# 基于时区与时区估计的层次调度模型*

黄必清 张 钹

(清华大学计算机系,北京 100084)

**摘要** 本文针对偏序集(POS)任务的调度问题,提出一种基于时区与时区估计的层次调度模型.该模型与界定搜索(Beam Search)方法有机结合,使得本文给出的调度算法具有搜索空间小、求解速度快的优点.

**关键词** 调度,任务位级,时区,界定搜索,FB下界.

调度问题在50's—70's基本上属于运筹学(Operations Research)的研究范畴,并且得到一些很有名的结果[1].70's—90's,由于调度问题的NP难解性,许多学者逐渐意识到人工智能和知识工程技术对调度研究的重要性.最著名的AI调度系统OPAL及ISIS系列,就是这一认识的产物[2,3].本文描述一类实用性极强的调度问题,即偏序集任务(任务之间的先序约束为一偏序关系)的调度,并采用AI方法对这一类调度问题做了一些有益的探讨.

## 1 调度问题的状态空间表示

### 1.1 问题的提出

给定一偏序集任务图(简称偏序图) $G=(T,<)$,顶点集T为一任务集合,$T=\{T_1,T_2,\cdots,T_n\}$,其中每个任务$T_i$的执行时间为$D_i=D(T_i)$,"$<$"为一偏序关系(Partially Ordered Relation),表示任务之间的先序约束.若$T_i$, $T_j\in T$, $T_i<T_j$,则表示$T_j$必须在$T_i$完成之后才能进行.我们的调度目标是,在满足任务之间先序约束的前提下,将n个任务分配到m个处理机上执行,使得所有任务尽可能早地完成.我们引入调度长度LS(Length of Schedule)的概念.对于一给定调度S,假定第j个处理机上最后一个任务的完成时间为$FF_j$,则

$$LS=\max_{1\leqslant j\leqslant m} FF_j \tag{1}$$

因而调度目标变为

$$\min_S LS=\min_S(\max_{1\leqslant j\leqslant m} FF_j) \tag{2}$$

本文讨论假定:(1)所有任务对任一处理机的执行时间一样,(2)任务不容许抢占调度

* 本文1992—10—08收到,1993—02—24定稿

作者黄必清,28岁,助教,在读博士生.主要研究领域为计算机应用,人工智能,Petri Net.张钹,59岁,教授,主要研究领域为计算机应用,人工智能,神经元网络.

本文通讯联系人:黄必清,北京 100084,清华大学计算机系

(nonpreemptive).即一旦某个任务被指派到某个处理机上执行,就不容许其它任务抢占该处理机,直到该任务执行完成为止.

### 1.2 调度的状态空间表示方法

多处理机的调度状态,可以通过单个处理机的调度状态来描述.假定 mt 为一 m－元组(m－tuples),形如:

$$mt=\{(QT_1,SF_1),(QT_2,SF_2),\cdots,(QT_m,SF_m)\}$$

第 j 个元素$(QT_j,SF_j)$描述第 j 个处理机的状态,$QT_j$ 表示第 j 个处理机上由先到后所执行的任务组成的队列,$SF_j$ 也是一队列,其元素分别为 $QT_j$ 相应任务的完成时间,令 $F_j$ 为 $SF_j$ 的队尾元素,即第 j 个处理机上当前任务的完成时间.这样我们就可以用 mt 来表示问题空间的一个状态了.

接着,我们定义操作 update.当调度处于状态 mt 时,假定有一处理机 $M_j$ 最早处于空闲状态,即 $F_j=\min F_i$,那么在 $t=F_j$ 时分配一任务 $T_r$ 给 $M_j$ 后,调度状态就变为:

$$\begin{aligned}mt' &= update(mt,\ j,\ T_r)\\ &=\{(QT_1,\ SF_1),\cdots,(QT_j\leftarrow T_r,\ SF_j\leftarrow F_j+D(T_r)),\cdots,(QT_m,\ SF_m)\}\end{aligned}$$

其中,a←b 表示将元素 b 加入队列 a 的队尾,返回一新队列 a;当 $T_r=0$ 时,$T_r$ 为空任务,$D(T_r)$为下次最早空闲时间与 $F_j$ 的差.

