计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2023,32(11):95−107 [doi: 10.15888/j.cnki.csa.009281]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

面向资源 Petri 网的自动制造系统死锁预防①

卢雪芹, 刘　伟

(山东科技大学 计算机科学与工程学院, 青岛 266590)
通信作者: 刘　伟, E-mail: liuwei_doctor@yeah.net

摘　要: 在自动制造系统 (automated manufacturing systems, AMSs) 中, 死锁是一个急需解决的问题, 其主要由资源的循环等待造成. 为了解决该问题, 本文首先基于面向资源 Petri 网 (resource-oriented Petri nets, ROPNs) 的特征, 建立特殊资源标记图 (special resource marked graphs, SRMGs). 其次, 在 SRMGs 中建立死锁与饱和回路之间的关系. 最后通过为一些特殊回路添加控制器, 阻止系统出现不安全标记. 考虑到资源故障问题, 为危险库所添加资源缓冲子网, 保证需要故障资源的零件不会阻塞其他零件的持续生产. 相比现有的控制器, 本文的监督控制器具有控制开关, 其通过实时改变控制库所的容量可以允许更多安全标记发生.

关键词: 自动制造系统; 面向资源 Petri 网; 资源故障; 饱和回路; 危险库所

引用格式: 卢雪芹,刘伟.面向资源 Petri 网的自动制造系统死锁预防.计算机系统应用,2023,32(11):95–107. http://www.c-s-a.org.cn/1003-3254/9281.html

Deadlock Prevention in Automated Manufacturing Systems Based on Resource-oriented Petri Nets

LU Xue-Qin, LIU Wei

(College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China)

Abstract: In automated manufacturing systems (AMSs), deadlock is an urgent problem to be solved, which is mainly caused by circular waiting for resources. To solve this problem, this study first builds special resource marked graphs (SRMGs) based on the characteristics of resource-oriented Petri nets (ROPNs). Secondly, the relationship between a deadlock and the saturated circuit is established in SRMGs. Unsafe markings can be prevented by adding controllers to some special circuits. Next, considering the problem of resource failure, the resource buffer subnet is added to the hazardous place. This ensures that parts requiring failed resources do not block the continuous production of other parts. Compared with existing controllers, each controller in this study has a control switch that allows more safe markings to occur by changing the capacity of the control place in real time.

Key words: automated manufacturing systems (AMSs); resource-oriented Petri nets (ROPNs); resource failure; saturated circuit; hazardous places

自动制造系统 (automated manufacturing systems, AMSs) 用一组有限的资源同时处理多种零件. 因此不适当的资源分配将导致系统出现循环等待, 带来巨大的经济损失[1–4]. AMSs 中可能出现故障的资源称为不可靠资源. 当不可靠资源故障时, 需要故障资源的零件无法继续加工. 此外, 资源故障可能造成系统阻塞, 导致不需要故障资源的零件也无法进行加工. 因此, 需要为 AMSs 开发有效的死锁控制策略, 以阻止系统出现死锁和阻塞状态[5–11].

目前广泛使用的死锁控制策略之一是死锁预防,

① 基金项目: 2022 年度青岛市社会科学规划研究项目 (QDSKL2201131)
收稿时间: 2023-04-20; 修改时间: 2023-05-17; 采用时间: 2023-05-23; csa 在线出版时间: 2023-09-15
CNKI 网络首发时间: 2023-09-19

其主要应用以下两种技术预防死锁. 一种是可达性分析[12−14], 该方式可以使系统达到最大许可性. 然而, 它存在状态爆炸的问题. 另一种是结构分析[15−19], 分为以下两种方法. 基于虹吸的方法通过预防标记不足的虹吸解决死锁问题[15,16]. 基于资源回路的方法通过阻止回路饱和解决死锁问题[17−19]. 对于各类复杂的 AMSs, 研究人员分别通过最大完美资源变迁 (maximal perfect resource transition, MPRT) 回路[18,19], 最大完美控制变迁 (maximal perfect control transition, MPCT) 回路[20], 最大关键资源变迁 (maximal critical resource transition, MCRT) 回路[21], 完美活动 (perfect activity, PA) 回路[22,23] 等解决死锁问题.

Li 等基于资源回路的概念划分 Petri 网 (Petri nets, PNs), 通过为划分后的子网设计管理器合成整个 PNs 的活性增强管理器. 这种分治策略相比以往方法可以降低复杂度, 但复杂度仍为指数级[17]. Liu 等对于具有资源的简单序列进程系统 (systems of simple sequential processes with resources, S^3PRs), 仅为有效变迁覆盖中的 MPRT 回路添加控制器以预防死锁. 这种添加方法可以降低外部控制器的结构, 但会拒绝一些可达标记[19]. Liu 等通过为 MPRT 回路添加控制器预防系统死锁, 并为 MPCT 回路添加控制器预防二次死锁, 解决具有关键资源的 S^3PRs 的死锁. 该策略考虑了造成死锁的一种特殊标记, 称为无死锁不安全标记, 但其控制器添加方式会拒绝一些可达标记[20]. Feng 等将 S^3PRs 的活性条件推广到具有共享资源的顺序系统 (systems of sequential systems with shared resources, S^4PRs), 通过饱和的 PA 回路表征 S^4PRs 中的死锁. 该方法考虑具有多资源获取的系统, 但控制器结构较为复杂[22]. 为此, Feng 等在文献 [22] 的基础上将可饱和 PA 回路分为独立和依赖两类, 通过为独立可饱和回路添加控制器预防死锁. 该方法已经取得突破性的进展, 但其没有考虑造成死锁的无死锁不安全标记[23]. 对于多数量资源获取的系统, Feng 等将死锁用饱和的 PRT 回路表征, 得到系统的变迁覆盖. 该方法复杂度较低, 但会拒绝一些可达标记, 且该方法可适用的系统较少[24].

上面提到的死锁控制方法没有考虑到资源故障的情况, 但现实生活中, 资源故障是不可避免的. 对于具有不可靠资源的 AMSs, 需要应用有效的控制策略预防系统死锁. Feng 等研究了具有一种不可靠资源, 且一次最多有一个不可靠资源故障的 S^3PRs, 并在变迁覆盖[17] 的基础上提出强变迁覆盖. 该方法确保即使资源故障, 系统也可以无限地处理所有类型的零件. 但他们没有考虑具有多种不可靠资源的系统[18]. Du 等对于具有多种不可靠资源的 S^3PRs, 通过线性约束控制死锁标记下的 MCRT 回路. 该控制器考虑了需要 MCRT 回路中不可靠资源的活动库所, 保证资源故障时进程的连续操作. 该方法考虑了多种资源故障的情况, 但 S^3PRs 中没有同时需要可靠和不可靠资源的阶段, 其不能解决这种阶段下资源故障造成的阻塞[21].

面向进程 Petri 网 (process-oriented Petri nets, POPNs) 可以直观地描述 AMSs, 但其模型结构复杂度较高, 且不易寻找造成死锁的回路. 为此, 人们提出面向资源 Petri 网 (resource-oriented Petri nets, ROPNs) 的建模方式. Zhao 等使用饱和的生产进程回路 (production process circuit, PPC) 描述死锁, 通过在线监督管理器, 前瞻预测托肯的前进. 该策略通过控制变迁的触发解决死锁问题. 但其复杂度较高[25]. 对于具有加权资源分配的简单顺序进程系统 (weighted S^3PRs, WS^3PRs), Liu 等定义其对应的 ROPN 模型, 提出一种基于结构分析的不需要外部监督的死锁预防方法. 该方法主要借助回路中弧线的权值合理分配资源, 其主要关注资源的利用率, 没有考虑可达标记的问题[26]. Chen 等定义 S^3PRs 对应的 ROPN 模型, 通过对其进行可达性分析揭示不良标记与强连通子网之间的关系. 他们仅为每个强连通子网添加一个死锁控制器, 因此控制器的结构较为简单. 但是这种借助可达分析的方法复杂度较高[27]. Chen 等对具有权重的 AMSs, 通过控制一些特定变迁的触发, 避免受控系统出现死锁. 该方法具有多项式复杂度, 但会拒绝一些可达标记[28].

通过分析发现, 现有死锁预防策略的问题集中体现在计算复杂度较高、拒绝一些可达标记、外部控制器的结构复杂、资源故障导致阻塞等多个方面. 由于 ROPN 模型的结构特征, 其更方便寻找造成死锁的可饱和回路. 然而现有的 ROPN 模型较为简单, 它们大多仅为具有灵活路径的 AMSs 建模. 基于这些前提, 本文研究一类具有多种不可靠资源的复杂 AMSs 的死锁问题, 其主要贡献总结如下.

(1) 通过建立 POPNs 与 ROPNs 的联系, 提出一种将 POPNs 转化为 ROPNs 的算法. 与现有 ROPN 模型相比, 本文的模型可以表示具有多类型资源获取、装配操作和灵活路径的 AMSs.

(2) 提出一种新的死锁监督控制器. 与现有控制器相比, 该控制器具有控制开关, 通过实时改变控制库所的容量, 系统可以接受更多的安全标记.

(3) 提出危险库所的概念, 并为它们添加资源缓冲子网. 使得即使不可靠资源故障, 不需要故障资源的零件仍然可以持续加工.

1 使用 ROPNs 建模

本节首先介绍了 ROPNs 的基本符号, 然后提出一种新的算法建立一类复杂 AMSs 的 ROPN 模型. 与文献 [25–28] 使用的模型不同, 本文的模型可以表示具有多类型资源获取、装配操作和灵活路径的 AMSs, 且其考虑了同时需要可靠和不可靠资源的特殊阶段.

1.1 ROPNs

在 AMSs 中, 零件在每个阶段获取不同的资源进行加工处理. ROPNs 使用资源库所表示 AMSs 中零件加工所需资源, 用变迁表示零件的加工阶段. 通过使用变迁将资源库所与零件存储中心相连接, 可以描述一种零件的加工过程. 由于一个 AMS 中具有多种不同的零件, 在 ROPNs 中引入颜色, 使用有色变迁区分不同零件的加工阶段. 此外, 库所中的有色托肯只能触发具有相同颜色的变迁. 通过这种触发方式, ROPNs 可以完整的描述 AMSs 中所有零件的加工进程.

