DISSERTAÇÃO DE MESTRADO Nº 936

PLANEJAMENTO DA PRODUÇÃO EM SISTEMAS A EVENTOS DISCRETOS - ANÁLISE LÓGICA E TEMPORAL

Lucas Vinícius Ribeiro Alves

DATA DA DEFESA: 27/07/2016

A474p

Alves, Lucas Vinícius Ribeiro.
Planejamento da produção em sistemas a eventos discretos - análise lógica e temporal [manuscrito] / Lucas Vinícius Ribeiro Alves. - 2016.
85 f., enc.: il.

Orientadora: Patrícia Nascimento Pena.

Coorientador: Ricardo Caldeira Takahashi.

Dissertação (mestrado) Universidade Federal de Minas Gerais, Escola de Engenharia.

Anexos: 81-85.

Bibliografia: f. 75-77.

1. Engenharia elétrica - Teses. 2. Sistemas de tempo discreto - Teses. I. Pena, Patrícia Nascimento. II. Takahaschi, Ricardo Caldeira. III. Universidade Federal de Minas Gerais. Escola de Engenharia. IV. Título.

CDU: 621.3(043)

Universidade Federal de Minas Gerais

Escola de Engenharia

Programa de Pós-Graduação em Engenharia Elétrica

PLANEJAMENTO DA PRODUÇÃO EM SISTEMAS A EVENTOS DISCRETOS - ANÁLISE LÓGICA E TEMPORAL

Lucas Vinícius Ribeiro Alves

Dissertação de Mestrado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do Título de Mestre em Engenharia Elétrica.

Orientadora: Profa. Patrícia Nascimento Pena

Belo Horizonte - MG

Julho de 2016

Este trabalho é dedicado à minha avó,
Maria Lúcia (in memoriam).

Agradecimentos

Os agradecimentos principais são direcionados à meus pais, Onofre e Flávia, e meu irmão Victor, por todo o apoio durante todos os anos de estudo. Agradeço também ao meu amigo Diego, pelo apoio e pelos conselhos em muitos momentos. Agradeço também aos meus demais amigos e familiares que de forma direta ou indireta contribuíram para o desenvolvimento desse trabalho

Agradecimentos especiais são direcionados ao Laboratório de Análise e Controle de Sistemas a Eventos Discretos[1] da Universidade Federal de Minas Gerais (LACSED), bem como à minha orientadora, professora Patrícia, e meu coorientador, professor Takahashi, pelos conselhos que foram essenciais ao desenvolvimento dessa dissertação.

[1] http://www.lacsed.eng.ufmg.br/

"If you believe in this world no one has died in vain
But don't you dare get to the top and not know what to do".
(11^{th} Dimension, Julian Casablancas)

Resumo

Este trabalho tem como objetivo apresentar soluções para o problema de planejamento da produção em Sistemas a Eventos Discretos. A aplicação da Teoria de Controle Supervisório limita o universo de busca por sequências de produção segura, de forma que faz sentido utiliza-se da estrutura do supervisor (ou da planta sob controle) para determinar as melhores sequências. São propostas quatro soluções para o problema, as duas primeiras buscam maximizar o paralelismo, e as últimas duas buscam minimizar o tempo de produção.

Palavras-chaves: Sistemas a Eventos Discretos, Escalonamento, Paralelismo

Abstract

This work aims to present solutions for a production planning problem in Discrete Event Systems. The application of Supervisory Control Theory limits the search universe for safe production sequences, so it makes sense to use the supervisor structure (or plant under control) to determine the best sequences. Four solutions to the problem are proposed, the first two seek to maximize the parallelism and the last two seek to minimize production time.

Key-words: Discrete Event Systems. Scheduling. Parallelism

Lista de ilustrações

Figura 1 – Exemplo de Autômato Finito Determinístico 32
Figura 2 – Diagrama de um Simulador em Sistemas a Eventos Discretos 35

Figura 3 – Diagrama da Pequena Fábrica 38
Figura 4 – Autômatos para as máquinas M_1 e M_2 e para a especificação E. 38
Figura 5 – Supervisor da Pequena Fábrica 39
Figura 6 – Forma genérica de um autômato de desenrolamento com um conjunto de eventos Σ e profundidade n 41
Figura 7 – Grafo acíclico obtido a partir do supervisor da Pequena Fábrica para uma busca com profundidade de 8 eventos. 41
Figura 8 – Exemplo de execução do algoritmo de Máximo Paralelismo Lógico . . . 43
Figura 9 – Grafo acíclico do supervisor da Pequena Fábrica com agendas de eventos associadas aos estados, para uma profundidade de 8 eventos. 49
Figura 10 – Exemplo de execução do algoritmo de Máximo Paralelismo com Restrições Temporais 51

Figura 11 – Grafo acíclico do supervisor da Pequena Fábrica com agendas de eventos associadas aos estados e duplicidade de estados, para uma profundidade de 8 eventos. 57
Figura 12 – Exemplo de execução do algoritmo de Menor Tempo Exato 58
Figura 13 – Grafo acíclico do supervisor da Pequena Fábrica com agendas de eventos associadas aos estados e duplicidade de estados e heuristica de poda, para uma profundidade de 8 eventos. 61

Figura 14 – Autômatos para as máquinas M_1 e M_2 e para a especificação E acompanhados do número de tarefas ativas em cada estado. 64
Figura 15 – Supervisor da Pequena Fábrica 64
Figura 16 – Diagrama da Sistema Flaxível de Manufatura 66
Figura 17 – Plantas do Sistema Flexível de Manufatura 67
Figura 18 – Gráficos relacionando o número de tarefas ativas no SFM para cada evento executado, para uma sequência gerada por cada algoritmo. 70

Lista de tabelas

Tabela 1 – Número de tarefas ativas em cada estado do supervisor da Pequena Fábrica 39

Tabela 2 – Intervalo de tempo entre pares de eventos da Pequena Fábrica 64
Tabela 3 – Tempo de execução médio dos algoritmos desenvolvidos, em milissegundos. 65
Tabela 4 – Intervalo de tempo entre pares de eventos 68
Tabela 5 – Tempo de execução médio dos algoritmos desenvolvidos, em segundos. 68
Tabela 6 – Tempo de produção médio do lote das sequências obtida pelo algoritmo em unidades de tempo. 69
Tabela 7 – Paralelismo acumulado ($F_{ta}(s^*)$) médio das sequências obtidas pelos algoritmos 70
Tabela 8 – Tempo de produção para tempos de trabalho das plantas gerados aleatoriamente com distribuição normal. 71

Índice de algoritmos

1 Calcula o caminho que maximiza o paralelismo no grafo acíclico de um supervisor 42

2 Calcula o caminho que maximiza o paralelismo no grafo acíclico de um supervisor com restrições temporais 52

3 Calcula o caminho que minimiza o *makespan* no grafo acíclico de um supervisor 56

4 Calcula o caminho que minimiza o *makespan* no grafo acíclico de um supervisor, utilizando uma heurística 59

Lista de abreviaturas e siglas

SED	Sistema a Eventos Discretos.
TCS	Teoria de Controle Supervisório.
MPL	Algoritmo de Máximo Paralelismo Lógico.
AGT	Algoritmo Guloso direcionado ao Tempo.
AGP	Algoritmo Guloso direcionado ao Paralelismo.
MPP	Algoritmo do Máximo Paralelismo Projetado.
MPT	Algoritmo do Máximo Paralelismo com Restrições Temporais.
MTH	Algoritmo de Menor Tempo Heurístico.
MPR	Algoritmo do Máximo Paralelismo com Recálculo.
MTE	Algoritmo de Menor Tempo Exato.

Lista de símbolos

G	Um autômato qualquer
S	Um supervisor qualquer
$\in$	Pertence a
$\notin$	Não pertence a
:	Tal que
$\exists$	Existe
$\forall$	Para todo
$\bigcap$	Operador interseção de conjuntos
$\bigcup$	Operador união de conjuntos
Σ	Alfabeto ou conjunto de eventos
Σ_c	Conjunto de eventos controláveis
Σ_u	Conjunto de eventos não controláveis
ϵ	Palavra nula
Σ^*	Conjunto de todas as palavras que podem ser formadas com os eventos de Σ incluindo a palavra nula ϵ
Q	Conjunto de estados
Q_m	Conjuntos de estados marcados
G, S	Autômatos
K, L	Linguagens
$\mathcal{L}(G)$	Linguagem Gerada do autômato G
$\mathcal{L}_m(G)$	Linguagem Marcada do autômato G
T	Função intervalo de tempo entre dois eventos relacionados
f_{ta}	Função de tarefas ativas
F_{ta}	Função acumulativa de tarefas ativas

f_T	Função temporal (função temporal expandida)
Δ	Função maior caminho entre dois estados
w	Função peso entre dois estados

Sumário

Introdução **23**

1 Revisão da Literatura **25**

2 Preliminares **29**

2.1 Sistemas a Eventos Discretos 29

2.1.1 Teoria de Linguagens e Autômatos 29

2.1.1.1 Linguagens 29

2.1.1.2 Autômatos 32

2.2 Teoria de Controle Supervisório 33

2.3 Ferramenta Computacional 34

2.4 Simulação de Sistemas a Eventos Discretos 34

2.5 Heurística de Poda 36

3 O Máximo Paralelismo **37**

3.1 Máximo Paralelismo Lógico 37

3.1.1 Definição do Problema 40

3.1.2 Manipulação do Supervisor e Algoritmo 40

3.1.2.1 Complexidade do Algoritmo 44

3.1.2.2 Corretude do Algoritmo 44

3.1.3 Vantagens e Desvantagens 46

3.1.4 Problemas na Execução Temporal e Recálculo 46

3.1.4.1 Complexidade do Algoritmo com Recálculo 47

3.1.5 Execução da Sequência Maximamente Paralela Projetada 47

3.2 Máximo Paralelismo com Restrições Temporais 47

3.2.1 Definição do Problema 49

3.2.2 Manipulação do Supervisor e Algoritmo 50

3.2.2.1 Complexidade do Algoritmo 52

3.2.2.2 Simulação e Estimação do Tempo 53

3.2.3 Vantagens e Desvantagens 54

4 Busca pelo Menor Tempo **55**

4.1 Menor Tempo Exato 55

4.1.1 Vantagens e Desvantagens 58

4.2 Menor Tempo Heurístico 58

4.2.1 Vantagens e Desvantagens 61

5 Testes e Análise de Resultados 63
5.1 Plantas para Teste de Execução 63
5.1.1 Pequena Fábrica (PF) 63
5.1.2 Sistema Flexível de Manufatura (SFM) 66
5.1.2.1 Tempos de Execução Aleatórios 71

Conclusão 73
5.2 Trabalhos Futuros 73

Referências 75

Apêndices 79

APÊNDICE A Log de Simulação 81

Introdução

O recurso mais valioso que uma indústria possui é o tempo e, tendo isso em vista, esse trabalho busca encontrar sequências de execução que minimizam o tempo de produção de uma determinada quantidade de produtos.

Grande parte dos sistemas atuais, em algum grau de abstração, reage à ocorrência de eventos, como o início e fim de uma tarefa ou o acionamento de um sensor. Esta característica, presente em sistemas de manufatura, sistemas robóticos, no tráfego e até mesmo em softwares é a principal característica que define um sistema a eventos discretos (SED).

Nesse contexto, dentre as inúmeros maneiras pelas quais é possível coordenar o funcionamento da planta garantindo segurança de operação, alguns modos de operação são mais eficientes que outros quanto à utilização de recursos.

Neste trabalho, será utilizada a representação de sistemas a eventos discretos por meio da teoria de autômatos e de linguagens (modelo RW). Essa abordagem permite a distinção entre o sistema que deseja-se controlar (planta) e o sistema que realiza o controle (supervisor).

A principal vantagem da representação de SED por autômatos e linguagens é a Teoria de Controle Supervisório (TCS), um *framework* que, a partir da planta e de especificações de segurança, gera um supervisor ótimo, no sentido de ser minimamente restritivo. A teoria de controle supervisório elimina a dependência da experiência do projetista na síntese de um dispositivo de controle.

Um supervisor obtido pela TCS permite um número elevado de sequências de operação diferentes, todas que não ferem especificações de segurança, porém essas sequências apresentam desempenho diferente quanto ao uso de recursos e tempo de execução.

No contexto da TCS, o problema de sequenciamento de tarefas tem como restrições a própria topologia do supervisor, de forma que o problema de sequenciamento pode ser tratado como um problema de busca no supervisor. Dessa forma, podemos tratar o problema de sequenciamento como um problema de planejamento em grafos.

Objetivos

O objetivo principal desse trabalho é propor métodos que permitam encontrar sequências em um supervisor, obtido pela Teoria de Controle Supervisório, que minimizem o tempo de produção de lotes de produtos. Essa busca deve ser realizada de maneira

indireta, ou seja, heurística, visto que a busca pela solução exata é inviável. Como objetivos secundários, podem-se citar:

- Desenvolver algoritmos determinísticos, com complexidade polinomial
- Avaliar a relação entre o paralelismo de tarefas e a minimização do tempo de produção.
- Realizar o que se propõe acima sem impor modificações na Teoria de Controle Supervisório, de forma que os algoritmos desenvolvidos possam ser usados de uma forma mais abrangente.

Organização do Trabalho

No capítulo 1 estão as preliminares, introduzindo os conceitos principais de Sistemas a Eventos Discretos, Teoria de Controle Supervisório e Grafos Direcionados.

Uma revisão da literatura sobre o problema de escalonamento de tarefas, do qual se trata esse trabalho, tanto no campo de sistemas a eventos discretos quanto na área de pesquisa operacional é apresentada no capítulo 2.

No capítulo 3 apresenta-se a primeira técnica desenvolvida nesse trabalho, baseada em encontrar sequências que maximizam o número de tarefas ativas a cada instante. Essa técnica é apresentada de duas formas, uma puramente lógica e outra que utiliza o tempo de execução das tarefas.

No capítulo 4, é apresentado um algoritmo de força bruta, que calcula o tempo verificando todos os caminhos existentes, e um algoritmo, baseado no de força bruta, com uma heurística.

O capítulo 5 faz testes dos quatro algoritmos desenvolvidos, bem como uma análise dos resultados, tanto em relação ao tempo de execução quanto em relação à otimalidade do resultado.

1 Revisão da Literatura

O problema de escalonamento de tarefas e planejamento da produção é recorrente área industrial. Minimizar recursos gastos é essencial na indústria e um dos recursos mais valiosos é o tempo. Em vista disso, a investigação e aplicação de técnicas eficientes para o escalonamento de tarefas e otimização são a chave para responder a esta demanda e assim aumentar a eficiência da produção. Segundo Wang, Yuan e Xiaobing (2008) por meio do aperfeiçoamento das técnicas de escalonamento de tarefas é possível obter vantagem competitiva na produção.

Encontrar uma sequência que minimiza o tempo de produção (*makespan*) de uma quantidade finita de produtos é um problema computacionalmente difícil, visto que a solução tem complexidade não polinomial (GAREY; JOHNSON, 1979).

O problema de escalonamento é recorrente na literatura, tendo sido usualmente divididos em duas classes, problemas determinísticos e problemas estocásticos (AYTUG et al., 2005).

Apresar de qualquer tempo de execução não ser puramente determinístico, em muitos casos as incertezas são previsíveis o suficiente para que possam ser ignoradas e os tempos possam ser considerados determinísticos. Nas abordagens determinísticas, usualmente o tempo de funcionamento de cada parte do sistema é dado por um valor constante ao invés de um variável aleatória. Neste trabalho, consideramos que os tempos são determinísticos e previsíveis, porém, as sequências obtidas pelos algoritmos apresentados podem ser utilizadas mesmo que os tempos do processo real sejam aleatórios ou diferentes daqueles utilizados na otimização.

Escalonamento de tarefas se refere à alocação, no tempo, de recursos finitos para tarefas do processo produtivo, dado um critério de otimização (PINEDO, 2012). Durante os anos, diversos formalismos, para tratar o problema de escalonamento, foram desenvolvidos, como no contexto de programação matemática (SCHRIJVER, 1986), Redes de Petri (LóPEZ-MELLADO; VILLANUEVA-PAREDES; ALMEYDA-CANEPA, 2005; LIU et al., 2016), Autômatos Temporizados (ABDEDDAïM; ASARIN; MALER, 2006), Verificação (HERZIG et al., 2014), entre outros.

No contexto da Teoria de Controle Supervisório (TCS) (RAMADGE; WONHAM, 1989) de Sistemas a Eventos Discretos, o problema de sequenciamento é extremamente importante, visto que esta abordagem fornece a solução minimamente restritiva, mas não indica uma sequência de produção que garanta algum critério de otimalidade. Nesse sentido, existem diversas abordagens, na literatura, referentes ao escalonamentos de Sistemas a Eventos Discretos (KOBETSKI; FABIAN, 2006; PARK; YANG, 2009; PINHA;

QUEIROZ; CURY, 2011; SU; SCHUPPEN; ROODA, 2012), focando usualmente na minimização do *makespan* (horizonte finito), ou na maximização do *throughput* da produção contínua.

Como a solução exata, quanto à minimização do *makespan*, é em geral intratável computacionalmente, as abordagens mais comuns na literatura são as heurísticas e meta heurísticas (PENA et al., 2016; ALMEDER; MöNCH, 2011). A grande desvantagem da utilização de meta heurísticas é o tempo gasto analisando sequências infactíveis, ou seja, que nem sequer são fisicamente realizáveis na planta.

Quando o problema de escalonamento não leva em consideração restrições quanto à quantidade de recursos usados na operação, definindo apenas restrições temporais e de precedência, ele se torna um processo de planejamento (GHALLAB; NAU; TRAVERSO, 2004). O uso da Teoria de Controle Supervisório na obtenção de um supervisor garante uma operação segura e restrições de precedência, permitindo que o problema de encontrar sequência de eventos ótima, dado um critério de otimização qualquer, seja tratado como um problema de planejamento (*planning*).

As técnicas de planejamento mais comuns são aquelas baseadas em busca, como busca em profundidade, busca em largura ou busca pelo menor/maior caminho. Na área de inteligência artificial, um plano é uma sequência de ações (eventos) para atingir um objetivo a partir de um estado inicial e, usualmente, a abordagem mais comum é tratar o problema como um problema de encontrar um caminho no espaço de estados (MACKWORTH, 2010).

No contexto de Sistemas a Eventos Discretos, em especial utilizando a representação por autômatos, existem trabalhos que utilizam a abordagem de planejamento para resolver problemas de escalonamento de eventos (WARE; SU, 2016), mas usualmente esses trabalhos não utilizam a Teoria de Controle Supervisório tradicional.

Este trabalho visa tratar o problema de planejamento em Sistemas a Eventos Discretos sem realizar modificações nas técnicas de controle já consolidadas na área. Todas as informações adicionais são armazenadas em funções auxilares, que não influenciam na síntese de supervisores nem nas propriedades de controlabilidade e não bloqueio dos supervisores já existentes.

Dentre as soluções propostas, existem soluções não exatas, que utilizam heurísticas para encontrar resultados satisfatórios para o problema (OLIVEIRA et al., 2013; COSTA; PENA; TAKAHASHI, 2013; PENA et al., 2016) . Heurísticas são, critérios, métodos ou princípios que auxiliam a decidir, entre várias alternativas de ação aquela que promete ser mais efetiva no sentido de alcançar algum objetivo (PEARL, 1984).

Um jogador de xadrez, quando se depara com uma situação na qual existem diversos movimentos possíveis, escolhe um movimento particular como mais efetivo, simples-

mente porque esse movimento resulta em um tabuleiro que “aparenta ser melhor” do que o alcançado pelos demais movimentos. O critério de “parecer melhor” é muito mais simples de ser aplicado pelo jogador do que, de fato, determinar rigorosamente qual movimento leva a um xeque-mate. O fato de que jogadores de xadrez nem sempre vencem indica que a heurística utilizada por eles não garante a seleção do movimento mais efetivo.

