Deadlock Avoidance For Flexible Manufacturing Systems With Choices Based On Digraph Circuit Analysis

†Wenle Zhang and Robert P. Judd
School of Electrical Engineering and Computer Science
Ohio University
Athens, Ohio 45701

Abstract

Due to existence of concurrent part flow and resource sharing in modern automated flexible manufacturing systems, deadlock is a common problem and its occurrence causes loss of productivity. According to [12], when a manufacturing system is modeled by a digraph, existence of circuits in such a graph is a necessary condition for deadlock. Our research further shows that the knot and order of a circuit is closely related to impending deadlocks – a type of deadlock that is more difficult to detect. In this paper, a deadlock avoidance method for flexible manufacturing systems with flexible part routing is presented together with new concepts such as broken circuit, supremal circuit. The new method is highly permissive since the effective free space calculation captures more parts flow dynamics, especially when there exist multiple knots in the digraph model. The online policy runs in polynomial time once the set of basic circuits of the digraph is computed offline. Simulation results on selected examples are given.

Keywords: flexible manufacturing system, choice, digraph, circuit, deadlock avoidance.

1. Introduction

In recent years, research on deadlock detection, prevention and avoidance for flexible manufacturing systems has been very active. Some of the significant work have adopted Petri net (PN) models [1,2,4,6,11,13,15] as a formalism to describe the manufacturing system. Banaszak[1] proposed a *deadlock avoidance algorithm* (DAA) that developed a restriction policy to guarantee that no circular wait situations will occur. Viswanadham [11] developed a deadlock avoidance algorithm which suggested using a recovery mechanism in case of system deadlock. Zhou [15] developed the *sequential mutual exclusions* (SME) and *parallel mutual exclusions* (PME) concepts and derived the sufficient conditions for a PN containing such structures to be bounded, live, and reversible. Structural properties of PNs such as siphons and traps are used in [2,4] to determine potential deadlock situations.

Another formalism is to describe the manufacturing system using graphs [3,5,8-9,12,14]. In this approach the vertices represent resources and the arcs (edges) represent product part flows between resources. Cho [3] developed the concept of bounded circuits with empty and non-empty shared resources to detect deadlock. Judd [8] derived a set of static linear inequalities that when they are satisfied deadlock is avoided. Lipset [9] expanded upon [8] and quantified both necessary and sufficient conditions for deadlock to occur in a manufacturing system.

It has been challenging to find a solution to avoid impending deadlocks that is arbitrary steps away from a primary one. Fanti [5] studied second level deadlock – the impending deadlock one step away from primary deadlock. Barkaoui [2] used a one step look-ahead controller, which cannot avoid impending deadlocks that are more than one step away. Lawley [10] studied special cases of choices in part routings.

The major contribution of this paper is the extension of our previous results on deadlock avoidance for systems *without choice* in part routing presented in Zhang [14] and Lipset [9] to systems *with choice* in part routing, called free choice. Free choice has been well studied in the Petri net formalism, however, we have not found that it has been studied systematically in the digraph formalism. Because of choices introduced, part flow dynamics become much more complex, order evaluation method given by [14] is no longer valid due to the added routing alternative. A systematic circuit analysis is presented. New concepts such as broken circuit, basic circuit, supremal circuit, are presented. These concepts help to decrease number of circuits for space checking thus increasing efficiency and to increase permissiveness of our deadlock avoidance policy thus increasing productivity. Commitment, order and effective free space of a circuit are all redefined to deal with choice in part routing. The new deadlock avoidance policy with flexible part routing supports still applies to systems without choice. It is also unique in that it avoids both primary deadlocks and impending deadlocks that are arbitrary steps away from a primary one. It can be shown that the proposed policy runs in polynomial time once the set of circuits of the digraph is computed. The rest of the paper is organized as: section 2 describes the system model. Section 3 performs circuit analysis. Section 4 studies the order and effective free space calculation of circuits. Section 5 presents the deadlock avoidance policy and analysis. Section 6 provides some simulation results and section 7 concludes the paper.

2. The System Model And Deadlock

A flexible manufacturing system consists of a set R of a finite number of resources (such as, NC machines, robots, buffers, etc.), a set P of a finite number of part types that the system can produce. Each part type $p \in P$ is assigned a process plan that defines a finite number of steps of operations need to be performed on parts of the type. We assume that each step be performed on exactly one resource. Thus a process plan p can be represented as a sequence of resources $p=r_1r_2\ldots r_m$. Each resource $r\in R$ has a *capacity*, denoted as C_r, which can be considered as a multiple of identical units. The capacity can be naturally extended to a set of resources, $R_1 \subseteq R$, as C_{R1}.

