PLANEJAMENTO DA PRODUÇÃO BASEADO NO CRITÉRIO DO MÁXIMO PARALELISMO COM RESTRIÇÕES TEMPORAIS

Lucas V. R. Alves*, Patrícia N. Pena†, Ricardo H. C. Takahashi‡

*Universidade Federal de Minas Gerais
Colégio Técnico - Setor de Instrumentação
Belo Horizonte, MG, Brasil

†Universidade Federal de Minas Gerais
Departamento de Engenharia Eletrônica
Belo Horizonte, MG, Brasil

‡Universidade Federal de Minas Gerais
Departamento de Matemática
Belo Horizonte, MG, Brasil

Emails: lucasvra@ufmg.br, ppena@ufmg.br, taka@mat.ufmg.br

Abstract— This paper deals with the planning problem in manufacturing systems, by proposing a heuristic method to find a sequence of events in a supervisor that maximizes the parallelism among equipment and also assures temporal correctness. The supervisor is obtained using the Supervisory Control Theory and implements the supremal controllable sub-language contained in the desired behavior of the system. The main objective is to find, among all executions allowed by the supervisor, the one that accumulates more tasks working at the same time during the whole production process. To do so, a function that defines, for each state of the automaton, the information of how many tasks are active is established. Them, the problem can be transformed into a problem of the longest path in a acyclic graph.

Keywords— Discrete Event Systems, Schedulling, Planning.

Resumo— Este trabalho lida com o problema de planejamento da produção em sistemas de manufatura, propondo um método heurístico de encontrar uma sequência de eventos em um Supervisor que maximiza o paralelismo entre equipamentos, garantindo também a corretude temporal. O supervisor é obtido usando a Teoria de Controle Supervisório e implementa a suprema sub-linguagem controlável contida no comportamento desejado do sistema. O principal objetivo é encontrar, entre todas as execuções permitidas pelo supervisor, aquela que acumula mais tarefas sendo executadas ao mesmo tempo durante todo o processo produtivo. Para isso, uma função que define, para cada estado do autômato, a informação sobre quantas tarefas estão ativas é definida. Dessa forma, o problema pode ser transformado em um problema de maior caminho em um grafo acíclico.

Palavras-chave— Sistemas a Eventos Discretos, Escalonamento, Planejamento.

1 Introdução

A produtividade é um dos aspectos mais importantes no contexto da automação industrial e o tempo é usualmente o recurso mais valioso. Nesse cenário, a pesquisa em técnicas de escalonamento e planejamento da produção são a chave para responder à demanda por eficiência (Wang et al., 2008).

Escalonamento de tarefas é a alocação, no tempo, de recursos finitos a tarefas, utilizando para tal um critério de otimização (Pinedo, 2012). Quando um problema de escalonamento não leva em consideração restrições quanto aos recursos que estão sendo utilizados e apenas usa restrições temporais e de precedência, ele pode ser tratado como um problema de planejamento (Ghallab et al., n.d.). A utilização de autômatos e linguagens para modelar Sistemas a Eventos Discretos (SEDs) e utilizando a Teoria de Controle Supervisório (TCS) para obter um supervisor que garante a operação segura, já fornece as restrições de precedência de eventos, permitindo que o problema de sequenciamento possa ser tratado como um problema de planejamento da produção.

Dentre os diversos formalismos para tratar SEDs, pode-se observar, na literatura, diversas técnicas que buscam minimizar o tempo de produção (*makespan*) ou maximizar o *throughput* na produção contínua, como no contexto de programação matemática (Schrijver, 1986), redes de Petri (López-Mellado et al., 2005), Autômatos Temporizados (Abdeddaïm et al., 2006), Verificação Formal (Herzig et al., 2014), entre outros.

O problema de minimização de tempo de produção é, no caso geral, não polinomial (Garey and Johnson, 1979), tornando boa parte dos problemas industriais computacionalmente intratáveis. Desse modo, usualmente são aplicadas heurísticas e metaheurísticas na solução desses problemas (Oliveira et al., 2013).

