基于Petri网的RGVs系统中环路死锁研究

吴长庆 罗 键 陈火国 庄进发 彭彦卿
(厦门大学信息科学与技术学院 厦门 361005)

摘 要 为实现自动小车存取系统的实时控制,基于双重着色赋时Petri网(Colored Timed Petri Nets,CTPN)构建了RGVs系统(Rail-Guided Vehicles system,RGVs)的动态模型。同时为了提高RGVs系统的存储效率,对RGV小车采用基于最短路径的调度策略。并针对RGVs系统的临界状态即将发生环路(环路链)死锁的状况,提出了一种死锁预防的方法。最后基于VC.NET验证其有效性。

关键词 自动小车存取系统,RGVs系统,环路死锁,Petri网
中图不分类号 TP273 **文献标识码** A

Cycle-deadlock Control of Rail Guided Vehicles Systems via Petri Nets

WU Chang-qing LUO Jian CHEN Huo-guo ZHUANG Jin-fa PENG Yan-qing
(School of Information Science and Technology, Xiamen University, Xiamen 361005, China)

Abstract In order to implement the real time control of the Rail-Guided Vehicles systems, a deadlock control modeling method via Dual Colored Timed Petri Nets was proposed. Moreover, the scheduling strategy for the RGVs system based on shortest path method was addressed to improve the efficiency of storage and retrieval. Then, the critical state in deadlock free was identified and FCFS policy was applied to solve it. Finally, experimental results verified the effectiveness of the policies.

Keywords Autonomous vehicle storage and retrieval systems, Rail-guided vehicles systems, Cycle-deadlock, Petri nets

1 引言

自动化立体仓库(automated storage and retrieval systems)是现代物流技术领域内一种仓储方式。它是以高层货架为主体,以成套搬运设备为基础,以计算机控制技术为手段的高效率、大容量存储的机电一体化集成系统,目前已经逐渐向"3I"(Intelligent, Integrated, Information)仓库系统过渡。如本文研究的自动小车存取系统(Autonomous Vehicle Storage and Retrieval Systems, AVS/RS)就是一种新型货物存取系统。作为AS/RS的一种可替代的系统[1],它在各个方面都体现出了更强的性能,2000年后在欧洲等发达国家开始逐渐受到关注。AVS/RS主要包括轨道导引小车系统(Rail-Guided Vehicles system, RGVs)、升降机系统和货架以及货物缓存区,如图1[2]所示。

在AVS/RS中最主要的控制系统是RGVs系统,提高其运行效率对于提高AVS/RS的出入库效率、系统出入库的正常有序进行有着至关重要的意义。RGVs系统是属于典型的离散事件动态系统(DEDS),对于这类系统动态模型,Petri网已经证明其有效性。田国会等人把面向对象技术引入其中,提出了采用面向对象方法的着色Petri网(Oriented Object Colored Petri Nets, OOCPN)来构建AS/RS模型[3,4];意大利的Dotoli和Fanti等把时间这一要素引入到着色Petri网中,以堆垛机和小车作为托肯,任务路径作为颜色,从而构建了AS/RS的动态模型[5];郝扬和古天龙利用着色Petri网来解决Internet电话端系统业务的冲突检测[6]。不论采用着色或者面向对象的方法,其主要目的都是为了简化Petri网模型,而在这方面着色方法表现得更有优势。本文在完善文献[7]的理论基础上,应用着色赋时Petri网构建了RGVs系统的动态模型,同时为了提高RGVs系统的存储效率,对RGV小车采用基于最短路径的调度策略。针对RGVs系统的临界状态,即将发生环路死锁的状况,提出了一种死锁预防的方法。

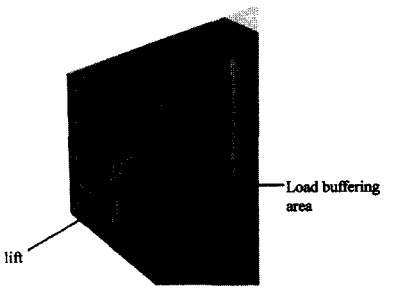

图1 自动存取系统结构简图

2 基于双重着色的赋时Petri网的RGVs系统动态模型[7]

