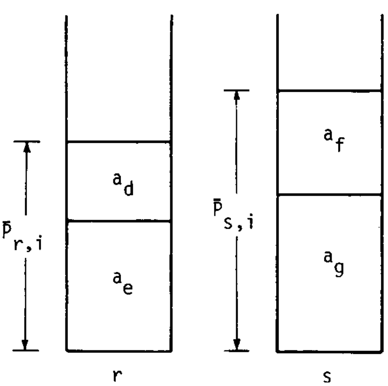

Fig. 3. BIN_r and BIN_s in the proof of Lemma 5.

This is again a contradiction to the minimality of $\sum_{j=h}^{k+1} \bar{P}_{j,i}$. On the other hand, if $\bar{P}_{h-1,i} = \bar{P}_{h,i} = \bar{P}_{s,i}$, we regard the new placement with $\hat{P}_{j,i}$ as the optimum $\bar{P}_{j,i}$, and repeat the argument given so far. The new placement however, satisfies $\tilde{P}_{s,i} < \bar{P}_{s,i} = \bar{P}_{h,i}$ and $\tilde{P}_{r,i} < \bar{P}_{s,i} = \bar{P}_{h,i}$. If the number of bins with the level $\bar{P}_{h,i}$ is α, therefore, this argument repeats at most α times.

Case (I.4). By the assumption $r<h$, we have $\bar{P}_{r,i} \le \bar{P}_{h,i}$. If $\bar{P}_{r,i} < \bar{P}_{h,i}$, it holds that

$$\sum_{j=h}^{k+1} \bar{P}_{j,i} > \sum_{j=h}^{k+1} \bar{P}_{j,i} - (\bar{P}_{h,i} - \bar{P}_{r,i}) = \sum_{j=h+1}^{k+1} \tilde{P}_{j,i} + \tilde{P}_{r,i} = \sum_{j=h}^{k+1} \hat{P}_{j,i}, \tag{10}$$

contradicting the minimality of $\sum_{j=h}^{k+1} \bar{P}_{j,i}$. On the other hand, if $\bar{P}_{r,i} = \bar{P}_{h,i}$, it can be treated similarly to the second case of (I.3).

Cases (II)–(IV). We also consider the new placement $\tilde{P}_{j,i}$ defined in Case (I). Then, the following eight cases are possible.

(II.1) $(\tilde{P}_{h-2,i}<)\ \tilde{P}_{r,i} \le \tilde{P}_{h,i} \le \tilde{P}_{s,i}$ or $(\tilde{P}_{h-2,i}<)\ \tilde{P}_{r,i} \le \tilde{P}_{s,i} \le \tilde{P}_{h,i}$.
(II.2) $(\tilde{P}_{h-2,i}<)\ \tilde{P}_{s,i} \le \tilde{P}_{h,i} \le \tilde{P}_{r,i}$ or $(\tilde{P}_{h-2,i}<)\ \tilde{P}_{s,i} \le \tilde{P}_{r,i} \le \tilde{P}_{h,i}$.
(II.3) $(\tilde{P}_{h-2,i}\le)\ \tilde{P}_{h,i} \le \tilde{P}_{r,i} \le \tilde{P}_{s,i}$ or $(\tilde{P}_{h-2,i}\le)\ \tilde{P}_{h,i} \le \tilde{P}_{s,i} \le \tilde{P}_{r,i}$.
(III.1) $\tilde{P}_{r,i} \le \tilde{P}_{h-1,i} \le \tilde{P}_{s,i}\ (<\tilde{P}_{h+1,i})$ or $\tilde{P}_{h-1,i} \le \tilde{P}_{r,i} \le \tilde{P}_{s,i}\ (<\tilde{P}_{h+1,i})$.
(III.2) $\tilde{P}_{s,i} \le \tilde{P}_{h-1,i} \le \tilde{P}_{r,i}\ (<\tilde{P}_{h+1,i})$ or $\tilde{P}_{h-1,i} \le \tilde{P}_{s,i} \le \tilde{P}_{r,i}\ (<\tilde{P}_{h+1,i})$.
(III.3) $\tilde{P}_{r,i} \le \tilde{P}_{s,i} \le \tilde{P}_{h-1,i}$ or $\tilde{P}_{s,i} \le \tilde{P}_{r,i} \le \tilde{P}_{h-1,i}$.
(IV.1) $\tilde{P}_{h-2,i} < \tilde{P}_{r,i} \le \tilde{P}_{s,i} < \tilde{P}_{h+1,i}$.
(IV.2) $\tilde{P}_{h-2,i} < \tilde{P}_{s,i} \le \tilde{P}_{r,i} < \tilde{P}_{h+1,i}$.