Neste trabalho, dá-se continuidade à ideia apresentada em (Alves et al., 2015), de, ao invés de minimizar o tempo de produção, maximiza-se o número de tarefas executadas em paralelo. Naquela abordagem, o algoritmo leva em consideração apenas as restrições lógicas, ou seja, as restrições de precedência, mas não garante que a a

sequência obtida possa ser executada no tempo, devido à posição dos eventos não controláveis. Dessa maneira, neste trabalho propõe-se uma expansão do algoritmo de (Alves et al., 2015) levando em consideração o tempo de execução, apenas para garantir a corretude temporal da mesma.

A adição dessas restrições temporais ao algoritmo torna o problema tão difícil quanto a minimização de tempo de produção, de forma que torna-se necessário uma solução heurística, que não garante a maximização do paralelismo, mas em geral gera boas soluções.

2 Preliminares

Seja Σ um conjunto finito não vazio de eventos, chamado de alfabeto. O comportamento de um SED é modelado como cadeias sobre Σ. O Fechamento Kleene Σ^* é o conjunto de todas as cadeias que podem ser construídas utilizando os símbolos do alfabeto Σ, incluindo a cadeia vazia ϵ. Um subconjunto $L \subseteq \Sigma^*$ é chamado linguagem. A concatenação de cadeias $s, u \in \Sigma^*$ é representada como su.

Uma cadeia $s \in \Sigma^*$ é chamada *prefixo* de $t \in \Sigma^*$, escrito $s \leq t$, se existe $u \in \Sigma^*$ tal que $su = t$. O prefixo fechamento $\overline{L}$ de uma linguagem $L \subseteq \Sigma^*$ é o conjunto de todos os prefixos de cadeias em L, isto é, $\overline{L} = \{\, s \in \Sigma^* \mid s \leq t \text{ para algum } t \in L \,\}$.

Definição 1 *Um autômato finito determinístico é um 5-tupla $G = (Q, \Sigma, \delta, q_0, Q_m)$ onde Q é um conjunto finito de estados, Σ é um conjunto finito de eventos (alfabeto), $\delta : Q \times \Sigma \to Q$ é a função de transição, $q_0 \in Q$ é o estado inicial e $Q_m \subseteq Q$ é o conjunto de estados marcados.*

A função de transição pode ser estendida para reconhecer cadeias sobre Σ^* como $\delta(q, \sigma s) = q'$ se $\delta(q, \sigma) = x$ e $\delta(x, s) = q'$.

As linguagens gerada e marcada são, respectivamente, $\mathcal{L}(G) = \{s \in \Sigma^* | \delta(q_0, s) = q' \wedge q' \in Q\}$ e $\mathcal{L}_m(G) = \{s \in \Sigma^* | \delta(q_0, s) = q' \wedge q' \in Q_m\}$.

A função de eventos ativos, definida por $\Gamma : Q \to 2^{\Sigma}$, é, dado um estado q, o conjunto de eventos $\sigma \in \Sigma$ para os quais $\delta(q, \sigma)$ é definido.

Definição 2 *Seja $G_1 = (Q_1, \Sigma_1, \delta_1, q_{01}, Q_{m1})$ e $G_2 = (Q_2, \Sigma_2, \delta_2, q_{02}, Q_{m2})$. O produto síncrono de G_1 e G_2 é:*

$$G_{12} = (Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, \delta_{12}, (q_{01}, q_{02}), Q_{m1} \times Q_{m2})$$

onde

$$\delta((q_1, q_2), e) = \begin{cases} (\delta_1(q_1, e), \delta_2(q_2, e)), & \text{se } e \in \Gamma_1(q_1) \cap \Gamma_2(q_2) \\ (\delta_1(q_1, e), q_2), & \text{se } e \in \Gamma_1(q_1) \backslash \Sigma_2 \\ (q_1, \delta_2(q_2, e)), & \text{se } e \in \Gamma_2(q_2) \backslash \Sigma_1 \\ \text{indefinido}, & \text{caso contrário.} \end{cases}$$

e $\Gamma_{1||2}(q_1, q_2) = [\Gamma_1(q_1) \cap \Gamma_2(q_2)] \cup [\Gamma_1(q_1) \backslash \Sigma_2] \cup [\Gamma_2(q_2) \backslash \Sigma_1]$.