在 ROPNs 中, K 表示资源库所的容量. 资源库所中的托肯表示当前状态下已经被占用的资源空间, 托肯的移动表示零件的加工过程. 此外, P 表示库所集, T 表示变迁集, I 表示输入函数, O 表示输出函数, M 表示当前标记.

定义 1. $\forall p\in P$, p 的输入变迁为: $\bullet p=\{t: t\in T, O(p, t)>0\}$, p 的输出变迁为: $p\bullet=\{t: t\in T, I(p, t)>0\}$. 与 p 类似, $\forall t\in T$, t 的输入库所为: $\bullet t=\{p: p\in P, I(p, t)>0\}$, t 的输出库所为: $t\bullet=\{p: p\in P, O(p, t)>0\}$.

在 ROPNs 中, 每个变迁 t_{id} 只有一个独特的颜色 c_{id}, 用来区分不同的零件阶段. 一个库所 p_l 可能包含多种颜色的托肯, 且不同库所中可能有相同颜色的托肯, 用 (p_l, c_i) 表示 p_l 中 c_i 颜色的托肯. $M(p_l)$ 表示标记 M 下 p_l 中所有颜色的托肯数量, $M(p_l, c_{id})$ 表示标记 M 下 p_l 中 c_{id} 颜色的托肯数量.

定义 2. 给定可达标记 $M\in R(N, M_0)$, 当同时满足式 (1) 和式 (2) 时, 称变迁 t_{id} 是使能的.

$$M(p_l,c_{id})I(p_l,t_{id}) \tag{1}$$

$$K(p_l)M(p_l)I(p_l,t_{id})\cdot(c_{id})+O(p_l,t_{id})\cdot(c_{id}) \tag{2}$$

当仅满足式 (1) 时, 表明 t_{id} 的输入库所中都有足够的 c_{id} 颜色的托肯, 此时 t_{id} 为进程使能. 当仅满足式 (2) 时, 表明 t_{id} 的输出库所都有足够的资源空间, 此时 t_{id} 为资源使能. 用 T_{EN} 表示标记 M 下的使能变迁集.

定义 3. 给定可达标记 $M\in R(N, M_0)$, 如果触发使能变迁 t_{id} 后 M 变为 M', 用符号表示为 $M[t_{id}\rangle M'$.

$$M'(p_l)=M(p_l)I(p_l,t_{id})\cdot(c_{id})+O(p_l,t_{id})\cdot(c_{id}) \tag{3}$$

1.2 SRMGs

多类型资源获取阶段可以表示同时需要可靠和不可靠资源的零件阶段, 装配操作可以提高零件的生产效率, 灵活路径可以增强零件加工的灵活性, 这些结构都具有很重要的现实意义. 但是现有的 POPNs 不能为同时具有这些结构的 AMSs 建模. 因此本节建立 POPNs 与 ROPNs 的联系, 提出一种新的 ROPN 模型, 称为特殊资源标记图 (special resource marked graphs, SRMGs).

定义 4. 一个 SRMG 表示为一个八元组 $N=(P, T, C, X, I, O, M, K)$, 其结构为:

(1) $P=P_0\cup P_R$, P_0 为零件存储中心. $P_R=P_{Rr}\cup P_{Ru}$ 是一组有限的资源库所集, 其中 P_{Rr} 是一组可靠资源库所集, P_{Ru} 是一组不可靠资源库所集. $\forall p_a, p_b\in P$, $|p_a\bullet|\geqslant 1$ 且 $|\bullet p_b|\geqslant 1$, 表示系统可以具有灵活路径.

(2) $T=\cup T_i$, T_i 表示第 i 种零件加工过程中的变迁. $\forall t_c, t_d\in T_i$, $|t_c\bullet|\geqslant 1$ 且 $|\bullet t_d|\geqslant 1$. 若 $t_c\bullet=\bullet t_d$, 表示多类型资源获取阶段. 若 $t_c\bullet\neq\bullet t_d$, 表示装配操作. $P\cup T\neq\varnothing$ 且 $P\cap T=\varnothing$.

(3) C: $C(p_l)$ 和 $C(t_{id})$ 分别代表 p_l 中托肯的颜色和 t_{id} 的颜色.

(4) $X=\cup X_i$, $i\in\mathbb{N}_n^+=\mathbb{N}_n\setminus\{0\}=\{1, \cdots, n\}$ 是零件类型集, X_i 表示第 i 种零件.

(5) I: $P\times T\to\mathbb{N}=\{0, 1, 2, \cdots\}$, 表示从库所到变迁的有向弧.

(6) O: $P\times T\to\mathbb{N}$, 表示从变迁到库所的有向弧.

(7) M: $P\to\mathbb{N}$, 表示库所中托肯的数量, M_0 为初始标记.

(8) K: $P\to\mathbb{N}^+=\mathbb{N}\setminus\{0\}=\{1, 2, \cdots\}$, $K(p_l)$ 表示 p_l 一次可以具有的最大托肯数, $K(P_0)=\infty$.

假设 AMSs 中没有零件会连续使用同一资源, 如果有, 将这些连续操作视为一个加工阶段. $\forall p_l\in P_R$, p_l 有多个输入变迁表示零件对 p_l 的资源空间的竞争,

多个输出变迁表示 p_l 可选择的加工零件.

下面使用算法 1 建立 ROPN 模型, 其主要借助 AMSs 的 POPN 模型, 通过删减活动库所以及更改部分弧线, 在已有 POPN 模型的基础上生成其对应的 ROPN 模型. 对于可以描述多类型资源获取、装配操作和灵活路径的 ROPN 模型, 称它们为 SRMGs.

算法 1. 基于 ROPNs 建立的模型

输入: AMSs 的 POPN 模型
输出: 对应的 ROPN 模型

1) 删除 POPN 模型中的 P_B 和 P_A, 以及连接这些库所的弧;
2) 删除为资源库所标记的容量, 并将 r_l 转化为对应的 p_l, 其中 $K(p_l) = M_0(r_l)$;
3) 将剩余网中弧的方向反转;
4) 添加零件存储中心 P_0;
5) 在 P_0 与没有输入库所的变迁 t_i 之间添加 $I(P_0, t_i)$, 在 P_0 与没有输出库所的变迁 t_j 之间添加 $O(P_0, t_j)$;
6) 为变迁赋予颜色, $C(t_{id}) = c_{id}$;

对于一个没有资源循环等待的标记 M, 其是一个无死锁标记. 如果 M 中所有使能变迁触发后都会到达死锁标记, 称该标记为无死锁不安全标记.

定义 5. 给定可达标记 $M \in R(N, M_0)$, 如果 M 为死锁或无死锁不安全标记, 称 M 为不安全标记. 否则, 称 M 为安全标记.

将触发后不会导致不安全标记的使能变迁称为活性变迁, 用 T_{RL} 表示标记 M 下的活性变迁集. 下面, 介绍本文的死锁预防策略, 其通过计算活性变迁, 阻止系统到达不安全标记.

2 死锁预防策略

本文的目标是解决 AMSs 的死锁问题, 因此需要破坏死锁的循环等待条件. 在方法的选取上, 采用死锁预防的思想. 其通过分析模型找到造成死锁的特殊结构, 然后添加控制器阻止这些特殊结构饱和达到预防死锁的目的. 基于这一思想, 本文借助 SRMGs 的结构特征, 找到造成死锁的可饱和回路, 通过离线方式为这些回路设计控制器. 控制器通过限制变迁的触发阻止系统到达不安全标记. 考虑到资源故障问题, 为危险库所添加资源缓冲子网. 这种子网起到缓冲区的作用, 使得资源故障不会影响系统的正常运行.

2.1 基本控制器

对于触发后导致回路没有剩余资源空间的变迁, 现有死锁控制器阻止它们的触发. 这导致控制策略会拒绝一些可达安全标记. 针对这一问题, 本节设计一种具有开关的控制器. 该方法可以使系统接受更多的可达安全标记. 针对控制器的结构复杂性问题, 基于简化策略仅为基本资源等待回路添加基本控制器, 在降低外部结构的基础上预防系统到达死锁标记.

定义 6. 在 SRMGs 中, 将每个库所 (变迁) 视为一个节点. 一条路径是有序的节点序列, 表示为$< x_1, x_2, \cdots, x_n >$. 其中 (1) $\{x_1, x_2, \cdots, x_n\} \subseteq P \cup T$; (2) $x_{i+1} \in x_i\bullet$. 对于一条路径, 如果 $x_1 = x_n$, 称该路径为回路. 如果一个回路不包含 P_0, 称该回路为资源等待回路 (RWC).

定义 7. 在 SRMGs 中, 如果回路 R_l 中的所有资源库所都是回路 R_k 的资源库所, 称 R_k 包含 R_l. 如果一个 RWC 不包含任何其他的 RWC, 称其为基本资源等待回路 (BRWC). 用 R_i 表示一个 BRWC, R_I 表示 R_i 集.

定义 8. 如果 t 不在 R_i 中, t 的输入 (输出) 库所在 R_i 中, 且 t 的输出 (输入) 库所不在 R_i 中, 称 t 为 R_i 的输出 (输入) 变迁. $T_I(R_i)(T_O(R_i))$ 表示 R_i 的输入 (输出) 变迁集.

根据定义 8 可知, 变迁 t 不可能同时为一个回路的输入和输出变迁.

定义 9. 对于变迁 t_{id}, 用 R_{Iid} 表示输入变迁是 t_{id} 的 BRWC 集, R_{Oid} 表示输出变迁是 t_{id} 的 BRWC 集.

定义 10. 具有多个输出库所的变迁称为多资源获取变迁, 用 t_{ma} 表示, T_{ma} 表示 t_{ma} 集. 具有多个输入库所的变迁称为多资源释放变迁, 用 t_{mr} 表示, T_{mr} 表示 t_{mr} 集. t_{ma} 的输出库所用 P_{Oma} 表示, t_{mr} 的输入库所用 P_{Imr} 表示, 如果 $P_{Oma} = P_{Imr}$, 称 t_{ma} 和 t_{mr} 是相对应的变迁.

由于 t_{ma} 具有多个输出库所, 如果 t_{ma} 为 R_i 的输入变迁, 其触发后可能占用 R_i 的多个资源空间. 当然, t_{ma} 的输出库所不一定都位于 R_i 中.

定义 11. $R_i(P) = \{p_1, \cdots, p_r\}$表示 R_i 中的库所集, $R_i(T) = \{t_1, \cdots, t_r\}$表示 R_i 中的变迁集.