本文是利用双重着色的赋时Petri网来构建RGVs系统

到稿日期:2008-05-13 本文得到厦门大学985二期信息创新平台项目资助。
吴长庆 博士研究生,研究方向为物流自动化系统,E-mail:king7779@sina.com.cn;罗 键 教授,博士生导师,研究方向为计算机集成制造系统和物流自动化系统;陈火国 硕士研究生,研究方向为MES的控制与优化;庄进发 博士研究生;彭彦卿 博士研究生。

的动态模型，关于建模的详细过程在文献[7]已做了详细的说明。如图2所示，库所 P_1 和 P_2 表示 RGV 小车运行区域，库所 P_1 和 P_2 中的托肯表示 RGV 小车(空闲、载货和指令下达状态)。

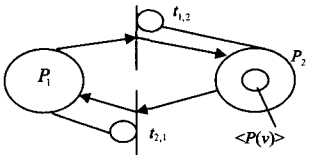

图2 RGV 小车的 CTPN 模型

图2描述了 RGV 小车输送的 CTPN 模型，并且托肯着色如下：

1) 如果 RGV 小车 v 处于“任务下达”状态，如入库时，则颜色域为 $pp(v)=(p_1,p_2)$，表示 RGV 小车从当前的位置 p_1 到接货位置 p_2 需要经过的路径；

2) 如果 RGV 小车处于“空闲”状态，则颜色域为 $pp(v)=(p_i)$，其中 $i=10$ 或者 $i=11$，表示 RGV 小车在当前的位置 p_i 等待接受任务状态；

3) 如果 RGV 小车处于“载货”状态，则它的颜色域 $pp(v)$ 和所载的任务颜色域相同。

实际建模过程中，我们使用着色托肯 $\langle pp(j)\rangle$ 来描述每个任务 $j\in J$，而 RGV 的状态则由 CTPN 的标志 M 来表示。具体表示如下：

1) $M(p_i)=\langle pp(v)\rangle$，表示 RGV 小车 v 在库所 p_i 上，并且处于“指令下达”状态；

2) $M(p_i)=\langle pp(j)\rangle$，表示 RGV 小车 v 在库所 p_i 上并处于“载货”状态，所载任务为 j；

3) $M(p_i)=\langle p_i\rangle$，表示 RGV 小车在库所 p_i 上，状态为空闲；

4) $M(p_i)=\langle 0\rangle$，表示库所 p_i 上没有 RGV 小车。

初始状态 M_0 定义如下：如果 $pp(v)=p_i$，那么 $M_0(p_i)=\langle p_i\rangle$，否则 $M_0(p_i)=\langle 0\rangle$。

假设在 τ 时刻，托肯 v 到达库所 p_i，经过 $\delta(p_1)$ 时间间隔后，托肯 v 的时间戳 $s(v)=\delta(p_i)$，这时变迁 $t_{i,m}$ 在 $\tau+\delta(p_i)$ 时刻并且在标志 M 状态下准备好使能。对于每一变迁 $t_{i,m}\in T_{i,m}$，其使能条件为：

1) $M(p_m)=0$；

2) $\lambda(p_i)\geqslant ti(p_i)$；

3) $M(p_i)\geqslant C^-(p_i,t_{i,m})(c)=\langle p_i,p_m\rangle$。

如果 $t_{i,m}\in T_{i,m}$ 使能，那么新的状态标志 M' 计算公式如下：

$$M'(p_m)=C^+(p_m,t_{i,m})(c)+M(p_m)=\langle p_m,\cdots\rangle$$

$$M'(p_i)=0$$

如果 $P_1\subset P$，$P_1=\{p_1,p_2,\cdots,p_n\}$，$n\in N$，$p_i\in P_1$ 且 $Co(p_i)=\{\pi_1,\pi_2,\cdots,\pi_u\}$，那么 P_1 的颜色 $Co(p_1)$ 为 $\langle p_1,p_2,\cdots,p_m\rangle$，则称之为双重着色。