Now let us introduce choice into a process plan. In case of a choice step, the next resource can be chosen from more than one resource. A choice step is indicated by a pair of parentheses in the process plan. Inside the pair of parentheses are the possible next resources separated by commas. A choice step can recursively contain choice steps. Every process plan has a loading and one or more unloading step, for a part to enter and exit the system. The

† Corresponding author. Phone: 740-597-1481, Fax: 740-593-0007, Email: zhangw@bobcat.ent.ohiou.edu

loading step which may also be a choice step, called step 0, is intentionally left out of the process plan representation since it does not contribute an arc to the system graph for our purpose. An unloading step is kept in the plan for indicating the last step although it does not contribute an arc to the system graph. The load/unload station is represented as resource R_0. An example plan with choices is given in the following,

```
p1=R1-(R4-R6,R2-(R1,R5),R3-R5)-R4-(R5,R6)
   1   2  3  4   5  6   7  8   9   10 11
```

There are three choice steps in this plan. The first choice step is that after R1 the part can go to any one of R4, R2 or R3, thus leading to 3 choice branches which merge at second R4. The second choice is that after R2 the part goes to either R1 or R5. And the third choice is that after the second R4 the part goes to either R5 or R6. The steps are sequentially numbered in the order they are listed. If a part is at step 7, then its next step is 8. If a part is at step 4, then its next step is 5 or 6.

Once the system is in operation, there will exist a set Q of parts in the system at any given time. Each part $q \in Q$ belongs to a part type $p \in P$, denoted as $P_q=p$. Each part has a unique current step, denoted as S_q, which can be considered as the state of the part. The *state* of the system, denoted as $\boldsymbol{n}$, is defined as a vector of C_R elements corresponding to all parts currently in the system. An element of $\boldsymbol{n}$ has value 0 for an empty resource unit. The state changes with parts flowing through the system, such as loading a new part, unloading a finished part or transporting a part from one resource to the next resource. When a new part q is first loaded into the system, $S_q = 1$. A part q that has exited the system or is still waiting for being loaded has $S_q = 0$.

For deadlock avoidance purpose, flexible manufacturing systems can be modeled by a directed graph—called a *wait relation graph* (WRG), which is constructed from all process plans. The WRG $G = (R, A)$ consists of a set R of *vertices* and a set A of *directed arcs*. Each vertex represents a resource. A directed arc a is drawn from vertex r_1 to vertex r_2, if r_2 immediately follows r_1 (including choices) in at least one process plan, denoted as $a= r_1r_2$. So, a step has one or more arcs (called *choice arcs*), corresponding to the current resource (vertex), called *tail*, and one next resource for a non-choice step or more than one next resources for a choice step, called *head*(s).

A *subgraph* $G_1 = (R_1, A_1)$ of G consists of a subset of the vertices and a subset of arcs of G such that all the arcs in A_1 connect vertices in R_1. From graph theory, we know that a *path* is defined to be a sequence of vertices $r_0r_1r_2\ldots r_k$, and a *circuit* is a path with $r_0 = r_k$. A circuit is *simple* if it does not contain any other circuit. With choice introduced, we extend a circuit (except for simple circuit) to be a subgraph that is strongly connected.

A part q is *enabled* in state $\boldsymbol{n}$ if any of its next resource(s) is free. If part q is currently being processed in resource r_1 and the next free resource is r_2, then q enabled means that once r_1 finishes its operation on q, the part can be transported or moved from r_1 to r_2. A system state $\boldsymbol{n}$ is *live* if a sequence of part moves exists such that the system can be emptied. Otherwise, the state is in *deadlock*.

Deadlocks can be further categorized into two major types, *primary deadlock* and *impending deadlock*. A system state $\boldsymbol{n}$ is in *primary deadlock* if a circular wait situation exists [1]. A primary deadlock can be understood as a siphon in the Petri net model of the system becomes empty of tokens, or as a circuit in the digraph model is filled with parts where no part is enabled. A system state $\boldsymbol{n}$ is in *impending deadlock* if parts exist in the system that can be moved; however, the system will inevitably enter a primary deadlock after a finite number of part moves.

3. Circuits Analysis

Given a system digraph, there are existing methods that find all simple circuits [7] [12]. In the following, we assume all simple circuits are given as a set C_S.

Several operations can be defined on circuits. The *intersection* of two circuits c_1 and c_2, denoted as $c_1 \cap c_2$, is the subgraph whose vertices and arcs are on both c_1 and c_2. The *union* of two circuits c_1 and c_2, denoted as $c_1 \cup c_2$, is the subgraph whose vertices and arcs are on c_1 or c_2. The union is said to be *vertex joined* if the intersection of the two circuits has at least a vertex.

In the following, we will discuss how to find the set of circuits necessary for deadlock avoidance—called deadlock circuits (or DA) set, denoted as C.

Obviously, a choice arc can be on more than one simple circuit and a simple circuit can have only one choice arc of any choice step, but it can have more than one choice arc of different choice steps. If all choice arcs $a_1, a_2, \ldots, a_k$ of a choice step are on k different simple circuits, $c_1, c_2, \ldots, c_k$, then the union circuit $c = c_1 \cup c_2 \cup \ldots \cup c_k$ is called the *choice circuit* of the choice step and the choice step is called the *pivot* of the choice circuit. A choice circuit may have one or more choice steps other than the pivot. The union of two choice circuits is said to be *choice joined* if the intersection of two choice circuits has at least a choice arc from each pivot.