事实上, 回路中的库所数一定等于变迁数.

定义 12. 在标记 M 下, 对于 R_i 中的某个托肯, 如果其触发 R_i 的活性输出变迁后仍在 R_i 中, 称该托肯为标记 M 下 R_i 的循环托肯. 否则, 称该托肯为标记 M 下 R_i 的可离开托肯.

定义 13. 给定标记 M, 如果回路 R_i 同时满足以下条件, 称其为饱和回路: (1) $M(R_i) = K(R_i) = \sum K(p_l)$; (2) 在标记 M 下, R_i 中没有可离开托肯.

为了控制 BRWC 中托肯的数量, 算法 2 为 R_I 添加

具有开关的基本控制器. R_i 的基本控制器定义如下: $R_{ci}=\{p_{ci}, s_{ci}, I(p_{ci}), O(p_{ci}), W(p_{ci}), K(p_{ci})\}$. p_{ci} 表示基本控制库所, 其分为基本可用空间 p_{ci1} 和基本缓冲空间 p_{ci2}. s_{ci} 表示基本控制开关. $I(p_{ci})$ 表示 p_{ci} 的输入弧, $O(p_{ci})$ 表示 p_{ci} 的输出弧. $W(p_{ci})$ 表示 p_{ci} 的弧的权重, 意味着变迁的触发会占用或释放 p_{ci} 的多个资源空间. $K(p_{ci})=K(p_{ci1})\oplus K(p_{ci2})$ 表示 p_{ci} 的容量, 其是根据开关状态动态变化的. 初始时, $M_0(s_{ci})=$ off, $K(p_{ci})=K(p_{ci1})$. 当 $s_{ci}=$ on 时, $K(p_{ci})=K(p_{ci1})+K(p_{ci2})$. 由于触发 t_{ma} 可能会占用 R_i 的多个资源空间, 用$|O(R_i(P), t_a)|$表示 t_a 指向 R_i 中库所的弧的数量, 则 $W(O(p_{ci}, t_a))$ 表示触发 t_a 占用的 p_{ci} 的资源空间数. 触发 t_{mr} 可能会释放 R_i 的多个资源空间, $|I(R_i(P), t_b)|$表示 R_i 中库所指向 t_b 的弧的数量, $W(I(p_{ci}, t_b))$ 表示触发 t_b 释放的 p_{ci} 的资源空间数.

算法 2. 基本控制器的添加方法

输入: R_I
输出: R_{CI}

1) **for**($i=1$; $i\leqslant|R_I|$; i++)
2) $R_i=R_i\cup p_{ci}\cup s_{ci}$;
3) $M_0(s_{ci})=$ off;
4) $K(p_{ci1})=\sum K(p_l)-1$;
4) $K(p_{ci2})=1$; $K(p_{ci})=K(p_{ci1})\ddot{\cup}K(p_{ci2})$; //$\forall p_l\in R_i$
5) **for**($a=1$; $a\leqslant|T_I(R_i)|$; a++)
6) $R_i=R_i\cup O(p_{ci}, t_a)$; //$\forall t_a\in T_I(R_i)$, $W(O(p_{ci}, t_a))=|O(R_i(P), t_a)|$
7) **for**($b=1$; $b\leqslant|T_O(R_i)|$; b++)
8) $R_i=R_i\cup I(p_{ci}, t_b)$; //$\forall t_b\in T_O(R_i)$, $W(I(p_{ci}, t_b))=|I(R_i(P), t_b)|$

定义 14. 给定标记 M, $\forall p_l\in R_i$, p_{ci} 的剩余资源空间为 $Re(M(p_{ci}))=\sum Re(M(p_l))$, 其中 $Re(M(p_l))=K(p_l)-M(p_l)$. p_{ci} 的可用资源空间为 $Ra(M(p_{ci}))=Re(M(p_{ci}))-K(p_{ci2})$. 如果$\exists t_{il}\in T_{RL}\cap T_O(R_i)$, $M[t_{il}\rangle M_1$, p_{ci} 的潜在资源空间为 $Rp(M(p_{ci}))=Re(M'(p_{ci}))$. 否则, $Rp(M(p_{ci}))=Re(M(p_{ci}))$.

当 $Re(M(p_{ci}))>0$ 时, R_i 至少有一个未被占用的资源空间. 根据定义 13 可知, R_i 不是饱和的. 即使 $Re(M(p_{ci}))=0$, 只要标记 M 下 R_i 有活性输出变迁, $Rp(M(p_{ci}))>0$. 此时 R_i 具有可离开托肯, 该托肯离开 R_i 后可以释放占用的资源空间, R_i 不是饱和的. 当 $Re(M(p_{ci}))=0$, 且 R_i 没有活性输出变迁时, $Rp(M(p_{ci}))=0$. 此时 R_i 没有可离开托肯, R_i 是饱和的.

引理 1[22]. 对于 S^4PR, 其死锁等价于系统中存在饱和的 PA 回路.

定理 1. 对于 SRMGs, 其死锁等价于系统中存在饱和的 RWC.

证明: 根据引理 1 可知, 系统死锁时部分资源库所没有剩余资源空间, 且这些资源库所之间存在循环等待. 在 SRMGs 中, 这些资源库所构成饱和的回路. 此外, 由于 $K(P_0)=\infty$, 包含 P_0 的回路永远不会饱和. 因此, 当 AMSs 出现死锁时, SRMGs 中存在饱和的 RWC. 同理, 当某个回路饱和时, 该回路一定是 RWC. 由于该回路是饱和的, 回路中一定没有剩余资源空间, 且其输出变迁不能触发. 此时回路中的托肯相互等待其他托肯占用的资源空间, 因此需要这些资源的零件永远无法完成加工, 系统出现死锁.

定理 2. 如果 R_k 包含 R_l, 只要 R_l 不饱和, R_k 也不会饱和.

证明: 根据定义 13 和定义 14, 对于 R_l, 以下两种情况下它不会饱和: 1) $Re(M(p_{cl}))\neq 0$, 此时不满足定义 13 的条件 1. 由于 R_k 包含 R_l, R_l 中具有剩余资源空间的库所也一定在 R_k 中, 因此 $Re(M(p_{ck}))\neq 0$; 2) $Re(M(p_{cl}))=0$, 且 R_l 有活性输出变迁, 表明 R_l 有可离开托肯, 此时不满足定义 13 的条件 2. 由于 R_k 包含 R_l, R_l 的活性输出变迁一定是 R_k 的输出变迁, 因此 R_k 不会饱和. 综上所述, 只要 R_l 不饱和, R_k 也不会饱和.

$R_{CI}=\{p_{CI}, s_{CI}, I(p_{CI}), O(p_{CI}), W(p_{CI}), K(p_{CI})\}$表示 R_I 的基本控制器. $(N_{CI}, M_{0CI})=(N, M_0)\otimes R_{CI}$ 表示为一个 SRMG 添加基本控制器, 在控制开关状态不变的情况下, $M(R_i)=M(p_{ci})\leqslant K(R_i)-1$. 根据定理 2 可知, 此时所有回路都不会饱和.

2.2 总控制器

现有的死锁预防策略很少考虑无死锁不安全标记. 由于这种特殊标记一定会导致死锁, 因此有必要阻止系统到达这些标记. 前文通过添加基本控制器的方式仅能预防回路饱和, 不能阻止系统到达无死锁不安全标记. 为此, 本节引入总控制器.

定义 15. 由 n ($n\geqslant 2$) 个 BRWC 组成的回路称为交互式基本资源等待回路 (IBC), B_j 表示一个 IBC.

与 BRWC 中的定义类似, $T_I(B_j)(T_O(B_j))$ 表示 B_j 的输入 (输出) 变迁集. B_{Iid} 表示输入变迁是 t_{id} 的 IBC 集, B_{Oid} 表示输出变迁是 t_{id} 的 IBC 集. $B_j(P)$ 表示 B_j 中的库所集, $B_j(T)$ 表示 B_j 中的变迁集.

为了控制 IBC 中托肯的数量, 算法 3 为部分回路添加总控制器, 其中 B_J 表示需要添加总控制器的所有回路. 与基本控制器类似, B_j 的总控制器定义如下: $B_{cj}=$

$\{P_{cj}, S_{cj}, I(P_{cj}), O(P_{cj}), W(P_{cj}), K(P_{cj})\}$. 假设 B_j 由 n_j 个 BRWC 组成, 将 $S_{cj}(P_{cj2})$ 分成 n_j 份, 每一份用$S^i_{cj}(P^i_{cj2})$ 表示, 对应于组成 B_j 的回路 R_i 的 $s_{ci}(p_{ci2})$. $K(P_{cj}) = K(P_{cj1}) \oplus K(P_{cj2})$, 其中 $K(P^i_{cj2}) = 1$, $K(P_{cj2}) = \sum K(P^i_{cj2}) = n_j$. 子开关$S^i_{cj}$与 s_{ci} 具有相同的状态, 用于控制子缓冲空间P^i_{cj2}的资源空间. 初始时, 所有子开关处于关闭状态, $M_0(S_{cj}) = \text{off}$, $K(P_{cj}) = K(P_{cj1})$. 当 $s_{ci} = \text{on}$ 时, 其对应的子开关$S^i_{cj} = \text{on}$, $K(P_{cj}) = K(P_{cj1}) + K(P^i_{cj2})$.

算法 3. 系统中总控制器的添加方法

输入: B_J

输出: B_{CJ}

1) **for**($j = 1; j \leqslant |B_J|; j$++)
2) $B_j = B_j \cup P_{cj} \cup S_{cj}$;
3) $M_0(S_{cj}) = \text{off}$;
4) $K(P_{cj1}) = \sum K(p_l) - n_j$; $K(P^i_{cj2}) = 1$; $K(P_{cj}) = K(P_{cj1}) \oplus K(P^1_{cj2}) \oplus \cdots \oplus K(P^{n_j}_{cj2})$; //$\forall p_l \in B_j$
5) **for**($c = 1; c \leqslant |T_I(B_j)|; c$++)
6) $B_j = B_j \cup O(P_{cj}, t_c)$; //$\forall t_c \in T_I(B_j)$, $W(O(P_{cj}, t_c)) = |O(B_j(P), t_c)|$
7) **for**($d = 1; d \leqslant |T_O(B_j)|; d$++)
8) $B_j = B_j \cup I(P_{cj}, t_d)$; //$\forall t_d \in T_O(B_j)$, $W(I(P_{cj}, t_d)) = |I(B_j(P), t_d)|$

定义 16. 给定标记 M, $\forall p_l \in B_j$, P_{cj} 的剩余资源空间为 $Re(M(P_{cj})) = \sum Re(M(p_l))$, P_{cj} 的可用资源空间为 $Ra(M(P_{cj})) = Re(M(P_{cj})) - K(P_{cj2})$. 如果$\exists t_{il} \in T_{RL} \cap T_O(B_j)$, $M\,[t_{il}\rangle\, M_1$, P_{cj} 的潜在资源空间为 $Rp(M(P_{cj})) = Re(M'(P_{cj}))$. 否则, $Rp(M(P_{cj})) = Re(M(P_{cj}))$.