一般，国内外学者都采用随机 Petri 网(SPN)和着色 Petri 网(CPN)来构建 AS/RS 模型。然而它是属于一种离散事件的模型，在对此模型进行性能评价时，时间往往是最关键的因素，因此可在此基础上增加时间变量[8]，从而形成赋时 Petri 网。另一方面，着色 Petri 网通过增加颜色属性扩展了 Petri 网元组，引入颜色属性简化了那些具有相同或者类似结构的系统(Petri 网图形)。在宏观结构方面，AVS/RS 每层的货架结构是相同的，应用着色 Petri 网分析方法可以构建出相对简单直观的模型；然而从微观角度考虑，为了描述 AVS/RS 每个节点小车通过的情况，有必要为这些节点增加相应的颜色属性。本文从模型的简单化和可计算化的角度出发，利用双重着色的赋时 Petri 网建立 RGVs 系统的模型，主要在于把性质和属性相似的站台或者货位作为 Petri 网中的颜色集之一，而把站台上运行托盘路径作为另外一种颜色集。这样，既考虑了着色 Petri 自身理论方法适用性，也充分考虑了实际应用中信息的整体性。

3 RGVs 系统中环路死锁及其避免策略

3.1 基于最短路径的 RGV 小车的调度

在该系统中，RGV 小车的调度策略采用最近原则，即判断哪个有任务的站台距离 RGV 小车最近，就优先选择哪辆空闲小车。具体步骤如下：当有出库或者入库任务产生后，系统开始搜索访问站台信息表中 RGV 小车、出库、入库、退库站台的信息，找出有货物的站台位置；通过计算路径表，基于最短路径原理搜索出对应的 RGV 小车来完成存取任务，若此时刚好有退库任务且其最短路径相等，则取退库任务优先；分别标示货物与 RGV 小车的位置 ID，RGV 小车接到下达的任务开始进入存取系统，结束。RGV 小车最短路径的计算是根据 Dijkstra 算法的原理，同时由图中各节点的入度信息来提高算法的搜索速度。首先计算赋权有向图中各个节点的入度，若图中其他节点到该节点的最短路径都为无穷或者不可达，则可以删除该节点以简化路径的搜索。

3.2 RGVs 系统环路(环路链)的识别

RGVs 系统是属于单容量资源(unit capacity resources)控制系统，每个位置资源容量都为1，不能互为缓冲区域，这区别于 Fanti 和吴乃骐等人研究的多容量控制系统。因为多容量库所本身可以作为缓冲区域，当有多辆小车作业时，有利于避免死锁[5,9]。在多容量控制系统中形成的是回路死锁，它是由多个资源组成的；而在单容量控制系统中形成的是环路死锁。环路虽是回路的特例，但是由多个环路组成环路链后，当系统进入到环路的临界状态，其死锁避免控制的方式与回路死锁控制就有很大的区别，故而对于这样的系统需要进一步分析。

定义1 有向图 $G=(V,E)$，V 为图 G 所有顶点的非空集合，而 E 为 G 所有有向弧的集合。如果 G 中有一路径从 p_i 到 p_j，$p_i,p_j\in V$，那么称 p_i 到 p_j 可达；如果从 $p_i(p_j)$ 到 $p_j(p_i)$ 可达，那么称 p_i 和 p_j 相互可达。

定义2 如果有向图 G 中两相邻的顶点 $p_i,p_j\in V$ 相互可达，那么由 p_i 和 p_j 构成的子图 $G_C=(V_C,E_C)$ 称为环路。

定义3 如果有向图 G 中 p_1 和 p_2，p_2 和 p_3，$\cdots$，p_{n-1} 和 $p_n(p_1,p_2,\cdots,p_n\in V,n>2)$ 相互可达，那么这些顶点构成的子图 $G_{CC}=(V_{CC},E_{CC})$ 称为环路链。

定义4 给定路径图 $G_R=(V_R,E_R)$，对于任意环路 G_C(或环路链 G_{CC})，如果其输入库所 p_i,p_j 中存在 $M(p_i)=\pi_{k1}$，$M(p_j)=\pi_{k2}$，k_1 和 k_2 为 RGV 编号，其中 $\pi_{k1}\in d^+(G_C)$，$\pi_{k2}\in d^-(G_C)$(或 $\pi_{k1}\in d^+(G_{CC})$，$\pi_{k2}\in d^-(G_{CC})$)，那么称为环路(或环路链)临界状态。