Because of the introduction of choices, a choice arc may form an escape path from a circuit. Such a circuit is called a broken circuit.

Definition 3.1: A circuit is broken if there is a resource (vertex) on the circuit where all choice steps using the resource as a tail have at least one choice arc that is not on the circuit.

Simple circuits, choice circuits and more complicated circuits can all be broken. This definition is rather complicated, examples 1-3 in the following will help clarify it.

Definition 3.2: A *basic circuit* is defined as a non-broken circuit that does not contain any other non-broken circuit.

Basic circuits are the smallest deadlock units of a system graph, let C_B denote the set of all basic circuits of a system graph. All other deadlock units can be formed by vertex joined union circuits of basic circuits.

Theorem 3.1**:** A broken circuit c that does not contain any basic circuit will not generate deadlock.

Proof: Because a choice arc a is not included by c, a part q with a as a choice arc can always go along arc a, which means q does not necessarily need a resource of c, then c can not form a circular wait and thus can not generate a deadlock. ■

Given N (>1) different circuits all with the same set of vertices $c_1 = (R_1, A_1)$, $c_2 = (R_1, A_2)$, …, and $c_N = (R_1, A_N)$, if $A_1 \subset A_k$, $A_2 \subset A_k$, …, $A_{k-1} \subset A_k$, $A_{k+1} \subset A_k$, …, $A_N \subset A_k$, then c_k is said to be the *supremal* circuit among the N circuits. We also say that c_k *covers* every other circuits. So, if given two circuits c_1, c_2 and c_2 covers c_1, then c_1 and c_2 have the same set of resources and c_2 has all arcs of c_1 and at least one more arc that c_1 does not have.

*Theorem*3.2: Given two circuits c_1 and c_2, if c_2 covers c_1, then c_1 can be removed from the DA set C, that is, its space does not need to be checked for deadlock avoidance.

Proof: Since c_1 has the same resources but less arcs than c_2, so c_2 has all order one knots and all commitments of c_1 (defined later) in any given state. Then the effective free space of c_2 is less than or equal to that of c_1, thus all deadlocks avoided by space(c_1) > 0 are also avoided by space(c_2) > 0. Therefore, c_1 can be removed from the DA set C for deadlock avoidance. ■

So, the DA set C of circuits necessary for deadlock avoidance should include all basic circuits in C_B not covered by and all

supremal or covering *vertex joined union* circuits of basic circuits. So a circuit in C is either a basic circuit or a union circuit of two or more basic circuits.

An algorithm for calculating the set of basic circuit and the DA set C is under development for another paper. In the following, we assume the DA set C is given.

Example 3.1. Figure 1 shows a simple manufacturing system that makes three types of parts with one choice step $\{a_1, a_2\}$. The simple circuits identified are $c_1 = (\{r_1, r_2\}, \{a_1, a_3\})$ and $c_2 = (\{r_1, r_3\}, \{a_2, a_4\})$ and both are broken circuits. They do not have deadlock individually. The only basic circuit for this system is the non-broken choice circuit $c_3 = c_1 \cup c_2 = (\{r_1, r_2, r_3\}, \{a_1, a_2, a_3, a_4\})$ since it includes both choice arcs a_1 and a_2. A deadlock occurs if a part at the choice step r_1 needs to move along either a_1 or a_2, a part at r_2 moves along arc a_3 and a part at r_3 moves along arc a_4. Therefore, the DA set is $C = C_B = \{c_3\}$.

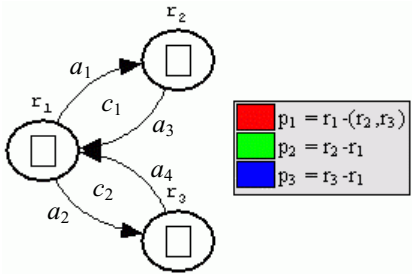


Figure 1. Broken circuit example

Example 3.2. A manufacturing system has a digraph as shown in figure 2. The system makes two types of parts. The first type of parts has a process plan $p_1 = r_1\text{-}(r_2, r_5)\text{-}r_3\text{-}r_4$. The second type of parts has a process plan $p_2 = r_4\text{-}r_3\text{-}r_2\text{-}r_1$. It is easy to identify that the system graph consists of 4 simple circuits:

$c_1 = (\{r_1, r_2\}, \{a_1, a_2\})$, $c_2 = (\{r_2, r_3\}, \{a_3, a_4\})$

$c_3 = (\{r_3, r_4\}, \{a_5, a_6\})$, $c_4 = (\{r_1, r_2, r_3, r_5\}, \{a_2, a_4, a_7, a_8\})$

So, $C_S = \{c_1, c_2, c_3, c_4\}$. Arcs a_1 and a_7 are choice arcs of the choice step of p_1.

