Discrete Applied Mathematics 10 (1985) 165–177
North-Holland

FIRST FIT DECREASING SCHEDULING ON UNIFORM MULTIPROCESSORS

Manfred KUNDE and Horst STEPPAT

Institut für Informatik und Praktische Mathematik, Universität Kiel, Olshausenstr. 40–60, D-2300 Kiel, W. Germany

Received 15 January 1984

Independent tasks are nonpreemptively scheduled on $m \geq 2$ processors which are assumed to have different speeds. The purpose of this paper is to show that the worst case ratio of the multifit algorithm MF, which is based on the bin-packing method FFD (first fit decreasing), depends on the order of the processors and that the MF has a better worst case behaviour than the well-known LPT algorithm for certain processor configurations.

1. Introduction and basic notations

Scheduling independent tasks on a nonpreemptive multiprocessor system is one of the fundamental problems in deterministic scheduling theory. In this paper we regard uniform multiprocessor systems, a generalization of the classical homogeneous systems [1,10,11].

Formally, there is a finite set $\mathcal{T} = \{T_1, \ldots, T_n\}$ of tasks, each having an execution time $\mu(T_i)$ and a set of $m \geq 2$ processors with speeds $s_1, s_2, \ldots, s_m$. Without loss of generality we might assume that all $s_i \geq 1$. The m-tuple of speeds is abbreviated by $S_m = (s_1, \ldots, s_m)$. A (nonpreemptive) schedule for a task set $\mathcal{T}$ and a multiprocessor system S_m is an m-tuple $\Pi(\mathcal{T}, S_m) = (P_1, \ldots, P_m)$ of m disjoint subsets of $\mathcal{T}$ with $\bigcup_{i=1}^{m} P_i = \mathcal{T}$. P_i, $1 \leq i \leq m$, denotes that subset which is executed on the ith processor. Then the finishing time or maximum completion time of $\Pi(\mathcal{T}, S_m)$ is given by

$$\omega(\Pi(\mathcal{T}, S_m)) = \max_{1 \leq i \leq m} \mu(P_i)/s_i,$$

where $\mu(X) = \sum_{T \in X} \mu(T)$ for any $X \subseteq \mathcal{T}$.

For given $\mathcal{T}$ and S_m let A be an algorithm constructing a schedule $\Pi_A(\mathcal{T}, S_m)$ whose finishing time is denoted by

$$\omega_A(\mathcal{T}, S_m) = \omega(\Pi_A(\mathcal{T}, S_m)).$$

It would be most desirable to find an algorithm OP that efficiently produces optimum schedules $\omega_{\mathrm{OP}}(\mathcal{T}, S_m)$ satisfying

$$\omega_{\mathrm{OP}}(\mathcal{T}, S_m) \leq \omega(\Pi(\mathcal{T}, S_m)) \quad \text{for all possible schedules } \Pi.$$

But determining an optimum schedule is NP-hard [14,15] and thus finding computational tractable optimum algorithms is unlikely. These bad circumstances make us seek efficient heuristics generating 'near optimum' schedules.

A measure of the quality of a heuristic algorithm A is the so called m-processor performance ratio (worst case ratio) which we analyse for various classes of uniform multiprocessors. In order to show that the performance ratio of some algorithms depends on the arrangement of the processors we distinguish seven classes:

$$\mathrm{UNIF} = \{S_m \mid m \geq 2,\ S_m = (s_1, \ldots, s_m)\},$$
the class of all uniform processor systems,

$$\mathrm{INC} = \{S_m \mid m \geq 2,\ S_m = (s_1, \ldots, s_m),\ 1 \leq s_1 \leq s_2 \leq \cdots \leq s_m\},$$
the class containing those S_m with increasing speeds, and

$$\mathrm{DEC} = \{S_m \mid m \geq 2,\ S_m = (s_1, \ldots, s_m),\ s_1 \geq s_2 \geq \cdots \leq s_m \geq 1\},$$
the class consisting of those S_m with decreasing speeds.

As important subclasses we consider uniform processor systems consisting of only one processor with a high speed and the remaining processors with all the same, slower speed.

$$\mathrm{INC1} = \{S_m \mid m \geq 2,\ S_m = (1, \ldots, 1, s_m),\ s_m \geq 1\}$$

$$\mathrm{DEC1} = \{S_m \mid m \geq 2,\ S_m = (s, 1, \ldots, 1),\ s_1 \geq 1\}$$

$$\mathrm{UNIF1} = \{S_m \mid m \geq 2,\ S_m = (s_1, \ldots, s_m),\ \exists i,\ 1 \leq i \leq m,\ \forall j,\ 1 \leq j \leq m,\ j \neq i,\ s_j = 1,\ s_i \geq 1\}.$$

The very important class of homogeneous processors we denote by

$$\mathrm{HOM} = \{S_m \mid m \geq 2,\ S_m = (1, \ldots, 1)\}.$$

If CLASS is any of the defined multiprocessor classes, then the m-processor performance ratio of a scheduling algorithm A (with respect to CLASS) is defined by

$$R_m(A, \mathrm{CLASS}) = \sup\left\{ \frac{\omega_A(\mathcal{T}, S_m)}{\omega_{\mathrm{OP}}(\mathcal{T}, S_m)} \;\middle|\; \text{all task sets } \mathcal{T},\ \text{all } S_m \in \mathrm{CLASS} \right\}.$$

The order of the processors could also be considered to be part of the algorithm; however, the concept of processor classes is more profitable to a uniform representation.

