ALGORITMO GENÉTICO PARA ESCALONAMENTO DE TAREFAS EM SISTEMAS DE MANUFATURA COM CONTROLE SUPERVISÓRIO E AUTÔMATOS COM PARÂMETROS

MAILLA C. SPRICIGO, ROMEU REGINATTO, SANDRO BATTISTELLA

Programa de Pós-Graduação em Engenharia Elétrica e Computação, Centro de Engenharias e Ciências Exatas, Universidade Estadual do Oeste do Paraná
Avenida Tancredo Neves 6731, Foz do Iguaçu, PR, Brasil
E-mails: maillaspricigo@gmail.com, romeu.cece@gmail.com, sandro.battistella@unioeste.br

Abstract— The job-shop scheduling aims to find optimal sequences of events that increases productivity indexes in manufacturing systems. This paper employs genetic algorithms such metaheuristics to obtain optimal scheduling solutions, combined with the supervisory control theory (SCT) in the automatically generation of sequences of operations involved in the production of parts in a didactic manufacturing system. The SCT allows deriving control structures (supervisors) that formally guarantee the plant operation correctness and safeness. In turn, the genetic algorithm searches for the best possible sequence of events, among all enabled by the supervisors, exploring the plant parallel-ism, allowing to meet production goals while complying with safety and operational specifications. An example based on a didactic manufacturing cell illustrates the used approach, where it is possible to observe the improvement of the task scheduling compared to sequential scheduling.

Keywords— Discrete Event Systems, manufacturing systems, supervisory control, genetic algorithm.

Resumo— O escalonamento de tarefas visa encontrar sequências ótimas de eventos que permitam aumentar os índices de produtividade em um sistema de produção. Este trabalho emprega algoritmos genéticos como metaheurística para obtenção de soluções ótimas de escalonamento, combinado com a teoria do controle supervisório (TCS) na geração automática de sequências de operações na produção de peças em um sistema didático de manufatura. A abordagem da TCS permite derivar estruturas de controle (supervisores) que formalmente garantem o funcionamento correto e seguro da planta. Por sua vez, o algoritmo genético busca pela melhor sequência possível de eventos, entre todas as sequências habilitadas pelos supervisores, explorando o paralelismo da planta e, assim, permitindo cumprir com objetivos de produção ao mesmo tempo que atende especificações de segurança e funcionamento. Um exemplo baseado em uma célula didática de manufatura ilustra a abordagem empregada, onde é possível observar a melhora do escalonamento de tarefas em comparação ao escalonamento sequencial.

Palavras-chave— Sistemas a eventos discretos, sistemas de manufatura, controle supervisório, algoritmo genético.

1 Introdução

O escalonamento em sistemas de manufatura busca encontrar sequências ótimas de tarefas, de modo que os objetivos de produção sejam alcançados, entre eles, aumento da produtividade e maximização na utilização de equipamentos (Hoitomt, Luh e Pattipati, 1993; Shi e Pan, 2005; Pinha, Queiroz e Cury, 2011).

Os problemas de escalonamento de tarefas são resolvidos por métodos analíticos e fornecem soluções ótimas para problemas que sejam pequenos, porém, quando o problema se torna maior, o esforço computacional cresce de maneira exponencial com o número de tarefas, operações e máquinas (Chan e Chan, 2004). O alto custo para obtenção de soluções ótimas pode ser proibitivo, pois o problema de escalonamento pertence à classe de problemas NP-complexo, cuja a solução polinomial não é conhecida (Oliveira *et al*, 2013). Ao comparar a otimização tradicional e métodos iterativos, as metaheurísticas usualmente podem encontrar boas soluções com menor esforço computacional e conseguem amenizar a dificuldade que alguns métodos heurísticos têm de escapar de ótimos locais (Gao *et al*., 2014).

Outra grande dificuldade na abordagem do escalonamento de tarefas está relacionada com a descrição analítica de restrições do problema. Restrições são necessárias para garantir que as sequências de produção sejam consideradas "legais" ou factíveis. Através de abordagens baseadas em sistemas a eventos discretos (SED), é possível encontrar uma representação formal para o problema, em termos de ações e eventos, possibilitando descrever matematicamente o problema e restrições associadas. Por exemplo, a teoria de controle supervisório (TCS) objetiva a modelagem, análise e controle de SED através de estruturas de controle (supervisores) que sejam minimamente restritivos, ou seja, evitam somente trajetórias de eventos consideradas proibidas ou indesejáveis (Silva *et al.*, 2011). Dessa forma, o escalonamento de tarefas pode ser formulado a partir da representação formal do sistema físico sob atuação dos supervisores, modelado de acordo com a TCS. Os modelos discretos obtidos pela TCS permitem representar a dinâmica do sistema através da sequência de eventos, permitindo a aplicação de métodos de otimização para encontrar a melhor sequência (escalonamento) de tarefas dentre todas aquelas habilitadas e permitidas pelo controle supervisório (CS). O CS fornece o conjunto de todas as sequências possíveis para a otimização, garantindo, assim, que a sequência encontrada no escalonamento seja uma boa solução, senão a ótima, além de ser factível, "legal" ou possível e segura de ser realizada pela planta. Assim, é possível combinar a natureza ótima da TCS com algoritmos e métodos de otimização para o escalonamento de sistemas de manufatura automatizados (Oliveira *et al*., 2013).