A Teoria de Controle Supervisório é um método formal, baseado na teoria de linguagens e autômatos, para o cálculo sistemático de supervisores. O sistema a ser controlado é chamado *planta*, o controlador é o agente chamado *supervisor* e o problema de controle é encontrar um supervisor que garanta especificações de desempenho e segurança da maneira minimamente restritiva.

A planta é modelada por um autômato $G = (Q, \Sigma, \delta, q_0, Q_m)$ e $\Sigma = \Sigma_c \cup \Sigma_{nc}$ onde Σ_c é o conjunto de eventos controláveis, que podem ser desabilitados por um agente externo, e Σ_{nc} é o conjunto de eventos não controláveis, que não podem ser desabilitados por um agente externo, como o supervisor. A planta representa o modelo lógico de um SED, o comportamento do sistema sem nenhuma ação de controle. O papel de um supervisor S é regular o comportamento da planta e alcançar um comportamento K desabilitando eventos controláveis.

Seja E um autômato que representa especificações imposta à planta G. Dizemos que $K = \mathcal{L}_m(G||E) \subseteq \mathcal{L}_m(G)$ é controlável com respeito a G se $\overline{K}\Sigma_{nc} \cap \mathcal{L}(G) \subseteq \overline{K}$. Existe um supervisor não bloqueante V para G tal que $\mathcal{L}_m(V/G) = K$ se, e somente se, K é controlável com respeito à G. Se K não satisfaz a condição, então a suprema sub-linguagem controlável e não bloqueante $sup\mathcal{C}(K, G)$ pode ser sintetizada, representando o supervisor não bloqueante minimamente restritivo. Para G e K, um supervisor monolítico, representado por um autômato S, pode ser computado para representar $sup\mathcal{C}(K, G)$ de modo que $\mathcal{L}_m(S) = sup\mathcal{C}(K, G) \subseteq K$ (Ramadge and Wonham, 1989).

As linguagens gerada e marcada de uma planta G sobre a ação de um supervisor S são, respectivamente, $\mathcal{L}(S/G)$ e $\mathcal{L}_m(S/G) \subseteq \mathcal{L}(S/G)$. Um supervisor S é dito não bloqueante quando $\overline{\mathcal{L}_m(S/G)} = \mathcal{L}(S/G)$.

A partir daqui, chama-se supervisor um autômato S tal que $S = S||G$, ou seja, a dinâmica do supervisor engloba a dinâmica da planta sob controle.

3 Resultados Anteriores

A ideia de tarefa é introduzida como uma propriedade de um estado de um autômato finito determinístico. Nesse contexto, um determinado estado pode ter zero ou mais tarefas sendo executadas. Caso o objetivo seja apenas maximizar o número de máquinas funcionando, o número de tarefas associado a cada estado deve ser 0 se é um estado onde a máquina está ociosa ou 1 se a máquina está trabalhando. Se a máquina possui um

paralelismo intrínseco, como um processador com múltiplos núcleos, o número de tarefas associado a cada estado pode ser qualquer valor inteiro não negativo.

Uma maneira de representar o número de tarefas ativas em um estado é por meio da definição de uma função, chamada função de tarefas ativas:

Definição 3 *Seja* $G = (Q, \_\_, \_\_, \_\_, \_\_)$ *um autômato. a função de tarefas ativas associada à* G*, definida como* $f_{ta} : Q \to \mathbb{Z}^*$*, é tal que, para cada estado* $q \in Q$*, associa um número inteiro não negativo de tarefas.*

Usualmente é mais simples estabelecer o número de tarefas nos autômatos das plantas, ao invés do autômato do supervisor, tornando então necessário a definição de uma nova função de tarefas ativas para a composição das plantas.