One of the first and famous heuristics is the LPT algorithm [5,6] which allocates the largest, not yet assigned task to that processor which completes its execution earliest. This algorithm satisfies

$$R_m(\mathrm{LPT}, \mathrm{HOM}) = 4/3 - 1/(3m) \quad [6]$$

and

$$R_m(\mathrm{LPT}, \mathrm{UNIF}) = R_m(\mathrm{LPT}, \mathrm{INC}) = R_m(\mathrm{LPT}, \mathrm{DEC}) \leq 2 - 2/(m+1) \quad [5].$$

It is easily seen that the performance ratio of the LPT is independent of the order

of the processors. Determining the exact ratio in this case is still an open question, but it is known that

$$\lim_{m\to\infty} R_m(\text{LPT}, \text{UNIF}) \geq 3/2 \qquad [5].$$

Moreover it was shown in the same paper that

$$R_2(\text{LPT}, \text{INC1}) = R_2(\text{LPT}, \text{DEC1}) = (\sqrt{17}+1)/4$$

and

$$4/3 \leq R_m(\text{LPT}, \text{INC1}) = R_m(\text{LPT}, \text{DEC1}) \leq 3/2 - 1/2m \quad \text{for } m \geq 3.$$

As an improvement to the LPT the so called multifit algorithm MF was presented in [2]. It is based on the bin-packing algorithm FFD (First Fit Decreasing) [4] and was originally developed for homogeneous multiprocessors with worst case bound

$$R_m(\text{MF}, \text{HOM}) \leq 1.22 \qquad [2].$$

A generalization of this method to arbitrary uniform machines leads to a further improvement of the LPT. It was shown that

$$R_3(\text{MF}, \text{INC}) = (\sqrt{17}+1)/4 \qquad \text{[Kunde, unpublished]},$$

$$(\sqrt{17}+1)/4 \leq R_m(\text{MF}, \text{INC}) \leq 3/2 - 1/(2m) \quad \text{for } m = 4, 5 \qquad [7,8,13],$$

$$1.341 \leq R_m(\text{MF}, \text{INC}) \leq 7/5 \quad \text{for } m \geq 6 \qquad [3].$$

Moreover for the case where all but one processor have the same speed we get the exact bounds

$$R_2(\text{MF}, \text{INC1}) = \sqrt{6}/2 \qquad [7]$$

and

$$R_m(\text{MF}, \text{INC1}) = (\sqrt{17}+1)/4 \quad \text{for } m \geq 3 \qquad [7,9].$$

Hence for all $m \geq 2$ we obtain

$$R_m(\text{MF}, \text{INC1}) < R_m(\text{LPT}, \text{INC1}) \quad \text{and} \quad R_m(\text{MF}, \text{INC}) < R_m(\text{LPT}, \text{INC}).$$

In this paper we show that in contrast to the LPT the performance ratio of the MF strongly depends on the order of processors. If the processors are ordered by decreasing speeds, then it can be shown that

$$R_2(\text{MF}, \text{DEC1}) = (\sqrt{17}+1)/4$$

and

$$4/3 \leq \frac{1}{2m} + \frac{1}{2m}(8m^2 - 8m + 1)^{1/2} \leq R_m(\text{MF}, \text{DEC1}) \leq \sqrt{2} \quad \text{for } m \geq 3.$$

Hence the following inequalities hold for all $m \geq 2$

$$R_m(\text{MF}, \text{INC1}) < R_m(\text{MF}, \text{DEC1}) \leq R_m(\text{MF}, \text{UNIF1}).$$

In the more general case where the processors might have different speeds we prove that $\lim_{m\to\infty} R_m(\text{MF}, \text{DEC}) \geq 3/2$ and

$$R_m(\mathrm{MF}, \mathrm{INC}) < R_m(\mathrm{MF}, \mathrm{DEC}) \leq R_m(\mathrm{MF}, \mathrm{UNIF}) \quad \text{for } m \geq 2.$$

As an upper bound we derive

$$R_m(\mathrm{MF}, \mathrm{DEC}) \leq R_m(\mathrm{MF}, \mathrm{UNIF}) \leq 2 - 1/m.$$

The results of this and earlier papers show that with respect to the worst case ratio the multifit algorithm MF is better than the LPT, an increasing speed order provided. It is an open question whether or not the LPT is better than the MF in the case of a decreasing speed order.

2. First fit decreasing scheduling

The difference between list scheduling [1,6] and scheduling using packing is that, apart from a list, a capacity bound is needed [2]. Thus, the problem of finding a good a priori bound C emerges.

Let $\mathcal{T}$, m, S_m and a capacity bound C be given. Throughout the paper we assume the tasks to be in a decreasing order, that is, $\mu(T_1) \geq \mu(T_2) \geq \cdots \geq \mu(T_n)$. Informally, the MF algorithm tries to put the largest, not yet assigned task on the processor with the smallest index such that the capacity bound is not violated. In the case $S_m \in \mathrm{INC}$ this means that one attempts to fill the slowest processors first.

The following boolean function FFD($\mathcal{T}, C, S_m$) gives an exact description of first fit decreasing scheduling and tells us whether it is possible to assign all tasks within bound C in this manner.

