

**UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS GRADUAÇÃO EM
ENGENHARIA ELÉTRICA**

Octávio Francisco Paschoal Silva Ribeiro dos Santos

**MODELAGEM DE SISTEMAS FLEXÍVEIS DE
MANUFATURA PARA TOMADA DE DECISÃO EM
TEMPO REAL**

**Dissertação submetida ao Programa de Pós-Graduação
em Engenharia Elétrica como parte dos requisitos para
obtenção do título de Mestre em Ciências em Engenharia
Elétrica**

**Área de Concentração: Automação e Sistemas Elétricos
Industriais**

**Orientador: Prof. Dr. Guilherme Sousa Bastos
Coorientador: Prof. Dr. Luiz Edival de Souza**

**Junho de 2013
Itajubá**

**UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS GRADUAÇÃO EM
ENGENHARIA ELÉTRICA**

Octávio Francisco Paschoal Silva Ribeiro dos Santos

**MODELAGEM DE SISTEMAS FLEXÍVEIS DE
MANUFATURA PARA TOMADA DE DECISÃO EM
TEMPO REAL**

Dissertação aprovada por banca examinadora em 24 de Junho de 2013, conferindo ao autor o título de **Mestre em Ciências em Engenharia Elétrica**

Banca Examinadora:

Prof. Dr. Guilherme Sousa Bastos (Orientador)

Prof. Dr. Luiz Edival de Souza (Coorientador)

Prof. Dr. Carlos Henrique Costa Ribeiro

Prof. Dr. Edmilson Marmo Moreira

**Itajubá
2013**

À Walkyria e Artur, a pedra fundamental
À Valdinha e Odete, o exemplo
À Francisco e Heber, as estrelas mais brilhantes do céu.

Agradecimentos

Agradeço primeiramente a Deus, cuja luz me guiou nos caminhos mais difíceis desta empreitada.

Aos meus pais, Walkyria e Artur, pelas palavras de apoio e companhia.

Ao professor Guilherme Sousa Bastos, além de orientador, um amigo leal. Este trabalho não teria sido concluído sem sua ajuda.

Ao professor Luiz Edival de Souza, pela ajuda com as redes de *Petri* e suporte fundamental no início deste mestrado.

Aos demais professores do CRTI, pelo companheirismo.

Aos colegas da Smar, por suportarem meu mau humor e pela tolerância.

Aos demais amigos, por entenderem que minha ausência é passageira e por serem um porto seguro nas horas mais solitárias.

À Plant, Page, Gilmour, Waters e demais monstros do rock, cujas notas me fizeram companhia nas madrugadas de silêncio e concentração.

*"Para se ter sucesso é necessário
amar de verdade o que se faz."
(Steve Jobs)*

Resumo

Há cerca de dois séculos foi iniciada a revolução industrial. O desenvolvimento de equipamentos como máquina a vapor e teares mecânicos transformou a forma com que o homem passou a tratar sua produção. Esta revolução perdura até os dias atuais, com paradigmas sendo quebrados com a introdução de novas tecnologias e métodos de fabricação. As linhas de manufatura sofreram uma grande mudança durante este período, passando de produção em massa no início do século XX até a produção enxuta, a partir da década de 1970. As tecnologias de fabricação também evoluíram e atualmente a modelagem e simulação de processos produtivos são etapas muito importantes na concepção de um sistema de manufatura. Além disso, busca-se na inteligência artificial e tomada de decisão formas de otimizar os processos, com objetivo de aumento de produtividade e principalmente redução de custos. Esta dissertação tem como base a modelagem e simulação de sistemas flexíveis de manufatura com aplicação da tomada de decisão de forma a maximizar a recompensa do sistema. Primeiramente é apresentada a rede de *Petri* colorida limitada como ferramenta de modelagem de sistemas a eventos discretos. A partir desta modelagem é possível determinar todos os estados atingíveis pelo sistema e apresentar os processos decisórios de *Markov*, ferramenta de tomada de decisão aplicada no trabalho. Para a resolução do MDP, além dos estados do sistema, são calculadas as recompensas a partir do processamento em servidores, as máquinas do sistema de manufatura. Finalmente, de posse das políticas calculadas no MDP, pode-se fazer a simulação de alguns modelos de *flowshop* e *jobshop*, que são duas classes de linhas de manufatura muito encontradas no ambiente industrial, e verificar a tomada de decisão em tempo real. O resultado das simulações permitiu a comparação de sistemas com diferentes concepções de recompensa e com heurísticas de produção manuais.

Palavras-chave: sistemas flexíveis de manufatura, redes de *Petri* coloridas limitadas, processos decisórios de *Markov*, tempo real.

Abstract

About two centuries ago it was started the industrial revolution. The development of equipment such as steam engine and mechanical looms transformed the way in which man has treated its production. This revolution endures to the present day, with paradigms being broken with the introduction of new technologies and manufacturing methods. Manufacturing lines have undergone a major change during this period, from mass production in the early twentieth century to the lean production from the 1970s. Manufacturing technologies have also evolved and now production processes modeling and simulation are very important steps in the design of manufacturing systems. Moreover, artificial intelligence and decision making methods are searched to optimize processes, in order to increase productivity and reduce costs. This thesis is based on the modeling and simulation of flexible manufacturing systems with decision making used to maximize the system's reward. First is presented the colored and limited *Petri* net as a modeling tool for discrete events systems. From this model it is possible to determine all system attainable states and present the *Markov* decision processes, the decision-making tool applied in this work. For the MDP resolution, the rewards are calculated from the processing servers that are the machines of the manufacturing system. Finally, simulations could be done for *flowshop* and *jobshop* models according to the MDP calculated policies, to verify the real-time decision making. *Flowshop* and *jobshop* are two flexible manufacturing system classes much found in the industrial environment. The results of the simulations allowed the comparison of systems with different rewards conceptions and manual production heuristics.

Keywords: flexible manufacturing systems, limited colored Petri nets, Markov decision processes, real-time.

Lista de ilustrações

Figura 1 – Rede de <i>Petri</i> com representação dos 4 elementos: lugares (círculos), transição (barra), arcos orientados (setas) e <i>token</i> (ponto)	18
Figura 2 – Regra de disparo: (a) marcação antes do disparo da transição habilitada t, (b) marcação após disparo de t, transição desabilitada	19
Figura 3 – Notação matricial: (a) rede de <i>Petri</i> , (b) matriz de entrada <i>E</i> , (c) matriz de saída <i>S</i> , (d) matriz de incidência <i>C</i>	22
Figura 4 – Representação do problema dos cinco filósofos: (a) rede de <i>Petri</i> , (b) rede de <i>Petri</i> colorida	24
Figura 5 – Exemplo de MDP com 3 estados: (a) representação em RdP, (b) diagrama de estados	29
Figura 6 – Iteração de valor para $\gamma = 0,8$	30
Figura 7 – Iteração de valor para $\gamma = 0,2$	31
Figura 8 – Arena da competição <i>FESTO Logistics League</i> [®] (FESTO, 2012)	33
Figura 9 – Diagrama de blocos de um sistema de produção <i>flowshop</i>	35
Figura 10 – Diagrama de blocos de um sistema de produção <i>jobshop</i>	36
Figura 11 – <i>Layout</i> de sistema <i>jobshop</i>	37
Figura 12 – RdP que modela sistema simples de produção	44
Figura 13 – Modelo em <i>SimEvents</i> [®]	44
Figura 14 – Árvore de alcançabilidade do problema dos cinco filósofos: (a) representação em RdP colorida, (b) árvore gerada por marcações equivalentes (HUBER et al., 1986).	48
Figura 15 – Construção da árvore de alcançabilidade para RdP colorida: (a) RdP completa, (b) RdP segregada para cor A, (c) RdP segregada para cor B, (d) árvore resultante	49
Figura 16 – Variação da quantidade de estados x capacidade do lugar	54
Figura 17 – Funções de utilidade para as máquinas do FMS	56
Figura 18 – RdP para demonstração da ações	57
Figura 19 – Grafo de estados de um sistema para transições relacionadas à mesma ação e suas recompensas	58
Figura 20 – Recompensas acumuladas para o estado <i>s</i> ₀ , mesma ação para eventos não controláveis	59
Figura 21 – Recompensas acumuladas para o estado <i>s</i> ₀ , ações separadas para eventos não controláveis	60

Figura 22	–Diagrama de blocos do exemplo de aplicação 1	61
Figura 23	–Rede de <i>Petri</i> e parâmetros do sistema (exemplo de aplicação 1)	62
Figura 24	–Grafo de estados do sistema (exemplo de aplicação 1)	62
Figura 25	–Modelo do exemplo de aplicação 1 em <i>SimEvents</i> [®]	64
Figura 26	–Diagrama de blocos do exemplo de aplicação 2	65
Figura 27	–Rede de <i>Petri</i> e parâmetros do sistema (exemplo de aplicação 2)	66
Figura 28	–Grafo de estados do sistema (exemplo de aplicação 2)	66
Figura 29	–FMS geral (<i>flowshop</i>)	73
Figura 30	–Máquina do FMS	74
Figura 31	–Diagrama de blocos (<i>flowshop</i> 1)	77
Figura 32	– <i>Flowshop</i> 1 representado por RdP	78
Figura 33	–Variação das recompensas totais (<i>flowshop</i> 1)	81
Figura 34	–Comportamento de filas (variação de γ)	82
Figura 35	–Comportamento de filas (variação de capacidade)	83
Figura 36	–Diagrama de blocos (<i>flowshop</i> 2)	84
Figura 37	– <i>Flowshop</i> 2 representado por RdP	85
Figura 38	–Variação das recompensas totais (<i>flowshop</i> 2)	85
Figura 39	–Diagrama de blocos (<i>flowshop</i> 3)	87
Figura 40	– <i>Flowshop</i> 3 representado por RdP	87
Figura 41	–Variação das recompensas totais (<i>flowshop</i> 3)	89
Figura 42	–Modelo do <i>jobshop</i> 1 em <i>SimEvents</i> [®]	90
Figura 43	–Diagrama de blocos (<i>jobshop</i> 1)	91
Figura 44	– <i>Jobshop</i> 1 representado por RdP	91
Figura 45	–Variação das recompensas totais (<i>jobshop</i> 1)	94
Figura 46	–Tipos produzidos para o <i>jobshop</i> 1 ($\gamma = 0,6$)	94
Figura 47	–Tipos produzidos para o <i>jobshop</i> 1 ($\gamma = 0,9$)	95
Figura 48	–Diagrama de blocos (<i>jobshop</i> 2)	96
Figura 49	–Modelagem em <i>SimEvents</i> [®] do exemplo <i>jobshop</i> 2	97
Figura 50	– <i>Jobshop</i> 2 representado por RdP	98
Figura 51	–Variação das recompensas totais (<i>jobshop</i> 2)	101
Figura 52	–Tipo de peças produzidas para máxima recompensa total (<i>jobshop</i> 2)	101
Figura 53	–Quantidade total produzida (<i>jobshop</i> 2)	102
Figura 54	–Comportamento das filas para modelo com recompensa na fonte	103
Figura 55	–Fila de 6 posições	109
Figura 56	–Fila de 3 posições para produto <i>A</i>	110
Figura 57	–Fila de 3 posições para produtos <i>A</i> e <i>B</i>	111

Lista de tabelas

Tabela 1	– Probabilidades de transições	30
Tabela 2	– Recompensas	30
Tabela 3	– $Q(s, a)$ para $\gamma = 0,8$	31
Tabela 4	– Solução do MDP para $\gamma = 0,8$	31
Tabela 5	– $Q(s, a)$ para $\gamma = 0,2$	32
Tabela 6	– Solução do MDP para $\gamma = 0,2$	32
Tabela 7	– Descrição das máquinas de <i>FESTO Logistics League</i> [®]	32
Tabela 8	– Equivalência de blocos do <i>SimEvents</i> [®] e elementos da RdP	43
Tabela 9	– Estados gerados por uma fila de três posições	52
Tabela 10	– Relação entre transições da RdP e ações do MDP, para eventos não controláveis representando ação “fazer nada”	57
Tabela 11	– Nova relação entre transições da RdP e ações do MDP, limitando-se a ação “fazer nada”	59
Tabela 12	– Estados do sistema (exemplo de aplicação 1)	61
Tabela 13	– Recompensas geradas pela transição $t4$	63
Tabela 14	– Distribuição de recompensas para todos os estados do sistema (exemplo 1)	63
Tabela 15	– Comparativo de políticas para o exemplo de aplicação 1	63
Tabela 16	– Comparativo de políticas para o exemplo de aplicação 1 com recompensas invertidas	65
Tabela 17	– Estados do sistema (exemplo de aplicação 2)	67
Tabela 18	– Distribuição de recompensas para todos os estados do sistema (exemplo 2)	68
Tabela 19	– Comparativo de políticas para o exemplo de aplicação 2	69
Tabela 20	– Comparativo de políticas para o exemplo de aplicação 2 com recompensas invertidas	70
Tabela 21	– Valores de γ	74
Tabela 22	– Tempos de produção e valor agregado (<i>flowshop</i> 1)	77
Tabela 23	– Definição dos elementos da RdP (<i>flowshops</i>)	78
Tabela 24	– Relação entre transições da RdP e ações do MDP (<i>flowshops</i>)	78
Tabela 25	– Heurística 1 para os <i>flowshops</i>	79
Tabela 26	– Heurística 2 para os <i>flowshops</i>	79
Tabela 27	– Recompensas totais (<i>flowshop</i> 1)	80

Tabela 28 – Tempos de produção e valor agregado (<i>flowshop 2</i>)	84
Tabela 29 – Recompensas totais (<i>flowshop 2</i>)	86
Tabela 30 – Tempos de produção e valor agregado (<i>flowshop 3</i>)	87
Tabela 31 – Recompensas totais (<i>flowshop 3</i>)	88
Tabela 32 – Tempos de produção e valor agregado (<i>jobshop 1</i>)	91
Tabela 33 – Definição dos elementos da RdP (<i>jobshop 1</i>)	92
Tabela 34 – Relação entre transições da RdP e ações do MDP (<i>jobshop 1</i>)	92
Tabela 35 – Heurística 1 para <i>jobshop 1</i>	92
Tabela 36 – Heurística 2 para <i>jobshop 1</i>	93
Tabela 37 – Recompensas totais (<i>jobshop 1</i>)	93
Tabela 38 – Tempos de produção e valor agregado (<i>jobshop 2</i>)	95
Tabela 39 – Definição dos elementos da RdP (<i>jobshop 2</i>)	96
Tabela 40 – Relação entre transições da RdP e ações do MDP (<i>flowshops</i>)	98
Tabela 41 – Heurística proposta para o <i>jobshop 2</i>	99
Tabela 42 – Recompensas totais (<i>jobshop 2</i>)	100

Sumário

1	Introdução	14
2	Revisão da Literatura	17
2.1	Redes de <i>Petri</i>	17
2.1.1	Propriedades das redes de <i>Petri</i>	19
2.1.2	Notação matricial das redes de <i>Petri</i>	21
2.1.3	Disparo de transições	21
2.2	Redes de <i>Petri</i> coloridas	22
2.3	Redes de <i>Petri</i> limitadas	25
2.4	Processos Decisórios de <i>Markov</i>	25
2.4.1	Solução do MDP	28
2.4.2	Exemplo de MDP	29
2.5	<i>FESTO Logistics League</i> [®]	32
2.6	Sistemas Flexíveis de Manufatura (FMS)	33
2.6.1	Classificação dos sistemas de produção	35
2.6.2	Eficiência versus flexibilidade e origem do FMS	36
2.7	Trabalhos relacionados	38
3	Modelagem do FMS	41
3.1	O <i>SimEvents</i> [®]	41
3.1.1	<i>SimEvents</i> [®] e redes de <i>Petri</i>	42
3.2	Redes de <i>Petri</i> e processos de decisão de <i>Markov</i>	44
3.2.1	Modelagem da RdP e construção da árvore de alcançabilidade	45
3.2.2	Filas FIFO (<i>First-In-First-Out</i>)	50
3.2.3	Cálculo da quantidade de estados do sistema	51
3.2.4	Função de utilidade	53
3.2.5	Geração das matrizes de estados e recompensa do MDP	55
3.3	Exemplos de aplicação	61
3.3.1	Exemplo 1: Sistema FIFO/servidor, fila de 1 posição, peças A e B	61
3.3.2	Exemplo 2: Sistema FIFO/servidor, fila de 2 posições, peças A e B	65
4	Simulações e resultados	71
4.1	Ambiente e métodos de simulação	71
4.2	Considerações sobre recompensas	75
4.3	Simulações do comportamento em tempo real dos FMS	76
4.3.1	Simulação modelo <i>flowshop</i> 1	77

4.3.2	Simulação modelo <i>flowshop</i> 2	84
4.3.3	Simulação modelo <i>flowshop</i> 3	87
4.3.4	Simulação modelo <i>jobshop</i> 1	89
4.3.5	Simulação modelo <i>jobshop</i> 2	95
5	Conclusões e Trabalhos Futuros	104
	Referências	106
APÊNDICE A	Definição da expressão de cálculo da quantidade de estados do sistema	109

1 Introdução

A revolução industrial iniciada no século XVIII caracterizou-se pelo desenvolvimento da técnica de produção industrial e comunicações. As principais invenções que trouxeram impacto econômico foram as máquinas a vapor e teares mecânicos, que permitiram a implantação de manufaturas e usinas, o que abalou o modelo de produção vigente, baseado em exploração artesanal de oficinas familiares (LAROUSSE, 1998). Nestes quase dois séculos, o desenvolvimento de novas tecnologias e paradigmas fez com que a revolução industrial se transformasse continuamente, sendo dividida em quatro fases, tendo a terceira e quarta fases forte influência no desenvolvimento deste trabalho.

A terceira fase, ocorrida no período entre guerras (LAROUSSE, 1998) traz como grandes expoentes Henry Ford e Frederick Taylor e a criação do *fordismo*. Ford pregava que a eficiência do sistema de manufatura em massa (aquele em que a produção é feita em altos volumes estandardizados) encontra-se na economia de tempo mediante aumento de velocidade com que os materiais são trabalhados (COSTA, 2000). Além disso, Taylor argumentava que os tempos e movimentos que um trabalhador utiliza na execução de uma tarefa podem ser estudados e otimizados. O conceito de linha de produção (*flowline*) surge como grande inovação no âmbito do chão de fábrica, cujo princípio é a alteração do *layout* de máquinas de forma que estas sejam dispostas na sequência das operações de fabricação de certo produto. As consequências deste modelo são a redução do tempo de deslocamento de material e a criação de uma nova classe de trabalhadores, altamente especializados em uma única tarefa. O sistema proposto por Ford e Taylor pode ser visto de maneira cômica no filme “Tempos Modernos” (CHAPLIN, 1936).

A produção em massa, no entanto, gera um elevado grau de desperdício (superprodução, estoque e produção de produtos defeituosos (COSTA, 2000)), e seu declínio está ligado em partes a isto. A crise do petróleo na década de 1970 levou várias economias à recessão. Apesar disso, a empresa japonesa Toyota passou por este período sem os problemas encontrados no resto do mundo graças a um novo modelo produtivo que estava em desenvolvimento desde o fim da segunda guerra mundial. O sistema Toyota de produção, ou *toyotismo*, foi concebido para eliminar absolutamente o desperdício, diminuindo custos e produzindo apenas o necessário (BATISTA, 2008). Este sistema está embasado em dois pilares básicos, o *Just-in-Time* e *automação* (junção das palavras autonomia e automação).

A chamada quarta fase da revolução industrial tem início na década de 1960 e é caracterizada pela introdução da automação e robótica em vários setores industriais

(LAROUSSE, 1998). Passados cerca de cinquenta anos, as tecnologias da automação industrial muito evoluíram com o desenvolvimento de sistemas mais complexos, confiáveis e inteligentes. E é com o advento da inteligência artificial aplicada à indústria que surge um novo conceito de modelo de produção, os Sistemas Flexíveis de Manufatura (*Flexible Manufacturing Systems* ou FMS). Se desde o *fordismo* já havia uma preocupação sobre movimentação de materiais e arranjo de máquinas, os FMS incorporam um novo elemento ao ambiente de chão de fábrica: um sistema de tomada de decisão (SILVA; VALETTE, 1990).

A motivação deste trabalho vem do desafio proposto pela FESTO *Logistics League*[®] 2012 (FESTO, 2012). O objetivo da competição é desenvolver um sistema capaz de organizar unidades robóticas num *layout* pré-determinado de máquinas, a fim de se obter uma solução flexível e de alto desempenho para fluxo de material e informação através de uma linha de produção industrial. Nesta proposta, os robôs são os agentes inteligentes, uma vez que estes decidem qual tarefa executar e quando executar. Um conjunto de restrições deve ser obedecido, já que uma linha de produção é um ambiente organizado, definido por rotas e tempos. Além disso, mais que apenas um sistema automático de transporte de produtos, os robôs são obrigados a trabalhar com paralelismo de máquinas (PINEDO, 2012) e tomada de decisão (BASTOS, 2010; LITTMAN; DEAN; KAELBLING, 1995; PELLEGRINI; WAINER, 2007) a partir de dados obtidos por sensores.

De forma a simplificar o problema e partir para outra abordagem sobre inteligência, este estudo considera linhas de produção sequenciais, sem paralelismo de máquinas e/ou transporte automático de peças. A característica sequencial do problema faz com que as linhas de produção sejam interpretadas como sistemas a eventos discretos (SILVA; VALETTE, 1990), ou seja, a mudança de estados do sistema não é contínua. Por isso, são definidos como sistemas dinâmicos com espaço de estados discretos e instantes de transição imprevisíveis (RAMADGE; WONHAM, 1989). Pode-se definir o estado de um sistema de manufatura como uma fotografia da linha de produção tirada em determinado instante. A observação destes estados torna possível que um agente tome uma decisão, interferindo ou não no sistema, de forma a otimizar os parâmetros de produção.

O agente tomador de decisão escolhido como objeto deste estudo é o processo de decisão de *Markov* (MDP) (BASTOS, 2010; PELLEGRINI; WAINER, 2007; LITTMAN; DEAN; KAELBLING, 1995), que através da observação dos estados do sistema, executa ações baseado em valores de recompensas. Assim, com base na observação do sistema de manufatura, é esperado que o MDP forneça um mapa de qual melhor ação tomar em cada estado atingível pelo sistema. No entanto, antes de aplicar o MDP neste problema, é necessário que as linhas de manufatura sejam modeladas de uma forma que se consiga boa representação do modelo e seja possível a determinação dos estados atingíveis pelo sistema.

Uma ferramenta que é muito usada para a análise de FMS (VEUNG et al., 1996) e apresenta estas características é a rede de *Petri* (RdP) (PETRI, 1962; MURATA, 1989), visto que é uma ferramenta gráfica intuitiva que possibilita a representação de sistemas dinâmicos a eventos discretos. Além disso, possuem uma representação matemática simples, porém poderosa, o que possibilita o desenvolvimento de algoritmos computacionais que facilitam a análise e execução do sistema. Outra vantagem em utilizar as redes de *Petri* é que ao determinar sua árvore de alcançabilidade, consegue-se automaticamente determinar os estados da cadeia de *Markov* associada, pois a dinâmica de uma rede de *Petri* depende apenas do estado atual e do evento executado. Os eventos podem ser classificados em eventos controláveis (aqueles que sofrem influência direta de um agente externo) e eventos não controláveis (aqueles que não são influenciados externamente).

As redes de *Petri* já vêm sendo utilizadas há um bom tempo em representação de processos industriais. Enquanto o trabalho de Agerwala (1979) fundamenta os conceitos de redes de *Petri* presentes na época e exemplifica as propriedades para modelos industriais, Aalst (1994) trata do mesmo assunto considerando um novo desmembramento da ferramenta: redes de *Petri* de alto nível. Entre as redes de *Petri* de alto nível, um dos expoentes é a rede de *Petri* colorida (JENSEN, 1997), cuja definição é aplicável ao conceito de linhas de manufatura, pois modelam problemas que envolvem tipos de entidades (tipos de peças). Assim podem ser representados diferentes tipos de marcas e, com isso, a representação da rede fica mais enxuta.

O escopo deste trabalho é a modelagem de diferentes tipos de linhas de produção, determinação matemática do modelo, execução do algoritmo de análise dinâmica da rede, cálculo do MDP e simulação dos resultados. Através das redes de *Petri*, é possível elaborar um algoritmo de determinação dos estados desta linha de manufatura e geração dos parâmetros de entrada do cálculo do MDP. Tendo sido resolvido o MDP, pode-se fazer uma comparação entre o comportamento do FMS a partir da tomada de decisão e um comportamento qualquer, aleatório ou baseado em heurística. Um comportamento aleatório pode ser definido pela escolha não deliberada de quais ações devem ser executadas, enquanto a heurística é a tomada de decisão baseada em conhecimento empírico, pré-definido. Após isso, são feitas simulações que visam determinar a produção em tempo real da linha e verificar qual resultado se mostrou mais satisfatório.

De forma a melhor entender os componentes utilizados neste trabalho, o capítulo 2 apresenta a revisão bibliográfica, onde são definidas redes de *Petri*, redes de *Petri* coloridas, redes de *Petri* limitadas, processos decisórios de *Markov* e é apresentado o estado da arte sobre linhas de produção industrial e sistemas flexíveis de manufatura. O capítulo 3 trata da modelagem do problema, definindo a execução sequencial de atividades e ferramentas empregadas. As simulações para FMS e os resultados obtidos são mostrados no capítulo 4, enquanto o capítulo 5 contém a conclusão e sugestão de trabalhos futuros.

2 Revisão da Literatura

2.1 Redes de *Petri*

Inicialmente definida no trabalho *Kommunikation mit Automaten* (PETRI, 1962), as redes de *Petri* são uma ferramenta gráfica e matemática aplicável a vários sistemas (MURATA, 1989). Após o término do trabalho de Carl Adam Petri, na Alemanha, o conceito de Redes de *Petri* (RdP) chamou a atenção de vários cientistas e pesquisadores nos Estados Unidos e outros países da Europa, levando a um desenvolvimento bastante grande desta ferramenta, principalmente entre as décadas de 1970 e 1980, com o surgimento de modelos e definições mais complexas, como as Redes de *Petri* de Alto Nível e Redes de *Petri* Inteligentes. Entre as Redes de *Petri* de Alto Nível, podem ser citadas as Redes de *Petri* coloridas, que são também objeto de estudo deste trabalho.

As RdPs podem ser usadas para modelagem, simulação e estudo de sistemas caracterizados como concorrentes, assíncronos, distribuídos, paralelos, não determinísticos, e/ou estocásticos, tendo sido aplicada com sucesso em áreas como protocolos de comunicação, avaliação de desempenho, sistemas distribuídos, sistemas paralelos, sistemas de controle industrial e manufatura flexível, entre outros (MURATA, 1989).

Inicialmente será apresentada a definição da RdP, um conceito preliminar que será estendido para a definição de RdPs coloridas, que será a ferramenta de modelagem utilizada para os sistemas de manufatura apresentados, devido sua flexibilidade em se trabalhar com diferentes tipos de dados, o que torna o modelo do sistema muito mais enxuto. Generaliza-se simplesmente por redes de *Petri* aquelas que não são de alto nível, como uma versão básica da ferramenta. Estas poderiam ser chamadas RdPs ordinárias, onde os pesos dos arcos tem valor igual a um, porém, por conveniência, este termo é omitido.

Informalmente, as RdPs são grafos direcionados, bipartidos, compostos basicamente por quatro tipos de elementos: lugares, transições, marcas (*tokens*) e arcos orientados. Gráficamente, os lugares são representados por círculos, transições são representadas por uma barra (quadrados e retângulos também são encontrados em algumas literaturas), *tokens* são representados por pontos e os arcos são setas, ligando lugares às transições e transições aos lugares. A figura 1 mostra uma RdP simples, com seus quatro elementos.

Definição formal: a RdP é a quintupla $R = (P, T, F, W, M_0)$, onde:

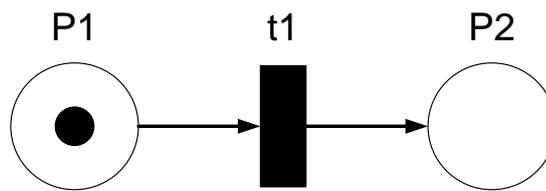


Figura 1 – Rede de *Petri* com representação dos 4 elementos: lugares (círculos), transição (barra), arcos orientados (setas) e *token* (ponto)

- $P = \{p_1, p_2, \dots, p_n\}$ é um conjunto finito de lugares;
- $T = \{t_1, t_2, \dots, t_m\}$ é um conjunto finito de transições;
- $F \subseteq (P \times T) \cup (T \times P)$ é o conjunto dos arcos orientados;
- $W : F \rightarrow \{1, 2, 3, \dots\}$ é a função peso do arco;
- $M_0 : P \rightarrow \{0, 1, 2, 3, \dots\}$ é a marcação inicial da rede;
- $P \cap T = \emptyset$ e $P \cup T \neq \emptyset$.

De um ponto de vista real, os lugares representam os recursos, elementos que sofrem ação do sistema (por exemplo, peça a ser processada por uma máquina num sistema de manufatura, ou um pacote a ser enviado num sistema de comunicação). As transições são os elementos ativos do sistema ou os eventos que ocorrem no decorrer do processo (início de processamento de peça ou indicação de recebimento de uma mensagem). Os arcos orientados definem as relações entre estes elementos e os *tokens* representam a quantidade de recursos disponíveis em cada lugar. Uma marcação é um vetor que relaciona cada lugar da rede com a quantidade de *tokens* em cada um deles. Uma RdP tem representação gráfica se, e somente se, os arcos orientados ligam lugares às transições e transições aos lugares (relação de fluxo).

Os lugares de entrada de uma transição são conhecidos como lugares de pré-condição enquanto os lugares de saída de uma transição são conhecidos como lugares de pós-condição. A dinâmica do sistema é descrita pelo disparo das transições, isto é, caso uma transição esteja habilitada, esta retira a quantidade de *tokens* estabelecida pelo peso dos arcos que ligam seus lugares pré-condição e deposita nos lugares pós-condição a quantidade de *tokens* estabelecida nos arcos que ligam a transição a eles.

Murata (1989) exemplifica esta relação de fluxo modelando a reação química $2H_2 + O_2 \rightarrow 2H_2O$ através da figura 2. Neste exemplo, caso apenas uma molécula de H_2 esteja disponível, a transição que realiza a reação química está desabilitada e não é disparada.

Pode-se representar o conjunto de arcos por duas funções, $I(t)$ e $O(t)$, que identificam respectivamente os lugares de entrada ($P \rightarrow T$) e os lugares de saída ($T \rightarrow P$) de

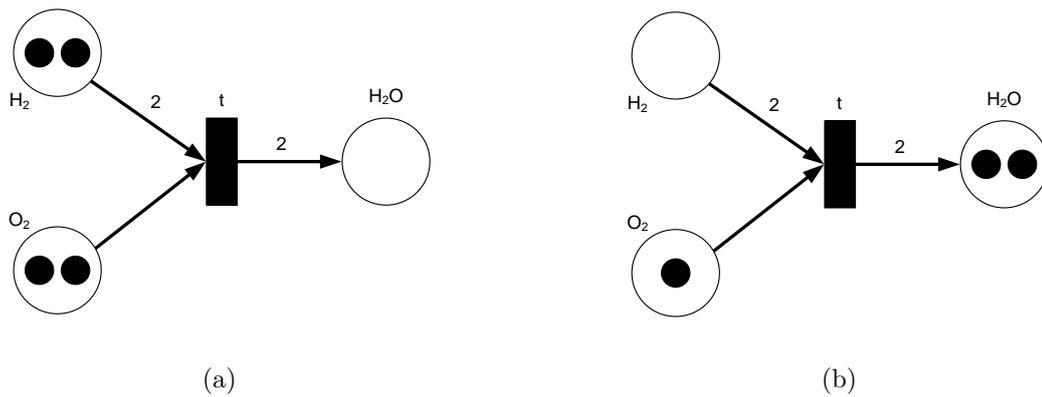


Figura 2 – Regra de disparo: (a) marcação antes do disparo da transição habilitada t , (b) marcação após disparo de t , transição desabilitada

uma transição. Com esta divisão é possível estabelecer o conceito de multiplicidade que define o número de ocorrências de lugares nos subconjuntos $I(t_j)$ e $O(t_j)$. A multiplicidade é definida como:

- $\#(p_i, I(t_j)) \geq 0$, multiplicidade do lugar p_i no subconjunto $I(t_j)$, também chamado peso de entrada do arco que conecta p_i a t_j ;
- $\#(p_i, O(t_j)) \geq 0$, multiplicidade do lugar p_i no subconjunto $O(t_j)$, também chamado peso de saída do arco que conecta t_j a p_i .

Com esta definição pode-se demonstrar matematicamente que uma transição t_j está habilitada através da relação:

$$\forall p_i \in I(t_j) \rightarrow M(p_i) \geq \#(p_i, I(t_j)) \quad (2.1)$$

onde $M(p_i)$ é a marcação do lugar p_i .

Em palavras, esta expressão define que uma transição t_j qualquer está habilitada apenas se a marcação de todos os lugares pré-condição p_i é maior ou igual que a multiplicidade de p_i no conjunto $I(t_j)$.

2.1.1 Propriedades das redes de *Petri*

Para fins de análise de uma RdP, existem dois tipos de propriedades que podem ser estudadas a partir de um modelo: propriedades dependentes da marcação e propriedades independentes da marcação. As propriedades independentes da marcação são estruturais e não são muito difundidas na literatura. No entanto, as propriedades dependentes da marcação são de relevante importância no que diz respeito à dinâmica do sistema. Algumas destas propriedades são (MURATA, 1989; BRESSAN, 2002):

- *Alcançabilidade*: é a propriedade fundamental para análise dinâmica de uma RdP. Uma sequência de disparos de transições irá gerar marcações de acordo com as regras de habilitação e disparo de transições vistas anteriormente. Assim, para uma análise em longo prazo, num horizonte infinito ou finito com determinado número de disparos, é importante verificar se um estado M_n (desejável ou mesmo indesejável) é atingível a partir da marcação inicial M_0 . Esta verificação pode ser feita de forma matemática através de uma equação algébrica ou gráfica através da árvore de alcançabilidade (ou de cobertura). A árvore de alcançabilidade é um elemento de destaque neste trabalho, visto que através dela é possível determinar todos os estados possíveis do sistema, o que é importante para o cálculo das políticas do MDP.
- *Limitação*: um lugar p_i é dito k limitado se para toda marcação M' pertencente ao conjunto de marcações alcançáveis, $M'(p_i) \leq k$. Uma RdP é limitada se todos os lugares da rede são limitados.
- *Segurança*: um lugar p_i é dito seguro se para toda marcação M' pertencente ao conjunto de marcações alcançáveis, $M'(p_i) \leq 1$. Uma RdP é segura se todos os lugares da rede são seguros. As propriedades de limitação e segurança são importantes para garantir que não haverá aumento descontrolado de *tokens* em algum dos lugares da rede (*token overflow*).
- *Conservação*: a propriedade de conservação indica que para qualquer estado atingível na RdP, a soma de todas as marcações se mantém constante.
- *Impasse (bloqueio mortal)*: dada uma rede de *Petri*, existe a situação de impasse ou bloqueio mortal, se para um subconjunto T' de transições desta rede, para uma marcação qualquer M' , alguma transição t_j pertencente a T' está morta. Se o subconjunto T' é igual ao conjunto T de todas as transições desta rede, existe a situação de impasse total, nenhuma transição pode ser disparada e a toda a rede é considerada morta.