如图 3 所示，当小车 V_2 由库所 p_9 到达库所 p_8，那么 CTPN 模型的标志为 $M(p_L)=\pi_1$，$M(p_{20})=\pi_3$，$M(p_8)=\pi_2$，由于 $N_1^+=0$，$N_1^-=0$，$N_2^+=0$，$N_2^-=0$，因此输入变迁 $t_{L,1}$，$t_{8,7}$ 可以同时使能。一旦这些变迁激发，小车 V_1 和 V_2 同时进入环路链 G_{C3}，这样 V_1 和 V_2 必然在环路链 G_{C3} 上进入死锁状态，此时达到系统的临界状态。

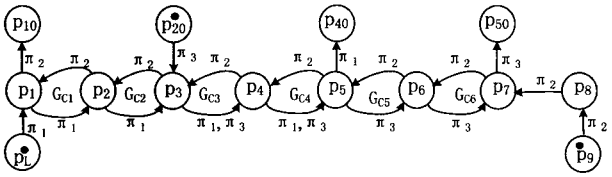

图 3　某层 RGVs 系统环路临界状态简化图

RGV 路径的环路(环路链)识别算法：

第一步　产生路径 path1，path2，…，path$n$，完成初始化。

第二步　查找任意两条路径，判断两者是否有相匹配的路径。如果是，则返回它们各自的输入库所或者相同路径的首个站台。如此循环，直至所有路径搜索完成，跳出循环；如果否，则两条路径互不关联，转到第六步。

第三步　判断两条路径是否形成环路或环路链。如果是，将其保存到数据库表中，转到下一步；如果否，转第六步。

第四步　判断当前的变迁是否属于输入变迁。如果属于，判断其所在的环路或者环路链上是否处于临界状态；如果是，转入第五步；如果否，转入第六步。

第五步　根据下面提出的环路和环路链的临界状态控制策略，选择任务路径生成时间最早的小车前行，结束操作。

第六步　判断相应的输入变迁使能条件。

第七步　如果 RGVs 的路径图更新，那么返回第一步；否则返回第四步。

环路(或环路链)临界状态控制策略：设 $t(\pi_{k1})$，$t(\pi_{k2})$ 为环路(或环路链)临界状态路径 π_{k1}，π_{k2} 的生成时间，如果 $t(\pi_{k1})>t(\pi_{k2})$，那么颜色 π_{k1} 对应的输入变迁首先获得使能标志。根据临界状态控制策略，那么 $\min(t(\pi_1),t(\pi_2))$ 输入变迁应首先获得使能标志。

4　基于.NET 的 RGVs 系统模型实现

4.1　两条路径构成的环路(环路链)识别

函数：CString GetSamePath(CString m_path1，CString m_path2)

参数说明：m_path1，m_path2 表示产生的任意两条路径。

函数伪代码如下：

```
CString GetSamePath (CString m_path1,CString m_path2) //查找 path1 路径的各个站台,按顺序存放在数组 path1[k]中。Path1length 表示数组 path1[k]的长度,置 Path1length=数组长度; 查找 path2 路径的各个站台,按顺序存放在数组 path2[k]中。Path2length 表示数组 path2[k]的长度,置 Path2length=数组长度。
int i=0;
CString Samepath=" ";//存放返回的环路(环路链)路径
Int FrongPort,BackPort; // 分别存放前输入库所,后输入库所
 while(i<path1length)
   {
     j=0;
     while(j<path2length)
       {
         if(Path1[i]==Path2[j])
         //查找到相同的站台
           {
             m=i+1,n=j+1;
    if((m<path1length)&&(n<path2length)&&(Path1[m]==Path2[n])) //表明具有冲突或相同的路径部分,并查找相同路径和输入库所
{
while((m<path1length)&&(n<path2length)&&(Path1[m]==Path2[n])) //按顺序比较下一站台是否相同
    {
        m++;
            n++
     }
for(k=i;k<m;k++) //合并相同的路径
    {
        Samepath= Samepath +Path1[k];
}
FrontPort=Path1[i-1]; //取出前输入库所
BackPort=Path2[j-1]; //取出后输入库所
Samepath =Path1[i]; //对冲突或路径相同部分的部分,只取第一个站台
Samepath=FrontPort+"-"+BackPort+"-"+Samepath;
    return Samepath; //返回前后输入库所和相同路径部分
}
else //如果正向没有相同的路径部分,另一路径按相反方向查找,查询两条路径间是否构成环路或环路链
{
    m=i+1,n=j-1;
if((m<path1length)&&(0<=n<path2length)&&(Path1[m]==Path2[n]))
{
while((m<path1length)&&(0<n<path2length)&&(Path1[m]==Path2[n])) //查询环路(环路链)站台
    {
        m++;
            n--;
    }
for(k=i;k<m;k++) //合并环路(环路链)站台
     {
    Samepath = Samepath +Path1[k];
    }
  FrontPort=Path1[i-1]; //取前输入库所
BackPort=Path2[n];   //取后输入库所
Samepath=FrontPort+"-"+BackPort+"-"+Same path;
  return Samepath; //返回前后输入库所和环路(环路链)
                }
              }
         else
         {
           j++;
         }
    }
    i++;
}
```

