Discrete Applied Mathematics 20 (1988) 233–242
North-Holland

MULTIPROCESSOR SCHEDULING: COMBINING LPT AND MULTIFIT*

Chung-Yee LEE and J. David MASSEY
Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL 32611, USA

Received 13 March 1987

We consider the problem of scheduling a set of n independent jobs on m identical machines with the objective of minimizing the total finishing time. We combine the two most popular algorithms, LPT and MULTIFIT, to form a new one. MULTIFIT is well known to have better error bound than LPT with the price paid in running time. Although MULTIFIT provides a better error bound, in many cases LPT still can yield better results. This motivates the development of this new combined algorithm, which uses the result of LPT as the incumbent and then applies MULTIFIT with fewer iterations. The performance of this combined algorithm is better than that of LPT because it uses LPT as an incumbent. The error bound of this combined algorithm is never worse than that of MULTIFIT. For example, the error bound of implementing this combined algorithm to the two-processor problem is $\frac{1}{9}$, compared to the error bound of $\frac{1}{7}$ in MULTIFIT. Empirical results of the comparison for schedules obtained by the combined algorithm, MULTIFIT and LPT are also provided.

Keywords. Multiprocessor scheduling, bin packing, worst-case analysis, heuristic method.

1. Introduction

The problem of scheduling a set of n independent jobs to m identical, parallel processors with the objective of minimizing the total finishing time is one of the most well-studied problems in scheduling theory (see, e.g., [1–4, 6–9]. In this problem, we are given a set of n independent jobs with processing time p_i $(i=1,\ldots,n)$ and m identical machines. Our purpose is to assign each job to be processed in one machine so as to minimize the makespan, i.e., the total finishing time. Let N denote the set of n jobs and let $M^*(N,m)$ be the optimal makespan of the problem. Without ambiguity, we will use M^* to denote $M^*(N,m)$.

It is well known that this problem is NP-complete [5]. It is unlikely to have an efficient polynomial algorithm to find the optimal solution. Hence several heuristics have been proposed. Among them, the following two methods are the most widely adopted.

* This research was supported in part by the NSF Grant DMC-8504786.

(i) Longest Processing Time (LPT) method [7]. LPT algorithm first sorts the data such that $p_1 \geq p_2 \geq \cdots \geq p_n$ and then assigns the job in succession to the minimally loaded machine. Graham [7] has shown that the LPT algorithm will always find a schedule within $(\frac{4}{3} - 1/3m)M^*$. The time complexity of LPT is $O(n \log n + n \log m)$.

(ii) MULTIFIT method [1]. This method utilizes a bin-packing method in conjunction with a bisection search over bin capacity (where capacity is measured in time) to try to find the minimum capacity such that all n jobs will fit into the m bins. For each fixed bin capacity, the First Fit Decreasing (FFD) method is used to fit the jobs to the bin. Assume that the jobs have been sorted such that $p_1 \geq p_2 \geq \cdots \geq p_n$. The FFD method assigns the job in succession to the lowest indexed machine which can complete the job within the capacity.

Let

$$A = \sum_{i=1}^{n} p_i/m \tag{1}$$

and let $C_{\text{upper}} = \max(p_1, 2A)$ and $C_{\text{lower}} = \max(p_1, A)$. We can describe the MULTIFIT algorithm as follows:

MULTIFIT Algorithm

Step 0. Set $I = 0$.

Step 1. Set $C = \frac{1}{2}(C_{\text{lower}} + C_{\text{upper}})$. Set $I = I + 1$.

Step 2. Apply FFD with capacity C.

Step 3. If we can assign all the jobs in N into m machines, then set $C_{\text{upper}} = C$ and go to Step 4. Otherwise set $C_{\text{lower}} = C$ and go to Step 4.

Step 4. If $I = k$ then stop. Otherwise go to Step 1.

Remarks. (1) The final C_{upper} is the minimal capacity that can fit all jobs into m machines and is the makespan obtained by MULTIFIT.

(2) k is predetermined and is the number of iterations of FFD. Usually, we set $k = 7$ for the reason stated below.

