Planejamento da Produção em Sistemas a Eventos Discretos usando Heurística

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

* Colégio Técnico, Universidade Federal de Minas Gerais, MG
(email: lucasvra@ufmg.br)

** Departamento de Engenharia Eletrônica, Universidade Federal de Minas Gerais, MG
(e-mail: ppenna@ufmg.br)

*** Departamento de Matemática, Universidade Federal de Minas Gerais, MG
(e-mail: taka@ufmg.br)

Abstract: In this work we address the problem of production planning in Discrete Event Systems by applying a pruning heuristic to an exact algorithm of minimization of makespan. The heuristic assumes that controllable events usually start tasks and non-controllable events model the end of tasks, indicating that it would be better to execute a controllable event than to wait for the occurrence of a non-controllable event, reducing the branching factor of the exact algorithm.

Resumo: Neste trabalho tratamos o problema de planejamento da produção em Sistemas a Eventos Discretos aplicando uma heurística de poda a um algoritmo exato de minimização de *makespan*. A heurística parte do princípio de que, usualmente, eventos controláveis iniciam tarefas e eventos não-controláveis modelam o fim de tarefas, indicando que seria melhor executar um evento controlável do que esperar a ocorrência de um evento não-controlável, reduzindo o *branching factor* do algoritmo exato.

Keywords: Discrete Event Systems; Supervisory Control Theory; Production Planning

Palavras-chaves: Sistemas a Eventos Discretos; Teoria de Controle Supervisório; Planejamento da Produção.

1. INTRODUÇÃO

O problema de planejamento da produção é de extrema importância pois está diretamente relacionado à eficiência de sistemas industriais. Usualmente existe uma infinidade de maneiras de produzir a mesma coisa, simplesmente alterando as sequências operacionais e mesmo no subconjunto das sequências operacionais válidas e seguras, existem sequências que produzem a mesma coisa em mais ou menos tempo, ou seja, dentre as inúmeras maneiras pelas quais é possível coordenar o funcionamento da planta garantindo segurança de operação, alguns modos de operação são mais eficientes que outros quanto ao tempo de produção.

A área de pesquisa que se interessa em determinar a alocação de recursos finitos às tarefas é o Escalonamento de Tarefas (Pinedo, 2012). Problemas de escalonamento onde não se leva em consideração restrições quanto aos recursos que estão sendo utilizados e apenas usa restrições temporais e de precedência, podem ser tratados como problemas de planejamento (Ghllab et al., 2005).

Neste trabalho trata-se o problema de planejamento da produção no contexto de Sistemas a Eventos Discretos (SED), modelados por linguagens e autômatos e a Teoria de Controle Supervisório (TCS). Desta forma, o autômato sobre o qual se realiza o processo de otimização (supervi-

sor) já apresenta as garantias de operação segura e fornece as restrições de precedência de eventos.

Na área de Sistemas a Eventos Discretos, existem diversas abordagens (Kobetski and Fabian, 2006; Pinha et al., 2011; Su et al., 2012) que buscam tratar o problema de encontrar sequências de operação que minimizam o *makespan* (produção em lotes) ou maximizam o *throughput* (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; Malik and Pena, 2018), Algoritmos Evolutivos (Pena et al., 2016; Costa et al., 2018; Rafael and Pena, 2018; Rafael et al., 2018), entre outros. Existem também técnicas construtivas, usando planejamento priorizado Ware and Su (2016) e projeção sequencial de linguagem Ware and Su (2017a,b).

O problema de minimização do tempo de produção é, no caso geral, não polinomial (Garey and Johnson, 1979), tornando os problemas de planejamento da produção exata em sistemas industriais inviáveis na grande maioria dos casos. Dessa forma, usualmente, utilizam-se heurísticas e metaheurísticas para tratar esse problema (Pena et al., 2016).