Applying an argument similar to Case (I), we obtain a contradiction in each of the above cases.

Case (V). We construct the new placement with level $\tilde{P}_{j,i}$ from the placement with level $\bar{P}_{j,i}$ by switching the elements in BIN_j with $1 \le j \le h-1$ so that BIN_j's for $1 \le j \le h-1$ are consistent with the LPT rule.

Arrange all the bins in the nonincreasing order of $\tilde{P}_{j,i}$, and denote the resulting order by $\hat{P}_{1,i} \le \hat{P}_{2,i} \le \cdots \le \hat{P}_{k+1,i}$. Then we have $\sum_{j=h}^{k+1} \bar{P}_{j,i} = \sum_{j=h}^{k+1} \tilde{P}_{j,i} = \sum_{j=h}^{k+1} \hat{P}_{j,i}$, because

$$(\bar{P}_{h,i} = \tilde{P}_{h,i} \ge)\ \bar{P}_{h-1,i} \ge \max\{\tilde{P}_{j,i} \mid j = 1, 2, \ldots, h-1\}$$

by the optimality of LPT algorithm for $T = h-1 \le k$. Therefore, we consider the new placement with $\hat{P}_{j,i}$ as an optimum $\bar{P}_{j,i}$ and repeat the same argument.

Case (VI). Similar to Case (V), we consider the new placement by switching the elements in BIN_j's for $h \le j \le k+1$ according to the LPT rule and repeat the argument given so far.

Consequently, we can assume that there exists no pair of bins which are not consistent with the LPT rule. So we obtain $\sum_{j=h}^{k+1} \bar{P}_{j,i} = \sum_{j=h}^{k+1} P_{j,i}$ for any h, proving the lemma statement. □

Proof of Theorem 3. Place all the elements A_i's with $a_i > \frac{1}{3} P_{\text{OPT}}$ by the LPT algorithm, and assume that $P_{1,I} \le P_{2,I} \le \cdots \le P_{T,I}$ holds by rearranging bins if necessary, where I is the maximum index i of the element with $a_i > \frac{1}{3}P_{\mathrm{OPT}}$ (recall that $a_1 \geq a_2 \cdots \geq a_n$ is assumed).

Then, we classify the bins into the following three sets.

$$\begin{aligned} S_1 &= \{\mathrm{BIN}_j \mid P_{j,I} - P_{1,I} \leq \tfrac{1}{3}P_{\mathrm{OPT}}\}, \\ S_2 &= \{\mathrm{BIN}_j \mid \tfrac{1}{3}P_{\mathrm{OPT}} < P_{j,I} - P_{1,I} \leq \tfrac{2}{3}P_{\mathrm{OPT}}\}, \\ S_3 &= \{\mathrm{BIN}_j \mid \tfrac{2}{3}P_{\mathrm{OPT}} < P_{j,I} - P_{1,I} \leq P_{\mathrm{OPT}}\}. \end{aligned} \tag{11}$$

(By Lemma 5, we have $P_{T,I} \leq P_{\mathrm{OPT}}$.)