(3) For each iteration in Step 2, the time complexity is $O(n \log m)$. There are k iterations. Hence the total time complexity is $O(kn \log m)$. If we include the sorting of the data, which needs $O(n \log n)$ operations, then the total time complexity of MULTIFIT algorithm is $O(n \log n + kn \log m)$.

Coffman et al. [1] have shown that implementing the MULTIFIT algorithm to the multiprocessor scheduling problem will always get a schedule within $(1.22 + (\frac{1}{2})^k)M^*$. Note that $(\frac{1}{2})^k$ comes from the binary search, because the value of M^* is not known. After k iterations of FFD, $C_{\text{upper}} - C_{\text{lower}} \leq (\frac{1}{2})^k M^*$. Note also that $(\frac{1}{2})^7 \leq 0.008$, hence we usually use $k = 7$. Later, Friesen [3] improved the error bound to $(0.20 + (\frac{1}{2})^k)$.

Note that LPT has an advantage in running time over MULTIFIT if $k > 1$. However, MULTIFIT provides a better error bound. It is very important to note that it does not mean that MULTIFIT will have a better performance in all cases. Empirical tests show that in many cases LPT performs better than MULTIFIT.

(Please see for example, Section 3, [2] and [9]). A typical example is shown in the following. This example was used by [1] to show the tight error bound for 2-processors problem.

Example 1. Consider the scheduling problem with the following data.

$$m=2 \quad \text{and} \quad n=6 \quad \text{with} \quad p_1=3, \quad p_2=3,$$

$$p_3=2, \quad p_4=2, \quad p_5=2 \quad \text{and} \quad p_6=2.$$

Optimal solution: Assign jobs 1, 3 and 4 to machine 1 and the remaining jobs to machine 2. Makespan = 7.

MULTIFIT algorithm: If $7 \le C < 8$, then FFD will assign jobs 1 and 2 to machine 1, jobs 3, 4 and 5 to machine 2, and job 6 is left. Hence MULTIFIT will yield a makespan which is greater than or equal to 8.

LPT algorithm: Assigns jobs 1, 3 and 4 to machine 1 and the remaining jobs into the second machine. Makespan = 7, the optimal solution.

The above discussion motivates us to develop an algorithm which will always have a performance not worse than that of LPT and will have an error bound not worse than that of MULTIFIT.

In this paper we develop a combined algorithm, call it COMBINE, which uses LPT as an incumbent schedule and then applies FFD with narrower gap between initial C_{upper} and C_{lower}. The time complexity of COMBINE is $O(n \log n + kn \log m)$, the same as MULTIFIT, yet the number of iterations, k, is not greater than that of MULTIFIT. Since COMBINE uses LPT as the incumbent, the performance of COMBINE is never worse than that of LPT. The error bound of implementing COMBINE to the multiprocessor problem is never worse than that of MULTIFIT. For example the error bound of implementing COMBINE to the two-processor problem is $\frac{1}{9}$, compared to $\frac{1}{7}$ for MULTIFIT.

2. The combined algorithm

Before we describe the combined algorithm, we first point out some important properties of the LPT algorithm.

Lemma 1. *Let A be defined in equation* (1) *and M be the makespan obtained by the* LPT *algorithm for the m-processor problem. Then we have*

$$\text{(i)} \qquad \max\{M/(\tfrac{4}{3}-1/3m), p_1, A\} \le M^* \le M. \tag{2}$$

(ii) LPT *will assign a job to a machine only if the current load in that machine is less than A.*

(iii) *If $M \ge 1.5A$, then $M^* = M$.*

All proofs are in the appendix.

Now we are ready to describe the combined algorithm, call it COMBINE.

COMBINE Algorithm

Step 1. Apply LPT algorithm and let M be the makespan. If $M \geq 1.5A$, then stop. Otherwise go to the next step.

Step 2. Apply MULTIFIT algorithm with initial $C_{\text{upper}} = M$ and $C_{\text{lower}} = \max\{M/(\frac{4}{3} - 1/3m), p_1, A\}$.

Remarks. (i) Either the value of M obtained in Step 1, when $M \geq 1.5A$, or the final C_{upper} in Step 2 is the makespan obtained by COMBINE.