Outra abordagem para tratar o problema de planejamento em Sistemas a Eventos Discretos se baseia em maximizar o paralelismo nas sequências (Alves et al., 2015, 2016b),

considerando que existe alguma correlação entre a maximização do paralelismo e a minimização do tempo. A ideia de paralelismo, sem levar em consideração o tempo, tem como desvantagem o fato de não garantir que as melhores sequências necessariamente podem ser executadas visto que a posição dos eventos não-controláveis na sequência não necessariamente será temporalmente factível, ou seja, restrições temporais físicas da planta impedem que esse comportamento, mesmo que logicamente coerente, ocorra. Nesse sentido foi proposta uma melhoria do algoritmo, inserindo restrições temporais (Alves et al., 2016a), que garantem que a posição dos eventos controláveis será obedecida quando os tempos são determinísticos.

A ideia de maximização do paralelismo pode ser usada como uma heurística para algoritmos puramente temporais visto que, em geral, existe uma clara relação entre os eventos controláveis e o início de uma tarefa e os eventos não-controláveis e o fim de uma tarefa. Em geral, faz sentido sempre executar um evento controlável do que aguardar que um evento não-controlável ocorra, permitindo, então, estabelecer uma heurística de poda que simplifica o problema geral de planejamento.

É importante notar que essa heurística somente remove ramos da busca que, em princípio, não são vantajosos visto que representam uma espera ociosa da planta. Dessa forma, em muitos casos, onde essa espera não faz sentido, o resultado obtido será o resultado exato.

Esse trabalho está organizado da seguinte forma. Na Seção 2 estão descritos os conceitos preliminares e principais definições que são base para o desenvolvimento do trabalho, a Seção 3 apresenta os principais resultados e o algoritmo desenvolvido. A Seção 4 consiste em um exemplo de aplicação do algoritmo. As conclusões e comentários finais estão na Seção 5.

2. PRELIMINARES

Seja Σ um conjunto finito não vazio de símbolos, chamado de alfabeto. Qualquer sequência de símbolos de Σ é denominado de cadeia. 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 . O número de símbolos em uma cadeia s é denotado por $|s|$.

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 $\bar{L}$ de uma linguagem $L \subseteq \Sigma^*$ é o conjunto de todos os prefixos de cadeias em L , isto é, $\bar{L} = \{s \in \Sigma^* \mid s \leq t \text{ para algum } t \in L\}$.

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), $\delta : Q \times \Sigma \rightarrow Q$ 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 por um autômato G são, respectivamente, $\mathcal{L}(G) = \{s \in \Sigma^* \mid \delta(q_0, s) = q' \wedge q' \in Q\}$ e $\mathcal{L}_m(G) = \{s \in \Sigma^* \mid \delta(q_0, s) = q' \wedge q' \in Q_m\}$.

A função de eventos ativos, definida por $\Gamma : Q \rightarrow 2^\Sigma$, 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

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

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

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 segurança da maneira minimamente restritiva.

A planta do sistema 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 impostas à planta G . Dizemos que $K = \mathcal{L}_m(G||E) \subseteq \mathcal{L}_m(G)$ é controlável com respeito a G se $\bar{K} \Sigma_{nc} \cap \mathcal{L}(G) \subseteq \bar{K}$. 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 a 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 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 sob 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 $\mathcal{L}_m(S/G) = \mathcal{L}(S/G)$.

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. PRINCIPAIS RESULTADOS

O algoritmo de minimização de *makespan* exato é extremamente custoso computacionalmente, o que inviabiliza sua aplicação para problemas de médio e grande porte, bem como para grandes lotes de produção. Uma maneira de simplificar o problema é aplicar uma heurística de poda, evitando a espera por eventos não-controláveis.