We then place b_i A_i's for each $i = I+1, \ldots, n$ according to the LPT algorithm. During this process, construct three other sets S_2', S_3' and S_3'', where $S_2' = S_3' = S_3'' = \emptyset$ holds initially (i.e., when the placement of A_I's is completed). We move some elements in S_2, S_3 or S_3' to S_2', S_3' or S_3'' respectively by the following rule, when the placement of b_i A_i's is completed for $i = I+1, \ldots, n$.

(1) For $\mathrm{BIN}_r \in S_2$, if there exists $\mathrm{BIN}_j \in S_1 \cup S_2'$ satisfying $P_{r,i} \leq P_{j,i}$, move BIN_r to S_2'.

(2) For $\mathrm{BIN}_s \in S_3$, if there exists $\mathrm{BIN}_j \in S_2 \cup S_3'$ satisfying $P_{s,i} \leq P_{j,i}$, move BIN_s to S_3'.

(3) For $\mathrm{BIN}_t \in S_3' \cup S_3$, if there exists $\mathrm{BIN}_j \in S_1 \cup S_2'$ satisfying $P_{t,i} \leq P_{j,i}$, move BIN_t to S_3''.

Let $S_{1,i}$ denote the set S_1 when the placement of A_i's is completed. We similarly define $S_{2,i}$, $S_{2,i}'$, $S_{3,i}$, $S_{3,i}'$ and $S_{3,i}''$.

Then the following three cases are considered separately.

Case (i): $S_{3,n} \neq \emptyset$. Consider the placement in $\mathrm{BIN}_q \in S_{1,n} \cup S_{2,n}' \cup S_{2,n}$, $\mathrm{BIN}_s \in S_{3,n}'$ and $\mathrm{BIN}_t \in S_{3,n}''$. If one A_i $(I+1 \leq i \leq n)$ is placed in $\mathrm{BIN}_r \in S_{3,n}$, the same A_i is also placed in BIN_q because $P_{r,i} > P_{q,i}$ has been maintained for all $i = I+1, \ldots, n$ by the LPT rule (otherwise BIN_r has been moved to S_3' or S_3''). So, the A_i in BIN_r can not be moved to bins in $S_{1,n} \cup S_{2,n}' \cup S_{2,n}$ by condition $m_i = 1$.

We define $P_{s,n}^*$ for $\mathrm{BIN}_s \in S_{3,n}'$ as follows.

$$P_{s,n}^* = P_{s,n} - (\text{the sum of } a_i\text{'s } (I+1 \leq i \leq n) \text{ which are placed in } \mathrm{BIN}_s \text{ but not placed in } \mathrm{BIN}_q \in S_{1,n} \cup S_{2,n}' \cup S_{2,n}). \tag{12}$$

Assume that BIN_s has been moved from S_3 to S_3' when the placement of A_l's is completed. Then, any A_i with $I+1 \leq i \leq l$ placed in BIN_s are also placed in BIN_q, and A_k with $l+1 \leq k \leq n$ placed in $\mathrm{BIN}_r \in S_{3,n}$ are also placed in BIN_q and in $\mathrm{BIN}_s \in S_{3,n}'$, because $P_{s,i} > P_{q,i}$ for $I \leq i \leq l-1$ and $P_{r,k} > P_{s,k}$ for $l \leq k \leq n$.

Both $\mathrm{BIN}_r \in S_{3,n}$ and $\mathrm{BIN}_s \in S_{3,n}'$ are initially included in $S_{3,I}$, and hence $|P_{r,I} - P_{s,I}| \leq \frac{1}{3}P_{\mathrm{OPT}}$ by definition of S_3. Applying Lemma 2 for $i = I+1, \ldots, n$, we obtain $P_{r,l} - P_{s,l} \leq \frac{1}{3}P_{\mathrm{OPT}}$ ($P_{r,l} > P_{s,l}$ because $\mathrm{BIN}_r \in S_{3,l}$ and $\mathrm{BIN}_s \in S_{3,l}'$). Since any A_k with $l+1 \leq k \leq n$ placed in BIN_r is also placed in BIN_s and BIN_q, and $P_{r,n} > P_{s,n} \geq P_{s,n}^*$ is obvious by $\mathrm{BIN}_r \in S_{3,n}$, we obtain $P_{r,n} - P_{s,n}^* \leq P_{r,l} - P_{s,l} \leq \frac{1}{3}P_{\mathrm{OPT}}$, i.e.