O controlador, denominado de supervisor, obtido no processo de síntese da TCS é considerado ótimo,

pois sua ação é minimamente restritiva, impedindo, pela desabilitação de eventos, a ocorrência de sequências consideradas indesejadas. A abordagem modular do controle supervisório permite aproveitar a natureza modular dos subsistemas da planta e especificações de controle para contornar o problema de explosão combinatória e reduzir a complexidade das estruturas de supervisão a serem implementadas nos dispositivos de controle. Em particular, esse trabalho emprega a abordagem do controle supervisório modular local (CSML) (Queiroz e Cury, 2002).

Diversas heurísticas têm sido usadas em problemas de escalonamento de tarefas, como algoritmos evolucionários, *Tabu Search*, *Simulated Annealing*, Algoritmo Colônia de Formigas, Sistemas Imunes Artificiais, entre outros (Becerra e Coello, 2005). Destaca-se também os Algoritmos Genéticos que podem ser definidos como em abordagens de permutação sistêmica baseada em atributos pré-definidos para encontrar uma melhor solução (Chan e Chan, 2004)

Esse trabalho utiliza uma adaptação da metodologia proposta por Pena *et al*. (2015) para a obtenção de sequências (escalonamento) de tarefas utilizando Algoritmos Genéticos (AGs) juntamente com a modelagem da TCS para tratamento das restrições do problema. Como estudo de caso para validação dessa abordagem, tem-se a aplicação em uma planta didática de manufatura. Como critério de otimização utilizado nesse trabalho, emprega-se o tempo de execução das tarefas.

O artigo é dividido conforme a seguir. A seção 2 traz uma breve introdução à TCS. A seção 3 apresenta o algoritmo genético com aplicação em problemas de escalonamento de tarefas. A metodologia que é proposta nesse trabalho é mostrada na seção 4. Para comprovação da metodologia, um estudo de caso é aplicado a esta metodologia. Esse estudo de caso e seus resultados obtidos são apresentados na seção 5. Ao final, são mostradas as conclusões do trabalho.

2 Teoria do Controle Supervisório (TCS)

A TCS permite a utilização de modelos formais baseados em autômatos e linguagens para representar o comportamento discreto ou dirigido a eventos de sistemas (Ramadge e Wonham, 1989; Cassandras e Lafortune, 2008). O comportamento em malha-aberta do sistema, denominado de planta, é restringido ou limitado de acordo com especificações de funcionamento e segurança. Tanto planta quanto especificações são modeladas por autômatos. Através do processo de síntese da TCS são obtidos os supervisores ou estruturas de controle em malha-fechada. Os supervisores, através da observação da evolução da planta, habilitam e desabilitam um subconjunto de eventos, permitindo que as especificações sejam atendidas.

A ocorrência de um evento, ou ação, no sistema pode ser representado por um símbolo e o conjunto de todas as possíveis sequências de símbolos que podem ocorrer fisicamente na planta é representado formalmente por uma linguagem *L*. Na TCS assume-se que os eventos são gerados de modo espontâneo pela planta, sendo possível inibir a ocorrência de um subconjunto deles. Assim, o alfabeto Σ que compõem a linguagem *L* é divido em dois subconjuntos, tal que $\Sigma = \Sigma_c \cup \Sigma_u$, onde Σ_c corresponde aos eventos controláveis, cuja ocorrência pode ser impedida por um elemento externo (supervisor); e Σ_u representa os eventos não-controláveis, cuja ocorrência não pode ser impedida.

A planta pode ser descrita por um autômato, ou gerador, *G* tal que: $G = (Q, \Sigma, \delta, q_0, Q_m)$, onde: Q é o conjunto de estados do autômato; Σ o alfabeto de símbolos de eventos; δ é a função de transição de estados; $q_0 \in Q$ o estado inicial; e $Q_m \in Q$ o conjunto de estados finais ou marcados contidos em *Q*. Um autômato pode ser visualizado como um grafo, onde vértices representam estados e arcos orientados, transições. Eventos controláveis são identificados por arcos com um traço. Denomina-se *linguagem gerada L(G)* o conjunto de todas as possíveis sequências geradas pelo autômato *G* e de *linguagem marcada* $L_m(G)$, o conjunto de todas as sequências possíveis de *G* que alcançam um estado marcado. Um autômato é dito bloqueante quando possui estados em que não se consegue evoluir até estados marcados, ou de modo equivalente, a sua linguagem corresponde ao prefixo-fechamento da sua linguagem marcada $\left(L(G) = \overline{L_m(G)}\right)$.

Supervisores são obtidos através do processo de síntese da TCS e garantem que especificações sejam atendidas pela observação da evolução da planta e pela ação de desabilitar a ocorrência dos eventos controláveis (Ramadge e Wonham, 1989). A condição necessária e suficiente para existência do supervisor não-bloqueante que atenda uma especificação *K* é a de controlabilidade, dada por $\overline{K}\Sigma_u \cap L(G) = \overline{K}$.

A abordagem do controle supervisório modular local (CSML) (Queiroz e Cury, 2002) permite explorar as características modulares dos componentes do sistema para a obtenção de supervisores modulares, com complexidade menor se comparada à abordagem monolítica em que um único supervisor é obtido.

O tempo, contudo, não é parte do modelo da TCS, mas em sistemas reais o tempo é inerente (Pena *et al.*, 2015). Para contornar o problema da não representação do tempo em modelos dirigidos a eventos discretos, Brandin e Wonham (1994) desenvolvem um modelo para sistemas a eventos discretos, denominada de Sistemas a Eventos Discretos Temporizados (SEDT), cuja representação temporal é adjacente a estrutura da TCS. Para contornar o problema de explosão de estados ocasionado pela modelagem em máquinas de estados, Chen e Lin (2000) propõem a modelagem usando máquinas de estados finitos com parâmetros (MEFcP). Ao inserir um parâmetro para a descrição de alguma situação, o número de estados pode ser drasticamente reduzido.