定理 3. 给定无死锁标记 M, 对于 n_j ($n_j > 1$) 个 BRWC 组成的 B_j, 当且仅当 B_j 中的剩余资源空间小于 n_j, 且这 n_j 个回路都没有活性输出变迁时, M 为无死锁不安全标记.

证明: 充分性: 由于 M 是无死锁标记, 此时系统中没有饱和的回路. 若 B_j 中的剩余资源空间小于 n_j, 表明只有多个 BRWC 共享的资源库所具有剩余资源空间. 此时任一 BRWC 的使能输入变迁触发后都会造成该回路饱和, 且已知回路没有活性输出变迁, 因此回路的使能变迁都不能触发, M 为无死锁不安全标记.

必要性: 当 M 为无死锁不安全标记时, 回路的使能变迁触发后都会到达死锁标记, 因此回路没有活性输出变迁. 对于由 n_j 个 BRWC 组成的 B_j, 由于多个 BRWC 具有共享的资源库所, $Re(M(P_{cj})) < \sum Re(M(p_{ci}))$. 由于 M 是无死锁标记, 且回路没有活性输出变迁, 因此每个 BRWC 都至少有一个剩余资源空间, 即 $\sum Re(M(p_{ci})) \leqslant n_j$. 因此 $Re(M(P_{cj})) < n_j$.

定理 4. 如果 B_k 和 B_l 由相同的资源库所组成, 且组成 B_k 的 BRWC 的数量小于等于组成 B_l 的. 假设已经为 B_l 添加总控制器, 则不需要为 B_k 添加总控制器.

证明: 假设 B_k、B_l 的控制器分别为 P_{ck}、P_{cl}, 由于 B_k 和 B_l 具有相同的资源库所, 那么 P_{ck}、P_{cl} 的剩余资源空间是相同的. 若组成 B_k 的 BRWC 的数量小于等于组成 B_l 的, 那么 P_{ck} 的可用资源空间大于等于 P_{cl} 的. 因此添加 P_{cl} 后, 无需添加 P_{ck} 就能控制 B_k 中允许存在的最大托肯数.

对于由 n_j 个 BRWC 组成的 B_j, 主要有以下 4 种构成方式: (1) 组成 B_j 的多个回路具有同一个 t_{ma} 和 t_{ma} 相对应的变迁 t_{mr}, 但这些 BRWC 中 t_{ma} 的输出库所 (t_{mr} 的输入库所) 不同; (2) 组成 B_j 的 n_j 个 BRWC 共享一个或多个资源库所; (3) 组成 B_j 的 n_j 个 BRWC 两两之间具有共享的一个或多个资源库所. (4) 组成 B_j 的 n_{j1} 个 BRWC 共享一个或多个资源库所, 且组成 B_j 的 n_{j2} 个 BRWC 两两之间具有共享的一个或多个资源库所.

根据定理 3 可知, 只要每个 BRWC 都至少有一个仅属于自己的剩余资源空间, 或者某些没有剩余资源空间的 BRWC 具有活性输出变迁, 系统就不会到达无死锁不安全标记. 为实现这一目标, 下面将结合 B_j 的 4 种构成方式, 分别介绍总控制器的添加规则.

总控制器的添加规则: 对于第 1 种情况, 由于 t_{mr} 是组成 B_j 的多个 BRWC 的输出变迁, 其触发会释放 B_j 的多个资源空间. 因此即使不添加额外的总控制器, 也能保证系统不会出现无死锁不安全标记. 对于第 2 种情况, 由于 n_j 个 BRWC 具有相同的共享资源库所, 仅需为组成 B_j 的所有 BRWC 两两之间构成的 IBC 添加总控制器. 对于第 3 种情况, 一个 BRWC 会与相邻的两个 BRWC 分别具有不同的共享资源库所. 因此, 需要分别为组成 B_j 的所有 2, 3, …, n_j 个 BRWC 构成的 IBC 添加总控制器. 对于第 4 种情况, 结合第 2 种和第 3 种情况下的控制器添加方式即可. 最后, 根据定理 4 删减冗余的控制器.

$B_{CJ} = \{P_{CJ}, S_{CJ}, I(P_{CJ}), O(P_{CJ}), W(P_{CJ}), K(P_{CJ})\}$表示 B_J 的总控制器. $(N_{CIJ}, M_{0CIJ}) = (N, M_0) \otimes R_{CI} \otimes B_{CJ} = (N_{CI}, M_{0CI}) \otimes B_{CJ}$ 表示为 (N_{CI}, M_{0CI}) 添加总控制器. 在控制开关状态不变的情况下, $M(B_j) = M(P_{cj}) \leqslant K(B_j) - n_j$, 且每个组成 B_j 的 R_i 都满足 $M(R_i) = M(p_{ci}) \leqslant K(R_i) - 1$, 因此构成 IBC 的每个 BRWC 至少具有一个属于自己的剩余资源空间, 保证系统不会出现无死锁不安全标记.

2.3 变迁控制

为回路添加控制器后, 回路的输入变迁触发后同时会占用控制库所的资源空间. 通过控制库所的容量限制可以控制回路中的托肯数. 以此可以判断使能变迁是否为活性变迁, 从而实现变迁的控制, 预防系统到达不安全标记.

对于使能变迁 t_{id}, 如果它不是任何回路的输入变迁, 那么触发 t_{id} 只会释放回路中库所的资源空间, 因此 t_{id} 一定是活性变迁. 给定无死锁标记 M, $M\,[t_{id}\rangle\,M'$, 算法 4 可以确定以 t_{id} 为输出变迁的回路在 M'下的状态. (1) 假设 t_{id} 是 R_i 的输出变迁, 由于 t_{id} 可能属于 T_{mr}, 则 t_{id} 触发后可能释放 p_{ci} 的多个资源空间. 若 $Re(M'(p_{ci}))\geqslant 1$, 表明 M'下 R_i 具有足够的可用资源空间, 因此无需借助基本缓冲空间 p_{ci2}, $M'(s_{ci})$= off. (2) 假设 t_{id} 同时也是 B_j 的输出变迁, 若 $Re(M'(P_{cj}))\geqslant K(P_{cj2})$, 同理需要关闭 s_{ci} 对应的子开关S^i_{cj}, $M'(S^i_{cj})$ = off.

算法 4. R_i 的输出变迁 t_{id} 触发后控制器的状态

输入: 一个可达标记 $M\in R(N, M_0)$, R_i 的输出变迁 t_{id}
输出: M'

1) 初始化: $M'(p_{ci})=M(p_{ci})$, $M'(P_{cj})=M(P_{cj})$;
2) **if**($R_{Oid}\neq\varnothing$)
3) **for**($i=1$; $i\leqslant|R_{Oid}|$; i++) //$\forall R_i\in R_{Oid}$
4) $M'(p_{ci})=M(p_{ci})-W(I(p_{ci}, t_{id}))$;
5) **if**($M(s_{ci})$ = on 且$\exists S^i_{cj}$)
6) **if**($Re(M'(p_{ci}))\geqslant 1$)
7) $M'(s_{ci})$ = off;
8) **if**($B_{Oid}\neq\varnothing$)
9) **for**($j=1$; $j\leqslant|B_{Oid}|$; j++) //$\forall B_j\in B_{Oid}$
10) $M'(P_{cj})=M(P_{cj})-W(I(P_{cj}, t_{id}))$;
11) **if**($M(s_{ci})$ = on 且 $M(S^i_{cj})$ = on)
12) **if**($Re(M'(p_{ci}))\geqslant 1$ 且 $Re(M'(P_{cj}))\geqslant K(P_{cj2})$)
13) $M'(s_{ci})$ = off; $M'(S^i_{cj})$ = off;
14) $M'(p_l)=M(p_l)-I(p_l, t_{id})+O(p_l, t_{id})$;

算法 5 将输出库所是 P_0 的变迁 t 称为结束变迁 t_f, 并用 T_F 表示结束变迁集. 用 $M(R)$ 表示 $Ra(M(p_{ci}))<W(O(p_{ci}, t_{id}))$ 的 R_i 集, $M(B)$ 表示 $Ra(M(P_{cj}))<W(O(P_{cj}, t_{id}))$ 的 B_j 集. 需要依次检查标记 M'下 $M(R)$ 与 $M(B)$ 中的回路是否会饱和. 当所有的 $Rp(M(p_{ci}))\geqslant W(O(p_{ci}, t_{id}))$ 且 $Rp(M(P_{cj}))\geqslant W(O(P_{cj}, t_{id}))$ 时, 表明标记 M 下回路具有活性输出变迁 t_{il}. 由于 t_{id} 是回路的输入变迁, 触发 t_{id} 后占用的资源空间一定不是 t_{il} 需要的, 因此标记 M'下 t_{il} 仍是回路的活性输出变迁. 这意味着标记 M'下回路具有可离开托肯, M'是安全标记, 因此 t_{id} 是活性变迁.