函数解析：首先将 $path1$，$path2$ 两条路径的站台按顺序存放在数组 $path1[k]$，$path2[k]$中。对于 $path1[k]$中的值，取 $path2[k]$中的值逐一跟它比较。如此循环，直至找到 $path1[i]==path2[j]$为止。然后，分两种情况进行讨论：第一种情况是两条路径具有冲突或相同的路径部分，例如路径 $path1=(p_1,p_2,p_3,p_4,p_5,p_6,p_7,p_{41},p_{42})$，$path2=(p_{21},p_3,p_4,p_5,p_6,p_7,p_8,p_9)$，则当程序找到 $path1[2]=path2[1]=p_3$ 时，将各自数组指针加 1，按顺序继续查找，比较各自的下一站台，查找到 $path1[3]=path2[2]=p_4$，如此循环，直至查找到 $path1[7]=p_{41}$，$path2[6]=p_8$，$path1[7]\neq path2[6]$，跳出循环。接着，将查找到的站台合并，对于具有相同路径部分的站台，只需要取得其首个站台，避免其发生冲突即可。再取出输入库所，得到其返回值 $Samepath=p_2-p_{21}-p_3$。其中第一个站台 p_2 表示 $path1$ 的输入库所，第二个站台 p_{21} 表示 $path2$ 的输入库所，第三个站台 p_3 是路径 $path1$ 和路径 $path2$ 构成的具有相同路径部分的首个站台。第二种情况是两条路径具有共同的路径部分，但路径方向相反，也就是两条路径构成环路或者环路链。例如 $path1=(p_1,p_2,p_3,p_4,p_5,p_6,p_7,p_8,p_9,p_{L2})$，$path2=(p_{41},p_7,p_6,p_5,p_4,p_3,p_{21},p_{22})$，则当程序找到 $path1[2]=path2[5]=p_3$ 时，将各自数组指针加 1，按顺序继续查找，比较各自的下一站台，但是 $path1[3]=p_4$，$path2[6]=p_{21}$，$path1[3]\neq path2[6]$。针对这种情况，有可能是两条路径间构成环路(环路链)，因此可将 $path1[k]$的数组指针加 1，$path2[k]$的数组指针减 1，再比较二者是否相同。如本例中，$path1[3]=path2[4]=p_4$，如此循环，直至找到 $path1[7]=p_8$，$path2[0]=p_{41}$，$path1[7]\neq path2[0]$，跳出循环。接着，将查找到的站台合并，取出输入库所，返回 $Samepath=p_2-p_{41}-p_3-p_4-p_5-p_6-p_7$。其中第一个站台 p_2 表示 $path1$ 的输入库所，第二个站台 p_{41} 表示 $path2$ 的输入库所，$p_3-p_4-p_5-p_6-p_7$ 是路径 $path1$ 和路径 $path2$ 构成的环路(环路链)。

4.2 多条路径间的环路(环路链)识别

对于 RGVs 系统而言，一般都会有多条任务路径同时在执行，因此绝大部分情况下环路(环路链)的识别是在多条路径间进行的。对于多条路径间的环路(环路链)识别，首先查找系统中所有正在执行的任务并存放在数组中，然后对数组中的两两任务路径进行识别。可调用 4.1 节中的两条路径间的环路(环路链)识别方法，即函数 CString GetSamePath(CString m_path1,CString m_path2)，返回得到输入库所和环路(环路链)路径，并写入到数据库中。

函数：void GetCirclePath()