因此,问题的状态空间可表示为:

(a)状态以 m－元组的形式表示;

(b)初始状态为{(0,0),(0,0),…,(0,0)},表示没有任务分配给任何一个处理机;

(c)目标状态为所有任务都已分配并且满足(2)式情形下的 m－元组.

## 2 基于任务位级的滤波器

### 2.1 任务位级(Level)

本文给出一任务位级的概念来描述任务的优先程度

$$Level(T_i)=Depth_i+\mu\sum_{T_j\in Succ(T_i)}Depth(T_j)\tag{3}$$

任务 $T_i$ 的深度 $Depth_i=Depth(T_i)$,是偏序图中任务 $T_i$ 到端点任务(无后继任务的任务)之间最大路径的长度,即

$$Depth(T_i)=\begin{cases}D_i & Succ(T_i)=\Phi\\ D_i+\max\limits_{T_j\in Succ(T_i)}Depth(T_j) & otherwise\end{cases}\tag{4}$$

对于一给定的偏序图,我们可以求出一处理机数目的下界 $M_L$,使得调度长度不超过关键路径 $t_{cp}$.如果已有的处理机数目小于且接近 $M_L$,为充分利用现有资源(处理机)则要考虑多开发一些并行性,即 $\mu$ 的选取要大.本文假定 $\mu=1$.

### 2.2 位级占优的任务滤波器

从某种程度上来说,任务位级可以认为是一种指导调度决策的启发信息.在调度决策的某一点,若存在很多就绪任务可供选择,那么根据启发信息 Level,选择一部分最有可能获得最优解的任务,即滤波,以避免组合爆炸,是一种有效缩小搜索空间的方法.滤波是优先

选择位级较高的任务.对任务集T滤波的方法是:首先求出任务位级最高的任务$T_i$,然后以$T_i$为基准,删除那些相对$T_i$而言位级很低的任务.我们称之为位级占优的任务滤波器,可用函数filter( )来描述之.

filter( );

1. filter:=[ ];

2. q:=max Level($T_i$);

3. if $T_j \in T$,且 Level($T_j$)$\geqslant \eta q$,then filter:=filter+[$T_j$];

4. 对集合filter进行Level的降序排列;

一般认为$\eta \leqslant 0.5$时,基本能保证解的最优性,我们选$\eta=0.5$.

## 3 基于时区与时区估计的分层调度策略

我们提出的分层调度策略是由基于时区的状态扩展和基于时区估计的界定搜索有机地组合而成的.

### 3.1 基于时区的状态扩展方法

Chen的状态扩展方法[4]是基于时间点的,而我们的方法是基于时区的,即在某一时区内进行K次任务分配,完成一次状态扩展.显然,时区是随任务的完成时间不同而动态改变的.Chen的搜索树层数为n+1,而我们的搜索树层数为[n/k]+1.本文采用位级占优的任务滤波器,有效地进行状态扩展.k=2时,基于时区的状态扩展算法可描述为:

**算法 SE2(mt):**

1. 在mt中寻找最早处于空闲状态的处理机$M_{j1}$,计算$t_1=F_{j1}$时刻的就绪任务集$R(t_1)$,通过滤波得$FR(t_1)$,如果mt中所有的$F_j$不完全相等,则$FR(t_1):=FR(t_1)+[0]$.令$L_1=|FR(t_1)|$;

2. FOR i1:=1 TO $L_1$ DO

将$FR(t_1)$中的第i1个任务$T_{i1}$分配给$M_{j1}$,mt′=update(mt,j1,$T_{i1}$);

3. 在mt′中寻找最早处于空闲状态的处理机$M_{j2}$,计算$t_2=F_{j2}$时刻的就绪任务集$R(t_2)$,通过滤波得$FR(t_2)$,如果mt′中所有的$F_j$不完全相等,则$FR(t_2):=FR(t_2)+[0]$.令$L_2=|FR(t_2)|$;

4. FOR i2:=1 TO $L_2$ DO

将$FR(t_2)$中的第i2个任务$T_{i2}$分配给$M_{j2}$,mt″=update(mt′,j2,$T_{i2}$);

ENDFOR;

ENDFOR;

本次状态扩展是在时区$[F_{j1},F_{j2}]$上进行的.

### 3.2 基于时区估计的界定搜索方法

在SE2算法中,一次状态扩展使mt产生$L_1 * L_2$个子节点,搜索树增生极快,这将不利于调度求解.利用全局启发信息,优先扩展最有希望的部分子节点,即界定搜索是一个好主意.