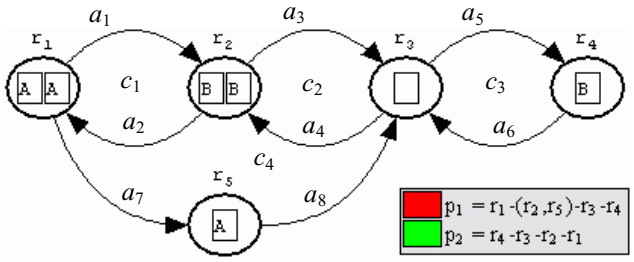
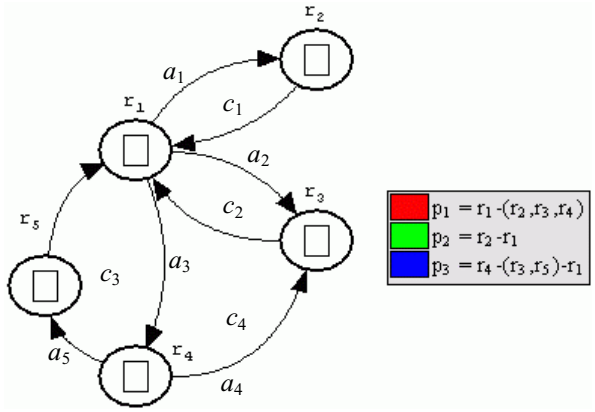


Figure 2. Basic circuits example

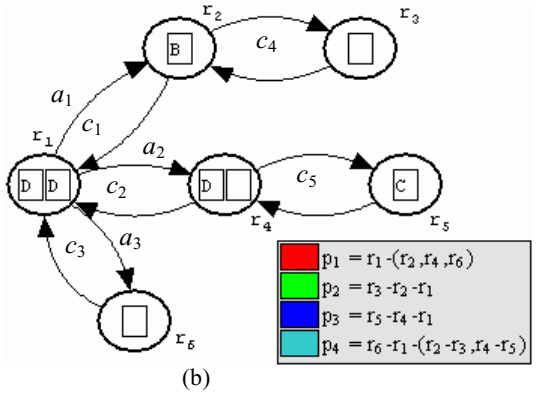
Simple circuits c_1 and c_4 are both broken because of choice arcs a_1 and a_7, c_2 and c_3 are non-broken. The only choice circuit $c_5 = c_1 \cup c_4$ is non-broken. So, $C_B = \{c_2, c_3, c_5\}$. Possible unions are: $c_2 \cup c_3$, $c_2 \cup c_5$, $c_3 \cup c_5$, $c_2 \cup c_3 \cup c_5$, but $c_2 \cup c_5$ covers c_5 and $c_2 \cup c_3 \cup c_5$ covers $c_3 \cup c_5$. So $C = \{c_2, c_3, c_2 \cup c_3, c_2 \cup c_5, c_2 \cup c_3 \cup c_5\}$.

Example 3.3. The system in figure 3(a) makes also three types of parts, there exists two choice steps $\{a_1, a_2, a_3\}$ and $\{a_4, a_5\}$. Four simple circuits are identified, c_1, c_2, c_3 and c_4. They are all broken. Arcs a_1, a_2 and a_3 are choices arcs of the choice step of p_1. Arcs a_4 and a_5 are choice arcs of the choice step of p_3. Although $c_1 \cup c_2 \cup c_3$ includes all choice arcs of the choice step of p_1, it also includes a choice arc of the choice step of p_3 but not all, so it is broken. Similarly, $c_3 \cup c_4$ is also broken. The choice joined union circuit that includes all choice arcs is $c_5 = c_1 \cup c_2 \cup c_3 \cup c_4$ is the only basic circuit. So $C = C_B = \{c_5\}$.

Figure 3(b) describes a manufacturing system that makes 4 types of parts, there are two choice steps $\{a_1, a_2, a_3\}$ and $\{a_1, a_2\}$. Arcs a_1 and a_2 are choice arcs of the choice step of p_4 and arcs a_1, a_2 and a_3 are choice arcs of the choice step of p_1. Individually, simple circuits c_1, c_2 and c_3 are all broken. $c_6 = c_1 \cup c_2$ is a basic circuit since it includes both choice arcs of the choice step of p_4 and $c_7 = c_1 \cup c_2 \cup c_3$ is also a basic circuit since it includes all three choice arcs of the choice step of p_1. $C_B = \{c_4, c_5, c_6, c_7\}$. It can be found that $C = \{c_4, c_5, c_6, c_7, c_4 \cup c_6, c_5 \cup c_6, c_4 \cup c_5 \cup c_6, c_4 \cup c_7, c_5 \cup c_7, c_4 \cup c_5 \cup c_7\}$.