$$P_{\mathrm{LPT}} - P_{s,n}^* \leq \tfrac{1}{3}P_{\mathrm{OPT}} \tag{13}$$

since one of the $\mathrm{BIN}_r \in S_{3,n}$ satisfies $P_{\mathrm{LPT}} = P_{r,n}$.

For $\mathrm{BIN}_t \in S''_{3,n}$, we define $P^*_{t,n}$ in the same way as $P^*_{s,n}$ of (12). Then, applying the same argument as above,

$$P_{\mathrm{LPT}} - P^*_{t,n} \le \tfrac{1}{3} P_{\mathrm{OPT}} \tag{14}$$

also follows.

By Lemma 5, the LPT algorithm minimizes $\sum_{j=T-\alpha+1}^{T} P_{j,I}$, where α is the number of bins included in $S_{3,n} \cup S'_{3,n} \cup S''_{3,n}$ (i.e., the sum is taken over the set of bins in $S_{3,n} \cup S'_{3,n} \cup S''_{3,n}$).

Now assume that some number of elements A_i's (for some i with $I+1 \le i \le n$) are placed in bins in $S_{3,n} \cup S'_{3,n} \cup S''_{3,n}$ and cannot be moved to $\mathrm{BIN}_q \in S_{1,n} \cup S'_{2,n} \cup S_{2,n}$ by condition $m_i = 1$ (i.e., A_i's are already placed in all bins in $S_{1,n} \cup S'_{2,n} \cup S_{2,n}$). The condition on A_i then implies that at least the same number of A_i's are placed in bins in $S_{3,n} \cup S'_{3,n} \cup S''_{3,n}$ in the optimal placement. These observations lead to the following inequality.

$$\sum_{\mathrm{BIN}_r \in S_{3,n}} P_{r,n} + \sum_{\mathrm{BIN}_s \in S'_{3,n}} P^*_{s,n} + \sum_{\mathrm{BIN}_t \in S''_{3,n}} P^*_{t,n} \le \alpha P_{\mathrm{OPT}}.$$

From (13), (14), property $P_{\mathrm{LPT}} = P_{r,n}$ for some $\mathrm{BIN}_r \in S_{3,n}$, and property $P_{\mathrm{LPT}} - P_{r,n} \le \frac{1}{3} P_{\mathrm{OPT}}$ for all $\mathrm{BIN}_r \in S_{3,n}$, we obtain

$$P_{\mathrm{LPT}} + (\alpha - 1)(P_{\mathrm{LPT}} - \tfrac{1}{3} P_{\mathrm{OPT}}) \le \alpha P_{\mathrm{OPT}},$$