(ii) Without considering the time of sorting the data, the time complexity of running LPT is $O(n \log m)$, exactly the same as one iteration of FFD in MULTIFIT. We usually use 7 iterations of FFD in MULTIFIT to narrow the gap between C_{upper} and C_{lower} to be less than $(\frac{1}{2})^7 M^*$, $((\frac{1}{2})^7 = 0.0078)$. In COMBINE, although we first implement LPT, which has the same time complexity as FFD, yet LPT has narrowed down the gap between C_{upper} and C_{lower} to be less than $\frac{1}{3}M^*$. Hence we need at most 6 iterations of FFD to make the gap to be less than $\frac{1}{3}(\frac{1}{2})^6 M^*$, $(\frac{1}{3}(\frac{1}{2})^6 = 0.0052)$.

(iii) Instead of pre-setting the value of k, we may stop the algorithm once $C_{\text{upper}} - C_{\text{lower}} \leq \alpha A$, where α is a small positive number and is determined by decision maker. In Section 3, we use $\alpha = 0.005$.

(iv) In most practical problems, the makespan obtained by LPT is not greater than $1.2A$, hence we may need fewer iterations of FFD in COMBINE. In Example 1, since the makespan obtained by LPT is equal to A already, hence no iteration of FFD is needed in COMBINE. Usually, when the number of jobs compared to the number of machines available is large, then the makespan obtained by LPT is close to A (please see for example, Section 3 and [9]). Hence very few iterations of FFD are needed in COMBINE. This can be shown empirically in Section 3.

From the discussion in the above remark (ii) and Lemma 1, it is easy to see that the following theorem is true.

Theorem 1. *The error bound of implementing* COMBINE *with* $k-1$ *iterations of* FFD *to the multiprocessor problem is not worse than that of* MULTIFIT *with* k *iterations of* FFD.

Remark. Although we do not know for the general multiprocessor problem, how much improvement is obtained in the error bound for COMBINE over MULTIFIT, the following theorem shows that the improvement is significant for the two-processor problem.

Theorem 2. *If we apply the* COMBINE *algorithm, with one iteration in* LPT *and*

$k-1$ iterations in FFD, *to a two-processor problem, then the makespan is always within* $(\frac{10}{9}+(\frac{1}{2})^k)M^*$.

Remarks. (i) The error bound of applying the MULTIFIT algorithm, with k iterations of FFD, to a two-processor problem is $(\frac{1}{7}+(\frac{1}{2})^k)$. It is easy to see that the improvement for COMBINE over MULTIFIT in this case is significant.

(ii) More accurately, we should replace $(\frac{10}{9}+(\frac{1}{2})^k)M^*$ in Theorem 2 by $\min\{\frac{7}{6}M^*, (\frac{10}{9}+\frac{1}{6}(\frac{1}{2})^{k-1})M^*\}$ for the reason stated in the following. Since we use the result of LPT as the incumbent for COMBINE, and the error bound of LPT is $\frac{1}{6}$ for the two-processor problem, hence the makespan of COMBINE is not greater than $\frac{7}{6}M^*$ even if we do not use any iteration of FFD. Also we already reduce the value of $C_{\text{upper}}-C_{\text{lower}}$ to be less than or equal to $\frac{1}{6}M^*$ by LPT, hence after $k-1$ iterations of FFD, $C_{\text{upper}}-C_{\text{lower}}\leq\frac{1}{6}(\frac{1}{2})^{k-1}M^*$.

Example 2. Consider a scheduling problem with the following data.

$$m=2 \quad \text{and} \quad n=6 \quad \text{with} \quad p_1=5, \quad p_2=3,$$

$$p_3=3, \quad p_4=3, \quad p_5=2, \quad \text{and} \quad p_6=2.$$

Optimal solution: Assign jobs 1, 5 and 6 to machine 1 and the remaining jobs to machine 2. Makespan $=4=9$.

COMBINE algorithm: LPT will get a makespan of 10. For any $9\leq C<10$, FFD will assign jobs 1 and 2 to machine 1, jobs 3, 4 and 5 to machine 2, and job 6 is left. Hence COMBINE algorithm will yield a makespan of 10.

