# Planejamento da Produção em um Sistema Flexível de Manufatura com Demandas Estocásticas

**Daniel Sarsur C.** [*] **Patrícia N. Pena** [**]
**Ricardo H. C. Takahashi** [***]

[*] *Programa de Pós-Graduação em Engenharia Elétrica - Universidade Federal de Minas Gerais, Belo Horizonte, MG, Brasil (e-mail: danielsc@ufmg.br).*
[**] *Departamento de Engenharia Eletrônica - Universidade Federal de Minas Gerais, Belo Horizonte, MG, Brasil (e-mail: ppena@ufmg.br).*
[***] *Departamento de Matemática - Universidade Federal de Minas Gerais, Belo Horizonte, MG, Brasil (e-mail: taka@mat.ufmg.br).*

**Abstract:** This paper approaches a scheduling problem where a flexible manufacturing system in continuous production receives a new production demand. An algorithm is proposed to find chains of events that minimize the total makespan and the new demand makespan. The solution search universe is the closed loop behavior of the system, obtained through the Supervisory Control Theory and, therefore, ensures that the sequences are logically feasible and safe. In addition, temporal information is added, also ensuring the temporal feasibility. The results show that minimizing the original and the new demand makespan are conflicting and that the algorithm comes up with diversified and fast solutions.

**Resumo**: Este trabalho trata do problema de escalonamento de tarefas de um sistema flexível de manufatura, em um regime de produção contínua, que recebe uma nova demanda de produção. Um algoritmo para encontrar cadeias de eventos que minimizem o makespan total e o makespan da nova demanda é proposto. O universo de busca das soluções é o comportamento em malha fechada do sistema, obtido por meio da Teoria de Controle Supervisório e, por isso, garante-se que as sequências são logicamente factíveis e respeitam as especificações. Além disso, informações temporais são adicionadas, assegurando, também, a factibilidade em relação a esse critério. Os resultados mostram que a minimização do makespan da produção da nova demanda e da produção total são conflitantes e que o algoritmo apresenta soluções diversificadas e rápidas.

*Keywords:* Discrete Event Systems; Supervisory Control Theory; Scheduling; Optimization; Application.

*Palavras-chaves:* Sistemas a Eventos Discretos; Teoria de Controle Supervisório; Escalonamento de tarefas; Otimização; Aplicação.

## 1. INTRODUÇÃO

As atividades de produção remontam ao início da civilização, ainda que executados de forma simples e sem uma organização sistemática. Desde a Revolução Industrial, no final do século XVIII, os sistemas de planejamento e controle de produção vêm evoluindo, se estruturando e se organizando (Lustosa et al., 2008). Pelo caráter contínuo dessa evolução, é necessário que diversas áreas ainda se ocupem de pesquisar soluções que possam melhorar os processos produtivos. Entre os possíveis objetivos dessas melhorias, podemos citar a redução de insumos, o aumento de produtividade, a redução de makespan (tempo para produção), o aumento de throughput (taxa de produção), a redução de perdas ao longo do processo, entre outros resultados (Petrônio and Laugeni, 2005).

Uma das formas de melhorar esses fatores é por meio do planejamento e controle da produção e do escalonamento de tarefas, ou seja, encontrando cadeias de alocação dos recursos disponíveis para a produção de um bem, de forma a otimizar um ou mais objetivos. *Scheduling* é o termo comumente encontrado na literatura para designar o processo de tomada de decisão baseado nos critérios que levem à melhoria dos objetivos que se desejam maximizar ou minimizar (Pinedo, 2008).

As propriedades intrínsecas dos problemas de otimização do escalonamento de tarefas fazem com que os algoritmos desenvolvidos para os resolver se enquadrem na classe de problemas denominada NP-Difícil (Garey and Johnson, 1979).

No cotidiano e, principalmente, em sistemas de manufatura, é comum nos deparamos com processos que podem ser modelados como Sistemas a Eventos Discretos (SEDs). Como o nome sugere, nesse tipo de modelagem, os estados dos sistemas é um conjunto discreto e se modificam em decorrência de eventos cuja ocorrência é instantânea. Esse


DOI: 10.17648/sbai-2019-111176

tipo de sistema se diferencia daqueles dirigidos pelo tempo, por exemplo, em que o estado varia continuamente, e suas soluções utilizam recursos da Teoria de Controle Clássica.