函数功能：对于任务表中所有正在执行的任务，按照它们的路径图计算出环路(环路链)，并将其相关信息写进到数据库表 Collide_Path 里。

```
void GetCirclePath()
{
  从任务表中取出所有 Task_status=1 的任务存放在 Path[k]，任务对应的 Token 存放在 TokenID[k]里。
   int max=k; //k 为数组长度
   int i=k=1;
 while(i<max)
   {
      j=i+1;
   while(j<max) //对于数组中的所有路径，进行两两比较
{
      path1=Path[i];
      path2=Path[j];
SamePath=GetSamePath(path1,path2); //调用两条路径间的环路(环路链)识别方法：
   if(SamePath! ="")
   {
     取 FrontPort，BackPort，取 FrontPort，BackPort 对应的 RGV 小车，并将 FrontPort，BackPort 以及对应的 RGV 小车和冲突或环路路径写入到数据库表 Collide_Path 里
    }
        j++;
   }
   i++;
   }
}
```

函数解析：对于所有正在执行的任务，首先将其所有的任务路径信息取出来，存放在数组中。并设置两个变量 i，j，将数组中的路径两两组合。对于任意的两条路径，可调用 4.1 节中的两条路径间的环路(环路链)识别方法，即函数 CString GetSamePath(CString m_path1，CString m_path2)。如果返回值为空，表明这两条路径互不关联；否则，函数将返回前后输入库所 FrontPort，BackPort 以及环路(环路链)路径。最后，分别取前后输入库所的值以及它们对应的小车和环路(环路链)路径等信息，存放到数据库表 Collide_Path 里，如图 4、图 5 所示。

图 4 RGV 产生的任务路径图

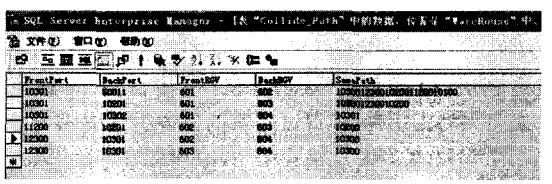

图 5 多台小车形成的环路(环路链)

4.3 小车环路条件判断

完成路径间的环路(环路链)识别后，必须判断小车移入到下一位置时是否会造成死锁现象的发生。根据文献[7]提出的系统无死锁条件，下面函数实现小车环路条件判断。

函数：BOOL JudgeRGVCondiction(int RGVToken_ID)

参数说明：RGVToken_ID，指定的 RGV 小车。

函数功能：该函数主要用于判断如果 RGV 小车移动到下一站台时，是否会造成系统的冲突或环路(环路链)死锁。如果结果返回值 0，则表示不会造成冲突或死锁，RGV 小车可以移动到下一站台；如果返回值 1，则表明造成冲突或环路、环路链的站台里有托肯存在，RGV 小车不可以移动到下

(下转第 260 页)

本文	BER	0.0012	0	0	0.502	0.0549	0.4792	0.4631
算法	PSNR	41.6832	28.9861	26.8374	22.9996	37.7097	26.2881	26.3525

		TSM (+1%)	TSM (−1%)	TSM (−2%)	TSM (−3%)	TSM (−4%)	抖动攻击 (1/100000)	抖动攻击 (1/50000)
本文	BER	0.4395	0.4543	0.4463	0.4597	0.4348	0.0007	0.0234
算法	PSNR	26.7864	27.0071	27.1599	26.7324	26.6203	35.3953	31.3474

结束语 针对现有量化调制音频水印所存在的问题，本文结合音频统计均值稳定特性及同步码技术，提出了一种新的数字音频水印嵌入算法，该算法选取稳健的16位巴克码作为同步标记，通过量化音频样本统计均值嵌入同步码，同时结合听觉掩蔽特性量化低频小波系数平均值嵌入数字水印。仿真实验表明，该算法不仅具有较好的不可感知性，而且对常规信号处理、部分去同步攻击均具有较好的鲁棒性。

参 考 文 献

[1] Cox I J, Matthew L M, Jeffrey A B, et al. Digital watermarking and steganography. Second Edition. Burlington, MA, Morgan Kaufmann Publishers (Elsevier), 2007

[2] 李伟，袁一群，李晓强，等. 数字音频水印技术综述[J]. 通信学报，2005，26(2)：100-111

[3] Barni M, Cox I J, Kalker T. Digital watermarking // Proceedings of 4th International Workshop on Digital Watermarking (IWDW). Siena, Italy, 2005：15-19