A maior parte dos problemas complexos requerem a avaliação de um número imenso de possibilidades a fim de determinar uma solução exata, e o tempo para avaliar essas possibilidades pode, facilmente, ser maior que um tempo de vida. Em geral, boas heurísticas fornecem indicação, entre várias ações possíveis, aquela que seria melhor, mas não garante que a ação indicada é, de fato, a melhor.

2 Preliminares

Neste capítulo são definidos os conceitos básicos necessários para o desenvolvimento e compreensão desta dissertação. Primeiramente serão introduzidos os Sistemas a Eventos Discretos, em seguida será discutida a Teoria de Controle Supervisório e, por último, é realizada uma breve introdução aos grafos direcionados e sua relação com os autômatos.

2.1 Sistemas a Eventos Discretos

Sistemas a eventos discretos são sistemas dinâmicos que interagem com entidades externas por meio de estímulos, chamados eventos. Esses eventos podem ser o início ou fim de uma tarefa, o acionamento de um sensor, além de eventos internos, como o fim de uma temporização. Em geral, define-se estado como a configuração do sistema entre a ocorrência de dois eventos consecutivos, ou seja, a ocorrência de um evento acarreta em uma transição de estado no sistema.

Pode-se classificá-los como sistemas dinâmicos não lineares de estados discretos e orientados a eventos. Devido à estrutura discreta dos eventos, e sua assincronia, as representações convencionais de sistemas, por equações diferenciais e equações de diferenças, não são convenientes. Se fez necessário então desenvolver outros formalismos a fim de representar sistemas a eventos discretos. Alguns desses formalismos são:

- Linguagens e Autômatos (Modelo RW)
- Redes de Petri
- Álgebra (Max, +)
- Teoria de Filas
- Álgebra de Processos

Neste trabalho, a representação de SEDs é feita por meio de linguagens e autômatos.

2.1.1 Teoria de Linguagens e Autômatos

2.1.1.1 Linguagens

Seja Σ um conjunto finito de símbolos, usualmente referido como alfabeto. Seja Σ^+ o conjunto de todas as sequências finitas de símbolos, na forma $\sigma_1\sigma_2...\sigma_k$ onde $k > 0$

e $\sigma_i \in \Sigma$. A sequência vazia, com nenhum símbolo, é representada por ϵ e:

$$\Sigma^* = \{\epsilon\} \cup \Sigma^+$$

Uma sequência $s \in \Sigma^*$ é, usualmente, chamada de palavra ou cadeia sobre o alfabeto Σ. O comprimento de uma sequência $|s|$ é definido como:

$$|\epsilon| = 0$$
$$|\sigma_1\sigma_2...\sigma_k| = k.$$

A concatenação de duas cadeias $s, u \in \Sigma^*$ forma a palavra $t = su$, onde $t \in \Sigma^*$. Neste caso, diz-se que s é um prefixo de t e u é um sufixo de t.

Um subconjunto qualquer $L \subseteq \Sigma^*$ é chamado de linguagem e, sobre elas, é possível definir diversas operações em conjuntos (união, complemento, interseção e diferença) e a concatenação.

Neste trabalho, a fim de simplificar a notação, uma linguagem composta por apenas uma sequência $\{u\}$ será representada apenas por u, o símbolo da união $\cup$ será substituído pelo símbolo $+$, a concatenação é representada pela justaposição de duas linguagens, a linguagem vazia é representada por $\emptyset$.

A concatenação de duas linguagens L_1 e L_2 é dado por:

$$L_1L_2 = \{u_1u_2 \,|\, u_1 \in L_1 \wedge u_2 \in L_2\}$$

A concatenação é uma operação associativa que admite como identidade linguagem composta pela sequência vazia ϵ, ou seja, $L\epsilon = \epsilon L$ e, além disso, é uma operação distributiva sobre a união, logo, $L(L_1 \cup L_2) = LL_1 \cup LL_2$.

A potência de uma linguagem é definida como $L^0 = \epsilon$, $L^1 = L$ e, por indução, $L^n = L^{n-1}L$. A estrela de Kleene de uma linguagem L, denotada por L^*, é a união de todas as potências de L:

$$L^* = \sum_{n \geq 0} L^n$$

O prefixo fechamento de uma linguagem L, denotado por $\overline{L}$ é uma linguagem que contém todos os prefixos de sequências pertencentes a L, incluindo a sequência vazia ϵ:

$$\overline{L} = \{s \,|\, su \in L\}$$

Outra operação comum, aplicada em sequências e linguagens, é a projeção natural, partindo de um alfabeto Σ_1 para um alfabeto menor Σ_2, tal que $\Sigma_2 \subseteq \Sigma_1$. A projeção para uma sequência é definida como:

$$\begin{aligned} P(\epsilon) &= \epsilon \\ P(\sigma) &= \begin{cases} \sigma & \text{se } \sigma \in \Sigma_2 \\ \epsilon & \text{se } \sigma \in \Sigma_1 \setminus \Sigma_2 \end{cases} \\ P(s\sigma) &= P(s)P(\sigma) \text{para } s \in \Sigma_1^*, \sigma \in \Sigma_1. \end{aligned}$$

A projeção inversa de uma sequência mapeia uma sequência de um alfabeto menor Σ_2 em um alfabeto maior Σ_1, sendo definida como:

$$P^{-1}(t) = \{s \in \Sigma_1 \,|\, P(s) = t\}.$$

É importante observar que a projeção inversa retorna todas as sequências do alfabeto Σ_1 que projetadas formam t, ou seja, $s \in P^{-1}(P(s))$. Ambas as definições, de projeção e projeção inversa podem ser estendidas para linguagens como mostrado a seguir:
Para $L \subseteq \Sigma_1^*$:

$$P(L) = \{t \in \Sigma_2^* \,|\, (\exists s \in L)\,[P(s) = t]\}$$

Para $L \subseteq \Sigma_2^*$:

$$P^{-1}(L) = \{s \in \Sigma_1^* \,|\, (\exists t \in P(L))\,[P(s) = t]\}$$

Dado um alfabeto Σ, existe um conjunto de linguagens, chamado conjunto de linguagens regulares, $F \in 2^{\Sigma^*}$ que apresenta as seguintes propriedades:

1. F contém as linguagens $\emptyset$ e σ para $\sigma \in \Sigma$.
2. F é um conjunto fechado sob as operações de união, concatenação e estrela de Kleene.

Uma característica importante das linguagens regulares é a possibilidade de representá-las como autômatos ou expressões regulares. Uma expressão regular é uma expressão formal definida recursivamente como:

1. As linguagens $\emptyset$ e ϵ são expressões regulares.
2. Qualquer símbolo $\sigma \in \Sigma$ é uma expressão regular.
3. Se e_1 e e_2 são expressões regulares, então $(e_1 + e_2)$, $(e_1 e_2)$ e e^* também são expressões regulares.

2.1.1.2 Autômatos

Um autômato finito determinístico é definido por uma 5-tupla $G = (Q, \Sigma, \delta, q_0, Q_m)$, onde Q é um conjunto finito de estados, Σ é o conjunto finito de símbolos (alfabeto), $\delta : Q \times \Sigma \to Q$ é a função de transição de estados, $q_0 \in Q$ é o estado inicial e $Q_m \subseteq Q$ é o conjunto de estados marcados de Q. Normalmente, a representação mais conveniente de autômatos é por meio de grafos direcionados com arestas representando transições, vértices representando estados, o estado inicial é indicado por uma seta e os estados marcados são indicados por círculos concêntricos, como mostrado na Figura 1.

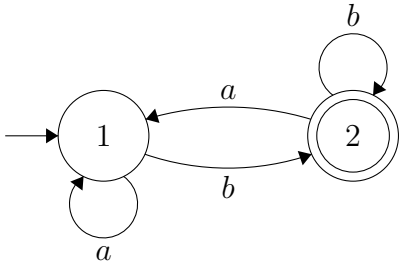


Figura 1 – Exemplo de Autômato Finito Determinístico

Um caminho, em um autômato, é denotado por:

$$c : q_0 \xrightarrow{\sigma_1} q_1 ... q_{n-1} \xrightarrow{\sigma_n} q_n$$

ou, de uma forma compacta, como $c : q_0 \xrightarrow{\sigma_1 ... \sigma_n} q_n$. O conjunto de todas as sequências de símbolos do alfabeto Σ que, partindo do estado inicial, formam um caminho num autômato G compõem a linguagem gerada pelo autômato, representada por $\mathcal{L}(G)$. Além disso, a linguagem marcada de um autômato é definida como um subconjunto da linguagem gerada, $\mathcal{L}_m(G) \subseteq \mathcal{L}(G)$, composto por todas as sequências que terminam em um estado marcado.

Existe também o caminho vazio, representado por $q \xrightarrow{\epsilon} q$, onde ϵ é a palavra vazia, e para qualquer autômato G, $\epsilon \in \mathcal{L}(G)$.

Existem diversas operações que podem ser aplicadas a autômatos. As principais operações aplicadas a um autômato são a de Parte Acessível, Parte Coacessível e Trim.

A parte acessível de um autômato é similar ao autômato original, porém todos os estados que não podem ser alcançados a partir do estado inicial são removidos. Esta

operação não modifica a linguagem gerada nem a linguagem marcada do autômato, pois não existe um caminho que, partindo do estado inicial leve aos estados removidos.

A parte coacessível de um autômato é similar ao autômato original, porém todos os estados que não podem alcançar um estado marcado são removidos. Essa operação modifica a linguagem gerada do autômato, porém não modifica a linguagem marcada.

O Trim de um autômato é o resultado da aplicação das operações de parte acessível e parte coacessível, em qualquer ordem, sobre o autômato. A operação não modifica a linguagem marcada do autômato.

Existem também operações de composição de múltiplos autômatos, permitindo a representação do comportamento conjunto dos autômatos originais. As principais operações entre dois ou mais autômatos são o produto e a composição paralela (ou produto síncrono).

O produto de dois autômatos $G_1 = (Q_1, \Sigma_1, \delta_1, q_{01}, Q_{m1})$ e $G_2 = (Q_2, \Sigma_2, \delta_2, q_{02}, Q_{m2})$ é dado por $G_{1\times2} = (Q_1 \times Q_2, \Sigma_1 \cap \Sigma_2, \delta_{1\times2}, (q_{01} \times q_{02}), Q_{m1} \times Q_{m2})$ onde:

$$\delta_{1\times2}((q_1, q_2), \sigma) = \begin{cases} (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma)) \text{ se } \delta_1(q_1, \sigma) \text{ e } \delta_2(q_2, \sigma) \text{ estão definidos} \\ \text{indefinido caso contrário} \end{cases}$$

Do ponto de vista de linguagens, o produto entre dois autômatos pode ser representado como $\mathcal{L}(G_{1\times2}) = \mathcal{L}(G_1) \cap \mathcal{L}(G_2)$.

A composição paralela entre dois autômatos $G_1 = (Q_1, \Sigma_1, \delta_1, q_{01}, Q_{m1})$ e $G_2 = (Q_2, \Sigma_2, \delta_2, q_{02}, Q_{m2})$ é dada por $G_{1||2} = (Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, \delta_{1||2}, (q_{01} \times q_{02}), Q_{m1} \times Q_{m2})$ onde:

$$\delta_{1||2}((q_1, q_2), \sigma) = \begin{cases} (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma)) \text{ se } \delta_1(q_1, \sigma) \text{ e } \delta_2(q_2, \sigma) \text{ estão definidos} \\ (\delta_1(q_1, \sigma), q_2) \text{ se } \delta_1(q_1, \sigma) \text{ está definido e } \sigma \notin \Sigma_2 \\ (q_1, \delta_2(q_2, \sigma)) \text{ se } \delta_2(q_2, \sigma) \text{ está definido e } \sigma \notin \Sigma_1 \\ \text{indefinido caso contrário} \end{cases}$$

Do ponto de vista de linguagens, a composição paralela entre dois autômatos pode ser representada como $\mathcal{L}(G_{1||2}) = P_{\Sigma_2}^{-1}(\mathcal{L}(G_1) \cap P_{\Sigma_1}^{-1}(\mathcal{L}(G_2))$.

2.2 Teoria de Controle Supervisório

A Teoria de Controle Supervisório (TCS) (WONHAM, 2014) é uma abordagem que permite, a partir de plantas e especificações de segurança modeladas por autômatos finitos determinísticos, encontrar um agente de controle, capaz de desabilitar certos eventos a fim de garantir um comportamento seguro e não bloqueante, em malha fechada.

Na TCS, o conjunto $\Sigma = \Sigma_c \cup \Sigma_{nc}$ é o conjunto de eventos de um sistema, sendo particionado em Σ_c, o conjunto de eventos controláveis, e Σ_{nc}, o conjunto de eventos não controláveis, e que, por sua vez, não podem ser desabilitados pelo agente de controle.

Sejam G uma planta e E uma especificação, a condição necessária e suficiente para a existência de um supervisor não bloqueante S para G, tal que $\mathcal{L}_m(S/G) = \mathcal{L}(G)||E = K$, é que K seja controlável em relação a $\mathcal{L}(G)$ e Σ_{nc}, ou seja, $\overline{K}\Sigma_{nc} \cap \mathcal{L}(G) \subseteq \overline{K}$. Caso o supervisor não seja controlável, então deve-se calcular a máxima sub linguagem controlável da linguagem desejada, denotada por $Sup\mathcal{C}(K, G)$.

2.3 Ferramenta Computacional

Neste trabalho, todos os algoritmos foram desenvolvidos em uma ferramenta chamada *UltraDES*. O *UltraDES* foi desenvolvido na linguagem `C#`, muito utilizada na área de Tecnologia da Informação (TI) para a indústria, estando baseada no *.NET Framework* como plataforma de execução. Dessa maneira, o *UltraDES* pode ser utilizado em qualquer linguagem que suporte o *.NET Framework* inclusive em Visual C++ e IronPython, isto é, as versões *.NET* das linguagens C++ e Python (ALVES; CIPRIANO; PENA, 2015).

2.4 Simulação de Sistemas a Eventos Discretos

Neste trabalho, a avaliação de desempenho de sequências de eventos é feita por meio de simulação de sistemas a eventos discretos. A simulação em computador é similar a um experimento em laboratório, onde os as plantas são substituídas por *softwares* e a aleatoriedade é substituída por geradores de números aleatórios (CASSANDRAS; LAFORTUNE, 2008).

A utilização de simulações se justifica pela praticidade e repetibilidade dos simuladores frente aos sistemas reais. Além disso, uma simulação, usualmente, é muito mais rápida que a execução de uma sequência de eventos numa planta real, permitindo que o processo de otimização leve em consideração informações temporais, envolvendo até mesmo a execução de sequências parciais.

A simulação clássica de Sistemas a Eventos Discretos, conforme mostrada na Figura 2, é composta dos seguintes componentes:

1. *Estado:* uma lista onde as variáveis de estado são armazenadas.

2. *Tempo:* uma variável onde o tempo de simulação é armazenado.

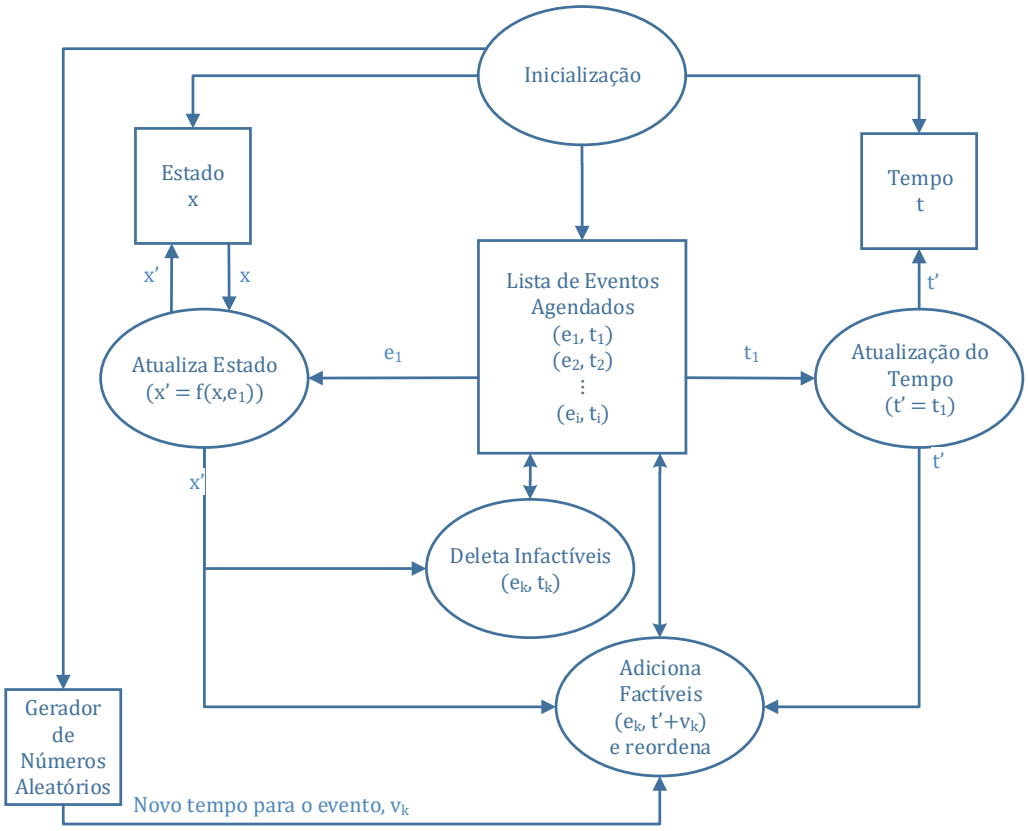


Figura 2 – Diagrama de um Simulador em Sistemas a Eventos Discretos

3. *Lista de Eventos Agendados:* uma lista onde todos os eventos agendados são armazenados, associados ao tempo em que vão ocorrer. É sempre mantida ordenada, de forma que os eventos acontecem na ordem em que estão na lista.

4. *Rotina de Inicialização:* uma rotina que inicializa as estruturas de dados no inicio da execução da simulação.

5. *Rotina de Atualização de Tempo:* um rotina que identifica o próximo evento a ocorrer e atualiza o tempo de simulação dada a ocorrência desse evento.

6. *Rotina de Atualização do Estado:* uma rotina que atualiza o estado do sistema baseado no próximo evento que vai ocorrer.

7. *Gerador de Números Aleatórios:* dada uma distribuição de probabilidade, gera os tempos de ocorrência para os eventos.

8. *Rotina de Remoção de Infactibilidades:* Como nem todo evento está habilitado em todo estado, esta rotina remove da Lista de Eventos Agendados eventos não factíveis, ou seja, que não estão habilitados no estado atual, até que o primeiro evento da lista seja factível.

9. *Programa Principal:* realiza a coordenação dos componentes. Ele chama, repetidamente, as rotinas de atualização temporal, atualização de estado e atualiza a Lista de Eventos Agendados. Além disso, termina a simulação baseado em um critério definido pelo usuário.

2.5 Heurística de Poda

Uma das formas mais efetivas de resolver problemas grandes é por meio da poda (*pruning*) da árvore de busca (EDELKAMP; SCHROEDL, 2012). As técnicas de poda são utilizadas quando certos ramos não levam ao estado objetivo ou levam a soluções inferiores. Usualmente, a poda reduz o *branching factor*, economizando tanto tempo computacional quanto espaço.

Dado que um supervisor obtido pela Teoria de Controle Supervisório é controlável e não bloqueante, a utilização de heurísticas de poda é interessante pelo fato de supervisor já evitar *dead locks* e *live locks*. Além disso, como o supervisor é sintetizado utilizando apenas informações lógicas, certas transições são temporalmente infactíveis. Neste caso, a poda diminui o espaço de busca, mas não interfere na otimalidade da solução, visto que os estados não visitados gerariam soluções infactíveis.