Boolean function FFD($\mathcal{T}, C, S_m$)
begin FFD := *true*; $i := 1$; $j := 1$;
  for k **from** 1 **to** m **do begin** $P_k := \emptyset$; $C_i := s_i \cdot C$ **end**;
  repeat if $\mu(P_i) + \mu(T_j) \leq C_i$
    then begin $P_i := P_i \cup \{T_j\}$; $j := j + 1$; $i := 1$ **end**
    else $i := i + 1$;
  until ($j > n$ **or** $i > m$);
  if $i > m$ **then** FFD := *false*
end

Note that FFD constructs a schedule $\Pi_{\mathrm{FFD}} = (P_1, \ldots, P_m)$ with $\omega(\Pi_{\mathrm{FFD}}) \leq C$ if and only if FFD($\mathcal{T}, C, S_m$) = *true*.

The following lemma will give a first estimation of the magnitude of C such that a successful assignment of the tasks is guaranteed. Let $S_m = (s_1, \ldots, s_m)$ be any speed tuple of UNIF and $\mathcal{T} = \{T_1, \ldots, T_n\}$, $\mu(T_1) \geq \cdots \geq \mu(T_n)$. Assume that we have rearranged the speeds of S_m such that $s_{j_1} \geq s_{j_2} \geq \cdots \geq s_{j_m}$. Let $X_k = \sum_{i=1}^{k} \mu(T_i)$, $k = 1, \ldots, n$, denote the sum of the k largest execution times and $Y_h = \sum_{i=1}^{h} s_i$, $h = 1, \ldots, m$, be the sum of the h largest speeds. Then a lower and an upper bound can be defined in the following way:

$$\mathrm{CL}(\mathcal{T}, S_m) = \max\left\{X_n/Y_m, \max_{1 \le k \le m} X_k/Y_k\right\}$$

and

$$\mathrm{CU}(\mathcal{T}, S_m) = (2 - 1/m) \cdot \mathrm{CL}(\mathcal{T}, S_m).$$

Lemma 2.1. (a) *For all* $C < \mathrm{CL}(\mathcal{T}, S_m)$, $\mathrm{FFD}(\mathcal{T}, C, S_m) = \mathit{false}$.
(b) *For all* $C \ge \mathrm{CU}(\mathcal{T}, S_m)$, $\mathrm{FFD}(\mathcal{T}, C, S_m) = \mathit{true}$.

Proof. Assume there is a $C < \mathrm{CL}(\mathcal{T}, S_m)$ with $\mathrm{FFD}(\mathcal{T}, C, S_m) = \mathit{true}$. Then $\omega_{\mathrm{OP}}(\mathcal{T}, S_m) \le C < \mathrm{CL}(\mathcal{T}, S_m)$ contradicts a well-known theorem [12].

Suppose that there is a set of tasks $\mathcal{T} = \{T_1, \ldots, T_n\}$, a $C \ge \mathrm{CU}(\mathcal{T}, S_m)$ and an $S_m = (s_1, \ldots, s_m)$ such that $\mathrm{FFD}(\mathcal{T}, C, S_m) = \mathit{false}$. Let $m \ge 2$ and n be minimal with this property. If $n < m$, then $\mathrm{CL}(\mathcal{T}, S_m) > X_n/Y_m$ and FFD cannot construct a valid schedule on the n fastest processors, contradicting the fact that m was minimal. In the case $n \ge m$ let $(P_1, \ldots, P_m)$ denote the schedule of $\mathcal{T} - \{T_n\}$ constructed by $\mathrm{FFD}(\{T_1, \ldots, T_{n-1}\}, C, S_m)$. This means, $\sum_{i=1}^{m} \mu(P_i) = \sum_{i=1}^{n-1} \mu(T_i)$. From $\mathrm{FFD}(\mathcal{T}, C, S_m) = \mathit{false}$ follows

$$\mu(P_i) + \mu(T_n) > C \cdot s_i \quad \text{for } i = 1, \ldots, m. \tag{1}$$

From the definition of $\mathrm{CL}(\mathcal{T}, S_m)$ and $\mathrm{CU}(\mathcal{T}, S_m)$ we immediately get

$$C \ge \mathrm{CU}(\mathcal{T}, S_m) \ge (X_n + (1 - 1/m) X_m)/Y_m. \tag{2}$$

Thus

$$\sum_{i=1}^{n} \mu(T_i) = \sum_{i=1}^{m} \mu(P_i) + \mu(T_n) > \sum_{i=1}^{m} s_i \cdot C - (m-1)\mu(T_n)$$
$$\ge Y_m \cdot \mathrm{CU}(\mathcal{T}, S_m) - (m-1)\mu(T_n) \ge X_n = \sum_{i=1}^{n} \mu(T_i),$$

a contradiction. Therefore (b) must hold. □

Normally the bound $\mathrm{CU}(\mathcal{T}, S_m)$ can be improved easily. We want to give bounds which are valid for all task sets and for all processor systems of a given class. The best expansion factor of a processor class, denoted by CLASS, is defined by

$$r_m(\mathrm{CLASS}) = \sup\{r \mid \exists \mathcal{T}, \exists S_m \in \mathrm{CLASS}, \mathrm{FFD}(\mathcal{T}, r\omega_{\mathrm{OP}}(\mathcal{T}, S_m), S_m) = \mathit{false}\}.$$

The proof of the following technical lemma is similar to that of lemma 2.1 in [2] and is therefore omitted.