3. Empirical results

We compare the schedules obtained by COMBINE, MULTIFIT and LPT over different combinations of the number of jobs and machines available. For each fixed number of jobs and machines, 100 randomly generated problems were solved. The job sizes were generated from a normally distributed random variable truncated between 100 and 800. In applying COMBINE and MULTIFIT to the problem, we terminate the algorithm when $C_{\text{upper}}-C_{\text{lower}}\leq 0.005A$. The makespans of each trial were normalized by dividing it by $\max\{p_1, A\}$. This is a lower bound on the actual makespan, so the relative errors reported for each algorithm is somewhat exaggerated. We present the average performance of these three algorithms in Table 1.

We are also interested in the reduction of time complexity in using COMBINE compared to MULTIFIT. Hence Table 1 also reports the number of iterations of FFD used in COMBINE and MULTIFIT respectively, where we count the application of LPT in the combine as the first iteration.

As we mentioned in Section 2, Table 1 shows that COMBINE used fewer iterations than MULTIFIT. On the total 2000 trials, COMBINE used an average of 3

Table 1
Comparison for schedules obtained by COMBINE, MULTIFIT, and LPT.

Number of jobs	Number of machines	COMBINE		MULTIFIT		LPT
			*		**	
10	2	1.004	(1.7)	1.050	(8.0)	1.006
10	3	1.054	(6.1)	1.055	(8.0)	1.145
30	2	1.001	(1.0)	1.014	(8.0)	1.001
30	3	1.003	(1.3)	1.013	(8.0)	1.004
30	4	1.025	(5.0)	1.025	(8.0)	1.050
30	6	1.006	(2.2)	1.042	(8.0)	1.009
30	8	1.056	(5.0)	1.063	(8.0)	1.064
30	10	1.014	(3.3)	1.060	(8.0)	1.018
50	2	1.001	(1.0)	1.007	(8.0)	1.001
50	3	1.010	(3.0)	1.014	(8.0)	1.015
50	4	1.010	(4.0)	1.010	(8.0)	1.029
50	6	1.021	(5.0)	1.022	(8.0)	1.057
50	8	1.033	(5.7)	1.032	(8.0)	1.083
50	10	1.007	(2.4)	1.043	(8.0)	1.010
100	2	1.000	(1.0)	1.004	(8.0)	1.000
100	3	1.004	(2.9)	1.004	(8.0)	1.012
100	4	1.001	(1.0)	1.006	(8.0)	1.001
100	6	1.006	(3.0)	1.007	(8.0)	1.016
100	8	1.015	(4.0)	1.014	(8.0)	1.030
100	10	1.003	(1.3)	1.020	(8.0)	1.004

* Number of iterations, including one iteration of LPT, for COMBINE.
** Number of iterations for MULTIFIT.

iterations compared to 8 for MULTIFIT. It is important to note that on the average, even though COMBINE uses fewer iterations than MULTIFIT, it yields a better makespan.

We believe that COMBINE provides a good alternative to both LPT and MULTIFIT, since it takes advantage of both algorithms. Theorem 1 shows that its error bound is not worse than that of MULTIFIT, and hence is also better than that of LPT. Also it provides a better average performance than either, as we can see from Table 1.

Appendix A. Proofs

In all the proofs we assume that the jobs are already sorted so that $p_1 \geq p_2 \geq \cdots \geq p_n$.

Proof of Lemma 1. (i) Note that it is trivial that M is an upper bound of M^*. It is obvious that p_1 and A are both lower bounds of M^*. Since Graham [7] has proved that $M \leq (\frac{4}{3} - 1/3m)M^*$, it is also true that $M/(\frac{4}{3} - 1/3m)$ is a lower bound of M^*.

(ii) Note that LPT always assigns a job to a current lowest loaded machine. Hence it is obvious that the current load in that machine is less than A. Otherwise it means that the current load of every machine is not less than A, which implies that the total load already is greater than or equal to mA, a contradiction.