Este trabalho trata da modelagem de linhas de manufatura, sendo necessária a introdução do conceito de filas, depósitos intermediários (*buffers*), capacidade de máquinas, etc. Assim, será abordado mais adiante o conceito de RdP limitada, o que torna alguma destas propriedades intuitivas apenas pela análise da capacidade individual de cada lugar da rede. Além disso, fora a regra de habilitação e disparo descrita anteriormente, é possível criar mecanismos de restrição de disparo baseados na capacidade dos lugares. Isto foi feito no algoritmo principal desenvolvido para este trabalho, de forma que uma transição habilitada não seja disparada enquanto uma máquina esteja ocupada ou uma fila em depósito intermediário esteja cheia, por exemplo.

2.1.2 Notação matricial das redes de *Petri*

Além de uma ferramenta gráfica, as RdPs podem ser representadas matematicamente em forma de matrizes. Assim é possível a determinação de algoritmos de análise e avaliação de desempenho. As matrizes que descrevem uma RdP são determinadas substituindo as funções $I(t)$ e $O(t)$ por uma matriz que relaciona as transições e seus lugares pré-condições em uma matriz de entrada E e as transições e seus lugares pós-condições em uma matriz de saída S .

Para uma rede com n transições e m lugares, essas matrizes serão de dimensão $n \times m$, e estas serão preenchidas com o valor do peso do arco que liga um lugar p_i a uma transição t_j (no caso da matriz E), ou o peso do arco que liga uma transição t_j ao lugar p_i (no caso da matriz de S). Para as posições da matriz onde não há interação qualquer entre o lugar e a transição em questão, o valor a ser considerado é zero.

De forma a condensar as informações das matrizes E e S em uma única matriz, existe a matriz de incidência (matriz C) que é definida por:

$$C = S - E \quad (2.2)$$

A matriz de incidência é importante no algoritmo, pois é componente obrigatório para a determinação do próximo estado de uma rede de *Petri*, dadas a marcação atual e transições habilitadas. Uma consideração importante acerca da matriz de incidência é que, apesar desta conter o comportamento dinâmico imediato da rede, a determinação direta desta sem as matrizes E e S pode ocasionar perda de informação. Devido a este fato, toda análise feita no algoritmo principal foi realizada levando em consideração as matrizes E e S . A figura 3 mostra um exemplo de RdP com suas matrizes de entrada E , saída S e incidência C .

2.1.3 Disparo de transições

Sejam M a marcação atual da rede, M' a próxima marcação, C a matriz de incidência e e_j um vetor de n componentes que indica qual transição está habilitada (BREISSAN, 2002). Para t_j habilitada, tem-se que o j -ésimo componente de e_j é 1, os demais são zero. Assim, o próximo estado da rede de *Petri* é definido pela função:

$$M' = M + e_j * C \quad (2.3)$$

Com base nas regras de habilitação de transições, representação matricial da rede de *Petri* pelas matrizes de entrada e saída E e S e a função próximo estado da rede de *Petri* é possível construir um algoritmo que explorará a RdP de forma a gerar sua árvore de alcançabilidade. Isto se faz necessário porque, uma vez definida a árvore de alcançabilidade

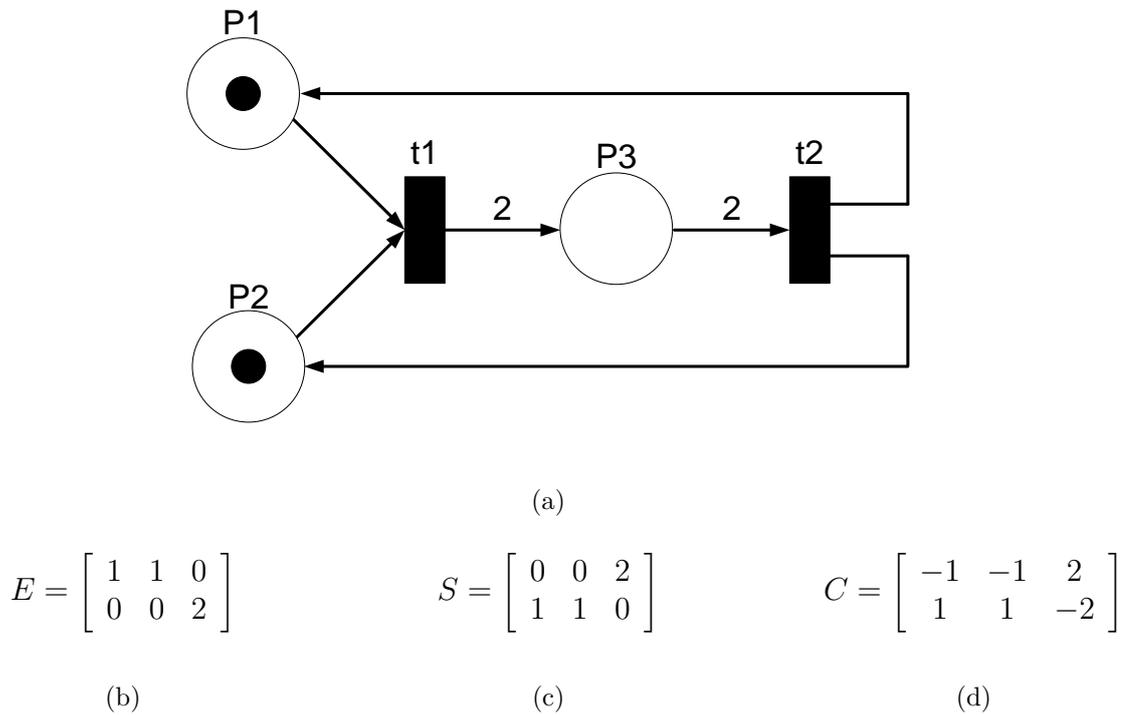


Figura 3 – Notação matricial: (a) rede de *Petri*, (b) matriz de entrada E , (c) matriz de saída S , (d) matriz de incidência C

de uma rede, cada marcação da árvore corresponde a um estado da cadeia de *Markov* associada (SALES, 2002; BRESSAN, 2002; CARDOSO; VALETTE, 1997; MATOS et al., 2011), de forma que este conjunto de estados será base para a geração da matriz de estados do MDP associado a um problema particular.

2.2 Redes de *Petri* coloridas

Conforme apresentado, as redes de *Petri* são uma ferramenta muito poderosa para modelagem de sistemas discretos, tanto de forma gráfica quanto algébrica. No entanto, este primeiro modelo de rede de *Petri* tem uma fraqueza no que diz respeito à representação de *tokens*. Até este momento um *token* é indistinguível por si só, e para representação de diversos tipos de dados em uma rede é necessária a criação de muitos outros estados e transições. Este é um problema evidente nas linhas de manufatura, pois numa mesma linha diversos tipos de produtos podem ser processados. Além disso, a quantidade de estados de uma rede de *Petri* aumenta exponencialmente adicionando lugares na sua representação. Desta forma, é necessário um esforço computacional muito grande para o processamento dos algoritmos de análise. Como exemplo, para uma rede com 10 lugares para representação de 10 tipos diferentes de *tokens*, existe um total de 2^{10} estados possíveis (1024 estados).

No entanto, na década de 1970, surgem as redes de *Petri* de alto nível (FERNÁNDES, 2010) para as quais os *tokens* deixam de ser apenas uma unidade de certo recurso

e passam a ser considerados objetos complexos, que representam desde o estado de um processo (marcação da rede) até dados contendo diversos tipos de informação (partes de uma mensagem em protocolo de comunicação – (JENSEN, 1997)). Um dos expoentes destas novas redes de *Petri* de alto nível é a rede de *Petri* colorida (ou CPN segundo a literatura).

Proposta inicialmente por Kurt Jensen, da Universidade de *Århus*, Dinamarca, as redes de *Petri* coloridas permitem que cada *token* seja individualizado, cada um representando um tipo diferente de recurso, porém pertencentes ao mesmo grupo. Isto é, um lugar pode conter diversos *tokens*, cada um de um tipo diferente, uma cor diferente. O nome rede de *Petri* colorida vem justamente desta capacidade de distinguir vários tipos de *tokens* (JENSEN, 1997): enquanto na rede de *Petri* ordinária existe apenas um tipo de *token* representado por um ponto preto, sem cor, nas redes de *Petri* coloridas pode-se atribuir uma cor para cada tipo de *token*.

Do ponto de vista gráfico, a rede de *Petri* colorida continua sendo representada por lugares (círculos), transições (barras) e arcos orientados (setas). Nos lugares podem ser encontrados *tokens* de várias cores diferentes, porém de um mesmo conjunto (*color set*). Um exemplo de lugar e seu conjunto de cores é a representação de uma máquina numa linha de produção que pode processar três tipos de peça (peças A, B e C). Logo, este lugar pode conter *tokens* do conjunto PEÇAS, com cores A, B e C.

Devido ao fato de os *tokens* carregarem o mais variado tipo de informação, outro avanço em relação às redes de *Petri* ordinárias é a presença de expressões de arco – as expressões de guarda. Enquanto na rede de *Petri* os arcos têm apenas pesos, representando a quantidade de *tokens* necessárias para habilitar uma transição ou quantidade de *tokens* gerada após algum disparo, na rede de *Petri* colorida é possível que uma expressão gerencie as transições, dando mais flexibilidade e poder de representação em um modelo mais simplificado. Em uma analogia feita pelo próprio professor Jensen, enquanto a rede de *Petri* pode ser comparada a uma linguagem de baixo nível (*assembly*) devido ao poder de representação limitado a um tipo de token, as redes de *Petri* coloridas são comparadas a linguagens orientadas a objetos, com possibilidade de representação de diversos tipos de dados e habilitação de transições através de expressões.

Como outro exemplo do poder de representação da rede de *Petri* colorida de forma simplificada, a figura 4 mostra o clássico problema dos cinco filósofos modelado em rede de *Petri* e em rede de *Petri* colorida. Enquanto são necessários quinze estados e dez transições para representação deste problema em RdP, na RdP colorida são necessários apenas três lugares e duas transições. No segundo caso, ao invés de cada filósofo e cada palito ser representado por um lugar distinto, existe apenas um lugar um representando o conjunto de cores FILO (filósofos) e outro contendo CS (palitos – ou *chopsticks*). Cada *token* presente nestes lugares e pertencentes a seu respectivo conjunto de cor representa

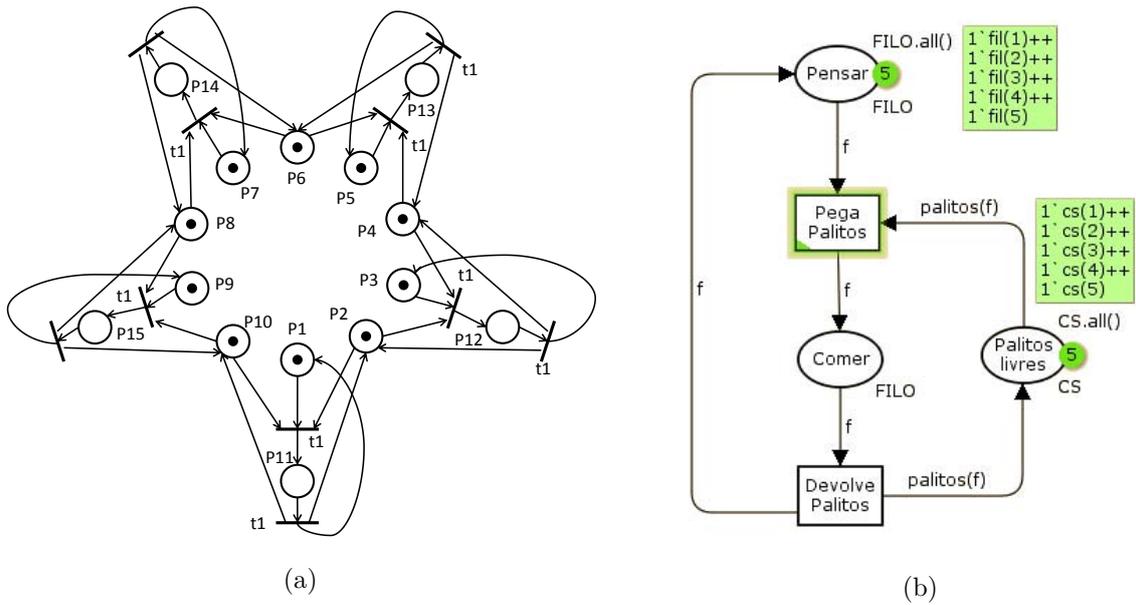


Figura 4 – Representação do problema dos cinco filósofos: (a) rede de Petri, (b) rede de Petri colorida

um filósofo ou um palito. No caso das transições tem-se uma amostra do poder de representação da RdP colorida, devido a presença de uma expressão (*palitos(f)*) no arco entre o lugar “Palitos livres” e a transição “Pega palitos” e entre a transição “Devolve palitos” e o lugar “Palitos livres”. Esta expressão garante que a transição “Pega palitos” seja habilitada apenas se estiverem livres os palitos à direita e à esquerda de um filósofo que esteja pensando. A expressão também faz com que sejam liberados apenas os palitos à direita e à esquerda de um filósofo que terminou de comer, caso seja disparada a transição “Devolve palitos”. No ambiente de modelagem de RdP *CPN tools* (JENSEN; KRISTENSEN; WELLS, 2007), a função *palitos(f)* é definida pela expressão 2.4.

$$fun_palitos(filo(i)) = 1'cs(i) + +1'cs(if i = n then 1 else i + 1) \quad (2.4)$$

O tema rede de Petri coloridas vem ganhando cada vez mais notoriedade com o surgimento de ferramentas poderosas de análise e simulação e a utilização cada vez mais recorrente em problemas reais de grande porte. A teoria decorrente deste desenvolvimento também se tornou bastante extensa, levando em conta particularidades como inserção de tempo nos lugares/transições (JENSEN; KRISTENSEN; WELLS, 2007), hierarquia de redes e sub-redes e análise de invariantes (JENSEN, 1997). No entanto, o uso das redes de Petri coloridas neste trabalho é limitado à representação dos modelos de linhas de produção, de forma a tornar o desenvolvimento mais enxuto. Portanto, outros aspectos ligados às redes de Petri coloridas não serão abordados.

2.3 Redes de *Petri* limitadas

As redes de *Petri* podem também ser classificadas de acordo com a quantidade de *tokens* que cada lugar pode receber, podendo ser de capacidade finita ou infinita (MURATA, 1989). Afirmar que um lugar tem capacidade infinita é interessante para analisar o comportamento de um processo ao longo de um espaço de tempo suficientemente grande, porém não é factível de se assumir num sistema real. Além disso, os lugares representam recursos, condições, e na modelagem de um sistema é importante considerar um limite para estes.

Assim, e tendo como base a propriedade de limitação e segurança, é muito conveniente considerar que um lugar tenha capacidade k , ou seja, possa receber no máximo uma quantidade k de *tokens*, sendo k um número inteiro não negativo. Também é muito conveniente que existam redes onde k seja um para todos os lugares. Portanto, existem duas classificações das redes de *Petri* que podem ser adotadas (EBOLI, 2010):

- *redes condição – evento*: são as redes onde $\forall p \in P, k = 1$;
- *redes lugar – transição*: são as redes onde $\forall p \in P, k > 1$.

No caso de linhas de manufatura é importante definir restrições quanto ao uso das máquinas, de forma que estas apresentem o processamento de apenas uma peça por vez. Assim, a modelagem de uma linha de manufatura poderia ser expressada apenas pela entrada de recursos (transições fontes), saída de recursos (transições dreno) e as máquinas (lugares com capacidade $k = 1$).

No entanto, este não seria um modelo fiel ao real, já que muitos depósitos intermediários (*buffers*) com capacidade $k > 1$ são encontrados neste tipo de aplicação. Desta forma, faz-se necessária a apresentação de uma variação de rede de *Petri*, que nada mais é a aplicação da propriedade de limitação: redes de *Petri* limitadas (BATA et al., 2008).

$$\exists k \in \mathbb{N} \forall M : M(p) \leq k \quad (2.5)$$

Esta definição é importante para o algoritmo de definição de estados da rede, pois um vetor K contendo a capacidade de cada lugar é utilizado para definir os *buffers*.

2.4 Processos Decisórios de *Markov*

Tendo sido definidos os conceitos de redes de *Petri*, redes de *Petri* coloridas e redes de *Petri* limitadas, tem-se a base para a modelagem e análise de linhas de manufatura, bem como os requisitos para geração do algoritmo de determinação dos estados do sistema.

No entanto, a análise principal deste trabalho não está completa sem outro componente essencial proposto, que é a utilização de um agente tomador de decisões no sistema.

Mais que uma estratégia de programação de produção (*scheduling*) (PINEDO, 2012; REIS, 1996), a presença de um agente tomador de decisões é uma ferramenta usada para se otimizar alguma variável determinada no início da modelagem do sistema. Esta recompensa tem relação com algum tipo de variável do sistema, desde maximizar o valor agregado do que for produzido até minimizar o tempo total de produção. De forma a analisar o sistema e tomar decisões de qual ação executar num determinado estado, foi usada a técnica dos Processos Decisórios de *Markov* (*Markov Decision Processes* – MDP) (PELLGRINI; WAINER, 2007; BASTOS, 2010; LITTMAN; DEAN; KAELBLING, 1995).

Seja considerado o seguinte problema (BASTOS, 2010): um médico recebe uma série de pacientes que acabaram de sofrer um acidente, cada um com grau diferente de seriedade. O médico tem que atender os três de forma que a sequência de cada paciente atendido e as ações tomadas para cada um deles maximizem a probabilidade de que suas vidas sejam salvas. Por outro lado, não há a certeza que a ação tomada pelo médico vai influenciar o paciente da melhor forma possível, apenas após algum tempo o paciente dará uma resposta ao tratamento.

Neste caso, compreende-se que este problema é sequencial e contém certas incertezas. O médico é o agente tomador de decisões que analisa o estado do sistema (condição dos pacientes) e age de forma sequencial (um paciente por vez), executando as ações que trarão melhor recompensa.

O MDP é uma forma de modelagem de sistemas onde as transições são probabilísticas, os estados são observáveis e é possível que um agente tomador de decisão interfira no sistema através de ações. A definição formal de um MDP é a tupla (S, A, T, R) :

- S é o conjunto finito de estados do sistema;
- A é o conjunto finito de ações disponíveis ao agente tomador de decisão;
- $T : S \times A \times S \rightarrow [0, 1]$ é a função que dá a probabilidade de o sistema passar para um estado $s' \in S$, a partir de um estado $s \in S$, se tomada uma ação $a \in A$. É denotada por $T(s'|s, a)$;
- $R : S \times A \rightarrow \mathfrak{R}$ é uma função que dá a recompensa por tomar uma decisão $a \in A$ quando o processo se encontra em um estado $s \in S$.

Os MDPs tem esse nome, pois seguem a propriedade de *Markov* (BAUSE; KRITZINGER, 2002): o efeito de uma ação depende apenas da ação tomada e do estado onde o sistema se encontra. Isto é, não depende do histórico de ações nem de como o sistema chegou até o estado atual.

O agente observador verifica em qual estado o sistema está e a partir disso decide qual ação deve ser tomada. O intervalo de tempo em que as decisões são tomadas é chamado de época de decisão e o mapeamento das ações para todos os estados do sistema, segundo a função $R : S \times A$, é a política π . O conceito de épocas de decisão aplicadas neste trabalho será definido no capítulo de simulações. Além disso, é possível calcular a recompensa total esperada de uma determinada ação, também conhecida como utilidade (BASTOS et al., 2011). A recompensa de uma ação varia de acordo com a variável de processo que se espera otimizar. Neste estudo é esperado que o sistema maximize a recompensa total de produção.

Isto significa que maior será o valor da recompensa quanto menor seja o tempo de produção de um tipo de peça em cada uma das máquinas e quanto maior seja um fator de valor agregado para este tipo. As funções de utilidade utilizadas neste trabalho são apresentadas no capítulo sobre modelagem dos sistemas de manufatura.

A recompensa total esperada é calculada por:

$$E \left[\sum_{k=0}^{z-1} r_k \right] \quad (2.6)$$

onde r é a recompensa imediata e z é o horizonte finito. Horizonte de um MDP é o número de épocas de decisão disponíveis para a tomada de decisões, sendo finito quando há um número de fixo de decisões a tomar (PELLEGRINI; WAINER, 2007).

Pode ser definida outra forma para o cálculo da recompensa total: a recompensa total descontada. Neste caso é usado um fator de desconto $\gamma \in]0, 1[$ que define a importância das decisões tomadas em épocas futuras. O valor zero não dá importância para recompensas futuras (comportamento guloso – *greedy behavior*) (BASTOS, 2010), enquanto o valor um não gera descontos na recompensa total acumulada. Seu cálculo para horizonte finito é definido por:

$$E \left[\sum_{k=0}^{z-1} \gamma^k r_k \right] \quad (2.7)$$

Caso a tomada de decisão seja feita repetidamente, o horizonte do MDP é infinito e o fator de desconto γ é usado para garantir a convergência do valor da recompensa total esperada (PELLEGRINI; WAINER, 2007). Seu cálculo é definido por:

$$E \left[\lim_{z \rightarrow \infty} \sum_{k=0}^{z-1} \gamma^k r_k \right] \quad (2.8)$$

A recompensa total esperada ótima é dada pela função valor $V^*(s)$. Quando a recompensa total esperada para cada estado é máxima, a política é ótima (π^*). A função valor é definida por:

$$V^*(s) = \max_{a \in A} \left[R(s, a) + \gamma \sum_{s' \in S} T(s'|s, a) V^*(s') \right] \quad (2.9)$$

Para uma política π , dado que o sistema se encontra em um estado $s \in S$, pode-se definir por $Q^\pi(s, a)$ o valor de uma ação $a \in A$ no estado s considerando a recompensa imediata de a e as recompensas esperadas após sua execução, desde que as ações tomadas após a sejam determinadas pela política π :

$$Q^\pi(s, a) = R(s, a) + \gamma \sum_{s' \in S} T(s'|s, a) V^\pi(s') \quad (2.10)$$

Para uma política ótima π^* , Q^* é definida por:

$$Q^*(s, a) = R(s, a) + \gamma \sum_{s' \in S} T(s'|s, a) V^*(s') \quad (2.11)$$

Analisando as duas equações, percebe-se que existe a seguinte relação entre as funções V^* e Q^* :

$$V^*(s) = \max_{a \in A} [Q^*(s, a)] \quad (2.12)$$

A política ótima π^* que mapeia as ações que retornam o máximo valor de Q para cada estado é definida por:

$$\pi^*(s) = \arg \max_{a \in A} Q^*(s, a) \quad (2.13)$$

2.4.1 Solução do MDP

A solução de um MDP se baseia em encontrar a política ótima π^* que produz a máxima função valor ($V^*(s)$) para todos os estados. O algoritmo utilizado para a solução do MDP neste trabalho é o de iteração de valor (Algoritmo 1). Este algoritmo usa programação dinâmica para determinar o valor de $V^*(s)$ de cada estado $s \in S$, em cada época de decisão (PELLEGRINI; WAINER, 2007).

O critério de parada deste algoritmo, conhecido como erro de *Bellman*, para que a precisão da solução seja no mínimo um erro ϵ é dado por:

$$\forall s \in S, |V(s) - V'(S)| \leq \frac{\epsilon(1 - \gamma)}{2\gamma} \quad (2.14)$$

Algoritmo 1: Iteração de valor

Entrada: MDP (S, A, T, R)
Saída: V^*
para cada $s \in S$ **faça**
 | $V_0(s) \leftarrow \max_{a \in A} R(s, a);$
fim
 $i \leftarrow 1;$
enquanto *critério de parada não satisfeito* **faça**
 | **para cada** $s \in S$ **faça**
 | | $V_i(s) \leftarrow \max_{a \in A} [R(s, a) + \gamma \sum_{s' \in S} T(s'|s, a) V_{i-1}(s')];$
 | **fim**
 | $i \leftarrow i + 1;$
fim enquanto
devolva $V;$

2.4.2 Exemplo de MDP

Seja considerada uma máquina que trabalha as matérias primas A e B não simultaneamente. Após o processamento da matéria prima, o produto pronto é retirado da mesma, tornando-a livre para o próximo processamento. A figura 5 mostra a RdP que modela esse sistema e o diagrama de estados. Este exemplo é muito similar ao proposto por Bastos (2010).

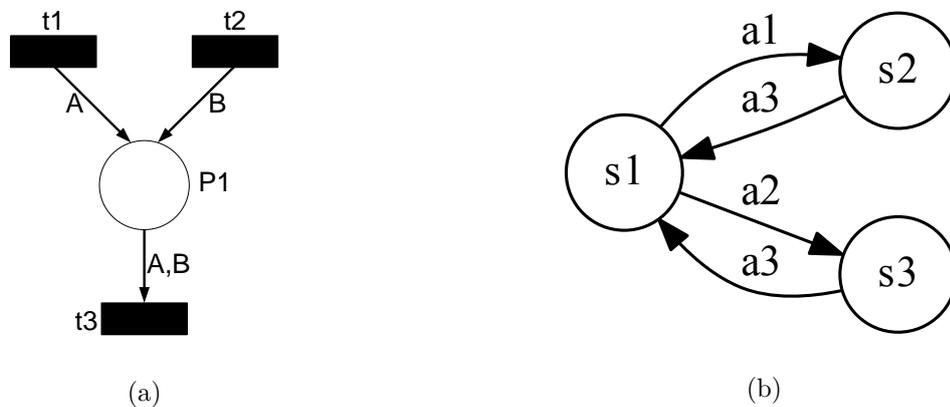


Figura 5 – Exemplo de MDP com 3 estados: (a) representação em RdP, (b) diagrama de estados

A tabela 1 apresenta as probabilidades de transição. Vale notar que as probabilidades de transição são 100% para todas as ações devido ao fato de não se tratar de sistemas concorrentes. Assim, sempre que uma ação for executada (por exemplo, disparo da transição $t1$), o sistema sempre tem apenas um estado subsequente a ser atingido (no caso peça A na máquina). As recompensas para cada ação podem ser vistas na tabela 2.

A interpretação da tabela 1 é que, caso o sistema esteja em $s1$, o próximo estado é $s2$ se for tomada a ação $a1$ ou $s3$ se for tomada a ação $a2$, ambos com 100% de probabilidade. Caso esteja em $s2$ ou $s3$, o próximo estado sempre será $s1$, mediante ação $a3$

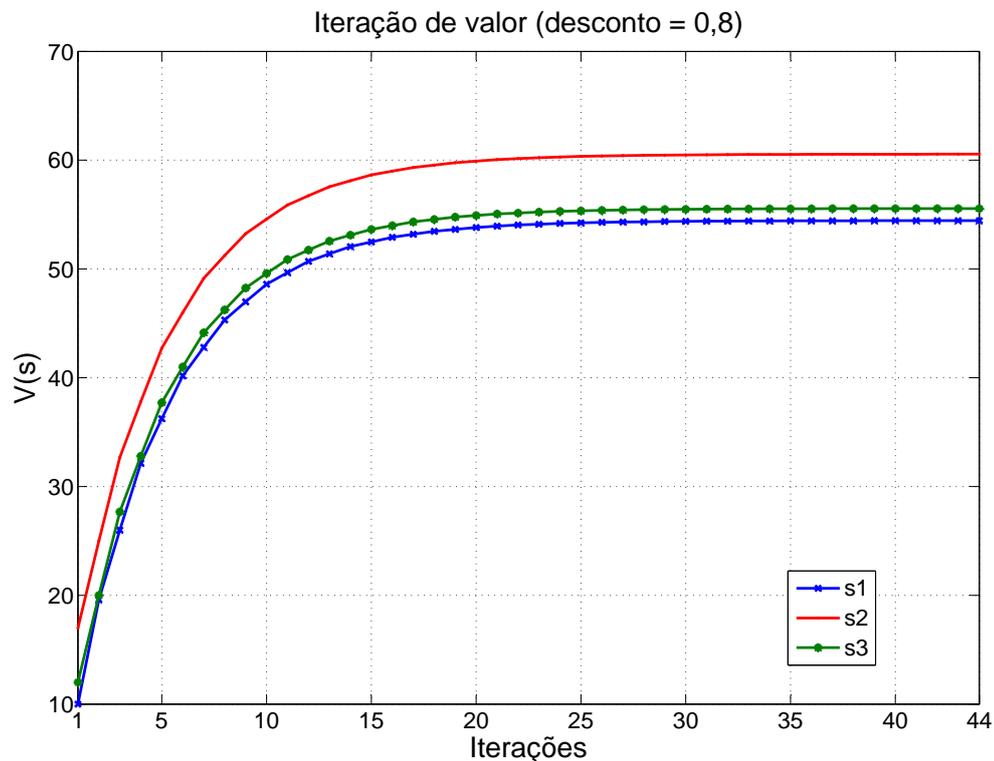
Tabela 1 – Probabilidades de transições

Estado	$a1$	$a2$	$a3$
$s1$	$(s2,1)$	$(s3,1)$	–
$s2$	–	–	$(s1,1)$
$s3$	–	–	$(s1,1)$

Tabela 2 – Recompensas

Estado	$a1$	$a2$	$a3$
$s1$	5	10	–
$s2$	–	–	17
$s3$	–	–	12

que significa fim de processamento da máquina. Para analisar a influência do fator de desconto na resolução do MDP, as figuras 6 e 7 mostram o resultado da função valor de cada estado para $\gamma = 0,8$ e $\gamma = 0,2$ respectivamente. Outra conclusão interessante é o número de iterações necessárias para a convergência do algoritmo iteração de valor, dependendo do fator de desconto: quanto menor o valor de γ , mais rápido ocorre a convergência.

Figura 6 – Iteração de valor para $\gamma = 0,8$

A análise dos dois gráficos permite que seja observado o resultado da máxima função valor para cada estado, ou seja, a máxima recompensa que este pode receber. O MDP, no entanto, considera o máximo valor entre todas as ações para cada estado. Neste

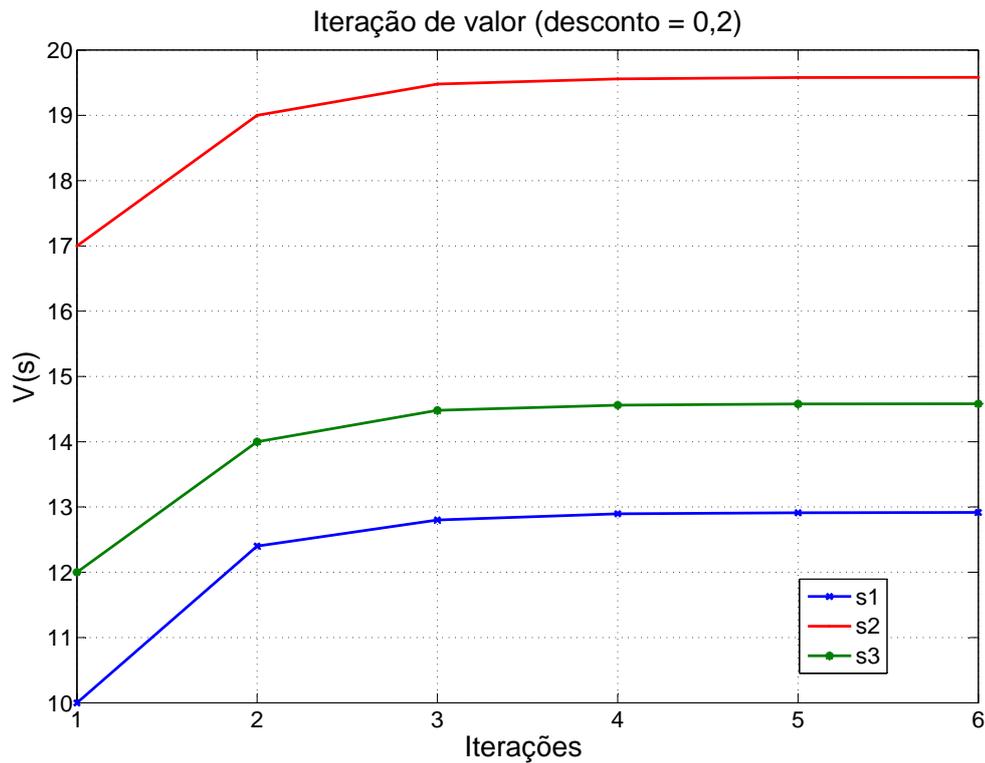


Figura 7 – Iteração de valor para $\gamma = 0,2$

exemplo, o único estado para o qual deve haver a seleção de maior valor é o estado $s1$, pois apenas este estado tem mais de uma ação. Para $s2$ e $s3$ existe apenas uma ação e, portanto, apenas um valor de recompensa. O cálculo do MDP fornece a política indicando quais ações devem ser tomadas em cada estado. As tabelas 3 e 4 mostram os valores da função $Q(s, a)$ e a política para $\gamma = 0,8$, enquanto as tabelas 5 e 6 consideram $\gamma = 0,2$.

