UM NOVO ALGORITMO PARA DETERMINAÇÃO DOS CICLOS FECHADOS DE ESPERA EM GRAFO DE ALOCAÇÃO DE RECURSOS

FRANCISCO YASTAMI NAKAMOTO

Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos
Escola Politécnica da Universidade de São Paulo
Avenida Professor Mello Moraes, 2231, São Paulo, SP, 05508-900
E-mails: francisco.nakamoto@poli.usp.br

Faculdade de Engenharia "Eng. Celso Daniel"
Centro Universitário Fundação Santo André
Avenida Príncipe de Gales, 821, Santo André, SP, 09060-650
E-mails: yastami@fsa.br

PAULO EIGI MIYAGI, DIOLINO JOSÉ DOS SANTOS FILHO

Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos
Escola Politécnica da Universidade de São Paulo
Avenida Professor Mello Moraes, 2231, São Paulo, SP, 05508-900
E-mails: pemiyagi@usp.br, diolinos@usp.br

Abstract— The fundamental factor to guarantee the survival of the industries in a dynamic and competitive market is attend the needs to demand for products and services more personalized with a gradual reduction of life cycle of these products. This scenery encourages the creation of new paradigms and promotes the development of innovative solutions with efficient and effective form. The proposal of the Flexible Manufacturing Systems (FMSs) aims to attend the needs of this context. FMSs are systems, which execute multiple processes simultaneously with sharing intense of a same finite set of resource. The processes are an activity sequence pre-established in which to each activity the product receives a transformation operation and/or transportation. Moreover, the product can suffer an increase of the aggregate value to each activity. The characteristic of FMSs with intense sharing of a same finite set of resources can cause an undesirable phenomenon in which the flow of the processes is permanently impeded. In this work will be introduced a new representation of the Resources Allocation Graph (RAG) and the algorithm for determination of the condition of circular wait to avoid the deadlock state.

Keywords— Flexible manufacturing system, Resource allocation, Deadlock avoidance, Petri nets.

Resumo— Atender as necessidades de demanda por produtos e serviços cada vez mais personalizados e produtos com ciclo de vida mais reduzidos são fatores fundamentais para garantir a sobrevivência das indústrias em um mercado dinâmico e competitivo. Este cenário incentiva a quebras de paradigmas e promovem o desenvolvimento de soluções inovadoras com a eficiência e a eficácia em primeiro plano. A proposta dos Sistemas de Manufatura Flexíveis (SMFs) vem ao encontro de atender as necessidades deste contexto. Os SMFs são sistemas que executam múltiplos processos simultaneamente com intenso compartilhamento de um mesmo conjunto finito de recurso. Entende-se por processo como uma seqüência de atividades pré-estabelecidas em que a cada atividade ou etapa, o produto ou serviço passa por uma operação de transformação e/ou de transporte podendo, inclusive, sofrer um aumento do valor agregado. A característica dos SMFs de compartilhamento intenso de um mesmo conjunto finito de recursos pode causar um fenômeno indesejável em que o fluxo dos processos são permanentemente impedidos. Neste trabalho serão apresentados uma nova representação do Grafo de Alocação de Recursos (GAR) e o algoritmo para determinação da condição de espera circular, permitindo-se desta forma, determinar as regras adicionais de controle para evitar o deadlock.

Palavras-chave— Sistema de manufatura flexível, Alocação de recursos, deadlock avoidance, redes de Petri.

1 Introdução

Atender as necessidades de demanda por produtos e serviços mais personalizados e produtos com ciclo de vida mais reduzidos são fatores fundamentais para garantir a sobrevivência das indústrias em um mercado dinâmico e competitivo, característica principal da globalização. Tal cenário instiga a quebras de paradigmas e promove o desenvolvimento de soluções inovadoras em busca da eficiência e eficácia.