Definição 4 *Sejam* $G_1 = (Q_1, \_\_, \_\_, \_\_, \_\_)$ *e* $G_2 = (Q_2, \_\_, \_\_, \_\_, \_\_)$ *autômatos e sejam* f_{ta_1} *e* f_{ta_2} *as funções de tarefas ativas associadas, respectivamente, à* G_1 *e* G_2*. A função de tarefas ativas do autômato* $G_{1||2} = (Q_1 \times Q_2, \_\_, \_\_, \_\_, \_\_)$ *é definida como:*

$$f_{ta_{1||2}}((q_1, q_2)) = f_{ta_1}(q_1) + f_{ta_2}(q_2)$$

onde $q_1 \in Q_1$ *e* $q_2 \in Q_2$*.*

Especificações não executam tarefas, visto que são apenas restrições de comportamento, e dessa forma, para todas as especificações, a função de tarefas ativas é sempre zero para todos os estados. Esse mesmo procedimento pode ser utilizado quando determinada planta não for de interesse no processo de otimização.

Para se avaliar o paralelismo de um sequência finita, é necessária a definição da função acumulativa de tarefas ativas:

Definição 5 *Seja* $S = (Q, \Sigma, \delta, \_\_, \_\_)$ *um supervisor. A função acumulativa de tarefas ativas,* $F_{ta} : Q \times \Sigma^* \to \mathbb{Z}^*$*, de* S *é definida como:*

$$\begin{cases} F_{ta}(q, \epsilon) & = f_{ta}(q) \\ F_{ta}(q, \sigma s) & = f_{ta}(q) + F_{ta}(\delta(q, \sigma), s) \end{cases}$$

onde $\sigma s \in \Sigma^*$*.*

Quanto maior o valor da função acumulativa de tarefas ativas determinada sequência apresenta, maior o seu paralelismo. A função acumulativa de tarefas ativas somente deve ser usada para a comparação de sequências de mesmo tamanho.

4 Principais Resultados

Para adicionar informações temporais ao máximo paralelismo, se faz necessário avaliar o tempo até que determinado evento ocorra em um supervisor dada a sequência já executada até então:

Definição 6 *Seja* $S = (\_\_, \Sigma, \_\_, \_\_, \_\_)$ *um supervisor. A função temporal,* $f_T : \Sigma^* \times \Sigma \to \mathbb{R}^*$*, de* S *é definida como:*

$$f_T(s, \sigma) = \begin{cases} t & \text{se } \delta(s, \sigma)! \\ \infty & \text{caso contrário} \end{cases}$$

onde, para um evento $\sigma \in \Sigma$ *e uma sequência* $s \in \mathcal{L}(S/G)$*,* t *é o tempo até que o evento* σ *ocorra, dado que a sequência* s *acabou de ser executada.*

Outra informação temporal importante é quanto tempo é necessário para executar uma sequência de eventos. Para isso, podemos expandir a função temporal para retornar o tempo total de execução de uma sequência como:

Definição 7 *Seja* $S = (\_\_, \Sigma, \_\_, \_\_, \_\_)$ *um supervisor. A função temporal expandida,* $f_T : \Sigma^* \to \mathbb{R}^*$*, de* S *é definida como*

$$\begin{cases} f_T(\epsilon) & = 0 \\ f_T(s\sigma) & = f_T(s) + f_T(s, \sigma) \end{cases}$$

Dessa forma, precisa-se encontrar uma sequência da linguagem marcada do supervisor que maximize o paralelismo, mas também apresente um tempo de execução menor que infinito. O principal problema dessa proposta é que esse objetivo é tão difícil quanto o de minimização de *makespan*, de forma que a aplicação do paralelismo não traz ganhos significativos em desempenho.

Dessa maneira, o máximo paralelismo com restrições temporais é executado de maneira heurística.

4.1 Definição do Problema

Seja S o supervisor de um sistema de produção $G = ||_{k=1}^{N} G_k$, onde G_k é uma planta que compõe o sistema e seja f_{ta} a função de tarefas ativas do sistema sob supervisão S/G. Seja n o número de eventos necessários para a produção de um lote de produtos e seja o universo de busca o conjunto $L = \{s \in \mathcal{L}_m(S/G) : n = |s| \wedge f_T(s) \neq \infty\}$.