算法 5. 判断使能变迁 t_{id} 是否为活性变迁

输入: 一个可达标记 $M\in R(N, M_0)$, 使能变迁 t_{id}
输出: $M'(p_{ci})$, $M'(P_{cj})$, $M'(s_{ci})$, $M'(S^i_{cj})$, Fl; //Fl = true 表示 t_{id} 是活性变迁

1) 初始化: Fl = true;
2) **if**($B_{Iid}\cap M(B)=\varnothing$且 $R_{Iid}\cap M(R)=\varnothing$)
3) M' = 算法 4(M, t_{id});
4) **if**($R_{Iid}\cap M(R)\neq\varnothing$)
5) **if**($t_{id}\bullet\bullet\in T_F$)
6) $M'(s_{ci})$ = on; $M'(S^i_{cj})=M'(s_{ci})$;
7) **else**
8) **for**($i=1$; $i\leqslant|R_{Iid}\cap M(R)|$; i++)//$\forall R_i\in B_j\cap R_{Iid}\cap M(R)$
9) **if**($Rp(M(p_{ci}))<W(O(p_{ci}, t_{id}))$)
10) Fl = false; break;
11) **if**($Rp(M(p_{ci}))\geqslant W(O(p_{ci}, t_{id}))$)
12) $M'(s_{ci})$ = on; $M'(S^i_{cj})=M'(s_{ci})$;
13) **if**($B_{Iid}\cap M(B)\neq\varnothing$)
14) **if**($t_{id}\in T_F$)
15) $M'(s_{ci})$ = on; $M'(S^i_{cj})=M'(s_{ci})$;
16) **else**
17) **for**($j=1$; $j\leqslant|B_{Iid}\cap M(B)|$; j++) //$\forall B_j\in B_{Iid}\cap M(B)$
18) **if**($Rp(M(P_{cj}))<W(O(P_{cj}, t_{id}))$)
19) Fl = false; break;
20) **if**($Rp(M(P_{cj}))\geqslant W(O(P_{cj}, t_{id}))$)
21) $M'(S^i_{cj})$ = on; $M'(s_{ci})=M'(S^i_{cj})$;
22) **if**(Fl = true)
23) $M'(p_{ci})=M(p_{ci})+W(O(p_{ci}, t_{id}))$;
24) $M'(P_{cj})=M(P_{cj})+W(O(P_{cj}, t_{id}))$;
25) $M'(p_l)=M(p_l)-I(p_l, t_{id})+O(p_l, t_{id})$;

在算法 5 中: (1) 由于 $K(P_0)=\infty$, 如果使能变迁 t_{id} 是结束变迁, 其一定是活性变迁. (2) 如果 $t_{id}\bullet\bullet$是结束变迁, 由于结束变迁的前集库所中的托肯一定是可离开托肯, 因此触发 t_{id} 不会使回路饱和, t_{id} 一定是活性变迁. 若此时回路的可用资源空间不足, 只需打开控制开关, 回路一定具有充足的剩余资源空间. (3) 如果 t_{id} 是某些回路的输入变迁, 其触发后会占用回路中库所的资源空间, 可能导致回路饱和.

定理 5. 算法 5 得到的活性变迁触发后不会到达不安全标记.

证明: 算法 2 为所有 BRWC 添加基本控制器. 对于算法 5 求得的任意一个活性变迁 t_{id}, $M\,[t_{id}\rangle\,M'$. 对于$\forall R_{Iid}$, 当 $Ra(M(p_{ci}))\geqslant W(O(p_{ci}, t_{id}))$ 时, 由于 $Ra(M(p_{ci}))=Re(M(p_{ci}))-1$, 即 $Re(M(p_{ci}))-1\geqslant W(O(p_{ci}, t_{id}))$, $Re(M'(p_{ci}))=Re(M(p_{ci}))-W(O(p_{ci}, t_{id}))\geqslant 1$, 因此触发 t_{id} 不会导致 R_i 饱和. 当 $Ra(M(p_{ci}))<W(O(p_{ci}, t_{id}))$ 时, $Rp(M(p_{ci}))\geqslant W(O(p_{ci}, t_{id}))$ 表明 R_i 具有活性输出变

迁, 根据定义 13 和定义 14 可知, 标记 M'下 R_i 不是饱和的. 若 $Rp(M(p_{ci})) < W(O(p_{ci}, t_{id}))$, 表明 p_{ci} 的潜在资源空间不足. 算法 5 不允许这样的变迁 t_{id} 触发, t_{id} 不是活性变迁. 算法 5 可以通过限制基本控制库所中的托肯阻止 BRWC 饱和, 使所有回路有剩余资源空间或可离开托肯, 因此触发 t_{id} 不会导致系统到达死锁标记.

算法 3 为 B_J 添加总控制器. 对于 B_J 中的任意一个 B_j, 与 R_i 类似, 当 $Ra(M(P_{cj})) \geqslant W(O(P_{cj}, t_{id}))$ 时, 由于 $Re(M(P_{cj})) - n_j \geqslant W(O(P_{cj}, t_{id}))$, $Ra(M(P_{cj})) = Re(M(P_{cj})) - n_j$, $Re(M'(P_{cj})) = Re(M(P_{cj})) - W(O(P_{cj}, t_{id})) \geqslant n_j$. 此时每个回路都有属于自己的一个剩余资源空间, M'不是无死锁不安全标记. 当 $Ra(M(P_{cj})) < W(O(P_{cj}, t_{id}))$ 且 $Rp(M(P_{cj})) \geqslant W(O(P_{cj}, t_{id}))$ 时, 表明 B_j 具有活性输出变迁. 标记 M'下托肯可以存放在 P_{cj} 的总缓冲空间, 此时组成 B_j 的每个 BRWC 都至少有一个仅属于自己的资源空间, 因此 M'不是无死锁不安全标记. 当 $Ra(M(P_{cj})) < W(O(P_{cj}, t_{id}))$ 且 $Rp(M(P_{cj})) < W(O(p_{ci}, t_{id}))$ 时, P_{cj} 的潜在资源空间不足. 此时算法 5 不允许 t_{id} 触发, t_{id} 不是活性变迁. 算法 5 可以限制总控制库所中的托肯, 因此触发 t_{id} 不会导致系统到达无死锁不安全标记.

由于 t_{id} 是在 T_{RL} 中任意选取的一个变迁, 因此对于算法 5 求得的任一活性变迁, 其触发后到达的 M'一定是安全标记.

由于算法 5 得到的活性变迁触发后不会到达不安全标记, 因此本文的控制策略允许这些变迁触发. 该策略在系统具有未完成加工的零件时, 允许处理新的零件, 这可以提高零件的生产效率.

2.4 资源缓冲子网

本节考虑资源故障造成的阻塞问题. 现有的死锁预防策略通常没有考虑同时需要可靠和不可靠资源的阶段. 针对具有这种复杂结构的 AMSs, 本节提出危险库所的概念, 通过为这些库所添加资源缓冲子网, 确保资源故障时系统不会出现阻塞.

定义 17. 给定 $t_{ik} \in T_{ma} \cap T_i$, 如果$\exists t_{ir} \in T_{mr} \cap T_i$, 且 $t_{ik}\bullet = \bullet t_{ir}$, 称 t_{ik} 为危险变迁, t_{ir} 为修复变迁. $p_k = t_{ik}\bullet \cap P_{Rr}$, $p_u = t_{ik}\bullet \cap P_{Ru}$. 称 p_k 为危险库所, p_k 和 p_u 为相对 t_{ir} 的相关库所. 用 P_K 表示危险库所集.

下面通过算法 6 为 P_K 添加资源缓冲子网, 使得即使不可靠资源故障, 托肯也能及时释放占用的危险库所的资源空间.

算法 6. 为 SRMGs 添加资源缓冲子网

输入: SRMG(N, M_0)
输出: SRMG(N_q, M_{0q}) //添加资源缓冲子网后的 SRMG

```
1 for(k = 1; k ≤ |P_K|; k++)
2   if(p_k ∈ •t_ir ∩ P_Rr) //∀p_k ∈ P_K
3     N = N ∪ q_k ∪ t_fir ∪ t_rir;
4     N = N ∪ I(p_k, t_fir) ∪ I(q_k, t_rir) ∪ O(q_k, t_fir) ∪ O(p_k, t_rir);
5     if(p_u ∈ •t_ir ∩ P_Ru)
6       K(q_k) = min{K(p_u), K(p_k)};
```

算法 6 为每个危险库所 p_k 添加资源缓冲子网, 其包含资源故障变迁 t_{fir}, 资源恢复变迁 t_{rir}, 恢复库所 q_k, 以及输入弧和输出弧. 其中 $K(q_k) = \min\{K(p_u), K(p_k)\}$用来保证 q_k 具有足够的资源空间放置 (p_k, c_{ir}).

对于本文的死锁预防策略: (1) 如果托肯的剩余路径中不需要故障资源, 那么只需根据算法 5 确定该托肯能否前进. (2) 否则, 在算法 5 的基础上, 仅允许托肯前进到故障资源库所. (3) 当 p_u 故障时, 其输出变迁 t_{ir} 无法触发. 1) 如果该阶段托肯没有占据可靠资源的资源空间, 只需等待故障资源修复, 系统就可以回到正常运行状态. 2) 如果该阶段托肯同时占据危险库所 p_k 的资源空间, 令 p_k 触发资源缓冲子网中的资源故障变迁 t_{fir}. 此时 (p_k, c_{ir}) 前进到恢复库所 q_k 并释放 p_k 的资源空间, $Re(M(p_k)) = K(p_k) - M(p_k) - M(q_k)$. 同理可以计算出含有 p_k 的回路的剩余资源空间、可用资源空间和潜在资源空间. 当 $Re(M(q_k)) = K(q_k) - M(q_k) = 0$ 时, 不允许托肯触发危险变迁 t_{ik}. (4) p_u 修复后, 如果 p_k 具有剩余资源空间, 允许 q_k 中的托肯可以触发 t_{rir} 回到 p_k, 系统回到正常运行状态.

综上所述, 算法 5 可以与资源缓冲子网结合, 求得任何状态下系统的活性变迁. 使得 p_u 的故障不会影响需要 p_k 的零件的生产, 且故障资源修复后系统仍能持续的加工所有零件.

3 例子

本文使用 CPN tools 进行仿真分析. 首先通过弧线连接库所与变迁, 为库所设置容量. 然后为库所中不同颜色的托肯赋予不同的类型描述它们的颜色. 此外, 通过赋值也可以描述变迁的颜色. 最终建立一个完整的 ROPN 模型. 在未限制变迁的触发时, 使能变迁的触发可能导致系统死锁, 此时系统无法继续运行. 通过在现有 ROPN 模型上加入控制器限制变迁的触发, 可以使

系统正常运行, 系统不会出现死锁.