Lemma 2.2.

$$\forall \mathcal{T}, \forall S_m \in \mathrm{CLASS}, \forall r \ge r_m(\mathrm{CLASS}) \quad \mathrm{FFD}(\mathcal{T}, r\omega_{\mathrm{OP}}(\mathcal{T}, S_m), S_m) = \mathit{true}.$$

The pure multifit algorithm MF for CLASS works as follows:

Let $C = r_m(\mathrm{CLASS}) \cdot \omega_{\mathrm{OP}}(\mathcal{T}, S_m)$, apply FFD, then $\mathrm{FFD}(\mathcal{T}, C, S_m) = \mathit{true}$. Define $\Pi_{\mathrm{MF}}(\mathcal{T}, S_m) = \Pi_{\mathrm{FFD}}(\mathcal{T}, S_m)$.

It is easily seen that $R_m(\mathrm{MF}, \mathrm{CLASS}) = r_m(\mathrm{CLASS})$. In the following we will give exact values or estimations for some r_m. Determining $\omega_{\mathrm{OP}}(\mathcal{T}, S_m)$ is well known to be NP-hard. Nevertheless we will show that it is possible to construct efficient versions $\mathrm{MF}(k)$ of the MF with worst case ratio very near to r_m.

```
Procedure MF(k)(𝒯, S_m)
begin CL := CL(𝒯, S_m); CU := CU(𝒯, S_m);
      for i from 1 to k do
      begin C := (CL + CU)/2;
            if FFD(𝒯, C, S_m) then CU := C else CL := C
      end
end
```

This procedure $\mathrm{MF}(k)$ is the same one as given in [2] for homogeneous multiprocessors. From Lemma 2.1 we obtain the lower and the upper bound needed for the binary search. The final value CU gives the smallest capacity bound C found for which $\mathrm{FFD}(\mathcal{T}, C, S_m) = \mathit{true}$.

The complexity analysis of $\mathrm{MF}(k)$ as given in [2] shows that including the initial sorting of the tasks $O(n \log n + knm)$ steps are needed. Hence the complexity of the $\mathrm{MF}(k)$ is comparable to that of the LPT where $O(n \log n + nm)$ steps are necessary.

The first theorem demonstrates that even for small k, $R_m(\mathrm{MF}(k), \mathrm{CLASS})$ very closely approaches $R_m(\mathrm{MF}, \mathrm{CLASS})$.

Theorem 1 ([2]). *For all $m \geq 2$ and for all $k \geq 0$*

$$R_m(\mathrm{MF}(k), \mathrm{CLASS}) \leq R_m(\mathrm{MF}, \mathrm{CLASS}) + 2^{-k}.$$

Thus it is possible to construct efficient versions $\mathrm{MF}(k)$ of the MF with a performance ratio very close to $R_m(\mathrm{MF}, \mathrm{CLASS})$.

Before discussing further theorems a short example may illustrate how the multifit algorithm MF works and how the finishing times of the schedules are influenced by the processor configuration.

Example 2.1.

$$\mathcal{T} = \{T_1, \ldots, T_7\}, \qquad \begin{aligned} \mu(T_i) &= 10 \quad \text{for } i = 1, 2, 3, 4 \\ \mu(T_j) &= 7 \quad \text{for } j = 5, 6, \\ \mu(T_7) &= 6. \end{aligned}$$

Let $m = 5$; 4 processors with speed 1 and one processor with speed 2.

S_5	$\omega_{\mathrm{MF}(k)}(\mathcal{I}, S_5)$, $k \geq 4$
(1, 1, 1, 1, 2)	10
(1, 1, 1, 2, 1)	11.5
(1, 1, 2, 1, 1)	13
(1, 2, 1, 1, 1)	13
(2, 1, 1, 1, 1)	13

One immediately checks that

$$\omega_{\mathrm{OP}}(\mathcal{I}, S_5) = 10 \quad \text{and} \quad \omega_{\mathrm{LPT}}(\mathcal{I}, S_5) = 11.5.$$

Theorem 2.

(a) $R_m(\mathrm{MF}, \mathrm{DEC}) \leq R_m(\mathrm{MF}, \mathrm{UNIF}) \leq 2 - 1/m$ *for* $m \geq 3$.

(b) $\lim_{m\to\infty} R_m(\mathrm{MF}, \mathrm{DEC}) \geq 3/2$.