A proposta dos Sistemas de Manufatura Flexíveis (SMFs) vem ao encontro do atendimento das necessidades dentro deste contexto. Os SMFs são sistemas que executam múltiplos processos simultaneamente com intenso compartilhamento de um mesmo conjunto finito de recurso. Entende-se por processo como uma seqüência de atividades pré-estabelecidas, em que cada atividade ou etapa, o produto ou serviço sofre por uma operação de transformação e/ou transporte. Os SMFs pertencem a classe de Sistemas Dinâmicos a Eventos Discretos (SDED) (Cassandras, 1993), isto é, são sistemas em que a evolução dos estados ocorre de forma assíncrona, com bases na ocorrência de eventos que causam uma transição de estados. Devido a esta natureza, poderá ocorrer ainda paralelismo, assincronismo e conflito

de eventos. O intenso compartilhamento de recursos entre os processos possibilita a evolução do sistema para uma condição altamente indesejável, denominado estado de auto travamento ou *deadlock*. O *deadlock* é um fenômeno que ocorre quando o fluxo das atividades são permanentemente impedidos devido à falta de informações e/ou quando as operações dos processos não podem ser executadas. O problema de *deadlock* é objeto de estudo e pesquisa em diversas áreas em que há compartilhamento de informações e/ou recursos.

Em trabalhos anteriores foram apresentados algoritmos para a determinação das cadeias cíclicas fechadas (Nakamoto et al., 2002a), denominadas de Ciclos Fechados de Espera (CFEs), os algoritmos para determinação das regras adicionais de controle de recursos (Nakamoto et al., 2002b) e a sistematização do processo de geração do controle de alocação de recursos para SMF (Nakamoto et al., 2003a; 2003b; 2003c) considerando-se apenas uma única instância de cada recurso. Como continuação dos trabalhos de pesquisa, verificou-se a necessidade de aprimoramento dos algoritmos propostos para atender os SMFs com múltiplas instâncias de recursos. Desta forma, o presente trabalho tem como objetivo apresentar uma nova estrutura de dados para representar o grafo de alocação de recursos (GAR) e um algoritmo para a determinação dos CFEs.

2 Deadlock em SMFs

A área da computação apresenta muitos estudos referentes ao problema de *deadlock*, como por exemplo em redes de computadores (Tanenbaum, 1996), sistemas operacionais (Tanenbaum, 1995), banco de dados (Date, 2004), entre outras disciplinas. Em SMFs, o *deadlock* é caracterizado quando o fluxo das atividades dos processos são permanentemente impedidos. Entretanto, cabe ressaltar que além do fluxo de informações, comparado com os sistemas computacionais, há também um fluxo de materiais. Em tais situações, o processo de abordagem do *deadlock* torna-se mais complexo.

Na literatura são apresentadas quatro condições necessárias e suficientes para que um sistema possa evoluir para um estado de *deadlock* (Cho, 1993; Fanti et al., 1997; Fanti et al., 2000; Santos Filho, 2000): mútua exclusão, retenção do recurso enquanto aguarda, não preempção e espera circular. As três primeiras condições são inerentes ao SMFs, ocorrendo-se a condição de espera circular, o sistema poderá evoluir para o estado de *deadlock*.

O *deadlock* pode ser abordado das seguintes formas: (i) Método de Detectar e Restabelecer (*Deadlock detection and recovery*): consiste em monitorar o sistema quanto a alocação de recursos entre os processos, e uma vez detectado o estado de travamento, executa-se procedimentos para restabelecer o sistema do estado de deadlock; (ii) Método de prevenir (*Deadlock Prevent*): consiste implementar o sistema de alocação de recursos de forma estruturada com o intuito de não formar condições que levem o sistema ao travamento; (iii) Método de evitar (*Deadlock Avoidance*): com bases em um algoritmo que monitora a alocação de recursos (ao detectar um estado anterior ao *deadlock*) executa procedimentos que evitam a evolução do sistema para o travamento. Tais métodos de abordagem possuem vantagens e desvantagens dependendo da quantidade de recursos compartilhados, quantidade de processos executados simultaneamente e a quantidade de atividades de cada processo. De forma geral, o método de detectar e restabelecer possui a desvantagem quanto ao tempo e o custo de restabelecimento, uma vez que o tempo elevado pode comprometer todo o controle do SMF, além de sofrer o risco da atividade travada ser descartada em sua totalidade. A maior dificuldade do método prevenir é a explosão de estados alcançáveis dos SMFs. E, quanto ao método de evitar, o excesso de restrições pode comprometer o fluxo dos processos.