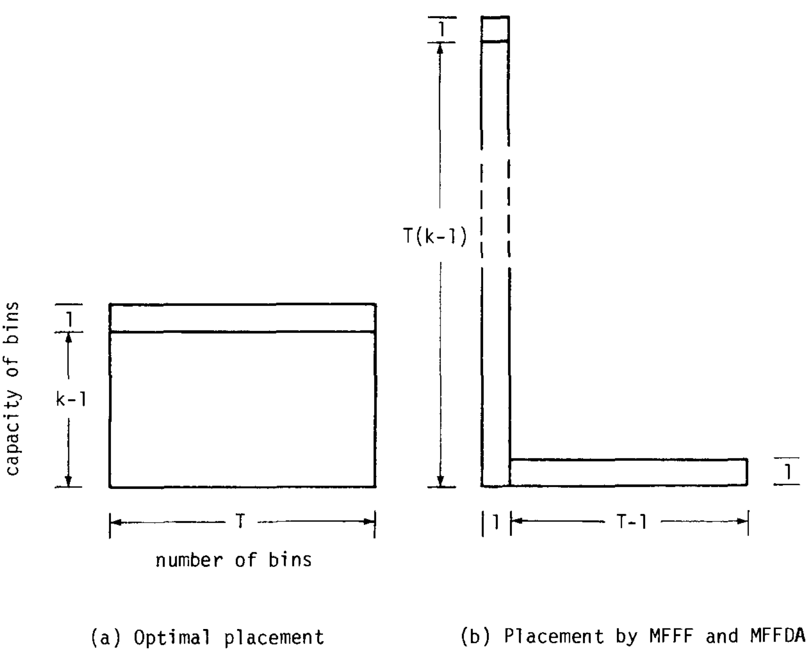

Fig. 4. Worst case example of MFFF and MFFDA algorithms for Problem 2.

which is equal to

$$\frac{P_{\mathrm{LPT}}}{P_{\mathrm{OPT}}} \leq \frac{4}{3} - \frac{1}{3\alpha} \leq \frac{4}{3} - \frac{1}{3T} \qquad (\text{by } \alpha \leq T).$$

Case (ii): $S_{3,n} = \emptyset$ and $S_{2,n} \cup S'_{3,n} \neq \emptyset$. Consider the placement in $\mathrm{BIN}_q \in S_{1,n}$, $\mathrm{BIN}_r \in S_{2,n} \cup S'_{3,n}$, $\mathrm{BIN}_s \in S'_{2,n}$ and $\mathrm{BIN}_t \in S''_{3,n}$. Treating BIN_r in the same manner as in Case (i), we can obtain

$$\frac{P_{\mathrm{LPT}}}{P_{\mathrm{OPT}}} \leq \frac{4}{3} - \frac{1}{3T}.$$

Case (iii): $S_{3,n} = \emptyset$ and $S_{2,n} \cup S'_{3,n} = \emptyset$. For any BIN_j $(1 \leq j \leq T)$, we have, $P_{\mathrm{LPT}} - P_{j,n} \leq \frac{1}{3} P_{\mathrm{OPT}}$. Therefore, we obtain by Lemma 4 that

$$\frac{P_{\mathrm{LPT}}}{P_{\mathrm{OPT}}} \leq \frac{4}{3} - \frac{1}{3T}.$$

Finally, this bound is best possible since it is the best possible bound for the multiprocessor scheduling problem (i.e., Problem 2 with $b_i = m_i = 1$) [4]. □

6. Worst case behavior of the multifit algorithms for Problem 2

The worst case bound of MFFDA algorithm known for the case of $b_i = m_i = 1$ [1] is $P_{\mathrm{MFFDA}}/P_{\mathrm{OPT}} \leq \frac{20}{17}$. But, if $b_i \neq m_i$, the worst case bounds for various multifit type algorithms become much worse as shown below.