(iii) Let M be the makespan obtained by LPT and let q be the number of jobs in the machine with total processing time equal to M. If there are several machines with total processing time equal to M, then let q be the largest number of jobs among them. Also let j be the last job that was assigned to that particular machine. If $q \geq 3$, then (ii) implies the total load in that machine excluding job j is less than A. Hence it is easy to see that $p_j < 0.5A$, because $p_i \geq p_j$ for all $i \leq j$. Thus $M < 1.5A$.

Now suppose that $q = 1$. Then it is obvious that $p_1 = M = M^*$. The remaining case is that $q = 2$. In this case, let M occur in machine i and let j be the last job in machine i. If before job j was assigned, there is already one machine, say machine k, loaded with more than two jobs, then, similar to the discussion in the last paragraph we see that the processing time of the current last job in machine k is less than $0.5A$. Hence $p_j < 0.5A$ and $M < 1.5A$. On the other hand, if before jobs j was assigned, all the machines have at most two jobs then $j = 2m - i + 1$ and the first $2m - i + 1$ jobs will be scheduled in the following way.

Machine number	Job assigned	
1	p_1	
2	p_2	
⋮	⋮	
i	p_i	p_{2m-i+1}
⋮	⋮	⋮
$m-1$	p_{m-1}	p_{m+2}
m	p_m	p_{m+1}

Note that $p_i + p_{2m-i+1} = M$. Suppose that $M \geq 1.5A$. We will show that in this case $M^* = M$. Note that for any scheduling, we will have the following two cases.

(a) Job i and job $2m - i + 1$ are assigned to the same machine.

(b) Job i was assigned to one machine, say machine k, and job $2m - i + 1$ is not assigned to machine k. Then we have three subcases.

(i) One job $q < 2m - i + 1$ is assigned to machine k.

(ii) There are two jobs j and r being assigned to the same machine, where $j \in \{1, \ldots, i-1\}$, and $r \in \{1, 2, \ldots, 2m - i + 1\}$.

(iii) There are more than two jobs being assigned to the same machine and all these jobs belong to $\{1, 2, \ldots, 2m - i + 1\}$.

For cases (a) and (b)(i), it is obvious that the makespan for that scheduling is not less than $p_i + p_{2m-i+1}$. For case (b)(ii), note that $p_j \geq p_i$ for all $j \leq i$, and hence we

have $p_j+p_r \geq p_i+p_{2m-i+1}$ if $j \in \{1,\ldots,i-1\}$ and $r \in \{1,2,\ldots,2m-i+1\}$. Namely, the makespan for that scheduling is not less than p_i+p_{2m-i+1}. For case (b)(iii), note that Lemma 1(ii) implies that $p_i<A$. This implies that $p_{2m-i+1}>0.5A>0.5p_i$. Hence any job set which contains more than two jobs from $\{1,2,\ldots,2m-i+1\}$ will have a total processing time greater than p_i+p_{2m+i-1}.

The above discussion has shown that the makespan for any scheduling is not less than p_i+p_{2m-i+1} and hence $M^*=p_i+p_{2m-i+1}=M$.

Proof of Theorem 2. The proof of the factor $(\frac{1}{2})^k$ is omitted here. (Please see [1], or the remark (ii) immediately following COMBINE algorithm.)

We will show that implementing COMBINE to a two-processor scheduling problem, either LPT will yield a makespan not greater than $\frac{10}{9}M^*$, or for any $C \geq \frac{10}{9}M^*$ FFD will fit all the jobs into two machines. Equivalently, we will show that for any $C \geq \frac{10}{9}M^*$ if FFD cannot fit all jobs into two machines, then LPT will yield a makespan that is not greater than $\frac{10}{9}M^*$.

For $i=1$ and 2, let N_i be the set of jobs assigned to machine i in an optimal solution and M_i^* be the total processing time of jobs assigned to N_i. Without loss of generality, assume that N_1 contains the first job. Now suppose that we apply FFD to the problem for $C \geq \frac{10}{9}M^*$ and let r be the largest index job that was not put into the first machine. Also let R be the sum of the processing times of all job j with $j>r$. For $i=1$ and 2, let M_i denote the total processing time of jobs in machine i when FFD is applied. Also let $|N_i|_r$ denote the number of jobs in N_i with index larger than r.

