第 23 卷 第 3 期
1997 年 5 月

自 动 化 学 报
ACTA AUTOMATICA SINICA

Vol. 23, No. 3
May, 1997

短 文

避免死锁调度的数学建模与求解

杨 盛 吴 澄

（清华大学自动化系 北京 100084）

摘 要 通过引入“资源分配函数”的概念建立了以最短加工时间为目标函数的制造系统避免死锁调度的数学模型，给出了该模型在两进程情况的最优解算法和多进程情况的可行解算法.

关键词 死锁，调度，制造系统.

1 引 言

死锁现象广泛存在于计算机操作系统[1]、网络系统[2]、生产制造系统[3-4]等多进程、多作业的复杂系统. 产生死锁的原因在于系统资源的有限和进程推进顺序的不合理. 死锁的产生将严重损害系统的性能，因此处理死锁是多进程系统的一个重要方面.

文[1]论述了计算机操作系统中产生死锁的充要条件和消除死锁的主要方法. 由于生产制造系统与操作系统相比有其自身的特点，两者的行为、约束和优化目标都不尽相同. 本文以生产制造系统为背景研究避免死锁调度的数学建模与求解.

2 数学建模与求解

在复杂的生产制造系统中往往同时存在多类生产资源和多个生产任务. 一个生产任务的一次实现称为该任务的一个进程.

定义 1. 设生产系统 Σ 中存在 n 类资源，其中第 i 类资源的数目为 k_i，$i=1,2,\cdots,n$. 定义系统的资源向量 $\boldsymbol{k}=[k_1,k_2,\cdots,k_n]^T$. 这个系统用二元组$(\Sigma,\boldsymbol{k})$表示.

定义 2. 已知系统$(\Sigma,\boldsymbol{k})$. 在任意时刻 t，某个进程 p 对第 i 类资源的占用数为 $r_i(t)$，$i=1,2,\cdots,n$. 定义该进程的资源分配函数 $\boldsymbol{r}(t)=[r_1(t),r_2(t),\cdots,r_n(t)]^T$，$\boldsymbol{r}(t)\leq\boldsymbol{k}$.

进程 p 的资源分配函数(RAF)可以分为两种：无延迟资源分配函数(NRAF)和可延迟资源分配函数(DRAF). 前者是进程 p 在零等待无延迟情况下的资源分配函数；后者是系统中存在多个并发的进程，进程 p 对资源的申请可能需要等待情况下的资源分配函数. 前者是后者一个特例.

定义单位阶跃函数 $u(t\geq0)=1,u(t<0)=0$. 由于进程 p 对资源的占用和释放均是

收稿日期 1994-06-03

在一系列离散时间点上进行的，有如下性质．

性质 1. 已知系统$(\Sigma,\boldsymbol{k})$. 以进程 p 首次占用资源的时间点作为时间零点，则进程 p 的无延迟资源分配函数 $\boldsymbol{r}(t)$可以表示为 $\boldsymbol{r}(t)=\sum_{i=0}^{m}\boldsymbol{a}_i u(t-t_i)$，其中 $t_0=0$，$t_j>t_{j-1}$，$j=1,2,\cdots,m$；向量 $\boldsymbol{a}_j\neq 0$ 而且 $0\leq\sum_{i=0}^{j}\boldsymbol{a}_i\leq\boldsymbol{k}$，$j=0,1,2,\cdots,m$；$\sum_{i=0}^{m}\boldsymbol{a}_i=0$.

性质 2. 已知进程 p 的无延迟资源分配函数 $\boldsymbol{r}(t)=\sum_{i=0}^{m}\boldsymbol{a}_i u(t-t_i)$，则它的可延迟资源分配函数 $\boldsymbol{s}(t)$ 可以表示为 $\boldsymbol{s}(t)=\sum_{i=0}^{m}.\,\boldsymbol{a}_i u(t-t_i')$，其中 $t_0'\geq 0$，$t_i'-t_{i-1}'\geq t_i-t_{i-1}$，$i=1,2,\cdots,m$.