Muitos trabalhos adotam os métodos apresentados anteriormente para abordar o problema de *deadlock* em SMFs. (Banaszak and Krogh, 1990; Viswanadham et al., 1990; Cho, 1993, Kumaran et al., 1994; Fanti et al., 1997; Fanti et al., 2000; Santos Filho, 2000). A complexidade de implementação destes métodos é proporcional a complexidade do sistema (Edmonds, 1995) e podem tornar-se inviáveis do ponto de vista computacional devido à quantidade de processamento. Banaszak e Krogh (1990) apresentam uma política de alocação de recursos (*Deadlock Avoidance Algorithm* - DAA) aplicando-se o método de deadlock avoidance em um modelo de redes de Petri. Viswanadham et al. (1990) propõe a implementação do método de prevenir utilizando um grafo de alcançabilidade em um modelo de redes de Petri e um algoritmo de controle em tempo real utilizando-se o método de *deadlock avoidance* no mesmo modelo. Fanti et al. (1997) propõe uma estratégia de *deadlock avoidance* aplicada no controle de recursos utilizando-se a teoria de grafos. Os autores apresentam dois tipos de dígrafos, ou grafos orientados (Cormen et. al, 2003), chamados *de working procedure digraph* e *transition digraph*. Fanti et al., (2000) apresentam uma comparação e a relação entre o método de teoria de grafos e a rede de Petri para abordar o problema de *deadlock* aplicando-se o método de *deadlock avoidance*. Banaszak e Polak (2002) apresenta um estudo sobre a relação entre a capacidade do sistema de alocação de recursos e o estado inicial do sistema com políticas de alocação de recursos em um processo cíclico seqüencial. Ezpeleta et al. (2002) propõe uma extensão do algoritmo do Banqueiro (*Banker's Algorithm*) aplicados em uma classe de redes de Petri denominada de *S*PR* para abordar o problema de *deadlock* em sistemas seqüenciais de alocação de recursos (*sequential resource allocation systems S-RAS*) com rotas alternativas e múltiplas instâncias de recursos. Ezpeleta e Recalde (2004)

apresentam o método de *deadlock avoidance* aplicado em um sistema não-sequencial de alocação de recursos que utiliza uma adaptação do algoritmo do banqueiro aplicada a um grafo de alcançabilidade em uma rede de Petri (Peterson, 1981; Reisig, 1985; Murata, 1998). Santos Filho (2000) propõe um método de *deadlock avoidance* em SMFs com bases na partição do sistema de controle em dois níveis: (i) controle de processos que é responsável pelo seqüenciamento das atividades dos processos e, (ii) controle de recursos que é responsável pelo gerenciamento da utilização dos recursos, aplicando-se o método de restrições (*Flow in Suppression*).

Considerando-se a natureza e a dinâmica do controle de SMFs o presente trabalho adota o método de *deadlock avoidance*.

3 Arquitetura do Sistema de Controle do SMF

Segundo Miyagi (1996) controlar pode ser interpretado como a aplicação de uma ação, ou de um conjunto de ações pré-estabelecidas para que aquilo que se considera como objeto de controle possa atingir determinados objetivos pela imposição de um comportamento dinâmico desejado ao sistema. Em SMFs, tais ações devem garantir a execução das atividades e/ou operações nos vários níveis de controle.

A grande dificuldade do controle dos SMFs é a formulação do comportamento global do sistema. Segundo Edmonds (1995), a complexidade é a propriedade da representação em que o comportamento global é difícil de ser formulado mesmo possuindo-se todas as informações dos componentes e suas inter-relações. A complexidade é uma propriedade que o modelo de um sistema real pode possuir (Santos Filho, 2000), isto é, a complexidade pode surgir quando é imposto um determinado comportamento dinâmico ao sistemas. Impor um comportamento dinâmico para um único processo é relativamente simples, pois envolve apenas a garantia do seqüenciamento das atividades (ou etapas) predeterminadas do processo. Entretanto, quando um conjunto de processos são executados simultaneamente com um intenso compartilhamento de recursos, não é mais possível determinar o comportamento global do sistema de modo puramente seqüencial. A complexidade de formular o comportamento global do sistema deve-se a coexistência de dois níveis de indeterminismo (Santos Filho, 2000): (i) O indeterminismo em relação ao tempo, pois a evolução dos estados em SMFs ocorre da decorrência de eventos (SDED), isto é, não é possível pré-determinar quando um determinado evento irá ocorrer; (ii) O indeterminismo em relação a seqüência de ocorrência de eventos, pois devido à característica dos SMFs de executarem múltiplos processos simultaneamente, compartilhando um conjunto finito de recursos, impossibilita a determinação da seqüência de disparo dos eventos no contexto global, ou seja, não é possível determinar qual evento precede outro.