Outra técnica de poda utilizada nesse trabalho parte do critério de que, no contexto industrial, esperar que um dispositivo finalize sua operação em vez de ligar outro dispositivo em paralelo não costuma ser uma boa prática. Nesse sentido, a heurística leva o algoritmo de busca a não seguir transições que desligam dispositivos, a menos que não existam mais dispositivos a serem ligados. Essa poda, diferente da anterior, pode interferir na otimalidade da solução, visto que sequências factíveis não são testadas, porém, em geral, obtém soluções satisfatórias.

3 O Máximo Paralelismo

O problema de escalonamento, com o objetivo de minimizar o tempo de produção, é não polinomial, tornando sua solução exata difícil de ser obtida computacionalmente. Uma maneira, sub ótima, de minimizar o tempo de produção sem levar em consideração o tempo é aumentar o número de máquinas operando em cada instante de tempo. Neste caso, maximizando o paralelismo, espera-se minimizar o tempo de produção. É importante notar que não há garantia de otimalidade quanto ao tempo, visto que o paralelismo não utiliza informações sobre o tempo de produção de cada máquina.

Nas próximas seções serão apresentados dois métodos de escalonamento baseados no critério do máximo paralelismo. O primeiro é um método puramente lógico, que utiliza apenas o paralelismo e o supervisor para encontrar uma sequência de execução. O segundo método utiliza, além do paralelismo, informação temporal para guiar o processo, porém, como a adição do tempo tornaria o algoritmo não polinomial, utiliza-se uma heurística para garantir que o método seja executado em tempo polinomial.

3.1 Máximo Paralelismo Lógico

Considerando um autômato $S = (Q, \Sigma, \delta, q_0, Q_m)$, representando o supervisor de um sistema de produção, deseja-se encontrar um sequência $s \in \mathcal{L}_m(S)$ de tamanho $|s| = n$, onde n é um valor predefinido, tal que não exista outra sequência em $\mathcal{L}_m(S)$ de tamanho n que apresente um maior paralelismo de tarefas que L.

No contexto desse trabalho, uma tarefa é um conceito arbitrário, que é associado aos estados de um autômato, ou seja, um estado qualquer apresenta zero ou mais tarefas ativas e a mudança no número de tarefas ativas em um autômato se dá sempre pela mudança de estado. Nos exemplos apresentados, uma máquina, quando em funcionamento possui uma tarefa em execução e quando desligada possui zero tarefas em execução.

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

Definição 3.1. Seja $G = (Q, \_\_, \_\_, \_\_, \_\_)$ um autômato. A função de tarefas ativas $f_{ta} : Q \to \mathbb{Z}^*$ é a função que associa, para cada estado $q \in Q$, um número inteiro não negativo de tarefas. —

Exemplo 3.1.1. Neste capítulo, será utilizado como exemplo, uma planta chamada Pequena Fábrica, cujo diagrama está mostrado na Figura 3. A planta é composta de duas máquinas conectadas por um *buffer* unitário, e o objetivo é evitar o *overflow* e *underflow*

do *buffer*. Os autômatos que representam o comportamento dinâmico das máquinas (M_1 e M_2) bem como a especificação estão mostrados na Figura 4. Consideramos que cada máquina ligada representa uma tarefa ativa. Os eventos representados por números ímpares são controláveis e os eventos representados por números pares são não controláveis.

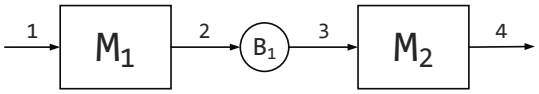
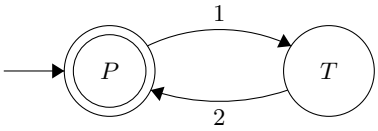
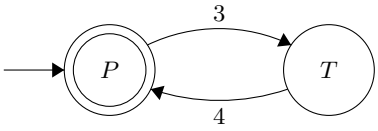


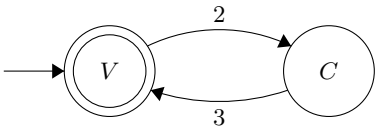
Figura 3 – Diagrama da Pequena Fábrica



(a) Máquina M_1 ($P \leftarrow$ Parada, $T \leftarrow$ Trabalhando)



(b) Máquina M_2 ($P \leftarrow$ Parada, $T \leftarrow$ Trabalhando)



(c) Especificação E ($V \leftarrow$ Vazio, $C \leftarrow$ Cheio)

Figura 4 – Autômatos para as máquinas M_1 e M_2 e para a especificação E.

No caso da Pequena Fábrica, $f_{ta}(T) = 1$ e $f_{ta}(P) = 0$. Isso indica que cada máquina trabalhando contribui com uma tarefa ativa.

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

Definição 3.2. Sejam os autômatos $G_1 = (Q_1, \_\_, \_\_, \_\_, \_\_)$ e $G_2 = (Q_2, \_\_, \_\_, \_\_, \_\_)$ e as funções de tarefas ativas associadas f_{ta_1} e f_{ta_2}. A função de tarefas ativas do autômato $G_{1||2} = (Q_1 \times Q_2, \_\_, \_\_, \_\_, \_\_)$ é definida como:

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

onde $q_1 \in Q_1$ e $q_2 \in Q_2$. —

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

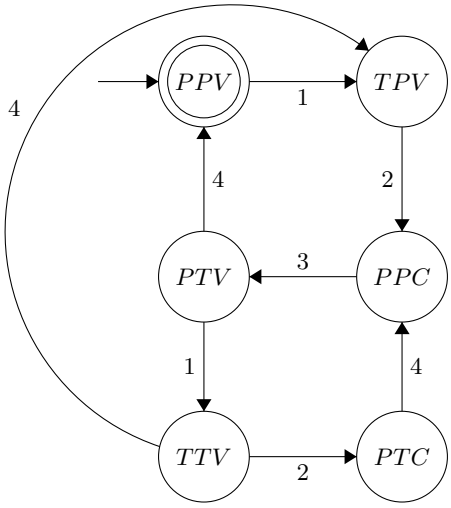


Figura 5 – Supervisor da Pequena Fábrica

Estado (q)	Número de Tarefas Ativas ($f_{ta}(q)$)
PPV	$0+0+0=0$
TPV	$1+0+0=1$
PTV	$0+1+0=1$
PPC	$0+0+0=0$
TTV	$1+1+0=2$
PTC	$0+1+0=1$

Tabela 1 – Número de tarefas ativas em cada estado do supervisor da Pequena Fábrica

Exemplo 3.1.2. Na Figura 5, está mostrado o supervisor da Pequena Fábrica. Como pode ser observado, cada estado é gerado por estados das plantas e da especificação, ou seja $f_{ta}(TTV) = 2$ porque $f_{ta}(T) = 1$ e $f_{ta}(V) = 0$. O número de tarefas ativas dos estados do supervisor da Pequena Fábrica estão mostrados na Tabela 1.

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

Definição 3.3. Seja um autômato $A = (Q, \Sigma, \delta, \_\_, \_\_)$, a função acumulativa de tarefas ativas $F_{ta} : Q \times \Sigma^* \to \mathbb{Z}^*$ é definida como:

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

onde $\sigma s \in \Sigma^*$. —

Sejam duas cadeias $s_1, s_2 \in \mathcal{L}_m(A)$, tais que $|s_1| = |s_2|$. A cadeia que possui o maior F_{ta} apresenta maior paralelismo.

Exemplo 3.1.3. Suponha duas cadeias para a produção de dois produtos na Pequena Fábrica, $s_1 = (1, 2, 3, 4, 1, 2, 3, 4)$ e $s_2 = (1, 2, 3, 1, 4, 2, 3, 4)$, temos que $F_{ta}(PPV, s_1) = 4$,

porém $F_{ta}(PPV, s2) = 6$. A sequência s_2 é mais paralela porque liga M_1 enquanto M_2 ainda está funcionando ao produzir a segunda peça.

3.1.1 Definição do Problema

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

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

$$s^* = \underset{s \in L}{\operatorname{argmax}} \; F_{ta}(q_0, s)$$

Onde s^* é uma das sequências que maximizam número de tarefas executando paralelamente no sistema, e q_0 é o estado inicial da otimização.

Neste trabalho, a produção contínua ótima de X produtos é chamada de produção de um lote de X produtos, ou seja, para se produzir kX produtos pode-se produzir k lotes de X produtos ou um lote de kX produtos. Note que a segunda opção, usualmente, apresenta um melhor desempenho, visto que não se faz necessário terminar completamente a produção de um lote para continuar a produzir outro lote até obter k lotes.

Considera-se que uma sequência $s \in L$ tal que $|s| = kp$, onde p é o número de eventos necessários para produzir um produto, produz k produtos. Tal restrição impede que as plantas apresentem algumas característica, como quebra ou retrabalho, por exemplo. Todos os algoritmos apresentados ainda podem ser utilizados quando as plantas apresentam tais características, porém faz-se necessário definir uma "receita" limitando o número de vezes que cada evento pode ocorrer.

Exemplo 3.1.4. Para o exemplo da Pequena Fábrica, a produção de uma peça equivale a uma sequência de 4 eventos que leve ao estado marcado. Dessa mesma forma, a produção de k produtos equivale a encontrar uma sequência com $4k$ eventos que leve ao estado marcado.

3.1.2 Manipulação do Supervisor e Algoritmo

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

Para tornar o autômato do supervisor em um grafo acíclico com profundidade n, faz-se necessário compor o supervisor com um autômato, aqui chamado de autômato de desenrolamento G_d, como mostrado na Figura 6, onde $\mathcal{L}_m(G_d) = \{s \in \Sigma^* : |s| = n\}$, Σ é o conjunto de eventos de S e n é o número de eventos necessários para a produção de um lote.

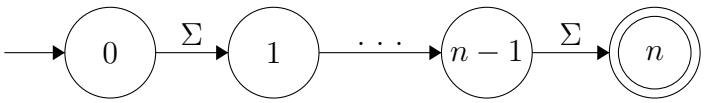


Figura 6 – Forma genérica de um autômato de desenrolamento com um conjunto de eventos Σ e profundidade n

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

Exemplo 3.1.5. Compondo o supervisor da Pequena Fábrica com uma autômato de desenrolamento, para uma profundidade de 8 eventos, obtemos o grafo acíclico mostrado na Figura 7.

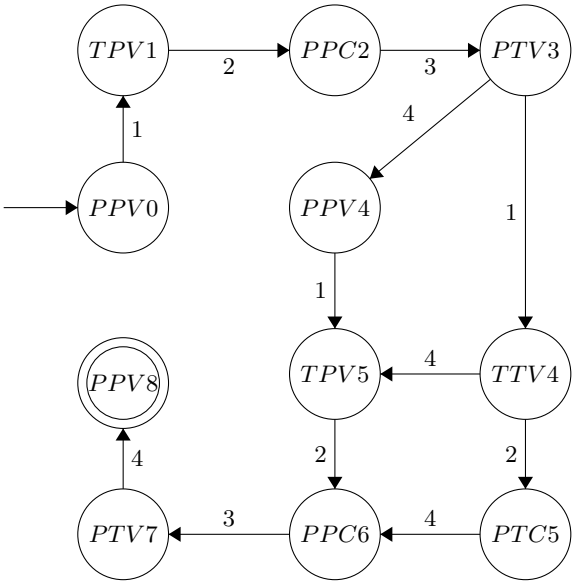


Figura 7 – Grafo acíclico obtido a partir do supervisor da Pequena Fábrica para uma busca com profundidade de 8 eventos.

As informações necessárias para a execução do algoritmo são o conjunto de estados do supervisor (Q), a função de transição de estados (δ), a função de eventos ativos (Γ), o estado inicial (q_0) e a profundidade de busca n. Como resultado, o algoritmo devolve a estrutura *path* que armazena o maior caminho entre o estado inicial $(q_0, 0)$ e qualquer outro estado alcançado na busca. Entre as linhas 1 e 11, a estrutura de dados d, que armazena a maior distância entre um estado e o estado inicial $(q_0, 0)$, e *path*, que armazena o maior caminho entre um estado e o estado inicial $(q_0, 0)$, são inicializados. Computacionalmente

Algoritmo 1: Calcula o caminho que maximiza o paralelismo no grafo acíclico de um supervisor

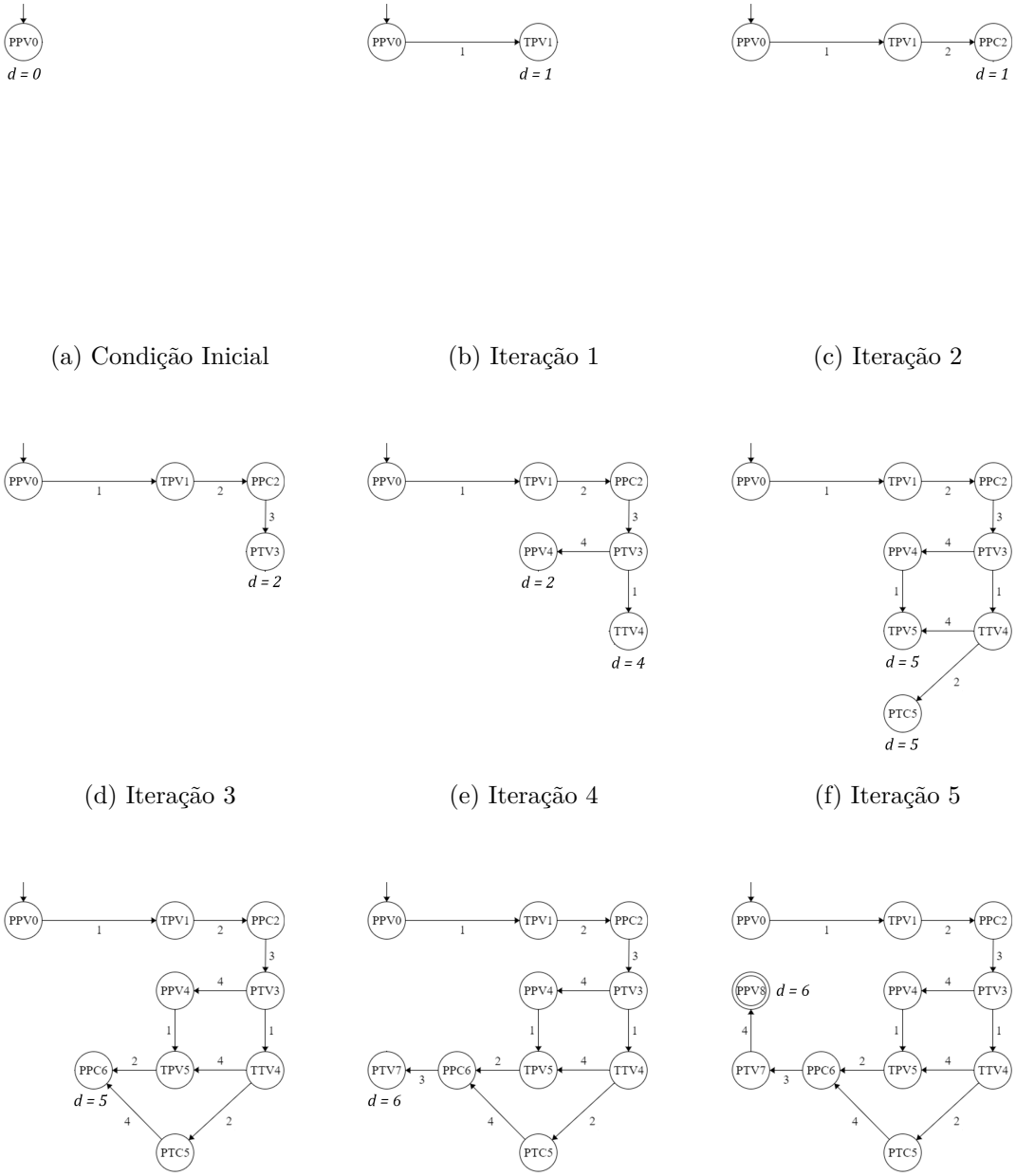
```
1  foreach state q in Q do
2      for i ← 0 to n do
3          if (q, i) = (q_0, 0) then
4              d[(q, i)] ← 0
5              path[(q, i)] ← ϵ
6          else
7              d[(q, i)] ← −∞
8              path[(q, i)] ← ∅
9          end
10     end
11 end
12
13 F ← (q_0, 0)
14
15 while F is not empty do
16     (q, i) ← F
17     if i = n then continue
18
19     foreach event σ in Γ(q) do
20         v ← δ(q, σ)
21         if F does not contain (v, i + 1) then
22             F ← (v, i + 1)
23         end
24         w ← f_ta(v)
25         if d[(q, i)] + w > d[(v, i + 1)] then
26             d[(v, i + 1)] ← d[(q, i)] + w
27             path[(v, i + 1)] ← path[(q, i)]σ
28         end
29     end
30 end
```

essa inicialização pode ser feita durante a execução do algoritmo principal, visto que muitos dos estados inicializados não são acessíveis.

Na linha 13 é inicializada uma fila chamada F, que armazena os estados alcançados em ordem topológica, ou seja, na ordem em que são alcançados a partir do estado inicial. Entre as linhas 15 e 30, é executado um laço *while* enquanto existirem estados a serem visitados na fila F.

Na linha 16, o primeiro estado na fila é removido e na linha 17 se verifica se aquele estado atingiu a profundidade máxima de busca (n). Caso o estado tenha profundidade igual a n, não existe nenhum estado que pode ser alcançado a partir dele.

Da linha 19 à 28, cada estado alcançável a partir de (q, i) é adicionado à fila e é



(a) Condição Inicial (b) Iteração 1 (c) Iteração 2

(d) Iteração 3 (e) Iteração 4 (f) Iteração 5

(g) Iteração 6 (h) Iteração 7 (i) Iteração 8

Figura 8 – Exemplo de execução do algoritmo de Máximo Paralelismo Lógico

avaliado quanto ao paralelismo. Na linha 24 computa-se o peso da transição, relativo ao número de tarefas ativas no estado de destino.

Entre as linhas 25 e 28, verifica-se se aquela transição aumenta o caminho percorrido até então e, caso positivo, as estruturas d e *path* são atualizadas.

Exemplo 3.1.6. Considerando o problema da Pequena Fábrica, para uma profundidade de 8 eventos, equivalente à produção de 2 produtos. A variável d armazena o tamanho do caminho percorrido, referente à função acumulativa de tarefas ativas. Como pode ser

observado na Figura 8 o algoritmo é iniciado no estado inicial do supervisor PPV com profundidade 0, a cada iteração são explorados os estados alcançados pela ocorrência de um evento a partir dos estados visitados na iteração anterior. Na iteração 5, os estados $TTV4$ e $PPV4$ convergem no estado $TPV5$ e, nesse caso, o caminho que é mantido é aquele que maximiza o paralelismo, ou seja, o caminho que passa por $TTV4$. Essa mesma convergência ocorre novamente na iteração 6, porém, nesse caso, o estado $PPC6$ é alcançado com o mesmo valor de paralelismo acumulado, de forma que apenas um deles é escolhido, apesar de ambos serem ótimos até então. O algoritmo é finalizado na iteração 8, onde o estado marcado é alcançado. Caso exista mais de um estado marcado alcançado, com o mesmo paralelismo acumulado, ambas as sequências são ótimas, no sentido de maximizar o paralelismo, porém apenas uma delas é escolhida como solução final. Neste exemplo o algoritmo admite duas soluções:

$$s_1 = (1, 2, 3, 1, 4, 2, 3, 4)$$
$$s_2 = (1, 2, 3, 1, 2, 4, 3, 4)$$

É importante notar, porém, que caso o tempo de produção da máquina 2 seja menor que o tempo de produção da máquina 1, a sequência s_2 não é temporalmente factível, visto que o trecho $(3, 1, 2, 4)$ é impossível.