[4] 孙圣和，陆哲明，牛夏牧. 数字水印技术及应用[M]. 北京：科学出版社，2004

[5] Tu Ronghui, Zhao Jiying. A novel semi-fragile audio watermarking scheme // IEEE International Workshop on Haptic, Audio and Visual Environments and their Applications. Ottawa, Ontario, Canada, September 2003：89-94

[6] 王秋生，孙圣和，郑为民. 数字音频信号的脆弱水印嵌入算法[J]. 计算机学报，2002，25(5)：520-525

[7] Miyazaki A, Okamoto A. Analysis of watermarking systems in the frequency domain and its application to design of robust watermarking systems // Proceedings of IEEE International Conference on Image Processing. Vol. 2, Oct. 2001：506-509

[8] 王让定，徐达文. 基于提升小波的多重数字音频水印[J]. 电子与信息学报，2006，28(10)：1820-1825

[9] Xu Zhao, Wang Ke, Qiao Xiao-hua. Digital audio watermarking algorithm based on quantizing coefficients // 2006 International Conference on Intelligent Information Hiding and Multimedia. 2006：41-46

[10] Xiang Shijun, Huang Jiwu. Analysis of D/A and A/D Conversions in quantization-based audio Watermarking. International Journal of Network Security, 2006, 3(3)：230-238

(上接第253页)

一站台，必须在该站台等待。

函数解析：首先根据站台上的托盘信息查找具体RGV小车编号，然后在FrontRGV，BackRGV输入库所里查找该RGV小车，并将其输入库所，相同路径或环路、环路链的路径存放在数组里。若数组为空，则表明没有与该RGV小车相同或冲突的路径，返回值0；否则，对于数组里的每一条记录做如下处理：查找小车当前所在站台，判断当前站台是否是输入库所。如果否，则当前小车可以进入；如果是，对于该输入库所的冲突或环路(环路链)路径，查找其路径站台上是否有托肯。如果有，则小车不可进入，返回1；如果没有，则继续查询下一条记录，直至记录里的所有值都没有，返回值0。

结束语 本文应用双重着色赋时Petri网构建了RGVs系统的动态模型。为避免RGVs发生路径死锁，探讨了其临界状态，即将发生环路(环路链)死锁的状况，实现了RGV小车环路(环路链)的识别，并提出了可靠的无环路死锁的控制策略。最后通过实验验证了其有效性。本文研究的是AVS/RS中同一层RGVs系统环路死锁的问题，对于不同层、多层RGV路径以及升降机路径所形成的混杂环路死锁必将更加复杂，这是今后需要继续完善的工作。

参 考 文 献

[1] Zizi D V. Whats New in the Equipment Field, 2000 International Material Handling Research Colloquium. Material Handling Institue, York, Pennsylvania, 2000

[2] Malmborg C J. Conceptualizing tools for autonomous vehicle storage and retrieval systems [J]. International Journal of Production Research, 2002, 40(8)：1807-1822

[3] 田国会，刘长友，徐心和. 自动化仓库输送过程调度问题研究[J]. 计算机集成制造系统，1998，4(2)：51-54

[4] 常发亮，刘长友. 自动化立体仓库输送系统调度的优化仿真及其应用研究[J]. 系统仿真学报，1998，10(5)：14-19

[5] Dotoli M, Fanti M P. Modeling of an AS/RS Serviced by Rail-Guided Vehicles with Colored Petri Nets：a Control Perspective [C]//Proceedings of the 2002 IEEE International Conference on Systems, Man and Cybernetics. Hammamet, Tunisia, 2002：162-167

[6] 郝扬，古天龙. 基于着色Petri网的Internet电话端系统业务冲突检测[J]. 计算机科学，2007，34(12)：41-45

[7] He S J, Cheng F, Luo J. Modeling and Implementing of an Automated Warehouse via Colored Timed Petri Nets-a Behavior Perspective[C]// Proceedings of the IEEE International Conference on Control and Automation. New York：Institute of Electrical and Electronics Engineers Inc., 2007：2823-2828

[8] 李鹏，李勋，顾庆，等. TCPN的组合可调度分析[J]. 计算机科学，2008，5(1)：290-293

[9] Wu N Q, Zhou M C. Deadlock modeling and control of automated guided vehicle systems[J]. IEEE ASME Trans. Mechatronics, 2004, 9(1)：50-57