Assim sendo, mesmo conhecendo-se o comportamento individual de cada processo (seqüência das atividades), não é possível prescrever o comportamento global do sistema. Esta característica de sistemas complexos presente nos SMFs pode gerar uma combinação exponencial de estados alcançáveis em função do número de processos, recursos e outras variáveis envolvidas, dificultando o controle sobre o mesmo.

A solução para abordar a complexidade do controle dos SMFs contempla três aspectos fundamentais. Em primeiro lugar é aplicado o método *Flow in Suppression* (Santos Filho, 2000). A proposta têm como base o fato de que não é possível determinar todos os estados alcançáveis do sistema porém, através de regras adicionais de controle, é possível evitar que o sistema atinja determinados estados indesejáveis. O segundo aspecto é relacionado ao elemento humano. O modelo do sistema de controle é antropocêntrico (Ito, 1991; Santos Filho, 2000). Os sistemas antropocêntricos são sistemas feitos pelo homem e para o homem (man-made systems), ou seja, o elemento humano é parte integrante do sistema. São sistemas que incorporam novas tecnologias em que o elemento humano, possuidor do conhecimento, participa na especificação de requisitos, desenvolvimento, implementação e monitoramento e, quando necessário, executando ações que interferem na dinâmica do sistema. O terceiro aspecto refere-se a abordagem quanto aos dois níveis de indeterminismo. Santos Filho (2000) propôs a divisão do sistema de controle em dois níveis (Fig. 1):

- Controle de processos: Responsável por garantir o seqüenciamento das atividades (ou etapas) dos processos, alocando recursos disponíveis;
- Controle de recursos: Responsável por gerenciar a utilização dos recursos pelos processos.

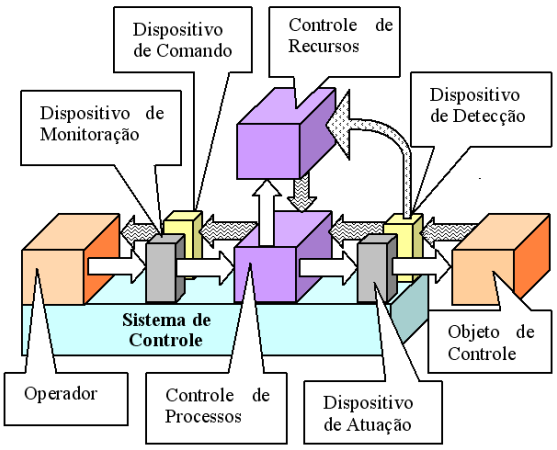


Figura 1. Arquitetura do Sistema de Controle de SMFs.

O presente trabalho adota a arquitetura do sistema de controle proposto por Santos Filho (2000) pois, além de permitir a modelagem do controle de SMFs de forma eficiente, permite a aplicação do método de deadlock avoidance de forma eficaz.

4 Proposta de Estrutura de Dados para o GAR

Conforme Ziviani (2004), programar consiste em estruturar dados e construir algoritmos, e que, a própria representação dos dados estabelece a forma pela qual estes dados serão processados.

O controle de recurso é modelado utilizando-se um grafo denominado grafo de alocação de recursos (GAR). O GAR é um grafo bipartido, não marcado e que representa a utilização de recursos pelos processos. O GAR **G = (R, A)** é um conjunto não vazio tal que **R** é um conjunto de nós, que representam os recursos $\mathbf{R = \{r_1, r_2, r_3, ..., r_n\}}$, e **A** o conjunto de arestas ou arcos orientados, que representam a alocação e requisição de recursos. Um arco orientado é um par ordenado $\mathbf{a_{ij} = (\ r_i, r_j)}$, onde $\mathbf{r_i}$ e $\mathbf{r_j}$ são elementos de **R**.