Uma máquina de estados finitos (MEF) é um autômato finito. A introdução de parâmetros em uma MEF permite estender a representação formal de autômatos possibilitando, assim, a inclusão de outras variáveis, como o tempo. Para isso, é preciso introduzir

alguns conceitos, descritos a seguir, conforme Chen e Lin (2000). Seja $p \epsilon P$ um vetor de parâmetros e P um espaço vetorial, e seja $g_a \epsilon G_a$ as guardas, ou predicados, dos parâmetros p. A função de transição δ, definida pela expressão 1 em uma MEF, passa a ser representada pela expressão 2. Assim, uma transição, apresentada na Figura 1, passa a ser interpretada da seguinte maneira: se no estado x_1 a guarda g for verdadeira e o evento σ ocorrer, então o próximo estado é x_2 e os parâmetros p são atualizados de acordo com uma função $f(p)$.

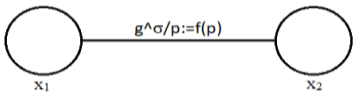

Figura 1. Transição de uma MEF com parâmetro.
Fonte: Chen e Lin (2000).

$$\delta: \Sigma \times Q \rightarrow Q \quad (1)$$

$$\delta: \Sigma \times Q \times Ga \times P \rightarrow Q \times P \quad (2)$$

Complementando a proposta de MEF com parâmetros (MEFcP), atividades de manufatura podem ser representadas como ações de início e fim. Nesta abordagem, um evento não-controlável modela uma resposta a um evento controlável, geralmente um comando ou ação de início de atividade. Como forma de inserir e apenas representar o parâmetro tempo que será utilizado no problema de escalonamento, e evitando assim o problema de complexidade e/ou explosão de estados que surge na abordagem SEDT ao inserir o tempo na modelagem, optou-se por utilizar a proposta de MEFcP. O autômato associado ao tempo com a guarda pode ser visualizado na Figura 2 (Costa, 2015; Pena *et al.*, 2015), onde o evento controlável de início de atividade é representado por a, o fim de atividade pelo evento não-controlável b, t_a o tempo de simulação e x a duração da atividade.

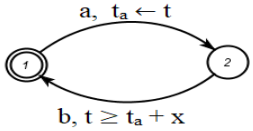

Figura 2. Autômato com guarda. Adaptado de Costa (2015).

3 Algoritmos Genéticos

3.1 Conceitos Básicos

Os algoritmos genéticos (AGs) foram propostos em (Holland, 1975) e se baseiam no princípio da seleção natural e recombinação genética. AGs correspondem a uma família de modelos computacionais inspirados pela evolução biológica e que codificam uma solução em potencial para um problema específico em uma estrutura de dados, denominada cromossomo, onde se aplicam operadores que objetivam preservar as informações críticas (Whitley, 1994). Um símbolo na cadeia (cromossomo) é chamado de gene (Croce *et al.*, 1995). A aptidão é geralmente definida como a probabilidade em que o indivíduo irá viver ou se reproduzir (Mitchell, 1998). *Crossover*, que representa a troca de informação genética entre indivíduos, e a mutação, alteração na codificação genética de um indivíduo de uma população, são exemplos de operadores empregados em AG.

3.2 Aplicação de Algoritmos Genéticos ao problema de escalonamento de tarefas

O primeiro passo para a construção de um AG constitui na definição de uma codificação que permite mapear as soluções como um todo em um conjunto de cadeias de símbolos (Croce *et al.*, 1995). Na questão da codificação para problemas de escalonamento de tarefas, diversos tipos de codificações existentes podem gerar escalonamentos que sejam inválidos (ilegais) (Becerra e Coello, 2005). Cheng *et al.* (1995) apresentada 9 técnicas de representação (codificação e decodificação) para o problema de escalonamento: baseada em operação; baseada em lista de preferência; baseada em tarefa, em chaves aleatórias, entre outras. Neste trabalho foi escolhida a representação baseada em operação. Essa representação codifica um escalonamento como uma sequência de operações, onde cada gene representa uma dessas operações e geralmente, esses genes são representados utilizando-se números naturais (Cheng *et al.*, 1995). Porém, a permutação desses números pode gerar escalonamentos ilegais devido as restrições de procedência e, para contornar essa situação, Gen, Tsujimara e Kobota (1994) propuseram a nomeação de todas as operações de uma tarefa com o mesmo símbolo e cuja interpretação está relacionada com a ordem de ocorrência. O tamanho do cromossomo é definido pelo número de operações realizadas. Essa forma de representação foi escolhida pela facilidade de interpretação, leitura, codificação e decodificação. A avaliação do indivíduo, neste trabalho, é realizada calculando a função objetivo, ou seja, o tempo de produção total do lote.

O operador de seleção, por sua vez, seleciona os cromossomos para a reprodução considerando a qualidade do cromossomo (Mitchell, 1998). Existem diversos métodos de seleção, como por exemplo, roleta, torneio ou classificação. Neste trabalho foi utilizado o método por classificação para a seleção de dois pais para o *crossover*. Neste método se seleciona a população e se atribui a cada cromossomo um valor de adequação determinado pela sua classificação e realiza-se o sorteio dos pais.