Note first that if $M_i \geq \min\{\frac{8}{9}M_1^*, \frac{8}{9}M_2^*\}$ for some i, then the total processing time of the remaining jobs will not be greater than $\frac{10}{9}M^*$, and hence we can fit all the jobs into two machines. Now suppose that $p_r \leq \frac{2}{9}M^*$, then it is easy to see that $M_1 \geq \frac{8}{9}M^*$ and we are done. For the remaining case suppose that $p_r > \frac{2}{9}M^*$. In this case it is easy to see that $|N_i|_r \leq 4$ for $i=1$ and 2.

Case (i): $|N_1|_r=1$. In this case it is easy to see that $M_1 \geq M_1^*$.

Case (ii): $|N_1|_r=2$. In this case it is also easy to see that $M_1 \geq M_1^*$.

Case (iii): $|N_1|_r=3$. Suppose that $2 \in N_1$. Since job 1 also belongs to N_1, it is easy to see that $M_1 \geq M_1^*$. If $2 \in N_2$ and $p_2 \leq p_k+\frac{1}{9}M^*$, where k is the second job in N_1, then we will have $M_1 \geq M_1^*$ because $C \geq \frac{10}{9}M^* \geq M_1^*+\frac{1}{9}M^*$.

Now suppose that $2 \in N_2$ and $p_2>p_k+\frac{1}{9}M^*$. If there exists one job j such that $\frac{8}{9}M^* \leq p_1+p_2+p_j \leq \frac{10}{9}M^*$, then $p_1+p_2+p_j \leq C$ and hence $M_1 \geq p_1+p_2+p_j \geq \frac{8}{9}M^*$ and we are done. Hence in the following we will assume that no such j exists and we will show that either $M_i \geq \min\{\frac{8}{9}M_1^*, \frac{8}{9}M_2^*\}$ or the makespan M yielded by LPT is not greater than $\frac{10}{9}M^*$. Note that applying LPT to the problem, we will first assign job 1 to machine 1 and jobs 2 and 3 to machine 2. Since $p_2>p_k+\frac{1}{9}M^* \geq p_r+\frac{1}{9}M^*>\frac{3}{9}M^*$, we have $p_2+p_3>\frac{5}{9}M^*$. Also the definitions of p_r and $|N_1|_r$ imply that $p_1<\frac{5}{9}M^*$. Hence $p_1<p_2+p_3$ and job 4 will be assigned to machine 1.

Let M be the makespan obtained by LPT and define q as the number of jobs in the machine on which M occurs. If M occurs in both machines, then choose the larger number of jobs as q. We have the following cases.

(a) $q=1$. Since $p_1<\frac{5}{9}M^*$, it is impossible that $q=1$.

(b) $q=2$. Then $p_1<\frac{5}{9}M^*$ implies that $M\leq\frac{10}{9}M^*$.

(c) $q=3$. Recall the definition of job r. If $r\leq 5$, then $|N_1|_r=3$ implies $M_1^*\geq p_1+p_4+p_5$. Also we have $|N_2|_r\leq 2$ and $M_2^*\leq p_2+p_3+R$. Note that $M_1\geq p_1+p_2+R\geq p_2+p_3+R\geq M_2^*$, and we are done.

Now suppose that $r>5$; then $p_6\geq p_r>\frac{2}{9}M^*$. Hence, if jobs 5 and 6 are assigned to the same machine, then the total processing time in that machine will be greater than M^*. This will imply a contradiction to the assumption that $q=3$. Thus we will consider the case that job 5 and job 6 are assigned to different machines.

Suppose that $|(p_1+p_4)-(p_2+p_3)|\leq\frac{2}{9}M^*$. It is easy to see that $p_5-p_6\leq\frac{2}{9}M^*$. From the various assumptions of this case made so far, $q=3$ will imply that $M-(2A-M)\leq\frac{2}{9}M^*$ and we have $M\leq A+\frac{1}{9}M^*\leq\frac{10}{9}M^*$. Now suppose that $|(p_1+p_4)-(p_2+p_3)|>\frac{2}{9}M^*$. Then either $p_1>p_2+\frac{2}{9}M^*>\frac{5}{9}M^*$ or $p_3>p_4+\frac{2}{9}M^*>\frac{4}{9}M^*$. The former case is impossible. The latter case will imply that $p_1>\frac{4}{9}M^*$. Since each machine contains at least three jobs from $\{1,2,\ldots,6\}$, we see that the total processing time in each machine is greater than $\frac{8}{9}M^*$ and we are done.