O tipo de abstração a que leva os SEDs é de grande utilidade por oferecer diversas ferramentas capazes de modelar, simular e analisar sistemas dessa natureza, com o fim de solucionar problemas relacionados a requisitos de segurança e não-bloqueio. Podemos citar como exemplos de abstrações de SEDs as Redes de Petri (López-Mellado et al., 2005; de Pádua et al., 2004); autômatos temporizados (Abdeddaïm et al., 2006); linguagens e autômatos (Hopcroft et al., 2001; Cassandras and Lafortune, 2009).

A representação por linguagens e autômatos tem uma grande aliada: a Teoria de Controle Supervisório (TCS), proposta por Ramadge and Wonham (1989). Juntas, além de modelar o comportamento das plantas a serem estudadas, elas permitem a modelagem das restrições às quais as plantas estão submetidas. Isso assegura um comportamento minimamente restritivo e que garante a vivacidade e segurança do sistema. Sem o conjunto de cadeias infactíveis, o espaço de estados é significativamente reduzido, o que é interessante no contexto do problema da explosão do espaço de estados, que se caracteriza pelo rápido crescimento do espaço de estados do problema em função do crescimento do tamanho da entrada.

Como já foi apresentado, o problema de otimização do escalonamento de tarefas é NP-Difícil e, portanto, a obtenção da solução ótima para grandes instâncias torna-se impossível. Uma alternativa comumente utilizada para contornar essa situação é o uso de heurísticas (Costa et al., 2018; Pena et al., 2016; Santos et al., 2019; Liao et al., 2015). Heurísticas são algoritmos que analisam apenas algumas das possíveis soluções, definidas com base em um critério que as aproxime da solução ótima, descartando as outras que aparentam levar ao sentido contrário (Pearl, 1984). Dessa forma, não há como garantir a otimalidade da solução encontrada, mas o tempo de execução do algoritmo torna-se viável.

Este trabalho utiliza o conjunto de ferramentas apresentado anteriormente como meio de solucionar o problema da otimização do escalonamento de tarefas. Em particular, a solução proposta visa a minimização do makespan de uma sequência de produção contínua e das demandas estocásticas que podem aparecer. Esse problema ilustra uma situação do contexto industrial em que um sistema, que compartilha recursos, está programado para produzir um lote de um tipo de produto e, em um instante de tempo qualquer, recebe uma nova demanda de outro tipo de produto. Deseja-se encontrar a cadeia de eventos que minimize os tempos de produção total e do novo lote recebido.

Este artigo apresenta os conceitos preliminares na Seção 2. A Seção 3 apresenta a motivação para a realização deste trabalho. Resultados anteriores são apresentados na Seção 4 e os resultados principais na Seção 5. A Seção 6 apresenta a aplicação dos resultados obtidos em um problema prático e a Seção 7 apresenta as conclusões.

## 2. PRELIMINARES

Define-se $\Sigma$ como o conjunto finito e não vazio de eventos. Uma sequência finita de eventos é chamada de cadeia. O conjunto formado por todas as cadeias possíveis, utilizando esse alfabeto $\Sigma$, é chamado de Fechamento Kleene $\Sigma^*$, incluindo a cadeia vazia $\epsilon$. Além disso, define-se que uma linguagem é um subconjunto $L \subseteq \Sigma^*$ e que a concatenação de duas cadeias é representada por $su$, onde $s, u \in \Sigma^*$. O comprimento de uma cadeia é definido por $|\sigma_1\sigma_2...\sigma_i| = i$, onde $i > 0$. O comprimento de uma cadeia vazia, representada pelo símbolo $\epsilon$, é $|\epsilon| = 0$.

Uma cadeia $s$ é chamada prefixo de $t \in \Sigma^*$, $s \leq t$, se $\exists u \in \Sigma^*$ tal que $su = t$. Com isso, é possível definir prefixo fechamento $\bar{L}$ de uma linguagem $L \in \Sigma^*$, que é o conjunto de todos os prefixos de cadeias em $L$, que pode ser expresso por $\bar{L} = \{s \in \Sigma^* | s \leq t \text{ para algum } t \in L\}$.