Após o processo de seleção, os indivíduos selecionados formam um conjunto de pais e eles são cruzados (*crossover*) para a produção de seus descendentes (Falkenauer e Bouffouix, 1991). Como exemplos de operadores de *crossover* é possível citar: operador de *crossover* de 1, 2 ou multipontos; operador de *crossover* parcialmente mapeado; operador de *crossover* de ordem; entre outros. Neste trabalho foi aplicado o operador de *crossover* de 2 pontos, no qual são escolhidos dois pontos aleatoriamente e os descendentes são obtidos dos pais ao trocar os bits entre os dois pontos de corte (Croce et al., 1995).

Logo após a geração da descendência, aplica-se o operador da mutação. A mutação tem como objetivo a

introdução de novo material genético em um indivíduo, em consequência, adicionar diversidade para as características genéticas da população. Como os outros operadores, existem diversos tipos para a mutação. No caso especial do escalonamento de tarefas, Croce *et al.* (1995) propõem utilizar o operador de mutação chamado *swap*, uma vez que esse operador com uma probabilidade p_m escolhe aleatoriamente 2 genes e os troca de posição na cadeia genética, diminuindo assim as chances que a mutação produza indivíduos ilegais.

Por fim, após todos os processos, os indivíduos já podem ser avaliados e inseridos na população formando assim uma geração na execução de um AG. Em um AG existe um conjunto de parâmetros a serem sintonizados, como o tamanho da população, probabilidades de *crossover* e mutação, número de geração e o sucesso do algoritmo depende desses detalhes. Como exemplo, o Algoritmo 1 apresenta o pseudocódigo de um AG simplificado.

Algoritmo 1 – Pseudocódigo de um AG. Adaptado de Mitchell (1998).

1. Geração de uma população inicial com *n* indivíduos;
2. **Enquanto** (geração atual < número de gerações)
3. Avaliação da população;
4. Seleção dos pais para o *crossover*;
5. *Crossover* dos pais para a geração de herdeiros;
6. Com uma probabilidade de mutação, os herdeiros são submetidos a uma mutação;
7. Inserção dos herdeiros na população;
8. **Fim enquanto**
9. Apresentação de resultados;

4 Metodologia

Este trabalho emprega uma adaptação da metodologia desenvolvida por Pena *et al*. (2015). Nesse trabalho, utiliza-se AGs para a resolução do problema de otimização de escalonamento de produção em função da minimização do tempo de produção, utilizando como descrição analítica das restrições do problema a abordagem da TCS e para a representação do parâmetro tempo a abordagem MEFcP. É possível implementar a aplicação em sistemas reais através de da integração de sistemas SCADA, onde será realizada a otimização *offline*, com o controle de baixo nível através de controladores lógicos programáveis (CLPs). Um dos motivos para escolha da metaheurística dos AG consiste na possibilidade de contornar problemas como a limitação de hardware e conseguir resultados com um tempo viável, ainda que subóptimos. A metodologia adaptada neste trabalho consiste de 3 etapas:

Etapa 1 – Modelagem da planta e síntese dos supervisores: nesta etapa são modelados os subsistemas da planta, juntamente com as especificações para realizar a síntese dos supervisores no *software* Supremica (Akesson *et al.* 2006).

Etapa 2 – Aplicação de AG ao problema de escalonamento de tarefas em plantas representadas por SED, em particular, segundo a TCS. Nessa etapa, primeiramente, é escolhida a representação do cromossomo para o AG. Na sequência, o AG é implementado de modo a encontrar uma sequência de produção do sistema, subóptima, porém melhor que a produção sequencial do lote. Os modelos obtidos na etapa 1 são usados para a descrição das restrições do sistema.

Etapa 3 – Implementação das estruturas de controle supervisório em equipamento industrial adequado, como CLPs e implementação da estrutura de otimização em sistema SCADA, ambos interconectados por meio de rede industrial. Nesse ponto é realizado a tradução do AG em uma linguagem de programação compatível com o SCADA e as estruturas de supervisão da TCS, o controle do SED, em um CLP, como, por exemplo, proposto em Vieira *et al*. (2017).

Até o momento a etapa 1 e 2 foram concluídas. A implementação da etapa 3 será apresentada em trabalho posterior.

5 Estudo de Caso e Resultados

O estudo de caso a ser abordado nesse trabalho é apresentado na sequência.

5.1 Descrição e Etapa 1: Modelagem da Planta

A planta didática de manufatura considerada está composta por: um Servo Robô 5250 da empresa Lab-Volt (*R*); uma mesa giratória ou carrossel (*G*); dois magazines ou *buffers* para alimentação de peças no sistema (B_1 e B_2); duas estações de manufatura (M_1 e M_2); e dois *buffers* de depósito de peças terminadas (D_1 e D_2). Os equipamentos se encontram no Laboratório de Robótica da UNIOESTE, campus Foz do Iguaçu, PR. As Figuras 3.a e 3.b apresentam, respectivamente, o servo robô, e a mesa giratória com os magazines de peças. As estações de manufatura (M_1 e M_2) atualmente são simuladas por elementos eletrônicos, como botões e sensores. A planta, cujo diagrama geral é apresentado na Figura 4, produz dois tipos de peças.

a) b)

Figuras: 3.a) Servo Robô 5250. 3.b) Mesa giratória e *buffers* de peças.

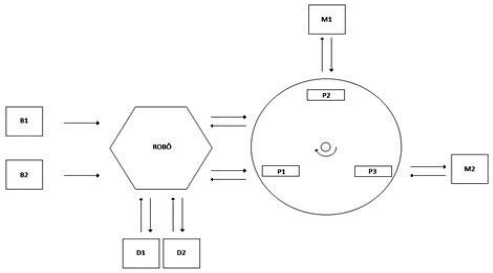

Figura 4. Diagrama geral do sistema de manufatura.