Tabela 3 – $Q(s, a)$ para $\gamma = 0,8$

Estado	$a1$	$a2$	$a3$
$s1$	53,4	54,4	–
$s2$	–	–	60,5
$s3$	–	–	55,5

Tabela 4 – Solução do MDP para $\gamma = 0,8$

Estado	$V(s)$	Ação
$s1$	54,4	2
$s2$	60,5	3
$s3$	55,5	3

Tabela 5 – $Q(s, a)$ para $\gamma = 0,2$

Estado	$a1$	$a2$	$a3$
$s1$	8,92	12,9	–
$s2$	–	–	19,6
$s3$	–	–	15,6

Tabela 6 – Solução do MDP para $\gamma = 0,2$

<i>Estado</i>	$V(s)$	<i>Ação</i>
$s1$	12,9	2
$s2$	19,6	3
$s3$	15,6	3

2.5 *FESTO Logistics League*[®]

A motivação deste trabalho surgiu a partir da leitura do livro de regras da competição *FESTO Logistics League*[®] 2012 (FESTO, 2012). Esta se trata de uma competição utilizando robôs didáticos *Robotino*[®], cujo objetivo é a simulação de veículos automaticamente guiados (*AVGs* ou *autonomous guided vehicles*). A arena da competição simula uma linha de produção de três estágios, onde a produção é feita passo a passo a partir de matérias primas ou subprodutos. As equipes, com até três robôs, devem completar tarefas de transporte de materiais entre servidores, sendo a principal tarefa o transporte do produto final até o depósito designado. Há uma maior recompensa para a entrega do produto pronto em relação à conclusão parcial das tarefas. A figura 8 mostra a vista superior da arena da competição, destacando a posição das máquinas.

Cada máquina realiza uma tarefa necessária para a conclusão do produto final e tem um tempo de execução da tarefa. Materiais consumidos podem ser reciclados para matéria prima ($S0$) em estações de reciclagem posicionadas nas quinas da arena. A tabela 7 mostra as entradas, saídas e tempos de produção de cada máquina.

Tabela 7 – Descrição das máquinas de *FESTO Logistics League*[®]

Tipo de máquina	Entrada	Saída	Tempos de produção
$M1$ (4 unidades)	1x $S0$ (mat. prima)	$S1$	WT1=3 a 8 seg
$M2$ (3 unidades)	1x $S0$, 1x $S1$	$S2$, 1 x cons.	WT2=15 a 25 seg
$M3$ (3 unidades)	1x $S0$, 1x $S1$, 1x $S2$	Produto, 2 x cons.	WT3=40 a 60 seg

A ideia do desenvolvimento de um sistema de tomada de decisão partiu do modelo proposto pela competição, uma linha de manufatura. Ao invés de o objeto de estudo ser o sistema de transporte, decidiu-se pela tomada de decisão a partir da observação do estado geral do sistema. Neste caso, o MDP é uma boa ferramenta de tomada de decisão a ser utilizada, pois prevê uma ação a ser executada para cada estado do sistema a partir das recompensas obtidas através de sua execução. Além disso, linhas de produção são

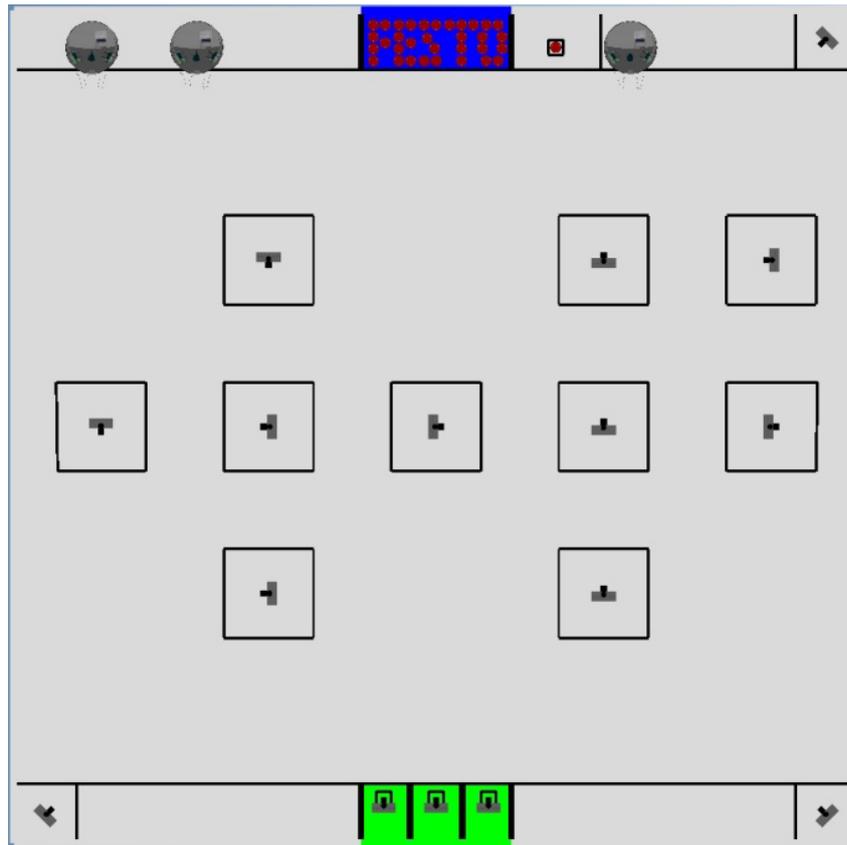


Figura 8 – Arena da competição *FESTA Logistics League*[®] (FESTA, 2012)

facilmente modeladas pelas RdPs, que é a ferramenta de modelagem escolhida para este trabalho.

No entanto, foram feitas algumas modificações na definição dos sistemas a serem simulados de forma a simplificar sua representação. Não foram consideradas mais de uma máquina de cada tipo nem operações que necessitam de diferentes subprodutos como entrada. Além disso, o arranjo sequencial de produção permite melhor caracterização dos modelos de linhas de manufatura que foram exploradas.

2.6 Sistemas Flexíveis de Manufatura (FMS)

Uma das características que torna possível a utilização de sistemas flexíveis de manufatura como objeto de aplicação neste trabalho é que sistemas de manufatura são chamados de sistemas de produção discretos (SILVA; VALETTE, 1990), pois não trabalham com um fluxo contínuo de material, mas sim com lotes e unidades. Esta característica permite que seja usada as RdP como ferramenta de modelagem, e então aplicar todas as suas propriedades para análise do sistema.

Um sistema a eventos discretos (SED) (RAMADGE; WONHAM, 1989) é um sistema dinâmico cuja mudança de estados ocorre em pontos discretos do tempo. Um SED,

modelado nas RdP, faz com que exista a dinâmica de estados, ou seja, o sistema atinja um estado S' a partir de um estado S caso uma determinada transição seja disparada. Estas transições da RdP devem, portanto, ser associadas às ações do sistema, e consequentemente, ações do MDP.

Ações são definidas como eventos, os quais podem ser classificados como controláveis ou não controláveis. A ideia de tomada de decisão tem influência nos eventos controláveis, de forma que um agente possa intervir no sistema provocando a evolução deste para outro estado. Os eventos não controláveis são aqueles que não dependem de um estímulo externo para serem executados. Num sistema de manufatura, um evento não controlável padrão pode ser exemplificado como o término de processamento de uma peça em uma máquina.

Um sistema de produção (*workflow*) em sua essência é formado por máquinas dispostas em determinado arranjo (*layout*), pelas quais alguns produtos passam numa rota pré-determinada. A fabricação de um produto é chamada de trabalho (*job*), e cada *job* está dividido numa série de operações que são executadas nas máquinas (MUNIZ, 2009). Indo além, Groover (2007) diz que um sistema de manufatura é um conjunto composto de recursos humanos e equipamentos integrados cuja principal função é desempenhar uma série de atividades, nas quais se desenvolvem operações de montagem e processamento sobre uma matéria prima, peça ou conjunto de peças de modo a obter-se um produto final.

Geralmente são estudados problemas de sistemas de produção de forma a determinar qual o melhor arranjo de máquinas para a diminuição do tempo total de produção (*makespan*). Esta tarefa é chamada de escalonamento ou programação de produção (*scheduling*). Desta forma, ao se conseguir o arranjo de produção que minimize o tempo total de produção, é conseguida a sequência de produção ótima para uma gama de produtos. No entanto, este não é o objeto de estudo principal deste trabalho, uma vez que a proposta é a verificação do uso do MDP na tomada de decisão em tempo real. Ou seja, independente do estado que o sistema se encontra, qual ação deve ser tomada de forma que se maximize a recompensa do sistema. Esta é uma característica importante a ser notada, pois, enquanto estudos de escalonamento buscam sempre minimizar o *makespan*, a utilização do MDP é mais flexível, uma vez que o conceito de recompensa do sistema não é relacionado exclusivamente a uma variável. A recompensa total do sistema pode ser a minimização do tempo ou a maximização de lucros, por exemplo. Esta definição depende do usuário que aplica a teoria ao sistema e a forma como foram utilizadas as funções de utilidade do MDP.

2.6.1 Classificação dos sistemas de produção

Os sistemas de produção podem ser classificados de várias formas, dependendo de características de arranjos e de tipos de máquinas, entre outros. No entanto, para a determinação dos tipos de sistemas a serem usados neste trabalho, optou-se pela classificação segundo a complexidade do processo (REIS, 1996). Os três principais tipos de sistema de produção segundo sua complexidade são: ambiente aberto (*openshop*), ambiente contínuo (*flowshop*) e ambiente intermitente (*jobshop*). Os sistemas de produção que serão modelados e simulados neste trabalho são o *flowshop* e o *jobshop*.

O *openshop* caracteriza-se pela ausência de ordem para execução de operações, não existe uma dependência entre as operações. Assim, não importa em qual máquina um produto seja processado, a operação será concluída sem dependência de uma tarefa anterior ou posterior. Devido a esta característica, não é possível a modelagem do modelo *openshop* e por isso não será abordado nas modelagens e simulações. O *flowshop* é o sistema sequencial em sua essência, aquele mostrado por Chaplin (1936) no filme “Tempos Modernos”. Neste sistema, todos os produtos tem a mesma rota, passando por todas as máquinas. Segundo a figura 9, cada peça inserida na linha é iniciada na máquina *M1* e finalizada na máquina *M3*, seguindo um caminho fixo de manufatura. O número de operações é o mesmo que o número de máquinas. Este tipo de sistema apresenta o maior número de estados, devido a presença de todas as peças em todos os lugares da RdP. Por outro lado, como os eventos controlados se baseiam na adição de apenas uma nova peça no início do processo, este se torna o processo de decisão mais simples. A figura 9 mostra um arranjo do tipo *flowshop* com três tipos de produtos e três máquinas, com os fluxos de produção dos seus produtos.

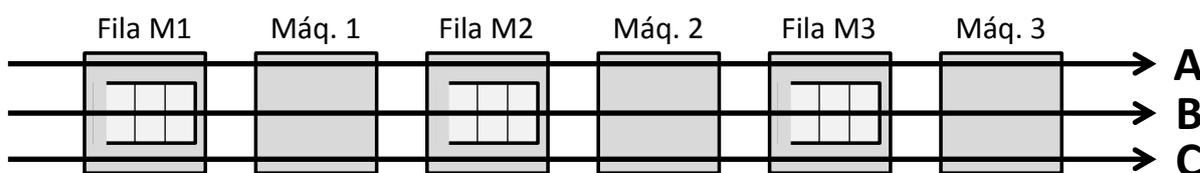


Figura 9 – Diagrama de blocos de um sistema de produção *flowshop*

O *jobshop* consiste de uma variação do *flowshop*, pois neste caso cada produto tem uma sequência de produção, porém estas não são idênticas. Além disso, o número de máquinas não representa necessariamente o número de operações. Algumas máquinas podem não ser usadas ou usadas mais de uma vez (*jobshop* com recirculação), dependendo da rota de cada produto na linha. Em outras palavras, o *jobshop* é um caso mais genérico, onde qualquer rota de produção pode ser considerada, enquanto o *flowshop* é um caso especial, onde a sequência de produção de todos os produtos é a mesma. A figura 10 mostra um arranjo do tipo *jobshop* para três tipos de produtos e três máquinas. Neste exemplo, a fabricação do produto tipo A inicia-se na máquina *M2* e é finalizada em *M3*.

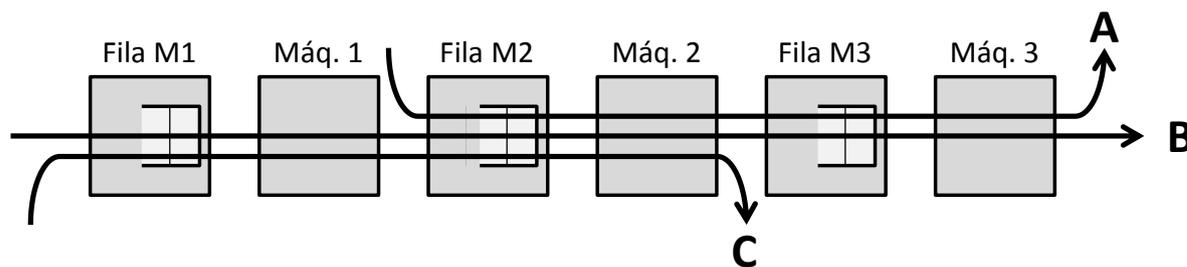


Figura 10 – Diagrama de blocos de um sistema de produção *jobshop*

O produto C é iniciado na máquina *M1* e finalizado em *M2*. Já o produto B é iniciado em *M1*, processado em *M2* e finalizado em *M3*. Diferentemente do *flowshop*, cada produto tem sua sequência de produção.

O trabalho proposto por Muniz (2009) traz exemplos muito didáticos para os três tipos de sistemas de produção propostos:

- *Openshop*: uma fábrica de iogurtes que produz três sabores. As operações que devem ser executadas são: enchimento das garrafas, colagem do rótulo na garrafa e impressão de data de validade e lote. As tarefas não têm ordem fixa, uma tarefa não depende da outra. Isto é, seja qualquer operação escolhida como partida na linha de produção e seja qualquer ordem de máquinas escolhida, o produto final será o mesmo;
- *Flowshop*: uma linha de manufatura de peças de aço conta com três máquinas: estampadora, forno de tratamento térmico e estação de usinagem para acabamento. São produzidos três tipos de produtos automotivos (engrenagens, aro de roda, careca de embreagem). Não importa qual produto esteja sendo produzido, a ordem de produção é sempre a mesma: estampagem, acabamento e tratamento térmico. A alteração de ordem de produção impossibilita a conclusão dos trabalhos de fabricação ou pode trazer não conformidades ao produto final;
- *Jobshop*: cinco produtos são produzidos numa fábrica que contém seis máquinas. Cada produto tem uma sequência individual de produção, sendo necessário ou não que sejam trabalhados em todas as máquinas (figura 11).

2.6.2 Eficiência versus flexibilidade e origem do FMS

Uma análise qualitativa que pode ser feita entre *flowshop* e *jobshop* é que o primeiro apresenta uma maior eficiência, pois a ideia de linha de produção permite que as máquinas estejam sempre ocupadas, processando algum produto, ou desocupadas, mas esperando que a tarefa subsequente seja finalizada para sequência de produção. No entanto,

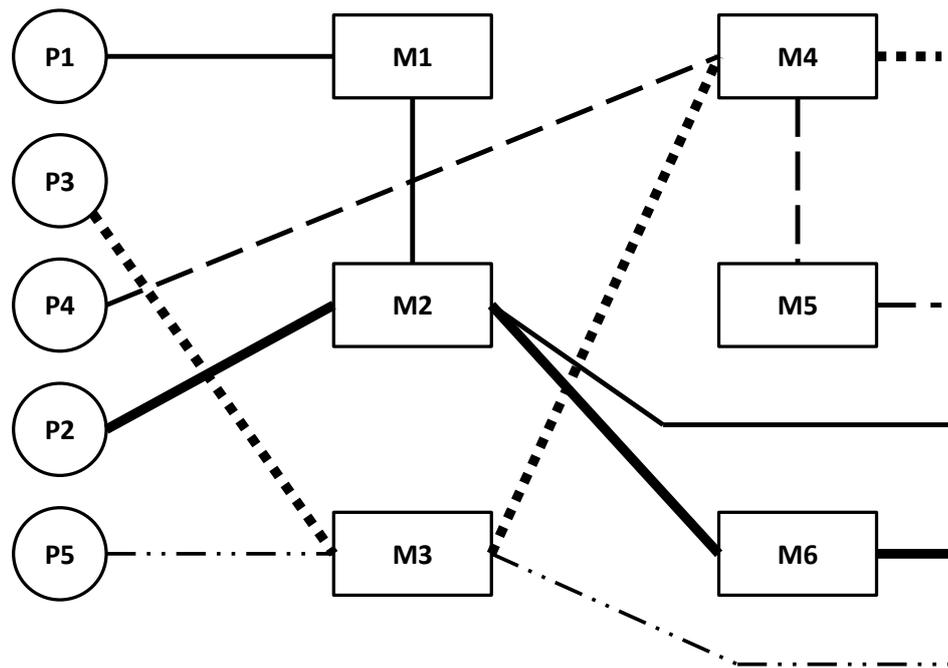


Figura 11 – Layout de sistema jobshop

é pouco flexível por apenas admitir que sejam fabricados produtos de mesma família, ou características semelhantes.

O *jobshop*, por outro lado, é mais versátil, devido à liberdade com relação às sequências de produção. Assim, enquanto o produto de uma família é produzido numa máquina i , um produto diferente pode iniciar seu processamento numa máquina $i + n$ que esteja livre no momento. Porém, devido ao maior número de rotas presentes, podem haver máquinas ociosas e produtos semi-processados em *buffers* intermediários durante maior parte do tempo de produção. Isso gera uma queda significativa no desempenho do arranjo, sendo necessário o desenvolvimento de um plano de escalonamento da produção ou a aplicação de teorias de decisão.

Os sistemas flexíveis de manufatura (*Flexible Manufacturing Systems* - FMS) surgem da tentativa de se unir a eficiência da linha de produção (*flowshop*) com a flexibilidade do *jobshop*, e são formados por (SILVA; VALETTE, 1990):

- Conjunto de máquinas flexíveis;
- Sistema automático de transporte;
- Sistema de tomada de decisão.

Apesar de uma das motivações deste trabalho ser a competição *FESTO Logistics League*[®], não foi considerado um sistema de transporte automático. As máquinas flexíveis

são representadas pelos próprios recursos na RdP. No entanto, a grande contribuição deste estudo é relativa ao sistema de tomada de decisão. Em FMS, o sistema de tomada de decisão é responsável por decidir o que produzir e em qual máquina. O MDP é um agente observador que verifica o estado geral do sistema (o *status* de cada máquina e *buffers*) e então age nos eventos controláveis do sistema em tempo real, decidindo qual novo produto deve ser introduzido no sistema de manufatura. Silva e Valette (1990) também expõem que o FMS é composto por vários níveis hierárquicos e existe a possibilidade de serem modelados por RdP. No entanto, não é interesse deste trabalho explorar a fundo cada um deles. Estes são:

- Planejamento;
- Escalonamento;
- Coordenação global;
- Coordenação de subsistemas;
- Controle local.

Dentre os níveis apresentados, o que traz mais interesse é a coordenação global, pois sua função é o monitoramento e tomada de decisões em tempo real. É neste nível que são verificados o estado atual do sistema, o *status* de cada recurso, quais ações são possíveis e qual deve ser considerada.

Uma observação que deve ser feita quando são tratados sistemas de manufatura é o comportamento contraditório entre flexibilidade e otimalidade. Uma ação que traz vantagens para flexibilidade geralmente traz desvantagens em termos de otimização. Por exemplo, ao se aumentar a capacidade de *buffers* intermediários consegue-se boa flexibilidade para o processo devido à quantidade de recursos disponíveis em qualquer momento. No entanto, é ruim para otimalidade devido à necessidade de grandes depósitos e matéria prima disponível. A utilização de RdP na modelagem de FMS permite que sejam colocadas restrições no sistema de forma que se consiga equilibrar os requisitos de flexibilidade com otimização. Assim, é possível que o sistema não tome proporções não desejadas (restrições de capacidade) e também tenha um ganho de eficiência (produção executada de acordo com a tomada de decisões do MDP).

2.7 Trabalhos relacionados

Com mais de meio século desde a primeira formalização das RdPs, muitos pesquisadores se dedicaram no desenvolvimento desta ferramenta, principalmente nas décadas

de 1980 e 1990. As RdPs temporizadas, RdPs coloridas, RdPs *fuzzy*, entre outros, podem ser citadas como contribuições que agregaram maior poder de representação a esta ferramenta. Por este motivo, muitos são os trabalhos de pesquisa que utilizam a RdP como ferramenta de desenvolvimento. Além disso, é possível que se faça a conexão entre RdP e MDP devido ao fato de se alcançar os estados markovianos de um sistema ao se determinar a árvore de alcançabilidade de uma RdP. Com isso, é plausível que seja utilizado o MDP como ferramenta de tomada de decisão para um sistema modelado por RdP. Finalizando os três principais elementos de estudo deste trabalho, tem-se o FMS, que pode ser representado como sistemas a eventos de discretos e para o qual uma das principais características é a presença de um elemento tomador de decisões. Com base nesta tríade, foi buscado na literatura o apoio de outros trabalhos que contemplam, mesmo que parcialmente, algumas características destes três elementos.

O trabalho de Eboli (2010) define um método de transformação de RdP colorida em MDP com probabilidades imprecisas. Apesar de tratar do tema de imprecisão nas probabilidades de transições entre estados do MDP, assunto não abordado nesta dissertação, tal trabalho serviu como ponto inicial, pois um dos objetivos buscados é esta transformação. Eboli (2010) define o MDP através de variáveis de estados, suas combinações e dependências. As recompensas são calculadas separadamente. No trabalho aqui apresentado, no entanto, o MDP é obtido diretamente da árvore de alcançabilidade da RdP, uma vez que, definidos os estados e feito o cálculo de recompensa no processamento da árvore, pode-se ter como resultados as matrizes de transição e recompensa do MDP. A solução do MDP é obtida através do cálculo das políticas de acordo com um fator de desconto. Esta é a principal diferença entre os dois trabalhos. Como semelhanças, pode-se citar o fato de as ações (ou eventos controláveis) serem definidas nas transições fontes da RdP e o uso da RdP colorida como ferramenta de modelagem ao invés da RdP ordinária.

Agerwala (1979) e Aalst (1994) tratam da RdP como ferramenta de uso em aplicações industriais. Além da formalização teórica (muito mais aprofundada por Murata (1989)), estes autores apresentam exemplos de sistemas simples, porém muito encontrados no ambiente industrial, modelados por RdP. Já Silva e Valette (1990), além de apresentarem as características do FMS, fazem uso da RdP como ferramenta primordial de modelagem. Seu trabalho traz detalhes sobre os conceitos FMS, apresentando todos os níveis presentes no planejamento e controle da produção. Estes autores vão além do “apenas produzir” e mostram como a RdP pode ser usada para modelagem de sistemas específicos de produção, como o *Just-In-Time*.

Explorando o tema FMS, Silva (2007) aplica a teoria de controle supervisorio (TCS) em um FMS, focado especificamente no sistema de rotas. Como visto na revisão bibliográfica, um dos componentes do FMS é um sistema automático de transporte. Este trabalho, além de dar outra visão sobre os conceitos de FMS, também abre o horizonte

para trabalhos futuros, uma vez que o sistema de transporte não está sendo considerado.

Mais além, Molina (2007) propõe o controle através da TCS com modelagem do sistema por RdP coloridas. Pode-se observar neste trabalho certa semelhança com o proposto por esta dissertação, uma vez que a modelagem se dá por RdP colorida e existe um agente tomador de decisões. A diferença encontra-se primeiramente na ferramenta de controle proposta por Molina (2007), já que a TCS tem uma modelagem diferente que a do MDP, além de tratar com muitos detalhes uma célula de manufatura específica. Já esta dissertação considera a aplicação em sistemas mais genéricos, com abstração de detalhes, além de ter a transformação de RdP em MDP como um importante resultado.

3 Modelagem do FMS

Como exposto no capítulo de revisão de literatura, a ferramenta utilizada para a modelagem dos FMS é a rede de *Petri* colorida limitada. Com esta ferramenta é possível representar os recursos do sistema, como filas e máquinas, e eventos, controláveis e não controláveis. A representação em RdP, além da característica gráfica, é muito útil no desenvolvimento de algoritmos devido a facilidade de representação algébrica através de matrizes que regem o comportamento do sistema.

Através da notação matricial da RdP também é possível a geração dos estados do sistema que são os estados do MDP. Assim, computadas a matriz de transições, a matriz de recompensas e um fator de desconto, é possível determinar a política ótima do MDP de forma a maximizar os ganhos do sistema.

Contudo, lançando-se mão apenas da representação gráfica/matricial da RdP do sistema e da política ótima para a tomada de decisão, a análise de resultados não é intuitiva. A verificação de estados, bem como dos resultado das ações e processamento de recursos fica limitada e a análise do sistema em tempo real é praticamente impossível. A solução encontrada para a simulação dos FMS foi o uso do *SimEvents*[®] (THE MATHWORKS, 2005–2010), um *toolbox* para para simulação de sistemas a eventos discretos pelo *software* matemático *MatLab*[®].

3.1 O *SimEvents*[®]

O *SimEvents*[®] é um simulador de eventos discretos utilizado como biblioteca do *Simulink*[®] com o qual pode ser modelada a comunicação entre componentes permitindo análise de desempenho de sistemas. Todos os elementos do FMS podem ser representados no *SimEvents*[®] através de blocos pré-definidos, como servidores, filas e elementos de roteamento. Além disso, é possível que sejam adicionadas funções customizadas pelo usuário, recurso utilizado extensivamente neste trabalho nos processos de verificação de estados e execução das ações.

Devido a estas características, são encontrados muitos exemplos de modelagem de processos de manufatura com o auxílio deste *toolbox* para determinação de exigência de recursos, análise de planos de produção, identificação de gargalos entre outras análises de desempenho. Além da modelagem, existe uma série de blocos de visualização e análise em tempo real, importantes ferramentas na geração de resultados.

3.1.1 *SimEvents*[®] e redes de *Petri*

Apesar de todo poder para simulação de sistemas a eventos discretos, é necessário que seja feita uma analogia dos elementos da RdP com os blocos do *SimEvents*[®]. Considerando que o FMS é primeiramente modelado em RdP e toda geração de estados é feita através da construção da árvore de alcançabilidade, é necessário que a ferramenta de simulação consiga representar o sistema com a mesma exatidão e fidelidade com a qual este foi modelado. A tabela 8 mostra a equivalência destes elementos. É possível observar que existem blocos pré-definidos do *SimEvents*[®] que realizam exatamente a mesma tarefa de determinados elementos da RdP. Em alguns casos, a composição de blocos é necessária (como o caso da transição fonte que insere um *token* de determinada cor no sistema). Também é possível observar que o uso de uma sequência de blocos torna desnecessário o uso de algum elemento da RdP (a representação de algumas transições da RdP não se faz necessária, como no caso do arranjo fila e servidor, onde a própria sequência dos blocos realiza a transição, respeitando restrições).

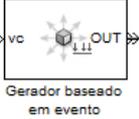
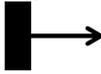
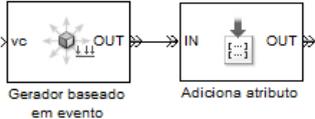
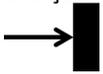
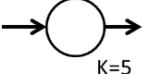
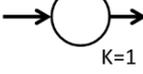
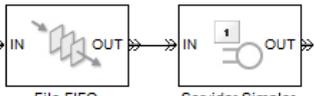
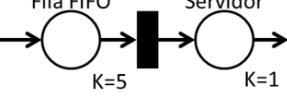
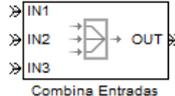
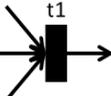
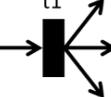
No entanto, apesar destas particularidades, fica clara a representação dos elementos mais importantes na modelagem do FMS. Na RdP, as transições fontes realizam as ações do MDP (eventos controláveis), enquanto no *SimEvents*[®] pode-se utilizar um bloco de função customizada contendo uma função para análise de estados, cujo resultado pode disparar um bloco gerador de entidade baseado em evento.

O bloco *atributos* pode adicionar, por exemplo, a cor de uma entidade (tipo de peça), e já atribuir a esta o tempo de processamento numa determinada máquina, que será usado como característica temporal do sistema e está diretamente ligado ao cálculo de recompensas neste trabalho. Além disso, um bloco servidor simples representa um lugar de capacidade $k = 1$, que é uma máquina, e uma fila com n posições é o mesmo que um lugar de capacidade $k = n$ da RdP.

Pode-se mostrar a relação entre RdP e *SimEvents*[®] através de um exemplo. A figura 12 mostra um RdP que modela um pequeno sistema de manufatura onde são produzidos os produtos tipo A e tipo B. O produto A é inserido na linha pela transição $t1$ e seu processamento é iniciado no *buffer* $P1$ com capacidade $k = 5$, passando pela máquina $P2$ e sendo retirada pela transição $t4$. Já B é introduzido na linha pela transição $t2$, passa pelo *buffer* $P1$ e máquina $P2$, sendo também processado na máquina $P3$ e finalmente retirado pela transição $t6$.

A figura 13 mostra o mesmo exemplo modelado no ambiente *SimEvents*[®]. Existem dois geradores de entidades que representam as transições fontes para produtos A e B. Os blocos *atributos A* e *atributos B* classificam as entidades, adicionando cor e tempo de produção em cada máquina. O bloco *Fila(P1)* é uma fila FIFO (*First-In-First-Out*) representando o *buffer* $P1$ e os servidores *Máquina P2* e *Máquina P3* são as máquinas do

Tabela 8 – Equivalência de blocos do *SimEvents*[®] e elementos da RdP

Elemento do SimEvents™	Elemento da Rede de Petri	Descrição
 <p>Gerador baseado em evento</p>	<p>Transição fonte</p> 	<p>Transição fonte: insere um token qualquer no sistema</p>
 <p>Gerador baseado em evento Adiciona atributo</p>	<p>Transição Fonte A</p> 	<p>Transição fonte A: insere um token de cor A no sistema.</p>
 <p>Dreno</p>	<p>Transição dreno</p> 	<p>Transição dreno: retira um token do sistema.</p>
 <p>Fila FIFO</p>	<p>Fila FIFO</p>  <p>K=5</p>	<p>Fila <i>FIFO</i> (<i>first in, first out</i>): neste caso com cinco posições.</p>
 <p>Servidor Simples</p>	<p>Servidor Simples</p>  <p>K=1</p>	<p>Servidor simples: recurso de processamento, admite uma unidade por vez.</p>
 <p>Fila FIFO Servidor Simples</p>	<p>Fila FIFO Servidor</p>  <p>K=5 K=1</p>	<p>Fila e servidor: arranjo de fila e servidor em sequência.</p>
 <p>Combina Entradas</p>	<p>t1</p> 	<p>Transição: múltiplos lugares pré-condição, um lugar pós-condição.</p>
 <p>Divide Saída</p>	<p>t1</p> 	<p>Transição: um lugar pré-condição, múltiplos lugares pós-condição.</p>

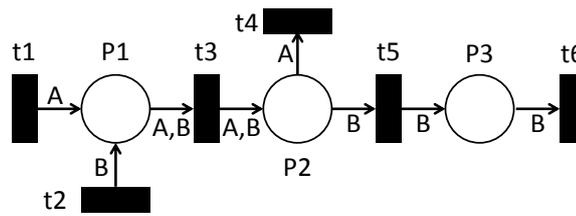


Figura 12 – RdP que modela sistema simples de produção

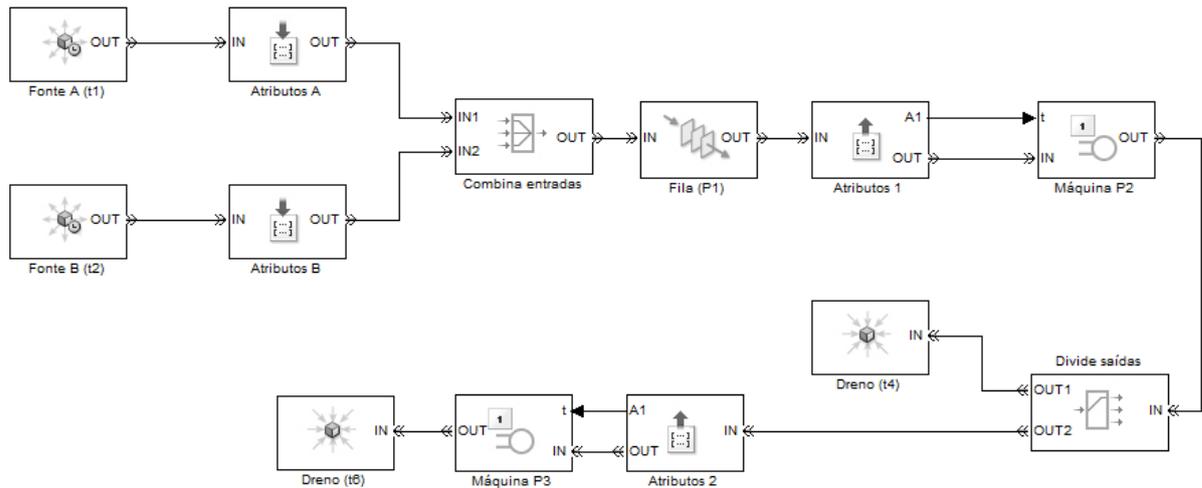


Figura 13 – Modelo em *SimEvents*[®]

sistemas. Finalizando, os blocos *Dreno (t4)* e *Dreno (t6)* representam as transições dreno *t4* e *t6* da RdP e os blocos *divide saídas* e *combina entradas* são blocos de roteamento.

3.2 Redes de *Petri* e processos de decisão de *Markov*

A ferramenta utilizada neste trabalho para tomada de decisão foi o MDP (*Markov Decision Processes*), na qual uma recompensa total é calculada de acordo com os estados de um sistema e cada ação tomada. Para sistemas muito pequenos é possível analisar todos os estados atingíveis e relacionar uma determinada recompensa para cada ação tomada. No entanto, para sistemas maiores, a quantidade de estados cresce exponencialmente, principalmente levando em consideração que serão abordados diversos tipos de peças e filas com tamanhos variáveis. Sendo assim, nada mais plausível que utilizar esforço computacional para determinar o conjunto de estados de um sistema, além de determinar os parâmetros de entrada para o cálculo do MDP, que retorna a política ótima para este sistema.

O uso de redes de *Petri* neste trabalho foi escolhido devido a possibilidade de representação matemática dos modelos através de sua notação matricial. Desta forma, foi possível a elaboração de algoritmos para análise de restrições e simulação dinâmica das

redes. Tais algoritmos foram desenvolvidos no software *Mathworks Matlab*[®], que além de apresentar uma linguagem bastante simples, permite o compartilhamento de resultados com outras funções e *toolboxes* disponíveis.

