第 23 卷 第 1 期
1997 年 1 月

自 动 化 学 报
ACTA AUTOMATICA SINICA

Vol. 23, No. 1
January, 1997

短 文

同等并行处理机上独立任务的调度[1)]

康一梅
（北京计算机与电子应用技术研究所　北京　100081）

郑应平
（中国科学院自动化研究所　北京　100080）

摘　要　n 个独立任务在 m 个同等并行处理机上处理，使总完成时间最小的非抢先调度是确定性调度理论的一个基本问题. 文中提出一种算法——Bound Fit 算法，它的最坏情况性能至少和 MULTIFIT 算法一样甚至更好，而所需的时间却比 MULTIFIT 算法少.

关键词　同等并行处理机，非抢先调度，启发式算法.

1　引言

n 个独立任务在 m 个同等并行处理机上处理，使总完成时间最小的非抢先调度是确定性调度理论的一个基本问题. 已知 n 个独立任务，$T=\{T_1,T_2,\cdots,T_n\}$ 表示任务集，m 个处理机，$M=\{M_1,M_2,\cdots,M_m\}$ 表示处理机集，任务的处理时间为 p_i，且 p_i 为整数，$i=1,2,\cdots,n$. 假设任务按处理时间非增顺序排列，即 $p_1\geqslant p_2\geqslant\cdots\geqslant p_n$. $C_i(i=1,2,\cdots,m)$ 表示第 i 个处理机的完成时间，$C_{\max}$ 表示总完成时间，即 $C_{\max}=\max\limits_{1\leqslant i\leqslant m}C_i$，$C^*_{\max}$ 表示最优调度的总完成时间.

这是一个 NP 难题[1]，不可能找到一个多项式时间最优算法. 因此，只能退而求其次，去寻找有效的次优算法. 在这方面有许多启发式算法，其中最著名的是 LPT 算法[2]和 MULTIFIT 算法[3]. 而 MULTIFIT 算法的最坏情况性能比 LPT 算法好许多.

MULTIFIT 算法是 Coffman 等人在文献[3]中提出的，它是一种基于 bin-packing 的 FFD(First Fit Decreasing)方法的迭代算法. 以时间代替容量，寻找使所有任务都排在 m 个处理机上的最小容量，即最短总完成时间. FFD 方法是对每一个固定的库容量，顺序将按处理时间非增顺序排列的任务安排在满足容量约束、序号最小的处理机上.

Donald K. Freisen 在文献[4]中证明了 MULTIFIT 算法的最坏情况性能为 $(1.20+(1/2)^k)C^*_{\max}$，其中 k 是迭代次数. MULTIFIT 算法的时间复杂性为 $O(n\log n+kn\log m)$，包括任务的处理时间排序所花的时间.

1) 本文得到国家自然科学基金资助.

收稿日期　1994-04-20

C. Y. Lee 等在文献[5]中提出一种将 LPT 算法与 MULTIFIT 算法结合起来的方法，以达到减少运算时间的目的，但在有些情况下本文提出的 Bound Fit 迭代算法更节省时间，并且能达到很好的性能.

2 预备算法

定义 处理机 M_i 上已安排的任务的完成时间称为处理机 M_i 的调度长度，记为 L_i.

很显然，最优调度满足

$$C^*_{\max} \geqslant \max\left\{\frac{1}{m}\sum_{i=1}^{n} p_i, p_1\right\}. \tag{2.1}$$

当上式等号成立时，下面的预备算法就可找到最优调度.

预备算法：

1. 令 $B = \max\left\{\frac{1}{m}\sum_{i=1}^{n} p_i, p_1\right\}$.

2. 令 $L_j = 0(j = 1,2,\cdots,m)$， $i = 1$.

3. 若至少存在一个 $j(1 \leqslant j \leqslant m)$ 使 $L_j + p_i \leqslant B$，将 T_i 安排在使 $L_j + p_i \leqslant B$ 且序号最小的处理机上. 否则，将 T_i 安排在使 $L_j + p_i (j = 1,2,\cdots,m)$ 最小的处理机上.

4. 若 $i < n$，令 $i = i + 1$，转向 3. 否则，停机.

只要最优调度的总完成时间为 $\max\left\{\frac{1}{m}\sum_{i=1}^{n} p_i, p_1\right\}$，上述预备算法就可得到最优调度，否则，预备算法不一定能得到最优调度.