示例 1: 给定一个 AMS, 其零件所需资源为: X_1: $r_3 \to r_1 \cup r_5 \to r_2$, X_2: $r_2 \to (r_3 \to r_1 \to r_5) \cup (r_4 \to r_6 \to r_7) \to r_8$, X_3: $r_2 \to (r_3 \to r_6 \to r_4) \mid (r_8 \to r_1 \to r_7) \to r_5$, X_4: $r_3 \to r_4 \to r_6$. $\forall i \in \mathbb{N}_8^+ \backslash 1$, $M_0(r_i) = 2$, $M_0(r_1) = 3$, r_1 和 r_8 可能出现故障.

示例 1 中 r_1 和 r_5 之间的“∪”表示某个零件阶段同时需要 r_2 和 r_3. $(r_3 \to r_1 \to r_5)$ 和 $(r_4 \to r_6 \to r_7)$ 之间的“∪”用来区分并行加工的多个子零件分别需要的资源. “|”表示某零件具有多条灵活子路径. 对于该 AMS, 通过算法 1 可以得到其对应的 SRMG, 使用 CPN tools 对该 AMS 进行仿真如图 1.

对于示例 1 中 AMS 对应的 SRMG, $R_I = \{R_1, R_2, R_3, R_4, R_5\}$, 其中 $R_1 = <p_4, t_{24}, p_6, t_{36}, p_4>$, $R_2 = <p_2, t_{22}, p_3, t_{12}, p_1, t_{13}, p_2>$, $R_3 = <p_2, t_{22}, p_3, t_{12}, p_5, t_{13}, p_2>$, $R_4 = <p_1, t_{25}, p_5, t_{27}, p_8, t_{35}, p_1>$, $R_5 = <p_7, t_{27}, p_8, t_{35}, p_1, t_{37}, p_7>$. 通过算法 2 为 R_I 添加的基本控制器如表 1 所示.

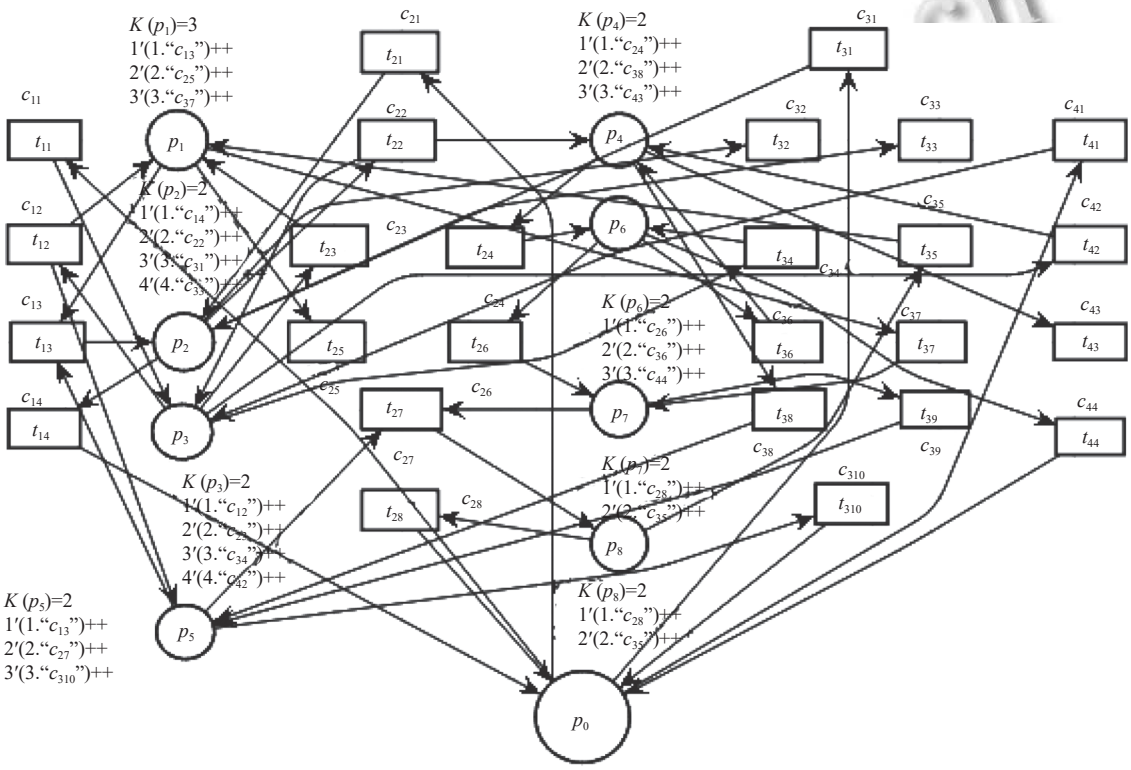

图 1 示例 1 中 AMS 的仿真图

表 1 示例 1 中 R_I 的基本控制器

回路	p_{ci}•	•p_{ci}	$K(p_{ci})$	包含的库所
R_1	t_{26}, t_{38}, t_{44}	t_{22}, t_{34}, t_{42}	$3 \oplus 1$	p_4, p_6
R_2	$t_{14}, t_{25}, t_{33}, t_{34}, t_{37}, t_{42}$	$t_{11}, t_{21}, t_{31}, t_{35}, t_{41}$	$6 \oplus 1$	p_1, p_2, p_3
R_3	$t_{14}, t_{23}, t_{27}, t_{34}, t_{310}, t_{42}$	$t_{11}, t_{21}, t_{25}, t_{31}, t_{38}, t_{39}, t_{41}$	$5 \oplus 1$	p_2, p_3, p_5
R_4	$t_{13}, t_{28}, t_{37}, t_{310}$	$t_{12}, t_{23}, t_{33}, t_{38}, t_{39}$	$6 \oplus 1$	p_1, p_5, p_8
R_5	$t_{13}, t_{25}, t_{28}, t_{39}$	$t_{12}, t_{23}, t_{26}, t_{33}$	$6 \oplus 1$	p_1, p_7, p_8

t_{12} 是 R_4 的输入变迁, t_{13} 是 R_4 的输出变迁, $t_{12}\bullet = \bullet t_{13} = \{r_1, r_5\}$. 由于触发 t_{12} 会同时占用 r_1 和 r_5 的资源空间, 且 R_4 中同时具有 r_1 和 r_5, 因此触发 t_{12} 会占用 p_{c4} 的两个资源空间. 同理, 触发 t_{13} 会释放 p_{c4} 的两个资源空间. 因此 $W(O(p_{c4}, t_{12})) = 2$, $W(I(p_{c4}, t_{13})) = 2$. p_{ci} 的其他输入、输出弧的权值都是 1.

对于由 R_2、R_3、R_4 和 R_5 组成的 B_j, 由于 R_2 和 R_3 组成的回路属于构成 IBC 的第 1 种情况, 不需要为其添加总控制器. 对于由 R_2、R_4 和 R_5 组成的 B_j, 其是构成 IBC 的第 2 种情况, 因此需要分别为 R_2 和 R_4, R_2 和 R_5, R_4 和 R_5 组成的回路添加总控制器. 对于 R_3、R_4 和 R_5 组成的 B_j, 其是构成 IBC 的第 3 种情况, 需要分别为 R_3 和 R_4, R_4 和 R_5, R_3、R_4 和 R_5 组成的回路添加总控制器. 由于 R_3 和 R_4 组成的回路与 R_2 和 R_4 组成回路的资源库所相同, 且构成 IBC 的 BRWC 的数量相同, 根据定理 4, 不需要额外为 R_3 和 R_4 组成的回路添加总控制器. 因此需要添加总控制器的 $B_J = \{B_1, B_2, B_3,$

$B_4\}$, 其中 B_1 = <p_2, t_{22}, p_3, t_{12}, p_1, t_{25}, p_5, t_{27}, p_8, t_{35}, p_1, t_{13}, p_2 >, B_2 = <p_2, t_{22}, p_3, t_{12}, p_1, t_{37}, p_7, t_{27}, p_8, t_{35}, p_1, t_{13}, p_2 >, B_3 = <p_1, t_{25}, p_5, t_{27}, p_8, t_{35}, p_1, t_{37}, p_7, t_{27}, p_8, t_{35}, p_1 >, B_4 = <p_2, t_{22}, p_3, t_{12}, p_5, t_{27}, p_8, t_{35}, p_1, t_{37}, p_7, t_{27}, p_8, t_{35}, p_1, t_{25}, p_5, t_{13}, p_2 >. 通过算法 3 为 B_J 添加的总控制器如表 2 所示.

表 2 示例 1 中 B_J 的总控制器

回路	P_{ci}•	•P_{ci}	$K(P_{ci})$	包含回路
B_1	t_{14}, t_{28}, t_{34}, t_{37}, t_{310}, t_{42}	t_{11}, t_{21}, t_{31}, t_{38}, t_{39}, t_{41}	9⊕1⊕1	R_2, R_4
B_2	t_{14}, t_{25}, t_{28}, t_{34}, t_{39}, t_{42}	t_{11}, t_{21}, t_{26}, t_{31}, t_{41}	9⊕1⊕1	R_2, R_5
B_3	t_{13}, t_{28}, t_{34}, t_{310}	t_{12}, t_{23}, t_{26}, t_{33}, t_{38}	7⊕1⊕1	R_4, R_5
B_4	t_{14}, t_{28}, t_{34}, t_{310}, t_{42}	t_{11}, t_{21}, t_{26}, t_{31}, t_{33}, t_{38}, t_{41}	10⊕1⊕1⊕1	R_3, R_4, R_5

与 R_4 添加的基本控制器的弧权重计算方式类似, 由于 t_{12} 是 B_3 的输入变迁, t_{13} 是 B_3 的输出变迁, 且 B_3 中同时具有 r_1 和 r_5. 因此触发 t_{12} 会占用 P_{c3} 的两个资源空间, 触发 t_{13} 会释放 P_{c3} 的两个资源空间. $W(O(P_{c3}, t_{12}))$ = 2, $W(I(P_{c3}, t_{13}))$ = 2. P_{cj} 的其他输入、输出弧的权值都是 1.

对于示例 1 中的 AMS, P_K = $\{p_5\}$. 通过算法 6 为 p_5 添加资源缓冲子网如图 2. 其中 $K(q_5)$ = min$\{K(p_1), K(p_5)\}$ = $K(p_5)$ = 2.