3.1.2.1 Complexidade do Algoritmo

O algoritmo de busca pelo máximo paralelismo processa todos os vértices e executa um laço percorrendo os vértices adjacentes. O número de estados no grafo acíclico pode ser estimado como $(n+1)|Q|$, onde Q é o conjunto de estados do supervisor, e o número de arestas pode ser estimado como $n|A|$, onde A é o conjunto de todas as transições existentes no supervisor. Dessa forma, a complexidade do algoritmo é dada por $\mathcal{O}(n|Q| + n|A|)$.

3.1.2.2 Corretude do Algoritmo

Sejam $S = (Q, \Sigma, \delta, q_0, Q_m)$ um supervisor monolítico e não bloqueante, $G_d = (\{0, 1, \ldots, n\}, \Sigma, \delta_d, 0, \{n\})$ um autômato de desenrolamento, onde n é a profundidade de busca. Seja $G = Trim(S||G_d)$ um autômato acíclico que implementa a linguagem $\mathcal{L}_m(G) = \{s \in \mathcal{L}_m(S) : |s| = n\}$, possui estado inicial $(q_0, 0)$ e seu conjunto de estados marcados é dado pelo conjunto $\{(q, n) : q \in Q_m\}$. Nesse contexto, os estados (q, k) do autômato G estão ordenados topologicamente quando todo estado (q_a, i) precede todo o estado (q_b, j) se $i < j$.
A função de maior caminho entre dois estados é dada por:

$$\Delta((q_0, 0), (q, k)) = \begin{cases} max\{F_{ta}(q_0, s) | (q_0, 0) \xrightarrow{s} (q, k)\} \text{ se existe caminho entre } (q_0, 0) \text{ e } (q, k) \\ -\infty \text{ caso contrário} \end{cases}$$

A função de peso relativa a uma transição é dada por:

$$w((q_1,k),(q_2,k+1)) = \begin{cases} f_{ta}(q) \text{ se } \exists\sigma \in \Sigma : (q_2,k+1) = \delta((q_1,k),\sigma) \\ -\infty \text{ caso contrário} \end{cases}$$

Seja $s = \sigma_1\sigma_2\sigma_3\ldots\sigma_m$ uma sequência de eventos que gera um caminho $C = (q_0,0) \xrightarrow{\sigma_1} (q_1,1) \xrightarrow{\sigma_2} (q_2,2) \xrightarrow{\cdots} (q_{m-1},m-1) \xrightarrow{\sigma_m} (q_m,m)$. Nesse sentido, os estados $\{(q_0,0),(q_1,1),(q_2,2),\ldots,(q_{m-1},m-1)\}$ são chamados de predecessores de (q_m,m) em C. Quando C é o maior caminho de $(q_0,0)$ até (q_m,m) e todos os predecessores de (q_m,m) estão em uma lista P, dizemos que C é o maior caminho de $(q_0,0)$ até (q_m,m) com respeito a P.

Lema 1. *Seja $(a_1,a_2,a_3,\ldots,a_t)$ a lista de estados do autômato G em ordem topológica. com $t=(n+1)|Q|$ e seja $a_1=(q_0,0)$. Logo após a j-ésima iteração do while, a estrutura $d[a_k]$, com $k>j$ contém o tamanho (peso) do maior caminho entre $(q_0,0)$ e a_k com respeito a P. Além disso, $d[a_{j+1}] = \Delta((q_0,0),a_{j+1})$.* —

Prova. *Base: ($j=1$). Após a primeira iteração do while, $P=\{(q_0,0)\}$ e $d[a_k] = w((q_0,0),a_k)$.*

Hipótese: Suponha que o lema é verdadeiro para $j=l<m-1$.

Indução: Considere $j=l+1$. Na $(l+1)$-ésima iteração, o estado a_{l+1} é incluído em P e a atualização de pesos é realizada para todo estado (q,k) tal que $\exists\sigma\in\Sigma : (q,k) = \delta((a_{l+1}),\sigma)$.

Se a_{l+1} não é alcançável a partir de $(q_0,0)$, então o estado (q,k) também não é alcançável a partir de $(q_0,0)$ e $d[(q,k)]$ continua com o valor de $-\infty$.

Se a_{l+1} é alcançável a partir de $(q_0,0)$, então o estado (q,k) também é alcançável a partir de $(q_0,0)$. Caso $d[(q,k)] < d[a_{l+1}] + w(a_{l+1},(q,k))$, ou seja, se o caminho de $(q_0,0)$ até (q,k) com respeito à lista $P=(a_0,a_1,\ldots,a_l)$ é mais curto que o maior caminho de $(q_0,0)$ até (q,k), com a_{l+1} como predecessor imediato, $d[(q,k)]$ é atualizado como $d[(q,k)] = d[a_{l+1}] + w(a_{l+1},(q,k))$. Pela hipótese, esse novo valor de $d[(q,k)]$ é o tamanho do maior caminho de $(q_0,0)$ até (q,k) com respeito à lista $P=(a_0,a_1,\ldots,a_{l+1})$. Além disso, se $(q,k)=a_{l+2}$, temos que $d[(q,k)] = \Delta((q_0,0),(q,k))$, pois, visto que os estados estão em ordem topológica, não existe outro estado em $(a_{l+3},a_{l+4},\ldots,a_t)$ que possa alcançar a_{l+2}.

Dessa forma, é possível observar que o lema é verdadeiro para a $(l+1)$-ésima iteração, completando a indução. —

Teorema 1. *O algoritmo do Máximo Paralelismo, executado em um autômato acíclico a partir do estado inicial* $(q_0, 0)$*, termina com* $d[(q,k)] = \Delta((q_0,0),(q,k))$ *para todo* $(q,k) \in Q \times \{k : 0 \leq k \leq n\}$. —

Prova. *Pelo Lema 1, após a* j*-ésima iteração,* $d[a_{j+1}] = \Delta((q_0,0), a_{j+1})$ *é finalizado. Após* $t-1$ *iterações,* $d[(q,k)] = \Delta((q_0,0),(q,k))$ *para todo* $(q,k) \in Q \times \{k : 0 \leq k \leq n\}$. —

3.1.3 Vantagens e Desvantagens

Como vantagem, pode-se citar que o algoritmo de Máximo Paralelismo Lógico é um algoritmo exato, quanto à maximização do paralelismo e apresenta complexidade polinomial. Por outro lado, sua principal desvantagem é ser puramente lógico e não levar em consideração nenhuma informação quanto ao tempo de execução de cada tarefa, gerando sequências que não são, necessariamente, temporalmente factíveis, ou seja, os eventos não controláveis não ocorrem na ordem preestabelecida pelo algoritmo.

3.1.4 Problemas na Execução Temporal e Recálculo

O algoritmo do máximo paralelismo lógico não utiliza nenhuma informação temporal do sistema, assinalando uma sequência de execução tanto para eventos controláveis quanto para eventos não controláveis. Como não há controle sobre a execução de eventos não controláveis, certas sequências obtidas pelo algoritmo do máximo paralelismo não serão obedecidas na execução temporal.

Os problemas de execução ocorrem quando um evento não controlável, previsto como o próximo evento a ser executado é precedido por um outro evento não controlável. Como ambos os eventos são não controláveis, sua ocorrência não pode ser evitada, levando o sistema a um estado não previsto na sequência ótima. Para solucionar o problema, executa-se o algoritmo, considerando como estado inicial o estado atual do sistema, para a profundidade restante, ou seja, a profundidade inicial menos o número de eventos já executados.

Devido à baixa complexidade do algoritmo, uma alternativa viável é recalcular o caminho maximamente paralelo *online* sempre que a sequência original for desobedecida. Dessa forma o algoritmo continua não necessitando de informações temporais e, mesmo assim, pode ser executado em um contexto real.

A grande vantagem de não se utilizar informações relativas ao tempo, no algoritmo, é permitir que a técnica seja utilizada em situações onde o tempo não é determinístico, o que é uma realidade em diversos setores industriais.

3.1.4.1 Complexidade do Algoritmo com Recálculo

Considerando que o algoritmo tem complexidade linear e supondo que o número de vezes em que o recálculo deve ser feito é proporcional ao tamanho da sequência (αn, com $1 \geq \alpha \geq 0$), podemos considerar que a complexidade do algoritmo, quando utilizado o recálculo, passa a ser $\mathcal{O}(n^2|Q| + n^2|A|)$.

3.1.5 Execução da Sequência Maximamente Paralela Projetada

Outra possível abordagem para tratar a infactibilidade temporal das sequências obtidas pelo algoritmo do máximo paralelismo seria a projeção natural da sequência ótima (s^*) no conjunto de eventos controláveis, obtendo uma sequência $s_c = P_{\Sigma_c}(s^*)$. Essa abordagem leva em consideração que os eventos não controláveis são respostas do sistema e sua ordem não pode ser estabelecida. Nesse sentido, apenas a ordem dos eventos controláveis é definida e os eventos não controláveis ocorrem quando necessário (VILELA; PENA, 2016).

Exemplo 3.1.7. No caso da sequência $s = (1, 2, 3, 4, 1, 2, 3, 4)$, a sequência projetada seria $s_c = (1, 3, 1, 3)$. A utilização, em um sistema real, da sequência s_c, pode acarretar na execução de uma sequência que não necessariamente é a sequência s.

3.2 Máximo Paralelismo com Restrições Temporais

Como uma das desvantagens do máximo paralelismo lógico é a ausência da informação temporal, aqui se define uma pequena modificação da técnica original, aplicando restrições temporais, de forma que, se os tempos de processamento são determinísticos, a sequência obtida é coerente no contexto temporal em que foi concebida.

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

Definição 3.4. Seja $S = (\_\_, \Sigma, \_\_, \_\_, \_\_)$ um supervisor. A função temporal $f'_T : \Sigma^* \times \Sigma \to \mathbb{R}^*$ 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, avaliando a diferença entre o tempo de execução do último evento da sequência s e o tempo de s. —

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 3.5. A função temporal expandida $f_T : \Sigma^* \to \mathbb{R}^*$ 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 maximize o paralelismo, mas também apresente um tempo de execução menor que infinito. O principal problema dessa proposta é que esse objetivo é tão difícil quanto o de minimização de *makespan*, de forma que a aplicação do paralelismo não traz ganhos significativos em desempenho.

Dessa maneira, o máximo paralelismo com restrições temporais é executado de maneira heurística, de forma que não se garante a otimalidade da solução.

Exemplo 3.2.1. Na Figura 9, podemos observar o grafo acíclico da Pequena Fábrica, em conjunto com as agendas de eventos de cada estado. É interessante notar que o estado $TPV5$ está representado com duas agendas de eventos diferentes, porque o caminho que leva a ele interfere no tempo em que os eventos vão ocorrer. Um algoritmo exato trata um mesmo estado com duas agendas como estados diferentes, mas isso leva a um crescimento do espaço de busca. O algoritmo de Máximo Paralelismo com Restrições Temporais desconsidera essa duplicidade e mantém apenas aquele que maximiza o paralelismo. Nesse caso, o estado $TPV5$ alcançado pela sequência $s_1 = (1, 2, 3, 4, 1)$ apresenta uma $F_{ta}(PPV0, s_1) = 3$ enquanto o estado $TPV5$ alcançado pela sequência $s_2 = (1, 2, 3, 1, 4)$ apresenta $F_{ta}(PPV0, s_2) = 5$, ou seja, o segundo caso, e sua respectiva agenda, são mantidos e o primeiro caso é ignorado. Nesta situação, essa simplificação não tem efeitos negativos porque ambos convergem para o mesmo estado, porém em situações onde isso não ocorre, essa simplificação pode trazer prejuízos à otimalidade.

Outro ponto importante a ser observado é que as restrições temporais podem gerar uma diminuição do espaço de busca, visto que eliminam-se as sequências que não são temporalmente factíveis. No estado $TTV4$, por exemplo, a transição para o estado $PTC5$ não precisa ser analisada, visto que, devido ao determinismo dos tempos do processo, o evento 4 vai ocorrer antes do evento 2, diminuindo o *branching factor* do algoritmo de busca.

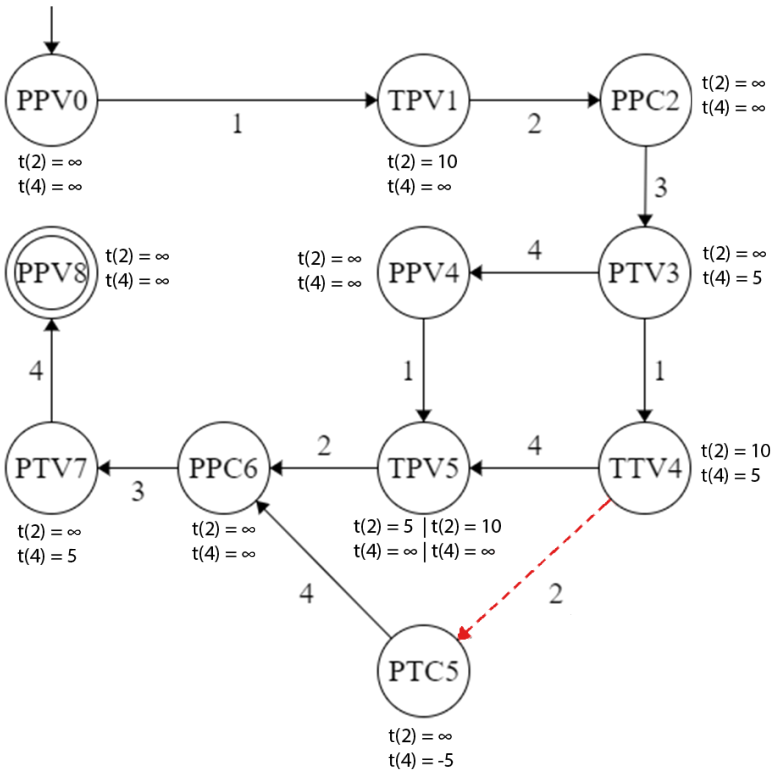


Figura 9 – Grafo acíclico do supervisor da Pequena Fábrica com agendas de eventos associadas aos estados, para uma profundidade de 8 eventos.

3.2.1 Definição do Problema

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

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

$$s^* = \underset{s \in L}{\operatorname{argmax}} \; F_{ta}(q_0, s)$$

Onde s^* é uma das sequências que maximizam o número de tarefas executando paralelamente no sistema e q_0 é o estado inicial do supervisor.

O problema a ser resolvido é similar ao problema do máximo paralelismo, porém a informação relativa à função temporal impede que o problema possa ser tratado, de maneira exata, como uma problema de maior caminho em um grafo acíclico, porque as restrições temporais fazem com que o caminho utilizado para alcançar certo estado influencie no melhor caminho entre aquele estado e o objetivo.

Neste trabalho, propõe-se um algoritmo de melhor caminho, desconsiderando que um mesmo estado, alcançado de maneiras diferentes, possui diferentes futuros, tomando um passo guloso pegando apenas o melhor caminho até aquele estado. Dessa forma, o algoritmo de máximo paralelismo com restrições temporais passa a ser um algoritmo heurístico, mas que garante a execução temporal da sequência obtida, mas não garante sua otimalidade quanto ao critério de paralelismo.

3.2.2 Manipulação do Supervisor e Algoritmo

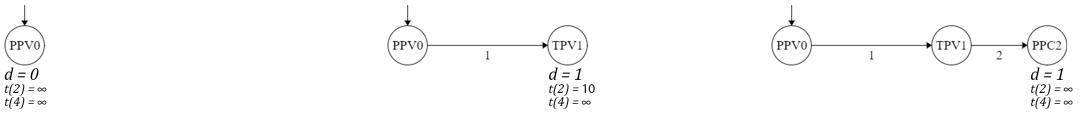
O problema de otimização será resolvido como uma problema de maior caminho em um grafo acíclico, e como já definido anteriormente, se faz necessário compor o autômato do supervisor com um autômato de desenrolamento, a fim de transformá-lo em um grafo acíclico. Faz-se necessário também a definição do número de eventos para a produção de um lote de produtos n e da definição da função temporal do supervisor, estabelecendo o tempo de operação de cada máquina.

Assim como no algoritmo do máximo paralelismo lógico, as informações necessárias para a execução do algoritmo são o conjunto de estados do supervisor (Q), a função de transição de estados (δ), a função de eventos ativos (Γ), o estado inicial (q_0) e a profundidade de busca n. Como resultado, o algoritmo devolve a estrutura *path* que armazena o maior caminho entre o estado inicial $(q_0, 0)$ e qualquer outro estado alcançado na busca.

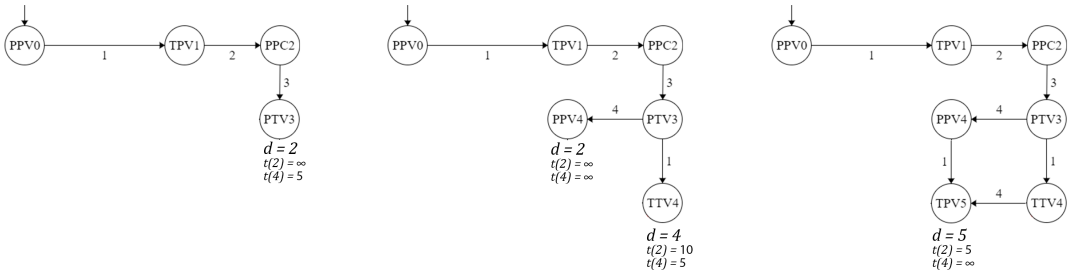
O algoritmo realiza a inicialização das variáveis, entre as linhas 1 e 13, para todos os estados, exceto o estado inicial, o paralelismo é iniciado com $-\infty$ ($d[(q,i)] \leftarrow -\infty$), o caminho é inicializado com vazio ($path[(q,i)] \leftarrow \emptyset$) e o tempo é inicializado com ∞ ($time[(q,i)] \leftarrow \infty$). A fila F é inicializada com o estado inicial.

Enquanto existirem elementos em F, é retirado o primeiro estado da fila (linha 18) e cada transição partindo desse estado é analisada. Primeiramente, verifica-se se seguir essa transição é temporalmente factível (linhas 23 e 24) e em seguida verifica-se se seguir essa transição gera uma melhora, ou seja, se aumenta o paralelismo (linha 32).