O GAR é utilizado para determinar a condição de espera circular que, neste trabalho, corresponde aos Ciclos Fechados de Espera (CFE). A figura 2 apresenta um exemplo de GAR com três CFEs formados pelos recursos: $\{r_1, r_2, r_4$ e $r_3\}$, $\{r_1, r_4$ e $r_3\}$ e $\{r_1, r_2, r_5, r_4$ e $r_3\}$.

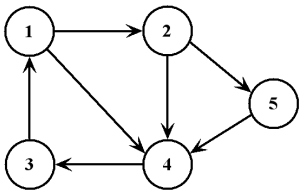
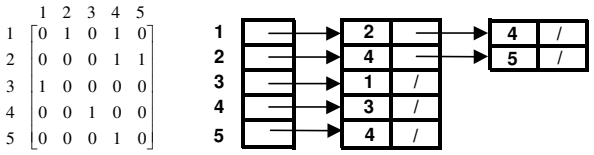


Figura 2. Exemplo gráfico de um GAR e os CFEs.

Em Nakamoto et al. (2002a) foi proposto um algoritmo para determinação dos CFEs e, a partir da sistematização do projeto do sistema de controle foi possível implementar uma ferramenta computacional (Nakamoto et al., 2003a; 2003b; 2003c). O algoritmo para a determinação dos CFEs realiza uma pesquisa primeiro na profundidade (Cormen et. al, 2003; Drozdek, 2002; Ziviani, 2004) em uma matriz de adjacência. A estratégia é pesquisar o mais profundo no grafo orientado, ou dígrafo (Drozdek, 2002), sempre que possível. Os arcos são exploradas a partir do nó descoberto mais recentemente que ainda possui arcos inexplorados saindo deste nó. Quando todos os arcos adjacentes forem explorados, o algoritmo regressa para o nó predecessor e explora os demais arcos adjacentes ainda não explorados. Um CFE é determinado no caso do nó descoberto ser um nó visitado anteriormente. O processo continua até que todos os nós do GAR sejam visitados. No presente trabalho, propõe-se adotar a representação do GAR por uma lista de adjacência em vez da matriz de adjacência. A lista de adjacência de um grafo **G = (R, A)** consiste em um arranjo **Adj** de **|R|** listas para cada nó em **R**. Para cada $\mathbf{r \in R}$, a lista **Adj[r]** consiste em todos os arcos adjacentes a **r** em **G** (Cormen et. al, 2003; Ziviani, 2004). A figura a seguir apresenta um exemplo de representação do GAR por uma matriz de adjacência e lista de adjacência.

Quando a lista de adjacência é dada mediante um formato tabular, esta representação é denominada estrela ou *Star* (Drozdek, 2002), que pode ser direta (*ForwardStar*) ou inversa (*ReverseStar*). A figuras 4 apresenta a representação direta e inversa do exemplo GAR da figura 2.



a) b)

Figura 3. Representação por matriz de adjacência (a) e lista de adjacência (b) da figura 2.

i	Origem	Destino	Arco
0	1	2	a_{12}
1	1	4	a_{14}
2	2	5	a_{25}
3	2	4	a_{24}
4	3	1	a_{31}
5	4	3	a_{43}
6	5	4	a_{54}

Origem	ForwardStar
0	-1
1	0
2	2
3	4
4	5
5	6
6	7

i	Origem	Destino	Arco
0	3	1	a_{31}
1	1	2	a_{12}
2	4	3	a_{43}
3	1	4	a_{14}
4	2	4	a_{24}
5	5	4	a_{54}
6	2	5	a_{25}

Destino	ReverseStar
0	-1
1	0
2	1
3	2
4	3
5	6
6	7

Figura 4 Representação de tabela ForwardStar e ReverseStar.

A vantagem da utilização da representação por lista de adjacência, em comparação com a matriz de adjacência adotada anteriormente, é com relação a complexidade do algoritmo. Enquanto que a matriz de adjacência possui complexidade de $\mathbf{\Theta(R^2)}$, a lista de adjacência possui complexidade de **Θ(R+A)** (Cormen et. al, 2003; Drozdek, 2002; Ziviani, 2004), o que consequentemente permitirá um melhor desempenho no tempo de processamento computacional para a determinação do CFE.