Neste caso, o problema de planejamento da produção pode ser definido como um problema de otimização na forma:

$$s^* = \underset{s \in L}{\operatorname{argmax}} \; F_{ta}(s)$$

onde s^* é uma das sequências que maximizam o número de tarefas executando paralelamente no sistema.

O problema a ser resolvido é similar ao problema do máximo paralelismo, porém a informação relativa à função temporal impede que o problema possa ser tratado, de maneira exata, como uma problema de maior caminho em um grafo acíclico, porque as restrições temporais fazem com

que o caminho utilizado para alcançar certo estado influencie no melhor caminho entre aquele estado e o objetivo.

Neste artigo, será utilizado um algoritmo de maior caminho, desconsiderando que um mesmo estado, alcançado de maneiras diferentes, possui diferentes futuros, tomando um passo guloso escolhendo apenas o maior caminho até aquele estado. Dessa forma, o algoritmo de máximo paralelismo com restrições temporais passa a ser um algoritmo heurístico, mas que garante a execução temporal da sequência obtida.

4.2 Manipulação do Problema

O problema de otimização definido anteriormente pode ser resolvido como um problema de maior caminho em um grafo acíclico, onde o peso de um transição é o número de tarefas ativas no estado de destino. A existência de ciclos no autômato do supervisor impede uma aplicação direta de algoritmos de maior caminho, visto que percorrer um ciclo sempre permite aumentar o caminho, adiando indefinidamente o fim do algoritmo.

Para tornar o autômato do supervisor em um grafo acíclico com profundidade n, faz-se necessário compor o supervisor com um autômato, aqui chamado de autômato de desenrolamento G_d, como mostrado na Figura 1, onde $\mathcal{L}_m(G_d) = \{s \in \Sigma^* : n = |s|\}$ onde Σ é o conjunto de eventos de S/G e n é o número de eventos necessários para a produção de um lote.

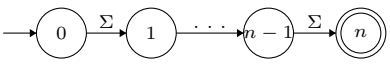


Figura 1: Autômato de desenrolamento com um conjunto de eventos Σ e profundidade n genérico

O autômato resultante da operação de composição é acíclico e permite a execução de um algoritmo de maior caminho.

As informações necessárias para a execução do algoritmo são o conjunto de estados do supervisor (Q), a função de transição de estados (δ), a função de eventos ativos (Γ), o estado inicial (q_0) e a profundidade de busca n. Como resultado, o algoritmo devolve a estrutura *path* que armazena o maior caminho entre o estado inicial $(q_0, 0)$ e qualquer outro estado alcançado na busca.

5 Algoritmo

Conhecendo o estado de destino e a profundidade desejada, basta usar o mapeamento *path* (estrutura de dados que associa cada estado ao máximo caminho que o alcança, a partir do estado inicial) para encontrar a sequência de eventos que maximiza o caminho, ou seja, maximiza o paralelismo entre tarefas.

Algoritmo 1: Máximo Paralelismo com Restrições Temporais

```
1  foreach q in Q do
2      for i ← 0 to n do
3          if (q, i) = (q_0, 0) then
4              d[(q, i)] ← 0
5              path[(q, i)] ← ϵ
6              time[(q, i)] ← 0
7          else
8              d[(q, i)] ← −∞
9              path[(q, i)] ← ∅
10             time[(q, i)] ← ∞
11         end
12     end
13 end
14
15 F ← (q_0, 0)
16
17 while |F| ≠ 0 do
18     (q, i) ← F
19     if i = n then continue
20
21     foreach σ in Γ(q) do
22         v ← δ(q, σ)
23         t ← f_T(path[(q, i)]σ)
24         if t = ∞ then continue
25
26         if (v, i + 1) ∉ F then
27             F ← (v, i + 1)
28         end
29         w ← f_ta(v)
30         d_q ← d[(q, i)]
31         d_v ← d[(v, i + 1)]
32         if d_q + w > d_v or (d_q + w = d_v and
             t < time[(v, i + 1)]) then
33             d[(v, i + 1)] ← d[(q, i)] + w
34             path[(v, i + 1)] ← path[(q, i)] · σ
35             time[(v, i + 1)] ← t
36         end
37     end
38 end
```

Entre as linhas 1 e 13 as estruturas de dados d, *path* e *time* são inicializadas. Na linha 15, a fila F, que armazena os vértices em ordem topológica, é inicializada.