3.2.1 Modelagem da RdP e construção da árvore de alcançabilidade

Inicialmente foi gerado um algoritmo de entrada de dados *gera_rdp*, basicamente uma interface onde a RdP é inicializada. Certos parâmetros de entrada são necessários apenas para a RdP, enquanto outros farão parte da geração das matrizes de transições e recompensas do MDP. São parâmetros de entrada do *gera_rpd*:

- *n_color*: quantidade de cores utilizadas na rede (quantos produtos são considerados no FMS);
- matriz de entrada *E*: para a quantidade de cores especificada por *n_color* é necessária uma matriz bidimensional de entrada *E*. A matriz *E* final será uma matriz tridimensional com dimensões $t \times l \times c$ (t = quantidade de transições da rede, l = quantidade de lugares de rede, c = quantidade de cores requeridas);
- matriz de saída *S*: também para a quantidade de cores especificada por *n_color* é necessária uma matriz bi dimensional de saída *S*. A matriz final *S* também terá dimensão $t \times l \times c$;
- vetor marcação inicial *M*: indica a marcação inicial da rede. Esta marcação será a primeira referência do algoritmo de processamento da rede;
- vetor de limitação *K*: relaciona cada lugar da rede com a sua capacidade. Normalmente utiliza-se o valor um para lugares que representam máquinas e um valor qualquer não nulo e positivo para lugares que representam *buffers*;
- matriz transição/ação *TA*: relaciona cada transição da rede com a ação correspondente no MDP. É uma matriz $t \times 1$, onde cada componente contém o valor da ação correspondente a transição;
- matriz transição/tempo *Temp*: relaciona para cada transição o valor correspondente ao tempo necessário para disparo da mesma, para cada cor da rede. Esta matriz fica restrita apenas às transições que fazem a retirada de uma peça de uma determinada máquina. Assim, para transições que não representam retiradas de peça de um servidor, seu valor em *Temp* é zero. Esta matriz é utilizada para geração da matriz $R(s, a)$ de recompensa do MDP e tem dimensão $t \times c$;
- vetor de máquinas *Tmaq*: este vetor relaciona cada lugar da RdP com a máquina o qual este representa. Estes dados são importantes para que possam ser usadas

diferentes funções de utilidade para cada máquina. Assim, a recompensa de produção em cada uma das máquinas vai depender da sua própria função de utilidade e não de uma função de utilidade única para todas elas. Tem dimensão $1 \times t$;

- vetor de valor agregado α : em muitos casos um produto pode ter baixa recompensa em sua produção devido ao alto tempo de processamento. No entanto, o valor agregado a este produto é alto, o que gera um aumento na recompensa final de produção deste. O vetor α apresenta um fator para cada tipo de peça (cada tipo de cor), agregando um peso à sua fabricação, e modificando a recompensa. Sua dimensão é $1 \times c$.

Tendo sido gerada a RdP, pode-se executar o algoritmo *run_rdp*. Este algoritmo utiliza as matrizes de entrada E e saída S para, a partir de um estado inicial, verificar quais transições estão habilitadas e definir os estados futuros do sistema através dos disparos delas. Resumindo, o algoritmo *run_rdp* determina os estados do sistema (e consequentemente os estados do MDP) gerando a árvore de alcançabilidade do sistema. Normalmente a árvore de alcançabilidade é construída através de uma busca em profundidade, como mostra o algoritmo 2 (MORAES; CASTRUCCI, 2001).

Algoritmo 2: construção de árvore de alcançabilidade com busca em profundidade

Inicialize o nó raiz da árvore com o vetor estado inicial X_0 , chamando-o de “novo”;

Entrada: Nó raiz (X_0)

Saída: Árvore de alcançabilidade

enquanto *houver nó “novo”, selecione um deles e chame-o de X*

Verifique cada transição de saída de X ; se não há transição habilitada, X é um nó terminal da árvore, assinale-o com T ;

Se uma transição é habilitada, execute-a, gerando X' como novo estado.

Chame-o de “novo”;

Se X' é igual a um estado qualquer já existente na RdP, chame-o de “duplicado” e assinale-o com D ;

Verifique outras transições;

fim enquanto

Verifique outros nós “novos”;

Quando todos os nós forem assinalados como terminais ou duplicados, a árvore está completa;

No entanto, para conveniência de programação, decidiu-se usar outra forma de geração da árvore de alcançabilidade através de busca em largura (RUSSELL et al., 2010). A busca em profundidade explora a árvore mudando-se o estado de referência toda vez que uma transição é disparada e novo estado é alcançado. Já na busca em largura, são verificadas todas as transições e efetuados todos os disparos possíveis a partir de um certo estado de referência. Assim, partindo da marcação inicial da RdP como nó raiz da árvore, são determinadas quais transições estão habilitadas e estas são disparadas. No

início do algoritmo este nó raiz (marcação inicial) é tomado como referência e armazenado na primeira posição do vetor de estados.

Cada estado originado por um disparo de transição é armazenado no vetor de estados na ordem em que são gerados. Quando não houver mais transições habilitadas, a referência de estado muda do estado inicial para o próximo item do vetor de estados. Estados duplicados são ignorados no vetor de estados. A busca em largura adotada é mostrada no algoritmo 3.

Algoritmo 3: Construção de árvore de alcançabilidade com busca em largura

Entrada: Referência = M_0 , $ref = 1$, $st = 1$, $estados(st) = M_0$

Saída: Árvore de alcançabilidade

```

enquanto  $st > ref$ 
  para  $i = 1$  a  $n\_color$  faça
    para  $j = 1$  a  $T$  faça
      se  $T(j)$  está habilitada então
        |  $dispara T(j)$ 
      se  $M' \notin estados$  então
        |  $st = st + 1$ ,  $estados(st) = M'$ 
        |  $j = j + 1$ ;
      fim para
     $i = i + 1$ ;
  fim para
   $ref = ref + 1$ ;
  Referência =  $estados(ref)$ ;
fim

```

Uma consideração deve ser feita sobre o procedimento de construção de árvore de alcançabilidade de redes de *Petri* coloridas, pois quando se trata de RdP de alto nível, a representação de todos os estados do sistema de uma forma sucinta é mais complexa. Os trabalhos de Huber et al. (1986) e Lin e Marinescu (1989) tratam de uma nova concepção na época para construção de árvore de alcançabilidade de RdP colorida, a árvore com marcação equivalentes. Nestes trabalhos, o exemplo principal abordado é o problema dos cinco filósofos. Nas marcas equivalentes não importa qual filósofo está comendo e com quais palitos, mas sim cada nó da árvore indica quantos filósofos estão comendo e quantos palitos estão disponíveis. Desta forma, com a equivalência de marcas, a árvore pode ser resumida e, explicitando todas as opções de disparo entre os nós, não há perda de informação. A figura 14 mostra a árvore de alcançabilidade deste problema, com representação por marcas equivalentes (transição “Pega Palitos” representada por $T1$ e “Devolve Palitos” por $T2$).

No entanto, mesmo que haja a explosão de estados, é imprescindível que todos os estados do sistema sejam mapeados num primeiro momento. Após o cálculo da política do MDP, todos os estados serão associados a uma ação que maximiza a recompensa total

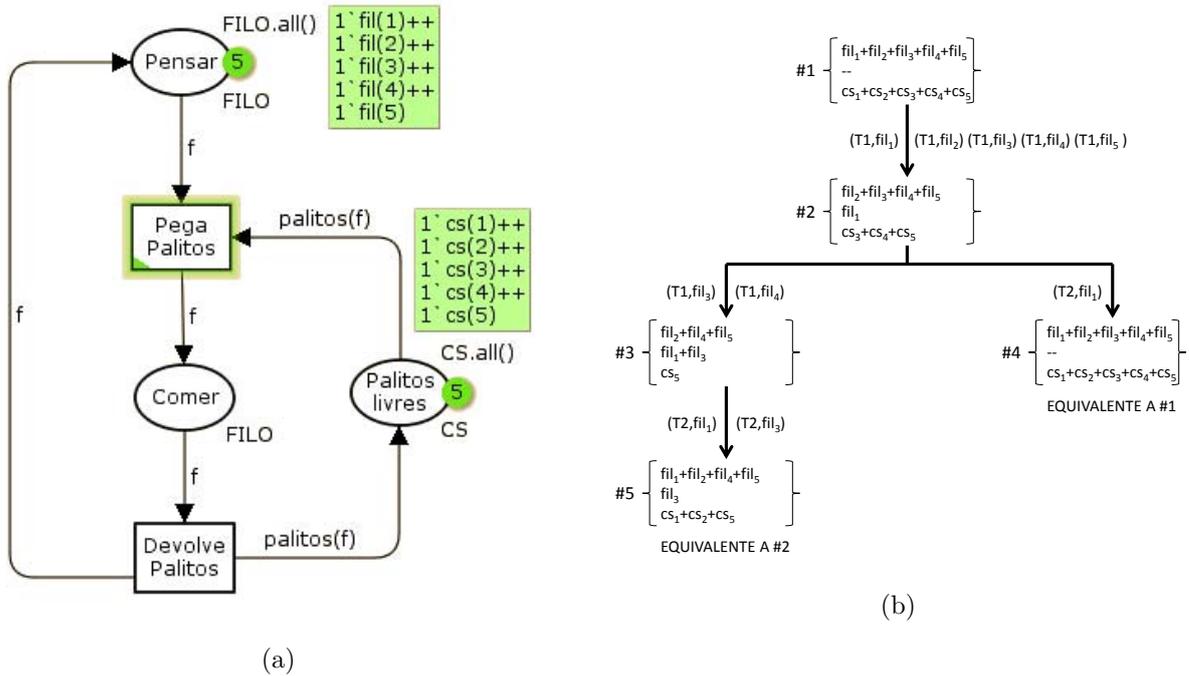


Figura 14 – Árvore de alcançabilidade do problema dos cinco filósofos: (a) representação em RdP colorida, (b) árvore gerada por marcações equivalentes (HUBER et al., 1986).

acumulada do sistema (que pode ser um evento controlável ou não controlável). Assim, caso algum estado seja ignorado na construção da árvore de alcançabilidade, certas ações podem não ser executadas durante o processamento do FMS.

O FMS é tratado neste trabalho como um sistema sequencial de processamento, ou seja, a partir do momento que uma peça entra na linha de manufatura, esta segue uma sequência de máquinas até o produto pronto ser retirado da linha. Assim, de forma a gerar a árvore de alcançabilidade da RdP que modela um certo FMS, foi desenvolvido um método de segregação da RdP para cada cor presente na modelagem. Na inicialização da RdP pelo algoritmo *gera_rpd* as matrizes de entrada e saída E e S são geradas na forma de “camadas” numa matriz tridimensional, de forma que a dinâmica de cada cor fique restrita a apenas uma dessas “camadas”. Estas matrizes podem ser representadas como $t \times l \times c$, onde t é a quantidade de transições, l são os lugares e c as cores da RdP. Assim, num FMS com dois tipos de produtos, A e B, a dinâmica do produto A na linha de produção fica restrita à matriz $t \times l \times 1$, enquanto a dinâmica do produto B fica restrita à $t \times l \times 2$. Com isso, é possível analisar a habilitação de todas as transições para todas as cores do sistema a partir de um estado inicial de referência. A segregação da RdP e a árvore de alcançabilidade da mesma são mostrados na figura 15. O próprio algoritmo se encarrega de verificar quais transições estão habilitadas para cada cor a partir do estado de referência, além de executar os disparos.

A figura 15a mostra uma RdP simples, com duas transições fontes, uma transição dreno e dois lugares. A transição $t1$ insere *tokens* do tipo B e $t2$ insere *tokens* do tipo A. Os

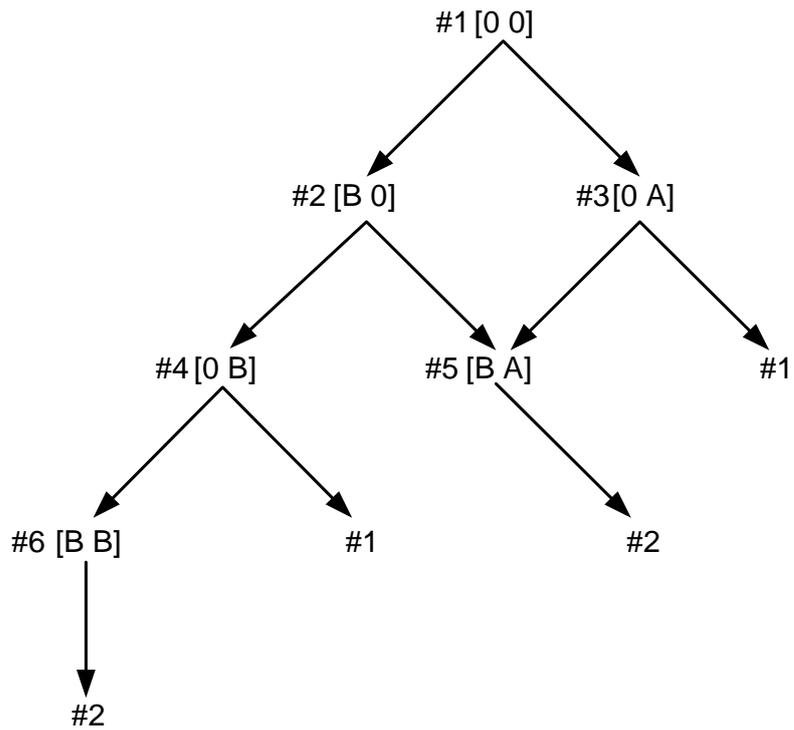
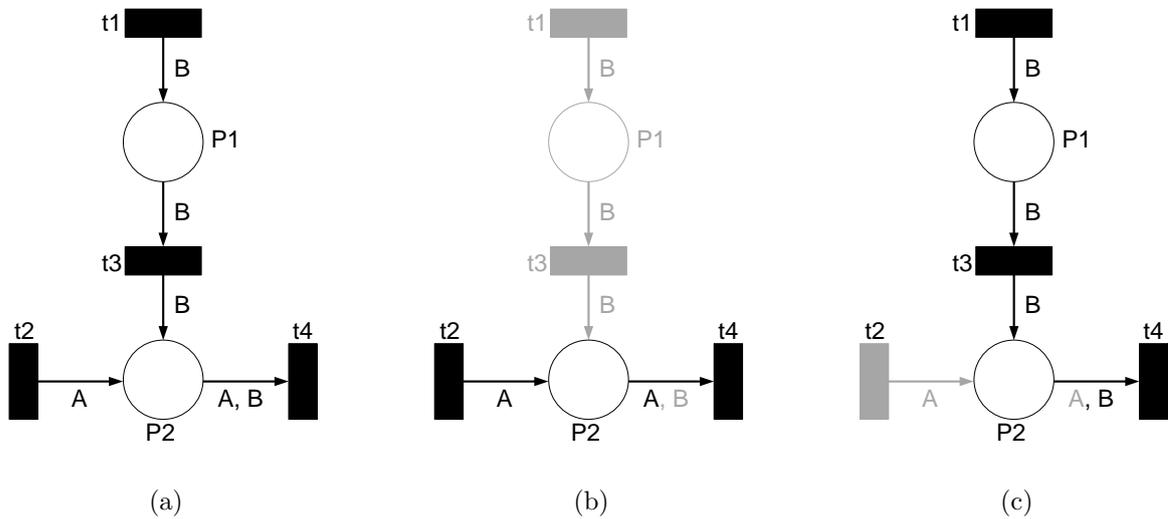


Figura 15 – Construção da árvore de alcançabilidade para RdP colorida: (a) RdP completa, (b) RdP segregada para cor A, (c) RdP segregada para cor B, (d) árvore resultante

tokens do tipo B passam pelos lugares $P1$ e $P2$, enquanto os *tokens* tipo A passam apenas pelo lugar $P2$ e ambos são retirados por $t4$. A segregação desta RdP consiste na geração de duas RdP individualizadas por cor. A figura 15b mostra a RdP original reduzida para *tokens* A e a figura 15c mostra reduzida para *tokens* B. A árvore de alcançabilidade total deste modelo é vista na figura 15d.

Durante a geração do vetor de estados através da construção da árvore de alcançabilidade outra importante tarefa também é executada. Com o auxílio das matrizes TA e $Temp$, são geradas as matrizes de transições P e de recompensas R que são parâmetros de entrada do MDP. Assim, partindo-se apenas da modelagem em RdP, pode-se calcular a política ótima para o sistema, que retorna qual ação a ser executada para cada estado do sistema.

3.2.2 Filas FIFO (*First-In-First-Out*)

Os FMS são tratados neste trabalho como processos sequenciais, onde existem entradas de peças (transições fontes), servidores (lugares representando as máquinas que processam as peças) e retiradas de peça (transições dreno). No caso de sistemas com mais de uma máquina, existem transições que garantem o sequenciamento do processo, ou seja, a passagem de subprodutos de uma máquina para outra. No entanto, as máquinas têm tempos de processamento diferentes para cada tipo de peça.

Seja considerado um FMS composto por duas máquinas, m e $m + 1$, para as quais o subproduto de m é entrada de $m + 1$. Caso a máquina m termine o processamento e a máquina $m + 1$ ainda esteja ocupada, m permanecerá ociosa, enquanto poderia estar processando outras peças. De forma a evitar que uma máquina permaneça ocupada após o término de processamento, é necessário que sejam considerados *buffers* entre as máquinas, que armazenam o subproduto de m enquanto $m+1$ ainda não terminou seu processamento. Além de armazenar subprodutos, é importante que estes depósitos sejam ordenados, isto é, preservem a ordem de chegada das peças. Nada adianta a tomada de decisão sobre qual peça produzir se a posição desta peça no sistema for perdida durante a sequência de processamento. Além disso, um desafio constante para a engenharia de produção no que diz respeito ao FMS é determinar um sequência de produção que atenda a certos requisitos (*scheduling*) (PINEDO, 2012). Portanto, a introdução do conceito de filas se faz necessária.

Filas são muito utilizadas em modelagem de sistemas de comunicação, onde servidores recebem diversos tipos de mensagem (BILLINGTON, 1991). Utilizando a mesma ideia, as máquinas do FMS recebem diversos tipos de subprodutos numa ordem bem definida determinada pela sequência de processamento de cada tipo de produto. Os dois tipos mais importantes de filas são:

- FIFO: *First-In-First-Out*
- LIFO: *Last-In-First-Out*

O modelo FIFO é o que melhor pode ser usado em FMS, pois preserva a sequência de peças. Como sugere a tradução literal do inglês, o primeiro item a entrar na fila é o primeiro a ser processado. O uso de filas FIFO em sistemas de eventos discretos é tão difundido que foi desenvolvido o conceito de redes FIFO (ROUCAIROL, 1987; MEMMI; FINKEL, 1985). Estes trabalhos definem uma rede FIFO como *FIFO Net* $N = (P, T, B, F, Q, M_0)$, onde:

- P : conjunto finito de lugares, também chamados filas;
- T : conjunto finito de transições;
- Q : alfabeto finito da fila;
- B e F : funções de mapeamento $P \times T \rightarrow Q$;
- M_0 : Marcação inicial.

A definição formal da rede FIFO é semelhante à definição da rede lugar-transição, porém existe uma particularidade com relação à regra de disparo. Enquanto nas redes lugar-transição uma transição qualquer retira *tokens* dos lugares pré-condição e insere *tokens* nos lugares pós-condição respeitando o conceito de multiplicidade, nas redes FIFO uma transição retira o item $B(p, t)$ localizado no topo da marcação $M(p)$ e acrescenta $F(p, t)$ no fim da fila resultante (ROUCAIROL, 1987). Isto é, uma transição retira o item que se encontra no início da fila de entrada, representada por um lugar pré-condição desta transição e o coloca no final da fila de saída, o lugar pós-condição.

Para a modelagem de filas FIFO no FMS foram, portanto, utilizados lugares limitados com certa capacidade k , que indica a capacidade de cada fila. A fim de se registrar a ordem da fila foi utilizada a metodologia de Billington (1991), que representa o conteúdo de uma fila através de uma cadeia de caracteres (*string*). Assim, ao chegar uma peça tipo A, esta fica armazenada no lugar que representa a fila. Após isso, caso uma peça B chegue à fila, a marcação do lugar é representada pela concatenação dos caracteres que representam estas duas peças, ou seja, AB (onde A ocupa a primeira e B a segunda posição da fila). Ao liberar o servidor, A é consumida e a marcação da fila passa a ser apenas B. O algoritmo *run_rdp* realiza os disparos e organiza as marcações nas filas automaticamente.

3.2.3 Cálculo da quantidade de estados do sistema

Ao se modelar um sistema de manufatura em RdP colorida, são estabelecidas quais cores e qual a quantidade de itens que podem ser armazenados nos lugares da rede. Para

o caso de lugares com capacidade igual a apenas um *token*, uma RdP com 10 lugares tem uma quantidade de estados igual a 3^{10} , assumindo as cores A e B. Isto significa que cada lugar da RdP pode assumir vazio, A e B. Em termos literais, pode-se deduzir a seguinte equação para determinação da quantidade de estados total produzida por um lugar de uma RdP:

$$S_{l_i} = (n_{c_i} + 1)^{n_pos_i} \quad (3.1)$$

onde S_{l_i} é a quantidade de estados gerada pelo lugar i , n_{c_i} é a quantidade de cores admitida no lugar i e n_pos_i é a capacidade do lugar. A quantidade total de estados de uma RdP é o produto da quantidade de estados de cada um dos lugares i da rede. No caso acima mostrado, para duas cores A e B ($n_{c_i} = 2$), uma rede composta de 10 lugares com capacidades $k = 1$ ($n_pos_i = 1$), a quantidade total de estados gerados pela rede é:

$$(2+1)^1 * (2+1)^1 * (2+1)^1 * (2+1)^1 * (2+1)^1 * (2+1)^1 * (2+1)^1 * (2+1)^1 * (2+1)^1 * (2+1)^1 = 3^{10}$$

Este cálculo é válido considerando a capacidade do lugar e todas as combinações possíveis dentro dele, não importando se este é ou não uma fila. Para filas FIFO, o cálculo fica mais complexo, uma vez que existem certos estados que não são atingíveis. Quando uma peça entra numa fila vazia, esta automaticamente assume a primeira posição da fila. Uma segunda peça assume a segunda posição da fila e assim por diante. Ao sair da fila, a primeira peça libera a posição um da fila e todas as outras peças assumem uma posição à frente. Assim, não faz sentido que sejam mapeados os estados onde há posições vagas à frente de outras peças na fila. A tabela 9 mostra todos os estados gerados por uma fila de três posições e cor A, com estados não atingíveis assinalados.

Tabela 9 – Estados gerados por uma fila de três posições

[0 0 0]	Atingível
[0 0 A]	Atingível
[0 A A]	Atingível
[A A A]	Atingível
[0 A 0]	Não Atingível
[A A 0]	Não Atingível
[A 0 A]	Não Atingível
[A 0 0]	Não Atingível

Para sistemas com poucos lugares e capacidades pequenas não é difícil calcular quantos estados são gerados na RdP. Porém, a situação muda quando se expande este sistema. Assim, para determinação da quantidade de estados do sistema, foi necessário determinar uma fórmula que retorne a quantidade de estados atingíveis no sistema. A expressão deduzida, para cada lugar da RdP é:

$$S_{l_i} = \sum_{j=0}^{n_pos_i} n_c_i^j \quad (3.2)$$

onde S_{l_i} é a quantidade de estados gerada pelo lugar i , n_c_i é a quantidade de cores admitida no lugar i e n_pos_i é a capacidade do lugar. A quantidade total de estados de uma RdP é novamente o produto de S_{l_i} para todos os lugares i da rede. Ao término da execução do algoritmo *run_rdp*, a quantidade de estados armazenados no vetor de estados do sistema deve ser a mesma calculada pela fórmula acima. A dedução da expressão 3.2 é encontrada no apêndice A.

A análise desta expressão indica um primeiro resultado deste trabalho, relativo à sensibilidade do sistema da forma com que este foi proposto. Nota-se que a quantidade de estados para um único lugar da RdP cresce de maneira exponencial, apenas aumentando de forma unitária a capacidade do mesmo. A figura 16 mostra a variação da quantidade de estados gerados por um lugar de uma RdP, considerando que o mesmo admita *tokens* de duas cores (A e B) e sua capacidade varie de uma a dez posições. Para capacidade unitária, o lugar poderá estar vazio ou conter uma unidade de A ou uma unidade de B, ou seja, são gerados 3 estados nesta condição. Caso seja considerada uma capacidade de dois tokens, cada uma das posições variará entre vazio, A e B, totalizando 9 estados diferentes, e assim por diante seguindo a equação 3.1. É necessário frisar que, num sistema mais complexo, a quantidade total de estados é dada pelo produto da quantidade de estados gerada por cada um dos lugares que compõe a RdP. Com isto, conclui-se que o sistema apresenta alta sensibilidade com relação à quantidade de estados. Variando-se de forma mínima a quantidade de lugares de uma RdP ou mesmo a capacidade de algum desses lugares, o aumento da quantidade de estados é significativa, o que interfere negativamente na eficiência computacional do sistema.

3.2.4 Função de utilidade

Na resolução do MDP é necessário que uma matriz de recompensas seja um parâmetro de entrada. Esta matriz relaciona um valor de recompensa para cada estado, dado que certa ação é tomada.

Por não se trabalhar explicitamente com tempo, convencionou-se que a recompensa está ligada ao tempo de produção de cada peça em cada servidor, informação dada pela matriz *Temp*. Esta matriz relaciona um valor de tempo para cada cor em todas as transições que executam a retirada de peça em um servidor. Outra forma de se considerar o tempo na resolução do MDP seria a utilização da teoria de TiMDP (BASTOS, 2010).

Os valores definidos na geração da RdP foram utilizados posteriormente na simulação com o *SimEvents*[®], sendo o tempo de utilização dos servidores para o processamento

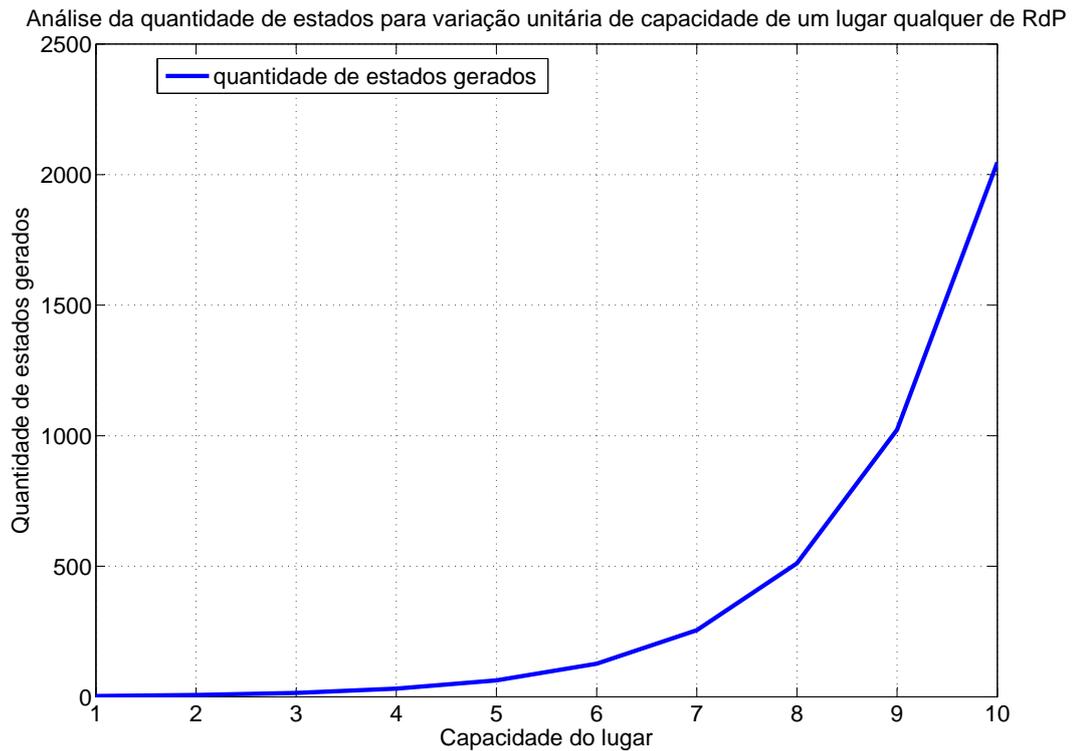


Figura 16 – Variação da quantidade de estados x capacidade do lugar

de cada tipo peça. Estes valores são atribuídos a cada entidade juntamente com sua cor pelo bloco *atributos*.

No entanto não se pode simplesmente utilizar o tempo como valor de recompensa. No caso do FMS o objetivo principal é produzir a maior recompensa total num determinado intervalo de tempo. O valor da recompensa total para cada peça produzida tem como base a recompensa calculada para cada máquina devido aos tempos de produção e o valor agregado a cada tipo de produto.

O valor agregado, como já apresentado, é apenas um peso que multiplica a recompensa total de cada tipo de produto. No entanto, para as recompensas dependentes do tempo, é natural que esta seja maior quanto menor seja o tempo de processamento. A relação entre o tempo de processamento e a recompensa é dada pela função de utilidade. Neste caso, a função de utilidade deve ser decrescente, ou seja, a recompensa diminui conforme maior é o tempo de processamento da peça. Qualquer função decrescente pode ser escolhida como função de utilidade, sendo muito utilizadas as funções exponenciais e lineares (BASTOS et al., 2011):

$$U(A, t) = U(A, t_0) \cdot e^{-k_1 \cdot t} \quad (3.3)$$

$$U(A, t) = U(A, t_0) - k_2 \cdot t \quad (3.4)$$

onde $U(A, t_0) \geq 0$ é a utilidade para a escolha da ação A no tempo t , t_0 é o tempo inicial e k_1 e k_2 são parâmetros de ajuste das funções exponencial e linear, respectivamente.

Para este trabalho foram definidas quatro funções de utilidade, a serem utilizadas para cada máquina nos problemas propostos. Cada máquina é associada a uma função de utilidade logo na parametrização da RdP. Assim, durante a execução do algoritmo que gera a árvore de alcançabilidade, para a criação da matriz R de recompensas do MDP é levado em consideração não apenas qual peça foi produzida (e conseqüentemente o tempo de processamento), mas também qual máquina foi utilizada, o que confere a cada processo uma utilidade única. Cada função de utilidade é aplicada por convenção neste trabalho a apenas uma máquina, ou seja, para todas as modelagens foi utilizada a mesma função para as máquinas 1, a mesma para as máquinas 2, assim por diante. No entanto, essas funções poderiam ter sido utilizadas aleatoriamente entre as máquinas. As quatro funções de utilidade aplicadas neste trabalho são encontradas abaixo e figura 17 mostra o comportamento da recompensa para cada uma das funções.

$$U(A, t) = 50 - (0,5 \cdot t) \quad (3.5)$$

$$U(A, t) = 50 \cdot e^{(-0,05 \cdot t)} \quad (3.6)$$

$$U(A, t) = 100 \cdot e^{(-0,0921 \cdot t)} \quad (3.7)$$

$$U(A, t) = 100 - (0,5 \cdot t) \quad (3.8)$$

3.2.5 Geração das matrizes de estados e recompensa do MDP

Tendo sido apresentado o procedimento de geração de estados da RdP e introduzido o conceito de função de utilidade, podem ser caracterizados os dois resultados mais importantes do algoritmo *run_rdp*: a matriz de transições $P(S \times S \times A)$ e a matriz de recompensa $R(S \times A)$. Para geração de ambas, o conceito da matriz transição/ação (TA), inicializada no algoritmo de geração da RdP, é muito importante.

Esta matriz relaciona cada transição da RdP a uma ação do MDP e a sua construção é exclusivamente manual, sendo que este deve ser um conhecimento prévio do problema. Como exemplo, pode-se ter uma RdP com n transições fontes que inserem um *token* na rede e podem ser caracterizadas como eventos controláveis. Além destas

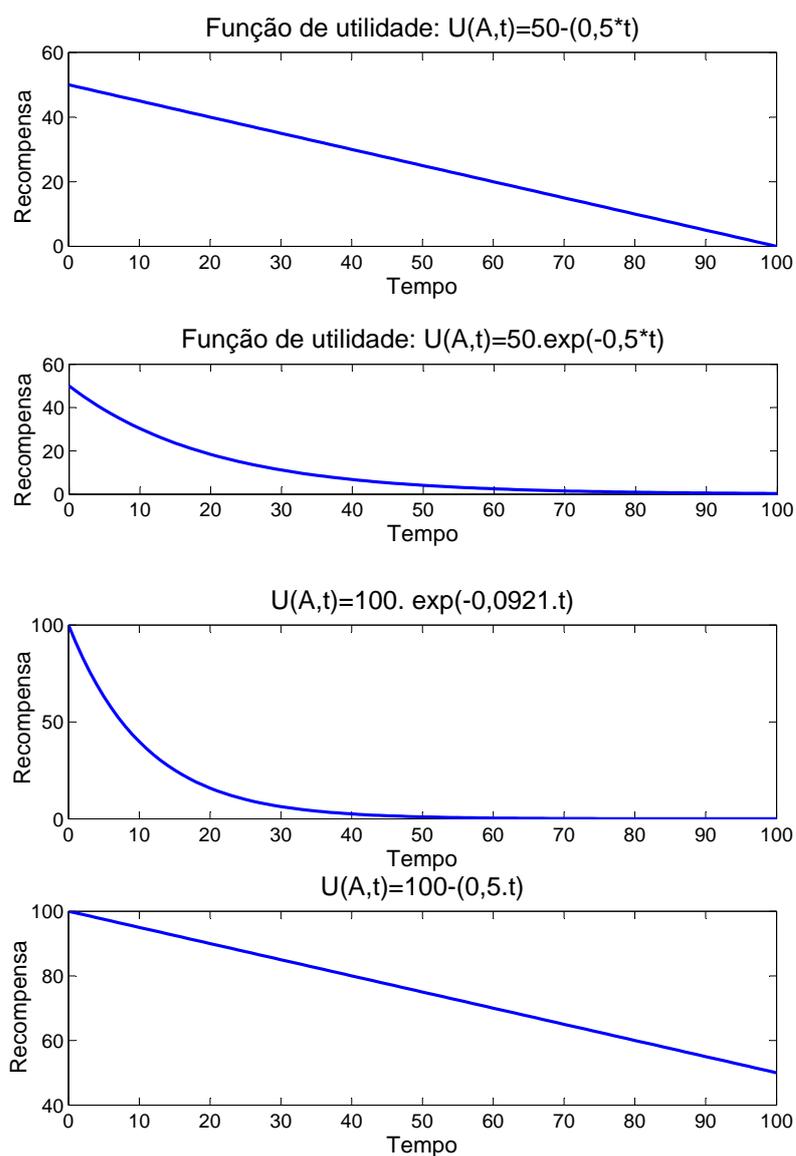


Figura 17 – Funções de utilidade para as máquinas do FMS

transições fontes, podem existir m transições inerentes ao processo, que são eventos não controláveis.