A ordem em que as mesmas percorrem o sistema de manufatura é mostrada pelas setas da Figura 4. Peças do tipo 2 são introduzidas no sistema pelo *buffer* B_2 e passam somente pelo primeiro processo de manufatura (M_1). O segundo tipo de peça (tipo 1), proveniente do *buffer* B_1, passa pelo processo de manufatura 1 (M_1) e, na sequência, pelo segundo processo (M_2). A mesa gira somente no sentido horário e possui 3 posições (P_1, P_2 e P_3) para peças. O robô (R) é responsável em colocar e retirar as peças da mesa giratória. As peças tipo 1 e 2 são colocadas na mesa na posição P_1. Peças do tipo 2 são retiradas da mesa na posição P_2 e peças do tipo 1, da posição P_3. Considera-se que os processos M_1 e M_2 são executados diretamente sobre as peças depositadas na mesa giratória, nas respectivas posições P_2 e P_3.

Seguindo então a metodologia proposta, dá-se início a Etapa 1. Nesta etapa da metodologia, os modelos formais da planta e especificações são realizados no *software* Supremica, onde são obtidos os supervisores para a modelagem da planta. O *software* Supremica possui ferramentas de verificação de controlabilidade das especificações e testes de conflito entre os supervisores modulares, no qual a modelagem realizada foi verificada. Após a modelagem, os modelos da planta e dos supervisores são codificados para o programa *Matlab*, onde são implementadas as funcionalidades de escalonamento com AG.

- **Modelagem da planta**

Utilizando a abordagem modular do CSML (Queiroz e Cury, 2002), para cada subsistema associado a planta, é modelado um autômato (G_x). Para cada subsistema pertencente a planta, é construído um autômato que o representa, a saber: robô (R); mesa giratória (G); processos de manufatura 1 e 2 (M_1 e M_2). Na tabela 1 encontra-se os eventos, classificados em controláveis e não-controláveis. Os subsistemas da planta são modelados utilizando as máquinas de estados finitos com parâmetros e são apresentados na Figura 5.

Tabela 1. Tabela de eventos do sistema.

Nome	Classificação	Descrição
E1	Controlável	Pegar material 1 em B_1 e depositá-la em G
E2	Não-controlável	Término de depósito do material 1 em G
E3	Controlável	Pegar material 2 em B_2 e depositá-la em G
E4	Não-controlável	Término do depósito do material 2 em G
E5	Controlável	Pegar peça 1 e depositá-la em D_1
E6	Não-controlável	Término do depósito da peça 1 em D_1
E7	Controlável	Pegar peça 2 e depositá-la em D_2
E8	Não-controlável	Término do depósito da peça 2 em D_2
E9	Controlável	Girar a mesa giratória
E10	Não-controlável	Término do giro
E11	Não-controlável	Sensor indica presença de material 1
E12	Não-controlável	Sensor indica ausência de material 1
E13	Não-controlável	Sensor indica presença de material 2
E14	Não-controlável	Sensor indica ausência de material 2
E15	Controlável	Iniciar o processo de manufatura 1 no material 1
E16	Não-controlável	Término do processo de manufatura no material 1
E17	Controlável	Iniciar o processo de manufatura 1 no material 2
E18	Não-controlável	Término do processo de manufatura 1 no material 2
E19	Controlável	Iniciar o processo de manufatura 2 no material 1
E20	Não-controlável	Término do processo de manufatura 2 no material 1

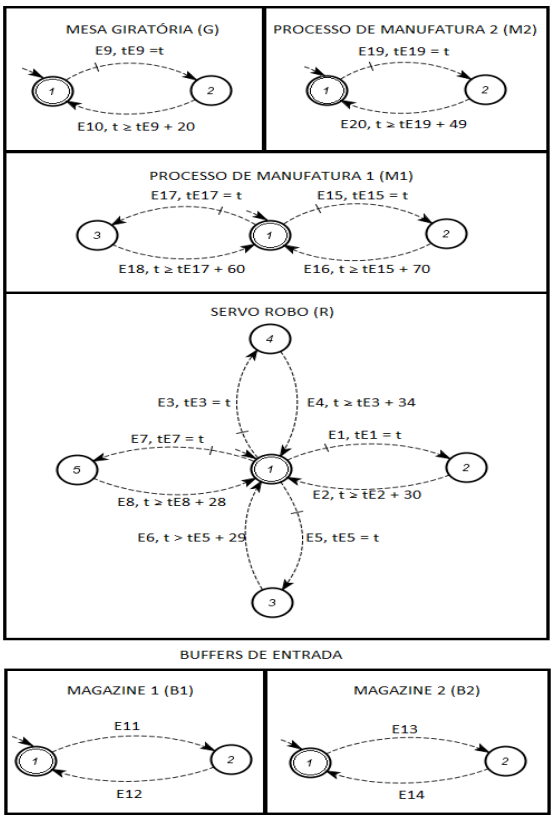

Figura 5. Modelos dos subsistemas da planta.

Nota-se que os subsistemas estão associados as suas respectivas guardas que apresentam o tempo (em segundos) de duração desses processos. Estes tempos

foram obtidos a partir de medição experimental dos processos executados sobre as bancadas. Também são incluídos na planta dois subsistemas que representam os sensores de chegada de peças nos *buffers* de entrada (B_1 e B_2).