Proof. (a) With the help of $\omega_{\mathrm{OP}}(\mathcal{I}, S_m) \geq \mathrm{CL}$ [12] we get for all $\mathcal{I}$, all $r \geq 2 - 1/m$ and all $S_m \in \mathrm{UNIF}$

$$r \cdot \omega_{\mathrm{OP}}(\mathcal{I}, S_m) \geq (2 - 1/m)\mathrm{CL} \geq \mathrm{CU}$$

and thus, by Lemma 2.1, $\mathrm{FFD}(\mathcal{I}, r \cdot \omega_{\mathrm{OP}}(\mathcal{I}, S_m), S_m) = \mathit{true}$. That is,

$$R_m(\mathrm{MF}, \mathrm{UNIF}) = r_m(\mathrm{UNIF}) \leq 2 - 1/m.$$

(b) In [5] there is given a general example for the LPT with $\lim_{m\to\infty} R_m(\mathrm{LPT}, \mathrm{UNIF}) \geq 3/2$. The same example implies $\lim_{m\to\infty} R_m(\mathrm{MF}, \mathrm{DEC}) \geq 3/2$ as demonstrated in [3,13]. □

It has been shown that

$$R_3(\mathrm{MF}, \mathrm{INC}) = (\sqrt{17}+1)/4 \qquad \text{[Kunde, unpublished]},$$

$$(\sqrt{17}+1)/4 \leq R_m(\mathrm{MF}, \mathrm{INC}) \leq 3/2 - 1/(2m) \quad \text{for } m = 4, 5 \qquad [7,8,13],$$

$$1.341 \leq R_m(\mathrm{MF}, \mathrm{INC}) \leq 7/5 \quad \text{for } m \geq 6 \qquad [3].$$

These bounds together with the above mentioned general example [3,5,13] demonstrate that $R_m(\mathrm{MF}, \mathrm{INC}) < R_m(\mathrm{MF}, \mathrm{DEC})$ for all $m \geq 3$. In all these cases (except $R_3(\mathrm{MF}, \mathrm{INC})$) the exact worst case ratio for the multifit algorithm MF is as unknown as for the LPT.

Theorem 3.

(a) $R_2(\mathrm{MF}, \mathrm{DEC1}) = R_2(\mathrm{MF}, \mathrm{UNIF}) = (\sqrt{17}+1)/4$.

(b) $(1 + \sqrt{8m(m-1)+1})/(2m) \leq R_m(\mathrm{MF}, \mathrm{DEC1}) \leq \sqrt{2}$ *for* $m \geq 3$.

The proof of the theorem will be given in the next section.

The real number $(\sqrt{17}+1)/4$ seems to play an important role in this special field of scheduling. It is known that

$$R_2(\mathrm{LPT}, \mathrm{UNIF}) = (\sqrt{17}+1)/4 \qquad [5]$$

and

$$R_m(\mathrm{MF}, \mathrm{INC1}) = (\sqrt{17}+1)/4 \quad \text{for all } m \geq 3 \qquad [7,9].$$

Moreover $R_3(\mathrm{MF}, \mathrm{INC})$ describes the same value. With Theorem 3(a) and $R_2(\mathrm{MF}, \mathrm{INC}) = \sqrt{6}/2$ [7] the worst case analysis of the MF and the LPT for two uniform processors is totally done:

$$R_2(\mathrm{MF}, \mathrm{INC}) < R_2(\mathrm{MF}, \mathrm{DEC}) = R_2(\mathrm{MF}, \mathrm{UNIF}) = R_2(\mathrm{LPT}, \mathrm{UNIF}).$$

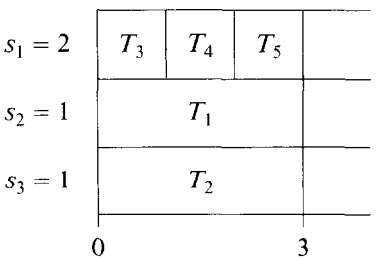
3. Proof of Theorem 3

For the proof of Theorem 3 we first give a general example establishing the lower bounds. The underlying ideas may be understood by the following simple example.

Example 3.1. $m = 3$, $S_3 = (2, 1, 1)$, $\mathcal{T} = \{T_1, \ldots, T_5\}$ with

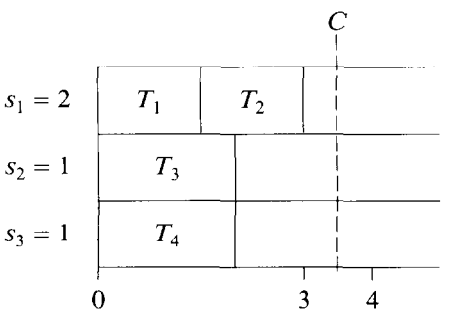
$$\mu(T_1) = \mu(T_2) = 3, \qquad \mu(T_3) = \mu(T_4) = \mu(T_5) = 2.$$

Processor



optimum schedule
$\omega_{\mathrm{OP}}(\mathcal{T}, S_3) = 3$.

Processor



$\mathrm{FFD}(\mathcal{T}, C, S_3) = \mathit{false} \quad \forall C < 4,$
$\mathrm{FFD}(\mathcal{T}, 4, S_3) = \mathit{true}.$