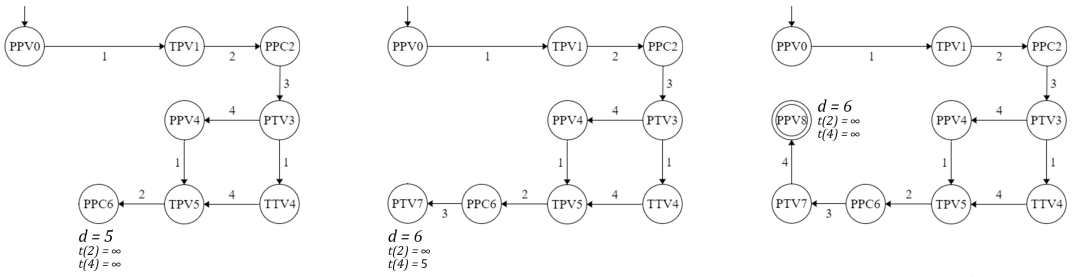
Exemplo 3.2.2. Considerando o problema da Pequena Fábrica, para uma profundidade de 8 eventos, equivalente à produção de 2 produtos. Neste exemplo, d é uma variável que armazena o paralelismo acumulado até o estado e $t(e)$ representa o tempo até que o evento e ocorra. Considerou-se que os eventos 1 e 3 ocorrem instantaneamente, ou seja, $t(1) = t(3) = 0$ em todos os estados. A execução do algoritmo é similar àquela desenvolvida no Exemplo 3.1.6, porém, nesse caso o estado $PTC5$ não é visitado, porque uma restrição temporal, no estado $TTV4$, garante que o evento 4 ocorrerá antes do evento 2, reduzindo o espaço de busca. Esse algoritmo, nestas condições, admite apenas uma solução:



(a) Condição Inicial (b) Iteração 1 (c) Iteração 2



(d) Iteração 3 (e) Iteração 4 (f) Iteração 5



(g) Iteração 6 (h) Iteração 7 (i) Iteração 8

Figura 10 – Exemplo de execução do algoritmo de Máximo Paralelismo com Restrições Temporais

Algoritmo 2: Calcula o caminho que maximiza o paralelismo no grafo acíclico de um supervisor com restrições temporais

```
1  foreach state q in Q do
2      for i ← 0 to n do
3          if (q, i) = (q_0, 0) then
4              d[(q, i)] ← 0
5              path[(q, i)] ← ϵ
6              time[(q, i)] ← 0
7          else
8              d[(q, i)] ← −∞
9              path[(q, i)] ← ∅
10             time[(q, i)] ← ∞
11         end
12     end
13 end
14
15 F ← (q_0, 0)
16
17 while F is not empty do
18     (q, i) ← F
19     if i = n then continue
20
21     foreach event σ in Γ(q) do
22         v ← δ(q, σ)
23         t ← f_T(path[(q, i)]σ)
24         if t = ∞ then continue
25
26         if F does not contain (v, i + 1) then
27             F ← (v, i + 1)
28         end
29         w ← f_ta(v)
30         d_q ← d[(q, i)]
31         d_v ← d[(v, i + 1)]
32         if d_q + w > d_v or (d_q + w = d_v and t < time[(v, i + 1)]) then
33             d[(v, i + 1)] ← d[(q, i)] + w
34             path[(v, i + 1)] ← path[(q, i)]σ
35             time[(v, i + 1)] ← t
36         end
37     end
38 end
```

$$s_1 = (1, 2, 3, 1, 4, 2, 3, 4)$$

3.2.2.1 Complexidade do Algoritmo

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

3.2.2.2 Simulação e Estimação do Tempo

Os eventos são conceitos primitivos e de difícil definição formal, porém, de uma maneira geral, eventos ocorrem instantaneamente e geram uma mudança de estado. Nesse sentido, o tempo associado a uma mudança de estado, nesse trabalho, é sempre nulo. Aqui, o tempo medido é o tempo de permanência nos estados, que pode ser representado pela diferença entre a ocorrência de dois eventos relacionados entre si. Por exemplo, o caminho c abaixo:

$$c : q_0 \xrightarrow{\sigma_1} q_1 \xrightarrow{\sigma_n} q_2$$

onde o evento σ_1 ocorre no instante t_1 e σ_2 ocorre no instante t_2 é executada em $\Delta t = t_2 - t_1$ unidades de tempo, caso a ocorrência dos eventos σ_1 e σ_2 estejam relacionadas. Além disso, podemos afirmar que o evento σ_2 ocorre Δt unidades de tempo após a ocorrência do evento σ_1. Dessa forma, Δt é o tempo de permanência no estado q_1.

Partindo do princípio de que, usualmente, eventos controláveis iniciam tarefas e eventos não controláveis finalizam tarefas, podemos dizer que certos eventos não controláveis são respostas da planta à execução de eventos controláveis.

Nesse sentido, podemos definir uma função que devolve o intervalo de tempo entre a ocorrência de dois eventos $\sigma_1, \sigma_2 \in \Sigma$:

Definição 3.6. Seja $S = (\_\_, \Sigma, \_\_, \_\_, \_\_)$ um supervisor, a função intervalo de tempo entre dois eventos, $T : \Sigma \times \Sigma \to \mathbb{R} \cup \{\infty\}$, é definida como:

$$T(\sigma_1, \sigma_2) = \begin{cases} \infty & \text{se } \sigma_1, \sigma_2 \text{ não são relacionados} \\ (t_{\sigma_2} - t_{\sigma_1}) & \text{caso contrário} \end{cases}$$

—

Seja $S = (Q, \Sigma, \delta, q_0, Q_m)$ um supervisor, seja $s = \sigma_1, \sigma_2, \ldots, \sigma_n$, tal que $s \in \mathcal{L}(S)$ uma sequência. Para um estado $q \in Q$, a função de eventos ativos, $\Gamma : Q \to 2^{\Sigma}$, retorna o conjunto de eventos que podem ocorrer nesse determinado estado.

Faz-se necessário definir uma estrutura, chamada agendador de eventos (AE) que, dado um evento, devolve o tempo necessário até que o evento ocorra. Neste trabalho o agendador de eventos é inicializado com tempo zero para os eventos controláveis (Σ_c) e tempo infinito para os eventos não controláveis.

Partindo do estado inicial e inicializando no tempo zero ($t = 0$), a simulação segue de seguinte forma:

1. Remova o primeiro evento σ_1 de s

2. Verifica em AE se não existe outro evento $\sigma \in \Gamma(q) \cap \Sigma_{nc}$ tal que $AE[\sigma] \leq AE[\sigma_1]$. Caso exista esse evento σ, $t = \infty$, caso não exista, $t = t + AE[\sigma_1]$.

3. Atualiza os intervalos de tempo em AE, para todo evento $\sigma \in \Sigma$ como $AE[\sigma] = max(0, AE[\sigma] - AE[\sigma_1])$.

4. A execução do evento σ_1 gera uma mudança de estado, de modo que $q = \delta(q, e)$. Ela também pode ativar o tempo de algum evento relacionado à σ_1, de forma que para todo evento $\sigma \in \Sigma : T(\sigma_1, \sigma) \neq \infty$, faz-se $AE[\sigma] = T(\sigma_1, \sigma)$.

5. Volta ao passo 1 enquanto existirem eventos em c.

Ao final da simulação, a variável t contém o tempo total de execução da sequência especificada em s. Caso $t = \infty$, podemos dizer que a sequência não é factível temporalmente, dada função T utilizada.

3.2.3 Vantagens e Desvantagens

A principal vantagem do algoritmo de Máximo Paralelismo com Restrições Temporais é gerar sequências que são temporalmente factíveis, dados intervalos de tempo preestabelecidos, mantendo a complexidade polinomial. A principal desvantagem, por sua vez, é que o algoritmo não é exato, visto que estados iguais com agendas de eventos diferentes são tratados como o mesmo estado.

4 Busca pelo Menor Tempo

Apesar do critério do máximo paralelismo ser uma abordagem satisfatória, muitas vezes é interessante realizar uma avaliação puramente temporal. Nas próximas seções estão descritos dois algoritmos de escalonamento temporal. O primeiro é um algoritmo exato que busca no grafo do supervisor o caminho que minimiza o tempo de produção. A segunda seção apresenta um algoritmo que implementa uma heurística *branch and bound*, evitando a espera por certos eventos quando outras ações podem ser tomadas.

4.1 Menor Tempo Exato

O menor caminho temporal exato é um problema difícil de ser resolvido computacionalmente, o princípio da busca é o mesmo dos demais casos, a ideia é utilizar um algoritmo de menor caminho, onde o peso de uma transição é dado pelo tempo necessário para executar determinada ação. A grande dificuldade é que, devido ao paralelismo, os pesos das transições a partir de um estado dependem do caminho que levou àquele estado, ou seja, um mesmo estado alcançado de duas maneiras diferentes deve ser tratado como dois estados distintos em um algoritmo de menor tempo.

A principal forma de simplificar esse problema é associar à cada estado uma agenda de eventos, que estabelece o tempo necessário para que um determinado evento ocorra. Desta forma, um mesmo estado com agendas diferentes é considerado como dois estados diferentes pelo algoritmo de menor tempo, por outro lado, estados iguais com agendas iguais são tratados como estados iguais mesmo que sejam alcançados por caminhos diferentes.

As informações necessárias para a execução do algoritmo são o conjunto de estados do supervisor (Q), o conjunto de eventos do supervisor ($\Sigma = \Sigma_c \cup \Sigma_{nc}$), a função de transição de estados (δ), a função de eventos ativos (Γ), o estado inicial (q_0) e a profundidade de busca n, e a agenda de eventos inicial (a_0). Como resultado, o algoritmo devolve a estrutura *path* que armazena o maior caminho entre o estado inicial $(q_0, a_0, 0)$ e qualquer outro estado alcançado na busca, e o tempo total *time*.

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 ($time[(q_0, a_0, 0)] \leftarrow 0$).

O estado inicial é inserido na fila F (linha 4) e, enquanto a fila possui elementos, o primeiro estado é retirado da fila, determina-se qual o menor tempo para a ocorrência de um evento não controlável, armazenado em t_{min} (linha 8). Após isso, cada transição que produza um acréscimo de tempo menor ou igual à t_{min} é avaliada. Esse passo se faz

Algoritmo 3: Calcula o caminho que minimiza o *makespan* no grafo acíclico de um supervisor

```
1  path[(q_0, a_0, 0)] ← ϵ
2  time[(q_0, a_0, 0)] ← 0
3
4  F ← (q_0, a_0, 0)
5
6  while F is not empty do
7  |   (q, a, i) ← F
8  |   t_min ← min_{σ∈Γ(q)∩Σ_nc} a[σ]
9  |   foreach event σ in Γ(q) ∩ {σ_t : σ_t ∈ Σ ∧ a[σ_t] ≤ t_min} do
10 |   |   v ← δ(q, σ)
11 |   |   t ← a[σ]
12 |   |   a_n ← update(a, σ)
13 |   |   if t = ∞ then continue
14 |   |
15 |   |   if F does not contain (v, a_n, i + 1) then
16 |   |   |   F ← (v, a_n, i + 1)
17 |   |   end
18 |   |   t_tot ← time[(q, a, i)] + t
19 |   |   if (∄ time[(v, a_n, i + 1)]) OR (t_tot < time[(v, a_n, i + 1)]) then
20 |   |   |   path[(v, a_n, i + 1)] ← path[(q, a, i)]σ
21 |   |   |   time[(v, a_n, i + 1)] ← t_tot
22 |   |   end
23 |   end
24 end
```

necessário porque um evento não controlável não pode ser impedido de acontecer, ou seja, não se pode realizar uma transição com tempo maior que o evento não controlável mais próximo.

Para cada transição, calcula-se o acréscimo de tempo gerado pela transição, atualiza-se a agenda de eventos e verifica-se se a sequência resultante é temporalmente factível (linhas 10, 11 e 12). Caso o estado de destino ainda não tenha sido avaliado, ele é colocado na fila. 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.

Como o estado é representado como um estado do supervisor, uma agenda de eventos e um índice, o espaço de busca, neste problema é muito maior, principalmente porque um mesmo estado do supervisor a uma mesma profundidade pode gerar diferentes estados no algoritmo pelo fato de ter agendas diferentes.

Exemplo 4.1.1. Para o problema da Pequena Fábrica, a Figura 11 mostra o grafo acíclico do supervisor, porém, o objetivo é minimizar o tempo decorrido de forma que, nesse caso, torna-se importante manter os estados $TPV5$ duplicados, pois a agenda de eventos deles é diferente, e transições acionadas pelo mesmo evento podem gerar acréscimos de tempo diferentes. Por isso, quando se calcula o menor tempo, o estado passa a ser definido pelo seu nome e sua agenda de eventos. Da mesma forma como acontece no algoritmo de Máximo Paralelismo com Restrições Temporais, em alguns caso as restrições relativas ao

tempo reduzem o espaço de busca, porém, em geral, a duplicação de estados devido às agendas de eventos é muito maior.

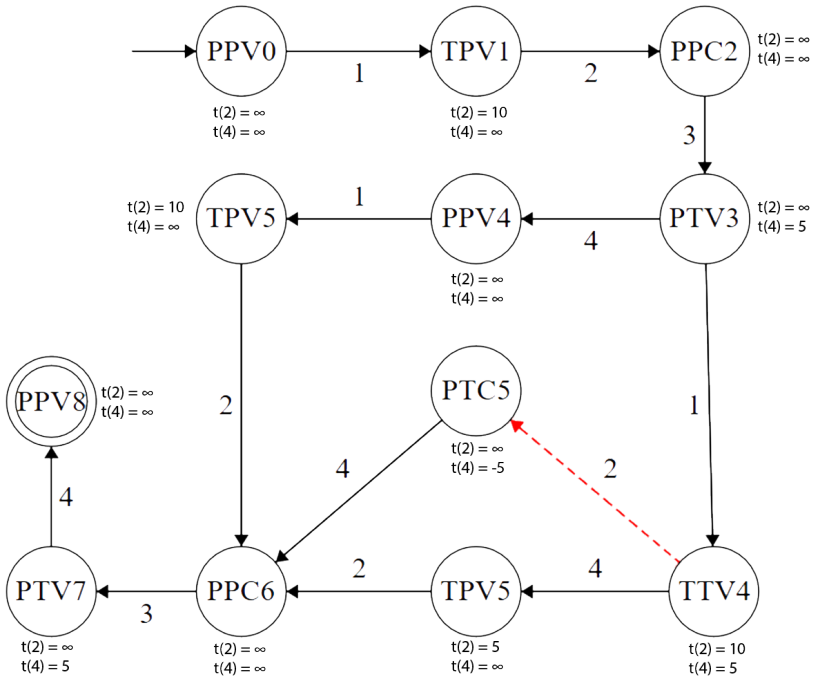
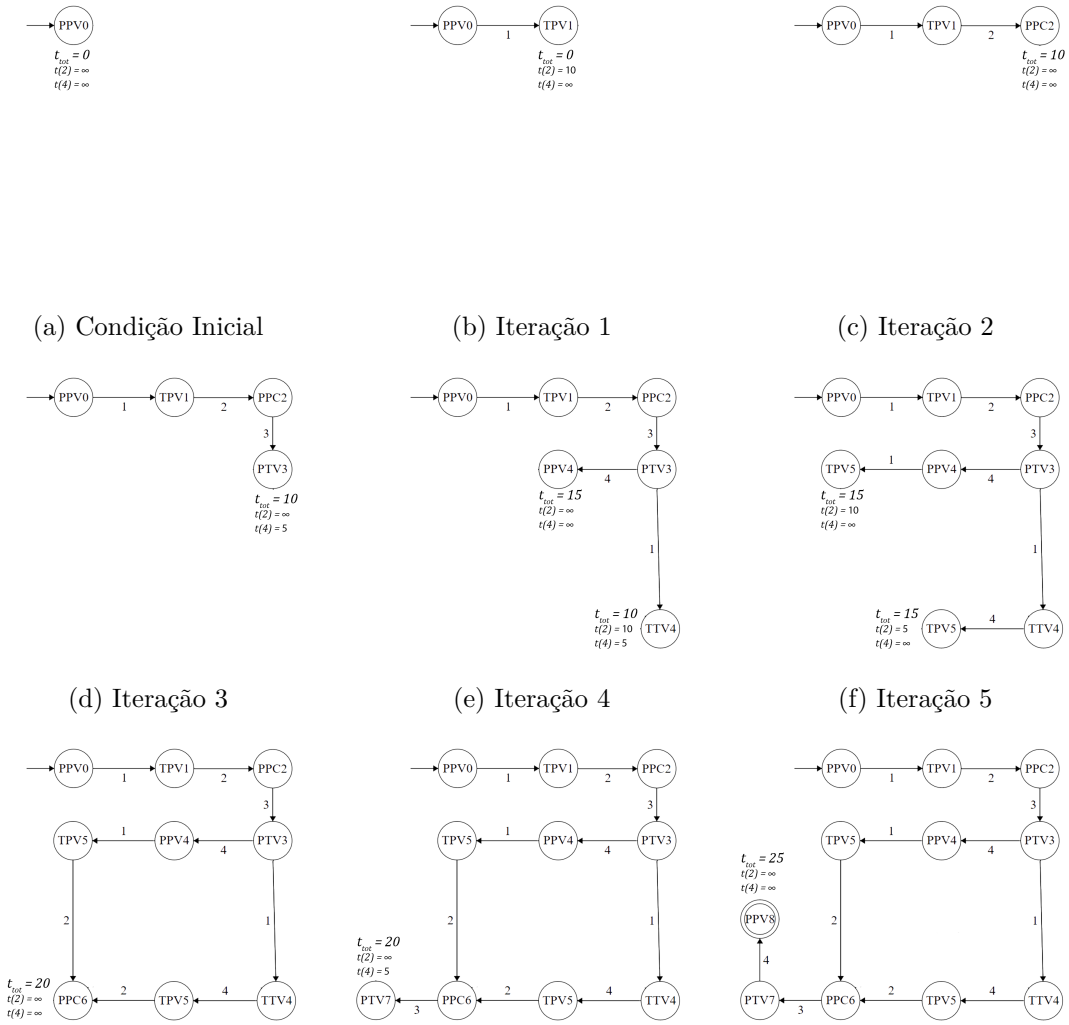


Figura 11 – Grafo acíclico do supervisor da Pequena Fábrica com agendas de eventos associadas aos estados e duplicidade de estados, para uma profundidade de 8 eventos.

Para uma profundidade de 8 eventos, equivalente à produção de 2 produtos foi realizada uma execução do algoritmo ilustrada na Figura 12. Neste exemplo, a variável t_{tot} armazena o tempo decorrido até aquela iteração, para cada caminho, e $t(e)$ representa o tempo até que o evento e ocorra. Novamente, $t(1) = t(3) = 0$; Como pode ser observado, o algoritmo é inicializado no estado inicial do autômato, acompanhado de profundidade zero e, de maneira similar ao que foi realizado no Exemplo 3.1.6 e no Exemplo 3.2.2, os estados são visitados a cada iteração. Um fato importante nesse exemplo é que o estado $TPV5$ é duplicado, porque quando alcançado por caminhos diferentes, apresenta agenda de eventos distinta. Na iteração 6, ambos os estados $TPV5$ alcançam $PPC6$, porém o caminho mantido é aquele que minimiza o tempo total. Após a iteração 8, o algoritmo finaliza, resultando em apenas uma sequência:

$$s_1 = (1, 2, 3, 1, 4, 2, 3, 4)$$

Este é o mesmo resultado alcançado no Exemplo 3.2.2, e, por sua vez, a sequência que minimiza o tempo de produção de duas peças na Pequena Fábrica.



(a) Condição Inicial (b) Iteração 1 (c) Iteração 2

(d) Iteração 3 (e) Iteração 4 (f) Iteração 5

(g) Iteração 6 (h) Iteração 7 (i) Iteração 8

Figura 12 – Exemplo de execução do algoritmo de Menor Tempo Exato

4.1.1 Vantagens e Desvantagens

A vantagem do algoritmo de Menor Tempo Exato é obter a melhor solução para o problema, encontrando a sequência que apresenta menor tempo de produção, porém o algoritmo apresenta complexidade não polinomial, sendo caro computacionalmente, tanto em processamento quanto em memória, não sendo aplicável para a maior parte dos problemas.

4.2 Menor Tempo Heurístico

O menor caminho temporal exato é um algoritmo extremamente custoso computacionalmente, o que inviabiliza sua aplicação para problemas de médio e grande porte, bem como para grandes lotes de produção. Uma maneira simples de simplificar o problema é aplicar uma heurística no estilo *branch and bound*, 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 seria interessante deixar de executar um evento controlável para executar um evento não controlável. 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 possui tempo zero, ou algum evento controlável gera um acréscimo de tempo superior ao tempo para a ocorrência do evento não controlável

As informações necessárias para a execução do algoritmo são as mesmas do algoritmo de menor tempo. O conjunto de estados do supervisor (Q), o conjunto de eventos do supervisor ($\Sigma = \Sigma_c \cup \Sigma_{nc}$), a função de transição de estados (δ), a função de eventos ativos (Γ), o estado inicial (q_0) e a profundidade de busca n, e a agenda de eventos inicial (a_0). Como resultado, o algoritmo devolve a estrutura *path* que armazena o maior caminho entre o estado inicial $(q_0, a_0, 0)$ e qualquer outro estado alcançado na busca, e o tempo total *time*. Além disso, neste algoritmo faz-se necessário que o agendador de eventos restrinja o número de ocorrências dos eventos controláveis, sendo que após o número máximo de ocorrências de um evento controlável σ, temos $a[\sigma] = \infty$.