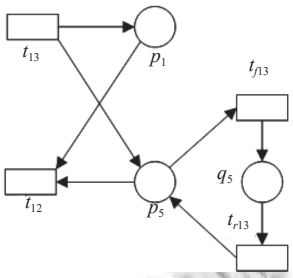

图 2 危险库所 p_5 的资源缓冲子网

对于示例 1 中的 AMS, 给定安全标记 M, $M(p_1, c_{13})$ = $M(p_1, c_{37})$ = 1, $M(p_2, c_{22})$ = $M(p_2, c_{32}/c_{33})$ = 1, $M(p_3, c_{12})$ = $M(p_3, c_{23})$ = 1, $M(p_4, c_{43})$ = 1, $M(p_5, c_{13})$ = $M(p_5, c_{27})$ = 1, $M(p_6, c_{26})$ = $M(p_6, c_{36})$ = 1, $M(p_7, c_{27})$ = $M(p_7, c_{39})$ = 1, $M(p_8, c_{35})$ = 1. $M(p_{c1})$ = 0 ♁ 1, $M(p_{c2})$ = 0 ♁ 1, $M(p_{c3})$ = 0 ♁ 0, $M(p_{c4})$ = 1 ♁ 1, $M(p_{c5})$ = 1 ♁ 1. $M(P_{c1})$ = 0 ♁ 1 ♁ 1, $M(P_{c2})$ = 0 ♁ 1 ♁ 1, $M(P_{c3})$ = 0 ♁ 1 ♁ 1, $M(P_{c4})$ = 0 ♁ 0 ♁ 1 ♁ 1. $M(s_{c3})$ = on, $M(S^3_{c4})$ = on. 使能变迁集 T_{EN} = $\{t_{23}, t_{27}, t_{33}, t_{35}, t_{36}\}$. 其中 $M(p_2, c_{32}\backslash c_{33})$ 表示 p_2 中的托肯根据当前标记决定触发哪个变迁.

下面, 通过本文的控制策略, 进行以下分析: 1) 没有资源故障时, 根据算法 5 判断使能变迁是否为活性变迁; 2) 不可靠资源故障时, 结合危险库所的资源缓冲子网, 根据算法 5 判断使能变迁是否为活性变迁.

t_{23} 是 R_4、R_5 和 B_3 的输入变迁, 且 $Ra(M(P_{c3}))$ = 0 < 1. 由于标记 M 下 B_3 没有使能输出变迁, $Rp(M(P_{c3}))$ = 0 < 1. 对于 M [t_{23}〉M_1, M_1 不是安全标记, $t_{23}\notin T_{RL}$.

t_{27} 不是任何回路的输入变迁, 因此 $t_{27}\in T_{RL}$. 由于 t_{27} 是 R_3 的输出变迁, $M(s_{c3})$ = on, $M(S^3_{c4})$ = on. 假设 M [t_{27}〉M_2, 由于 $Re(M_2(p_{c3}))$ = 1, $Re(M_2(P_{c4}))$ = $K(P_{c42})$ = 3, 根据算法 4, $M_2(s_{c3})$ = off, $M_2(S^3_{c4})$ = off.

与 t_{23} 类似, t_{33} 是 R_4、R_5 和 B_3 的输入变迁, 且 $Ra(M(P_{c3}))$ = 0 < 1. 由于标记 M 下 B_3 没有使能输出变迁, $Rp(M(P_{c3}))$ = 0 < 1. 对于 M [t_{33}〉M_3, M_3 不是安全标记, $t_{33}\notin T_{RL}$.

t_{35} 是 R_2 的输入变迁, 且 $Ra(M(p_{c2}))$ = 0 < 1. R_2 仅具有使能输出变迁 t_{33}, 由于 $t_{33}\notin T_{RL}$, R_2 没有活性输出变迁, $Rp(M(p_{c2}))$ = 0 < 1. 对于 M [t_{35}〉M_4, M_4 不是安全标记, $t_{35}\notin T_{RL}$.

t_{36} 不是任何回路的输入变迁, 因此 $t_{36}\in T_{RL}$. 由于 t_{36} 是 R_1 的输出变迁, 且 $M(s_{c1})$ = off, s_{c1} 的状态不变.

因此根据以上分析, T_{RL} = $\{t_{27}, t_{36}\}$.

在标记 M, 触发 t_{27} 会分别释放 p_5 和 p_7 的一个资源空间到达 M'. $M'(p_1, c_{13})$ = $M'(p_1, c_{37})$ = 1, $M'(p_2, c_{22})$ = $M'(p_2, c_{32}/c_{33})$ = 1, $M'(p_3, c_{12})$ = $M'(p_3, c_{23})$ = 1, $M'(p_4, c_{43})$ = 1, $M'(p_5, c_{13})$ = 1, $M'(p_6, c_{26})$ = $M'(p_6, c_{36})$ = 1, $M'(p_7, c_{39})$ = 1, $M'(p_8, c_{28})$ = $M'(p_8, c_{35})$ = 1. $M'(p_{c1})$ = 0 ♁ 1, $M'(p_{c2})$ = 0 ♁ 1, $M'(p_{c3})$ = 0 ♁ 1, $M'(p_{c4})$ = 1 ♁ 1, $M'(p_{c5})$ = 1 ♁ 1. $M'(P_{c1})$ = 0 ♁ 1 ♁ 1, $M'(P_{c2})$ = 0 ♁ 1 ♁ 1, $M'(P_{c3})$ = 1 ♁ 1 ♁ 1, $M'(P_{c4})$ = 0 ♁ 1 ♁ 1 ♁ 1.

在 M', 如果 p_1 和 p_8 同时故障. 在 p_1 未修复之前, p_1 的输出变迁 t_{13}、t_{25} 和 t_{37} 暂时不能触发. 在 p_8 未修复之前, p_8 的输出变迁 t_{28} 和 t_{35} 暂时不能触发. 使能变迁集 T'_{EN} = $\{t_{12}, t_{23}, t_{26}, t_{36}, t_{39}\}$. 此时, 危险库所 p_5 可以触发 t_{f13} 前进到 q_5, 并释放 p_5 的一个资源空间, 使需要 p_5 的零件不会因为 p_1 的故障出现阻塞.

t_{12} 是 R_4 和 R_5 的输入变迁, 由于 $W(O(p_{c4}, t_{12}))$ = 2, $Ra(M'(p_{c4}))$ = 1 < 2. 在 p_1 和 p_8 的故障修复之前, t_{28} 和 t_{37} 暂时不能触发, 标记 M'下 R_4 没有使能输出变迁, $Rp(M'(p_{c4}))$ = 1 < 2. 对于 M' [t_{12}〉M'_1, M'_1 不是安全标记, $t_{12}\notin T'_{RL}$

对于 t_{23}, 由于 $Ra(M'(p_{c4})) = Ra(M'(p_{c5})) = Ra(M'(P_{c3})) = 1 \geqslant 1$, 则 $t_{23} \in T'_{RL}$. 此外, 由于 t_{23} 是 R_3 的输出变迁, 且 $M'(s_{c3}) = \text{off}$, s_{c3} 的状态不变.

t_{26} 是 R_5、B_2、B_3 和 B_4 的输入变迁, 且 $Ra(M'(P_{c2})) = Ra(M'(P_{c4})) = 0 < 1$. B_2 有使能输出变迁 t_{39}, 但 B_4 没有使能输出变迁, $Rp(M'(P_{c4})) = 0 < 1$. 对于 $M'[t_{26}\rangle M'_3$, M'_3 不是安全标记, $t_{26} \notin T'_{RL}$.

t_{36} 不是任何回路的输入变迁, 因此 $t_{36} \in T'_{RL}$.

t_{39} 是 R_3、R_4 和 B_1 的输入变迁, 且 $t_{39}\bullet\bullet = t_{310} \in T_F$. 由于 $Ra(M'(p_{c3})) = Ra(M'(P_{c1})) = 0 < 1$, 此时 p_{c3} 和 P_{c1} 的可用资源空间不足, 因此开启控制器的开关. 对于 $M'[t_{39}\rangle M'_4$, $M'_4(s_{c3}) = \text{on}$, $M'_4(S^3_{c1}) = \text{on}$, M'_4 是安全标记, $t_{39} \in T'_{RL}$.

因此根据以上分析, $T'_{RL} = \{t_{23}, t_{36}, t_{39}\}$. 通过不断的迭代这一过程寻找不同标记下的活性变迁, 使得无论是否出现资源故障, 系统都不会到达不安全标记.

4 对比

在本节中, 简要的对比本文与现有的死锁预防策略, 结果如表 3[1–4,13–16,18,21,23,25,28–30] 所示. 其中第 2 列表示系统中是否具有灵活路径, “Y”表示“是”, “N”表示“否”. 第 3 列表示系统中是否具有装配操作. 第 4 列表示系统中是否具有多类型资源获取阶段. 第 5 列表示死锁控制方法的计算复杂度, “E”表示指数复杂度, “P”表示多项式复杂度. 第 6 列表示使用的建模方式. 第 7 列表示系统中是否存在不可靠资源, 如果存在, 是仅有一种不可靠资源, 还是具有多种不可靠资源. 第 8 列表示如果系统具有不可靠资源, 一次最多允许多少不可靠资源故障, “—”表示不存在这种情况. 第 9 列表示系统中是否存在同时需要可靠和不可靠资源的零件阶段.

表 3 死锁预防策略的比较结果

文献	灵活路径	装配操作	多类型资源获取	复杂度	建模方式	不可靠资源	不可靠资源故障	同时需要可靠、不可靠资源
[2]	Y	N	Y	P	POPN	一种	一种	Y
[3]	N	N	N	P	POPN	多种	多种	N
[4]	Y	N	N	E	POPN	多种	多种	N
[13]	N	Y	Y	E	POPN	N	—	—
[14]	Y	N	N	E	POPN	一种	一种	N
[15]	Y	Y	Y	E	POPN	N	—	—
[16]	N	N	N	E	POPN	一种	一种	N
[18]	N	N	N	E	POPN	一种	一个	N
[21]	Y	N	Y	E	POPN	多种	多种	Y
[23]	Y	N	Y	E	POPN	N	—	—
[30]	Y	Y	N	E	POPN	N	—	—
[1]	N	Y	N	P	ROPN	N	—	—
[25]	Y	N	N	P	ROPN	N	—	—
[28]	Y	N	N	P	ROPN	N	—	—
[29]	Y	N	N	P	ROPN	N	—	—
本文	Y	Y	Y	P	ROPN	多种	多种	Y

从表 3 可以看出, 大多数死锁预防方法是指数级复杂度. 为此, 研究人员提出各种策略降低计算复杂度, 以解决大规模 AMSs 的死锁问题. 对于具有不同结构的 AMSs, 研究人员先后提出多种不同的回路进行死锁控制. 此外, 他们通过减少控制器的添加数量不断优化控制策略. 后来, 人们发现借助 ROPNs 的结构特征, 可以很容易找到造成系统死锁的回路, 通过为这些回路添加控制器, 在阻止回路饱和的基础上可以预防死锁. 然而, 尽管基于 ROPNs 的建模方式复杂度较低, 但其很少考虑具有装配操作或多类型资源获取的复杂 AMSs. 另外, 人们在 ROPNs 的建模过程中, 很少考虑资源故障的问题.