假定界定搜索的宽度是BW(Beam Width).基于时区估计的界定搜索方法是:对SE2算法求得的子节点mt″作一调度长度的估计,保留BW个调度长度较小的子节点,其余子节点删除,以缩小搜索空间.

## 4　调度长度下界的计算

仔细研究一下处理机数目的下界 $M_L$ 和调度长度的下界 $t_L$,对我们的调度研究极有帮助.任务位级的计算需要 $M_L$,而 $t_L$ 可用于调度长度的估计,这正是界定搜索所必需的.

### 4.1　关于 FB 下界

FB 下界是建立在活动和任务分布函数两个基本概念之上的[5]. 当任务 $T_j$ 被安排在 $t=ST_j$ 时执行,$T_j$ 的活动可描述为

$$f(ST_j,t)=\begin{cases}1 & t\in[ST_j,ST_j+D_j]\\ 0 & \text{otherwise}\end{cases} \tag{5}$$

由此可得任务分布函数

$$F(ST,t)=\sum_{j=1}^{n}f(ST_j,t)\cdot T_j \tag{6}$$

我们称 ST 为任务开始准则,$ST_j$ 表示准则 ST 下任务 $T_j$ 的开始时间,EST 和 LST 分别表示任务最早开始准则和最晚开始准则.相应地,$EST_j$ 和 $LST_j$ 分别表示任务 $T_j$ 的最早、最晚开始时间,$EFT_j$ 和 $LFT_j$ 分别为任务 $T_j$ 的最早、最晚完成时间,那么,F(EST,t)与 F(LST,t)分别表示最早、最晚的任务分布函数,并且有

$$EST_j=\begin{cases}0 & Prec(T_j)=\Phi\\ \max\limits_{T_i\in Prec(T_j)}(EST(T_i)+D(T_i)) & \text{otherwise}\end{cases} \tag{7}$$

$$LST_j=t_{cp}-Depth(T_j) \tag{8}$$

$$EFT_j=EST_j+D(T_j) \tag{9}$$

$$LFT_j=LST_j+D(T_j) \tag{10}$$

$$F(EST,t)=\sum_{j=1}^{n}f(EST_j,t)\cdot T_j \tag{11}$$

$$F(LST,t)=\sum_{j=1}^{n}f(LST_j,t)\cdot T_j \tag{12}$$

由此引入包(Bag)的概念

$$B(\theta_1,\theta_2)=\sum_{t=\theta_1+1}^{\theta_2}F(ST,t) \tag{13}$$

$$EB(\theta_1,\theta_2)=\sum_{t=\theta_1+1}^{\theta_2}F(EST,t) \tag{14}$$

$$LB(\theta_1,\theta_2)=\sum_{t=\theta_1+1}^{\theta_2}F(LST,t) \tag{15}$$

其中,分布包 $B(\theta_1,\theta_2)$、最早分布包 $EB(\theta_1,\theta_2)$、最晚分布包 $LB(\theta_1,\theta_2)$均为以任务为元素的多集(Multi—set),分别简记为 B、EB、LB. 若 $n_j\cdot T_j\in B(\theta_1,\theta_2)$,则表示任务 $T_j$ 的执行落在 $[\theta_1,\theta_2]$内 $n_j$ 个时间单元.

FB 下界定理是指:对于偏序图 G,给定 m 个处理机,调度的 FB 时间下界(调度所需的最小完成时间)$t_L$ 为

$$t_L = t_{cp} + \lceil q \rceil \tag{16}$$

这里

$$q = \max_{[\theta_1,\theta_2] \subseteq [0,t_{cp}]} q[\theta_1,\theta_2] \tag{17}$$

$$q[\theta_1,\theta_2] = -(\theta_2 - \theta_1) + |EB \cap LB|/m \tag{18}$$

其中$\lceil q \rceil$表示大于等于 q 的最小整数.

### 4.2 FB 下界的推广

E. B. Fernandez 提出一种目前最好的下界求解方法[5],但这种方法是针对整个偏序图的. Chen 改进了它,将它用于搜索过程中某一点的调度估计,但 Chen 没有充分利用已有的调度决策,这是因为 Chen 将已调度但在 t=Fmin(N)时刻没有完成的任务 K(N)的剩余部分看成是未被调度的任务,增加了这些任务执行的灵活性,从而使 q 的计算偏低. 本文采用动态更新任务分布的方式克服了这一缺点.