定理 1. 当第 $k-1$ 次用预备算法所得调度的总完成时间 $C^{(k-1)}_{\max} > \max\left\{\frac{1}{m}\sum_{i=1}^{n} p_i, p_1\right\}$，若令 $B^{(k)} = C^{(k-1)}_{\max}$，则第 k 次用预备算法所得调度的总完成时间 $C^{(k)}_{\max} = C^{(k-1)}_{\max}$，其中 $k \geqslant 3$.

证明 设 $A = \max\left\{\frac{1}{m}\sum_{i=1}^{n} p_i, p_1\right\}$，当 $C^{(k-1)}_{\max} = A$ 时则找到了最优调度.

当 $k=2$ 时，$k-1=1$，第 1 次用预备算法时 $B^{(1)} = A$，只要 $C^{(1)}_{\max} \neq A$，一定会有任务在调度时使所有处理机的长度都超过 $B^{(1)}$. 第 k 次，即第 2 次用预备算法求调度时 $B^{(2)} = C^{(1)}_{\max}$，用预备算法安排任务最坏可得到与上次完全相同的调度，所得 $C^{(2)}_{\max} \leqslant C^{(1)}_{\max}$.

因此，当 $k \geqslant 3$ 时，令 $B^{(k)} = C^{(k-1)}_{\max}$，用预备算法安排任务与第 $k-1$ 次完全相同，即 $C^{(k)}_{\max} = C^{(k-1)}_{\max}$.

证毕.

3 Bound Fit 算法

定理 2. 当 $k \geqslant 3$ 时，用预备算法求任务调度若存在总完成时间 $C^{(k)}_{\max} \leqslant C^{(k-1)}_{\max} - 1$ 的调度，则令 $B^{(k)} = C^{(k-1)}_{\max} - 1$ 一定可得到 $C^{(k)}_{\max} \leqslant C^{(k-1)}_{\max} - 1$ 的调度.

证明 由定理 1 知，当 $k \geqslant 3$ 时令 $B^{(k)} = C^{(k-1)}_{\max}$ 所得调度满足 $C^{(k)}_{\max} = C^{(k-1)}_{\max}$. 由于假定任务处理时间是整数，因此为了得到更好的结果，令 $B^{(k)} = C^{(k-1)}_{\max} - 1$，这样若用预备算法

求任务调度存在 $C_{\max}^{(k)} \leqslant C_{\max}^{(k-1)}-1$ 的调度时就可得到此调度. 反之,若 $B^{(k)}$ 不论为何值都无法得到 $C_{\max}^{(k)} \geqslant C_{\max}^{(k-1)}-1$ 的调度,则所得调度的总完成时间 $C_{\max}^{(k)} \geqslant C_{\max}^{(k-1)}-1$.

证毕.

由以上分析得到下面算法.

Bound Fit 算法

1. 令 $B^{(1)}=\max\left\{\frac{1}{m}\sum_{i=1}^{n}p_i, p_1\right\}, k=1$.

2. 令 $L_j=0(j=1,2,\cdots,m), i=1$.

3. 若至少存在一个 $j(1\leqslant j\leqslant m)$使 $L_j+p_i\leqslant B^{(k)}$，将 T_i 安排在使 $L_j+p_i\leqslant B^{(k)}$且序号最小的处理机上. 否则,将 T_i 安排在使 $L_j+p_i(j=1,2,\cdots,m)$最小的处理机上. 改变相应的 L_j.

4. 若 $i<n, i=i+1$,转向 3.

5. 若 $\max\limits_{1\leqslant j\leqslant m}L_j=A$ 或 $k\geqslant 3$ 且 $\max\limits_{1\leqslant j\leqslant m}L_j>B^{(k)}$,停机. 否则,令 $k=k+1$,若 $k<3$,令 $B^{(k)}=\max\limits_{1\leqslant j\leqslant m}L_j$,转向 2;若 $k\geqslant 3$,令 $B^{(k)}=\max\limits_{1\leqslant j\leqslant m}L_j-1$,转向 2.

最后,若 $k\geqslant 3$,以 $B^{(k-1)}$再用预备算法求调度,所得调度即为 Bound Fit 算法的结果.

用 Bound Fit 算法可得到与 MULTIFIT 算法同样或更好的结果.