Outra definição importante é a de um autômato finito determinístico que consiste de uma 5-tupla $G = (Q, \Sigma, \delta, q_0, Q_m)$, sendo Q o conjunto finito de estados, $\Sigma$ o alfabeto, $\delta : Q \times \Sigma \to Q$ a função de transição, $q_0 \in Q$ o estado inicial e $Q_m \subseteq Q$ o conjunto de estados marcados. A função de transição é estendida para uma cadeia na forma $\delta(q, \sigma s) = q'$ se $\delta(q, \sigma) = x$ e $\delta(x, s) = q'$, em que, $s \in \Sigma^*$ e $\sigma \in \Sigma$. A função de eventos ativos $\Gamma : Q \to 2^\Sigma$ é dado pelo conjunto de eventos $\sigma \in \Sigma$, em um dado estado $q$, onde $\delta(q, \sigma)$ é definido, ou seja, $\delta(q, \sigma)!$

Com isso é possível definir a operação de composição síncrona entre dois autômatos. Seja $G_1 = (Q_1, \Sigma_1, \delta_1, q_{01}, Q_{m1})$ e $G_2 = (Q_2, \Sigma_2, \delta_2, q_{02}, Q_{m2})$. $G_{12} = G_1||G_2 = (Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, \delta_{12}, (q_{01}, q_{02}), Q_{m1} \times Q_{m2})$ onde

$$\delta_{12} = \delta((q_1, q_2), e) = \begin{cases} (\delta_1(q_1, e), \delta_2(q_2, e)), & \text{se } e \in \Gamma_1(q_1) \cap \Gamma_2(q_2) \\ (\delta_1(q_1, e), q_2), & \text{se } e \in \Gamma_1(q_1) \backslash \Sigma_2 \\ (q_1, \delta_2(q_2, e)), & \text{se } e \in \Gamma_2(q_2) \backslash \Sigma_1 \\ \text{indefinido}, & \text{caso contrário.} \end{cases}$$

e $\Gamma_{1||2}(q_1, q_2) = [\Gamma_1(q_1) \cap \Gamma_2(q_2)] \cup [\Gamma_1(q_1) \backslash \Sigma_2] \cup [\Gamma_2(q_2) \backslash \Sigma_1]$, em que a notação $\Gamma(q) \backslash \Sigma$ indica a subtração dos conjuntos.

A linguagem gerada por um autômato é dada por $\mathcal{L}(G) = \{s \in \Sigma^* | \delta(q_0, s)!\}$. A linguagem reconhecida por um autômato é dada por $\mathcal{L}_m(G) = \{s \in \Sigma^* | s \in \mathcal{L}(G) \text{ e } \delta(q_0, s) \in Q_m\}$.

Para mais detalhes sobre as informações apresentadas nessa seção, sugere-se consultar Cassandras and Lafortune (2009).

### *2.1 Teoria de Controle Supervisório*

A modelagem de SEDs utilizando autômatos possibilita a utilização da Teoria de Controle Supervisório - TCS (Ramadge and Wonham, 1989). Essa teoria permite, partindo-se das plantas de um sistema e suas especificações, encontrar um supervisor que seja não-bloqueante e minimamente restritivo. Isso significa que o comportamento em malha fechada desse sistema permite que as plantas executem todas aquelas tarefas possíveis de serem feitas e que não comprometam a vivacidade e segurança do sistema.


DOI: 10.17648/sbai-2019-111176

Nessa abordagem, o alfabeto é particionado em dois subconjuntos disjuntos $\Sigma = \Sigma_c \cup \Sigma_{uc}$. O primeiro, $\Sigma_c$, dos eventos controláveis e o segundo, $\Sigma_{uc}$, dos eventos não controláveis. Esse último, como o nome sugere, é composto dos eventos que não podem ser impedidos de ocorrer.

Para que o comportamento em malha fechada $\mathcal{L}_m(S/G)$ atenda ao critério da controlabilidade em relação a uma planta $G$ e suas especificações $E$, tal que $\mathcal{L}_m(S/G) = \mathcal{L}(G)||E = K$, é necessário que $K$ seja controlável em relação a $\mathcal{L}(G)$ e $\Sigma_{uc}$, ou seja, $\bar{K}\Sigma_{uc} \cap \mathcal{L}(G) \subseteq \bar{K}$. No caso de $S$ não atender a esse critério, deve-se calcular a máxima sublinguagem controlável, expressa por $Sup\mathcal{C}(K, G)$.

## 3. MOTIVAÇÃO

A motivação desse trabalho surge a partir de reflexões sobre problemas práticos a serem solucionados dentro do contexto industrial. Em particular, um dos cenários se mostrou relevante de ser abordado: o de um sistema que tem um fluxo contínuo de produção de um determinado tipo de produto e, em um instante qualquer, recebe uma nova demanda por um outro tipo de produto.