Entre as linhas 17 e 38, o primeiro elemento é retirado da fila, e entre as linhas 21 e 37 os vértices adjacentes são analisados a fim de verificar se eles aumentam o caminho, e por consequência o paralelismo.

5.1 Complexidade do Algoritmo

O algoritmo de busca pelo máximo paralelismo processa todos os vértices e executa um laço percorrendo os vértices adjacentes. O número de estados no grafo acíclico pode ser estimado como $(n+1)|Q|$, onde Q é o conjunto de estados do supervisor, e o número de arestas pode ser estimado como $n|A|$, onde A é o conjunto de todas as transições existentes no supervisor. A complexidade da função f_T é da ordem de $\mathcal{O}(k)$, onde $k = 0, 1, ..., n$ é o número de eventos ocorridos até o estado a ser analisado. Dessa forma, a complexidade do algoritmo, no pior caso, é dada por $\mathcal{O}(n^2|Q| + n|A|)$.

5.2 Testes de Execução do Algoritmo

O caso de estudo, onde será testado o algoritmo, é o Sistema Flexível de Manufatura (SFM),

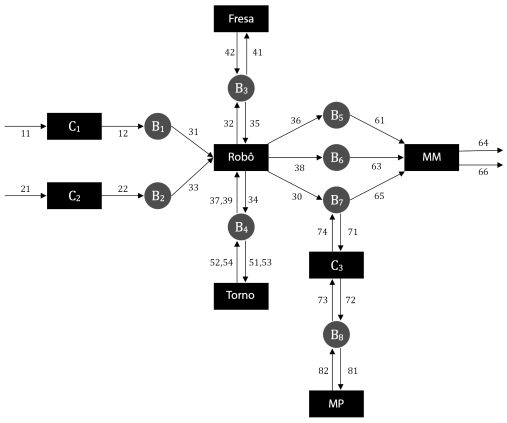


Figura 2: Diagrama do Sistema Flexível de Manufatura

como mostrado na Figura 2, também utilizado em (Oliveira et al., 2013). O SFM produz dois tipos de produtos a partir de blocos e tarugos, sendo eles, um bloco com pino cônico no topo (produto A) e um bloco com um pino cilíndrico pintado (produto B).

A planta consiste de oito máquinas, sendo elas, uma fresadora, um torno, um dispositivo de pintura, uma máquina de montagem, três esteiras e um robô. Além disso, são considerados oito *buffers* que agem como depósitos intermediários. Na Figura 3 estão representados os autômatos que modelam a dinâmica das máquinas que compõem o SFM. O número de tarefas ativas em cada estado está representado no nome dos estados, de forma que para um estado (k, i), temos $f_{ta}((k, i)) = i$.

Dado que se faz necessário avaliar o tempo, é necessário definir a duração das tarefas executadas pelas máquinas. Neste trabalho, o tempo de execução de uma tarefa é definido como o intervalo de tempo entre a ocorrência de um evento controlável e o evento não controlável correspondente. Essa abordagem considera que os eventos não controláveis são respostas da planta à execução de eventos controláveis.

Na Tabela 1 estão presentados os intervalos de tempo entre a execução de um evento controlável e a ocorrência de seu respectivo evento não controlável de resposta. Além do especificado na tabela, existe uma especificidade do SFM onde os eventos controláveis 63 e 65 somente podem ocorrer, no mínimo, 15 unidades de tempo após o evento 61.