例. 假设调度问题的数据如下:$m=3$，$n=8$，$p_1=100, p_2=p_3=90, p_4=42$，$p_5=41$，$p_6=30$，$p_7=20$，$p_8=7$.

最优调度的总完成时间是 141. Bound Fit 算法和 MULTIFIT 算法都得到了最优调度,但是MULTIFIT 算法需 5 次迭代得到最优调度,而 Bound Fit 算法只需 3 次迭代,如表 1 所示.

表 1　用 Bound Fit 算法与 MULTIFIT 算法求解比较

k	$C_{\max}$	
	Bound Fit	MULTIFIT
1	150	210
2	142	172
3	141	151
4		142
5		141

我们对 100 个随机产生的例子进行仿真,结果表明 Bound Fit 算法所需的迭代次数总比 MULTIFIT 算法所需的迭代次数少.

4　最坏情况性能分析

用 $C_{\max}^{BF}$表示用 Bound Fit 算法所得调度的总完成时间,我们可以证明下面的定理.

定理 3.　若 $C_{\max}$是用 FFD 方法在 m 个处理机上安排任务所得任一调度的总完成时间,则

$$C_{\max}^{BF}\leqslant C_{\max}$$

证明　用反证法证明.

设有一用 FFD 方法在 m 个处理机上安排任务所得调度的总完成时间 $C'_{\max}<C_{\max}^{BF}$. 由于 $C'_{\max}$是用 FFD 方法在 m 个处理机上将全部任务安排完所得的调度的总完成时间,FFD

方法在 $C=B$ 且能将 n 个任务全部安排在 m 个处理机上时的任务安排方法完全相同，那么令 $B=C'_{\max}$，用 Bound Fit 算法一定可得到与 FFD 方法所得相同的调度. 则由定理 2 可知，令 $B=C'_{\max}-1$，可得 $C^{BF'}_{\max}\leqslant C'_{\max}-1$；当不存在总完成时间小于或等于 $C'_{\max}-1$ 的调度时，Bound Fit 算法至少可得 $C^{BF}_{\max}=C'_{\max}$. 故与假设矛盾，定理得证.

由定理 3，可得下面的推论.

推论 1. 用 Bound Fit 算法所得的调度与 MULTIFIT 算法在 $k=\infty$ 时所得结果相同.

用 $R_m(BF)$ 表示 Bound Fit 算法的绝对性能，定理 4 给出 Bound Fit 算法的最坏情况性能.

定理 4. 对任意 T，$R_m(BF)\leqslant 1.20$.

证明 文献[4]中证明了 MULTIFIT 算法的绝对性能 $R_m(MUL)\leqslant 1.20+\left(\frac{1}{2}\right)^k$，由推论 1 可知

$$R_m(BF)=\lim_{k\to\infty}R_m(MUL),$$

则

$$R_m(BF)\leqslant\lim_{k\to\infty}\left[1.20+\left(\frac{1}{2}\right)^k\right]=1.20.$$

参 考 文 献

[1] J. D Ullman. Complexity of sequencing problems. Computer and Job/shop scheduling Theory. New York, 1976.

[2] Graham R L. Bounds on multiprocessing timing anomalies. *SIAM J. Appl. Math.*, 1969, (17): 416—429.

[3] F G Coffman Jr., Garey M R, Johnson D S. An application of bin-packing to multiprocessor scheduling. *SIAM J. Compt.*, 1978, (7): 1—17.

[4] Freisen D K. Tighter bounds for the MULTIFIT processor scheduling algorithm. *SIAM J. compt.*, 1984, (13): 170—181.

[5] C Y Lee, David Massey J. Multiprocessor scheduling: combining LPT and MULTIFIT. *Discrete Appl. Math.*, 1988, (20): 233—242.

INDEPENDENT TASKS SCHEDULING ON IDENTICAL PARALLEL PROCESSORS

KANG YIMEI

(*Beijing Institute of Computer & Electronics Application Beijing* 100080)

ZHENG YINGPING

(*Institute of Automation, The Chinese Acadamy of Seiences Beijing* 100080)

Abstract The problem of nonpreemtively scheduling n independent tasks on m identical parallel processors in order to minimize the makespan is one of the fundamental problems of deterministic scheduling theory. In this paper, an algorithm called Bound Fit is presented. This algorithm has the same or better worst case performance with shorter running time.

Key words Identical parallel processor, nonpreemptive scheduling, heuristic.