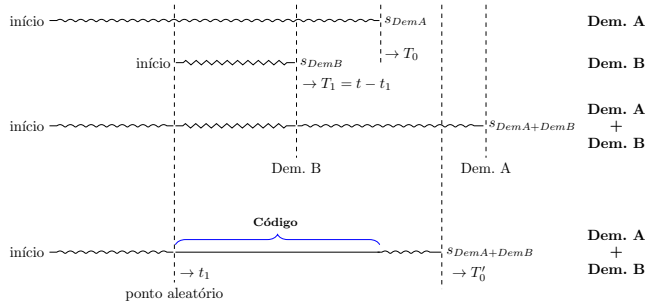


Figura 1. Diagrama ilustrativo do cenário abordado

A Figura 1 ilustra esse tipo de problema. O desenho da primeira linha, ondulada, representa a cadeia de eventos ótima $s_{DemA}$ que minimiza o makespan da demanda A, referente à produção contínua de um tipo de produto. A segunda linha, em *zigzag*, representa a cadeia de eventos ótima $s_{DemB}$ para a produção da demanda B, que chega em um instante de tempo aleatório, referente à produção de outro tipo de produto.

A terceira linha representa uma das possíveis cadeias de eventos $s_{DemA+DemB}$ para a produção dos dois lotes em questão. Para obtê-la, simplesmente suspende-se a produção da primeira demanda até que a segunda seja finalizada e, então, retoma-se a produção do ponto em que parou anteriormente. Embora essa seja a forma mais simples para encontrar a nova cadeia, não se aproveitam as possibilidades de execução de tarefas em paralelo, o que reduziria o makespan dessa cadeia.

Levanta-se o questionamento de qual seria, então, a forma mais rápida de produzir a nova demanda causando o menor atraso para a produção inicial. A quarta linha da Figura 1 representa esta situação, em que o algoritmo proposto realiza o cálculo da nova sequência, a partir da entrada da demanda B, até sua finalização.

### *3.1 Definição do problema*

Seja um sistema de manufatura capaz de produzir dois ou mais tipos de produtos com o compartilhamento de recursos. Sejam os tempos de operação das máquinas conhecidos e fixos. Seja $s^*_{DemA}$ uma cadeia otimizada(mínimo makespan) de eventos para a produção de um lote de uma quantidade arbitrária de uma demanda A, definida como $DemA$, cujo makespan($C_{max}$) é dado por $C_{max}(s^*_{DemA}) = T_0$. Dada a chegada da demanda de produção de um novo lote $DemB$ no instante de tempo $t_1$, sendo $0 < t_1 < T_0$, deseja-se encontrar uma nova cadeia de eventos $s_{DemA+DemB}$ de forma que o tempo $T_1$ $(t - t_1)$ de produção do novo lote e o tempo $T_0'$ de produção total (lote original + novo lote) sejam os menores possíveis.

## 4. RESULTADOS ANTERIORES

Alves (2016) propõem uma heurística para resolver o problema de minimização do makespan que prioriza os eventos controláveis em detrimento dos não controláveis. Essa lógica, segundo o autor, parte do princípio de que os eventos controláveis estão associados ao início de uma tarefa e, em geral, retardá-los para esperar que uma outra tarefa seja finalizada não traz melhorias para a solução do problema de minimização do makespan.

Os resultados encontrados pelo algoritmo Menor Tempo Heurístico (MTH) foram considerados satisfatórios pelo autor, uma vez que, mesmo sem a garantia da otimalidade das soluções, este foi um dos algoritmos que encontrou os menores valores de makespan para os problemas avaliados. Uma versão mais atualizada desse algoritmo consegue encontrar o valor ótimo de makespan para todas as instâncias em que ele é conhecido, para o problema que o algoritmo foi testado (Malik and Pena, 2018).

## 5. RESULTADOS PRINCIPAIS

Para solucionar o problema apresentado foi proposto o Algoritmo 1. O algoritmo inicia executando o Algoritmo MTH a fim de obter uma cadeia $s$ de produção para o lote inicial (linha 1). São passados como parâmetros o estado inicial $q_0$, o tempo inicial $t_0 = 0$, o agendador de eventos inicial (adaptado de Cassandras and Lafortune (2009)), a estrutura de restrição inicial $r_0$ e a demanda de produtos inicial $(A_0, B_0)$. A estrutura de restrição está relacionada com a quantidade de vezes que cada evento pode acontecer, implicitamente relacionado com a profundidade da busca a ser feita.