(d) $q=4$. First note that if $p_3\geq\frac{4}{9}M^*$, then it is easy to see that $\frac{8}{9}M^*\leq p_2+p_3\leq\frac{10}{9}M^*$ and we have $M\leq\frac{10}{9}M^*$. Now suppose that $p_3<\frac{4}{9}M^*$.

Let j be the last job in the machine in which M occurs. If $p_j\leq\frac{2}{9}M^*$, then $M\leq\frac{10}{9}M^*$ because before being assigned job j the total load in that machine was not greater than $(2A-p_j)/2$. Now suppose $p_j>\frac{2}{9}M^*$, then it is obvious that $p_1+p_2+p_j>\frac{8}{9}M^*$. Hence if we can prove that $p_1+p_2+p_j\leq\frac{10}{9}M^*$, then we have $M_1\geq p_1+p_2+p_j>\frac{8}{9}M^*$ and we are done.

Since $p_1+p_4<\frac{5}{9}M^*+p_4<(p_2+p_5)+p_4\leq p_2+p_3+p_5$, we will assign job 6 to machine 1 if we have assigned job 5 to machine 2. Since $p_3<\frac{4}{9}M^*$, we have $p_2+p_3\leq p_1+p_3<p_1+(p_4+p_5)$. Hence, if we assign job 5 to machine 1, we will assign job 6 to machine 2. Thus we see that $j\geq 7$ and $p_3+\cdots+p_{j-1}>4(\frac{2}{9})M^*=\frac{8}{9}M^*$. This implies that $p_1+p_2+p_j<2A-\frac{8}{9}M^*\leq\frac{10}{9}M^*$.

(e) $q\geq 5$. Let j be the last job in the machine in which M occurs. Then $p_j<\frac{2}{9}M^*$, because otherwise the total load in that machine before j was assigned was greater than M^* which contradicts Lemma 1(ii). Since $p_j<\frac{2}{9}M^*$, we have $M<\frac{10}{9}M^*$, because before job j was assigned the total load in that machine is not greater than $(2A-p_j)/2$.

Case (iv): $|N_1|_r=4$. In this case $p_1\leq\frac{3}{9}M^*$. If $|N_2|_r\leq 3$, then $M_1\geq p_1+p_2+p_3+R\geq M_2^*$ and we are done. If $|N_2|_r=4$, then $r=8$. In this case $p_1+p_2+p_3+p_4\leq 2A-(p_5+p_6+p_7+p_8)\leq 2A-4p_r\leq 2M^*-4(\frac{2}{9})M^*=\frac{10}{9}M^*$. Hence $M_1\geq p_1+p_2+p_3+p_4+R\geq\frac{8}{9}M^*$.

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] S.E. Elmaghraby, and A.A. Eliman, Knapsack-based approaches to the makespan problem on multiple processors, AIIE Trans. 12 (1980) 87–96.

[3] D.K. Friesen, Tighter bounds for the MULTIFIT processor scheduling algorithm, SIAM J. Comput. 13 (1984) 170–181.

[4] D.K. Friesen and M.A. Langston, Evaluation of a MULTIFIT-based scheduling algorithm, J. Algorithms 7 (1986) 35–59.

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

[6] R.L. Graham, Bounds for certain multiprocessing anomalies, Bell System Tech. J. 45 (1966) 1563–1581.

[7] R.L. Graham, Bounds on multiprocessing timing anomalies, SIAM J. Appl. Math. 17 (1969) 263–269.

[8] D.S. Hochbaum and D.B. Shmoys, Using dual approximations algorithms for scheduling problems: theoretical and practical results, Presented at 12th International Symposium on Mathematical Programming at MIT (1985).

[9] C.Y. Lee and J.D. Massey, Multiprocessor scheduling: an extension of MULTIFIT, Research Report No. 86-9, Department of Industrial and System Engineering, University of Florida (1986).