(a)



(b)

Figure 3. Circuits set calculation example

4. Space Calculation of Circuits

With choice introduced, commitment can no longer be calculated based on arcs as in [14] and [9] because of existence of choice arcs. Instead, they will be calculated with respect to a circuit. If part q is at a choice step, then potentially q can move along any one of the choice arcs. However, q will move along only one of the choice arcs.

A unit of resource r_1 that is processing a part q is said to be *committed* to circuit c if q's next resource is r_2 and arc $a = r_1$–r_2 is on c, or if q is at a choice step, then all choice arcs are on c. We also say that q commits to circuit c. Let $M_{r,c,\mathbf{n}}$ denote the number of units of resource r that are committed to circuit c when the system is in state $\boldsymbol{n}$.

Note that an empty unit is not committed and the total number of units of a resource committed can be less than the number of busy units. This happens when a busy unit is processing a part at its last step. This unit is also not committed. A unit of a resource that is on circuit c is *free* with respect to c if it is not committed to c.

The commitment can then be extended to the circuit $c = (R_1, A_1)$ as follows,

$M_{c,\mathbf{n}} = \Sigma\ M_{r,c,\mathbf{n}}$, for all $r \in R_1$

Given the system in state $\boldsymbol{n}$, the *slack* of a circuit $c = (R_1, A_1)$, denoted as $K_{c,\mathbf{n}}$, is defined as

$K_{c,\mathbf{n}} = C_{R1} - M_{c,\mathbf{n}}$

The slack can be understood as the number of available free resource units to allow for parts flow on the circuit.

Definition 4.1. Let $c_1, c_2,\ldots, c_m$, $m > 1$, be m component basic circuits of a circuit c of C. If $c_1 \cap c_2 \cap \ldots \cap c_m$ contains only a single capacity resource, then the resource is called a *knot* of c.

Given two circuit c_1 and c_2 of C, if k is a knot of c_1 and $c_2 \supset c_1$, then k is also a knot of c_2.

Definition 4.2: Given two component basic circuits c_1 and c_2 of a circuit c of C, with $k = c_1 \cap c_2$ being a knot. If in state $\boldsymbol{n}$, there exists a part on c that both has an arc of c_1 ending at k in its process plan and will commit to an arc of c_2 starting at k, then c_1 is said to be *connected* to c_2 with respect to c, denote as $c_1 \rightarrow c_2$. If $c_1 \rightarrow c_2$ and $c_2 \rightarrow c_1$, then c_1 and c_2 are called *cross-connected*, denoted as $c_1 \leftrightarrow c_2$. If c has m component basic circuits $c_1, c_2,\ldots, c_m$, with $k = c_1 \cap c_2 \cap \ldots \cap c_m$ being a knot, then $c_1, c_2,\ldots, c_m$ are *cyclically connected* if $c_1 \rightarrow c_2, c_2 \rightarrow c_3, \ldots, c_m \rightarrow c_1$.

Definition 4.3. Given a circuit c of C with knot k. The order of knot k with respect to the circuit c in state $\boldsymbol{n}$, denoted as $O_{k,c,\mathbf{n}}$, is defined as

$$O_{k,c,\mathbf{n}} = \begin{cases} 1, & \text{if two or more basic circuits intersecting at } k \\ & \text{are cyclically connected.} \\ 0, & \text{otherwise.} \end{cases}$$

Based on this definition, if c_1 and c_2, $c_2 \supset c_1$, are two circuits of C and K={all knots of c_1}. Then,

$O_{k,c2,\mathbf{n}} = O_{k,c1,\mathbf{n}}$, $\forall\ k \in K$.

The order definition can be extended to a circuit. Let c be a circuit that contains m knots, $k_1, k_2, \ldots, k_m$. Then, the order of c is given by

$$O_{c,\mathbf{n}} = \sum_{i=1}^{m} O_{ki,c,\mathbf{n}}$$

The order of a basic circuit is zero. Since a basic circuit either does not contain any other component basic circuits or has the intersection of the component basic circuits more than a node, so it has no knot. This is not the case in [9] or [14].

Definition 4.4. Let c be a circuit of C in state $\boldsymbol{n}$. The *effective free space* of c, denoted as $F_{c,\mathbf{n}}$, is given by,

$F_{c,\mathbf{n}} = K_{c,\mathbf{n}} - O_{c,\mathbf{n}}$

Theorem 4.1. Let G be the WRG of a manufacturing system and C be the DA set of of G. Then G is live in state $\boldsymbol{n}$ if $F_{c,\boldsymbol{n}} > 0$, $\forall c \in C$.

Proof. This can be similarly proved as the corresponding theorem on systems without choice in our previous work [9]. ■

As a sufficient condition for a system state to be live, theorem 4.1 will be used as the basis for developing our deadlock avoidance policy later.

Example 4.1. Consider the state given in figure 2 of example 2 where all parts A are of type p_1 and all parts B are of type p_2. The commitment, order and space calculation for all circuits of C of the system graph is shown in the table 1.