O programa segue executando a cadeia de eventos de $s$ e armazenando-os em $v$ (linha 4) enquanto uma nova demanda não é recebida. A cada evento as estruturas de estado atual $(q)$, tempo total $(t)$, agendador $(a)$ e de restrição $(r)$ são atualizadas (linhas 5-8).

Ao se observar a chegada de uma nova demanda, a estrutura de restrição $r_1$ recebe um novo valor em função do tamanho do novo lote (linha 11), podendo ser mesclado com parte da demanda do lote original. O Algoritmo MTH é, então, novamente executado partindo do ponto onde o sistema havia parado, retornando uma cadeia de eventos $x$, o novo estado atual e o tempo, além das estruturas do agendador e das restrições (linha 12).


DOI: 10.17648/sbai-2019-111176

Após isso, uma nova restrição $r_r$ é definida considerando a quantidade de eventos necessários para a produção do restante dos produtos do lote original que ainda não foram produzidos até então (linha 14). O Algoritmo MTH é executado pela terceira vez, porém, considerando a restrição $r_r$ e um lote do tamanho dos produtos que ainda não haviam sido produzidos. Uma cadeia de eventos $y$ e o tempo total de produção são retornados(linha 15).

O algoritmo finaliza concatenando as cadeias obtidas anteriormente resultando na cadeia $u$ (linha 17), que é a cadeia de eventos que minimiza o tempo de produção total e o tempo de produção do novo lote.

**Algoritmo 1** Calcula o caminho que minimiza o makespan total de uma produção e o makespan de uma nova demanda

1: $s \leftarrow MTH(q_0, t_0, a_0, r_0, (A_0, B_0))$
2:
3: **enquanto** $\neg$ *nova demanda* **faça**
4: $v \leftarrow \sigma \in s$
5: $q \leftarrow \delta(q, \sigma)$
6: $t \leftarrow a[\sigma]$
7: $a \leftarrow update(a, \sigma)$
8: **se** $\sigma \in \Sigma_c$ **então** $r[\sigma] = r[\sigma] - 1$
9: **fim**
10:
11: $r_1 \leftarrow nova\_restrição(A_1, B_1)$
12: $(x, q, t, a, r) \leftarrow MTH(q, t, a, r_1, (A_1, B_1))$
13:
14: $r_r \leftarrow nova\_restrição(A_r, B_r)$
15: $(y, t) \leftarrow MTH(q, t, a, r_r, (A_r, B_r))$
16:
17: $u \leftarrow vxy$

## 6. ESTUDO DE CASO

Para testar o algoritmo proposto foi escolhido o Sistema Flexível de Manufatura – SFM (Queiroz et al., 2004). Esse sistema é capaz de produzir dois tipos diferentes de produtos, sendo eles, o Produto A, bloco com pino cônico sem pintura, e o Produto B, bloco com pino cilíndrico pintado. O sistema é composto por três esteiras $C_1$, $C_2$ e $C_3$, um robô, uma fresa, um torno, uma máquina de pintura, uma máquina de montagem e oito buffers unitários, Figura 2.

A modelagem de cada planta está apresentada na Figura 3, enquanto a modelagem dos buffers está na Figura 4. Os eventos controláveis, pertencentes a $\Sigma_c$, são representados pelos números ímpares e estão relacionados com a inicialização de uma tarefa, enquanto os eventos não controláveis, pertencentes a $\Sigma_{uc}$, são representados pelos números pares e estão relacionados com a finalização de uma tarefa dos equipamentos da Figura 2.

A lógica de controle é adicionada por meio das especificações dos buffers para permitir o maior grau de liberdade para a produção das peças sem, contudo, permitir que ocorram *overflow* e *underflow*. Com isso, por meio da TCS é possível encontrar o supervisor monolítico para esse sistema que possui 45.504 estados e 200.124 transições.