Partindo do princípio de que, usualmente, eventos controláveis iniciam uma tarefa e eventos não-controláveis indicam o fim de uma tarefa, não é interessante deixar de executar um evento controlável para executar um evento não-controlável, visto que significaria esperar que uma máquina finalize sua operação enquanto poderia estar ligando outras máquinas em paralelo. Dessa maneira, a heurística aqui desenvolvida se resume em, dado um estado, explorar apenas as transições relativas a eventos controláveis, a menos que um evento não-controlável tenha que ser obrigatoriamente executado ou não haja eventos controláveis possíveis no estado. Um evento controlável deve ser obrigatoriamente executado pelo algoritmo apenas quando o tempo até que ele ocorra seja zero ou algum evento controlável gere um acréscimo de tempo superior ao tempo para a ocorrência do evento não-controlável.

3.1 Definições

Para realizar-se uma análise temporal do sistema, bem como computar os tempos de produção, faz-se necessário a definição de uma função que avalia o tempo até que um determinado evento seja executado:

Definição 3. Seja $S = (_, \Sigma, _, _, _)$ um supervisor. A função temporal $f'_T : \Sigma^* \times \Sigma \rightarrow \mathbb{R}^*$ de S é definida, tal que, dado um evento $\sigma \in \Sigma$ e uma sequência $s \in \mathcal{L}(S/G)$, $f'_T(s, \sigma) = t$, onde t é o tempo até que o evento σ ocorra, considerando que a sequência s acabou de ser executada. Se o evento σ não pode ser executado após a sequência s , temos $f'_T(s, \sigma) = \infty$.

Usualmente, a função temporal é implementada por meio de simulação de sistemas a eventos discretos (Cassandras and Lafortune, 2008).

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.

Definição 4. A função temporal expandida $f_T : \Sigma^* \rightarrow \mathbb{R}^*$ 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 apresente o menor tempo de execução.

3.2 Definição do Problema

Seja S o supervisor de um sistema de produção $G = \bigcup_{k=1}^N G_k$, onde G_k é uma planta que compõe o sistema. Seja n o número de eventos necessários para a produção de um lote de produtos e seja o universo de busca o conjunto $L = \{s \in \mathcal{L}_m(S/G) : |s| = n \wedge f_T(s) \neq \infty\}$. É importante notar que, em alguns casos, o universo de busca deve ser adaptado, quando, por exemplo, um lote de produtos puder ser produzido por cadeias de tamanhos diferentes.

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

$$s^* = \underset{s \in L}{\operatorname{argmin}} f_T(s)$$

Onde s^* é uma das sequências que minimizam o tempo de produção (*makespan*) de um lote de produto. É possível

que exista mais de uma sequência que minimize o *makespan*.

3.3 Algoritmo

O Algoritmo de Minimização de Tempo Heurístico (MTH), apresentado no Algoritmo 1, busca minimizar o tempo de produção seguindo, indiretamente, a ideia de maximização do paralelismo. Como, usualmente, a execução de eventos controláveis aumenta o número de dispositivos ligados, o MTH é um algoritmo heurístico orientado ao tempo que somente executa eventos não-controláveis quando não existem eventos controláveis a serem executados, indiretamente aumentando o paralelismo e reduzindo o tempo de produção.

Algoritmo 1: Algoritmo de Minimização de Tempo Heurístico (MTH)