Algoritmo 4: Calcula o caminho que minimiza o *makespan* no grafo acíclico de um supervisor, utilizando uma heurística

```
1  path[(q, a, 0)] ← ϵ
2  time[(q, a, 0)] ← 0
3
4  F ← (q_0, a, 0)
5
6  while F is not empty do
7      (q, a, i) ← F
8      if Γ(q) ∩ Σ_c ≠ ∅ AND ∀σ ∈ Γ(q) ∩ Σ_c, a[σ] = 0 then
9          E ← Γ(q) ∩ Σ_c
10     else
11         t_min ← min_{σ∈Γ(q)∩Σ_u} a[σ]
12         E ← Γ(q) ∩ {σ_t : σ_t ∈ Σ ∧ a[σ_t] ≤ t_min}
13     end
14     foreach event σ in E do
15         v ← δ(q, σ)
16         t ← a[σ]
17         a_n ← update(a, σ)
18         if t = ∞ then continue
19
20         if F does not contain (v, a_n, i + 1) then
21             F ← (v, a_n, i + 1)
22         end
23         t_tot ← time[(q, a, i)] + t
24         if (∄ time[(v, a_n, i + 1)]) OR (t_tot < time[(v, a_n, i + 1)]) then
25             path[(v, a_n, i + 1)] ← path[(q, a, i)]σ
26             time[(v, a_n, i + 1)] ← t_tot
27         end
28     end
29 end
```

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 ($time[(q_0, a_0, 0)] \leftarrow 0$).

O estado inicial é inserido na fila F (linha 4) e, enquanto a fila possui elementos, o primeiro estado é retirado da fila. A parte heurística do algoritmo está em considerar que eventos controláveis iniciam tarefas e que, no contexto de minimização do tempo, retardar o início de uma tarefa aguardando a ocorrência de um evento não controlável, em geral, não traz benefícios. Dessa forma, o algoritmo verifica se existem transições por eventos controláveis disponíveis e se todas elas não geram acréscimo de tempo (linha 8), ou seja, de o tempo total após a ocorrência do evento é igual ao tempo anterior à ocorrência do evento. Caso essas transições existam, o conjunto de eventos que serão avaliados (E) é formado apenas pelos eventos relacionados a essas transições. Caso não existam transições por eventos controláveis, ou transições controláveis que afetam a contagem de tempo, o procedimento do algoritmo exato é utilizado. Determina-se qual o menor tempo para a ocorrência de um evento não controlável, armazenado em t_{min} (linha 11) e, após isso, cada transição que produza um acréscimo de tempo menor ou igual a t_{min} é avaliada.

Para cada transição por um evento no conjunto E, calcula-se o acréscimo de tempo gerado pela transição, atualiza-se a agenda de eventos e verifica-se se a sequência resultante é temporalmente factível (linhas 15, 16 e 17). Caso o estado de destino ainda não tenha sido avaliado, ele é colocado na fila. 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.

Assim como no algoritmo de menor tempo exato, o estado é representado como um estado do supervisor, uma agenda de eventos e um índice, porém a heurística reduz de maneira significativa o *branching factor*, de forma que o algoritmo seja rápido, mesmo com um algoritmo não polinomial.

Uma restrição quanto à ocorrência de eventos controláveis, apesar de poder ser utilizada em todos os algoritmos descritos anteriormente, se faz necessária neste caso para garantir que os eventos não controláveis sejam executados em algum momento, desabilitando forçadamente certos eventos controláveis, garantindo assim que a heurística leve ao encontro de um estado marcado.

Exemplo 4.2.1. No exemplo da Pequena Fábrica, a aplicação da heurística utilizada no algoritmo de Menor Caminho Heurístico pode ser visto na Figura 13. Neste caso, além da transição entre o estado $TTV4$ e $PTC5$ não precisar ser analisada, tabém não se analisa a transição entre o estado $PTV3$ e $PPV4$, resultando em uma diminuição significativa no espaço de busca, tornando a solução, neste caso, trivial, visto que resta apenas um caminho que leva ao estado marcado $PPV8$. Essa heurística se justifica porque partimos do pressuposto que esperar 5 *u.t.* para que o evento 4 aconteça não é uma boa estratégia,

visto que seria possível ligar M_1 e manter ambas funcionando em paralelo.

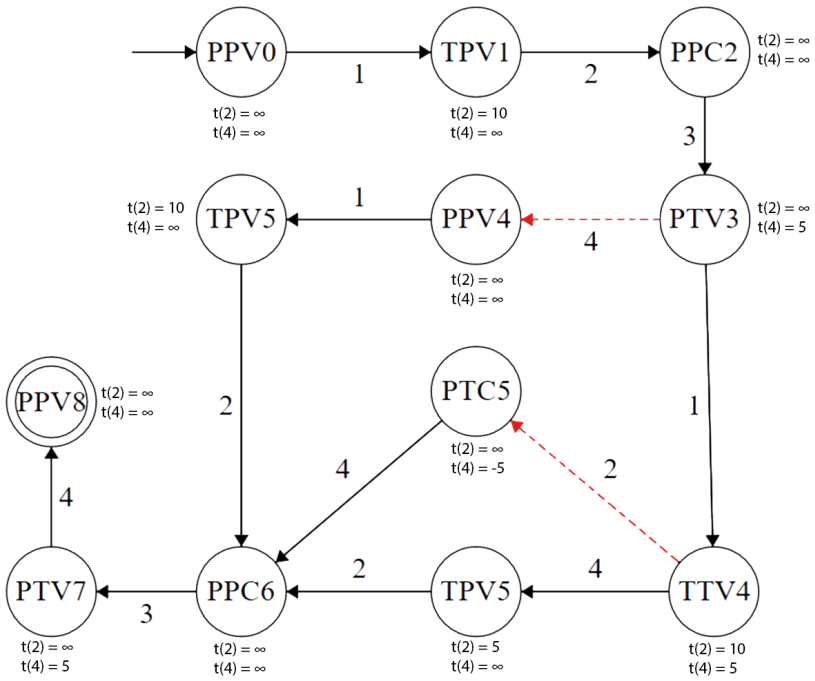


Figura 13 – Grafo acíclico do supervisor da Pequena Fábrica com agendas de eventos associadas aos estados e duplicidade de estados e heuristica de poda, para uma profundidade de 8 eventos.

Neste caso, a única sequência resultante da execução do algoritmo de Menor Tempo Heurístico foi a seguinte:

$$s_1 = (1, 2, 3, 1, 4, 2, 3, 4)$$

A sequência s_1 é a mesma sequência encontrada no Exemplo 3.2.2 e no Exemplo 4.1.1.

4.2.1 Vantagens e Desvantagens

O algoritmo de Menor Tempo Heurístico apresenta como vantagem uma redução significativa do tempo de execução do algoritmo de Menor Tempo Exato, gerando boas soluções. É importante notar, porém, que o algoritmo continua não polinomial, de forma que, dependendo da estrutura do supervisor e dos intervalos de tempo utilizados, a heurística pode não ser muito significativa quanto à redução do tempo de execução. Além disso, o algoritmo necessita de restrições quanto ao número de vezes que cada evento controlável pode ocorrer, a fim de alcançar um estado marcado.

5 Testes e Análise de Resultados

Foram executados testes para três plantas utilizando os quatro algoritmos descritos no trabalho. Na análise serão considerados o tempo de execução do algoritmos, o tempo de execução da sequência e o paralelismo da sequência.

5.1 Plantas para Teste de Execução

Foram escolhidas duas plantas para teste, a primeira, chamada Pequena Fábrica, é uma planta simples e recorrente na literatura de controle supervisório. A planta Sistema Flexível de Manufatura, por sua vez, é um problema maior e está implementada no LACSED.

5.1.1 Pequena Fábrica (PF)

A Pequena Fábrica, também conhecida como *Small Factory*, é uma planta bem conhecida na área de Sistemas a Eventos Discretos, sendo utilizada em (WONHAM, 2014). A planta é composta de duas máquinas conectadas por um *buffer* unitário. A primeira máquina trabalha no produto e deposita no *buffer*, a segunda máquina retira o produto do *buffer* e realiza outro trabalho sobre ele. Em geral, a especificação de segurança da Pequena Fábrica é garantir que não haja o *underflow* nem o *overflow* do *buffer*, ou seja, garantir que a primeira máquina não coloque um produto no *buffer* se ele já está cheio e que a segunda máquina não tente retirar um produto do *buffer* se este está vazio, como visto no diagrama da Figura 3.

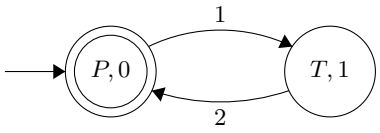
Os autômatos que representam o comportamento dinâmico das máquinas (M_1 e M_2) bem como a especificação estão mostrados na Figura 14. As funções de tarefas ativas f_{at} para M_1, M_2 e E estão definidas no próprio nome do estado onde $f_{ta}((q, i)) = i$.

Partindo dos autômatos das máquinas e do autômato da especificação, é possível obter um supervisor monolítico minimamente restritivo, controlável e não bloqueante, conforme mostrado na Figura 15.

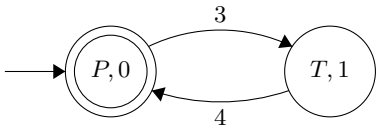
Utilizando as propriedades definidas anteriormente para a função de tarefas ativas, na Tabela 1 está definido o número de tarefas ativas para cada estado do supervisor.

Para que possa ser realizada a execução de algoritmos de menor/maior caminho, faz-se necessário tornar o autômato em um grafo acíclico. Tal operação é executada virtualmente dentro dos algoritmos.

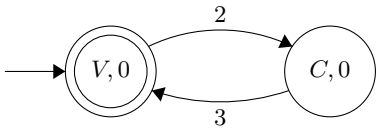
Para simular o sistema, foi definido um intervalo de tempo entre a ocorrência de



(a) Máquina M_1 ($P \leftarrow$ Parada, $T \leftarrow$ Trabalhando)



(b) Máquina M_2 ($P \leftarrow$ Parada, $T \leftarrow$ Trabalhando)



(c) Especificação E ($V \leftarrow$ Vazio, $C \leftarrow$ Cheio)

Figura 14 – Autômatos para as máquinas M_1 e M_2 e para a especificação E acompanhados do número de tarefas ativas em cada estado.

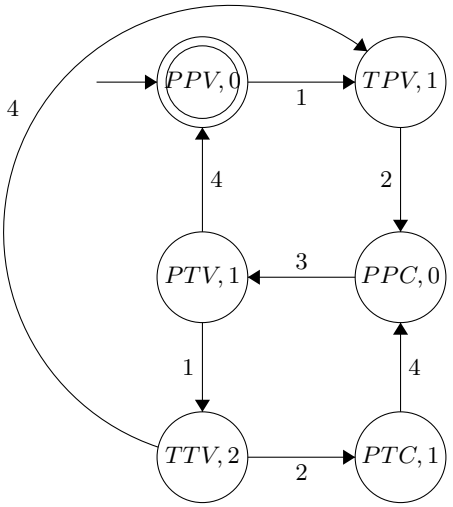


Figura 15 – Supervisor da Pequena Fábrica

Eventos Controláveis	Eventos Não Controláveis	Intervalo de Tempo [$u.t.$]
1	2	10
3	4	5

Tabela 2 – Intervalo de tempo entre pares de eventos da Pequena Fábrica

um evento controlável e um evento não controlável correspondente, conforme mostrado na Tabela 2. Do ponto de vista físico, esses tempos indicam que a máquina 1 tem um tempo de processamento de 10 $u.t.$ e a máquina 2 tem um tempo de processamento de 5 $u.t.$, ou seja, que existe um intervalo de tempo entre o evento de ligar a máquina e o evento de desligar a máquina.

Foram testados oito algoritmos diferentes, sendo eles:

MPL - Algoritmo do Máximo Paralelismo Lógico, que busca maximizar a função acumulativa de tarefas ativas do supervisor (F_{ta}), maximizando o número de tarefas em execução ao mesmo tempo, mas não garantindo a corretude temporal.

AGT - Algoritmo Guloso Temporal, que a cada passo escolhe o evento controlável que minimiza o tempo de execução. Como o algoritmo é guloso, ele é rápido, mas não garante otimalidade em nenhum sentido.

AGP - Algoritmo Guloso quanto ao Paralelismo, que a cada passo escolhe o evento controlável que leva ao estado com maior número de tarefas ativas. Também não

garante otimalidade, mas é rápido.

MPP - Algoritmo do Máximo Paralelismo Projetado nos eventos controláveis. Consiste na execução algoritmo do máximo paralelismo lógico e projeção natural da sequência de eventos resultantes no conjunto de eventos controláveis (Σ_c), deixando os eventos não controláveis ocorrerem quando necessário.

MPT - Algoritmo do Máximo Paralelismo com Restrições Temporais, que busca maximizar o paralelismo, mas somente percorre caminhos que são temporalmente factíveis. Não garante otimalidade temporal nem quanto ao paralelismo, mas gera boas soluções.

MTH - Algoritmo de Menor Tempo Heurístico, que busca minimizar o tempo de execução de maneira exata, porém somente permite a execução de eventos não controláveis quando não há nenhum evento controlável que pode ser executado. Não garante otimalidade com relação à minimização do tempo, mas gera resultados bons.

MPR - Algoritmo do Máximo Paralelismo Lógico com Recálculo, que consiste da execução do algoritmo de máximo paralelismo lógico, mas a cada vez que um evento não controlável não previsto na sequência é executado, o algoritmo é reexecutado a partir do último estado alcançado.

MTE - Algoritmo de Menor Tempo Exato, encontra uma sequência ótima que minimiza o tempo de execução. Como o problema é não polinomial, a solução ótima é difícil de ser obtida, principalmente para problemas maiores.

Foram realizados experimentos para seis tamanhos de lotes, para a produção de 1, 500, 1000, 1500, 2000 e 2500 produtos, onde cada produto é produzido com uma sequência de quatro eventos. O tempo de execução médio, para 30 execuções de cada experimento está explicitado na Tabela 3.

Produtos	**MPL**	**AGT**	**AGP**	**MPP**	**MPT**	**MTH**	**MPR**	**MTE**
1	**0,23**	0,27	0,23	0,83	0,33	0,37	0,23	0,37
500	**2,50**	2,90	6,03	5,87	7,07	6,30	5,37	11,93
1000	**5,40**	4,83	10,53	13,27	13,93	12,40	10,73	23,83
1500	**9,60**	8,50	14,86	20,40	22,67	18,13	16,00	33,40
2000	**11,47**	9,93	19,27	24,73	26,97	25,90	23,17	52,37
2500	**16,67**	12,53	36,90	35,77	36,63	34,53	25,03	55,77

Tabela 3 – Tempo de execução médio dos algoritmos desenvolvidos, em milissegundos.

Como pode ser observado na Tabela 3, o algoritmo mais rápido, desconsiderando os algoritmos gulosos, em quase todos os lotes foi o Máximo Paralelismo Lógico, sendo mais rápido, em alguns testes, que o Algoritmo Guloso Temporal. A não ser no lote

unitário, o algoritmo mais demorado foi o de Menor Tempo Exato, o que já era esperado, considerando que é um algoritmo exato.

Como a Pequena Fábrica é um exemplo pequeno, os resultados obtidos por todos os algoritmos, com relação ao tempo de produção dos lotes e número acumulado de tarefas ativas foi igual, mesmo pelos algoritmos gulosos, com exceção do algoritmo do Máximo Paralelismo Lógico, que gera sequências que podem não ser factíveis temporalmente.

5.1.2 Sistema Flexível de Manufatura (SFM)

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

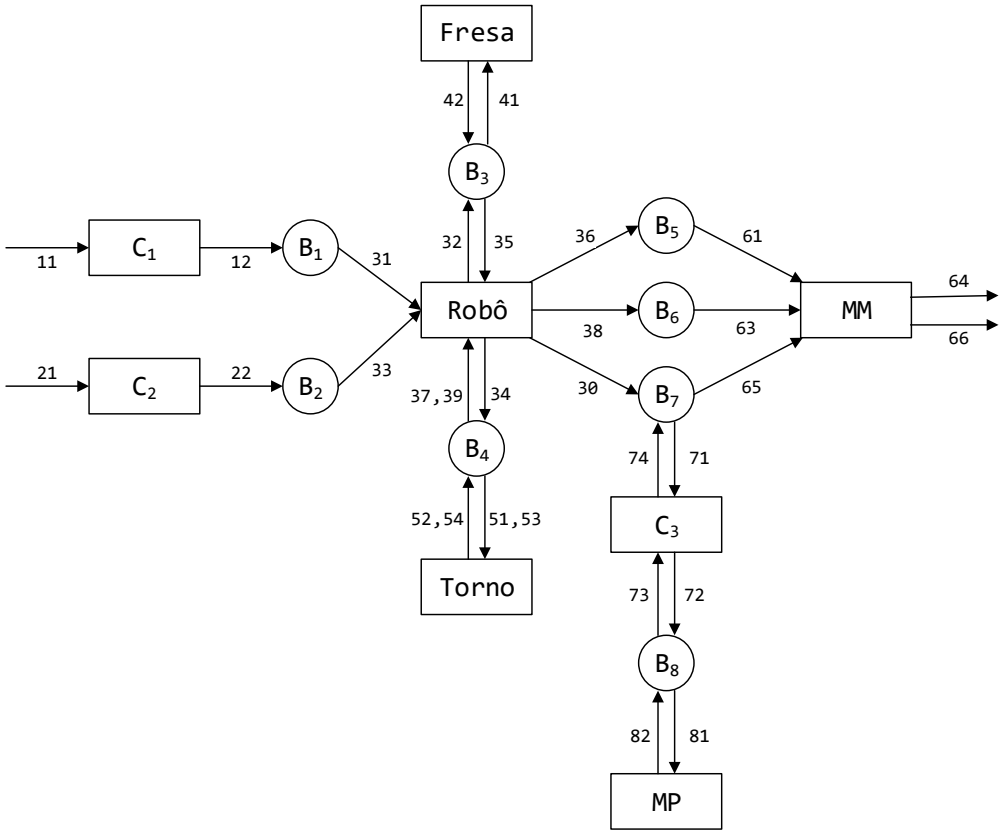
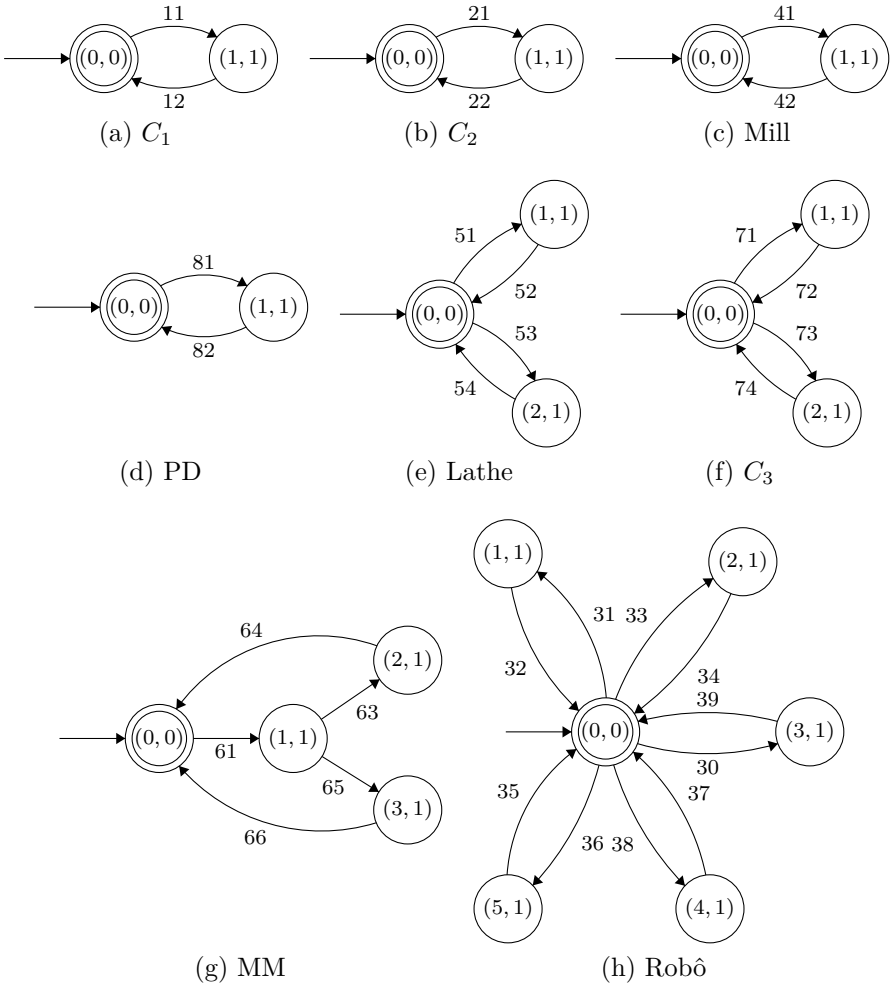


Figura 16 – Diagrama da Sistema Flaxível de Manufatura

Na Figura 17 estão representados os autômatos que modelam a dinâmica das máquinas que compõem o SFM. O número de tarefas ativas em cada estado está representado no nome dos estados, de forma que para um estado (k, i), temos $f_{ta}((k, i)) = i$.