Example 3.2. This example is also mentioned in [9].
$m \geq 2$, $S_m = (s, 1, 1, \ldots, 1)$, $s_1 = s$, $\mathcal{T} = \{T_1, \ldots, T_{2m-1}\}$. Let

$$a_m = \frac{1}{2m} + \frac{1}{2m}(8m^2 - 8m + 1)^{1/2},$$

$$\mu(T_i) = \begin{cases} 1, & i = 1, \ldots, m-1, \\ a_m/2, & i = m, \ldots, 2m-1, \end{cases} \quad \text{and}$$

$$s = \frac{m}{2} \cdot a_m = (1 + \sqrt{8m(m-1)+1})/4.$$

It is easily seen that $\Pi^+ = (P_1^+, \ldots, P_m^+)$ with $P_1^+ = \{T_m, \ldots, T_{2m-1}\}$ and $P_{i+1}^+ = \{T_i\}$, $i = 1, \ldots, m-1$, is an optimum schedule with $\omega_{\mathrm{OP}}(\mathcal{T}, S_m) = 1$. Moreover, $a_2 = (\sqrt{17}+1)/4$ and $a_m \leq \sqrt{2}$ for $m \geq 3$. Hence

$$\mu(T_1) = \cdots = \mu(T_{m-1}) > \mu(T_m) = \cdots = \mu(T_{2m-1}).$$

Let C_m be a capacity bound such that $a_m - 1/2m \leq C_m < a_m$. Then we state

(1) $\quad \mathrm{FFD}(\mathcal{T}, C_m, S_m) = \mathit{false}$ for all $m \geq 2$.

To prove (1) we first observe that

$$\left(\sum_{i=1}^{m-1} \mu(T_i)\right) \Big/ s = 4(m-1)/(1 + \sqrt{8m(m-1)+1})$$

$$= \frac{4(m-1)(\sqrt{8m(m-1)+1}-1)}{8m(m-1)} = a_m - \frac{1}{m} < C_m.$$

That means, that FFD assigns every T_i, $i = 1, \ldots, m-1$, to the first processor with speed s. But trying to allocate any of the smaller tasks T_j, $m \leq j \leq 2m-1$, must fail, because

$$\left(\sum_{i=1}^{m-1} \mu(T_i) + \mu(T_j)\right) \Big/ s = a_m - \frac{1}{m} + \frac{a_m}{2s} = a_m - \frac{1}{m} + \frac{1}{m} = a_m > C_m.$$

On the other hand, if none of the m smaller tasks can be assigned to the first processor, then FFD tries to put two of these tasks on one of the $m-1$ slower processors. Since for arbitrary i and j, $m \leq i, j \leq 2m-1$, $\mu(T_i) + \mu(T_j) = a_m > C_m$, this attempt again fails. Hence for every $m \geq 2$, $R_m(\mathrm{MF}, \mathrm{DEC1}) \geq a_m$, as stated in Theorem 3.

We are now going to prove the bounds $(\sqrt{17}+1)/4$ respectively $\sqrt{2}$ by contradiction. For technical reasons it is useful to have standardized forms of counterexamples. Throughout the whole proof we will use the following abbreviations

$$s = s_1 \geq 1, \qquad q = \omega_{\mathrm{OP}}(\mathcal{T}, S_m) \quad \text{and} \quad p = rq, \quad \text{where } r > 1.$$

Definition 1. p, q as above. $(\mathcal{T}, m, S_m)$ is called a *p/q-counterexample* iff

(1) $\mathrm{FFD}(\mathcal{T}, p, S_m) = \mathit{false}$,

(2) $\forall \mathcal{T}'$ with $|\mathcal{T}'| < |\mathcal{T}|, \forall S_m'$ with $\omega_{\mathrm{OP}}(\mathcal{T}', S_m') \leq q$: $\mathrm{FFD}(\mathcal{T}', p, S_m') = \mathit{true}$,

(3) $\forall m'$, $m' < m$, $\forall \mathcal{T}'$, $\forall S_{m'}$ with $\omega_{\mathrm{OP}}(\mathcal{T}', S_{m'}) \leq q$: $\mathrm{FFD}(\mathcal{T}', p, S_{m'}) = \mathit{true}$.

That means, a p/q-counterexample is minimal in the number of tasks (condition (2)) and minimal in the number of processors (condition (3)). As before let $\mathcal{T} = \{T_1, \ldots, T_n\}$ and $\mu(T_1) \geq \cdots \geq \mu(T_n)$. If $(\mathcal{T}, m, S_m)$ is a p/q-counterexample, then from condition (2) easily follows that $\mathrm{FFD}(\{T_1, \ldots, T_{n-1}\}, p, S_m) = true$. Let $\Pi = (P_1, \ldots, P_m)$ be the partition of $\{T_1, \ldots, T_{n-1}\}$ generated by FFD with bound p and let $\Pi^+ = (P_1^+, \ldots, P_m^+)$ be an optimal partition of $\mathcal{T}$. These notations will be used for the rest of this paper.

Definition 2. Let X and Y be subsets of $\mathcal{T}$. *X dominates Y* iff there is a 1–1 mapping $f: Y \to X$, such that $\mu(T) \leq \mu(f(T))$ for all T of Y.

The following two lemmata are proved in [8,13].

Lemma 3.1. *Let $(\mathcal{T}, S_m)$ be a p/q-counterexample. Then a set P_i, $1 \leq i \leq m$, cannot dominate any set P_j^+, $1 \leq j \leq m$, with $s_i \leq s_j$.*