- **Modelagem das Especificações**

O comportamento da planta, descrito pelos seis geradores da subseção anterior, representam o comportamento em malha-aberta do sistema. Porém, para a operação correta e segura do sistema, algumas restrições se impõem ao comportamento livre da planta. A partir das especificações são obtidas, a partir do processo de síntese da TCS, as estruturas de supervisão que irão garantir que a planta funcione em malha-fechada de acordo com os requisitos desejados. Assim, as especificações consideradas neste trabalho são:

1. Evitar o *underflow* e o *overflow dos buffers* M_1 e M_2 (especificações 1 e 2 relacionadas com os subsistemas robô, *buffer* 1 e *buffer* 2).
2. Evitar que ações de manufatura sejam realizadas e que o robô deposite ou retire peça enquanto a mesa estiver girando (especificações 5, 6, 7, 8, 10, 12, 13, 16 e 17 relacionadas com os subsistemas robô, mesa giratória, processo de manufatura 1 e 2).
3. Impedir que peças do tipo 2 sejam usinadas no processo M_2 antes de terem sido processadas em M_1 (especificação 15 relacionada com os subsistemas mesa giratória, processo de manufatura 1 e 2).
4. Evitar o giro da mesa quando ela estiver vazia (especificação 14 relacionada com os subsistemas mesa giratória, robô e processo de manufatura 1).

Na Figura 6 apresenta-se os geradores com as 17 especificações de operação e segurança desejadas para o comportamento da planta.

- **Síntese dos Supervisores**

Para cada uma das 17 especificações (E_i) são obtidos supervisores modulares S_i, a partir da especificação E_i e a planta local G_{loci}. G_{loci} é obtida a partir do produto síncrono dos modelos dos subsistemas envolvidos na respectiva especificação E_i. Por exemplo, para a especificação do depósito de peça na mesa giratória, é necessário compor o autômato que representa a planta local G_{loci} a partir do produto síncrono entre o autômato da mesa giratória (G) e o do robô (R). A partir dessa planta local, então, em composição com a respectiva especificação é possível obter o supervisor modular local S_i não-bloqueante e minimamente restritivo. Após a modelagem do sistema e síntese dos supervisores, esses modelos são codificados no software *Matlab*.

5.2 Etapa 2: Aplicação de AG ao problema do escalonamento representada pela TCS

Como comentado previamente, foi escolhida a representação baseada em operação para a implementação do AG. Para a produção de peças tipo 1 e 2, respectivamente, são necessárias as seguintes sequências de passos:

- Peça 1: sensor detecta presença do material 1, robô pega material 1 e o deposita na mesa giratória, mesa gira, o processo de manufatura 1 ocorre, a mesa gira novamente, processo de manufatura 2 ocorre e o robô retira a peça 1 da mesa giratória e a deposita no *buffer* de depósito 1;
- Peça 2: sensor detecta presença do material 2, robô pega material 2 e o deposita na mesa giratória, mesa gira, o processo de manufatura 1 ocorre e o robô retira a peça 2 da mesa giratória e a deposita no *buffer* de depósito 2.

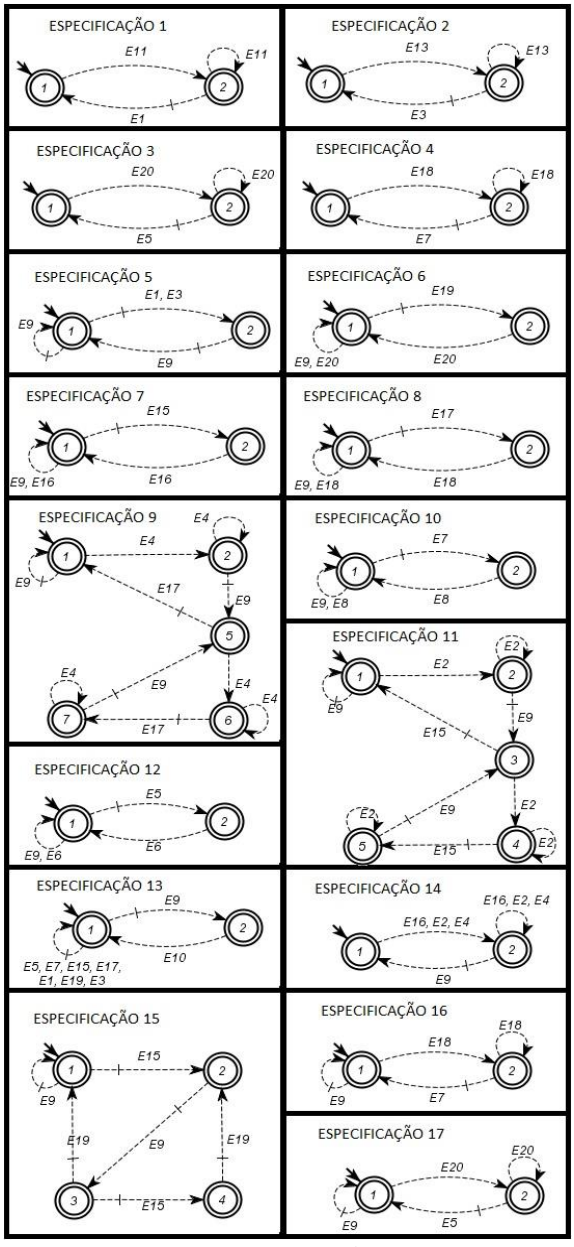

Figura 6. Modelagem das especificações da planta.

As sequências de passos correspondem, respectivamente, as seguintes sequências de eventos:

- Peça 1: E11-E1-E2-E9-E10-E15-E16-E9-E10-E19-E20-E5-E6;
- Peça 2: E13-E3-E4-E9-E10-E17-E18-E7-E8.