(a) C_1 (b) C_2 (c) Mill

(d) PD (e) Lathe (f) C_3

(g) MM (h) Robô

Figura 17 – Plantas do Sistema Flexível de Manufatura

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

Na Tabela 4 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 onde os eventos controláveis 63 e 65 somente podem ocorrer, no mínimo, 15 unidades de tempo após o evento 61.

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

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

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

Tabela 4 – Intervalo de tempo entre pares de eventos

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:

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

Utilizando o supervisor monolítico para o SFM, obtido pela Teoria de Controle Supervisório convencional, composto por 45.504 estados e 200.124 transições, em conjunto com as especificações temporais, executando os oito algoritmos para uma produção de 1, 5, 10, 15 e 20 pares de produtos A e B, sendo necessários 44 eventos para a produção de cada par, obtivemos os resultados mostrados a seguir:

A Tabela 5 contém o tempo de execução dos oito algoritmos para 1, 5, 10, 15, 20 e 25 lotes de pares de produtos (um produto A e um produto B). Um lote de 25 pares de produtos, por exemplo, consiste em uma sequência de 1100 eventos.

Lotes	**MPL**	**AGT**	**AGP**	**MPP**	**MPT**	**MTH**	**MPR**	**MTE**
1	**0,031**	0,251	0,240	**0,047**	0,243	0,253	0,194	41,093
5	0,502	0,252	0,250	0,515	**0,289**	**0,440**	8,772	-
10	2,343	0,328	0,259	2,371	**0,419**	**0,970**	7,662	-
15	6,058	0,251	0,251	6,071	**0,604**	**1,919**	278,318	-
20	11,790	0,251	0,253	11,723	**0,858**	**3,294**	701,601	-
25	19,329	0,253	0,264	19,317	**1,198**	**5,011**	1.433,889	-

Tabela 5 – Tempo de execução médio dos algoritmos desenvolvidos, em segundos.

Como pode ser observado na Tabela 5, todos as algoritmos, com exceção do Máximo Paralelismo com Recálculo e Mínimo Tempo Exato, apresentam um tempo de execução aceitável, mesmo para a produção de um número significativo de peças. Os algoritmos gulosos são executados em tempo constante e os algoritmos heurísticos baseados no tempo (MPT e MTH) apresentaram melhor desempenho, desconsiderando os algoritmos gulosos, principalmente porque as restrições temporais reduzem o *branching factor* e, dessa maneira, diminuindo o espaço de busca. Somente foi possível calcular o algoritmo exato para um par de peças.

Além do tempo de execução do algoritmo, outro dado interessante é o tempo de produção gerado pelas sequências obtidas por cada algoritmo. A Tabela 6 contém o tempo de execução das sequências obtidas por sete algoritmos para 1, 5, 10, 15, 20 e 25 lotes de pares de produtos. Nesta tabela estão omitidos os resultados das sequências obtidas pelo algoritmo de Máximo Paralelismo Lógico, porque as sequências obtidas por tal algoritmo não são temporalmente factíveis, porém as sequências obtidas com pelo Máximo Paralelismo Lógico geram as sequências do Máximo Paralelismo Projetado.

Lotes	**AGT**	**AGP**	**MPP**	**MPT**	**MTH**	**MPR**	**MTE**
1	321,0	321,0	282,2	286,0	**250,0**	261,0	**250,0**
5	1.055,0	1.055,0	1.146,8	**924,0**	**924,0**	978,0	-
10	1.965,0	1.965,0	2.174,2	**1.744,0**	**1.744,0**	1.888,0	-
15	2.875,0	2.875,0	3.208,6	**2.564,0**	**2.564,0**	2.798,0	-
20	3.785,0	3.785,0	4.236,8	**3.384,0**	**3.384,0**	3.704,0	-
25	4.693,9	4.695,0	5.304,4	**4.204,0**	**4.204,0**	4.618,0	-

Tabela 6 – Tempo de produção médio do lote das sequências obtida pelo algoritmo em unidades de tempo.

Como pode ser observado nos dados da Tabela 6, os piores resultados, excluindo o menor lote, foram gerados pelo algoritmo baseado puramente no Máximo Paralelismo Projetado (MPP), principalmente porque ele não utiliza informações temporais durante o processo de otimização e, além disso, o máximo paralelismo tende a evitar o desligamento das máquinas para maximizar o número de máquinas funcionando. Em sequência estão os algoritmos gulosos, com um desempenho similar entre eles. O algoritmo de Máximo Paralelismo com Recálculo obteve um resultado ligeiramente melhor que os algoritmos gulosos. Os melhores algoritmos foram o Máximo Paralelismo com Restrições Temporais (MPT) e o algoritmo de Menor Tempo Heurístico (MTH) apresentando resultados semelhantes, com exceção do menor lote, onde o MTH apresenta um resultado melhor que o MPT e igual ao resultado do cálculo exato, porém em um tempo significativamente menor.

Considerando que parte desse trabalho tem base em uma análise indireta do tempo, levando em conta a maximização do paralelismo como um critério de desempenho aplicado a sequências, é interessante analisar o paralelismo acumulado médio das sequências obtidas pelos algoritmos.

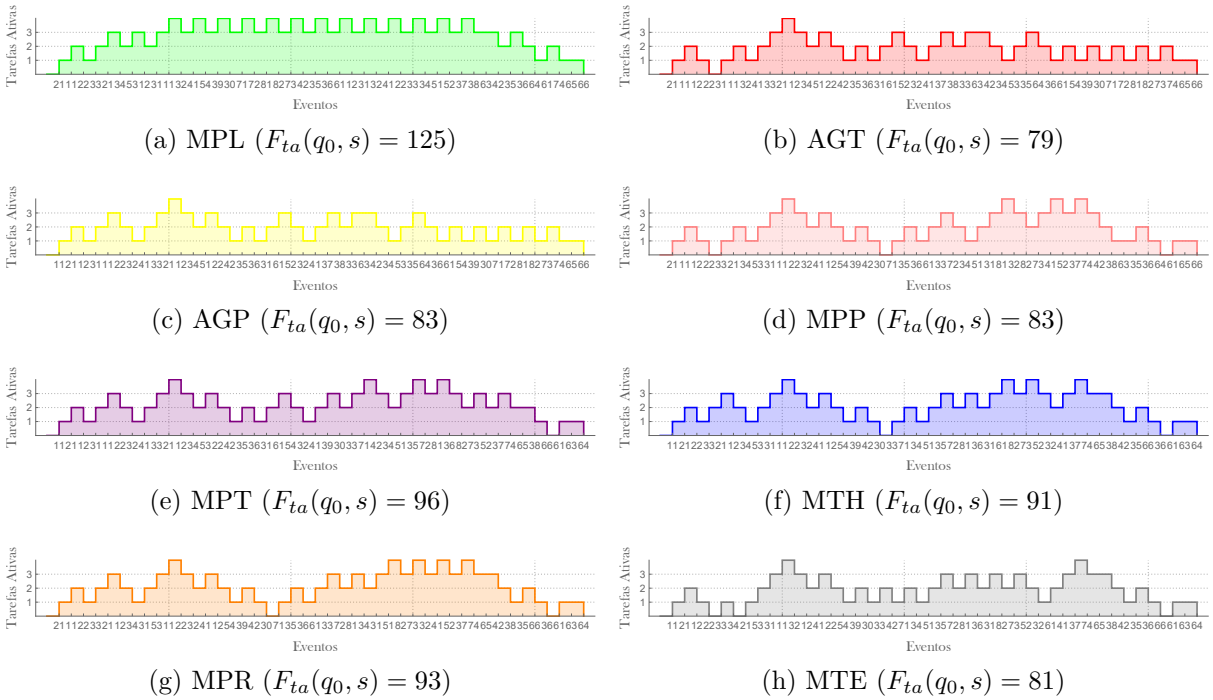
Como já é esperado, podemos observar na Tabela 7 que o algoritmo de Máximo

Lotes	MPL	AGT	AGP	MPP	MPT	MTH	MPR	MTE
1	**125**	79	83	80	**96**	91	91	81
5	**1.048**	625	629	535	**716**	692	602	-
10	**2.198**	1.320	1.324	1.116	**1.491**	1.437	1.227	-
15	**3.348**	2.015	2.020	1.697	**2.266**	2.182	1.852	-
20	**4.498**	2.710	2.715	2.275	**3.041**	2.927	2.477	-
25	**5.648**	3.394	3.409	2.833	**3.816**	3.672	3.102	-

Tabela 7 – Paralelismo acumulado ($F_{ta}(s^*)$) médio das sequências obtidas pelos algoritmos

Paralelismo Lógico (MPL) é aquele que apresenta maior paralelismo acumulado em todos os testes, seguido do Máximo Paralelismo com Restrições Temporais (MPT). Como o MPT gera, exceto para o menor lote, sequências com o menor tempo de produção e menor tempo de execução, desconsiderando os métodos gulosos, há um forte indicativo de que maximizar o paralelismo é uma boa estratégia para minimizar o tempo de produção. Ainda assim, o paralelismo acumulado da sequência obtida pelo algoritmo exato (MTE) mostra que existem sequências com baixo paralelismo que apresentam um bom desempenho com relação ao tempo de execução.

A Figura 18 apresenta gráficos de tarefas ativas por evento executado em sequências para a produção de um par de produtos no SFM. A área sob as curvas dos gráficos e o resultado a função acumulativa de tarefas ativas (F_{ta}).



(a) MPL ($F_{ta}(q_0, s) = 125$)

(b) AGT ($F_{ta}(q_0, s) = 79$)

(c) AGP ($F_{ta}(q_0, s) = 83$)

(d) MPP ($F_{ta}(q_0, s) = 83$)

(e) MPT ($F_{ta}(q_0, s) = 96$)

(f) MTH ($F_{ta}(q_0, s) = 91$)

(g) MPR ($F_{ta}(q_0, s) = 93$)

(h) MTE ($F_{ta}(q_0, s) = 81$)

Figura 18 – Gráficos relacionando o número de tarefas ativas no SFM para cada evento executado, para uma sequência gerada por cada algoritmo.

5.1.2.1 Tempos de Execução Aleatórios

Uma análise interessante é verificar o desempenho das sequências obtidas pelos algoritmos quando o tempo de execução da sequência é diferente do tempo simulado das plantas. Para isso, foi realizado um experimento, onde foi obtida uma sequência utilizando os algorítimos MPL, AGT, AGP, MPT e MTH, foi realizada a projeção natural da sequência no conjunto de eventos controláveis (Σ_c), de forma que a sequência de eventos controláveis resultante foi simulada para execuções onde o intervalo de tempo entre dois eventos relacionados é gerado por uma variável aleatória com distribuição normal.

Inicialmente, foi obtida uma sequência, para cada um dos algoritmos, utilizando os tempos da Tabela 4. Após isso, foram utilizados na simulação intervalos de tempo gerados por uma variável aleatória normal, cuja média é dada pelos tempos da Tabela 4 e o desvio padrão foi arbitrariamente definido como 5. A média e o desvio padrão dos resultados para 30 execuções de cada simulação, para lotes de 1, 5, 10, 15 e 20 pares de peças está mostrado na Tabela 8. Para os mesmos lotes e intervalos de tempo aleatório foram feitas 30 simulações do Algoritmo de Máximo Paralelismo com Recálculo (MPR), a fim de comparar os resultados.

Lotes	MPL		AGT		AGP	
	Média	Desv.	Média	Desv.	Média	Desv.
1	324,36	11,25	320,03	14,03	326,98	18,65
5	1.147,03	35,13	1.076,06	45,30	1.073,86	37,47
10	2.135,47	40,06	1.990,77	41,24	2.012,91	41,48
15	3.166,07	51,60	2.937,05	50,87	2.950,24	50,59
20	4.149,44	59,26	3.883,45	65,85	3.860,68	60,76
Lotes	MPT		MTH		MPR	
	Média	Desv.	Média	Desv.	Média	Desv.
1	282,25	13,67	**260,59**	12,46	**263,57**	20,06
5	**963,76**	25,87	**966,61**	22,40	982,40	25,18
10	**1.847,30**	39,47	**1.856,28**	34,85	1.897,76	40,64
15	**2.706,56**	40,32	**2.716,87**	49,78	2.824,96	56,09
20	**3.588,71**	54,43	**3.576,93**	46,46	3.729,03	71,43

Tabela 8 – Tempo de produção para tempos de trabalho das plantas gerados aleatoriamente com distribuição normal.

Como pode ser observado na Tabela 8, novamente os melhores resultados ficaram com os algoritmos de Máximo Paralelismo com Restrições Temporais (MPT) e com o algoritmo de Menor Tempo Heurístico (MTH), o algoritmo de Máximo Paralelismo com Recálculo (MPR), novamente obteve resultados melhores, porém muito próximos aos algoritmos gulosos (AGT e AGP). Dos algoritmos testados, o pior resultado foi o do Máximo Paralelismo Lógico (MPL), que nesse caso equivale ao algoritmo de Máximo Paralelismo Projetado (MPP).

Conclusão

Neste trabalho foram apresentadas e analisadas oito técnicas de escalonamento de eventos para sistemas a eventos discretos. Todas as técnicas utilizam informações estruturais do autômato do supervisor, diminuindo muito o tempo de execução dos algoritmos, principalmente pela garantia de que todas as sequências geradas são logicamente factíveis.

Foram propostas duas abordagens, uma baseada na maximização do paralelismo, ou seja, aumento do número de máquinas operando ao mesmo tempo, e uma baseada na minimização do tempo. Dentre os algoritmos, destacaram-se dois, o algoritmo de Máximo Paralelismo com Restrições Temporais e o algoritmo de Menor Tempo Heurístico, sendo ambos algoritmos heurísticos. Os dois algoritmos, apesar de não apresentarem garantia de otimalidade, encontraram as melhores soluções. De uma maneira geral, o algoritmo de Máximo Paralelismo com Restrições Temporais é mais interessante por ter complexidade polinomial, com relação ao número de eventos a serem executados, enquanto o tempo de execução do algoritmo de Menor Tempo Heurístico pode apresentar desempenho melhor ou pior dependendo da topologia do supervisor e das restrições impostas pelo tempo.

Outro fator importante é notar que a solução exata dos problemas de escalonamento são inviáveis. Para um problema maior, como o Sistema Flexível de Manufatura, somente foi possível encontrar a solução para o lote de um par de produtos. Essa dificuldade justifica a busca de métodos mais rápidos e que obtenham sequências de qualidade.

5.2 Trabalhos Futuros

Como proposta de trabalhos futuros, pode-se listar:

- Aplicar os algoritmos desenvolvidos para supervisores modulares, realizando a busca de maneira construtiva.
- Aplicar o conceito do máximo paralelismo em autômatos temporizados.
- Utilizar sequências obtidas pelo máximo paralelismo lógico como ponto de partida para algoritmos de busca local e otimização evolucionária.
- Desenvolver técnicas que envolvam o paralelismo na maximização do *throughput* em sistemas de manufatura.

Referências

ABDEDDAïM, Y.; ASARIN, E.; MALER, O. Scheduling with Timed Automata. *Theoretical Computer Science*, v. 354, n. 2, p. 272 – 300, 2006. ISSN 0304-3975. Citado na página 25.

ALMEDER, C.; MöNCH, L. Metaheuristics for scheduling jobs with incompatible families on parallel batching machines. *Journal of the Operational Research Society*, v. 62, n. 12, p. 2083–2096, 2011. Citado na página 26.

ALVES, L.; CIPRIANO, H.; PENA, P. ULTRADES - UMA BIBLIOTECA PARA MODELAGEM, ANALISE E CONTROLE DE SISTEMAS A EVENTOS DISCRETOS. In: *Anais do XII Simpósio Brasileiro de Automação Inteligente, SBAI'15*. Natal, Brazil: [s.n.], 2015. Citado na página 34.

AYTUG, H. et al. Executing production schedules in the face of uncertainties: A review and some future directions. *European Journal of Operational Research*, v. 161, p. 86–110, 2005. Citado na página 25.

CASSANDRAS, C.; LAFORTUNE, S. *Introduction to Discrete Event Systems*. 2. ed. [S.l.]: Springer, 2008. Citado na página 34.

COSTA, T.; PENA, P.; TAKAHASHI, R. Controle Supervisório e Otimização: Abordagem VNS-2Opt e Robustez à Perturbações. In: *Anais do XI Simpósio Brasileiro de Automação Inteligente, SBAI'13*. Fortaleza, Brazil: [s.n.], 2013. Citado na página 26.

EDELKAMP, S.; SCHROEDL, S. *Heuristic search: theory and applications*. [S.l.]: Morgan Kaufmann, 2012. 1–865 p. ISBN 9780123725127. Citado na página 36.

GAREY, M. R.; JOHNSON, D. S. *Computers and intractability*. [S.l.]: W.H. Freeman, 1979. Citado na página 25.

GHALLAB, M.; NAU, D. S.; TRAVERSO, P. *Automated planning*. [S.l.]: Elsevier/Morgan Kaufmann, 2004. Citado na página 26.

HERZIG, A. et al. 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)*. [S.l.: s.n.], 2014. p. 435–440. Citado na página 25.

KOBETSKI, A.; FABIAN, M. Scheduling of discrete event systems using mixed integer linear programming. In: *Discrete Event Systems, 2006 8th International Workshop on*. [S.l.: s.n.], 2006. p. 76–81. Citado 2 vezes nas páginas 25 e 26.

LIU, Z. et al. Optimal scheduling of time-constrained single-arm cluster tools with wafer revisiting. In: *Discrete Event Systems, 2016 13th International Workshop on*. [S.l.: s.n.], 2016. p. 355–360. Citado na página 25.

LóPEZ-MELLADO, E.; VILLANUEVA-PAREDES, N.; ALMEYDA-CANEPA, H. Modelling of batch production systems using petri nets with dynamic tokens.

Mathematics and Computers in Simulation, v. 67, n. 6, p. 541 – 558, 2005. ISSN 0378-4754. Citado na página 25.