性质 3. 如果 $\boldsymbol{s}_1(t)=\sum_{i=0}^{m_1}\boldsymbol{a}_i u(t-\alpha_i)$，$\boldsymbol{s}_2(t)=\sum_{i=0}^{m_2}\boldsymbol{b}_i u(t-\beta_i)$，则 $\boldsymbol{s}_1(t)+\boldsymbol{s}_2(t)$ 可以表示为 $\boldsymbol{s}_1(t)+\boldsymbol{s}_2(t)=\sum_{i=0}^{m_3}\boldsymbol{c}_i u(t-\gamma_i)$，其中 $\min(m_1,m_2)\leq m_3\leq m_1+m_2+1$.

定义 3. 已知资源分配函数 $\boldsymbol{s}(t)=\sum_{i=0}^{m}\boldsymbol{a}_i u(t-t_i)$，定义 $\boldsymbol{s}(t)$ 的第 j 个资源状态为 $\sum_{i=0}^{j}\boldsymbol{a}_i$，其持续时间 $\Delta_j(\boldsymbol{s}(t))=t_{j+1}-t_j$，$j=0,1,2,\cdots,m-1$. $\boldsymbol{s}(t)$ 总的持续时间定义为

$$\Delta(\boldsymbol{s}(t))=\sum_{j=0}^{m-1}\Delta_j(\boldsymbol{s}(t))=\sum_{j=0}^{m-1}(t_{j+1}-t_j)=t_m-t_0.$$

以系统总的加工持续时间最短为目标函数的避免死锁调度的数学模型为：设系统$(\Sigma,\boldsymbol{k})$中存在 n 个进程，其中第 j 个进程 p_j 的无延迟资源分配函数 $\boldsymbol{r}_j(t)=\sum_{k=0}^{m_j}\boldsymbol{a}_k^{(j)}u(t-\alpha_k^{(j)})$，$j=1,2,\cdots,n$. 要求 p_j 的可延迟资源分配函数 $\boldsymbol{s}_j(t)=\sum_{k=0}^{m_j}\boldsymbol{a}_k^{(j)}u(t-\beta_k^{(j)})$，使之满足 $\sum_{j=1}^{n}\boldsymbol{s}_j(t)\leq\boldsymbol{k}$(可行解条件) 和 $\Delta(\sum_{j=1}^{n}\boldsymbol{s}_j(t))=\mathrm{Min}$(最优解条件).

对于 $n=2$ 的情况可以通过如下的区域算法获得系统的最优解．

区域算法．

1) 构造平面直角坐标系，其中横轴 t_1 代表 $\boldsymbol{r}_1(t)$的各个资源状态 $\sum_{i=0}^{j}\boldsymbol{a}_i$ 的持续时间，纵轴 t_2 代表 $\boldsymbol{r}_2(t)$ 的各个资源状态 $\sum_{i=0}^{j}\boldsymbol{b}_i$ 的持续时间．两个坐标轴采用同样的比例尺．

2) 求出 $\boldsymbol{r}_1(t)$和 $\boldsymbol{r}_2(t)$的冲突区 $\mathrm{Conflict}(\boldsymbol{r}_1(t),\boldsymbol{r}_2(t))=\{P\mid\forall\ (t_1,t_2)\in P,\boldsymbol{r}_1(t_1)+\boldsymbol{r}_2(t_2)>\boldsymbol{k}\}$．具体求法是，考查 $\boldsymbol{r}_1(t)$的第 i 个资源状态 $\sum_{k=0}^{i}\boldsymbol{a}_k$ 与 $\boldsymbol{r}_2(t)$ 的第 j 个资源状态 $\sum_{i=0}^{j}\boldsymbol{b}_i$ 在坐标系中对应的小矩形 U_{ij}，如果 $\sum_{k=0}^{i}\boldsymbol{a}_k+\sum_{k=0}^{j}\boldsymbol{b}_k>\boldsymbol{k}$，则 $U_{ij}\subset(\boldsymbol{r}_1(t),\boldsymbol{r}_2(t))$.