A produção da base de ambos os produtos (A e B) é realizada pela execução dos seguintes pares de eventos (com os eventos controláveis na ordem apresentada):

$$b = \{(11, 12), (31, 32), (41, 42), (35, 36), (61)\}$$

Para produzir o pino do produto A, os pares executados são:

$$p_a = \{(21, 22), (33, 34), (51, 52), (37, 38), (63, 64)\}$$

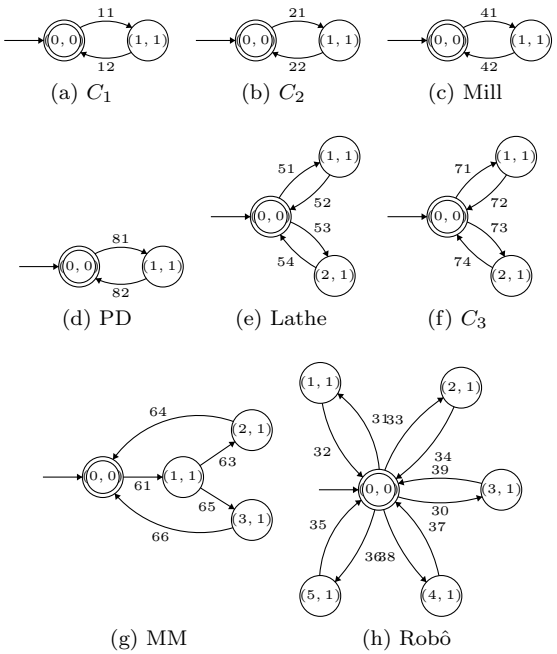
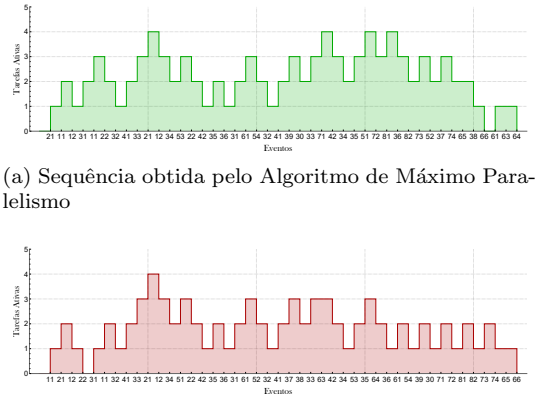


Figura 3: Plantas do Sistema Flexível de Manufatura



(a) Sequência obtida pelo Algoritmo de Máximo Paralelismo

(b) Sequência obtida por um algoritmo guloso

Figura 4: Sequências de eventos para a produção de um produto A e um produto B

Finalmente, para produzir um pino do produto B é necessário executar os pares:

$$p_b = \{(21, 22), (33, 34), (53, 54), (39, 30), (71, 72), (81, 82), (73, 74), (65, 66)\}$$

Utilizando o supervisor monolítico para o SFM, obtido pela Teoria de Controle Supervisório convencional, composto por 45.504 estados e 200.124 transições, em conjunto com as especificações temporais, executando o algoritmo para uma profundidade de 44 eventos (um produto A e um produto B), a sequência obtida está mostrada na Figura 4a. A sequência obtida pelo algoritmo, chamada s_o produz um par de produtos em 286 *u.t.*. Por motivo de comparação, foi desenvolvido um algoritmo guloso que, a cada passo, escolhe o evento que minimiza f_T, e para as mes-

Eventos Controláveis	Eventos Não Controláveis	Intervalo de Tempo $[u.t.]$
11	12	26
21	22	26
31	32	22
33	34	20
35	36	17
37	38	25
39	30	21
41	42	31
51	52	39
53	54	33
63	64	27
65	66	27
71	72	26
73	74	26
81	82	25

Tabela 1: Intervalo de tempo entre pares de eventos

mas condições, o algoritmo encontrou a sequência chamada s_g, que produz um par de produtos em 321 $u.t.$ e está mostrada na Figura 4b. Comparando o paralelismo entre as sequências, $F_{ta}(s_o) = 96$ e $F_{ta}(s_g) = 79$, indicando que s_o apresenta uma paralelismo acumulado maior que s_g.

Um detalhe importante é que existem múltiplas sequências com o mesmo paralelismo que podem ser encontradas pelo algoritmo, gerando diferentes tempos de produção.