MACKWORTH, D. L. P. A. K. *Artificial intelligence : foundations of computational agents*. [S.l.]: Cambridge University Press, 2010. ISBN 978-0-511-72946-1,978-0-521-51900-7,0511729464,9780511726163,0511726163,9780511794797,0511794797. Citado na página 26.

OLIVEIRA, A. C. et al. Clonal selection algorithms for task scheduling in a flexible manufacturing cell with supervisory control. In: IEEE. *Evolutionary Computation (CEC), 2013 IEEE Congress on*. [S.l.], 2013. p. 982–988. Citado na página 26.

PARK, S.-J.; YANG, J.-M. Supervisory control for real-time scheduling of periodic and sporadic tasks with resource constraints. *Automatica*, v. 45, n. 11, p. 2597 – 2604, 2009. ISSN 0005-1098. Citado 2 vezes nas páginas 25 e 26.

PEARL, J. *Heuristics: intelligent search strategies for computer problem solving*. [S.l.]: Addison-Wesley Pub. Co, 1984. (The Addison-Wesley series in artificial intelligence). ISBN 0201055945,9780201055948. Citado na página 26.

PENA, P. N. et al. Control of flexible manufacturing systems under model uncertainty using supervisory control theory and evolutionary computation schedule synthesis. *Information Sciences*, v. 329, p. 491 – 502, 2016. ISSN 0020-0255. Special issue on Discovery Science. Citado na página 26.

PINEDO, M. L. *Scheduling: Theory, Algorithms, and Systems*. 3rd. ed. [S.l.]: Springer Publishing Company, Incorporated, 2012. ISBN 0387789340, 9780387789347. Citado na página 25.

PINHA, D.; QUEIROZ, M. de; CURY, J. Optimal scheduling of a repair shipyard based on supervisory control theory. In: *Automation Science and Engineering (CASE), 2011 IEEE Conference on*. [S.l.: s.n.], 2011. p. 39–44. ISSN 2161-8070. Citado 2 vezes nas páginas 25 e 26.

RAMADGE, P. J. G.; WONHAM, W. M. The Control of Discrete Event Systems. *Proc. of the IEEE*, v. 77, n. 1, p. 81–98, jan. 1989. Citado na página 25.

SCHRIJVER, A. *Theory of Linear and Integer Programming*. New York, NY, USA: John Wiley & Sons, Inc., 1986. ISBN 0-471-90854-1. Citado na página 25.

SU, R.; SCHUPPEN, J. van; ROODA, J. The synthesis of time optimal supervisors by using heaps-of-pieces. *Automatic Control, IEEE Transactions on*, v. 57, n. 1, p. 105–118, Jan 2012. ISSN 0018-9286. Citado 2 vezes nas páginas 25 e 26.

VILELA, J. N.; PENA, P. N. Supervisor abstraction to deal with planning problems in manufacturing systems. In: *Discrete Event Systems, 2016 13th International Workshop on*. [S.l.: s.n.], 2016. p. 117–122. Citado na página 47.

WANG, W.; YUAN, C.; XIAOBING, L. A fuzzy approach to multi-product mixed production job shop scheduling algorithm. In: *Fuzzy Systems and Knowledge Discovery, 2008. FSKD '08. Fifth International Conference on*. [S.l.: s.n.], 2008. v. 1, p. 95–99. Citado na página 25.

WARE, S.; SU, R. Incremental scheduling of discrete event systems. In: *Discrete Event Systems, 2016 13th International Workshop on*. [S.l.: s.n.], 2016. p. 147–152. Citado na página 26.

WONHAM, W. M. *Supervisory Control of Discrete-Event Systems*. Toronto, Canada: Systems Control Group, Department of Electrical & Computer Engineering, University of Toronto, 2014. Citado 2 vezes nas páginas 33 e 63.

Apêndices

APÊNDICE A – Log de Simulação

Log da simulação do cálculo de tempo para uma sequência de 44 eventos para o supervisor do Sistema Flexível de Manufatura, utilizado como exemplo na Subseção 5.1.2. A simulação é composta, a cada iteração de uma linha com o estado atual, uma tabela que relaciona os eventos habilitados no estado atual, o acréscimo de tempo que eles provocam, o número restante de ocorrências (99999 ocorrências caso não haja limite) e o acréscimo de paralelismo que ele gera. Em seguida está qual evento da tabela é escolhido, o tempo total simulado e o paralelismo acumulado até então.

```
Sequência:
21, 11, 22, 33, 21, 12, 34, 53, 31, 11, 22, 32,
41, 12, 54, 39, 42, 30, 33, 71, 34, 51, 35, 72,
81, 36, 31, 61, 82, 73, 52, 32, 41, 37, 74, 65,
38, 42, 35, 66, 36, 61, 63, 64.


-------------
Estado Atual: 0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      11 |     0 |           2 |        1 |
|      21 |     0 |           2 |        1 |

Evento Escolhido: 21
Tempo Total: 0
Paralelismo Total: 1

-------------
Estado Atual: 0|1|0|0|0|0|0|0|0|0|0|0|0|0|0|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      11 |     0 |           2 |        2 |
|      22 |    26 |       99999 |        0 |

Evento Escolhido: 11
Tempo Total: 0
Paralelismo Total: 3

-------------
Estado Atual: 1|1|0|0|0|0|0|0|0|0|0|0|0|0|0|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      12 |    26 |       99999 |        1 |
|      22 |    26 |       99999 |        1 |

Evento Escolhido: 22
Tempo Total: 26
Paralelismo Total: 4

-------------
Estado Atual: 1|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      12 |     0 |       99999 |        0 |
|      33 |     0 |           2 |        2 |

Evento Escolhido: 33
Tempo Total: 26
Paralelismo Total: 6

-------------
Estado Atual: 1|0|0|0|2|0|0|0|0|0|0|0|0|0|0|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      12 |     0 |       99999 |        1 |
|      21 |     0 |           1 |        3 |
|      34 |    20 |       99999 |        1 |

Evento Escolhido: 21
Tempo Total: 26
Paralelismo Total: 9

-------------
Estado Atual: 1|1|0|0|2|0|0|0|0|0|0|0|0|0|0|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      12 |     0 |       99999 |        2 |
|      22 |    26 |       99999 |        2 |
|      34 |    20 |       99999 |        2 |

Evento Escolhido: 12
Tempo Total: 26
Paralelismo Total: 11

-------------
Estado Atual: 0|1|0|0|2|0|0|0|1|0|0|0|0|0|0|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      22 |    26 |       99999 |        1 |
|      34 |    20 |       99999 |        1 |

Evento Escolhido: 34
```

```
Tempo Total: 46
Paralelismo Total: 12

-------------
Estado Atual: 0|1|0|0|0|0|0|0|1|0|0|1|0|0|0|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      22 |     6 |       99999 |        0 |
|      51 |     0 |           1 |        2 |
|      53 |     0 |           1 |        2 |
|      31 |     0 |           2 |        2 |

Evento Escolhido: 53
Tempo Total: 46
Paralelismo Total: 14

-------------
Estado Atual: 0|1|0|2|0|0|0|0|1|0|0|0|0|0|0|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      22 |     6 |       99999 |        1 |
|      54 |    33 |       99999 |        1 |
|      31 |     0 |           2 |        3 |

Evento Escolhido: 31
Tempo Total: 46
Paralelismo Total: 17

-------------
Estado Atual: 0|1|0|2|1|0|0|0|0|0|0|0|0|0|0|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      11 |     0 |           1 |        4 |
|      22 |     6 |       99999 |        2 |
|      54 |    33 |       99999 |        2 |
|      32 |    22 |       99999 |        2 |

Evento Escolhido: 11
Tempo Total: 46
Paralelismo Total: 21

-------------
Estado Atual: 1|1|0|2|1|0|0|0|0|0|0|0|0|0|0|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      12 |    26 |       99999 |        3 |
|      22 |     6 |       99999 |        3 |
|      54 |    33 |       99999 |        3 |
|      32 |    22 |       99999 |        3 |

Evento Escolhido: 22
Tempo Total: 52
Paralelismo Total: 24

-------------
Estado Atual: 1|0|0|2|1|0|0|0|0|1|0|0|0|0|0|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      12 |    20 |       99999 |        2 |
|      54 |    27 |       99999 |        2 |
|      32 |    16 |       99999 |        2 |

Evento Escolhido: 32
Tempo Total: 68
Paralelismo Total: 26

-------------
Estado Atual: 1|0|0|2|0|0|0|0|0|1|1|0|0|0|0|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      12 |     4 |       99999 |        1 |
|      41 |     0 |           2 |        3 |
|      54 |    11 |       99999 |        1 |

Evento Escolhido: 41
Tempo Total: 68
Paralelismo Total: 29

-------------
Estado Atual: 1|0|1|2|0|0|0|0|0|1|0|0|0|0|0|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      12 |     4 |       99999 |        2 |
|      42 |    31 |       99999 |        2 |
|      54 |    11 |       99999 |        2 |

Evento Escolhido: 12
Tempo Total: 72
Paralelismo Total: 31

-------------
Estado Atual: 0|0|1|2|0|0|0|0|1|1|0|0|0|0|0|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      42 |    27 |       99999 |        1 |
|      54 |     7 |       99999 |        1 |

Evento Escolhido: 54
Tempo Total: 79
Paralelismo Total: 32

-------------
Estado Atual: 0|0|1|0|0|0|0|0|1|1|0|3|0|0|0|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      42 |    20 |       99999 |        0 |
|      39 |     0 |           1 |        2 |

Evento Escolhido: 39
Tempo Total: 79
Paralelismo Total: 34

-------------
Estado Atual: 0|0|1|0|5|0|0|0|1|1|0|0|0|0|0|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      42 |    20 |       99999 |        1 |
|      30 |    21 |       99999 |        1 |

Evento Escolhido: 42
Tempo Total: 99
Paralelismo Total: 35

-------------
Estado Atual: 0|0|0|0|5|0|0|0|1|1|2|0|0|0|0|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      30 |     1 |       99999 |        0 |

Evento Escolhido: 30
Tempo Total: 100
```

```
Paralelismo Total: 35

-------------
Estado Atual: 0|0|0|0|0|0|0|0|1|1|2|0|0|0|1|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      33 |     0 |           1 |        1 |
|      35 |     0 |           2 |        1 |
|      71 |     0 |           1 |        1 |

Evento Escolhido: 33
Tempo Total: 100
Paralelismo Total: 36

-------------
Estado Atual: 0|0|0|0|2|0|0|0|1|0|2|0|0|0|1|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      21 |     0 |           0 |        2 |
|      34 |    20 |       99999 |        0 |
|      71 |     0 |           1 |        2 |

Evento Escolhido: 71
Tempo Total: 100
Paralelismo Total: 38

-------------
Estado Atual: 0|0|0|0|2|0|1|0|1|0|2|0|0|0|0|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      21 |     0 |           0 |        3 |
|      34 |    20 |       99999 |        1 |
|      72 |    26 |       99999 |        1 |

Evento Escolhido: 34
Tempo Total: 120
Paralelismo Total: 39

-------------
Estado Atual: 0|0|0|0|0|0|1|0|1|0|2|1|0|0|0|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      21 |     0 |           0 |        2 |
|      51 |     0 |           1 |        2 |
|      53 |     0 |           0 |        2 |
|      35 |     0 |           2 |        2 |
|      72 |     6 |       99999 |        0 |

Evento Escolhido: 51
Tempo Total: 120
Paralelismo Total: 41

-------------
Estado Atual: 0|0|0|1|0|0|1|0|1|0|2|0|0|0|0|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      21 |     0 |           0 |        3 |
|      52 |    39 |       99999 |        1 |
|      35 |     0 |           2 |        3 |
|      72 |     6 |       99999 |        1 |

Evento Escolhido: 35
Tempo Total: 120
Paralelismo Total: 44

-------------
Estado Atual: 0|0|0|1|3|0|1|0|1|0|0|0|0|0|0|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      21 |     0 |           0 |        4 |
|      52 |    39 |       99999 |        2 |
|      36 |    17 |       99999 |        2 |
|      72 |     6 |       99999 |        2 |

Evento Escolhido: 72
Tempo Total: 126
Paralelismo Total: 46

-------------
Estado Atual: 0|0|0|1|3|0|0|0|1|0|0|0|0|0|0|1
| Eventos | Tempo | Ocorrências | Paralel. |
|      21 |     0 |           0 |        3 |
|      52 |    33 |       99999 |        1 |
|      36 |    11 |       99999 |        1 |
|      81 |     0 |           1 |        3 |

Evento Escolhido: 81
Tempo Total: 126
Paralelismo Total: 49

-------------
Estado Atual: 0|0|0|1|3|0|0|1|1|0|0|0|0|0|0|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      21 |     0 |           0 |        4 |
|      52 |    33 |       99999 |        2 |
|      36 |    11 |       99999 |        2 |
|      82 |    25 |       99999 |        2 |

Evento Escolhido: 36
Tempo Total: 137
Paralelismo Total: 51

-------------
Estado Atual: 0|0|0|1|0|0|0|1|1|0|0|0|1|0|0|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      21 |     0 |           0 |        3 |
|      52 |    22 |       99999 |        1 |
|      31 |     0 |           1 |        3 |
|      61 |     0 |           2 |        3 |
|      82 |    14 |       99999 |        1 |

Evento Escolhido: 31
Tempo Total: 137
Paralelismo Total: 54

-------------
Estado Atual: 0|0|0|1|1|0|0|0|1|0|0|0|0|1|0|0|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      11 |     0 |           0 |        4 |
|      21 |     0 |           0 |        4 |
|      52 |    22 |       99999 |        2 |
|      32 |    22 |       99999 |        2 |
|      61 |     0 |           2 |        4 |
|      82 |    14 |       99999 |        2 |

Evento Escolhido: 61
```

```
Tempo Total: 137
Paralelismo Total: 58

-------------
Estado Atual: 0|0|0|1|1|1|0|1|0|0|0|0|0|0|0|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      11 |     0 |           0 |        5 |
|      21 |     0 |           0 |        5 |
|      52 |    22 |       99999 |        3 |
|      32 |    22 |       99999 |        3 |
|      82 |    14 |       99999 |        3 |

Evento Escolhido: 82
Tempo Total: 151
Paralelismo Total: 61

-------------
Estado Atual: 0|0|0|1|1|1|0|0|0|0|0|0|0|0|2
| Eventos | Tempo | Ocorrências | Paralel. |
|      11 |     0 |           0 |        4 |
|      21 |     0 |           0 |        4 |
|      52 |     8 |       99999 |        2 |
|      32 |     8 |       99999 |        2 |
|      73 |     0 |           1 |        4 |

Evento Escolhido: 73
Tempo Total: 151
Paralelismo Total: 65

-------------
Estado Atual: 0|0|0|1|1|1|2|0|0|0|0|0|0|0|0|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      11 |     0 |           0 |        5 |
|      21 |     0 |           0 |        5 |
|      52 |     8 |       99999 |        3 |
|      32 |     8 |       99999 |        3 |
|      74 |    26 |       99999 |        3 |

Evento Escolhido: 52
Tempo Total: 159
Paralelismo Total: 68

-------------
Estado Atual: 0|0|0|0|1|1|2|0|0|0|0|2|0|0|0|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      11 |     0 |           0 |        4 |
|      21 |     0 |           0 |        4 |
|      32 |     0 |       99999 |        2 |
|      74 |    18 |       99999 |        2 |

Evento Escolhido: 32
Tempo Total: 159
Paralelismo Total: 70

-------------
Estado Atual: 0|0|0|0|0|1|2|0|0|0|1|2|0|0|0|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      41 |     0 |           1 |        3 |
|      37 |     0 |           1 |        3 |
|      74 |    18 |       99999 |        1 |
|      11 |     0 |           0 |        3 |
|      21 |     0 |           0 |        3 |

Evento Escolhido: 41
Tempo Total: 159
Paralelismo Total: 73

-------------
Estado Atual: 0|0|1|0|0|1|2|0|0|0|0|2|0|0|0|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      11 |     0 |           0 |        4 |
|      21 |     0 |           0 |        4 |
|      42 |    31 |       99999 |        2 |
|      37 |     0 |           1 |        4 |
|      74 |    18 |       99999 |        2 |

Evento Escolhido: 37
Tempo Total: 159
Paralelismo Total: 77

-------------
Estado Atual: 0|0|1|0|4|1|2|0|0|0|0|0|0|0|0|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      11 |     0 |           0 |        5 |
|      21 |     0 |           0 |        5 |
|      42 |    31 |       99999 |        3 |
|      38 |    25 |       99999 |        3 |
|      74 |    18 |       99999 |        3 |

Evento Escolhido: 74
Tempo Total: 177
Paralelismo Total: 80

-------------
Estado Atual: 0|0|1|0|4|1|0|0|0|0|0|0|0|2|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      11 |     0 |           0 |        4 |
|      21 |     0 |           0 |        4 |
|      42 |    13 |       99999 |        2 |
|      38 |     7 |       99999 |        2 |
|      65 |     0 |           1 |        3 |

Evento Escolhido: 65
Tempo Total: 177
Paralelismo Total: 83

-------------
Estado Atual: 0|0|1|0|4|3|0|0|0|0|0|0|0|0|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      11 |     0 |           0 |        4 |
|      21 |     0 |           0 |        4 |
|      42 |    13 |       99999 |        2 |
|      38 |     7 |       99999 |        2 |
|      66 |    27 |       99999 |        2 |

Evento Escolhido: 38
Tempo Total: 184
Paralelismo Total: 85

-------------
```

```
Estado Atual: 0|0|1|0|0|3|0|0|0|0|0|0|0|1|0|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      11 |     0 |           0 |        3 |
|      21 |     0 |           0 |        3 |
|      42 |     6 |       99999 |        1 |
|      66 |    20 |       99999 |        1 |

Evento Escolhido: 42
Tempo Total: 190
Paralelismo Total: 86

-------------
Estado Atual: 0|0|0|0|0|3|0|0|0|0|2|0|0|1|0|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      11 |     0 |           0 |        2 |
|      21 |     0 |           0 |        2 |
|      35 |     0 |           1 |        2 |
|      66 |    14 |       99999 |        0 |

Evento Escolhido: 35
Tempo Total: 190
Paralelismo Total: 88

-------------
Estado Atual: 0|0|0|0|3|3|0|0|0|0|0|0|0|1|0|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      11 |     0 |           0 |        3 |
|      21 |     0 |           0 |        3 |
|      36 |    17 |       99999 |        1 |
|      66 |    14 |       99999 |        1 |

Evento Escolhido: 66
Tempo Total: 204
Paralelismo Total: 89

-------------
Estado Atual: 0|0|0|0|3|0|0|0|0|0|0|0|0|1|0|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      11 |     0 |           0 |        2 |
|      21 |     0 |           0 |        2 |
|      36 |     3 |       99999 |        0 |

Evento Escolhido: 36
Tempo Total: 207
Paralelismo Total: 89

-------------
Estado Atual: 0|0|0|0|0|0|0|0|0|0|0|1|1|0|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      11 |     0 |           0 |        1 |
|      21 |     0 |           0 |        1 |
|      61 |     0 |           1 |        1 |

Evento Escolhido: 61
Tempo Total: 207
Paralelismo Total: 90

-------------
Estado Atual: 0|0|0|0|0|1|0|0|0|0|0|0|1|0|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      11 |     0 |           0 |        2 |
|      21 |     0 |           0 |        2 |
|      63 |    16 |           1 |        1 |

Evento Escolhido: 63
Tempo Total: 223
Paralelismo Total: 91

-------------
Estado Atual: 0|0|0|0|0|2|0|0|0|0|0|0|0|0|0
| Eventos | Tempo | Ocorrências | Paralel. |
|      11 |     0 |           0 |        2 |
|      21 |     0 |           0 |        2 |
|      64 |    27 |       99999 |        0 |

Evento Escolhido: 64
Tempo Total: 250
Paralelismo Total: 91
```