3) 由冲突区求出死锁区 $\mathrm{Deadlock}(\boldsymbol{r}_1(t),\boldsymbol{r}_2(t))$．坐标系中两个相邻接冲突区的左下

角区域构成死锁区,如图 1 中的 D_1 和 D_2.

4) 两个进程的加工过程在坐标系中可以用水平线、垂直线和 45°线三种连续线段表示,从(0,0)开始到$(a_{m_1}^{(1)}, a_{m_2}^{(2)})$结束. 其中水平线表示 p_1 运行,p_2 等待; 垂直线表示 p_2 运行,p_1 等待;45°线表示 p_1 和 p_2 同时运行. 三种加工线段都不得进入冲突区和死锁区. 与其它解相比,最优解对应的加工路径中 45°线占的比例最大. 这样从原点出发, 只要有可能,就以 45°线向$(a_{m_1}^{(1)}, a_{m_2}^{(2)})$推进. 如果遇到冲突区或死锁区, 则改为沿冲突区或死锁区的下或左边界水平或垂直推进.

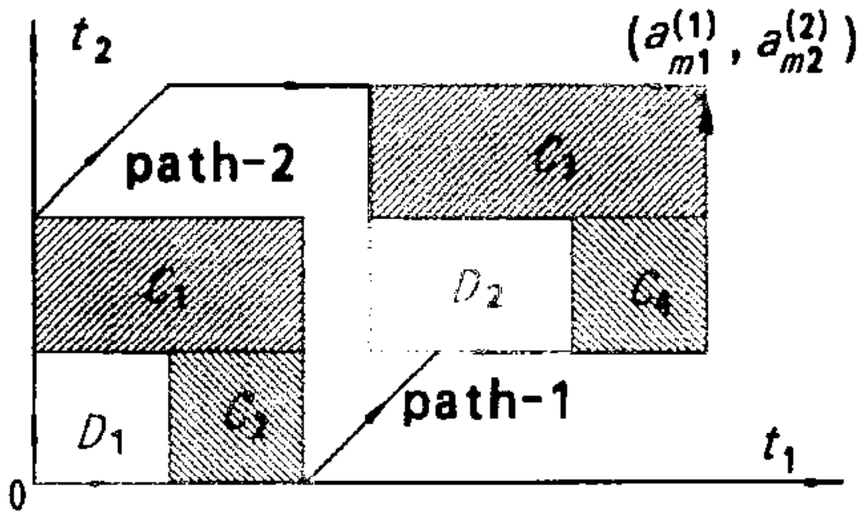

图 1　区域算法

5) 根据步骤 4)求出的最优路径, 求最优解 $\boldsymbol{s}_1(t)$和 $\boldsymbol{s}_2(t)$.

对于两进程的情况, 如果指定进程 p_1 较进程 p_2 有资源优先权,它可以不受阻碍地运行,即 $\boldsymbol{s}_1(t)=\boldsymbol{r}_1(t)$,则可以进一步将区域算法简化为一维算法的水池算法,其基本思想是,在保证 p_1 无延迟运行情况下,求出 p_2 的每个资源状态在时间区间$[0, a_{m_1}^{(1)}+a_{m_2}^{(2)})$上的允许区间集. 如果 p_2 能够从第一个资源状态连续地过渡到最后一个资源状态,并且每个资源状态的持续时间能够满足要求,则系统存在可行解,其中最先过渡到最后一个资源状态的路径对应系统的最优解. 这好比在 m_2 个水平面上分布着若干个水池. 每个水池能够向下一个水平面上与之相邻的水池排水. 从最上层水池经过中间各层水池最后流到最下层水池中的流水所形成的路径对应可行解.

水池算法.