Lemma 3.2. *Let $(\mathcal{T}, m, S_m)$ be a p/q-counterexample. Then*

$$\mu(T_n) > (p-q) \sum_{i=1}^{m} s_i/(m-1).$$

These two lemmata enable us to prove the first part of Theorem 3, where we consider $m = 2$ processors. In this case p and q are assumed to satisfy $p/q \geq (\sqrt{17}+1)/4$. If the number of tasks is $n \geq 4$, then Lemma 3.2 implies

$$(1+s)q \geq \mu(P_1^+) + \mu(P_2^+) \geq 4\mu(T_n) > 4(1+s)(p-q)$$

and thus $q > 4(p-q)$, which immediately yields the contradiction

$$(\sqrt{17}+1)/4 \leq p/q < 5/4.$$

Since the case $n \leq 2$ yields $P_2^+ = P_2 = \emptyset$, we only have to consider the case $n = 3$. Then the FFD rule generates $P_1 = \{T_1\}$ and $P_2 = \{T_2\}$. From $\mu(T_2) + \mu(T_3) > p > q$ we know that at most one task is contained in P_2^+. From Lemma 3.1 we derive that T_1 cannot be in P_1^+, and thus $P_2^+ = \{T_1\}$, and consequently $P_1^+ = \{T_2, T_3\}$. This whole situation implies the following inequalities

(1) $sq \geq \mu(T_2) + \mu(T_3) > p$,

(2) $p \geq \mu(T_1) > sp - \mu(T_3)$.

From (1), (2) and Lemma 3.2 we derive

$$(1+s)q > sp + \mu(T_2) > sp + (1+s)(p-q)$$

and thus

$$2q > p + 2s(p-q) \quad \text{or} \quad 2 > p/q + 2s(p/q - 1).$$

From (1) we get $s > p/q$ and therefore

$$\begin{aligned} 2 &> p/q + 2p/q(p/q-1) \\ &= p/q(2p/q-1) \geq (\sqrt{17}+1)/4((\sqrt{17}+1)/2-1) \\ &= 9/4 + \sqrt{17}/4 - \sqrt{17}/4 - 1/4 = 2, \end{aligned}$$

which is impossible.

For the rest of the proof we therefore may assume that $m \geq 3$ and $p/q \geq \sqrt{2}$.

Before making the final conclusions for proving the second part of Theorem 3 we state our last lemma:

Lemma 3.3. *Let $(\mathcal{T}, m, S_m)$ be a p/q-counterexample with $S_m \in$ DEC1 and $p/q \geq \sqrt{2}$. Then $P_i^+ \cap P_1 \neq \emptyset$ for all $i = 2, \ldots, m$.*

Proof. First observe that every P_i^+, $2 \leq i \leq m$, contains at most two tasks. Otherwise we conclude with the help of Lemma 3.2, $q \geq 3\mu(T_n) > 3(p-q)$, thus $4q > 3p$, giving us the contradiction $\sqrt{2} \leq p/q < 4/3$.

Now assume that there is P_i^+, $2 \leq i \leq m$, with $P_i^+ \cap P_1 = \emptyset$. We will show that in this case a set P_j, $j \geq 2$, exists which dominates P_i^+.

If P_i^+ contains only one task, then either $P_i^+ = \{T_n\}$, that is, every P_j, $2 \leq j \leq m$, dominates P_i^+, or $P_i^+ \neq \{T_n\}$. In the second case the assumption implies that P_i^+ must be subset of a P_j, $2 \leq j \leq m$. That is, P_j dominates P_i^+.

If $P_i^+ = \{T_x, T_y\}$ with $x < y$, that is, $\mu(T_x) \geq \mu(T_y)$, then first assume that $T_y = T_n$ and $T_x \notin P_1$. Hence T_x must be an element of a P_j, $2 \leq j \leq m$. Since $\mu(T_x) + \mu(T_n) = \mu(P_i^+) \leq q < p$ and $T_n \notin P_j$, another task T with $\mu(T) \geq \mu(T_n)$ must be contained in P_j. Therefore P_j dominates P_i^+.

If $T_y \neq T_n$, according to our assumption both tasks have been assigned to slow processors by the FFD, that is, there are indices j and k, $2 \leq j, k \leq m$, with $T_x \in P_j$ and $T_y \in P_k$. If $j = k$, P_j dominates P_i^+. Hence two cases are left:

(a) $2 \leq j < k \leq m$. As FFD did not assign T_y to P_j and $\mu(T_x) + \mu(T_y) \leq q < p$, there must have been placed a task $T \neq T_x$ into P_j with $\mu(T) \geq \mu(T_y)$. Consequently P_j dominates P_i^+.

(b) $2 \leq k < j \leq m$. Although $\mu(T_x) \leq q < p$, T_x was not put into P_k. Therefore another task T_h with $h < x$, that is, $\mu(T_h) \geq \mu(T_x)$, must have been assigned to P_k, which therefore dominates P_i^+.

In all cases the assumption leads to dominating sets contradicting Lemma 3.1. □

Now we are able to conclude the proof. For the p/q-counterexample $(\mathcal{T}, m, S_m)$ with $p/q \geq \sqrt{2}$ two cases are distinguished:

(a) $q \geq 2\mu(T_n)$. First observe that $|P_1| \geq m - 1$. This is an immediate consequence of the last lemma. In view of Lemma 3.2 we derive

$$ps \geq (m-1)\mu(T_n) > (m-1+s)(p-q)$$

and thus $s>(m-1)(p/q-1)$. Hence

$$q \geq 2\mu(T_n) > 2(p-q)(m-1+s)/(m-1)$$
$$> 2(p-q)\frac{p}{q} = \frac{2p^2}{q} - 2p.$$

The last inequality and $p/q \geq \sqrt{2}$ yield $1 > 2p/q(p/q-1) \geq 4-2\sqrt{2}$, a contradiction.