Para avaliar o desempenho do algoritmo para profundidades de busca maiores, foi realizado um teste para diferentes tamanhos de lote. Cada lote foi executado 30 vezes e a média do tempo de execução está mostrada na Figura 5. Como pode ser observado, o algoritmo tem tempo de execução quadrático, mas ainda assim é muito rápido, de modo que um lote de 25 pares de produtos, ou seja, uma sequência de 1100 eventos, é otimizada em menos de 2 segundos.

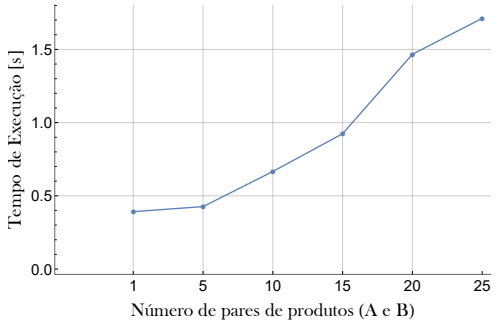


Figura 5: Avaliação da execução do algoritmo para lotes de diferentes tamanhos

6 Conclusão

Neste artigo foi desenvolvido um método de obtenção de cadeias que maximizam o paralelismo entre tarefas. A busca é lógica e utiliza o tempo apenas para garantir que a sequência obtida é factível do ponto de vista temporal.

Apesar de não garantir a otimalidade, ou seja, o máximo de paralelismo, a solução apresentada representa uma evolução ao trabalho anterior que não garante a factibilidade temporal e, ainda assim, gera resultados consistentes e rápidos.

Ainda é possível melhorar o desempenho computacional do algoritmo adicionando restrições quanto ao número de vezes que cada evento controlável pode ocorrer, reduzindo o *branching factor*.

Agradecimentos

O presente trabalho foi realizado com o apoio financeiro da Fapemig, CAPES - Brasil e CNPq.

Referências

Abdeddaïm, Y., Asarin, E. and Maler, O. (2006). Scheduling with Timed Automata, *Theoretical Computer Science* **354**(2): 272 – 300.

Alves, L., Bravo, H., Pena, P. and Takahashi, R. (2015). Escalonamento da produção baseado no critério de máximo paralelismo em sistemas a eventos discretos, *XII Simpósio Brasileiro de Automação Inteligente (SBAI)*, pp. 594–599.

Garey, M. R. and Johnson, D. S. (1979). *Computers and intractability*, W.H. Freeman.

Ghallab, M., Nau, D. and Traverso, P. (n.d.). *Automated Planning Theory and Practice*, Elsevier.

Herzig, A., de Menezes, M. V., de Barros, L. N. and Wassermann, R. (2014). On the revision of planning tasks, *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, pp. 435–440.

López-Mellado, E., Villanueva-Paredes, N. and Almeyda-Canepa, H. (2005). Modelling of batch production systems using petri nets with dynamic tokens, *Mathematics and Computers in Simulation* **67**(6): 541 – 558.

Oliveira, A. C., Costa, T. A., Pena, P. N. and Takahashi, R. H. (2013). Clonal selection algorithms for task scheduling in a flexible manufacturing cell with supervisory control, *Evolutionary Computation (CEC), 2013 IEEE Congress on*, IEEE, pp. 982–988.

Pinedo, M. L. (2012). *Scheduling: Theory, Algorithms, and Systems*, 3rd edn, Springer Publishing Company, Incorporated.

Ramadge, P. J. G. and Wonham, W. M. (1989). The Control of Discrete Event Systems, *Proc. of the IEEE* **77**(1): 81–98.

Schrijver, A. (1986). *Theory of Linear and Integer Programming*, John Wiley & Sons, Inc., New York, NY, USA.

Wang, W., Yuan, C. and Xiaobing, L. (2008). A fuzzy approach to multi-product mixed production job shop scheduling algorithm, *Fuzzy Systems and Knowledge Discovery, 2008. FSKD '08. Fifth International Conference on*, Vol. 1, pp. 95–99.