A atribuição de ações para as transições da RdP foi feita seguindo duas regras. A princípio cada transição que representa um evento controlável é considerada uma ação diferente do MDP, pois estas são as transições responsáveis por inserir um determinado tipo de produto na linha de produção. Feito isso, é necessário atribuir ações para cada um dos eventos não controláveis. Num primeiro momento, todas as transições não fontes representam a mesma ação, diferente daquelas atribuídas aos eventos controláveis. Esta é a ação “fazer nada”.

Na RdP que representa um FMS, após uma unidade de determinado produto ter sido inserida no início do processo por uma transição fonte, este segue sua sequência de processamento de acordo com a parametrização da rede. Isso quer dizer, não importa qual transição seja habilitada, um evento não controlável será executado até o disparo da transição dreno que retira o produto da linha. Assim, todas as transições que movimentam os recursos entre filas e servidores representam a mesma ação do MDP. A relação entre transições da RdP e as ações do MDP pode ser vista na figura 18 e tabela 10.

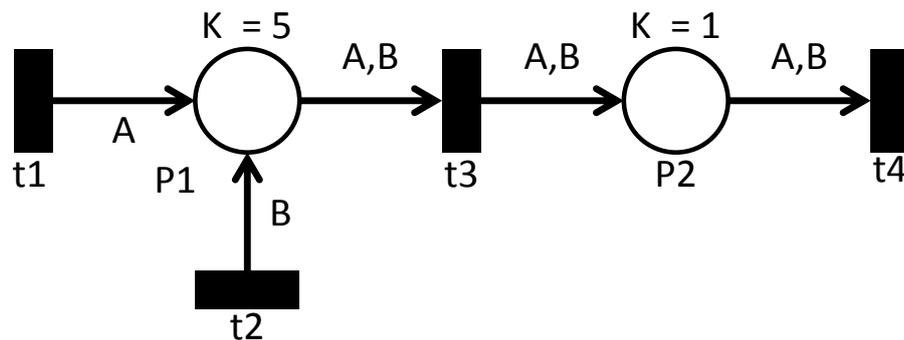


Figura 18 – RdP para demonstração da ações

Tabela 10 – Relação entre transições da RdP e ações do MDP, para eventos não controláveis representando ação “fazer nada”

Transição	Ação
t_1	1 (insere A)
t_2	2 (insere B)
t_3	3 (faz nada)
t_4	3 (faz nada)

Vale lembrar que entre transições presentes na RdP da figura 18, a única que tem valor de recompensa atribuído é a t_4 . Ou seja, para estados nos quais as transições t_3 e t_4 estejam habilitadas, caso t_3 seja disparada, a recompensa pela ação 3 será zero. Já, se t_4 é disparada, a ação 3 terá um valor não nulo de recompensa.

O fato de um estado ter mais de um valor de recompensa associado à mesma de ação acarreta um problema na resolução do MDP. Nesta situação, o *toolbox* utilizado para resolução do MDP realiza a média das recompensas atribuídas a cada uma das ações associadas a este estado, de forma a realizar uma ponderação entre as recompensas dos estado seguintes, sendo este valor a recompensa acumulada.

Seja s_1 um estado do MDP para o qual existam outros três estados mapeados na mesma ação “fazer nada”. Para duas transições há recompensas não nulas (com valores 10 e 5) e para a outra este valor é zero. Seja também outro estado s_2 na mesma situação, porém uma transição tem recompensa não nula (com valor 14) e as outras duas tem recompensa zero. Ambos estados são atingíveis a partir de um estado anterior s_0 , tal como a figura 19.

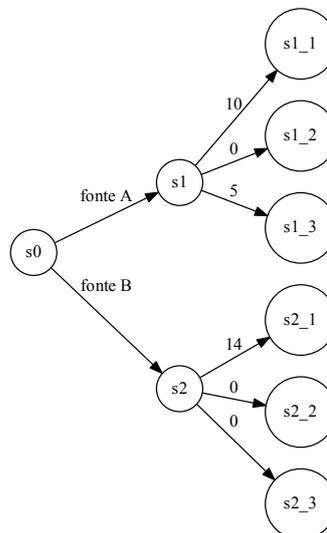


Figura 19 – Grafo de estados de um sistema para transições relacionadas à mesma ação e suas recompensas

Realizando a média dos valores de recompensa das transições associadas a s_1 e s_2 , as recompensas acumuladas serão 5 para s_1 e 4,67 para s_2 (figura 20).

Caso fosse resolvido o MDP, a ação que retorna maior valor de recompensa acumulada para o estado s_0 é a fonte A (valor da recompensa igual a 5). No entanto, analisando os estados posteriores a s_1 e s_2 , o maior valor de recompensa encontrado é para o estado s_{2_1} (igual a 14). Logo, conclui-se que a política foi calculada de forma incorreta, uma vez que a ação escolhida em s_0 deveria ter sido fonte B.

De forma a evitar que esta situação ocorra, foram identificados dois tipos de eventos não controláveis: eventos que movimentam um produto do *buffer* para a máquina (movimentadores), e eventos que retiram o produto das máquinas (finalizadores). Apenas este segundo tipo de evento não controlável terá um valor não nulo de recompensa associ-

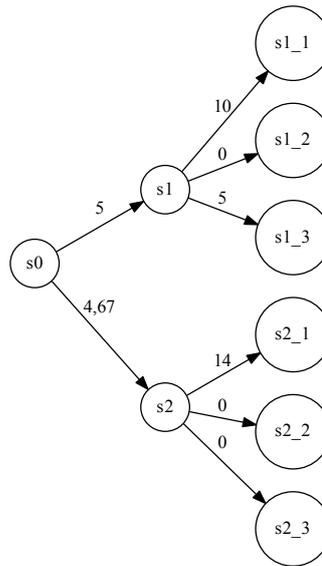


Figura 20 – Recompensas acumuladas para o estado s_0 , mesma ação para eventos não controláveis

ado. Assim, da mesma forma que foi feito para as transições fontes, cada um dos eventos finalizadores corresponderá a uma ação diferente do MDP, enquanto todos os eventos movimentadores representarão uma mesma ação, esta sim a “fazer nada”.

Desta forma, para um estado qualquer, caso várias ações “fazer nada” estejam mapeadas, o valor da recompensa acumulada para a média delas será sempre zero. Para o exemplo da figura 18, a nova relação entre as transições da RdP e ações do MDP é visto na tabela 11. O grafo da figura 21 mostra as recompensas acumuladas para as fontes A e B, sendo que agora a fonte B é a ação escolhida.

Tabela 11 – Nova relação entre transições da RdP e ações do MDP, limitando-se a ação “fazer nada”

Transição	Ação
t_1	1 (insere A)
t_2	2 (insere B)
t_3	3 (faz nada)
t_4	4 (retira de M1)

Esta modificação de interpretação da RdP influenciou diretamente na criação das matrizes P e R do MDP, pois ambas são geradas levando em consideração o total de ações. Da mesma forma que a construção da árvore de alcançabilidade, as matrizes P e R são geradas iterativamente, aproveitando a geração do vetor de estados da RdP.

Cada novo estado da RdP determinado pela árvore de alcançabilidade é adicionado ao vetor de estados em uma nova posição do vetor correspondente ao índice deste estado dentre o total gerado. Assim a marcação inicial da rede sempre será o primeiro estado do

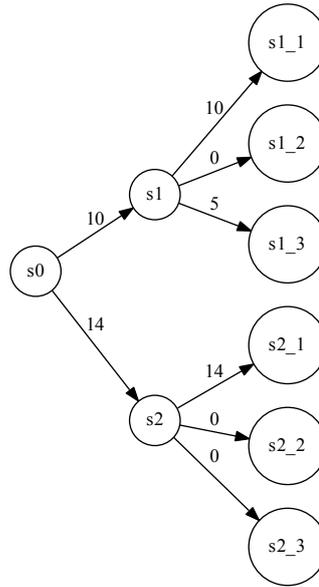


Figura 21 – Recompensas acumuladas para o estado s_0 , ações separadas para eventos não controláveis

sistema. Conforme um novo estado é adicionado ao vetor de estados, um laço verifica qual é o índice do estado predecessor, em qual índice do vetor de estados da RdP foi adicionado o novo estado e qual transição foi disparada para geração do novo estado. Assim, tendo-se qual o estado anterior, qual o novo estado, e buscando qual ação corresponde a esta transição na matriz TA , pode-se gerar a matriz de transição do MDP ($P(S \times S \times A)$).

A geração da matriz $R(S \times A)$ de recompensas é feita de forma semelhante. Primeiramente os valores de tempos de produção inseridos na matriz $Temp$ são convertidos em valores de recompensa segundo as funções de utilidade adotadas. Feito isso, caso um novo estado seja encontrado na execução do algoritmo, e após ser calculada sua contribuição na matriz P , sua recompensa é processada para a matriz R . Para o cálculo de $R(S \times A)$, primeiramente é verificado qual o índice do estado tomado como referência e qual ação foi tomada para geração do novo estado armazenado. Como as ações estão relacionadas com as transições da RdP pela matriz TA , e sabendo-se qual a cor está sendo processada, busca-se o valor de recompensa convertido pelas funções de utilidade.

Após a execução do algoritmo run_mdp , e determinadas as matrizes de estados $P(S \times S \times A)$ e de recompensa $R(S \times A)$, foi executado o algoritmo de iteração de valor no $MDPToolbox^{\text{®}}$ (CHADÈS et al., 2009) para a cálculo da política ótima. No entanto, qualquer outra plataforma poderia ser utilizada para determinação da política.

3.3 Exemplos de aplicação

De forma a compilar as informações descritas e facilitar o entendimento do algoritmo são propostos alguns exemplos de aplicação. Estes exemplos tratam de RdP muito simples e são apresentados todos os parâmetros de entrada. São calculadas as políticas para alguns valores de desconto do MDP, de forma a mostrar a variação das mesmas.

3.3.1 Exemplo 1: Sistema FIFO/servidor, fila de 1 posição, peças A e B

O primeiro exemplo de aplicação trata-se de uma RdP num sistema fila/servidor composto de quatro transições e dois lugares. A capacidade da fila (lugar $P1$) é uma unidade, são introduzidas peças de tipo A e B por transições fontes e ambas são processadas no servidor $P2$. O diagrama de blocos da figura 22 resume o sistema, detalhado pela RdP e seus parâmetros na figura 23. A figura 24 mostra o grafo de estados deste exemplo.

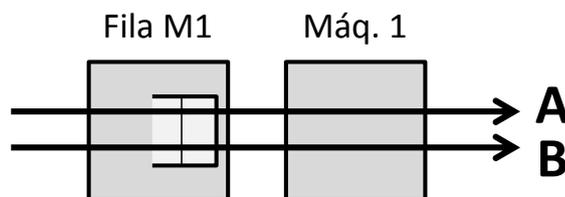


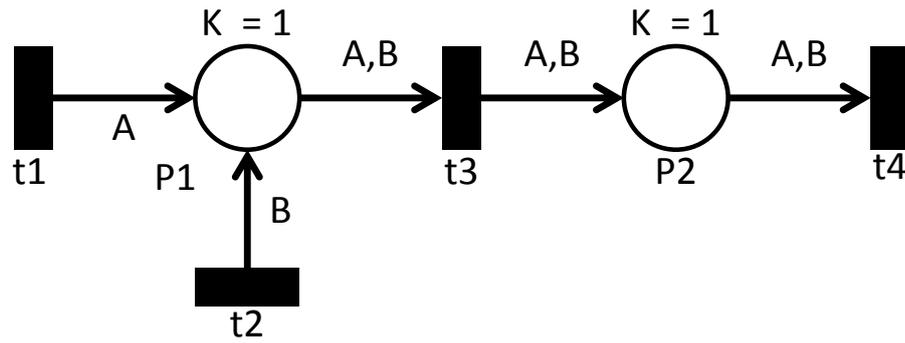
Figura 22 – Diagrama de blocos do exemplo de aplicação 1

A tabela 12 mostra a composição de cada estado do sistema, segundo sua árvore de alcançabilidade.

Tabela 12 – Estados do sistema (exemplo de aplicação 1)

Estado	Fila	Máquina
s_1	0	0
s_2	1	0
s_3	2	0
s_4	0	1
s_5	0	2
s_6	1	1
s_7	2	1
s_8	1	2
s_9	2	2

Para análise dos resultados foram executados os algoritmos de forma a determinar as políticas ótimas variando-se o fator de desconto (γ). Como pode ser observado na tabela $Temp$, a transição t_4 é aquela que retira peças da máquina (lugar $P2$), isto é, esta transição é que completa o processamento. Os valores de tempos de uso do servidor para as cores A e B são respectivamente 45 e 60 unidades de tempo. Vale lembrar que apenas



$n_color = 2$

$$M = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \quad E_A = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \quad S_A = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \quad E_B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 2 & 0 \\ 0 & 2 \end{bmatrix} \quad S_B = \begin{bmatrix} 0 & 0 \\ 2 & 0 \\ 0 & 2 \\ 0 & 0 \end{bmatrix} \quad TA = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

$$K = \begin{bmatrix} 1 & 1 \end{bmatrix}$$

$$Temp = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 45 & 60 \end{bmatrix}$$

Figura 23 – Rede de Petri e parâmetros do sistema (exemplo de aplicação 1)

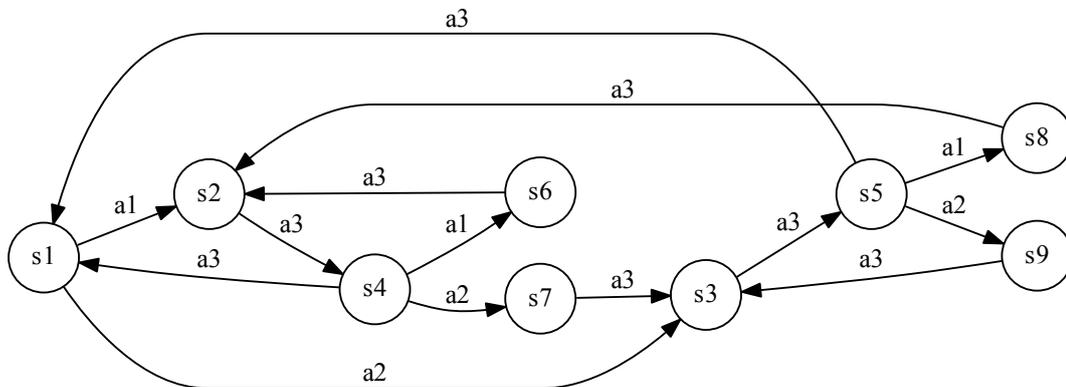


Figura 24 – Grafo de estados do sistema (exemplo de aplicação 1)

esta transição gera recompensas para este sistema, pois apenas $t4$ retira peças concluídas nesta linha de produção. Para os exemplos de aplicação, o vetor de valor agregado é considerado unitário. Conforme exposto anteriormente, a função de utilidade é aquela que define as recompensas no MDP, e da forma com que esta foi usada, a recompensa tende a ser mais baixa conforme maior é o tempo de processamento. Assim, considerando os valores de $Temp$ para $t4$ e a função de utilidade decrescente, as recompensas neste caso

são encontradas na tabela 13.

Tabela 13 – Recompensas geradas pela transição $t4$

	Peça tipo A	Peça tipo B
Recompensa transição $t4$	27,5	20

A distribuição de recompensas para todos os estados do sistema, junto da função valor para cada um deles é encontrada na tabela 14.

Tabela 14 – Distribuição de recompensas para todos os estados do sistema (exemplo 1)

Recompensas					
Estado	$a1$	$a2$	$a3$	$a4$	Função valor
$s1$	0	0	–	–	$V0(s1) = 0$
$s2$	–	–	0	–	$V0(s2) = 0$
$s3$	–	–	0	–	$V0(s3) = 0$
$s4$	0	0	–	27,5	$V0(s4) = 27,5$
$s5$	0	0	–	20	$V0(s5) = 20$
$s6$	–	–	–	27,5	$V0(s6) = 27,5$
$s7$	–	–	–	27,5	$V0(s7) = 27,5$
$s8$	–	–	–	20	$V0(s8) = 20$
$s9$	–	–	–	20	$V0(s9) = 20$

Para o exemplo de aplicação 1, devido seu tamanho muito reduzido, percebe-se que o algoritmo tende a escolher a ação 1, ou seja, insere a peça tipo A através da transição $t1$, pois esta tem menor tempo de processamento no servidor e por isso sua recompensa é maior. A figura 25 mostra a modelagem do sistema em *SimEvents*[®] e a tabela 15 mostra o resultado das políticas.

Tabela 15 – Comparativo de políticas para o exemplo de aplicação 1

Recompensas						
Estado	$\gamma = 0,01$	$\gamma = 0,1$	$\gamma = 0,3$	$\gamma = 0,5$	$\gamma = 0,8$	$\gamma = 0,95$
$s1$	$a1$	$a1$	$a1$	$a1$	$a1$	$a1$
$s2$	$a3$	$a3$	$a3$	$a3$	$a3$	$a3$
$s3$	$a3$	$a3$	$a3$	$a3$	$a3$	$a3$
$s4$	$a4$	$a4$	$a4$	$a4$	$a4$	$a4$
$s5$	$a4$	$a4$	$a4$	$a4$	$a4$	$a4$
$s6$	$a4$	$a4$	$a4$	$a4$	$a4$	$a4$
$s7$	$a4$	$a4$	$a4$	$a4$	$a4$	$a4$
$s8$	$a4$	$a4$	$a4$	$a4$	$a4$	$a4$
$s9$	$a4$	$a4$	$a4$	$a4$	$a4$	$a4$
<i>iterações</i>	4	6	10	16	48	229

De forma a verificar se o comportamento do sistema segue o padrão, foram invertidos na matriz *Temp* os valores correspondentes aos tempos de processamento dos dois tipos de peça e foram repetidos os experimentos para os mesmos valores de γ . O resultado

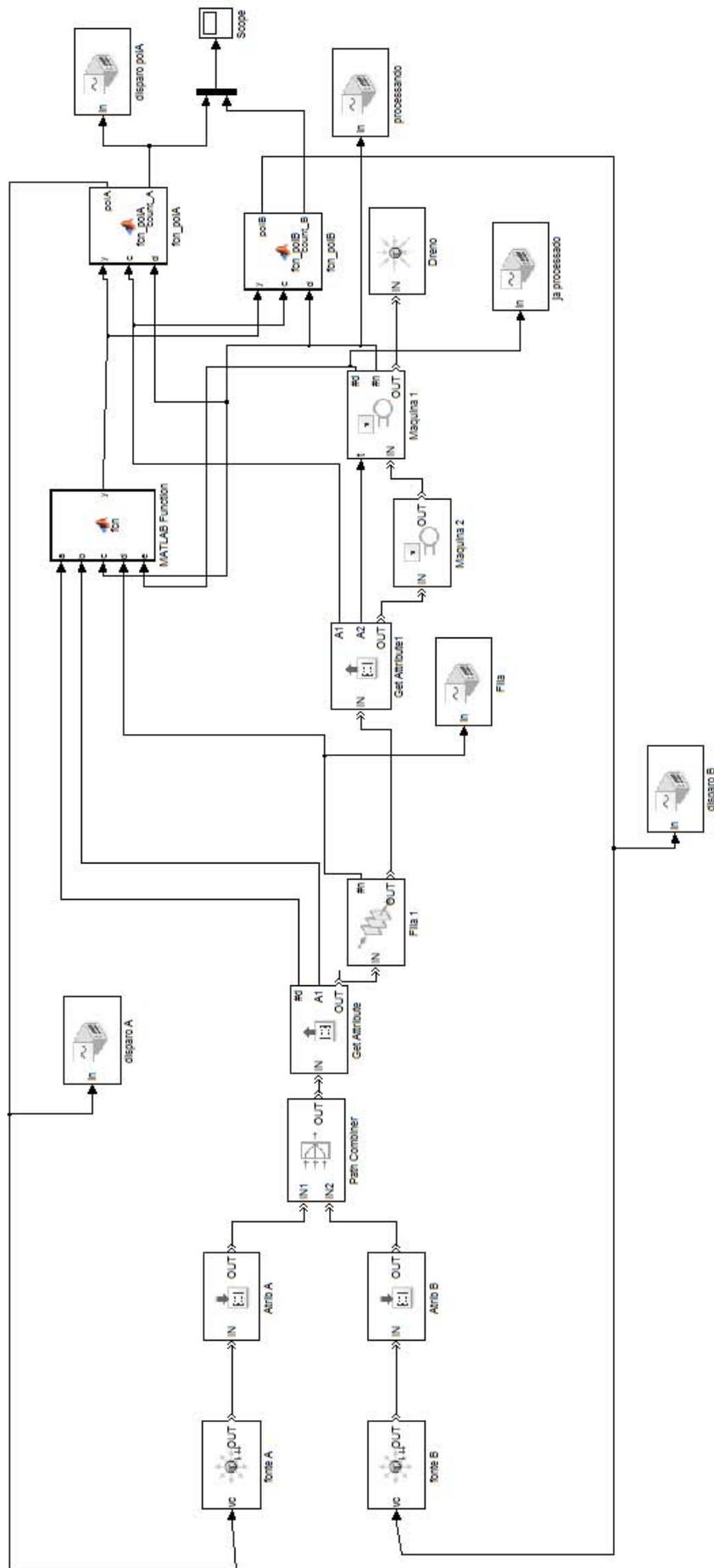


Figura 25 – Modelo do exemplo de aplicação 1 em *SimEvents*®

pode ser visto na tabela 16. Como esperado, o comportamento do sistema com os valores de recompensa invertidos foi exatamente o mesmo do primeiro exemplo, porém com ações invertidas. No exemplo de aplicação 2 a seguir é adicionada mais uma posição para a fila, a análise segue da mesma forma.

Tabela 16 – Comparativo de políticas para o exemplo de aplicação 1 com recompensas invertidas

Estado	Recompensas					
	$\gamma = 0,01$	$\gamma = 0,1$	$\gamma = 0,3$	$\gamma = 0,5$	$\gamma = 0,8$	$\gamma = 0,95$
s1	a2	a2	a2	a2	a2	a2
s2	a3	a3	a3	a3	a3	a3
s3	a3	a3	a3	a3	a3	a3
s4	a4	a4	a4	a4	a4	a4
s5	a4	a4	a4	a4	a4	a4
s6	a4	a4	a4	a4	a4	a4
s7	a4	a4	a4	a4	a4	a4
s8	a4	a4	a4	a4	a4	a4
s9	a4	a4	a4	a4	a4	a4
iterações	4	6	10	16	48	229

A execução de sistemas deste tipo para verificação da política, no entanto, fica muito limitada a apenas poucos lugares e filas com pequena capacidade devido à explosão de estados. No exemplo 1, para apenas um servidor e fila com uma posição, o sistema tem 9 estados. Ao se aumentar apenas uma posição na fila, como foi feito no exemplo de aplicação 2 abaixo, o número de estados sobe para 21. Caso um terceiro exemplo fosse executado seguindo a mesma lógica, 3 posições na fila, a quantidade de estados saltaria para 45.

3.3.2 Exemplo 2: Sistema FIFO/servidor, fila de 2 posições, peças A e B

O exemplo de aplicação 2 é uma variação do primeiro exemplo, com o aumento da capacidade da fila para duas unidades. A ideia principal se mantém a mesma. A figura 26 mostra o diagrama de blocos, a figura 27 apresenta a RdP da modelagem e as matrizes de parametrização e o grafo de estados é encontrado na figura 28.

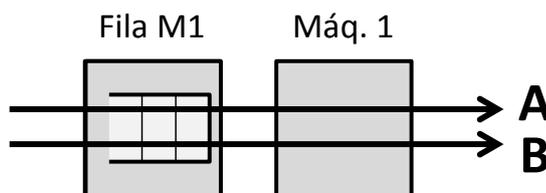
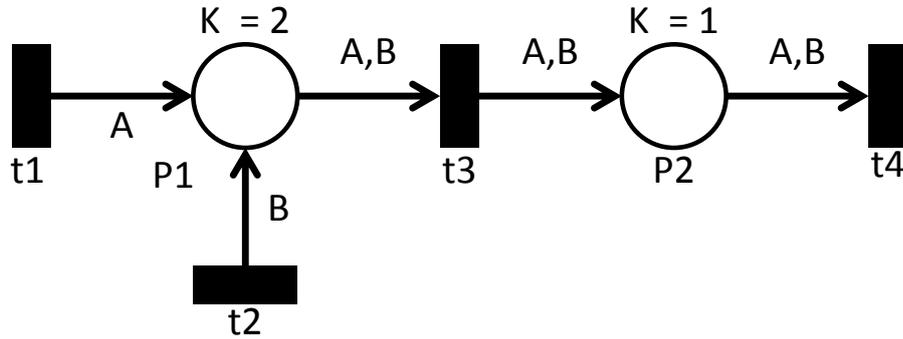


Figura 26 – Diagrama de blocos do exemplo de aplicação 2

A tabela 17 mostra a composição de cada estado do sistema, segundo sua árvore de alcançabilidade. Conforme já apresentado, este sistema com 2 posições na fila possui um



n_color= 2

$$M = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \quad E_A = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \quad S_A = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \quad E_B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 2 & 0 \\ 0 & 2 \end{bmatrix} \quad S_B = \begin{bmatrix} 0 & 0 \\ 2 & 0 \\ 0 & 2 \\ 0 & 0 \end{bmatrix} \quad TA = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

K= [2 1]

$$Temp = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 45 & 60 \end{bmatrix}$$

Figura 27 – Rede de Petri e parâmetros do sistema (exemplo de aplicação 2)

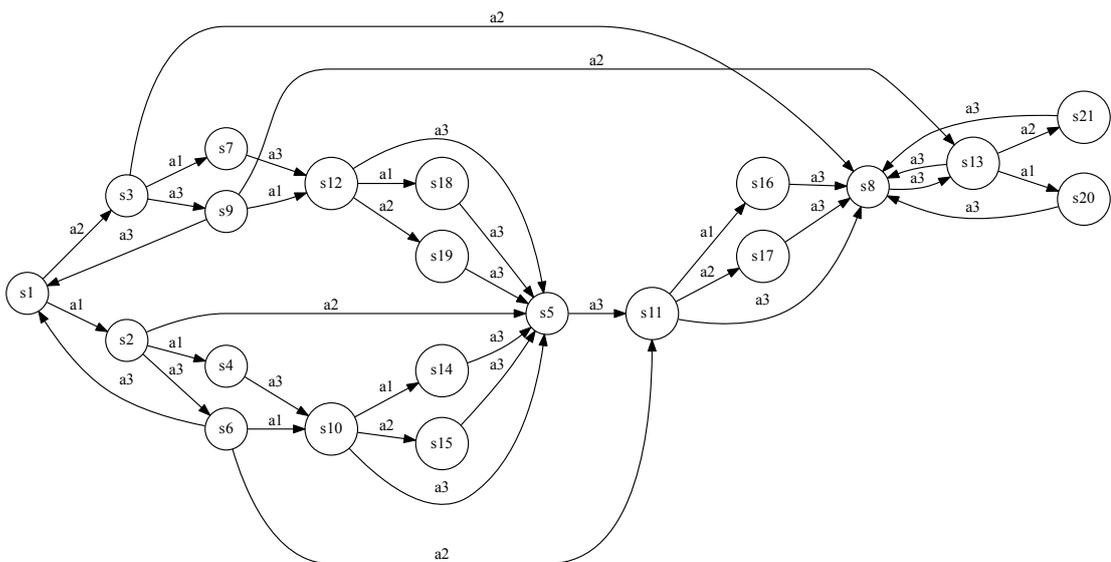


Figura 28 – Grafo de estados do sistema (exemplo de aplicação 2)

Tabela 17 – Estados do sistema (exemplo de aplicação 2)

Estado	Fila	Máquina
s1	[0 0]'	0
s2	[1 0]'	0
s3	[2 0]'	0
s4	[1 1]'	0
s5	[1 2]'	0
s6	[0 0]'	1
s7	[2 1]'	0
s8	[2 2]'	0
s9	[0 0]'	2
s10	[1 0]'	1
s11	[2 0]'	1
s12	[1 0]'	2
s13	[2 0]'	2
s14	[1 1]'	1
s15	[1 2]'	1
s16	[2 1]'	1
s17	[2 2]'	1
s18	[1 1]'	2
s19	[1 2]'	2
s20	[2 1]'	2
s21	[2 2]'	2

total de 21 estados em comparação aos 9 estados do exemplo de aplicação 1. A quantidade total de estados cresce exponencialmente seguindo a equação 3.2.

Da mesma forma como no exemplo de aplicação 1, foi feita a distribuição de recompensas em todos os estados (tabela 18) e a análise de políticas calculadas para este sistema variando-se o valor de γ (tabela 19). Analisando as políticas calculadas para diversos valores do fator de desconto, percebe-se que para os casos de $\gamma = 0,01$, $\gamma = 0,1$, $\gamma = 0,3$ e $\gamma = 0,5$, as ações tomadas são as mesmas, apenas no estado inicial. No entanto, para $\gamma = 0,8$ e $\gamma = 0,95$, a ação tomada passa a ser escolhida em outros estados, prevendo que filas sejam ocupadas que não seja esperado o fim de processamento para inserção de novo produto na fila.

Caso seja repetida a inversão dos valores de recompensa para a produção das duas peças, as ações escolhidas também são invertidas, da mesma forma observada no exemplo de aplicação 1 modificado. Neste caso, no entanto, o sistema passa a considerar a decisão de inserir peças na fila a partir do fator de desconto $\gamma = 0,8$ (tabela 20).

Tendo sido feitas estas análises, pode-se enfim partir para o objetivo principal deste trabalho: a tomada de decisões em tempo real para FMS. No próximo capítulo são feitas simulações completas de algumas classes de FMS, partindo da modelagem em RdP e finalizando com a análise de resultados do modelo em *SimEvents*[®].

Tabela 18 – Distribuição de recompensas para todos os estados do sistema (exemplo 2)

Recompensas					
Estado	$a1$	$a2$	$a3$	$a4$	Função valor
$s1$	0	0	–	–	$V0(s1) = 0$
$s2$	0	0	0	–	$V0(s2) = 0$
$s3$	0	0	0	–	$V0(s3) = 0$
$s4$	–	–	0	–	$V0(s4) = 0$
$s5$	–	–	0	–	$V0(s5) = 0$
$s6$	0	0	–	27,5	$V0(s6) = 27,5$
$s7$	–	–	0	–	$V0(s7) = 0$
$s8$	–	–	0	–	$V0(s8) = 0$
$s9$	0	0	–	20	$V0(s9) = 20$
$s10$	0	0	–	27,5	$V0(s10) = 27,5$
$s11$	0	0	–	27,5	$V0(s11) = 27,5$
$s12$	0	0	–	20	$V0(s12) = 20$
$s13$	0	0	–	20	$V0(s13) = 20$
$s14$	–	–	–	27,5	$V0(s14) = 27,5$
$s15$	–	–	–	27,5	$V0(s15) = 27,5$
$s16$	–	–	–	27,5	$V0(s16) = 27,5$
$s17$	–	–	–	27,5	$V0(s17) = 27,5$
$s18$	–	–	–	20	$V0(s18) = 20$
$s19$	–	–	–	20	$V0(s19) = 20$
$s20$	–	–	–	20	$V0(s20) = 20$
$s21$	–	–	–	20	$V0(s21) = 20$

Tabela 19 – Comparativo de políticas para o exemplo de aplicação 2

Estado	Recompensas					
	$\gamma = 0,01$	$\gamma = 0,1$	$\gamma = 0,3$	$\gamma = 0,5$	$\gamma = 0,8$	$\gamma = 0,95$
s1	a1	a1	a1	a1	a1	a1
s2	a3	a3	a3	a3	a1	a1
s3	a3	a3	a3	a3	a1	a1
s4	a3	a3	a3	a3	a3	a3
s5	a3	a3	a3	a3	a3	a3
s6	a4	a4	a4	a4	a1	a1
s7	a3	a3	a3	a3	a3	a3
s8	a3	a3	a3	a3	a3	a3
s9	a4	a4	a4	a4	a1	a1
s10	a4	a4	a4	a4	a4	a4
s11	a4	a4	a4	a4	a4	a4
s12	a4	a4	a4	a4	a4	a4
s13	a4	a4	a4	a4	a4	a4
s14	a4	a4	a4	a4	a4	a4
s15	a4	a4	a4	a4	a4	a4
s16	a4	a4	a4	a4	a4	a4
s17	a4	a4	a4	a4	a4	a4
s18	a4	a4	a4	a4	a4	a4
s19	a4	a4	a4	a4	a4	a4
s20	a4	a4	a4	a4	a4	a4
s21	a4	a4	a4	a4	a4	a4
iterações	4	6	10	16	48	223

Tabela 20 – Comparativo de políticas para o exemplo de aplicação 2 com recompensas invertidas

Estado	Recompensas					
	$\gamma = 0,01$	$\gamma = 0,1$	$\gamma = 0,3$	$\gamma = 0,5$	$\gamma = 0,8$	$\gamma = 0,95$
s1	a2	a2	a2	a2	a2	a2
s2	a3	a3	a3	a3	a2	a2
s3	a3	a3	a3	a3	a2	a2
s4	a3	a3	a3	a3	a3	a3
s5	a3	a3	a3	a3	a3	a3
s6	a4	a4	a4	a4	a2	a2
s7	a3	a3	a3	a3	a3	a3
s8	a3	a3	a3	a3	a3	a3
s9	a4	a4	a4	a4	a2	a2
s10	a4	a4	a4	a4	a4	a4
s11	a4	a4	a4	a4	a4	a4
s12	a4	a4	a4	a4	a4	a4
s13	a4	a4	a4	a4	a4	a4
s14	a4	a4	a4	a4	a4	a4
s15	a4	a4	a4	a4	a4	a4
s16	a4	a4	a4	a4	a4	a4
s17	a4	a4	a4	a4	a4	a4
s18	a4	a4	a4	a4	a4	a4
s19	a4	a4	a4	a4	a4	a4
s20	a4	a4	a4	a4	a4	a4
s21	a4	a4	a4	a4	a4	a4
iterações	4	6	10	16	48	223

4 Simulações e resultados

4.1 Ambiente e métodos de simulação

Como exposto no capítulo anterior, os FMS propostos foram simulados usando o *software SimEvents*[®], pacote do *Matlab SimEvents*[®]. Este software contém todos os elementos necessários para caracterização do FMS, além de ser possível fazer um paralelo com os elementos da RdP. Cada simulação foi executada num intervalo de tempo igual a 1000 ciclos do *Simulink*[®], representando turnos genéricos. Apesar de ser fixado um número de ciclos de execução da simulação, o horizonte do MDP em questão é infinito, uma vez que as decisões podem ser tomadas repetidamente e não há, a priori, um estado considerado final.