5 Algoritmo para Determinação dos CFEs

O algoritmo para determinação dos CFEs executa uma busca em profundidade em uma lista de adjacência do GAR em busca de ciclos fechados, visitando-se todos os nós, seguindo os arcos orientados em busca de nós adjacentes. A figura 5 apresenta o algoritmo proposto e as funções auxiliares (Fig. 6 e 7) que são chamados de forma recursiva.

Caso encontre um nó visitado anteriormente, descobre-se um CFE e, antes de ser armazenado em uma lista de CFEs, é verificando se o mesmo já foi descoberto anteriormente. Para atingir este objetivo, o algoritmo utiliza as seguintes estruturas de dados auxiliares:

- Pilha de armazenamento temporário de nós visitados pelo algoritmo (Cormen et. al, 2003; Drozdek, 2002; Ziviani, 2004). É uma estrutura de dados LIFO (*Last In*, *Fist Out*), isto é, o último elemento inserido na pilha será o primeiro a ser removido. O processo de inserção e remoção é realizada pelas funções *push()* e *pop()*;
- Lista de estado dos nós (vetor de status dos nós) composta de uma matriz unidimensional para armazenar as informações se o nó for visitado e se estiver na pilha.

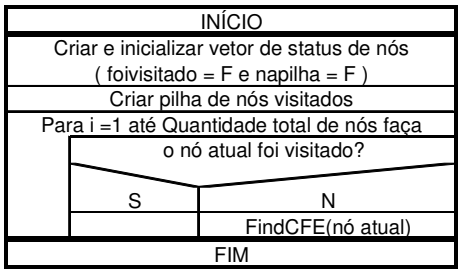


Figura 5 Algoritmo para determinação dos CFEs em uma lista de adjacência GAR.

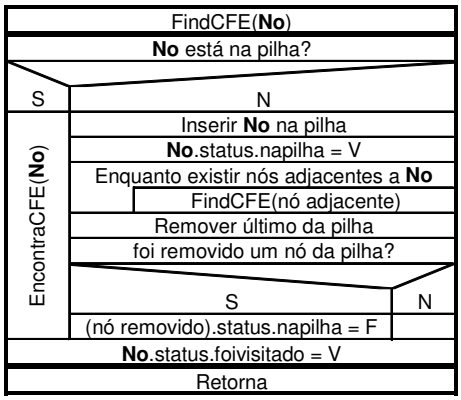


Figura 6 Função *FindCFE()*.

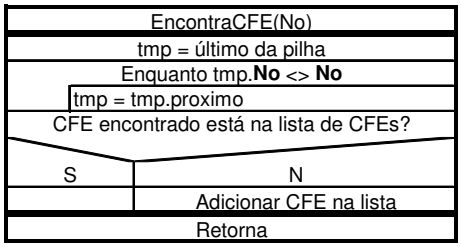
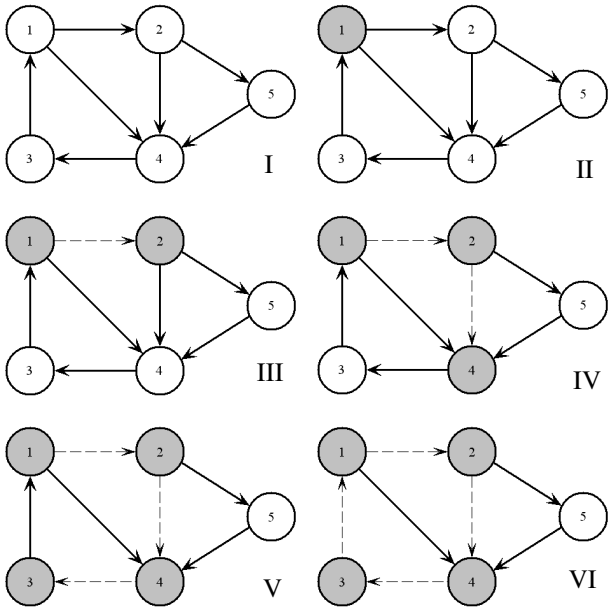


Figura 7 Função *EncontraCFE ()*.