Entradas: $Q, \delta, \Sigma = \Sigma_{nc} \cup \Sigma_c, \Gamma, q_0, depth$

Saídas : $path, time$

```

1   $path[(q, a_0, 0)] \leftarrow \epsilon$ 
2   $time[(q, a_0, 0)] \leftarrow 0$ 
3
4
5   $F \leftarrow (q_0, a_0, 0)$ 
6
7  while  $F$  is not empty do
8     $(q, a, i) \leftarrow F$ 
9    if  $\exists \sigma \in (\Gamma(q) \cap \Sigma_{nc}) : f_T(path[(q, a, i)]\sigma) < \infty$  then
10      $t_{min} \leftarrow \min_{\sigma \in \Sigma_{nc}} f_T(path[(q, a, i)]\sigma)$ 
11     if  $\exists \sigma \in (\Gamma(q) \cap \Sigma_c) : f_T(path[(q, a, i)]\sigma) \leq t_{min}$  then
12        $E \leftarrow \Gamma(q) \cap \{\sigma_t : \sigma_t \in \Sigma_c \wedge$ 
13          $f_T(path[(q, a, i)]\sigma_t) \leq t_{min}\}$ 
14     else
15        $E \leftarrow \Gamma(q) \cap \{\sigma_t : \sigma_t \in \Sigma_{nc} \wedge$ 
16          $f_T(path[(q, a, i)]\sigma_t) = t_{min}\}$ 
17     else
18        $E \leftarrow \Gamma(q) \cap \Sigma_c$ 
19
20   foreach event  $\sigma$  in  $E$  do
21      $v \leftarrow \delta(q, \sigma)$ 
22      $t \leftarrow f_T(path[(q, a, i)]\sigma)$ 
23      $a_n \leftarrow update(a, \sigma)$ 
24     if  $t = \infty$  then continue
25
26   if  $F$  does not contain  $(v, a_n, i + 1)$  then
27      $F \leftarrow (v, a_n, i + 1)$ 
28
29    $t_{tot} \leftarrow time[(q, a, i)] + t$ 
30
31   if  $(\nexists time[(v, a_n, i + 1)])$  OR
32      $(t_{tot} < time[(v, a_n, i + 1)])$  then
33      $path[(v, a_n, i + 1)] \leftarrow path[(q, a, i)]\sigma$ 
34      $time[(v, a_n, i + 1)] \leftarrow t_{tot}$ 

```

O algoritmo segue uma lógica similar a um algoritmo de força bruta. Desta forma, um estado, alcançado por caminhos diferentes, com diferentes agendas de eventos, é mantido duplicado na fronteira de busca. Um caminho qualquer somente é descartado quando outro caminho alcança o mesmo estado, com a mesma agenda de eventos num tempo menor.

Para a execução do algoritmo são necessários: o conjunto de estados do sistema (Q), o conjunto de eventos do sistema ($\Sigma = \Sigma_{nc} \cup \Sigma_c$), a função de transição (δ), a função de eventos ativos (Γ), o estado inicial (q_0) e a profundidade de busca ($depth$). Como resultado, o algoritmo retorna as

estruturas $path$ e $time$ tal que $path[(q, a, i)]$ e $time[(q, a, i)]$ armazenam, respectivamente, a melhor sequência de eventos e o menor tempo encontrados partindo do estado inicial até o estado q , com agenda de eventos a na profundidade de busca i . A estrutura a é uma agenda de eventos, tal que $a[\sigma]$ em um estado (q, i) (onde i é a profundidade do estado), é equivalente à $f_T(path[(q, a, i)]\sigma)$. A cada execução de evento, a agenda a é atualizada utilizando a função $update$.

O algoritmo é inicializado com o caminho até o estado inicial como a sequência vazia ($path[(q_0, a_0, 0)] \leftarrow \epsilon$) e o tempo inicial como zero ($time[(q_0, a_0, 0)] \leftarrow 0$) nas linhas 2 e 3.

O estado inicial é inserido na fila F (linha 5) e, enquanto a fila possui elementos, o primeiro estado é retirado da fila (linha 8). A parte heurística do algoritmo está em considerar que eventos controláveis iniciam tarefas e que, no contexto de minimização do tempo, deve-se dar prioridade à sua execução em detrimento dos eventos não-controláveis. Dessa forma, o algoritmo preenche um conjunto com os eventos que serão avaliados (E). Caso existam eventos controláveis e não-controláveis temporalmente factíveis, E é formado apenas de eventos controláveis com tempo até sua execução igual inferior ao tempo até a execução do evento não-controlável mais próximo (linha 12). Caso existam apenas eventos não-controláveis, E consiste dos eventos não-controláveis com menor tempo até sua execução (linha 14). Caso contrário, E consiste apenas de eventos controláveis (linha 16).