Observa-se que os eventos E11 (peça 1) e E13 (peça 2) são eventos que condicionam o início de operação do sistema. Além disso, também se observa que

todos os eventos seguintes, aos pares, demarcam o início e o fim de uma operação. Assim, na representação de um gene cada par de início e fim de uma operação será representado por um símbolo. Esse símbolo, na representação de AG utilizada é denominada de operação, daí o nome da representação.

O tamanho do cromossomo é determinado pelo número de peças do tipo 1 e do tipo 2. Por exemplo, caso um lote produza 2 peças do tipo 1 e 2 peças do tipo 2, o cromossomo terá 6 vezes 2 (número de peças tipo 1) mais 4 vezes 2 (número de peças tipo 2), ou seja, esse cromossomo terá 20 símbolos, um para cada operação necessária à produção da peça. Como símbolos dos cromossomos são empregados números naturais, sendo ímpar para indicar peças do tipo 1 e par para indicar peças do tipo 2. Desse modo, a sequências de eventos controláveis e não-controláveis é traduzida em uma sequência de operações (conjunto de comandos e respostas), e a cada operação está associado um gene no cromossomo, cujo número indica o tipo de peça que se está produzindo. Foi elaborado um *software* para a tradução da sequência de operações (cromossomo) em uma sequência de eventos que se baseia na seguinte representação: a primeira repetição do símbolo (operação) representa o primeiro par de eventos (controláveis e não-controláveis) da produção da peça; a segunda repetição, o segundo par de eventos; assim sucessivamente. Essa tradução leva em consideração a ocorrência de operações simultâneas.

Para a obtenção da melhor sequência de escalonamento (sequência com o menor tempo de produção), o AG foi implementado no software *Matlab*, versão 2017a, em um *notebook* com a seguinte configuração: processador Intel Core i7 3ª geração, 2GHz, 8GB de RAM e sistema operacional de 64 bits. O pseudocódigo implementado é apresentado no Algoritmo 2.

Algoritmo 2 – Pseudocódigo utilizado para a implementação.

```
% Geração da população inicial totalmente legal
1.  Enquanto (cromossomo atual < tamanho da população)
2.      Enquanto (gene atual < tamanho do cromossomo)
3.          Criação aleatória do gene no cromossomo;
4.          Decodificação do cromossomo na sequência de eventos correspondente;
5.          Verificação da legalidade do cromossomo;
6.          Se o cromossomo é legal
7.              Avança para o próximo gene;
8.          Senão
9.              Volta-se ao passo 3;
10.         Fim se
11.     Fim enquanto
12. Fim enquanto
% Inicio da evolução
13. Enquanto (geração atual < número de gerações)
14.     Avaliação da população através da função objetivo;
15.     Seleção dos dois pais para o crossover levando em consideração a qualidade de cada indivíduo;
16.     Crossover dos pais para a geração de 2 herdeiros;
17.     Decodificação dos herdeiros nas sequências de eventos correspondentes;
18.     Verificação da legalidade dos herdeiros pela TCS;
19.     Se os herdeiros são possíveis pela TCS
20.         Com uma probabilidade de mutação (pm), um dos herdeiros é submetido ao operador de mutação;
21.         Decodificação dos herdeiros nas sequências de eventos correspondentes;
22.         Verificação da legalidade dos herdeiros pela TCS;
23.         Se os herdeiros são possíveis pela TCS
24.             Inserção dos herdeiros na população, tomando lugar dos piores indivíduos;
25.             Avanço para a próxima geração;
26.         Senão
27.             Volta-se ao passo 15;
28.         Fim se
29.     Senão
30.         Volta-se ao passo 15;
31.     Fim se
32. Fim enquanto
33. Apresentação dos resultados;
% Fim do Algoritmo
```

5.3 Resultados obtidos

Tradicionalmente, em ambientes industriais, devido a sua natureza complexa, é comum realizar o escalonamento de tarefas de modo sequencial, ou seja, uma peça por vez, considerado o pior caso de produção. Ao utilizar o escalonamento baseado em algum critério de otimização, é possível encontrar sequências de processamento que são melhores que o modo sequencial.

Para o estudo de caso, considerou-se a representação baseada em operação para o cromossomo, com a produção de um lote de 20 peças, sendo 10 peças do tipo 1 e 10 peças do tipo 2. Os parâmetros escolhidos para o AG foram: tamanho de população de 20 indivíduos; probabilidade de mutação (p_m) de 25%; total de 500 gerações. No total de 100 simulações, o AG encontrou o resultado de 3400 u.t. Em contrapartida, o resultado para o processamento sequencial foi calculado em 3600 u.t. Observa-se que os resultados encontrados são significativamente superiores ao processamento sequencial, trazendo assim um grande benefício para implementação de um escalonamento de tarefas em um sistema de manufatura.

6 Conclusão

Neste trabalho foi apresentado uma metodologia para a otimização do tempo no sequenciamento de tarefas que combina a Teoria do Controle Supervisório (TCS) e com algoritmos genéticos (AGs). O comportamento dinâmico dirigido a eventos da planta é modelado através da TCS, sendo possível, assim, obter modelos formais que possibilitam a representação do sistema através de uma sequência de eventos. AG é empregado para encontrar a melhor sequência de eventos dentre todas habilitadas pelos supervisores, usando como base para representação a sequência de eventos descrita pelos modelos formais da TCS. Desse modo, as sequências ótimas encontradas para o escalonamen-

sempre atende a quesitos de segurança e operação da planta. Para validação da metodologia, tem-se a sua aplicação em uma planta didática de manufatura.