No *SimEvents*[®] o item que processado é chamado de entidade, sendo que pode-se adicionar a este item quaisquer atributos que sejam necessários. Um atributo é qualquer informação que caracteriza a entidade, como por exemplo, a cor do *token* na RdP colorida ou o tempo de processamento deste tipo de produto numa determinada máquina.

Outro conceito importante que foi configurado para a simulação é a época de decisão, isto é, em qual momento o MDP observa o sistema e executa a ação definida pela política. A tomada de decisão vai ter influência apenas nos eventos controláveis, aqueles que inserem um tipo de produto no sistema. Ou seja, apenas as transições modeladas como fonte pela RdP deverão ser consideradas pelo MDP. Para as simulações foi adotada a época de decisão em passos a cada 0,2 unidades de tempo de simulação do *Simulink*[®]. Com isso, tem-se um intervalo pequeno suficiente para que não deixe de ser observado nenhum estado do FMS durante a simulação.

Tendo sido determinadas as matrizes de transição e recompensas para um FMS qualquer e variando-se o fator de desconto de forma a avaliar o comportamento do sistema nestas situações, pode-se realizar a simulação. No entanto, é necessário que *Simulink*[®] tenha acesso aos estados do sistema e à política calculada, para então poder simular a tomada de decisão em tempo real. Um algoritmo auxiliar filtra as ações relativas às transições fontes e gera matrizes de dados segregadas contendo apenas os estados para os quais cada uma destas ações devem ser tomadas. Este artifício foi a solução encontrada para minimizar o tempo de varredura de estados enquanto o FMS está em execução. Assim, em cada época de decisão, ao invés do sistema analisar todos os estados individualmente, verificar qual deles é o estado atual do FMS e então executar a ação relativa a

este estado, é verificada apenas uma pequena lista contendo os estados que geram disparo desta ação. Considerando que a maioria das ações do sistema consiste da movimentação *buffer*/máquina e retirada de peça pronta da máquina, grande parte dos estados gerados pela RdP será ignorada pelo sistema.

A figura 29 mostra um exemplo geral de um FMS, um *flowshop* com três máquinas e três tipos de produtos. O processamento das peças é sequencial, seguindo o arranjo máquina 1 – máquina 2 – máquina 3. Nela podem ser identificados os blocos fonte A, fonte B e fonte C, que são geradores de entidade baseados em eventos. A porta de habilitação para cada um destes blocos está ligada na saída das funções *fcn_polA*, *fcn_polB* e *fcn_polC*. Estas funções fazem a leitura das matrizes segregadas entre as ações e observam o estado do sistema, verificando o *status* de cada fila e cada máquina. Com isso, cada função faz a comparação entre o estado atual do FMS e a lista de estados para o qual sua ação é executada. Caso a comparação seja satisfeita, a saída da função habilita o bloco gerador que insere uma unidade deste tipo de produto na linha. No FMS modelado em *SimEvents*[®] existe um bloco gerador de entidade baseado em eventos e uma função *fcn_pol* para cada cor representada na RdP, ou seja, cada tipo de produto manufaturado nesta linha.

Feito o disparo de determinada fonte, existe uma rota determinada para que seja cumprido seu processamento. No caso dos *flowshops*, como visto anteriormente, a sequência de processamento não sofre variações entre as famílias de produtos, isto é, todos seguem a mesma rota de produção. Já no caso dos *jobshops*, elementos de roteamento devem ser inseridos de forma a levar cada produto para a máquina certa, dependendo de sua sequência de manufatura.

Foi criado um arranjo para representar cada máquina no FMS do *SimEvents*[®], uma vez que o bloco *servidor*, individualmente, não é capaz de fornecer todos os parâmetros de observação do estado. Toda máquina é composta basicamente de uma fila, um servidor e uma função fila. Dois blocos *get attributes* são utilizados para determinar algumas características do item que chega à máquina.

A função *Fcn_Fila* é responsável por determinar o estado das filas, uma vez que não é possível observar diretamente do bloco. Isto é importante, pois, no estado geral do FMS, os lugares que representam as filas são caracterizados não apenas pela quantidade de produtos na fila, mas também pelos tipos e a ordem que se encontram na fila. Seja um FMS composto de uma fila com duas posições, uma máquina e que produza peças tipo A e B. Um estado deste sistema que tenha máquina vazia e a sequência [A B] na fila é diferente de outro estado que tenha máquina vazia e sequência [B A] na fila. Além dos dois estados não serem funcionalmente iguais, cada um pode apresentar diferentes recompensas para uma mesma ação executável a partir deles. Ou seja, caso a ordem da fila não fosse levada em consideração, a tomada de decisão poderia ser feita de forma

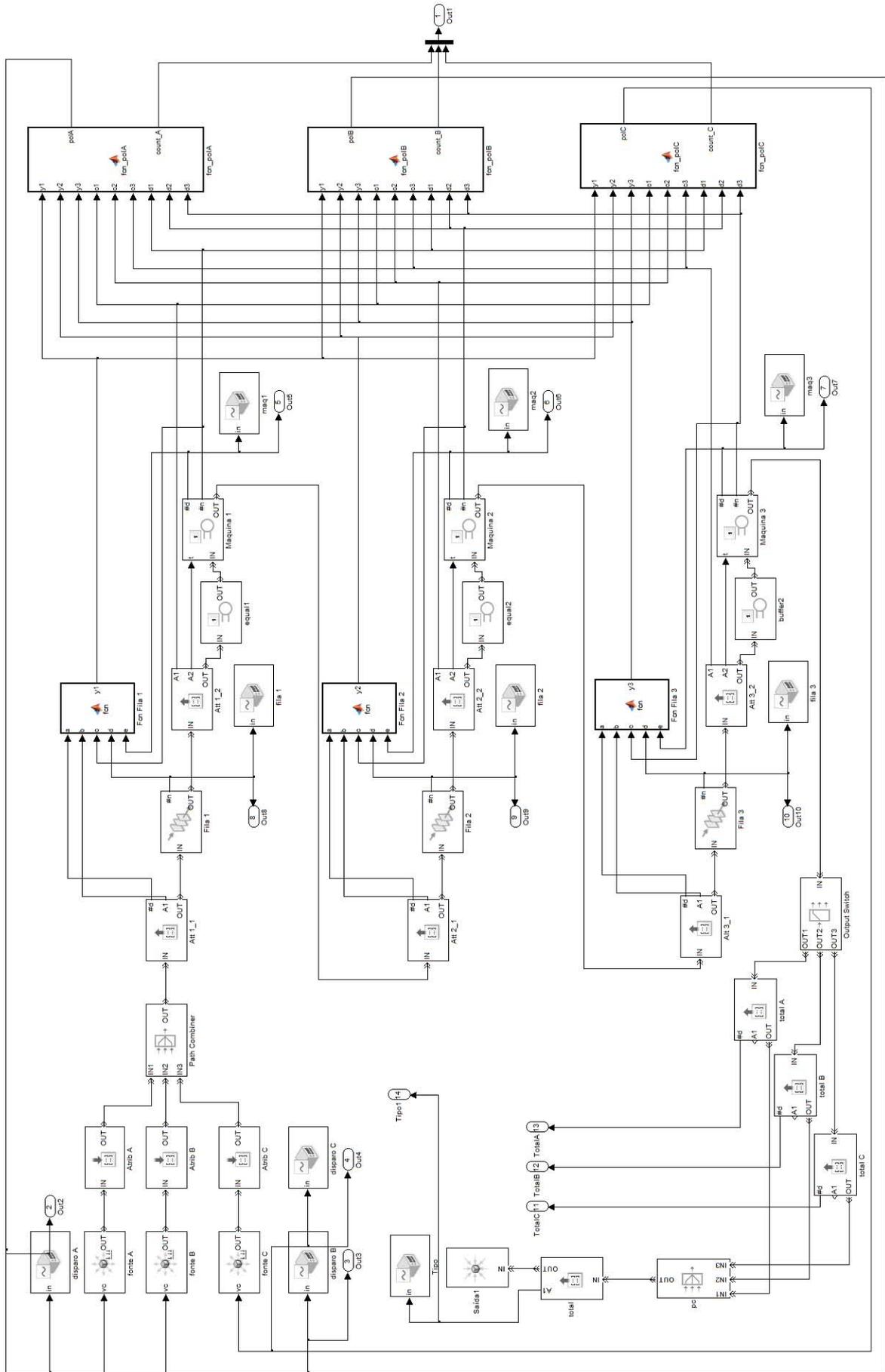


Figura 29 – FMS geral (flowshop)

errada, já que a recompensa tomada para o cálculo do MDP estaria incorreta. A figura 30 mostra uma máquina do FMS.

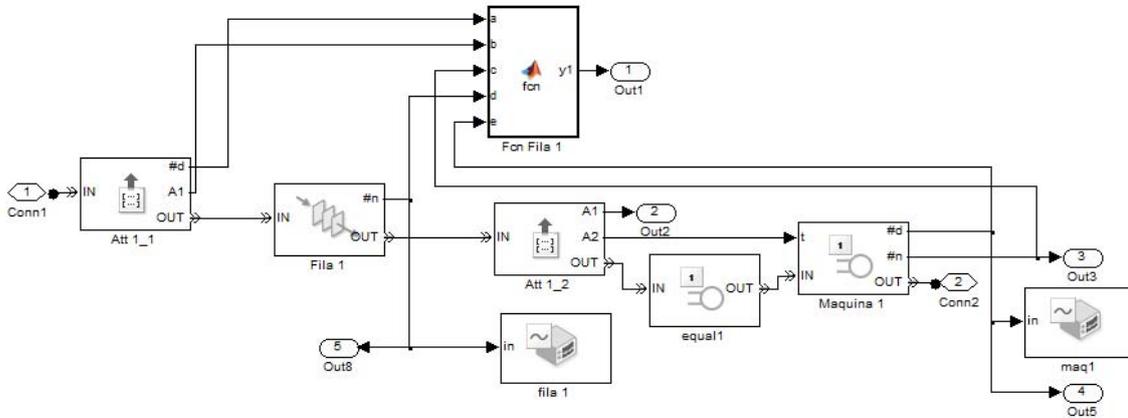


Figura 30 – Máquina do FMS

Todas as máquinas de um sistema deverão ser representadas por este arranjo, variando-se apenas o tamanho da fila. O primeiro bloco *get attribute* (*Att 1_1*) fornece ao bloco *Fcn_Fila1* dados relativos ao tipo de produto que entra na fila e quantidade de peças que passaram pelo bloco. Já o bloco *Att 1_2* fornece ao servidor o valor do tempo de processamento do produto que chega à máquina.

Para todos os modelos em *SimEvents*[®] existem certos blocos que separam a saída da última máquina de forma a determinar quantas unidades de cada produto foram produzidas. Com este valor, além de possibilitar a representação gráfica da produção por tipo de peça, pode-se calcular o valor final de recompensa para o arranjo. A recompensa total é calculada considerando o total de cada peça produzida, multiplicado pela recompensa total das máquinas e o valor agregado ao produto. Para comparação das recompensas, foram utilizados três métodos: (1) Variação do fator de desconto; (2) Heurísticas; e (3) Variação de recompensa.

No MDP é natural verificar o comportamento do sistema variando-se o fator de desconto γ , uma vez que este é o responsável pelo comportamento do acúmulo de recompensa. Neste trabalho, foram calculadas as políticas para doze valores de γ , partindo de próximo de zero (comportamento guloso) até próximo de um. Os valores de γ considerados são:

Tabela 21 – Valores de γ

γ	0,01	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	0,95	0,99
----------	------	-----	-----	-----	-----	-----	-----	-----	-----	-----	------	------

Tendo sido calculadas as políticas do MDP considerando estes doze valores, pode-se então realizar doze simulações. A cada simulação foram armazenados todos os dados

gerados, como utilização das filas, momento dos disparos de ações, produção total e tipos produzidos, por exemplo. Com base na produção total, foi calculada a recompensa total para o FMS utilizando este fator de desconto. Após as doze simulações é possível então verificar como se comporta a recompensa final do FMS com relação à variação de γ . Assim é possível decidir entre as políticas qual é aquela que apresenta maior recompensa, e, num sistema real, tomar as ações segundo esta política.

Outro método para geração de resultados nos FMS foi a utilização de heurísticas. Neste caso, foram determinadas manualmente algumas políticas que poderiam gerar bons resultados na simulação do FMS. Enquanto existe um sistema autônomo que resolve o MDP e retorna a política que na teoria seria ótima, o uso de heurísticas é basicamente a geração manual de uma política a partir do conhecimento empírico do sistema. Foram a princípio geradas diferentes heurísticas para os problemas dependendo da sua característica de modelagem. Uma vez determinados manualmente em quais estados devem ser executadas as ações e quais ações devem ser executadas nestes estados, foram gerados vetores de ações similares às políticas definidas pelo MDP. Assim o procedimento de simulação é o mesmo, havendo a mesma segregação de política entre as ações para o *SimEvents*[®]. As heurísticas de produção são apresentadas na descrição de cada sistema simulado.

4.2 Considerações sobre recompensas

Finalizando os métodos de comparação, existe a variação de recompensa. No entanto esta não é a comparação de sistemas com tempos de produção diferentes e, portanto, recompensas diferentes. Isso não é possível, pois se tratariam de FMS distintos. A princípio são consideradas recompensas apenas para as transições que realizam retirada de peça das máquinas. O restante das transições e conseqüentemente ações do MDP têm recompensa nula. As políticas obtidas com este método mostraram que as ações de inserção de peças no sistema foram preteridas na maioria dos estados. A análise de algumas políticas concluiu que em vários dos estados ignorados para ação das transições fontes, o disparo de transições fonte era possível. Isto se deve ao fato que os FMS são cíclicos e o MDP tem horizonte infinito. Assim, as ações tendem a ter o mesmo valor da função $Q(s, a)$ no infinito, ou seja, com $\gamma = 1$, e por isso não houve mudança entre as ações, prevalecendo as computadas logo no primeiro fator de desconto considerado ($\gamma = 0,01$).

De forma a evitar que os valores de $Q(s, a)$ não sejam iguais e que haja mudança nas políticas, de forma a modificar a produção conforme se altera o valor do fator de desconto, foi proposto um segundo método de modelagem das recompensas, adicionando um valor de recompensa ligeiramente maior que a maior recompensa de todos os eventos não controláveis em todos os estados do sistema. Desta forma, nos estados onde existe

transição fonte da RdP habilitada, existe a execução desta ação devido sua maior recompensa em relação à todas as outras. Para as simulações, foram mantidos os dois modelos, de forma a analisar a diferença de comportamento entre eles.

4.3 Simulações do comportamento em tempo real dos FMS

As primeiras simulações foram executadas em *flowshops*, devido a sua maior simplicidade na modelagem por RdP. De acordo com a expressão que avalia a quantidade de estados de um sistema, os *flowshops* apresentam maior número de estados, uma vez que todos os tipos de produto passam por todas as máquinas. Foi proposto um mesmo modelo de FMS com três máquinas e três fontes, já que a dinâmica do sistema é similar caso sejam adicionadas mais destes recursos no sistema. A análise é realizada em três modelos com tempos de produção e valores agregados diferentes, sendo modificada a capacidade das filas. A sensibilidade do sistema da forma em que foi modelado é notada nestas simulações. Devido a este fato, não é possível ampliar o sistema para filas com maiores dimensões, uma vez que o recurso computacional necessário para calcular a árvore de alcançabilidade e demais parâmetros do MDP aumentaria exponencialmente.

Após isto são realizadas duas simulações de *jobshop*. A primeira é baseada no modelo proposto por Matcovschi, Mahulea e Pastravanu (2003) como aplicação de um *toolbox* para RdP. Este FMS é composto também por três máquinas e três fontes, mas cada produto tem uma rota definida. Além deste, foi proposto um segundo modelo de *jobshop* contando com três fontes e quatro máquinas. Neste modelo, optou-se por um roteamento mais complexo, havendo mais concorrência de máquinas de forma a gerar maior utilização das filas.

A RdP modela o FMS considerando várias restrições de capacidades de lugares e de habilitação de eventos e a árvore de alcançabilidade é gerada conforme estas regras. Porém, na simulação em *SimEvents*[®], observou-se uma deficiência do sistema com relação à ocupação das filas. Estas foram caracterizadas no ambiente do *SimEvents*[®] com uma capacidade maior àquela modelada na RdP, de forma a verificar a robustez do sistema no disparo peças. Primeiro, observou-se que a fila presente na primeira máquina utilizável do FMS obedece fielmente à sua modelagem, uma vez que a execução de uma ação depende do seu estado. No entanto, filas subsequentes apresentaram um transbordamento, isto é, dependendo dos tempos de produção definidos para o FMS, se uma máquina termina o processamento, a próxima máquina está ocupada e existe lugar na fila, obviamente o sistema aloca esta peça pronta na fila de forma a continuar a produzir na máquina ociosa. Considerando que a fila da máquina ocupada esteja cheia de acordo com a capacidade definida na RdP, porém definida no *SimEvents*[®] com posições vazias, o MDP não tem como impedir esta ação. A tomada de decisão, porém, fica limitada até que o sistema

produza o que está parado nas filas e retorne a algum estado definido na RdP. Desta forma, todas as filas do sistema cuja entrada é a saída de uma máquina anterior tiveram sua capacidade limitada no *SimEvents*[®], não apenas pela RdP. Nos exemplos de *jobshop* este fenômeno é ainda mais importante, uma vez que existem filas intermediárias cujas entradas também são transições fontes. Assim, mesmo que a distribuição de tempos de produção das máquinas evite a formação de filas, pode haver o enchimento destas devido apenas à execução das ações.

4.3.1 Simulação modelo *flowshop* 1

A modelagem dos três modelos de *flowshop* no ambiente *SimEvents*[®] pode ser encontrado na figura 29. Nesta figura, pode-se concluir sobre a natureza do FMS devido a ausência de elementos de roteamento, a saída de uma máquina esta conectada diretamente na entrada da próxima. O diagrama de blocos que representa o primeiro modelo simulado de *flowshop* pode ser visto na figura 31.

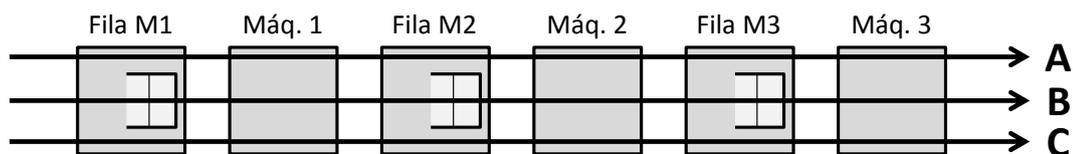


Figura 31 – Diagrama de blocos (*flowshop* 1)

O tempo de produção de cada produto em cada máquina e valor agregado (α) são definidos pela matriz *Temp* e vetor α no algoritmo de geração da RdP. Para este exemplo, esses dados são mostrados na tabela 22.

Tabela 22 – Tempos de produção e valor agregado (*flowshop* 1)

Produto	Máquina 1	Máquina 2	Máquina 3	α
A	20	30	5	1,0
B	10	25	35	1,1
C	5	40	15	0,9

A RdP que modela o FMS e seus demais parâmetros são encontrados na figura 32 e tabela 23.

Cada transição da RdP que modela o *flowshop* 1 corresponde a um evento. É importante que sejam bem definidas as ações do MDP e sua relação com as transições, já que estas são de importância fundamental para geração das matrizes de transição e de recompensas do MDP. No algoritmo de geração da RdP, esta definição é dada pela matriz *TA* e a tabela 24 mostra essa relação.

É importante também definir as funções de utilidade para o cálculo das recompensas geradas pelo processamento das máquinas. Da forma que foram apresentadas na

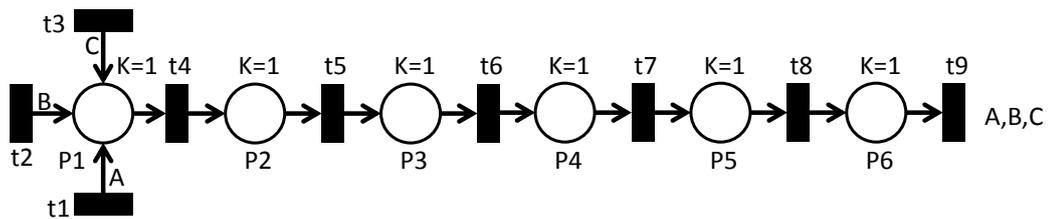


Figura 32 – Flowshop 1 representado por RdP

Tabela 23 – Definição dos elementos da RdP (*flowshops*)

Elemento	Tipo	Descrição
p1	Lugar	Fila da máquina 1
p2	Lugar	Máquina 1
p3	Lugar	Fila da Máquina 2
p4	Lugar	Máquina 2
p5	Lugar	Fila da máquina 3
p6	Lugar	Máquina 3
t1	Transição fonte	Inserir produto A
t2	Transição fonte	Inserir produto B
t3	Transição fonte	Inserir produto C
t4	Transição	Movimentação fila 1 / máquina 1
t5	Transição	Movimentação máquina 1 / fila 2
t6	Transição	Movimentação fila 2 / máquina 2
t7	Transição	Movimentação máquina 2 / fila 3
t8	Transição	Movimentação fila 3 / máquina 3
t9	Transição dreno	Retira produto pronto

Tabela 24 – Relação entre transições da RdP e ações do MDP (*flowshops*)

Transição RdP	Ação MDP	Tipo de evento
t1	Ação 1	Controlável
t2	Ação 2	Controlável
t3	Ação 3	Controlável
t4	Ação 4	Não controlável
t5	Ação 5	Não controlável
t6	Ação 4	Não controlável
t7	Ação 6	Não controlável
t8	Ação 4	Não controlável
t9	Ação 7	Não controlável

figura 17, convencionou-se o uso de (a) para máquina 1, (b) para máquina 2, (c) para máquina 3 e (d) para máquina 4, nos exemplos aplicáveis.

Determinada a modelagem do FMS *flowshop* 1 em RdP e demais parâmetros auxiliares, o algoritmo principal gera os estados do sistema e as matrizes de entrada para cálculo do MDP. De posse destes dados é possível que a simulação prossiga no ambiente do *SimEvents*[®]. No entanto, para finalizar os métodos de comparação, é necessário definir quais heurísticas foram aplicadas no problema. Para os três *flowshops* apresentados nesta seção foram criadas duas heurísticas que fornecem estratégias de produção determinadas manualmente.

As tabelas 25 e 26 mostram os estados definidos pelas heurísticas e quais ações são tomadas. Os produtos A, B e C são representados por 1, 2 e 3, respectivamente, tal como nos algoritmos e a representação dos estados está feita na forma de vetor:

$$[fila1 \text{ máquina1} \quad fila2 \text{ máquina2} \quad fila3 \text{ máquina3}]$$

Tabela 25 – Heurística 1 para os *flowshops*

Estado	Ação
000000	Ação 1 (Insere A)
010000	Ação 1 (Insere A)
010100	Ação 2 (Insere B)
020101	Ação 2 (Insere B)
020201	Ação 3 (Insere C)
030202	Ação 3 (Insere C)
030302	Ação 1 (Insere A)
010303	Ação 1 (Insere A)
010103	Ação 2 (Insere B)

Tabela 26 – Heurística 2 para os *flowshops*

Estado	Ação
000000	Ação 3 (Insere C)
000300	Ação 1 (Insere A)
000103	Ação 2 (Insere B)
000201	Ação 3 (Insere C)
000302	Ação 1 (Insere A)
000100	Ação 2 (Insere B)
000200	Ação 3 (Insere C)

Foram feitas quatro simulações:

- Simulação do *flowshop* sem recompensa nas transições fontes e doze valores de γ ;
- Simulação do *flowshop* com recompensa nas transições fontes e doze valores de γ ;

- Simulação do *flowshop* com heurística 1;
- Simulação do *flowshop* com heurística 2.

Os resultados das simulações são mostrados na figura 33 e tabela 27.

Tabela 27 – Recompensas totais (*flowshop* 1)

Modelo sem recompensa das transições fontes	
$\gamma=0,01$	1072,416952
$\gamma=0,1$	1072,416952
$\gamma=0,2$	1072,416952
$\gamma=0,3$	1072,416952
$\gamma=0,4$	2068,232693
$\gamma=0,5$	2068,232693
$\gamma=0,6$	2068,232693
$\gamma=0,7$	2068,232693
$\gamma=0,8$	2056,559693
$\gamma=0,9$	2056,559693
$\gamma=0,95$	2056,559693
$\gamma=0,99$	2056,559693
Modelo com recompensa das transições fontes	
$\gamma=0,01$	3618,454015
$\gamma=0,1$	3319,831172
$\gamma=0,2$	3582,09891
$\gamma=0,3$	2068,232693
$\gamma=0,4$	2068,232693
$\gamma=0,5$	2068,232693
$\gamma=0,6$	2068,232693
$\gamma=0,7$	2068,232693
$\gamma=0,8$	2068,232693
$\gamma=0,9$	2105,884799
$\gamma=0,95$	3656,10612
$\gamma=0,99$	3656,10612
Heurística 1 aplicada	
Recompensa	2403,210642
Heurística 2 aplicada	
Recompensa	2551,57925

Face ao exposto pelas considerações sobre a ocupação de filas, são escolhidos dois pontos que apresentaram valores discrepantes de recompensa e feita uma análise considerando o total de peças produzidas, o tipo de peças produzidas e o comportamento de cada uma das filas. Os pontos escolhidos são $\gamma = 0,2$ e $\gamma = 0,3$ para simulação com recompensa na fonte. Estes pontos foram escolhidos devido à diminuição do valor da recompensa total para estes dois casos.

A figura 34 mostra o comparativo das três filas, o total produzido pela última máquina e o tipo de peças produzidas. Com estes gráficos pode-se perceber a mudança

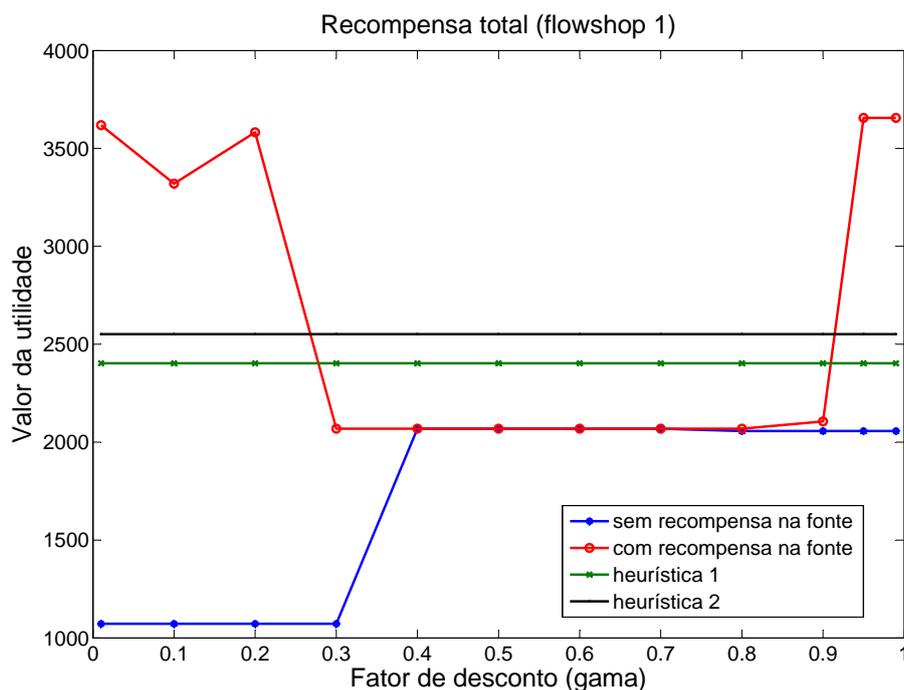


Figura 33 – Variação das recompensas totais (*flowshop 1*)

de comportamento das filas a partir do momento em que o MDP troca a ação de disparo, passando de peça tipo B (2) para peça tipo A (1) no caso de $\gamma = 0,2$. Esta troca acontece logo após os 200 ciclos de processamento e ao se analisar todas as filas e produção até este momento, os gráficos são rigorosamente iguais. Além disso, há um impacto na produção final, reflexo dos tempos de processamento do produto B em comparação ao produto A.

Observando os tempos de produção das peças A e B e o fator de valor agregado α , percebe-se que a produção do produto A tem uma recompensa maior com relação aos valores convertidos pelas funções de utilidade (118,25 contra 69,64), e mesmo tendo um fator de valor agregado (1,1 contra 1 - tabela 22), a produção maciça do produto B não gera uma recompensa total maior ao final do ciclo. Um estudo sobre a variação do valor de α e a reavaliação das recompensas totais pode ser feito para análise do comportamento do MDP.

Outra análise possível de ser feita é sobre o comportamento dos disparos de peças e total de produção variando-se o tamanho da fila no ambiente *SimEvents*[®]. Para este mesmo *flowshop* é feita primeiramente a simulação do modelo fiel à RdP da figura 27, com filas de tamanho unitário. Após isto, repetiu-se a simulação considerando filas com tamanho igual a cinco. O ponto escolhido para análise é $\gamma = 0,99$ para simulação com recompensa na fonte. Os resultados obtidos são mostrados na figura 35.

Este comparativo mostra que para os dois exemplos, a fila 1 mantém apenas uma

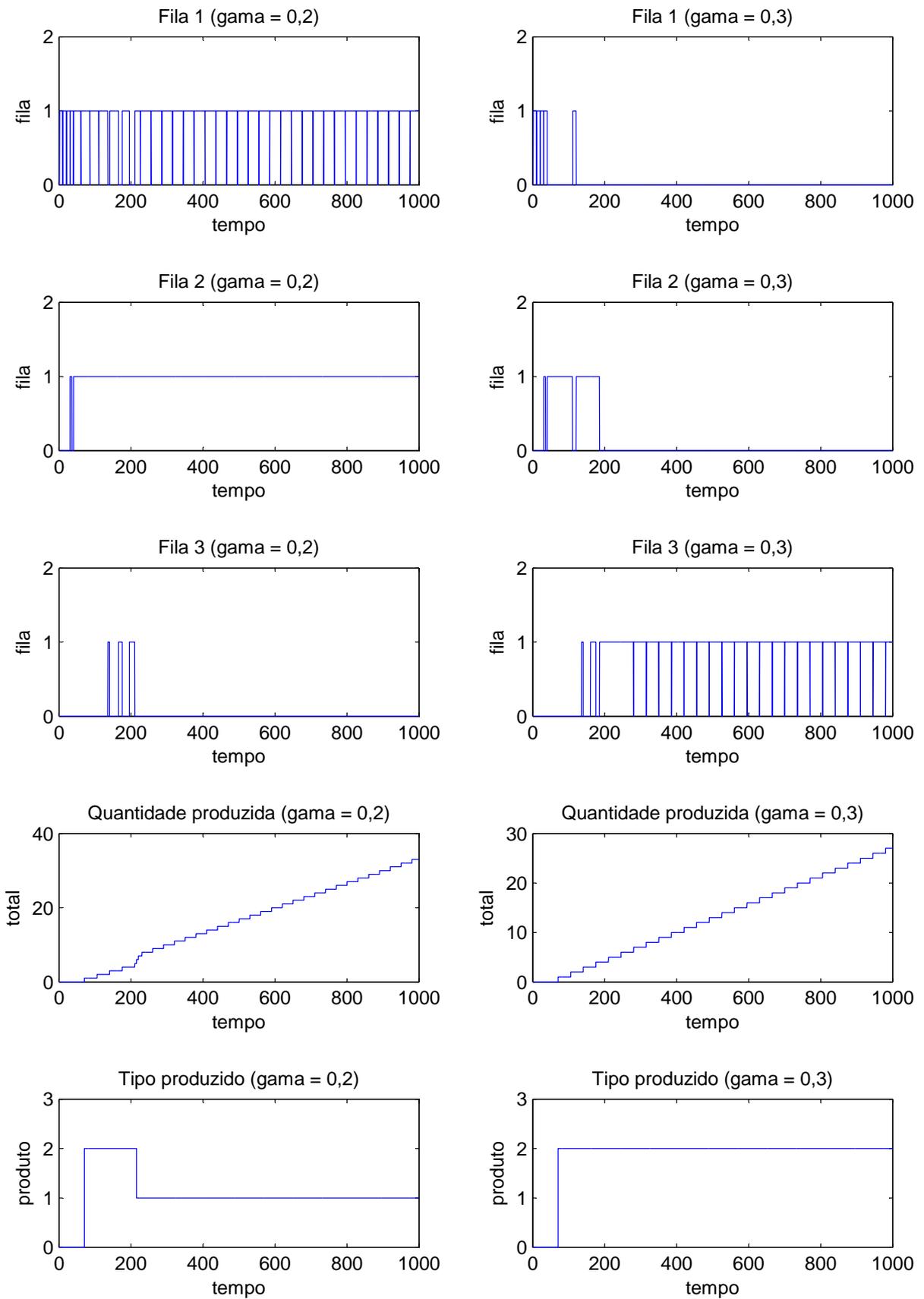


Figura 34 – Comportamento de filas (variação de γ)

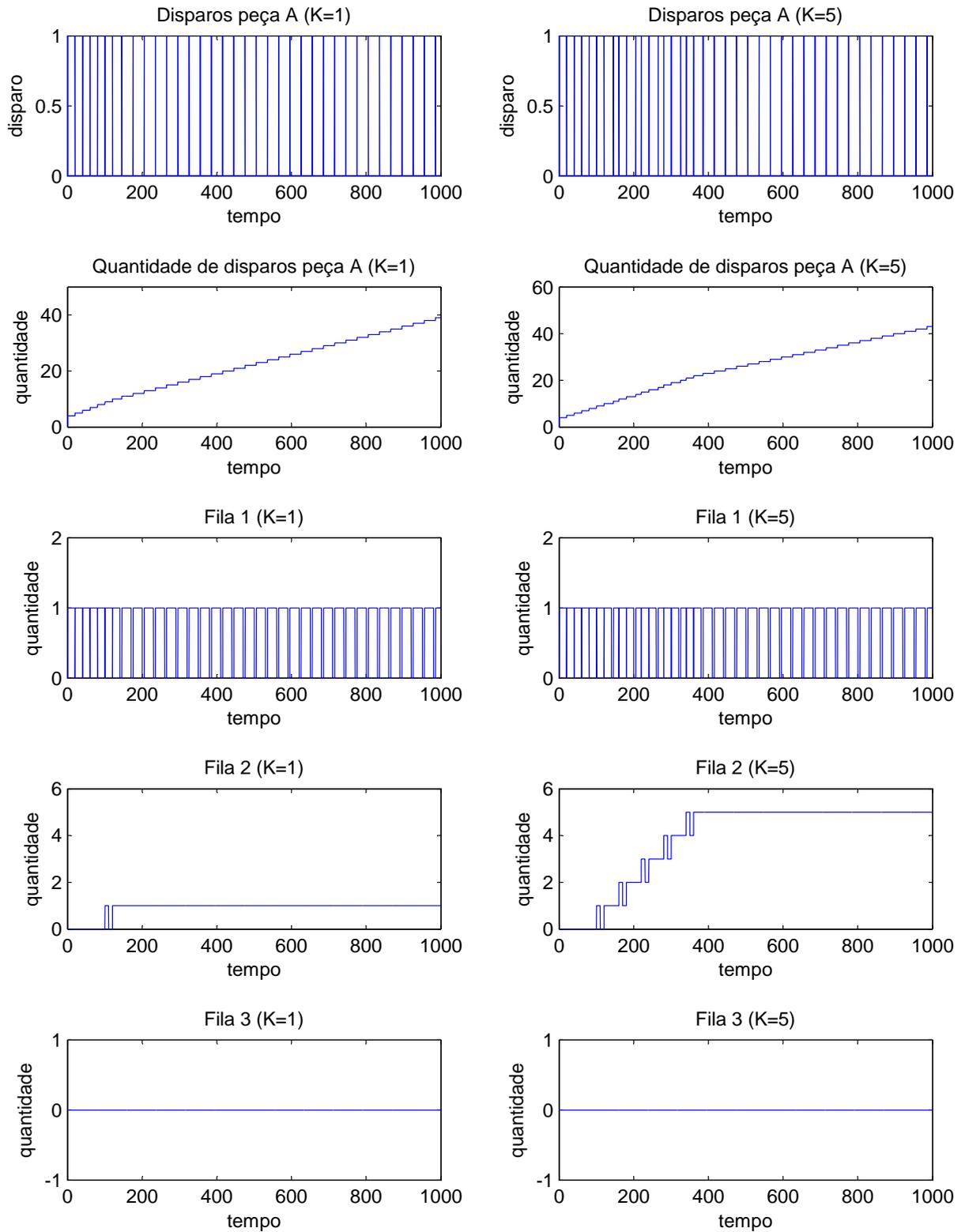


Figura 35 – Comportamento de filas (variação de capacidade)

Tabela 28 – Tempos de produção e valor agregado (*flowshop 2*)

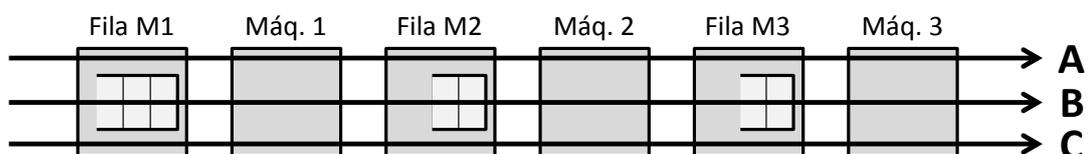
Produto	Máquina 1	Máquina 2	Máquina 3	α
A	20	30	5	0,8
B	5	25	35	1
C	40	10	15	1,3

unidade conforme modelado pela RdP. A fila 2, no entanto, sofre influência do tamanho, tendo ocupação total após certo tempo de processamento. A fila 3 não é ocupada em nenhum momento. Com relação ao comportamento do MDP, pode-se perceber uma diferença entre os dois resultados, embora seja esta sutil. A quantidade de disparos de peça A mostra que enquanto houve 39 disparos para o modelo com capacidade de filas unitária, este valor sobe para 43 para o modelo com capacidade cinco. Como houve transbordamento em apenas uma das filas, a diferença entre os resultados é exatamente a quantidade aumentada da capacidade desta fila. Isto acontece devido à interpretação dos estados pelo sistema. Mesmo havendo mais de uma peça na fila 2, o mapeamento de estados desta fila segundo a árvore de alcançabilidade da RdP considera apenas uma peça. Assim, uma vez que há peça na fila, o sistema executa a ação definida pelo MDP para este estado indefinidamente. Caso a fila 3 tivesse sido utilizada, o resultado possivelmente manteria o mesmo padrão, considerando também apenas uma peça para a fila 3.