6 Estudo de Caso

Considerando-se o exemplo da figura 2, a figura 8 apresenta a seqüência de passos executada pelo algoritmo proposto para determinação dos CFE em um GAR e a respectiva lista de estados (vetor de *status*) com os valores da seqüência I, II, III, IV, V e VI.

Observa-se que na seqüência VI, a função *FindCFE()* encontra o nó 1 (recurso R1) e o mesmo está na pilha. Neste ponto, realiza a chamada de função *EncontraCFE()* encontrando um CFE. A função *FindCFE()* retorna ao nó predecessor, ou seja, até o nó 2 (recurso R2), uma vez que os nós 3 (R3) e 4 (R4) não possuem mais arcos adjacentes.



	Recurso Status	1	2	3	4	5	6
I	*Visitado*	F	F	F	F	F	F
	Pilha	F	F	F	F	F	F
II	*Visitado*	V	F	F	F	F	F
	Pilha	V	F	F	F	F	F
III	*Visitado*	V	V	F	F	F	F
	Pilha	V	V	F	F	F	F
IV	*Visitado*	V	V	F	V	F	F
	Pilha	V	V	F	V	F	F
V	*Visitado*	V	V	V	V	F	F
	Pilha	V	V	V	V	F	F
VI	*Visitado*	V	V	V	V	F	F
	Pilha	V	V	V	V	F	F

Figura 8 Exemplo de terminação do CFE {R_1, R_2, R_4 e R_3}.

7 Conclusão

Neste trabalho foi apresentado um algoritmo para determinação dos CFEs com bases em nova representação do GAR comparado com trabalhos anteriores (Nakamoto et al. 2002a; 2002b; 2003a), ou seja, uma representação por lista de adjacência do GAR. A vantagem em adotar esta representação é quanto a redução do tempo de execução computacional do algoritmo para determinar o CFEs.

Uma vez descoberto todos os CFEs em um SMF, é possível definir as regras de controle para evitar que o sistema entre em estado de *deadlock*. Cabe salientar que este trabalho está em andamento e o próximo passo é a elaboração do algoritmo para determinação das regras adicionais de controle para SMFs com múltiplas instâncias de recursos.

Agradecimentos

Os autores agradecem a CAPES e CNPq pelo apoio a este trabalho.

Referências Bibliográficas

Banaszak, Z.A. and Krogh, B.H. (1990), Deadlock Avoidance in Flexible Manufacturing Systems with Concurrently Competing Process Flows, *IEEE Transctions on Robotics and Automation*, vol.6, pp.724-734.

Banaszak, Z.A. and Polak, M. (2002) Deadlock-free Distributed Control for Repetitive Flows, *Proceedings of the Sixth International Workshop on Discrete Event Systems*, pp.273-278, 2002.

Cassandras, C.G. (1993), Discrete Event Systems – Modeling and Performance Analysis, Richard D. Irvin, Inc. and Aksen Associetes, Incorporated Publichers, pp.1-139.

Cho, H. (1993), An Intelligent Workstation Controller for Computer Integrated Manufacturing, Doctor dissertation, Texas A&M University, pp.83-88.

Cormen, T.H., Leiserson, C.E., Rivest, E.L., Stein, C. (2003). Introduction to Algorithms, Second Edition, McGraw-Hill.

Date, C.J. (2004). Introdução a Sistemas de Bancos de Dados, Editora Campus.

Drozdek, A. (2002). Estrutura de Dados e Algoritmos em C++, Editora Pioneira Thomson Learning, São Paulo.

Edmonds, B. (1995), What is Complexity? – The Philosophy of Complexitu per se with Application to some Exemples in Evolution", Symposium: The Evolution of Complexitu, University of Brussels, Breussels, Belgium.

Ezpeleta, J., Trincas, F., Garcia Valles, F. and Colom, J.M. (2002) A Banker's Solution for Deadlock Avoidance in FMS with Flexible Routing and Multiresource States, in IEEE Transactions on Robotics and Automation, vol. 18, pp. 621-625.