Para realizar as avaliações temporais das simulações são utilizados os intervalos de tempo entre a ocorrência de um

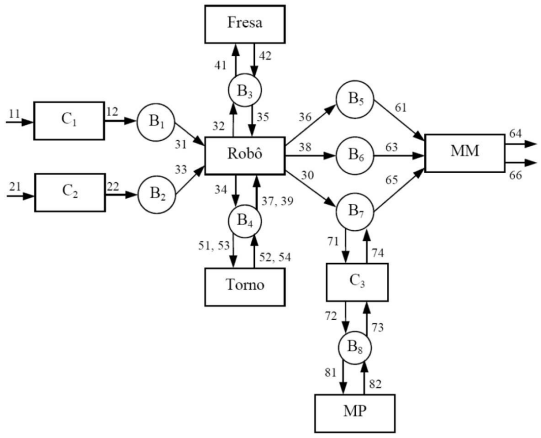


Figura 2. Sistema Flexível de Manufatura.

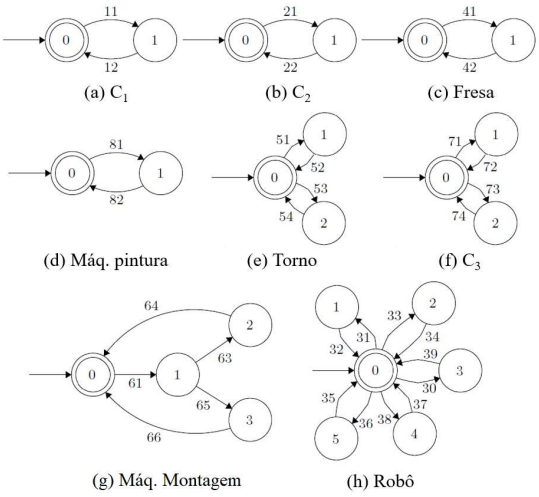


Figura 3. Modelagem das máquinas do Sistema Flexível de Manufatura.

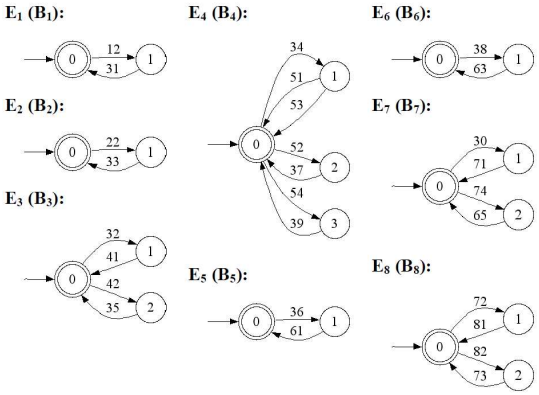


Figura 4. Modelagem dos buffers do Sistema Flexível de Manufatura.

evento controlável e seu respectivo evento não controlável, conforme apresentados na Tabela 1, de Alves et al. (2016).

A produção é um conjunto de três subprodutos, cada um deles conforme a sequência definida abaixo. Os eventos podem ser intercalados para gerar uma sequência que tenha menor makespan, desde que obedeçam a ordem de cada subconjunto. O Produto A é a junção da *base* com


DOI: 10.17648/sbai-2019-111176

Tabela 1. Intervalo de tempo entre os pares de eventos.

| Eventos controláveis | Eventos não controláveis | Intervalo de tempo [s] |
|---|---|---|
| 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 |

o $pino_A$ (cônico sem pintura) e o Produto B a junção da *base* com o $pino_B$ (cilíndrico pintado).

- $base = (11, 12), (31, 32), (41, 42), (35, 36), (61)$
- $pino_A = (21, 22), (33, 34), (51, 52), (37, 38), (63, 64)$
- $pino_B = (21, 22), (33, 34), (53, 54), (39, 30), (71, 72), (81, 82), (73, 74), (65, 66)$

### *6.1 Simulação*

Para a realização deste trabalho utilizou-se um computador Dell com o sistema operacional Windows 10 de 64 bits, memória RAM de $8,00GB$ e processador Intel Core i5-4210U / 1,70 GHz. Os algoritmos foram implementados no ambiente de desenvolvimento integrado Microsoft Visual Studio 2017, utilizando a linguagem de programação C#. Foi utilizada, também, a biblioteca para sistemas a eventos discretos UltraDES (Alves et al., 2017).

Para a simulação do Algoritmo 1 proposto foi definida uma demanda inicial de produção de 100 Produtos A e 0 Produtos B. No instante de tempo 3.000 segundos acrescenta-se uma nova demanda de 0 Produtos A e 40 Produtos B. Uma vez que a restrição $r_1$ é definida em função do tamanho do lote da nova demanda, podendo ser mesclada com os produtos ainda não finalizados da demanda inicial, há, portanto, mais de uma solução possível. No caso, verificou-se que, após 3.000 segundos, havia 36 Produtos A prontos. Dessa forma, é possível simular diversas mesclagens, que vão desde não mesclar nenhum produto $[(0, 40)]$ até mesclar todos os produtos restantes $[(64, 40)]$.