Para cada transição por um evento no conjunto E , calcula-se o acréscimo de tempo gerado pela transição (linha 20), atualiza-se a agenda de eventos (linha 21) e verifica-se se a sequência resultante é temporalmente factível (linha 22). Caso o estado de destino ainda não tenha sido avaliado, ele é colocado na fila (linhas 26 até 28). O tempo total é avaliado como o tempo até o estado de origem, acrescido do intervalo de tempo da transição. Caso o estado de destino não esteja definido em **time**, ou gere uma diminuição no tempo total, essa transição é usada para atualizar o caminho e o tempo total (linha 27).

O algoritmo utiliza uma noção de estado estendido, na forma (q, a, i) , onde cada estado visitado pelo algoritmo corresponde a um estado do sistema, uma agenda de eventos e a profundidade do estado, dessa forma, um mesmo estado q , em uma mesma profundidade i é tratado como dois estados distintos se acompanhado de agendas distintas a_1 e a_2 .

Como o algoritmo evita a execução de eventos não-controláveis, uma restrição quanto à ocorrência de eventos não-controláveis se faz necessária para garantir que os eventos não-controláveis sejam executados em algum momento. Essa restrição garante que o algoritmo não comece a produzir mais produtos do que o especificado no tamanho do lote.

4. EXEMPLO DE APLICAÇÃO

Como exemplo de aplicação do algoritmo, será utilizado o Sistema Flexível de Manufatura (SFM) (de Queiroz et al., 2005), cujo diagrama está apresentado na Figura 1. O Sistema Flexível de Manufatura é uma planta composta por oito dispositivos, três esteiras (C_1 , C_2 e C_3), uma fresadora, um torno, um robô, uma máquina de pintura

Figura 1. Diagrama do Sistema Flexível de Manufatura

(MP) e uma máquina de montagem (MM). O SFM produz dois tipos de produto a partir de blocos e tarugos, sendo eles uma base com pino cônico, chamado Produto A, e uma base com um pino cilíndrico pintado, chamado Produto B.

Na Figura 2 estão representados os autômatos que modelam a dinâmica das máquinas que compõem o SFM.

Na Tabela 2 estão apresentados 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 relativa aos eventos controláveis 63 e 65 que 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

Figura 2. Plantas do Sistema Flexível de Manufatura

Tabela 1. Comparação dos resultados dos algoritmos para lotes de tamanho 1 e múltiplos de 5.

Lote	MPT		MTH		Estimado (exato p/ $n \leq 15$)	
	Makesp.	Tempo Exec.	Makesp.	Tempo Exec.	Makesp.	Tempo Exec.
1	238	0,023 seg	238	0,026 seg	238	0,6 min
5	878	0,057 seg	866	0,325 seg	866	10,2 min
10	1.663	0,224 seg	1.651	0,723 seg	1.651	30,2 min
15	2.448	0,509 seg	2.436	1,136 seg	2.436	61,7 min
20	3.233	0,944 seg	3.221	1,79 seg	3.221	-
25	4.018	1,016 seg	4.006	2,607 seg	4.006	-
30	4.803	1,165 seg	4.791	3,432 seg	4.791	-
35	5.588	1,37 seg	5.576	4,299 seg	5.576	-
40	6.373	1,534 seg	6.361	5,157 seg	6.361	-
45	7.158	1,748 seg	7.146	5,901 seg	7.146	-
50	7.943	1,917 seg	7.931	6,878 seg	7.931	-

Tabela 2. Intervalo de tempo entre pares de eventos

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

(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 de eventos executados são:

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

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

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

Considerando o supervisor monolítico para o SFM, obtido pela Teoria de Controle Supervisório convencional, composto por 45.504 estados e 200.124 transições, a produção de um produto A e um produto B é considerado um lote de tamanho 1, dessa forma, um lote de tamanho N consiste na produção de N produtos A e N produtos B, de forma que o número de eventos necessário para a execução de um lote qualquer é dado por $44 N$.