Os gráficos de ocupação da fila 1 mostram que as ações tomadas pelo MDP a mantiveram com no máximo uma unidade, de forma que o modelo em *SimEvents*[®] respeita a capacidade definida na modelagem. No entanto, a necessidade de limitação do tamanho das filas intermediárias no próprio modelo em *SimEvents*[®] mostra uma fraqueza do sistema modelado em RdP com tomada de decisão com MDP. Uma possível alternativa a este estudo seria o uso da teoria de controle supervisorio (MOLINA, 2007; SILVA, 2007), de forma a eliminar as transições que colocam peças semiacabadas no *buffer* caso a capacidade da fila do *SimEvents*[®] atinja o valor da capacidade modelada em RdP.

4.3.2 Simulação modelo *flowshop 2*

O *flowshop 2* é similar ao *flowshop 1*, variando-se apenas a capacidade da fila 2 em uma unidade. O diagrama de blocos, RdP equivalente e demais parâmetros que representam este modelo são apresentados nas figuras 36 e 37 e tabela 28.

Figura 36 – Diagrama de blocos (*flowshop 2*)

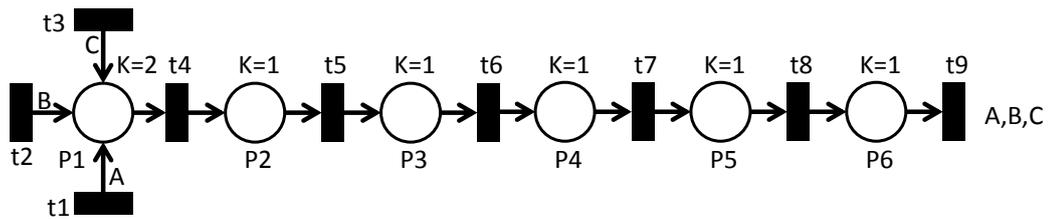


Figura 37 – *Flowshop 2* representado por RdP

Devido a similaridade entre os *flowshops*, a definição dos elementos da RdP (tabela 23) e relação entre transições da RdP e ações do MDP (tabela 24), bem como as heurísticas (tabelas 25 e 26) são omitidas para os exemplos *flowshop 2* e *flowshop 3*. Realizadas as quatro simulações, têm-se os seguintes resultados a seguir (tabela 29 e figura 38).

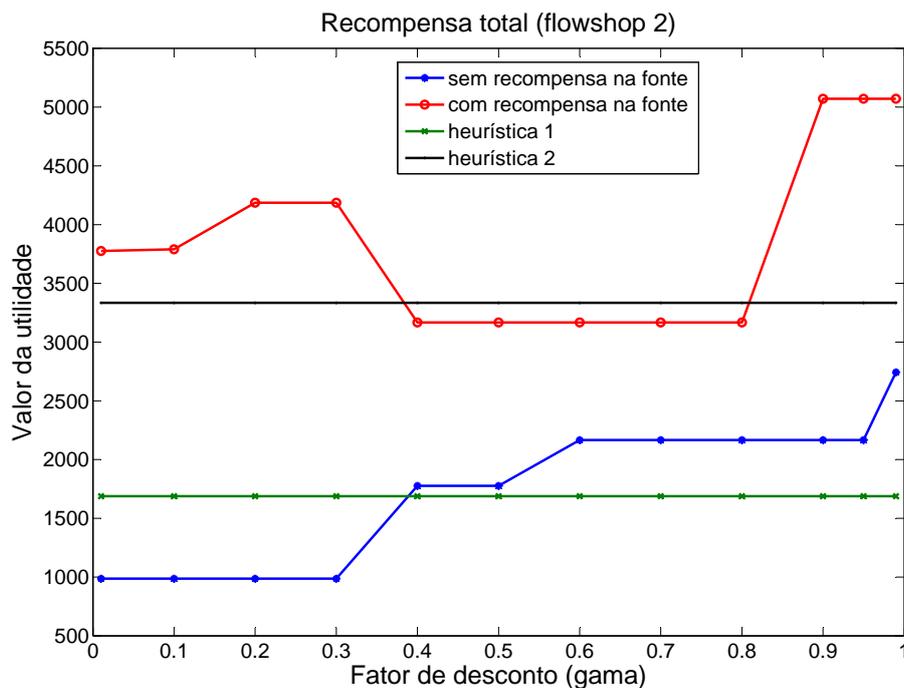


Figura 38 – Variação das recompensas totais (*flowshop 2*)

Apesar de a modelagem ser praticamente a mesma, sendo alterada apenas a capacidade da fila 1, os sistemas são diferentes quanto aos tempos de produção de cada produto nas máquinas e seus valores agregados. As diferenças de cada um dos sistemas podem ser vistas nas tabelas 22 e 28.

Os resultados das quatro simulações podem ser vistos na figura 38. As duas heurísticas têm recompensa total constante, pois este valor independe do fator de desconto (γ). No gráfico estas recompensas são estendidas para todos os valores γ para comparação

Tabela 29 – Recompensas totais (*flowshop 2*)

Modelo sem recompensa das transições fontes	
$\gamma=0,01$	987,1017876
$\gamma=0,1$	987,1017876
$\gamma=0,2$	987,1017876
$\gamma=0,3$	987,1017876
$\gamma=0,4$	1776,783218
$\gamma=0,5$	1776,783218
$\gamma=0,6$	2166,073333
$\gamma=0,7$	2166,073333
$\gamma=0,8$	2166,073333
$\gamma=0,9$	2166,073333
$\gamma=0,95$	2166,073333
$\gamma=0,99$	2743,692888
Modelo com recompensa das transições fontes	
$\gamma=0,01$	3775,014783
$\gamma=0,1$	3775,014783
$\gamma=0,2$	4186,470535
$\gamma=0,3$	4186,470535
$\gamma=0,4$	3166,674276
$\gamma=0,5$	3166,674276
$\gamma=0,6$	3166,674276
$\gamma=0,7$	3166,674276
$\gamma=0,8$	3166,674276
$\gamma=0,9$	5071,813667
$\gamma=0,95$	5071,813667
$\gamma=0,99$	5071,813667
Heurística 1 aplicada	
Recompensa	1689,092909
Heurística 2 aplicada	
Recompensa	3334,198779

com os valores de recompensas dos MDPs. Também com relação às heurísticas, mesmo sendo iguais para os exemplos de *flowshop*, há uma diferença considerável dos valores de recompensa total entre o *flowshop 1* e *flowshop 2*. Isto se deve à diferença dos tempos de produção entre os dois exemplos.

As simulações dos MDPs mostram que a recompensa total do *flowshop* sem recompensa na fonte tornou-se maior conforme se aumenta o valor de γ . Isto mostra que a recompensa acumulada apresentou um maior valor em comparação com a recompensa imediata. Além disso, o *flowshop* com recompensa na fonte apresentou o maior valor de recompensa total em boa parte das simulações, além de ser sempre maior que o *flowshop* sem recompensa na fonte. Isto se deve à maior execução das ações de disparo devido à priorização forçada pela recompensa na fonte.

4.3.3 Simulação modelo *flowshop* 3

O *flowshop* 3 apresenta o mesmo arranjo dos *flowshops* 1 e *flowshop* 2, porém conta com diferentes capacidades de fila, sendo representado pelo diagrama e RdP das figuras 39 e 40 e tabela 30 a seguir.

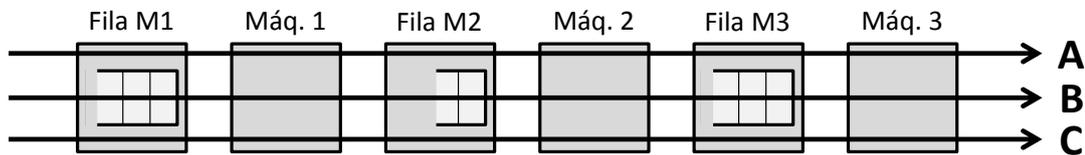


Figura 39 – Diagrama de blocos (*flowshop* 3)

Tabela 30 – Tempos de produção e valor agregado (*flowshop* 3)

Produto	Máquina 1	Máquina 2	Máquina 3	α
A	5	15	10	0,8
B	20	25	5	1
C	40	5	20	1,3

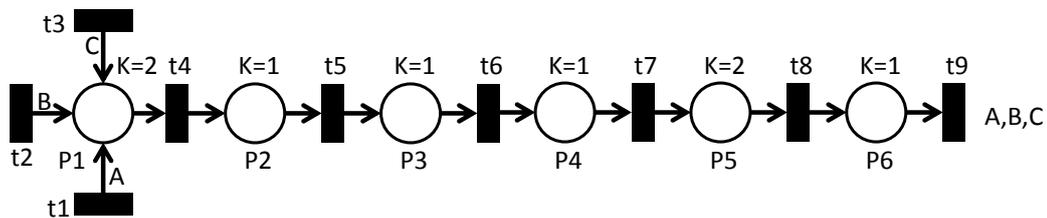


Figura 40 – *Flowshop* 3 representado por RdP

Os resultados obtidos para as simulações deste modelo, seguindo o mesmo procedimento anteriormente utilizado, são mostrados na tabela 31 e figura 41.

Após os dois primeiros exemplos de *flowshop* apresentarem bons resultados dos MDPs em relação às heurísticas, a figura 41 mostra um resultado oposto. As heurísticas apresentaram um valor de recompensa total maior que os dois MDPs para todos os valores de γ . Apesar disso, o *flowshop* com recompensa na fonte apresentou maior recompensa total que o *flowshop* sem recompensa na fonte, mesma característica observada nos outros dois modelos anteriores. A análise das políticas mostra que, para valores mais baixos de γ , o *flowshop* sem recompensa na fonte escolhe a ação produzir peça B apenas no estado inicial. Já o modelo com recompensa na fonte, escolhe ações de produção para 13312 estados (30% do total de estados do sistema), divididas entre produção da peça A (em 99,7% dos casos) e peça B (para os 0,3% restantes). A priorização das ações de produção para aproximadamente um terço dos estados do sistema faz com que o modelo com recompensa na fonte obtenha uma utilidade total maior que o modelo sem recompensa

Tabela 31 – Recompensas totais (*flowshop* 3)

Modelo sem recompensa das transições fontes	
$\gamma=0,01$	2231,018914
$\gamma=0,1$	2231,018914
$\gamma=0,2$	2149,427768
$\gamma=0,3$	2149,427768
$\gamma=0,4$	2149,427768
$\gamma=0,5$	2149,427768
$\gamma=0,6$	2149,427768
$\gamma=0,7$	2149,427768
$\gamma=0,8$	2149,427768
$\gamma=0,9$	3009,198875
$\gamma=0,95$	3009,198875
$\gamma=0,99$	4462,037828
Modelo com recompensa das transições fontes	
$\gamma=0,01$	4519,140311
$\gamma=0,1$	4540,998039
$\gamma=0,2$	4516,429176
$\gamma=0,3$	4457,648592
$\gamma=0,4$	3166,674276
$\gamma=0,5$	4362,084273
$\gamma=0,6$	4244,559636
$\gamma=0,7$	3439,084429
$\gamma=0,8$	3439,084429
$\gamma=0,9$	3439,084429
$\gamma=0,95$	3439,084429
$\gamma=0,99$	4462,037828
Heurística 1 aplicada	
Recompensa	4762,70396
Heurística 2 aplicada	
Recompensa	4858,268279

na maioria das simulações. Apenas na simulação com o fator de desconto próximo a 1 as utilidades totais dos dois modelos são iguais.

Este resultado atenta para a importância de estabelecer funções de recompensas razoáveis ao problema no momento da modelagem. Caso houvesse um melhor estudo sobre qual função de utilidade usar para cada máquina ou mesmo a análise do valor agregado ao tipo de produto, os resultados poderiam ter sido melhores em relação às heurísticas. Para os parâmetros considerados neste exemplo, o MDP obtém resultados ruins para as duas simulações. Vale lembrar que os parâmetros de modelagem dos FMS e as funções de utilidade foram adotados sem o conhecimento de FMS reais, o que seria mais indicado para verificação da atuação do MDP. Assim, mesmo que fossem testados exaustivamente diversos parâmetros para o sistema e diferentes funções de utilidade, não haveria garantia de melhores resultados.

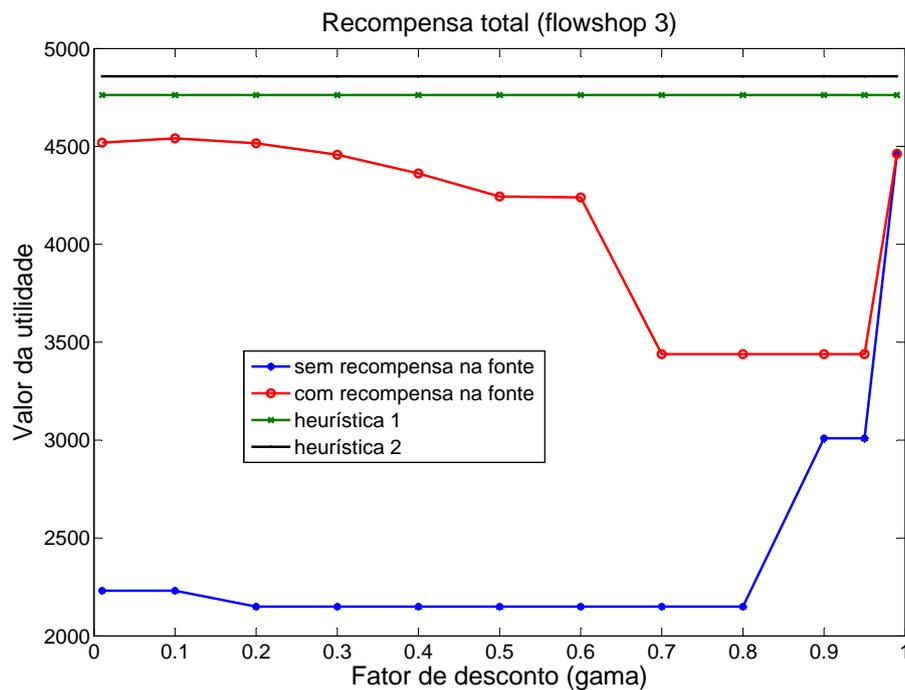


Figura 41 – Variação das recompensas totais (*flowshop 3*)

4.3.4 Simulação modelo *jobshop 1*

Tendo sido simulados os três modelos de *flowshop* propostos, são feitas as simulações dos *jobshops*. Apesar de apresentarem menor número de estados em relação aos *flowshops*, os *jobshops* são mais complexos por apresentarem diferentes rotas entre as máquinas para os diferentes produtos presentes no FMS. Os *flowshops* apresentam um comportamento cíclico mais evidente, com menos variação nos tipos de peças que são produzidas no sistema, fato que pode ser concluído caso seja analisada a produção final. Já os *jobshops*, devido a maior variedade de rotas, permite que haja um disparo simultâneo de peças, uma vez que podem existir fontes que alimentam máquinas diferentes. No entanto, isto não se aplica às simulações, já que o MDP é sequencial e não admite a execução de mais de uma ação em cada época de decisão. A figura 42 apresenta a modelagem do problema *jobshop* no ambiente do *SimEvents*[®].

O primeiro modelo de *jobshop* simulado é o mesmo proposto por Matcovschi, Mahulea e Pastravanu (2003), no qual existem três máquinas e três transições fontes. A capacidade de filas foi adotada aleatoriamente, como nos modelos de *flowshop*. Neste caso, existem blocos que fazem o roteamento das peças entre as máquinas, agrupando os tipos (bloco *path combiner*) ou separando-os (*output switch*). Desta forma, consegue-se a independência entre as entidades para que estas sejam processadas apenas pelas máquinas previstas na modelagem do FMS. Após isto, a figura 43 apresenta o diagrama de blocos que representa este FMS.

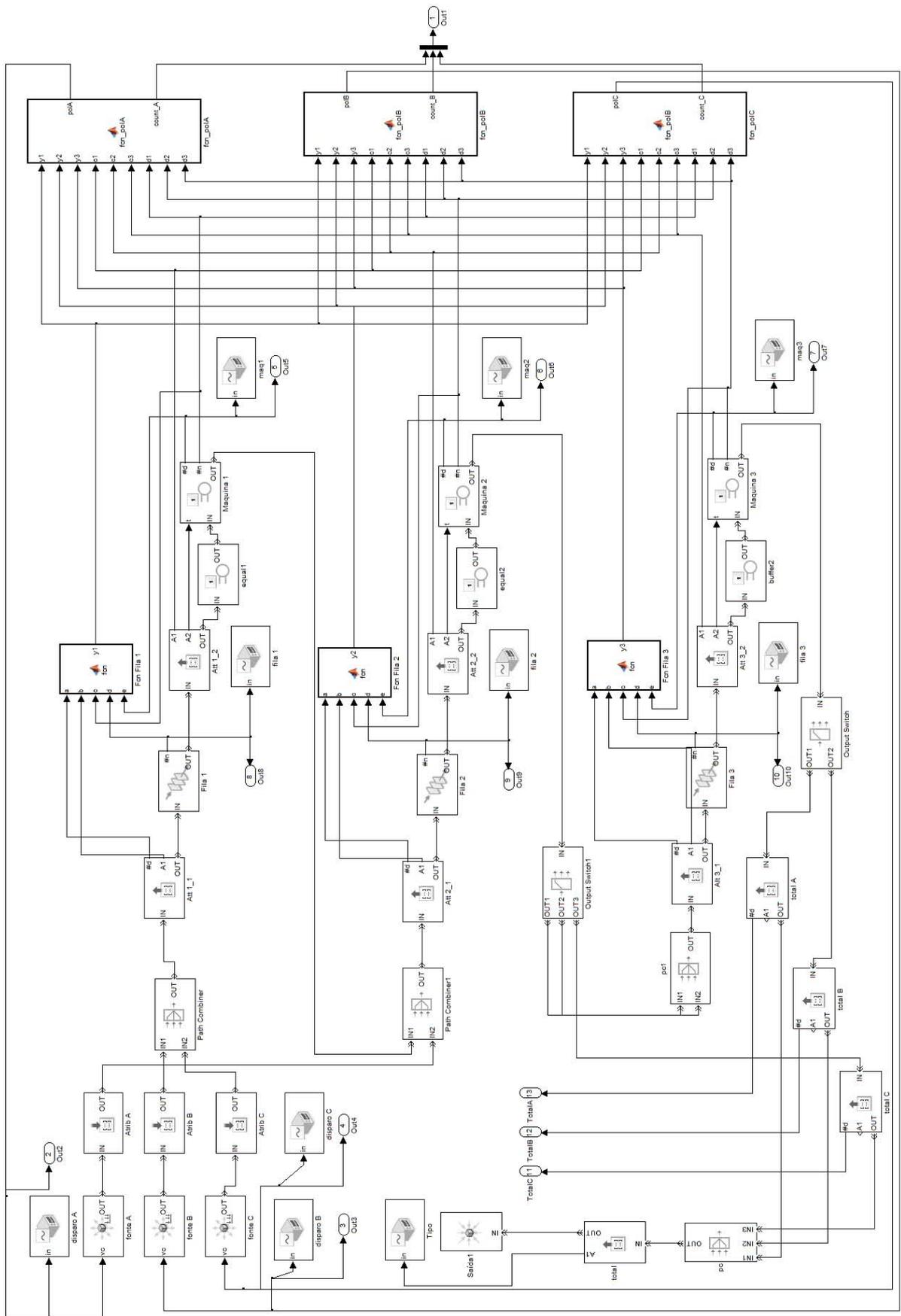


Figura 42 – Modelo do jobshop 1 em SimEvents®

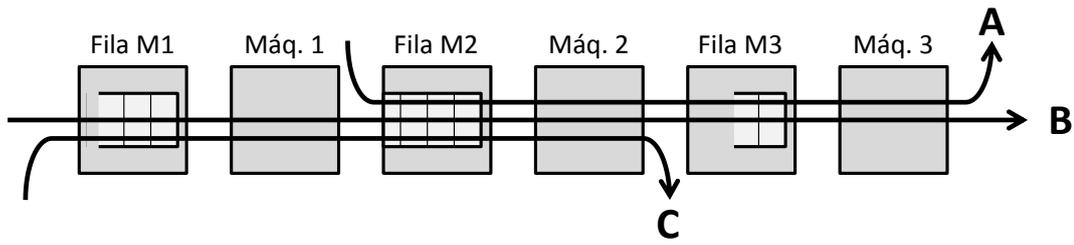


Figura 43 – Diagrama de blocos (*jobshop 1*)

Assim como feito nos *flowshops*, são definidos os dados que caracterizam o FMS. O tempo de produção de cada produto em cada máquina, valor agregado (α), RdP equivalente, definição dos elementos da RdP e relação entre transição do RdP e ações do MDP são encontrados abaixo (figura 44 e tabelas 32, 33 e 34).

Tabela 32 – Tempos de produção e valor agregado (*jobshop 1*)

Produto	Máquina 1	Máquina 2	Máquina 3	α
A	-	20	35	0,8
B	15	5	10	1
C	10	40	-	1,3

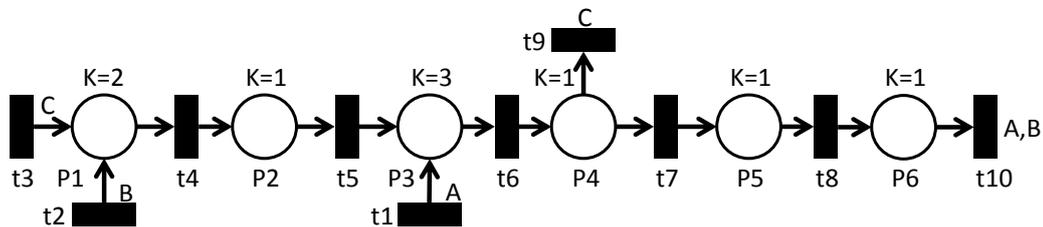


Figura 44 – *Jobshop 1* representado por RdP

Também da mesma forma que foi feito nas simulações dos *flowshops*, foram propostas duas heurísticas para a comparação do *jobshop*. As duas heurísticas foram geradas de acordo com observação, como um programador de produção decidindo qual ação tomar em cada estado do sistema. Da mesma forma que anteriormente, os estados foram representados nas tabelas como vetores. As tabelas 35 e 36 mostram as duas heurísticas.

Uma vez determinadas as heurísticas, foram realizadas as quatro simulações. Os resultados das recompensas totais obtidas são apresentados na tabela 37 e figura 45.

O resultado obtido na figura 45 mostra um comportamento similar ao ocorrido no *flowshop 3*. Mesmo considerando uma porcentagem muito maior de disparos, o modelo com recompensa na fonte priorizou a fabricação de produtos com um valor agregado menor, ocasionando assim uma recompensa total menor que a adquirida pelo modelo sem recompensa na fonte. Seja considerado o pior caso obtido para ($\gamma = 0,6$). A figura 46 mostra a produção total de cada modelo por tipo fabricado. Para o caso do modelo sem

Tabela 33 – Definição dos elementos da RdP (*jobshop 1*)

Elemento	Tipo	Descrição
p1	Lugar	Fila da máquina 1
p2	Lugar	Máquina 1
p3	Lugar	Fila da Máquina 2
p4	Lugar	Máquina 2
p5	Lugar	Fila da máquina 3
p6	Lugar	Máquina 3
t1	Transição fonte	Inserir produto A
t2	Transição fonte	Inserir produto B
t3	Transição fonte	Inserir produto C
t4	Transição	Movimentação fila 1 / máquina 1
t5	Transição	Movimentação máquina 1 / fila 2
t6	Transição	Movimentação fila 2 / máquina 2
t7	Transição	Movimentação máquina 2 / fila 3
t8	Transição	Movimentação fila 3 / máquina 3
t9	Transição dreno	Retira produto pronto C
t9	Transição dreno	Retira produtos prontos A e B

Tabela 34 – Relação entre transições da RdP e ações do MDP (*jobshop 1*)

Transição RdP	Ação MDP	Tipo de evento
t1	Ação 1	Controlável
t2	Ação 2	Controlável
t3	Ação 3	Controlável
t4	Ação 4	Não controlável
t5	Ação 5	Não controlável
t6	Ação 4	Não controlável
t7	Ação 6	Não controlável
t8	Ação 4	Não controlável
t9	Ação 6	Não controlável
t10	Ação 7	Não controlável

Tabela 35 – Heurística 1 para *jobshop 1*

Estado	Ação
000000	Ação 2 (Inserir B)
020000	Ação 1 (Inserir A)
020100	Ação 3 (Inserir C)
000302	Ação 2 (Inserir B)
000300	Ação 2 (Inserir B)
000002	Ação 1 (Inserir A)
000102	Ação 3 (Inserir C)
000100	Ação 2 (Inserir B)
000001	Ação 1 (Inserir A)
000101	Ação 3 (Inserir C)
000200	Ação 3 (Inserir C)

Tabela 36 – Heurística 2 para *jobshop* 1

Estado	Ação
000000	Ação 3 (Insere C)
030000	Ação 1 (Insere A)
030100	Ação 2 (Insere B)
000203	Ação 3 (Insere C)
000200	Ação 3 (Insere C)
000003	Ação 1 (Insere A)
000103	Ação 2 (Insere B)
000001	Ação 1 (Insere A)
000101	Ação 2 (Insere B)
020101	Ação 3 (Insere C)

Tabela 37 – Recompensas totais (*jobshop* 1)

Modelo sem recompensa das transições fontes	
$\gamma=0,01$	3411,947426
$\gamma=0,1$	3411,947426
$\gamma=0,2$	3411,947426
$\gamma=0,3$	3411,947426
$\gamma=0,4$	3411,947426
$\gamma=0,5$	3411,947426
$\gamma=0,6$	5686,579043
$\gamma=0,7$	5686,579043
$\gamma=0,8$	5686,579043
$\gamma=0,9$	2910,050671
$\gamma=0,95$	2910,050671
$\gamma=0,99$	3880,067561
Modelo com recompensa das transições fontes	
$\gamma=0,01$	2547,235751
$\gamma=0,1$	2547,235751
$\gamma=0,2$	2547,235751
$\gamma=0,3$	2513,469471
$\gamma=0,4$	2553,376014
$\gamma=0,5$	2400,386917
$\gamma=0,6$	2306,253499
$\gamma=0,7$	2520,117058
$\gamma=0,8$	2520,117058
$\gamma=0,9$	5097,196456
$\gamma=0,95$	3752,11548
$\gamma=0,99$	4052,957548
Heurística 1 aplicada	
Recompensa	5024,030957
Heurística 2 aplicada	
Recompensa	3517,791326

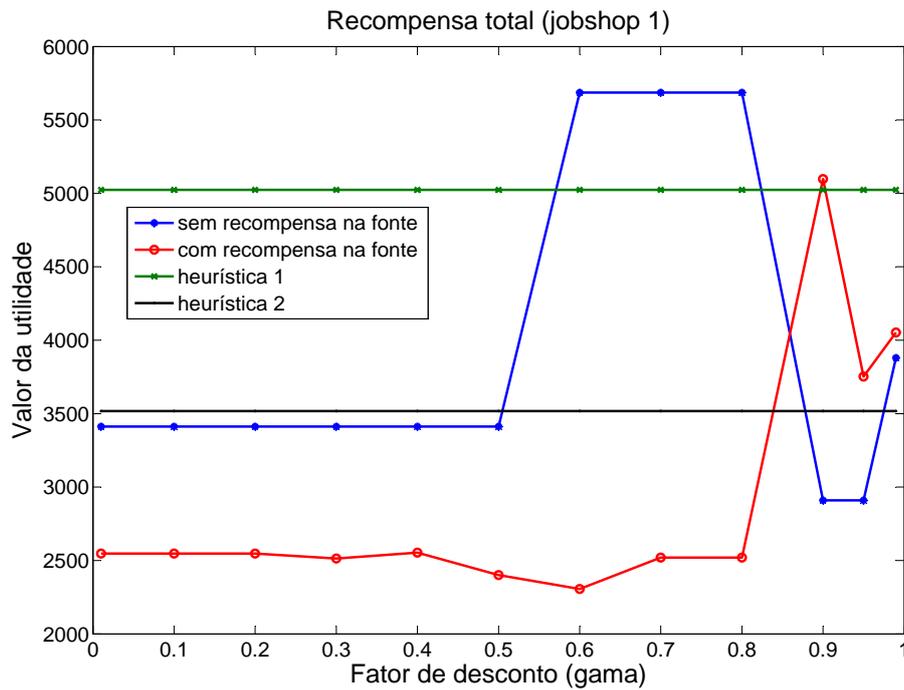


Figura 45 – Variação das recompensas totais (*jobshop 1*)

recompensa na fonte são produzidos exclusivamente produtos do tipo 3, que tem maior valor agregado. Já o modelo com recompensa na fonte, ao produzir os três diferentes tipos de peça, acabou recebendo uma recompensa total menor.

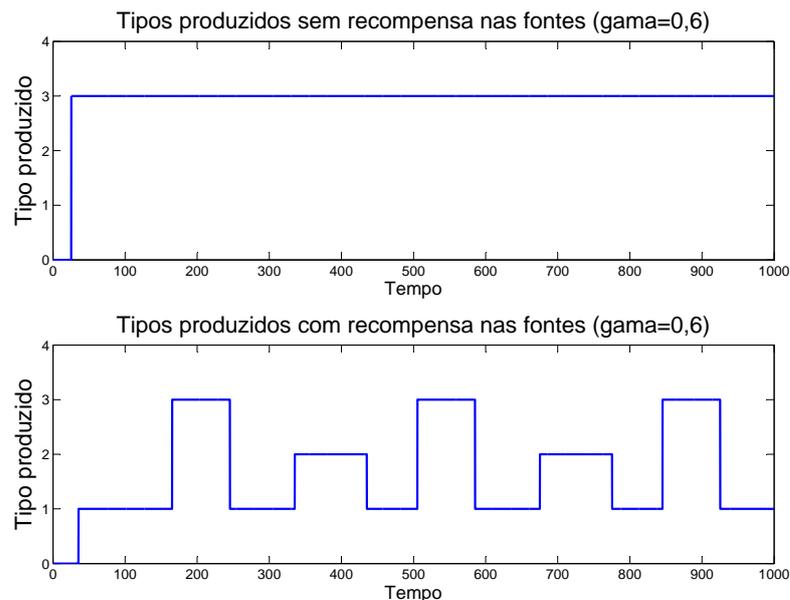


Figura 46 – Tipos produzidos para o *jobshop 1* ($\gamma = 0,6$)

Outro ponto analisado é para $\gamma = 0,9$, onde o modelo com recompensa na fonte tem a maior recompensa total. Comparando a produção entre os dois MDPs, neste caso o modelo sem recompensa muda a produção toda para produto tipo 2, com valor agregado intermediário, enquanto o modelo com recompensa passa a produzir em sua maioria o produto tipo 3, que tem maior valor agregado. A figura 47 mostra este caso.