我们对 q 的计算稍加一些改动,即可用于某一中间调度状态下的调度估计. 我们用 f(mt)来表示状态 mt 下的调度估计.

$$f(mt) = h(mt) + g(mt) \tag{19}$$

h(mt)表示在状态 mt,所有处理机的最小工作时间;

g(mt)表示在状态 mt,处理机完成剩余任务所需时间的下界. 由式(16)即得

$$g(mt) = \max \text{Depth}(T_j) + \lceil q \rceil \tag{20}$$

式(20)是对 G 的某一子图而言的. 该子图是由未调度任务和未完成任务的剩余部分所构成的.

在状态 mt 下,最早的处理机空闲时间为 $F_i$. 因此当 $t=F_i$ 时,有一些任务仍处于执行状态,考虑到任务的不可抢占性,这些任务的最早、最晚分布可由式(21—22)动态更新. 这正是我们不同于 Chen 的地方.

$$EST_j = LST_j = 0 \tag{21}$$

$$EFT_j = LFT_j = FT_j - t \tag{22}$$

其中 $FT_i$ 表示在状态 mt,$T_i$ 的实际完成时间.

## 5 调度实例及结论

我们仍引用 Chen 的例子来说明我们的算法,并与 Chen 的方法作一比较.

G=(T,<)是一初始偏序图,如图 1. 依据式(3)(4)(7)—(10)可计算出各个任务的深度、位级、最早开始时间、最晚开始时间、最早完成时间、最晚完成时间,其结果如表 1. 最早的任务分布 F(EST,t)和最晚的任务分布 F(LST,t)如图 2. 根据 F(EST,t)和 F(LST,t),可构造最早分布包和最晚分布包的交 EB∩LB (该项数据较多,本文从略). 根据式(19)(20),我们可以计算得初始偏序图对两个处理机的调度估计

$$f(mt_0) = h + g = h + t_{cp} + \lceil q \rceil = 0 + 15 + \lceil 0.5 \rceil = 16$$

调度过程中间状态 g(mt)的计算方法是:首先求出未调度任务(包括未完成任务)构成的子图,然后按上述方法求解 g(mt).

本文以两个处理机为例说明调度的实现过程. 我们采用深度优先的界定搜索方法,取

BW=4,即每次状态扩展之后,对每一扩展节点计算其评价函数,仅保留其中f(mt)较小的4个节点,并且优先扩展评价函数最小的节点.图3和表2描述了这一调度过程.图4为这一调度的Gantt图.

现将我们的算法与Chen的算法做比较,如表3.基于时区的状态扩展,使算法由点最优转化为局部最优,提高了求解问题的层次,而搜索效率的提高则受益于适当的评价函数和相应的界定搜索.两者的有机结合,使得本文提出的分层调度策略取得了搜索空间小、求解速度快的效果.

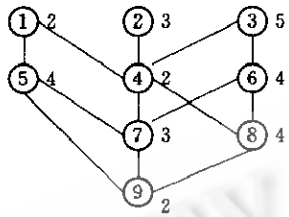


图1

注:圆圈内的数字为任务编号
圆圈边的数字为任务执行时间

表1

| $T_i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| $D_i$ | 2 | 3 | 5 | 2 | 4 | 4 | 3 | 4 | 2 |
| $Depth_i$ | 11 | 11 | 15 | 8 | 9 | 10 | 5 | 6 | 2 |
| $Level_i$ | 28 | 19 | 33 | 19 | 16 | 21 | 7 | 8 | 2 |
| $EST_i$ | 0 | 0 | 0 | 5 | 2 | 5 | 9 | 9 | 13 |
| $EFT_i$ | 2 | 3 | 5 | 7 | 6 | 9 | 12 | 13 | 15 |
| $LST_i$ | 4 | 4 | 0 | 7 | 6 | 5 | 10 | 9 | 13 |
| $LFT_i$ | 6 | 7 | 5 | 9 | 10 | 9 | 13 | 13 | 15 |