Para cada uma das simulações foi obtido o tempo entre o instante 3.000 segundos e o instante em que o último produto da nova demanda é finalizado e o tempo entre o início e o fim da produção da demanda inicial. Os resultados encontrados estão apresentados na Figura 5. O círculo indicado com o número 1 representa a solução onde não há mescla de nenhum produto $[(0, 40)]$ e o círculo 2 a solução com maior mescla possível $[(64, 40)]$.

É possível observar que a mescla de uma quantidade pequena de Produtos A com a nova demanda de Produtos B permite a finalização mais rápida do novo lote, mas provoca o aumento do makespan total, uma vez que prioriza mais os Produtos B em detrimento dos Produtos A. À medida em que há uma maior mesclagem dos produtos, o tempo de produção da nova demanda aumenta, pois a planta compartilha recursos, mas o makespan total é reduzido. A partir do momento em que a quantidade de produtos mesclados é aproximadamente igual $[(40, 40)]$, observa-se que não há variação no makespan total, embora o makespan da nova demanda cresça.

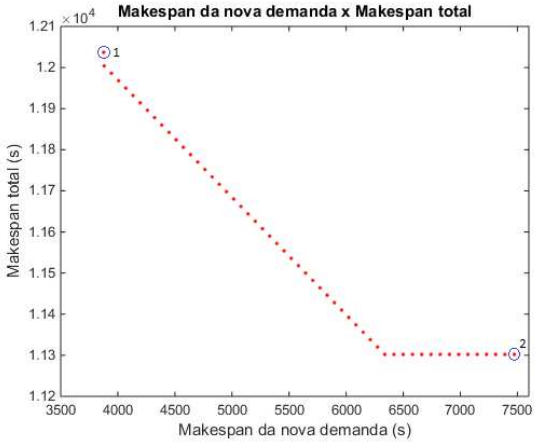


Figura 5. Valores de makespan das soluções encontradas.

A grande vantagem desse cenário está relacionada com as diferentes possibilidades de soluções encontradas. Fica clara, então, a necessidade de se fazer um *trade off* entre as duas medidas, com base nos critérios que cada instância desse problema pode apresentar, como *deadlines*, penalidades por atraso e outros.

Também foi registrado o tempo de execução do algoritmo para cada uma das soluções. Estes valores estão apresentados na Figura 6. As soluções com maior e menor mescla foram igualmente marcadas no gráfico.

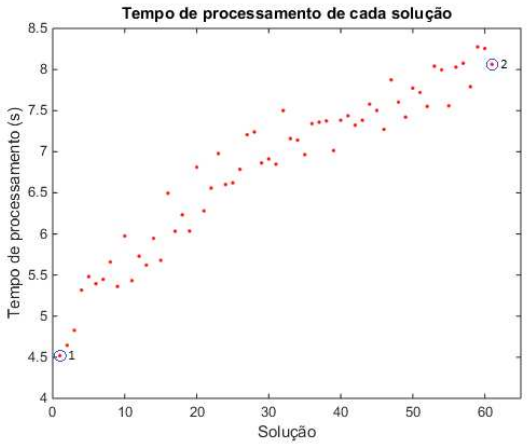


Figura 6. Tempo de execução do algoritmo para cada solução encontrada.

A partir do gráfico é possível observar que o tempo de execução do algoritmo cresce, aproximadamente, de forma linear em função do tamanho da quantidade de produtos mesclados. Uma explicação para isso pode estar relacionada com o *branch factor* que cresce à medida que


DOI: 10.17648/sbai-2019-111176

a quantidade de produtos também cresce. O tempo total para encontrar todas as soluções foi de seis minutos e cinquenta e cinco segundos.

## 7. CONCLUSÃO

Neste trabalho foi desenvolvido um algoritmo para a obtenção de cadeias de eventos que minimizam os tempos de produção em um sistema flexível de manufatura. Este trabalho estende resultados anteriores em que se encontram soluções para um único lote inicial, aumentando sua aplicabilidade.