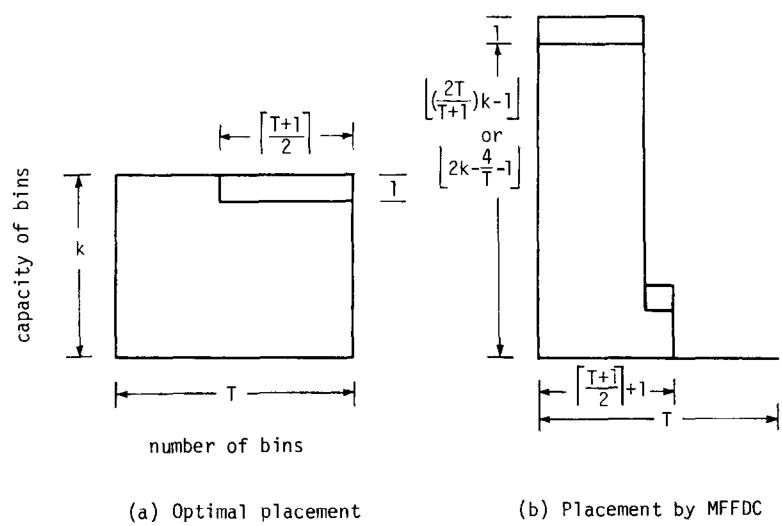

Fig. 5. Worst case example of MFFDC algorithm for Problem 2.

$$P_{\mathrm{MFFF}}\ (P_{\mathrm{MFFDA}}) \le TP_{\mathrm{OPT}} - T + 1$$

$$P_{\mathrm{MFFDC}} \le \begin{cases} \left(2 - \dfrac{2}{T+1}\right) P_{\mathrm{OPT}} & (T \text{ odd}), \\ 2P_{\mathrm{OPT}} - \dfrac{4}{T} & (T \text{ even}) \end{cases}$$

where T is the number of bins. The proofs for these results are found in [8]. Furthermore, these bounds are best possible as shown below.

Example 3. Let $A = \{1, 1\}$, $B = \{T(k-1), T\}$ (k is a positive integer), $M = \{T(k-1), 1\}$ and $T \ge 1$. Then $P_{\mathrm{OPT}} = k$ and $P_{\mathrm{MFFF}} = P_{\mathrm{MFFDA}} = T(k-1) + 1$ hold as illustrated in Fig. 4.

Example 4. If T is odd, let $A = \{1, 1\}$, $B = \{Tk - \frac{1}{2}(T+1), \frac{1}{2}(T+1)\}$ and $M = \{\lfloor 2Tk/(T+1) - 1 \rfloor, 1\}$ (k is a positive integer). Then $T_{\mathrm{OPT}} = k$ and $T_{\mathrm{MFFDC}} = \lfloor (2 - 2/(T+1))k \rfloor$ hold as illustrated in Fig. 5. On the other hand, if T is even, let $A = \{1, 1\}$, $B = \{Tk - \frac{1}{2}T - 1, \frac{1}{2}T + 1\}$ and $M = \{\lfloor 2k - 4/T - 1 \rfloor, 1\}$ (k is a positive integer). Then $T_{\mathrm{OPT}} = k$ and $T_{\mathrm{MFFDC}} = \lfloor 2k - 4/T \rfloor$ hold as illustrated in Fig. 5.

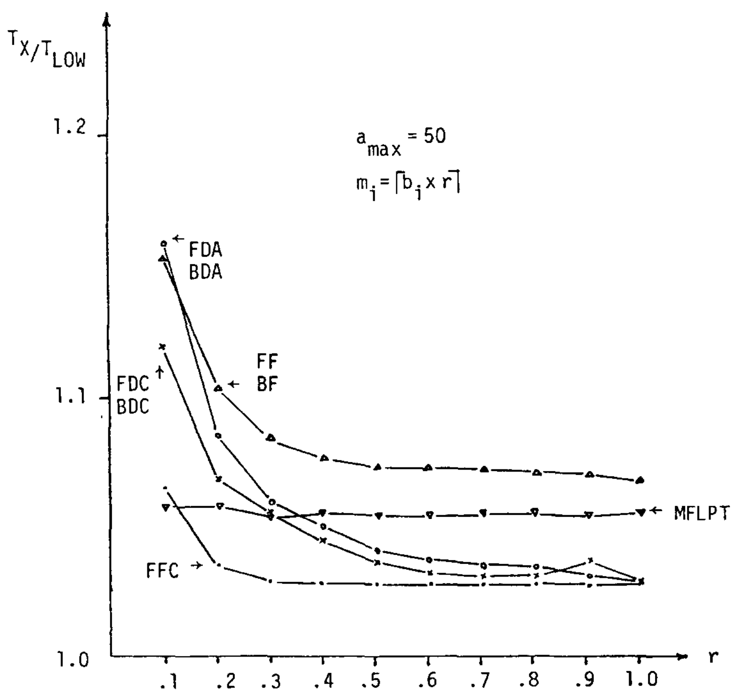

Fig. 6. Computational results for Problem 1.