Foram realizados testes com dois algoritmos sendo eles: o algoritmo de máximo paralelismo com restrições temporais (MPT) e o algoritmo de minimização do tempo com heurística (MTH). Foram feitas execuções para lotes de tamanhos 1 até 50. Os testes foram executados em um computador com CPU Intel Xeon E5-2667 2,90 GHz e 64GB de RAM.

Malik and Pena (2018) resolveram esse mesmo problema, usando Verificação Formal, de maneira exata, para lotes de $N \leq 15$. Os autores também forneceram uma fórmula que permite estimar o menor *makespan* para um lote de tamanho qualquer.

$$T(N) = 157N + 81. \quad (1)$$

A Tabela 1 apresenta o *makespan* obtido pelos dois algoritmos e o valor estimado por (1).

Como pode ser observado, o algoritmo MTH obteve, em todos os casos, o menor *makespan*. Além disso, o MTH obteve, em todos os testes, resultados iguais aos valores apresentados em (Malik and Pena, 2018) (Exatos para $N \leq 15$ e estimados para $N > 15$). A Figura 3 mostra o tempo de execução dos algoritmos do Máximo Paralelismo com Restrições Temporais (MPT) e da Minimização de Tempo Heurístico (MTH). O MTH, apesar de ser mais lento que MPT, apresenta melhores resultados e, ainda assim, um tempo muito inferior àquele necessário para obter o menor *makespan* por verificação formal.

Figura 3. Comparação do tempo de execução dos algoritmos MTH e MPT (Alves et al., 2016a)

A Tabela 3 apresenta o paralelismo acumulado das sequências obtidas pelos algoritmos MPT e MTH. Como pode ser observado, apesar de o paralelismo ser uma métrica que apresenta correlação com o *makespan*, uma sequência com paralelismo mais elevado não necessariamente apresenta um menor tempo de produção.

Tabela 3. Paralelismo acumulado nas sequências obtidas pelos algoritmos MPT e MTH em lotes de tamanho múltiplos de 5

Lote	Paralelismo		Lote	Paralelismo	
	MPT	MTH		MPT	MTH
5	713	635	30	4.588	4.035
10	1.488	1.315	35	5.363	4.715
15	2.263	1.995	40	6.138	5.395
20	3.038	2.675	45	6.913	6.075
25	3.813	3.355	50	7.688	6.755

É importante notar que o algoritmo de Menor Tempo Heurístico (MTH) encontra sequências com o mínimo makespan (obtido por Verificação Formal) em todas as instâncias testadas, com tempo de execução muito inferior.

5. CONCLUSÕES

Neste artigo foi apresentada uma heurística de planejamento da produção em Sistemas a Eventos Discretos visando a minimização de *makespan* baseada na priorização da execução de eventos controláveis.

O algoritmo não é exato, mas em todos os testes apresentou resultados iguais ou superiores ao algoritmo de Máximo Paralelismo com Restrições Temporais (MPT) (Alves et al., 2016a), obtendo *makespans* iguais aos resultados exatos obtidos por Verificação Formal (Malik and Pena, 2018). Apesar de ser mais lento que o MPT, o aumento de produtividade ocasionado por um plano de produção melhor pode, em muitos casos, compensar um maior tempo de execução do algoritmo.

O mesmo algoritmo apresentado nesse artigo pode ser facilmente adaptado para otimizar sistemas distribuídos obtidos aplicando Controle Modular Local de SEDs tratando como estado a tupla formada pelos estados dos diferentes componentes do sistema.