A simulação de um cenário ilustrativo do problema enunciado mostra que a minimização do makespan total e minimização do makespan de uma nova demanda concomitante são objetivos conflitantes. Esse cenário possibilita que seja agregado ao problema proposto todo o ferramental utilizado na solução de problemas multiobjetivos, como os métodos de auxílio à tomada de decisão. O algoritmo proposto foi capaz de encontrar soluções bem diversificadas no espaço de objetivos.

O tempo de execução do algoritmo pode ser considerado baixo em relação ao tempo total de produção da nova demanda ($\approx 0,13\%$ no caso simulado), o que torna essa solução viável e indicada para problemas dessa natureza.

Os resultados encontrados e o tipo de aplicação em que esse algoritmo pode ser utilizado levantam novas questões que poderão ser respondidas em trabalhos futuros, como formas mais eficientes de mesclar as demandas ou maneiras de encontrar soluções em pontos mais estratégicos para que não seja necessário computar todas as soluções.

## REFERÊNCIAS

Abdeddaïm, Y., Asarin, E., Maler, O., et al. (2006). Scheduling with timed automata. *Theoretical Computer Science*, 354(2), 272–300.

Alves, L.V.R. (2016). Planejamento da produção em sistemas a eventos discretos-análise lógica e temporal. *Dissertaçao de mestrado, UFMG*.

Alves, L.V., Martins, L.R., and Pena, P.N. (2017). Ultrades-a library for modeling, analysis and control of discrete event systems. *IFAC-PapersOnLine*, 50(1), 5831–5836.

Alves, L.V., Pena, P.N., Takahashi, R.H., and de Instrumentaçao, C.T.S. (2016). Planejamento da producao baseado no criterio do maximo paralelismo com restricoes temporais. In *Anais do Congresso Brasileiro de Automática*.

Cassandras, C.G. and Lafortune, S. (2009). *Introduction to discrete event systems*. Editora Springer, 2ª edition.

Costa, T.A., Pena, P.N., and Takahashi, R.H. (2018). Scoconcat: a solution to a planning problem in flexible manufacturing systems using supervisory control theory and optimization techniques. *Journal of Control, Automation and Electrical Systems*, 1–12.

de Pádua, S.I.D., da Silva, A.R.Y., Porto, A.J.V., and Inamasu, R.Y. (2004). O potencial das redes de petri em modelagem e análise de processos de negócio. *Gestão & Produção*, 11(1), 109–119.

Garey, M.R. and Johnson, D.S. (1979). *Computers and intractability: A Guide to the Theory of NP-Completness*, volume 29. W.H. Freeman New York.

Hopcroft, J.E., Motwani, R., and Ullman, J.D. (2001). Introduction to automata theory, languages, and computation. *Acm Sigact News*, 32(1), 60–65.

Liao, C.J., Lee, C.H., and Lee, H.C. (2015). An efficient heuristic for a two-stage assembly scheduling problem with batch setup times to minimize makespan. *Computers & Industrial Engineering*, 88, 317–325.

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.

Lustosa, L.J., de Mesquita, M.A., and Oliveira, R.J. (2008). *Planejamento e controle da produção*. Coleção Campus-ABEPRO. Editora Elsevier Brasil, 4ª edition.

Malik, R. and Pena, P.N. (2018). Optimal task scheduling in a flexible manufacturing system using model checking. *IFAC-PapersOnLine*, 51(7), 230–235.

Pearl, J. (1984). *Heuristics: intelligent search strategies for computer problem solving*. Artificial Intelligence. Addison-Wesley, 1ª edition.

Pena, P.N., Costa, T.A., Silva, R.S., and Takahashi, R.H. (2016). Control of flexible manufacturing systems under model uncertainty using supervisory control theory and evolutionary computation schedule synthesis. *Information Sciences*, 329, 491–502.

Petrônio, G.M. and Laugeni, P. (2005). *Administração da produção*. Editora Saraiva, 2ª edition.

Pinedo, M.L. (2008). *Scheduling: theory, algorithms, and systems*. Springer, New York, New York, 3ª edition.

Queiroz, M.H.d. et al. (2004). Controle supervisório modular e multitarefa de sistemas compostos.

Ramadge, P.J. and Wonham, W.M. (1989). The control of discrete event systems. *Proceedings of IEEE*, 77, 81–98.

Santos, H.G., Toffolo, T.A., Silva, C.L., and Vanden Berghe, G. (2019). Analysis of stochastic local search methods for the unrelated parallel machine scheduling problem. *International Transactions in Operational Research*, 26(2), 707–724.


**DOI:** 10.17648/sbai-2019-111176