According to theorem 4.1, the given system state is in deadlock since the last circuit has effective free space 0. And as a matter of fact, the state is actually an impending deadlock or second level deadlock [5].

Table 1. Commitment, order and space for example 4.1

	c_2	c_3	$c_2 \cup c_3$	$c_2 \cup c_5$	$c_2 \cup c_3 \cup c_5$
M	0	1	1	5	6
O	0	0	0	0	1
F	3	1	3	1	0

Example 4.2. Consider the state given in figure 3(b) of example 3 where part B is of type p_2 and all parts C are of type p_3 and all parts D are of type p_4. Selected circuits c_5, c_6 and $c_5 \cup c_6$ are calculated in table 2. If all other circuits of C have effective free space > 0, then the given state is not in deadlock and the fact is that it is not in deadlock.

Table 2. Commitment, order and space for example 4.2

	c_5	c_6	$c_5 \cup c_6$
M	2	3	5
O	0	0	0
F	1	2	1

5. Deadlock Avoidance Policies

Based on the sufficient conditions established earlier, two deadlock avoidance policies will be developed, which can be applied to process controllers of actual manufacturing systems to detect and avoid deadlocks. The difference of the two policies is they evaluate order of a knot or circuit differently. Policy I uses a lazy order evaluation and policy II uses an incremental order evaluation.

Lazy order evaluation: Given m basic circuits $c_1, c_2,\ldots, c_m$ and $c_1 \cap c_2 \cap \ldots \cap c_m = k$ is a knot. The order of k is calculated as one if there exist $i > 1$ circuits, say $c_1, c_2,\ldots, c_i$, such that there exist a type of part that needs to visit c_1 then c_2, a type of part that needs to visit c_2 then c_3, …, and a type of part that needs to visit c_i then c_1; calculated as zero, otherwise.

Lazy order evaluation determines the order of a knot by statically establishing cyclic connectedness among the basic circuits joined by the knot so that orders can be calculated offline. This evaluation implies as long as it is possible to have cyclic connectedness in the future evolution of the system, the order is set to one.

Policy I: Given the WRG G of a system starting from empty of parts. Let C be the DA set of G, which can be calculated offline or during system startup or initialization. The EFS of each circuit in C is calculated based on lazy order evaluation. Then, a part move is accepted only if after the part move, the EFS of all affected circuits remain positive (Theorem 4.1).

Incremental order evaluation: Given m basic circuits $c_1, c_2,\ldots, c_m$ and $c_1 \cap c_2 \cap \ldots \cap c_m = k$ is a knot. The order of k is initialized to zero and incrementally updated using definition 4.2 only when i) a part is loaded into the system and its process plan has k; ii) a part is moved to k.

According to definition 4.2, loading a part may cause two basic circuits to become connected and moving a part to a knot may cause two basic circuits to become disconnected. Then only when a part is loaded or moved to a knot, the order of the affected circuits needs to be updated.

Policy II: Policy II is policy I, except that the EFS calculation is based on incremental order evaluation.

Both policies need to calculate the DA set C of G, which might be expensive but can be done offline. An efficient string manipulation procedure to compute circuits was presented in [12]. [2], [4] and [5] have similar calculation to find all circuits or

siphons. Let K be the set of knots of G, $|p_i|$ be the length of process plan p_i, i =1~$|P|$, where $|P|$ is the number of process plans, and $L = \Sigma(|p_i|,i=1\sim|P|)$. In general, it can be assumed that $|K| << L$. Policy I can be computed in linear time ($O(|C|)$), since orders can be established offline while calculating C. Policy II dynamically calculates orders of knots and captures more dynamics of the system, thus is more permissive but more expensive. However, the incremental order calculation can be done in polynomial time. Since the connectedness of two basic circuits c_1 and c_2 with $c_1 \cap c_2 = k$ can be established by examining if arc $r_1 \rightarrow k \rightarrow r_2$ (assume r_1 on c_1 and r_2 on c_2) is contained in the process plan of all parts in the system, but for all parts of the same type, this test needs only be done once and the number of such tests is limited by the number of process plans. There are at most $|C|^2$ such pairs of basic circuits. The computation needed to update connectedness among all basic circuits is then in the order of $O(|C|^2 L)$. The cyclic connectedness (thus the order) can then be established by checking for connectedness among m basic circuits for all knots and the computation is trivial compared to update connectedness. Then policy II can be computed in polynomial time.

The deadlock avoidance algorithm will not allow any deadlock state – called *correct* in the literature, this is guaranteed by theorem 4.1. And it allows most live states with a very high average permissiveness as shown in the simulation section.

6. Simulation

The deadlock avoidance policies have been implemented in Java. Simulation has been run to calculate the state space allowed by the deadlock avoidance method on several examples. Simulation results show that the deadlock avoidance method is indeed correct. The *permissiveness*, calculated as the number of allowed live states by the algorithm against the actual number of live states is obtained on these examples as given in the following.