1) 在时间$[0, a_{m_1}^{(1)}+a_{m_2}^{(2)})$上求出 p_2 的第 j 个资源状态 $\sum_{k=0}^{j}\boldsymbol{b}_k$ 的允许区间集 $F_j=\{[a, b)\,|\,b-a\geq\Delta_j(\boldsymbol{r}_2(t))\ \wedge\ \forall\, t\,[a,b), \boldsymbol{k}-\boldsymbol{r}_1(t)\geq\sum_{k=0}^{j}\boldsymbol{b}_k\}, j=0,1,2,\cdots,m_2-1$. 指定 F_j 中的元素从小到大排列.

2) 由允许区间集 F_j 求有效区间集 V_j, $j=0,1,2,\cdots,m_2-1$. 下面给出以 C 伪码形式描述的递推求解算法.

输入: 允许区间集序列　　$\{F_0, F_1, \cdots, F_{m_2-1}\}$;

　　持续时间序列　　$\{\Delta_0, \Delta_1, \cdots, \Delta_{m_2-1}\}$.

输出:有效区间集序列　　$\{V_0, V_1, \cdots, V_{m_2-1}\}$.

```
{ Let set V0=F0;
  for (i=1; i≤m2−1; i++)
  { let set Vi=φ;
    for (j=1,k=1; j≤|Vi−1|;j++)
    { get the j-th element [a,b) of set Vi−1;
      while (k≤|Fi|)
      { get the k-th element [c,d) of set Fi;
        if( [c,d)<[a+Δi−1,b))   {k++; continue;}
        if ( [c,d)>[a+Δi−1,b))   break;
        let [p,q)=[c,d)∩[a+Δi−1,b);
        if (q-p≥Δi)Vi=Vi∪{[p,q)};
```

```
        k++;    }    }    }
return the sequence of valid-interval sets   {V_0, V_1, ..., V_{m_2-1}}
```

3) 从 V_{m_2-1}开始回推最优解 $s_2(t)$. 有效区间集 V_{m_2-1}中的每个元素对应一个可行解，其中第一个元素$[a,b)$对应最优解．系统的最短持续时间 $\Delta(r_1(t)+s_2(t))=a+\Delta_{m_2-1}$. V_0, V_1,…,V_{m_2-1}中每一个不在可行解路径上的区间都必定对应一个死锁状态．

对于 $n\neq 2$ 的一般避免死锁调度问题，由于计算复杂度为 NP 难的，其最优解往往难于获得．下面给出一种求可行解的方法．

首先对 $r_1(t)$和 $r_2(t)$，由区域算法求出其最优解 $s_1(t)$和 $s_2(t)$. 然后令 $r_2'(t)=s_1(t)+s_2(t)$. 对 $r_2'(t)$和 $r_3(t)$，由区域算法求出其最优解 $s_2'(t)$和 $s_3(t)$. 再令 $r_3'(t)=s_2'(t)+s^3(t)$,…,直到最后对 $r_{n-1}'(t)$和 $r_n(t)$，由区域算法求出其最优解 $s_{n-1}'(t)$和 $s_n(t)$. 根据函数序列 $s_n(t)$, $s_{n-1}'(t)$, $s_{n-1}(t)$, $s_{n-2}'(t)$,…,$s_2'(t)$, $s_2(t)$, $s_1(t)$可以构造出系统的一个可行解.