表2

| Nod n | mt | $h+t_{ep}+\lceil q\rceil=f$ |
|---|---|---|
| 1 | {(3,5),(1,2)} | 2+13+1=16 |
| 2 | {(3,5),(2,3)} | 3+12+1=16 |
| 3 | {(1,2),(2,3)} | 2+15+0=17 |
| 4 | {(1,2),(0,2)} | 2+15+0=17 |
| 5 | {(4,7),(2,5)} | 5+10+1=16 |
| 6 | {(6,9),(2,5)} | 5+10+1=16 |
| 7 | {(0,5),(2,5)} | 5+10+1=16 |
| 8 | {(2,8),(5,6)} | 6+10+0=16 |
| 9 | {(4,11),(5,9)} | 9+8+0=17 |
| 10 | {(6,9),(5,11)} | 9+7+0=16 |
| 11 | {(6,9),(0,9)} | 9+9+0=18 |
| 12 | {(5,13),(0,9)} | 9+9+0=18 |
| 13 | {(8,13),(7,14)} | 13+3+0=16 |
| 14 | {(8,13),(0,13)} | 13+5+0=18 |
| 15 | {(0,11),(8,15)} | 11+6+0=17 |
| 16 | {(0,11),(7,14)} | 11+6+0=17 |
| 17 | {(0,14),(9,16)} | 14+2+0=16 |

注:mt列中仅列出各状态下t一元组相应的队尾元素

表3

| | 扩展次数 | 扩展节点 |
|---|---|---|
| Chen's | 8 | 22 |
| Ours | 5 | 17 |

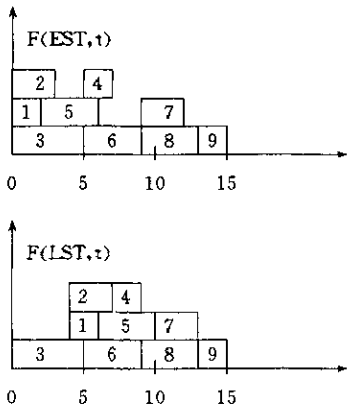


图2

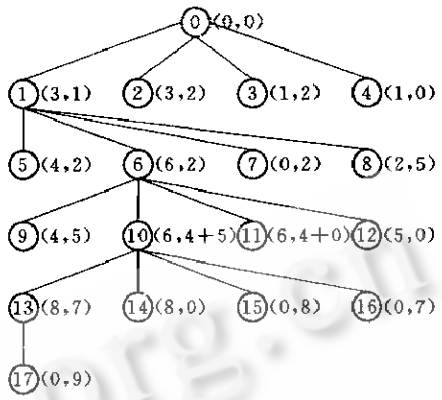


图3

注：圆圈内为状态编号；圆圈边的二元组分别描述两个处理机当前状态下所分配任务的序列

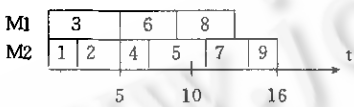


图4

## 参考文献


1 Graves S C(MIT). A review of production scheduling. Operations Research, 1981;**29(4)**:647－675.

2 Bensana E, Bell G, Dubois D. OPAL: a multi－knowledge－based system for industrial job－shop scheduling. Int. J. Prod. Res., 1988;**26(5)**:795－819.

3 Fox M S. Constraint guided scheduling: a short history of research at CMU. Computer in Industry, 1990;**14(1－3)**:79－88.

4 Chen C L, George C S Lee, Hou E S. Efficient scheduling algorithms for robot inverse dynamics computation on a multiprocessor system. IEEE Trans. on System, Man and Cybernetics, 1988;**18(5)**:729－743.

5 Fernandez E B, Bussell B. Bound on the number of processors and time for multiprocessor optimal schedules. IEEE Trans. on Computer, 1973;**C－22(8)**:745－751.


# TEMPORAL INTERVAL(TI)－BASED AND TI－EVALUATION －BASED HIERARCHICAL SCHEDULING MODEL


Huang Biqing and Zhang Bo

(*Department of Computer Science, Tsinghua University, Beijing* 100084)



**Abstract** Aimed at the scheduling problem of Partially Ordered Set(POS) tasks, Temporal Interval(TI)－based and TI－evaluation－based hierarchical scheduling model is established. It is organically combined with beam search method so that the scheduling algorithm in this paper has the advantages of small searching space and high solving speed.



**Key words** Scheduling, task level, temporal interval, beam search, FB lower bound.