Example 6.1: Consider the manufacturing cell shown in figure 4 (A case study in [7]). The cell is composed of three robots (R1, R2 and R3; each one can hold one product at a time) and four machines (M1, M2, M3 and M4; each one can process two products at a time). There are three loading buffers (named I1, I2 and I3) and three unloading buffers (named O1, O2 and O3) for loading and unloading the cell. The action area for robot R1 is I1, O3, M1, M3); for robot R2 is I2, O2, M1, M2, M3, M4; and for robot R3 is I3, O1, M2, M4.

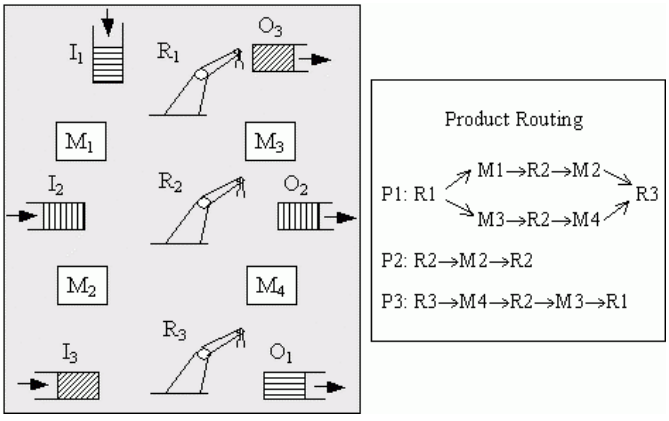


Figure 4. The manufacturing cell for example 6.1

The cell manufactures three types of products P1, P2 and P3, as described in the following:

i) A raw product of type P1 is taken from I1 and is placed in O1 once it has finished its processing. The sequence of operations for this type is M1→M2 (i.e. treatment in M1 and then in M2) or M3→M4 (treatment in M3 followed by treatment in M4).

ii) A raw product of type P2 is taken from I2, processed in M2 and routed to O2.

iii) A raw product of type P3 is taken from I3, processed in M4 followed by treatment in M3, and finally placed in O3.

The directed graph is shown is figure 5. The simple circuits identified are labeled in figure 5.

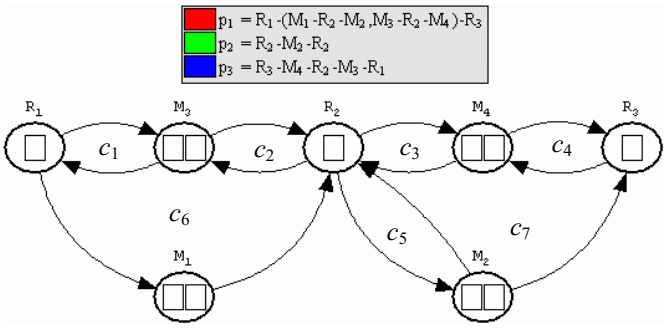


Figure 5. Digraph for example 6.1

Due to the choice step at R_1, both simple circuit c_1 and c_6 are broken. The choice circuit is $c_1 \cup c_6$ which is covered by $c_8 = c_1 c_1 \cup c_6 \cup c_2$. So the set of basic circuits is $C_B = \{c_2, c_3, c_4, c_5, c_7, c_8\}$. Basic circuit c_7 is covered by its supremal circuit $c_3 \cup c_4 \cup c_5 \cup c_7$. After removing circuits covered by their corresponding supremal circuits, the circuits set C is found to be,

$C = \{ c_2, c_3, c_4, c_5, c_8, c_2 \cup c_3, c_2 \cup c_5, c_3 \cup c_4, c_3 \cup c_5, c_8 \cup c_3, c_8 \cup c_5, c_2 \cup c_3 \cup c_4, c_2 \cup c_3 \cup c_5, c_8 \cup c_3 \cup c_4, c_8 \cup c_3 \cup c_5, c_3 \cup c_4 \cup c_5 \cup c_7, c_2 \cup c_3 \cup c_4 \cup c_5 \cup c_7, c_3 \cup c_4 \cup c_5 \cup c_7 \cup c_8\}$.

Simulation with policy II applied shows that 20801 live states are allowed out of total 22019 live states. That corresponds to a permissiveness as high as 20801/22019 = 94.5%.

More simulation results on examples given earlier are,

Apply to example 3.2, policy II allows 419 live states out of total 445 live states. The permissiveness is 94.2%.

Apply to example 3.3(b), policy II allows 10930 live states out of 11032 total live states. The permissiveness is calculated as 99.1%.

Apply to an example system without choice, example 6.1 in [14], policy II has a permissiveness of 90.7%

In general, policy II allows more live states than [2] and [4]. Essentially, this is because the order evaluation captures more parts flow dynamics, especially when there exist multiple knots. Method [2] allows more states than [4]. However, even with the one step look-ahead controller, method [2] still cannot guarantee to avoid all deadlocks. Also, policy II performs better than [5] when higher than second level deadlock exists.