(b) $q<2\mu(T_n)$. In this case every P_i^+, $2 \leq i \leq m$, contains exactly one task and by Lemma 3.3 it follows that $P_i^+ \subseteq P_1$. Since $(\mathcal{I}, m, S_m)$ is a p/q-counterexample, we get

$$ps < \mu(P_1) + \mu(T_n) \leq \sum_{i=2}^{m} \mu(P_1 \cap P_i^+) + \mu(P_1 \cap P_1^+) + \mu(T_n)$$
$$\leq (m-1)q + X, \quad \text{where } X = \mu(P_1 \cap P_1^+) + \mu(T_n).$$

That is, we can state the following upper bound for s

$$s < (m-1)\frac{q}{p} + \frac{X}{p}. \tag{1}$$

From $P_i^+ \subseteq P_1$ for $i=2,\ldots,m$ we conclude that $P_j \cap P_i^+ = \emptyset$ for $i=2,\ldots,m$ and $j=2,\ldots,m$. Hence $P_j \subseteq P_1^+$ for $j=2,\ldots,m$. The task T_n obviously is an element of P_1^+ and thus

$$qs \geq \mu(P_1^+) = \sum_{i=2}^{m} \mu(P_i) + \mu(P_1 \cap P_1^+) + \mu(T_n).$$

From the trivial inequalities $\mu(P_i) \geq \mu(T_n)$ and $\mu(P_i) > p - \mu(T_n)$ for all $i=2,\ldots,m$ we obtain $\mu(P_i) > p/2$ and thus

$$qs > (m-1)p/2 + X \text{ or } \frac{p}{q} < \frac{2s}{m-1} - \frac{2X}{q(m-1)}.$$

Combining this formula with (1) we get

$$\frac{p}{q} < 2\frac{q}{p} + \frac{2X}{m-1}\left(\frac{1}{p} - \frac{1}{q}\right).$$

Since $p>q$, we conclude $(p/q)^2<2$, a final contradiction.

4. Conclusions

In this paper we have demonstrated that in the case of uniform processors the performance behaviour of the multifit algorithm depends on the processor configuration of which LPT is independent. Especially we gave bounds for the worst case ratio of the MF for the processor classes DEC1, UNIF1 and DEC.

There are still a lot of open questions concerning the exact ratios for the different classes. Another open problem is for example whether or not $R_m(\text{MF}, \text{DEC1}) = R_m(\text{MF}, \text{UNIF1})$ for every $m \geq 3$.

We were informed by one of the referees that in yet unpublished results D.K. Friesen has obtained the following bounds:

$$13/11 \leq R_m(\mathrm{MF}, \mathrm{HOM}) \leq 6/5,$$

$$1.52 \leq \lim_{m \to \infty} R_m(\mathrm{LPT}, \mathrm{UNIF}) \leq 1.67.$$

Acknowledgements

The authors wish to thank the referees for their helpful comments.

References

[1] E.G. Coffman, Jr. (ed.), Computer and Job/Shop Scheduling Theory (Wiley, New York, 1976).
[2] 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.
[3] D.K. Friesen and M.A. Langston, Bounds for multifit scheduling on uniform processors, SIAM J. Comput. 12 (1983) 60–70.
[4] M.R. Garey and D.S. Johnson, Approximation algorithms for bin packing problems: A Survey, in: G. Ausiello and M. Lucertini, eds., Analysis and Design of Algorithms in Combinatorial Optimization (Springer, New York, 1981) 147–172.
[5] T. Gonzalez, O.H. Ibarra and S. Sahni, Bounds for LPT schedules on uniform processors, SIAM J. Comput. 6 (1977) 155–166.
[6] R.L. Graham, Bounds on multiprocessing timing anomalies, SIAM J. Appl. Math. 17 (1969) 416–429.
[7] M. Kunde, Bounds for multifit scheduling algorithms on uniform multiprocessor systems, Bericht 8203, Institut für Informatik und Praktische Mathematik, Kiel, 1982.
[8] M. Kunde, A multifit algorithm for uniform multiprocessor scheduling, in: Theoretical Computer Science, Proceedings, 1983, Lecture Notes in Computer Science 145 (Springer, New York, 1982) 175–185.
[9] M.A. Langston and J.M. Liu, On a special case of uniform processor scheduling, TR CS-83-107, Washington State University, Pullman, 1983.
[10] E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan, Recent developments in deterministic sequencing and scheduling: a survey, in: M.A.H. Dempster et al., ed., Deterministic and Stochastic Scheduling (D. Reidel, Dordrecht, 1982) 35–73.
[11] J.W.S. Liu and C.L. Liu, Bounds on scheduling algorithms for heterogeneous computer systems, Information Processing 74 (North-Holland, Amsterdam, 1974) 349–353.
[12] J.W.S. Liu and A. Yang, Optimal scheduling of independent tasks on heterogeneous computing systems, ACM National Conference 1974, 38–45.
[13] H. Steppat, Packungsalgorithmen für die Ablaufplanung in uniformen Mehrprozessorsystemen, Diplomarbeit, Kiel 1983.
[14] J.D. Ullman, NP-complete scheduling problems, J. Comput. System Sci. 10 (1975) 384–393.
[15] J.D. Ullman, Complexity of sequencing problems, in: [1], 139–164.