Table 1
Computational time for Problem 1 in milli-seconds

n	FF	BF	FDA	BDA	FDC	BDC	MFLPT	FFC
10	0.65	1.82	0.77	2.29	0.98	2.27	10.21	0.93
20	1.45	5.73	2.11	7.68	2.56	6.99	43.40	2.27
30	2.92	11.99	4.13	16.13	4.75	14.19	97.83	3.90

7. Computational results

The average performance of the proposed approximation algorithms are investigated by computational experiment. The program is written in FORTRAN and run on FACOM M-200 (which is roughly equivalent to IBM 3033). In each case, approximation algorithms are applied to 100 problem instances which are randomly generated. Since exact optimal solutions for the generated problems are not known, the ratios of the approximate values against its lower bounds

$$T_{\mathrm{LOW}} = \max\left\{\left\lceil \sum_{i=1}^{n} a_i b_i / P \right\rceil, \max_i \lceil b_i/m_i \rceil\right\} \tag{15}$$

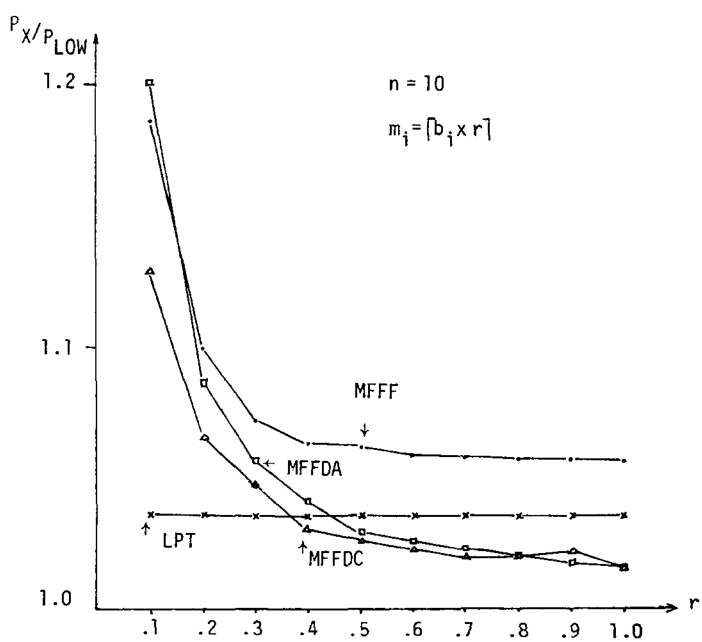

Fig. 7. Computational results for Problem 2.

for Problem 1, and

$$P_{\mathrm{LOW}} = \max\left\{\left\lceil \sum_{i=1}^{n} a_i b_i / T \right\rceil, \max_i a_i\right\} \tag{16}$$

for Problem 2, are used, thus resulting in an overestimation of the errors. The effectiveness of the algorithms are compared on the basis of the average value of the ratios for the generated 100 problems.

Average behavior of approximation algorithms for Problem 1

Instances of Problem 1 are generated as follows: $P = 100$, $n = 10$, a_i's and b_i's are randomly taken from the intervals $[1, a_{\max}]$ and $[1, 20]$ respectively, and m_i's are set to $\lceil b_i \times r \rceil$ or $\lceil b_i \times RV \times r \rceil$ for $i = 1, 2, \ldots, n$, where RV is randomly taken from $(0, 1]$, and $a_{\max}$ and r are the parameters specifying the type of problems.