7. Conclusions

A systematic circuit analysis for flexible manufacturing systems with choice in part routing is presented in this paper. Concepts of broken circuits, basic circuits and supremal circuits are introduced. These concepts help to decrease number of circuits for space checking thus increasing efficiency and to increase permissiveness of our deadlock avoidance policy. Definitions of commitment, order and effective free space of a circuit [14][9] are all extended and refined to handle choice in part routing. We found that basic circuits and knots in the WRG of a manufacturing system are the rudimentary causes of deadlocks. Especially, order one knots are essential to impending deadlocks –

a type of deadlock more difficult to detect (The impending deadlock one step away from primary deadlock is called second level deadlock [5]). Based on the space calculation, a sufficient condition is quantified for a system state to be live. Two deadlock avoidance policies are established on this sufficient condition. Several examples are simulated with the two policies applied. Simulation results show that policy II is highly permissive and provides more live states than results by related methods in the literature. One disadvantage of the proposed method is that the calculation of *C* might be expensive, which [2], [4] and [5] all suffer. However, the calculation can be done off line. A well-known string manipulation method for this calculation was given in [12]. Also, the necessary condition has not yet been established that may contribute to even higher permissiveness. Developing the necessary condition will be one of our future research topics.

8. References

[1] Banaszak, Z. and B. Krogh, "Deadlock Avoidance in Flexible Manufacturing Systems with Concurrently Competing Process Flows," IEEE Trans. on Robotics and Auto., vol. 6, no. 6, 1990, pp. 724-733.

[2] Barkaoui, K., and I. B. Abdallah, "Deadlock Avoidance in FMS Based on Structural Theory of Petri Nets," IEEE Symposium On Emerging Technologies and Factory Automation, V. 2, pp. 499-510, 1995.

[3] Cho, H., T.K. Kumaran, and R. Wysk, "Graph-Theoretic Deadlock Detection and Resolution for Flexible Manufacturing Systems," *IEEE Trans. on Robotics and Auto*., vol. 11, no. 3, pp. 550-527.

[4] Ezpeleta, J., J. M. Colom, and J. Martinez, "A Petri Net Based Deadlock Prevention Policy for Flexible Manufacturing Systems," *IEEE Transactions on Robotics and Automation*, V. 11, N. 2, pp. 173-184, April 1995.

[5] Fanti, M.P., Maione, B., Mascolo S., and Turchiano, B., "Event-Based Feedback Control for Deadlock Avoidance in Flexible Production Systems", IEEE Trans. on Robotics and Auto., Vol. 13, no. 6, 1997, pp. 347-363.

[6] Hsieh, F. and S. Chang, "Dispatching-driven deadlock avoidance controller synthesis for flexible manufacturing systems," *IEEE Trans. Robotics and Auto*., vol. 10, no. 2, 1994, pp. 196-209.

[7] Johnson, D. B., "Finding All The Elementary Circuits Of A Directed Graph", SIAM J. of Computing, Vol. 4, No. 1, 1975, pp. 77-84.

[8] Judd, R. P. and T. Faiz, "Deadlock Detection and Avoidance for a Class of Manufacturing Systems," *Proceedings of the 1995 American Control Conference*, pp. 3637-3641.

[9] Lipset, R., P. Deering, and R. P. Judd, "Necessary and Sufficient Conditions for Deadlock in Manufacturing Systems," *Proceedings of the 1997 American Control Conference*, vol. 2, pp. 1022-1026, June 1997, Albuquerque.

[10] Lawley, M. A., "Deadlock Avoidance for Production Systems with Flexible Routing", IEEE Trans. on Robotics and Auto., Vol. 15, No. 3, 1999, pp. 497-509.

[11] Viswanadham, N., Y. Narahari, and T. Johnson, "Deadlock Prevention and Deadlock Avoidance in Flexible Manufacturing Systems Using Petri Net Models," *IEEE Trans. on Robotics and Auto*., vol. 6, no. 6, 1990, pp. 713-723.

[12] Wysk, R., N. Yang and S. Joshi, "Detection of Deadlocks in Flexible Manufacturing Cells", IEEE Trans. on Robotics and Automation, Vol.7, No.6, 1991, pp.853-859.

[13] Xing, K., B. Hu and H. Chen, "Deadlock avoidance policy for Petri-net modeling of flexible manufacturing systems with shared resources," *IEEE Transactions on Automatic Control.*, vol. 41, no. 2, 1996, pp. 289-295.

[14] Zhang, W., R. P. Judd and P. Paul, "Evaluating Order Of Circuits For Deadlock Avoidance In A Flexible Manufacturing System", *Proceedings of the 2003 American Control Conference*, pp. 3679-3683, June 2003, Denver.

[15] Zhou, M. and F. DiCesare, "Parallel and Sequential Mutual Exclusion for Petri Net Modeling of Manufacturing Systems with Shared Resources," *IEEE Trans. on Robotics and Auto*., vol. 7, no. 4, 1992, pp. 550-527.