本文在现有的 ROPN 模型的基础上, 提出一种新的算法, 得到一类复杂 AMSs 的 ROPN 模型. 与文献 [13–16] 等不同, 本文的死锁预防方法是多项式复杂度的, 更加适用于大规模的系统. SRMGs 同时考虑了具有多类型资源获取、装配操作和灵活路径的一类复杂 AMSs, 而文献 [1,3,25,29,30] 等没有考虑具有多类型资源获取的系统, 文献 [2–4,16,18,21,29] 等没有考虑具有装配操作的系统, 文献 [1,13,16,18] 等没有考虑具有灵活路径的系统. 此外, 本文的监督控制器具有控制开关. 通过借助控制库所额外的缓冲空间, 受控系统允许更多的安全标记发生.

对一类同时需要可靠和不可靠资源, 且允许多种不可靠资源同时故障的系统, 本文通过为危险库所添加资源缓冲子网, 避免系统出现死锁和阻塞. 而文献 [1,13,25,27] 等没有考虑具有不可靠资源的系统, 文献 [2,14,16,18] 仅考虑了具有一种不可靠资源的系统. 文献 [3,4] 虽然考虑了多种不可靠资源同时故障的情况, 但系统中没有多类型资源获取阶段. 因此以上论文中的策略无法应用于同时需要可靠和不可靠资源的 AMSs.

综上所述, 本文使用的 SRMGs 不仅可以表示复杂的 AMSs, 而且具有更简单的结构. 本文的死锁预防策略考虑了资源故障的问题, 与现有控制策略相比, 其在计算复杂度、许可性和实用性等方面都较好.

5 结论与展望

本文主要研究了一类具有多类型资源获取、装配操作和灵活路径的 AMSs, 然后使用一种新的算法建立这类 AMSs 对应的 ROPN 模型. 称这类模型为 SRMGs, 其用不可靠资源库所来表示可能出现故障的资源. 为解决资源竞争造成的死锁问题, 首先建立死锁与饱和回路之间的关系. 通过为一些特殊回路添加控制器, 使得没有资源故障时, 系统不会出现不安全标记. 然后, 为每个危险库所添加资源缓冲子网, 使得即使不可靠资源故障, 不使用故障资源的零件仍然可以持续生产. 与现有的死锁控制器不同, 本文添加的监督控制器具有控制开关, 因此控制策略可以接受更多的安全标记. 此外, 本文的死锁预防策略是多项式复杂度的, 适用于具有装配操作的大规模 AMSs. 在未来, 可以考虑将本文的方法扩展到结构更复杂的 AMSs. 另外, 可以进一步优化监督控制器的结构, 通过减少添加的控制器数量设计最优的监督控制器.

参考文献

1 Wu NQ, Zhou MC, Li ZW. Resource-oriented Petri net for deadlock avoidance in flexible assembly systems. IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans, 2008, 38(1): 56–69. [doi: 10.1109/TSMCA.2007.909542]

2 Du N, Hu H, Zhou M. A survey on robust deadlock control policies for automated manufacturing systems with unreliable resources. IEEE Transactions on Automation Science and Engineering, 2020, 17(1): 389–406. [doi: 10.1109/TASE.2019.2926758]

3 Feng YX, Xing KY, Zhou MC, *et al*. Robust deadlock prevention for automated manufacturing systems with unreliable resources by using general Petri nets. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2020, 50(10): 3515–3527. [doi: 10.1109/TSMC.2018.2884316]

4 Du N, Hu HS. Robust deadlock detection and control of automated manufacturing systems with multiple unreliable resources using Petri nets. IEEE Transactions on Automation Science and Engineering, 2021, 18(4): 1790–1802. [doi: 10.1109/TASE.2020.3019684]

5 Zhang ZL, Liu GY, Barkaoui K, *et al*. Adaptive deadlock control for a class of Petri nets with unreliable resources. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2022, 52(5): 3113–3125. [doi: 10.1109/TSMC.2021.3062469]

6 关旋旋. 基于 Petri 网的死锁避免方法研究. 自动化应用, 2022, (6): 4–7.

7 林荣峰. 基于 Petri 网的自动制造系统多步向前死锁预测方法研究 [硕士学位论文]. 西安: 西安电子科技大学, 2020.

8 卫屏. 基于 Petri 网的资源共享装配系统活性构建方法 [硕士学位论文]. 西安: 西安电子科技大学, 2020.

9 王锋刚. 含变迁故障 Petri 网的鲁棒活性控制 [硕士学位论文]. 西安: 西安电子科技大学, 2022.

10 刘伟, 史晓浩, 孙红伟. 基于逻辑混合 Petri 网的混合系统建模与分析. 山东科技大学学报 (自然科学版), 2021, 40(4): 65–75. [doi: 10.16452/j.cnki.sdkjzk.2021.04.008]

11 陈金栋, 刘伟, 冯新, 等. 基于逻辑工作流网的有限无死锁组合. 山东科技大学学报 (自然科学版), 2020, 39(5): 89–97.

12 程学珍, 王常安, 李继明, 等. 基于自适应神经模糊 Petri 网的电机故障诊断. 山东科技大学学报 (自然科学版), 2020, 39(3): 109–117.

13 Hu H, Zhou MC. A Petri net-based discrete-event control of automated manufacturing systems with assembly operations. IEEE Transactions on Control Systems Technology, 2015, 23(2): 513–524. [doi: 10.1109/TCST.2014.2342664]

14 Liu GY, Li P, Li ZW, *et al*. Robust deadlock control for automated manufacturing systems with unreliable resources based on Petri net reachability graphs. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2019, 49(7): 1371–1385. [doi: 10.1109/TSMC.2018.2815618]

15 Hu H, Zhou MC, Li ZW, *et al*. Deadlock-free control of automated manufacturing systems with flexible routes and assembly operations using Petri nets. IEEE Transactions on Industrial Informatics, 2013, 9(1): 109–121. [doi: 10.1109/TII.

2012.2198661]
16 Wu YC, Xing KY, Luo JC, *et al*. Robust deadlock control for automated manufacturing systems with an unreliable resource. Information Sciences, 2016, 346–347: 17–28.
17 Li ZW, Zhu S, Zhou MC. A divide-and-conquer strategy to deadlock prevention in flexible manufacturing systems. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 2009, 39(2): 156–169. [doi: 10.1109/TSMCC.2008.2007246]
18 Feng YX, Xing KY, Gao ZX, *et al*. Transition cover-based robust Petri net controllers for automated manufacturing systems with a type of unreliable resources. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2017, 47(11): 3019–3029. [doi: 10.1109/TSMC.2016.2558106]
19 Liu HX, Xing KY, Zhou MC, *et al*. Transition cover-based design of Petri net controllers for automated manufacturing systems. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2014, 44(2): 196–208. [doi: 10.1109/TSMC.2013.2238923]
20 Liu HX, Wu WM. Deadlock control policy for a class of automated manufacturing systems with key resources. Proceedings of the 12th IEEE International Conference on Networking, Sensing and Control. Taipei: IEEE, 2015. 486–491.
21 Liu HX, Wu WM, Yang HY. Strong controllable siphon basis-based robust deadlock control for manufacturing systems with multiple unreliable resources. IEEE Access, 2020, 8: 269–277. [doi: 10.1109/ACCESS.2019.2958331]
22 Feng YX, Xing KY, Zhou MC, *et al*. Structural liveness analysis of automated manufacturing systems modeled by S^4PRs. IEEE Transactions on Automation Science and Engineering, 2019, 16(4): 1952–1959. [doi: 10.1109/TASE.2019.2905277]
23 Feng YX, Xing KY, Zhou MC, *et al*. Liveness analysis and deadlock control for automated manufacturing systems with multiple resource requirements. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2020, 50(2): 525–538. [doi: 10.1109/TSMC.2017.2767902]
24 Feng YX, Zhou MC, Tian F, *et al*. Deadlock prevention controller for automated manufacturing systems modeled by S^4PR. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2021, 51(12): 7403–7412. [doi: 10.1109/TSMC.2020.2971455]
25 Zhao YM, Chai XJ, Zhao LZ. An efficient deadlock avoidance policy for FMS using ROPN. Advanced Materials Research, 2014, 998–999: 751–754.
26 Liu D, Hou YF, Barkaoui K, *et al*. Intrinsically live structures in process and resource-oriented Petri nets modeling automated manufacturing systems. Proceedings of the 11th IEEE International Conference on Networking, Sensing and Control. Miami: IEEE, 2014. 578–583.
27 Chen HF, Wu NQ, Zhou MC. Resource-oriented Petri net-based approach to deadlock prevention of AMSs. Proceedings of the 2015 IEEE International Conference on Systems, Man, and Cybernetics. Hong Kong: IEEE, 2015. 515–520.
28 Chen HF, Wu NQ, Li ZW, *et al*. Liveness of disjunctive and strict single-type automated manufacturing system: An ROPN approach. IEEE Access, 2019, 7: 17760–17771. [doi: 10.1109/ACCESS.2019.2896254]
29 Chen HF, Wu NQ, Zhou MC. A novel method for deadlock prevention of AMS by using resource-oriented Petri nets. Information Sciences, 2016, 363: 178–189.
30 Chen C, Hu HS. Liveness-enforcing supervision in AMS-oriented HAMGs: An approach based on new characterization of siphons using Petri nets. IEEE Transactions on Automatic Control, 2018, 63(7): 1987–2002. [doi: 10.1109/TAC.2017.2758842]

(校对责编: 牛欣悦)