In order to investigate how approximate values change according to parameters P and m_i, we set $a_{\max}$ to 100, 50, 25, 20 and r to $0.1, 0.2, \ldots, 1.0$, respectively.

As a typical example, the results for the case of $a_{\max} = 50$ and $m_i = \lceil b_i \times r \rceil$ $(i = 1, 2, \ldots, n)$ are shown for $r = 0.1, 0.2, \ldots, 1.0$ in Fig. 6. From this as well as other results we may conclude as follows. When the machine constraint is not strong (i.e., r is not small) or P is small, FFC algorithm is most recommended; otherwise, MFLPT algorithm is recommended. Note that if $b_i = m_i$ (i.e., $r = 1.0$), FDA, FDC and FFC algorithms coincide. This may explain why these algorithms exhibit similar performance for large values of r.

The computation time for each algorithm is shown in Table 1 for $a_{\max} = 50$ and $m_i = \lceil b_i \times 0.1 \rceil$ for $i = 1, 2, \ldots, n$. It may be seen that very large problems can be practically solved by these approximation algorithms.

Average behavior of approximation algorithms for Problem 2

Instances of Problem 2 are generated as follows: $T = 20$, $n = 10$, a_i's and b_i's are randomly taken from the intervals $[1, 50]$ and $[1, 20]$ respectively, and m_i's are set to $\lceil b_i \times r \rceil$ or $\lceil b_i \times RV \times r \rceil$, where RV is randomly taken from $(0, 1]$. The results for $m_i = \lceil b_i \times r \rceil$ and $r = 0.1, 0.2, \ldots, 1.0$ are shown in Fig. 7. Similar results are also obtained for other cases of experiment.

Table 2
Computational time for Problem 2 in milli-seconds

n	LPT	MFFF	MFFDA	MFFDC
10	2.09	4.14	4.49	3.66
20	4.87	8.52	9.65	8.02
30	7.31	13.09	14.86	12.92

If the machine constraint is strong (i.e., r is small), LPT algorithm is better than others, while if r is not small, MFFDC and MFFDA algorithms seem to be the best ones. These results conform to the results obtained in the previous sections by the worst case analysis.

Table 2 is the computation time for each algorithm when $m_i = \lceil b_i \times 0.1 \rceil$ is used. There is not much difference between algorithms, and the computation time is always very short.

Acknowledgement

The authors wish to thank Dr. H. Kise of Kyoto Institute of Technology, for his helpful comments.

References

[1] E.G. Coffman Jr., M.R. Garey and D.S. Johnson, An application of bin-packing to multiprocessor scheduling, SIAM J. Comput. 7 (1978) 1–17.

[2] M.R. Garey and D.S. Johnson, Complexity results for multiprocessor scheduling under resource constaints, SIAM J. Comput. 4 (1975) 397–411.

[3] M.R. Garey and D.S. Johnson, Computers and Intractibility: A Guide to the Theory of NP-Completeness (Freeman, San Francisco, 1979).

[4] R.L. Graham, Bounds on the performance of multiprocessor scheduling algorithms, Chapter 5, in: E.G. Coffman, ed., Computer and Job/Shop Scheduling Theory (Wiley, New York, 1976).

[5] D.S. Johnson, Fast algorithms for bin packing, Proc. 13th. Annual IEEE Symp. Switching and Automata Theory (1972) 144–154.

[6] D.S. Johnson, A. Demers, J.D. Ullman, M.R. Garey and R.L. Graham, Worst-case performance bounds for simple one-dimensional packing algorithms, SIAM J. Comput. 3 (1974) 299–326.

[7] R.M. Karp, Reducibility among combinatorial problems, in: R.E. Miller and J.W. Thatcher, eds., Complexity of Computer Computations (Plenum Press, New York, 1972) 85–104.

[8] I. Morihara, Approximation algorithms for bin packing and related problems with some side constraints, Master Thesis, Department of Applied Mathematics and Physics, Kyoto University, Japan (1980).