Ezpeleta, J. and Recalde, L. (2004) A Deadlock Avoidance Approach for Nonsequential Resource Allocation Systems, in IEEE Transactions on Systems, Man and Cybernetics, Part A, vol. 34, pp.93-101.

Fanti, M.P., Maione, B., Mascolo, S., Turchiano, B. (1997), Event-Bases Feedback Control for Deadlock Avoidance in Flexible Production Systems, *IEEE Transaction on Robotics and Automation*, vol. 13, no.3, pp.347-363.

Fanti, M.P., Maione, B., Mascolo, S., Turchiano, B. (2000), Comparing Digraph and Petri Net Approaches to Deadlock Avoidance in FMS, *IEEE Transaction on Systems, Man, and Cybernetics*, vol. 30, no.5, pp.783-798.

Ito, Y. (1991) A Desirable Production Structure Looking Toward the 21th Century – Anthropocentric Intelligence-Bases Manufacturing, in Proceedings of XI COBEM Brazilian Congress of Mechanical Engineering, São Paulo, SP, Brazil, pp.23-32.

Kumaran, T.K., Chang, W., Cho, H., Wysk, R.A. (1994), A Structured Approach to Deadlock Detection, Avoidance and Resolution in Flexible Manufacturing Systems, *International Journal of Production Research*, vol.32, no.10, pp. 2361-2379.

Nakamoto, F.Y., Miyagi, P.E., Santos Filho, D.J. (2002a). Uma Proposta de Algoritmo para a Determinação dos Ciclos Fechados de Espera em Sistemas Produtivos Flexíveis, *XVI Congresso Brasileiro de Automática*, Natal/RN, Brasil.

Nakamoto, F.Y. (2002b). Sistematização do Projeto do Controle de Sistemas Produtivos, Dissertação de Mestrado, *Escola Politécnica da Universidade de São Paulo*, São Paulo/SP, Brasil.

Nakamoto, F.Y.; Miyagi, P.E.; dos Santos Filho, D.J. (2003a). Systematization of the Project of the Production System Control, *IEEE International Symposium on Industrial Electronics*, Vol. 2, pp. 868-873, Rio de Janeiro/RJ, Brazil.

Nakamoto, F.Y.; Miyagi, P.E.; dos Santos Filho, D.J. (2003b). Geração Automática do Algoritmo de Controle de Sistemas Produtivos Flexíveis, *VI Simpósio Brasileiro de Automação Inteligente*, Bauru/SP, Brasil.

Nakamoto, F.Y.; Miyagi, P.E.; dos Santos Filho, D.J. (2003c). Automatic Generation of the Productive System Control, *COBEM 2003 - 17th International Congress of Mechanical Engineering*, São Paulo/SP, Brasil.

Miyagi, P.E. (1996), Controle Programável – Fundamentos do Controle de Sistemas a Eventos Discretos, Ed.Edgard Blucher Ltda, São Paulo, Brazil.

Murata, T. (1998), Petri Nets: Properies, Analysis and Applications, *Procedings of the IEEE*, vol.77,no4, pp.541-580.

Peterson, J.L. (1981), Petri Net Theory and the Modeling of Systems, Prentice-Hall, 1981.

Reisig, W. (1985), Petri Nets, An Introduction", Springer-Verlag, Berlin Heidelberg.

Santos Filho, D.J., (2000), Aspectos do Projeto de Sistemas Produtivos, Tese de Livre Docência, Escola Politécnica da Universidade de São Paulo, São Paulo.

Tanenbaum, A.S., (1995), Sistemas Operacionais Modernos, LTC - Livros Técnicos e Científicos Editora S.A., Rio de Janeiro.

Tanenbaum, A.S., (1996), Computer Networks - Third Edition, Prentice Hall.

Viswanadham, N., Narahari, Y., Johnson, T.L. (1990), Deadlock Prevention and Deadlock Avoidance in Flexible Manufacturing Systems Using Petri Net Models, *IEEE Transactions on Robotics and Automation*, vol.6, no6, pp.713-723.

Ziviani, N. (2004). Projeto de Algoritmos com implementações em PASCAL e C, 2ª Edição, Editora Pioneira Thomson Learning, São Paulo.