Para trabalhos futuros prevê-se a implementação das estruturas de controle supervisório em conjunto com o algoritmo genético através da integração de componentes de controle industrial (CLPs) e sistemas de supervisão (SCADA), além da comparação com outras abordagens de otimização heurística.

Agradecimentos

Os autores agradecem ao Programa de Mestrado em Engenharia Elétrica e Computação da UNIOESTE, campus Foz do Iguaçu, à Fundação Araucária e à CAPES pelos recursos financeiros e ao professor Carlos Henrique Farias dos Santos pela cessão dos equipamentos do Laboratório de Robótica dessa mesma universidade.

Referências Bibliográficas

Akesson, K.; Fabian, M.; Flordal, H. and Malik, R. (2006). Supremica-an integrated environment for verification, synthesis and simulation of discrete event systems. Discrete Event Systems, 2006 8th International Workshopon. IEE. pp 384-385.

Becerra, R. L. and Coello, C. A. C. (2005). A Cultural Algorithm for Solving the Job Shop Scheduling Problems. In Knowledge Incorporatoin in Evolutionary Computation.

Brandin, B. A. and Wonham, W. M. (1994). Supervisory Control of Timed Discrete-Event System. IEEE Transactions on Automatic Control. V. 39. N. 2. pp. 329-342.

Cassandras, C. G. and Lafortune, S. 2008. Introduction to Discrete Event Systems. 2nd ed. Springer.

Chan, F. T. S. and Chan, H. K. (2004). A comprehensive survey and future trend of simulation study on FMS scheduling. Journal of Intelligent Manufacturing. V. 15. N. 1. pp. 87-102.

Chen, Y. L. and Lin, F. (2000) Modeling of Discrete Event Systems Using Finite State Machines with Parameters. International Conference on Control Applications. pp. 941-946.

Cheng, R.; Gen, M. and Tsujimura, Y. (1995). A tutorial survey of job-shop scheduling problems using genetic algorithms – I. Representation. Computers & industrial engineering. V. 30. N. 4. pp. 983-997.

Costa, T. A. (2015). Metodologia CSO – Controle Supervisório e Otimização Aplicada a Resolução de Problemas de Sequenciamentos de tarefas em Sistemas Flexíveis de Manufatura. Tese de doutorado. Universidade Federal de Minas Gerais. Belo Horizonte.

Croce, F. D; Tadei, R. and Vota, G. (1995) A Genetic Algorithm for the job-shop problem. Computers & Operations Research. V. 22. N. 1. pp. 15-24.

Falkenauer, E. and Bouffouix, S. (1991). A Genetic Algorithm for Job-Shop. In IEEE Proceedings of Robotics and Automation. pp 824-829.

Gao, H.; Kwong, S.; Fan, B. and Wang, R. (2014). A Hybrid Particle-Swan Tabu Search Algorithm for Solving Job Shop Scheduling Problems. IEEE Transactions on Industrial Informatics. V. 10. N. 4. pp. 2044-2054.

Gen, M.; Tsujimara, Y. and Kubota, E. (1994). Solving job-shop scheduling problems by genetic algorithm. Systems, Man, and Cybernetics. V. 2. pp. 1577-1592.

Hoitomt, D. J.; Luh, P. B. and Pattipati, K. R. (1993). A Practical Approach to Job-Shop Problems. IEEE Transactions on Robitics and Automation. V. 9. N. 1. pp. 1-13.

Holland, J. H. (1975). Adaptation in natural and artificial systems. An introductory analysis with application to biology, control, and artificial intelligence. MIT Press.

Mitchell, M. (1998). An introduction to genetic algorithms. MIT Press.

Oliveira, A. C.; Costa, T. A.; Pena, P. N. and Takahashi, R. H. C. (2013). Clonal Selection Algorithms for Task Scheduling in Flexible Manufacturing Cell with Supervisory Control. Proc. of the IEEE Congress on Evolutionary Computation. pp. 982-988.

Pena, P. N.; Costa, T. A; Silva, R. S. and Takahashi, R. H. C. (2015). Control of Flexible Manufacturing System under model uncertainty using Supervisory Control Theory and evolutionary computation schedule synthesis. Journal Information Sciences: an International Journal. V. 329. N. 2, pp. 491-502.

Pinha, D. C.; Queiroz, M. H. and Cury, J. E. R. (2011). Optimal Scheduling of a Repair Shipyard Based on Supervisory Control Theory. Proc. of IEEE International Conference on Automation Science and Engineering. pp. 39-44.

Queiroz, M. H and Cury, J. E. R. (2002). Synthesis and Implementation of Local Modular Supervisory Control for a Manufacturing Cell. Proc. of the 6th International WODES. pp. 377-382.

Ramadge, P. J. G. and Wonham, W. M. (1989). The Control of Discrete Event Systems. Proc. of the IEEE. Vol. 77. N. 1. pp. 81-98.

Shi, L. and Pan, Y. (2005). An Efficient Search Method for Job-Shop Scheduling Problems. Transactions on Automation Science and Engineering. V. 2. N. 1. pp. 73-77.

Vieira, A. D.; Santos, E. A. P.; Queiroz, M. H.; Leal, A. B.; Neto, A. D. P.; Cury, J. E. R. (2017). A Method for Implementation of Supervisory Control of Discrete Event Systems. IEEE Transactions on Control Systems Technology. V. 25. N. 1. pp 175-191.

Whitley, D. (1994). A genetic algortithm tutorial. Statiscs and computing. V. 4. N. 2. pp. 65-85.