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. In *XII Simpósio Brasileiro de Automação Inteligente (SBAI)*, 594–599.
- Alves, L., Pena, P., and Takahashi, R. (2016a). Planejamento da produção baseado no critério do máximo paralelismo com restrições temporais. In *XXI Congresso Brasileiro de Automação (CBA)*.
- Alves, L., Pena, P., and Takahashi, R. (2016b). Planning on discrete event systems: A logical approach. In *12th Conference on Automation Science and Engineering (CASE)*.
- Cassandras, C. and Lafortune, S. (2008). *Introduction to Discrete Event Systems*. Springer, 2 edition.
- Costa, T.A., Pena, P.N., and Takahashi, R.H.C. (2018). Sco-concat: a solution to a planning problem in flexible manufacturing systems using supervisory control theory and optimization techniques. *Journal of Control, Automation and Electrical Systems*, 29(4), 500–511. doi: 10.1007/s40313-018-0386-7. URL <https://doi.org/10.1007/s40313-018-0386-7>.
- de Queiroz, M.H., Cury, J.E.R., and Wonham, W.M. (2005). Multitasking supervisory control of discrete-event systems. *Discrete Event Dynamic Systems*, 15(4), 375–395.
- Garey, M.R. and Johnson, D.S. (1979). *Computers and intractability*. W.H. Freeman.
- Ghllab, M., Nau, D., and Traverso, P. (2005). *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. In *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, 435–440.
- Kobetski, A. and Fabian, M. (2006). Scheduling of discrete event systems using mixed integer linear programming. In *Discrete Event Systems, 2006 8th International Workshop on*, 76–81.
- López-Mellado, E., Villanueva-Paredes, N., and Almeyda-Caneпа, H. (2005). Modelling of batch production systems using petri nets with dynamic tokens. *Mathematics and Computers in Simulation*, 67(6), 541 – 558.
- Malik, R. and Pena, P.N. (2018). Optimal task scheduling in a flexible manufacturing system using model checking. In *2018 14th International Workshop on Discrete Event Systems (WODES)*, 241–246.
- Pena, P.N., Costa, T.A., and Takahashi, R.H.C. (2016). Control of flexible manufacturing systems under model uncertainty using supervisory control theory and evolutionary computation schedule synthesis. *Information Sciences*, 329, 491 – 502. doi:<https://doi.org/10.1016/j.ins.2015.08.056>. Special issue on Discovery Science.
- Pinedo, M.L. (2012). *Scheduling: Theory, Algorithms, and Systems*. Springer Publishing Company, Incorporated, 3rd edition.
- Pinha, D., de Queiroz, M., and Cury, J. (2011). Optimal scheduling of a repair shipyard based on supervisory control theory. In *Automation Science and Engineering (CASE), 2011 IEEE Conference on*, 39–44.
- Rafael, G.C., Alves, L., Batista, L., and Pena, P. (2018). Minimização de makespan e consumo energético em sistemas a eventos discretos. In *XXII Congresso Brasileiro de Automação (CBA)*.
- Rafael, G. and Pena, P. (2018). Using an abstraction of the supervisor to solve a planning problem in manufacturing systems. In *XXII Congresso Brasileiro de Automação (CBA)*.
- 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.
- Su, R., van Schuppen, J., and Rooda, J. (2012). The synthesis of time optimal supervisors by using heaps-of-pieces. *Automatic Control, IEEE Transactions on*, 57(1), 105–118.
- Ware, S. and Su, R. (2016). Incremental scheduling of discrete event systems. In *2016 13th International Workshop on Discrete Event Systems (WODES)*, 147–152.
- Ware, S. and Su, R. (2017a). Time optimal synthesis based upon sequential abstraction and its application to cluster tools. *IEEE Transactions on Automation Science and Engineering*, 14(2), 772–784.
- Ware, S. and Su, R. (2017b). Time optimal synthesis based upon sequential abstraction and maximizing parallelism. In *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*, 926–931.