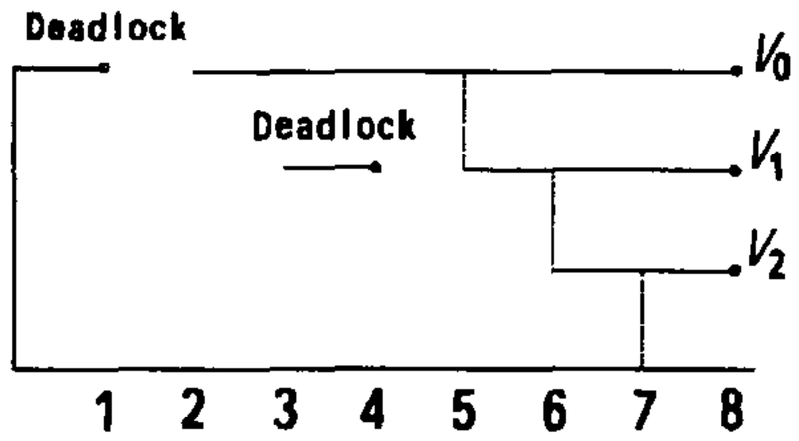

图 2 水池算法

例. 已知系统(Σ, k),$k=[1,1,1]^T$. 进程 p_1 和 p_2 的无延迟资源分配函数分别为

$$r_1(t)=\begin{bmatrix}1\\0\\0\end{bmatrix}u(t)+\begin{bmatrix}0\\1\\0\end{bmatrix}u(t-1)+\begin{bmatrix}-1\\-1\\0\end{bmatrix}u(t-2)+\begin{bmatrix}0\\0\\1\end{bmatrix}u(t-2.5)+\begin{bmatrix}1\\0\\0\end{bmatrix}u(t-4)+\begin{bmatrix}-1\\0\\-1\end{bmatrix}u(t-5),$$

$$r_2(t)=\begin{bmatrix}0\\1\\0\end{bmatrix}u(t)+\begin{bmatrix}1\\0\\0\end{bmatrix}u(t-1)+\begin{bmatrix}-1\\-1\\1\end{bmatrix}u(t-2)+\begin{bmatrix}0\\0\\-1\end{bmatrix}u(t-3).$$

图 1 是由区域算法求出的两条最优路径，其解析式分别为

$$\begin{cases}s_1(t)=r_1(t)\\ s_2(t)=\begin{bmatrix}0\\1\\0\end{bmatrix}u(t-2)+\begin{bmatrix}1\\0\\0\end{bmatrix}u(t-5)+\begin{bmatrix}-1\\-1\\1\end{bmatrix}u(t-6)+\begin{bmatrix}0\\0\\-1\end{bmatrix}u(t-7),\end{cases}\tag{1}$$

$$\begin{cases}s_1(t)=\begin{bmatrix}1\\0\\0\end{bmatrix}u(t-2)+\begin{bmatrix}0\\1\\0\end{bmatrix}u(t-3)+\begin{bmatrix}-1\\-1\\0\end{bmatrix}u(t-4)+\begin{bmatrix}0\\0\\1\end{bmatrix}u(t-4.5)\\ \qquad\quad+\begin{bmatrix}1\\0\\0\end{bmatrix}u(t-6)+\begin{bmatrix}-1\\0\\-1\end{bmatrix}u(t-7).\\ s_2(t)=r_2(t)\end{cases}\tag{2}$$

系统的最短持续时间 $\Delta(s_1(t)+s_2(t))=\min=7$. 图 2 是由水池算法求出的最优路径，其

解析式同上面的(1)式.

参　考　文　献

[1] Peterson J L, silberschatz A. Operating System Concepts, second edition, Addison-Wesley Publishing Company, Inc., 1985:271—306.

[2] Diaz M. Modelling and analysis of communication and cooperation protocols using petri net based models. *Computer Networks*, 1982, **6**(6):419—441.

[3] Banaszak Z A, Krogh B H. Deadlock avoidance in flexible manufacturing systems with concurrently competing process flows. *IEEE Trans. Robotics Automat.*, 1990, **6**(6):724—734.

[4] Viswanadham N, Narahari Y, Johnson T L. Deadlock prevention and deadlock avoidance in flexible manufacturing systems using petri net models. *IEEE Trans. Robotics & Automat.*, 1990, **6**(6):713—723.

MATHEMATICAL MODELING AND SOLUTION OF SCHEDULING FOR DEADLOCK AVOIDANCE

YANG SHENG　　WU CHENG

(*Department of Automation, Tsinghua University, Beijing* 100084)

Abstract　This paper presents the mathematical model of optimal scheduling for deadlock avoidance in manufacturing ysstems based on the conception of resource allocation function, aimed at reducing the system total processing time. Two algorithms are proposed for the optimal solution in the case of two processes and the feasible solution in the case of multi-processes, respectively.

Key words　Deadlock, scheduling, manufacturing system.