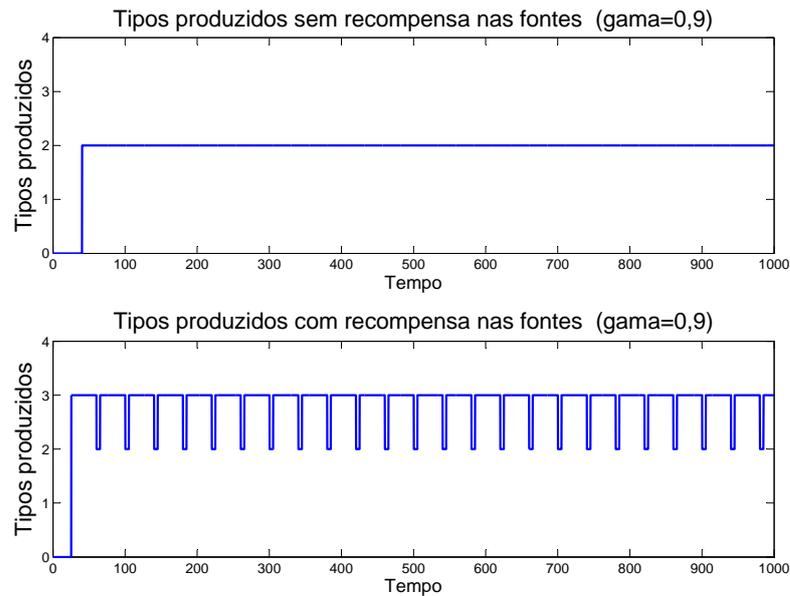


Figura 47 – Tipos produzidos para o *jobshop* 1 ($\gamma = 0,9$)

4.3.5 Simulação modelo *jobshop* 2

O segundo modelo de *jobshop* proposto é um FMS composto por três transições fontes e quatro máquinas. A ideia deste arranjo é verificar o comportamento do sistema para um *layout* de rotas mais elaborado contando com uma máquina a mais. O diagrama de blocos, modelo em *SimEvents*[®] e RdP do sistema são mostrados na figuras 48, 49 e 50.

Os tempos de produção das peças em cada máquina, a definição dos elementos da RdP, e a relação entre transições da RdP e ações do MDP são encontrados nas tabelas 38, 39 e 40 respectivamente.

Tabela 38 – Tempos de produção e valor agregado (*jobshop* 2)

Produto	Máquina 1	Máquina 2	Máquina 3	Máquina 4	α
A	20	5	-	10	0,8
B	5	10	20	-	1
C	-	-	15	5	1,3

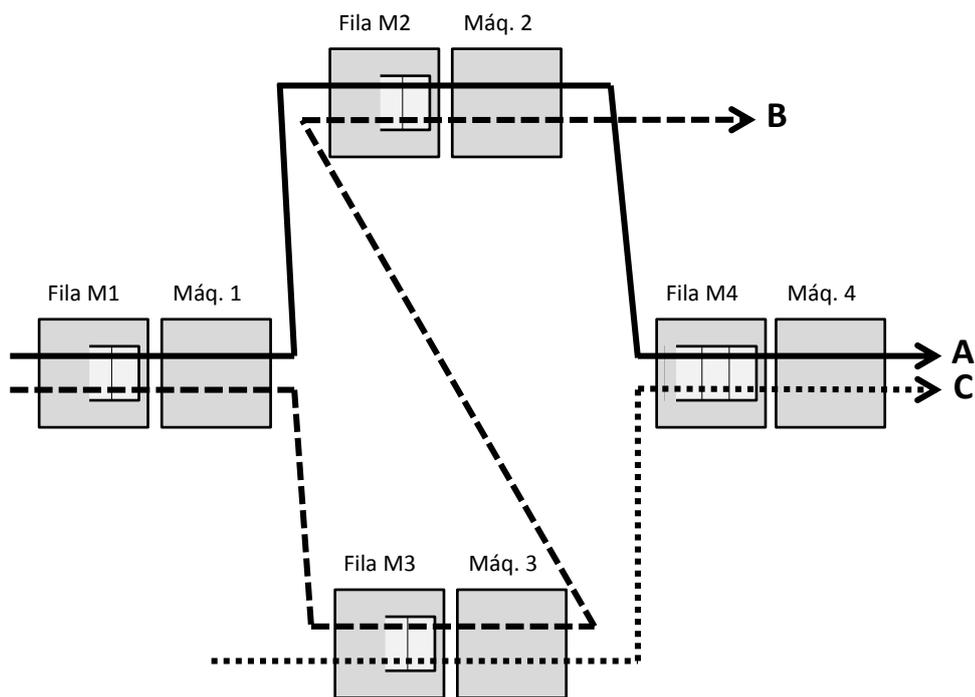


Figura 48 – Diagrama de blocos (*jobshop 2*)

Tabela 39 – Definição dos elementos da RdP (*jobshop 2*)

Elemento	Tipo	Descrição
p1	Lugar	Fila da máquina 1 ($K = 1$)
p2	Lugar	Máquina 1
p3	Lugar	Fila da Máquina 2 ($K = 1$)
p4	Lugar	Máquina 2
p5	Lugar	Fila da máquina 3 ($K = 1$)
p6	Lugar	Máquina 3
p7	Lugar	Fila da máquina 4 ($K = 2$)
p8	Lugar	Máquina 4
t1	Transição fonte	Inserir produto A
t2	Transição fonte	Inserir produto B
t3	Transição fonte	Inserir produto C
t4	Transição	Movimentação fila 1 / máquina 1
t5	Transição	Movimentação máquina 1 / fila 3
t6	Transição	Movimentação máquina 1 / fila 2
t7	Transição	Movimentação fila 3 / máquina 3
t8	Transição	Movimentação máquina 3 / fila 2
t9	Transição	Movimentação fila 2 / máquina 2
t10	Transição	Movimentação máquina 3 / fila 4
t11	Transição	Movimentação máquina 2 / fila 4
t12	Transição	Movimentação fila 4 / máquina 4
t13	Transição dreno	Retira produtos prontos A e C
t14	Transição dreno	Retira produto pronto B

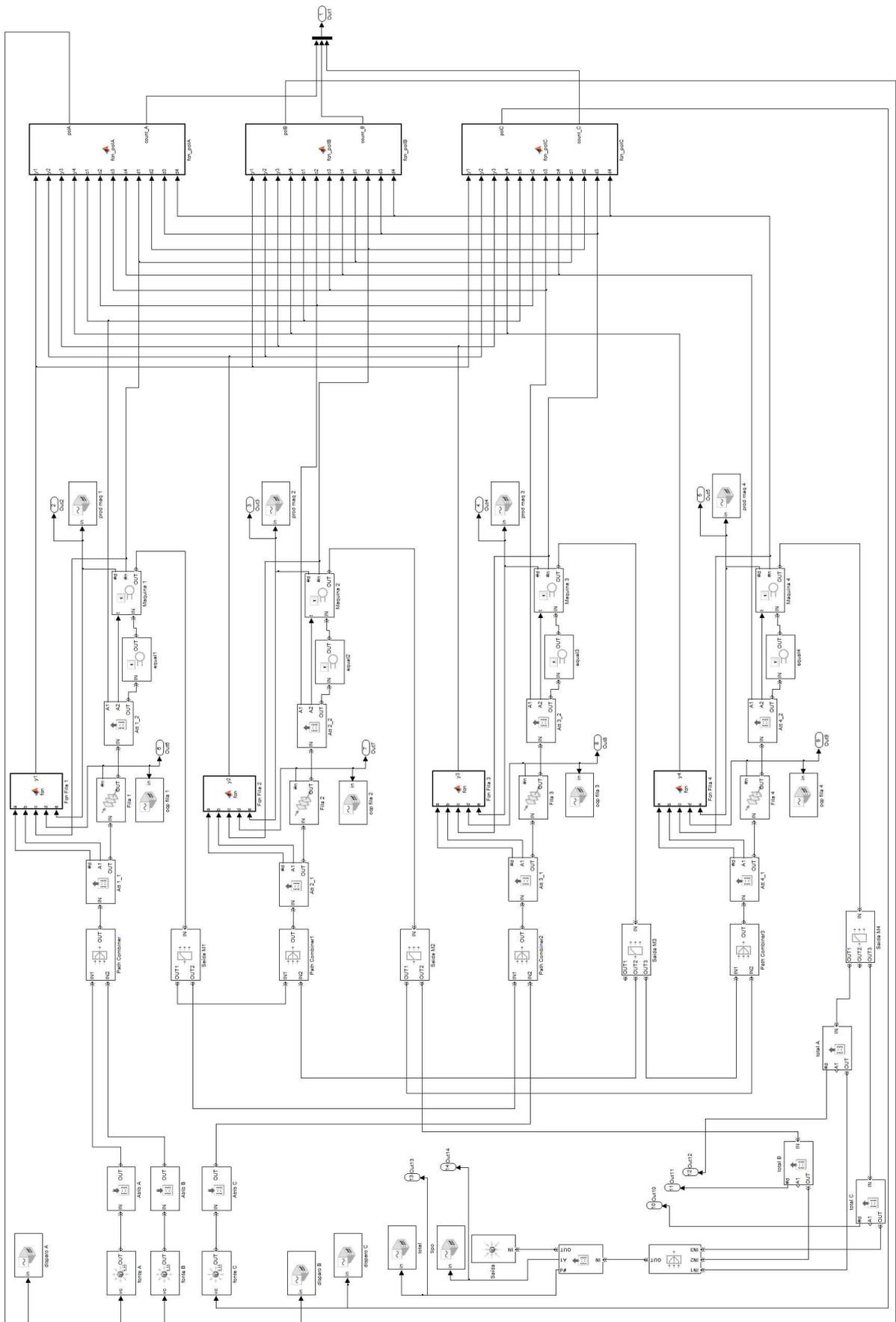


Figura 49 – Modelagem em *SimEvents*[®] do exemplo *jobshop 2*

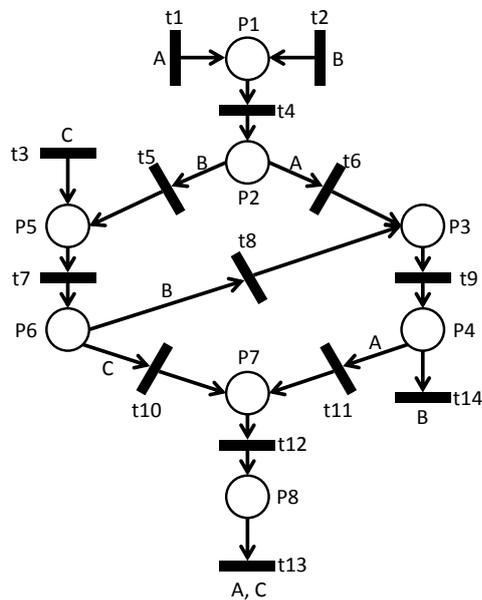


Figura 50 – Jobshop 2 representado por RdP

Tabela 40 – Relação entre transições da RdP e ações do MDP (*flowsops*)

Transição RdP	Ação MDP	Tipo de evento
T1	Ação 1	Controlável
T2	Ação 2	Controlável
T3	Ação 3	Controlável
T4	Ação 4	Não controlável
T5	Ação 5	Não controlável
T6	Ação 5	Não controlável
T7	Ação 4	Não controlável
T8	Ação 7	Não controlável
T9	Ação 4	Não controlável
T10	Ação 7	Não controlável
T11	Ação 6	Não controlável
T12	Ação 4	Não controlável
T13	Ação 8	Não controlável
T14	Ação 6	Não controlável

A tabela 41 mostra a heurística proposta para o *jobshop* 2. Para este exemplo, apenas uma heurística foi utilizada, sendo também baseada em observação, como nos casos anteriores.

Tabela 41 – Heurística proposta para o *jobshop* 2

Estado	Ação
00000000	Ação 1 (Insere A)
01000000	Ação 3 (Insere C)
00010300	Ação 2 (Insere B)
00010003	Ação 2 (Insere B)
00000301	Ação 2 (Insere B)
02010000	Ação 3 (Insere C)
02010003	Ação 3 (Insere C)
00020000	Ação 1 (Insere A)
00020100	Ação 1 (Insere A)
00020103	Ação 1 (Insere A)
01020000	Ação 3 (Insere C)
01020003	Ação 3 (Insere C)
02010000	Ação 3 (Insere C)
02010003	Ação 3 (Insere C)

A tabela 42 mostra as recompensas totais para as simulações sem recompensa na fonte, com recompensa na fonte e heurística. A figura 51 mostra graficamente os valores obtidos.

Este exemplo apresentou a utilidade total do *jobshop* com recompensa na fonte muito superior à heurística e *jobshop* sem recompensa na fonte. Além disso, a recompensa total da modelagem sem recompensa na fonte tornou-se maior conforme se aumentou o valor do fator de desconto, mostrando o acúmulo de recompensas no futuro. Alguns gráficos foram gerados para mostrar outras características deste sistema.

Para os três modelos foram escolhidos os valores de γ que geraram a maior recompensa total. Para o modelo sem recompensa na fonte, a maior recompensa total ocorre entre os valores $\gamma = 0,9$ e $\gamma = 0,99$. Para o modelo com recompensa na fonte, apenas um valor foi gerado entre $\gamma = 0,01$ e $\gamma = 0,99$. Para a heurística, apenas uma recompensa é calculada. A figura 52 mostra a produção final, por tipo de produto, que garantiu para cada modelo seu maior valor de recompensa total, apresentando maior versatilidade do modelo com recompensa na fonte em comparação ao modelo sem recompensa na fonte. O comportamento da heurística mostra a produção dos três tipos de produtos, porém era esperado que isto acontecesse devido à forma com que foi gerada (observação manual dos estados).

Outra análise que pode ser feita é a quantidade total de produção para cada tipo de modelo. Para os valores de fator de desconto com maior utilidade total, a figura 53 mostra

Tabela 42 – Recompensas totais (*jobshop* 2)

Modelo sem recompensa das transições fontes	
$\gamma=0,01$	2959,769237
$\gamma=0,1$	2959,769237
$\gamma=0,2$	2959,769237
$\gamma=0,3$	2959,769237
$\gamma=0,4$	2959,769237
$\gamma=0,5$	10154,17436
$\gamma=0,6$	10154,17436
$\gamma=0,7$	10154,17436
$\gamma=0,8$	10154,17436
$\gamma=0,9$	13469,82313
$\gamma=0,95$	13469,82313
$\gamma=0,99$	13469,82313
Modelo com recompensa das transições fontes	
$\gamma=0,01$	18924,58276
$\gamma=0,1$	18924,58276
$\gamma=0,2$	18924,58276
$\gamma=0,3$	18924,58276
$\gamma=0,4$	18924,58276
$\gamma=0,5$	18924,58276
$\gamma=0,6$	18924,58276
$\gamma=0,7$	18924,58276
$\gamma=0,8$	18924,58276
$\gamma=0,9$	18924,58276
$\gamma=0,95$	18924,58276
$\gamma=0,99$	18924,58276
Heurística aplicada	
Recompensa	10450,36782

a diferença de quantidade produzida entre modelo com recompensa na fonte e demais modelos. Embora este gráfico mostre apenas quantas unidades cada modelo produziu, este é um resultado importante devido à diferença de produção observada. Isto se deve a princípio pela priorização dos disparos das fontes. Ao se analisar a política, de 15309 estados possíveis neste sistema, a quantidade de estados com ação em transição fonte varia entre 8319 e 8505, o que significa inserção de peças para cerca de 55% de todos os estados. Já o *jobshop* sem recompensa na fonte obteve no máximo 1062 estados previstos com ação na fonte, o que representa cerca de 7% dos estados possíveis. A variação das ações entre produzir A e produzir C para o modelo com recompensa na fonte também influenciou no resultado, pois fez com que sua produção fosse quase o dobro que a produção do modelo sem recompensa na fonte. Isto se deve ao fato de haver diferentes rotas de produção para A e C, o que possibilitou a produção dos dois tipos de peças ao mesmo tempo, em máquinas diferentes.

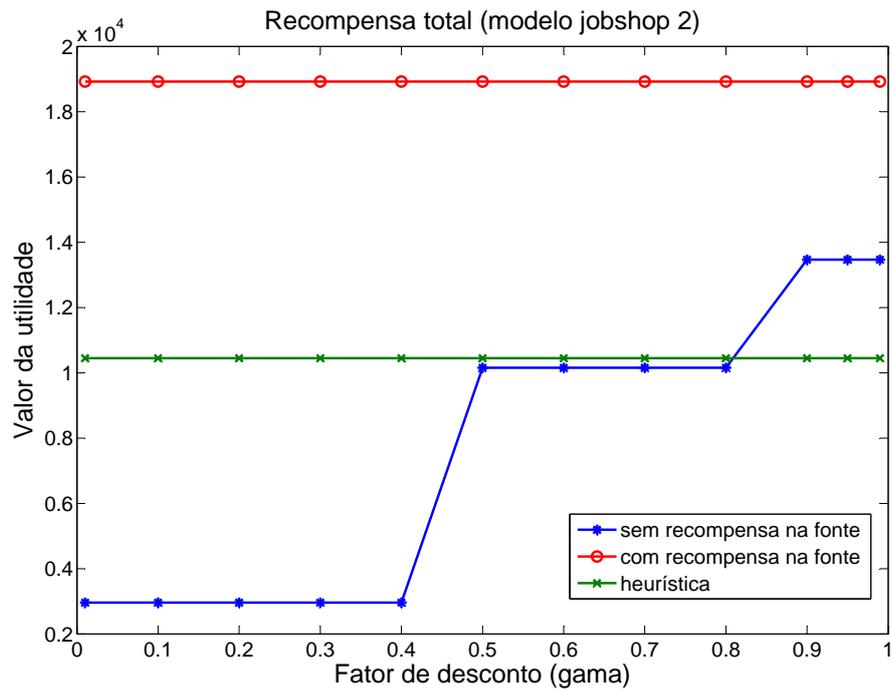


Figura 51 – Variação das recompensas totais (jobshop 2)

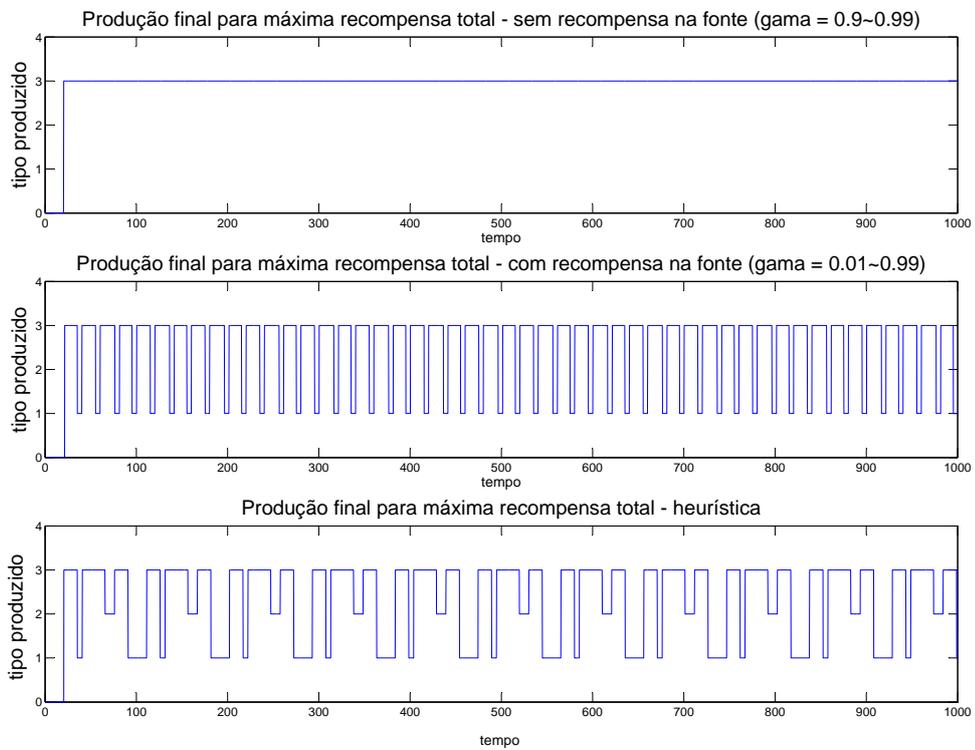
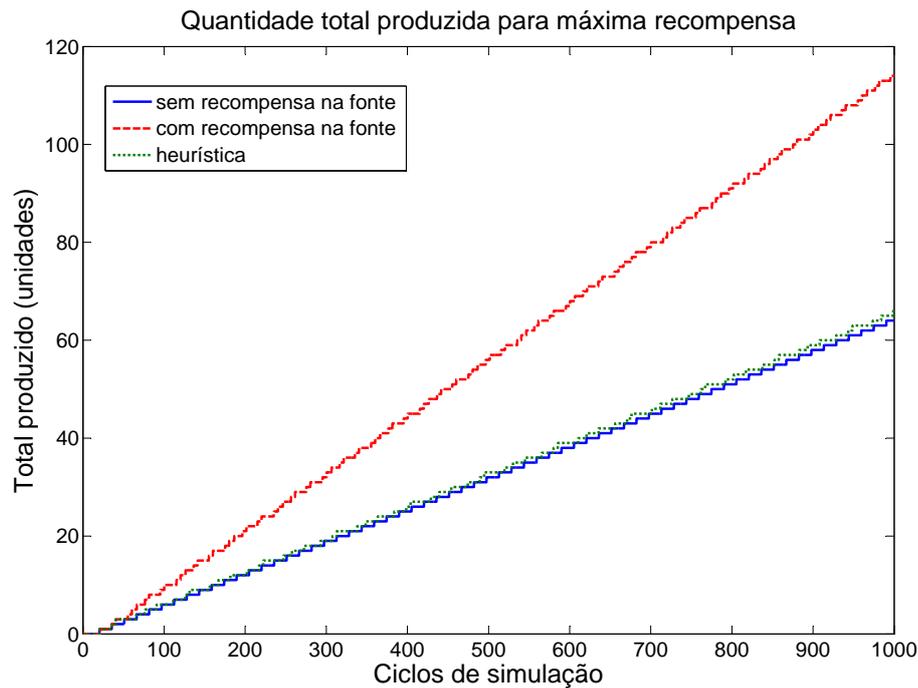


Figura 52 – Tipo de peças produzidas para máxima recompensa total (jobshop 2)

Figura 53 – Quantidade total produzida (*jobshop 2*)

Também levando em consideração o modelo com recompensa na fonte, pode-se verificar o comportamento das filas durante o processamento. Este caso específico é interessante, pois há o processamento de produto tipo A e tipo C, o que garante que todas as máquinas serão utilizadas no processo, havendo inclusive máquina comum ao processamento dos dois produtos (máquina 4). A figura 54 mostra que foram utilizadas as filas das máquinas 1 e 3, sendo que apenas A passa pela máquina 1 e apenas C passa pela máquina 3. Desta forma, foi evitada que se formasse fila na máquina 4, que, analisando as rotas de produção, é o gargalo do sistema por ser a única máquina compartilhada.

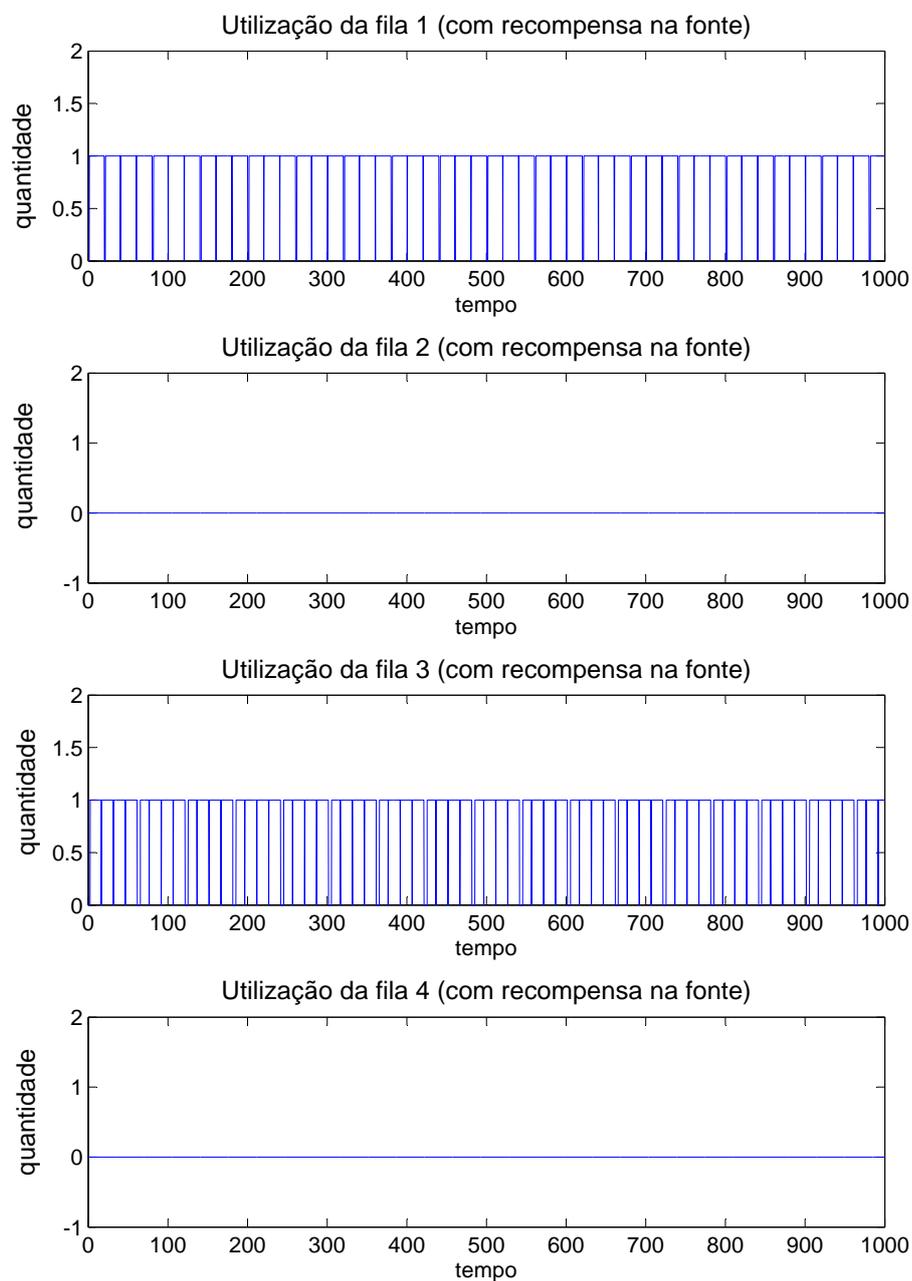


Figura 54 – Comportamento das filas para modelo com recompensa na fonte

5 Conclusões e Trabalhos Futuros

Neste trabalho foi aplicada a tomada de decisões em tempo real em sistema flexíveis de manufatura (FMS). A primeira abordagem foi sobre a modelagem dos FMS em redes de *Petri* coloridas limitadas. O uso desta ferramenta permitiu a que fossem modelados sistemas de produção que devem obedecer a certas restrições no que diz respeito a rotas e capacidade de recursos. Além disso, o uso das redes de *Petri* coloridas torna-se importante devido ao seu poder de representação de *tokens* com diferentes tipos de informações. Isto se faz necessário, pois um dos objetivos do FMS é a fabricação de diferentes tipos de produtos numa mesma linha de produção.

Feita a modelagem, a proposta era a aplicação de um agente tomador de decisão, que é a principal característica do FMS em relação a outros tipos de linhas de produção. A aplicação dos processos decisórios de *Markov* (MDP) é possível, pois através da árvore de alcançabilidade de uma RdP consegue-se determinar o conjunto de estados markovianos de um sistema. Esta é a primeira contribuição deste trabalho, uma vez que esta determinação pode ser feita utilizando exclusivamente as propriedades das RdPs.

A partir da árvore de alcançabilidade do FMS modelado por RdP foi possível então determinar também as matrizes de transições e recompensas do MDP. O algoritmo que recebe as informações da RdP, gera o conjunto de estados alcançáveis e determina os parâmetros do MDP é o principal desenvolvimento deste trabalho, já que este organiza a RdP, aplica as restrições do sistema, verifica as ações executadas e determina toda dinâmica segundo a regra de disparos.

As políticas do MDP são geradas e utilizadas na simulação dos FMS pelo pacote *SimEvents*[®] do software matemático *MatLab*[®]. Foram feitas simulações em duas classes de sistemas de manufatura, os *flowshops* e os *jobshops*, que se caracterizam pela presença de rotas bem definidas de produção. As simulações foram feitas segundo três métodos: (1) sistemas com recompensa na fonte, (2) sistemas sem recompensa na fonte, e (3) heurística manual de produção.

O resultado das simulações mostrou que o MDP toma decisões que visam a receber a maior recompensa acumulada, segundo um fator de desconto. Além disso, existe uma tendência no melhor resultado caso seja considerada recompensa nas fontes do sistema, pois, desta forma, prioriza-se a ação de inserção de novas peças no sistema. As heurísticas dependem do conhecimento empírico para que sejam obtidos bons resultados.

O desenvolvimento deste trabalho mostrou também que há algumas desvantagens

no uso destas técnicas para a modelagem e tomada de decisão. Primeiramente, um FMS modelado em RdP pode tomar grandes dimensões devido à presença de *buffers* com alta capacidade e diferentes tipos de famílias de produtos. A quantidade de estados de um sistema, por menor que seja, cresce exponencialmente ao se adicionar um novo produto na linha de manufatura ou aumentando a capacidade de um *buffer*, mesmo de forma unitária. Isto mostra o elevado grau de sensibilidade do sistema com relação à quantidade de estados. Além disso, a política do MDP define qual ação o sistema deve tomar em determinado estado, mas não evita que o sistema possa atingir certo estado que não é de interesse. Este fato é exposto nas considerações de capacidade de *buffers*, pois nas simulações do modelo em *SimEvents*[®] não evitou que filas com tamanho maior que o modelado em RdP transbordassem no decorrer do processo.

O objetivo deste trabalho, como aplicação dos processos decisórios de *Markov* em sistemas flexíveis de manufatura a partir da modelagem por redes de *Petri*, foi atingido, mesmo tendo sido encontrados estes problemas que limitaram o poder destas ferramentas. De forma a progredir no estudo deste tema, alguns temas de trabalhos futuros são apresentados.

Primeiramente, toda máquina que faz parte de um sistema de manufatura tem um tempo de processamento para um determinado tipo de produto. O tempo de processamento foi utilizado neste trabalho para a determinação das recompensas das máquinas, pois foi considerada uma função de utilidade dependente do tempo de processamento para cada uma delas. O modelo, no entanto, pode apresentar melhor desempenho caso os tempos de produção sejam considerados através de redes de *Petri* temporizadas ou TiMDP (Processos decisórios de *Markov* dependentes do Tempo). Além disso, o uso da teoria de controle supervisorio pode trazer uma alternativa ao MDP, de forma a eliminar transições que possam levar o sistema a estados indesejados (como o caso dos *buffers* na simulação em *SimEvents*[®]). Sugere-se também a repetição da metodologia apresentada em outros modelos de sistemas de forma a fazer uma melhor análise entre as recompensas adquiridas nas máquinas e valores agregados nos produtos, uma vez que o MDP utiliza esta recompensa total para a determinação da política ótima.

Referências

- AALST, W. Van der. Putting high-level petri nets to work in industry. *Computers in Industry*, 1994. Elsevier, v. 25, n. 1, p. 45–54, 1994. Citado 2 vezes nas páginas 16 e 39.
- AGERWALA, T. Special feature: Putting petri nets to work. *Computer*, 1979. IEEE, v. 12, n. 12, p. 85–94, 1979. Citado 2 vezes nas páginas 16 e 39.
- BASTOS, G. S. *Methods for Truck Dispatching in Open-Pit Mining*. Tese (Thesis of Doctor in Science) — Technological Institute of Aeronautics, São José dos Campos, 2010. Citado 5 vezes nas páginas 15, 26, 27, 29 e 53.
- BASTOS, G. S. et al. Time-dependent utility decision making using the timdp model. *X SBAI (Simpósio Brasileiro de Automação Inteligente)*, 2011. X, p. 647–653, 2011. ISSN 2175-8905. Citado 2 vezes nas páginas 27 e 54.
- BATA, R. et al. Petri nets application for management of biodegradable components of municipal waste. *Wseas Transactions on Environment and Development*, 2008. v. 4, p. 1057–1066, 2008. ISSN 1790-5079. Citado na página 25.
- BATISTA, E. Fordismo, taylorismo e toyotismo: apontamentos sobre suas rupturas e continuidades. *III Simpósio Lutas Sociais da América Latina. GEPAL. Londrina*, 2008. 2008. Citado na página 14.
- BAUSE, F.; KRITZINGER, P. S. *Stochastic Petri Nets*. [S.l.]: Springer, 2002. Citado na página 26.
- BILLINGTON, J. *Extensions to coloured Petri nets and their application to protocols*. Tese (Doutorado) — University of Cambridge, 1991. Citado 2 vezes nas páginas 50 e 51.
- BRESSAN, G. Modelagem e simulação de sistemas computacionais. *Capítulo sobre Redes de Petri, LARC-PCS/EPUSP*, 2002. 2002. Citado 3 vezes nas páginas 19, 21 e 22.
- CARDOSO, J.; VALETTE, R. *Redes de Petri*. [S.l.]: Editora da UFSC, 1997. Citado na página 22.
- CHADÈS, I. et al. *Markov Decision Processes (MDP) Toolbox*. 2009. Citado na página 60.
- CHAPLIN, C. Modern timer. *Roteiro: Charles Chaplin. Los Angeles: Charles Chaplin Productions*, 1936. 1936. Citado 2 vezes nas páginas 14 e 35.
- COSTA, A. B. da. Inovações e mudanças na organização industrial. *Ensaio FEE*, 2000. v. 21, n. 2, 2000. Citado na página 14.
- EBOLI, M. G. *Transformação de redes de Petri coloridas em processos de decisão markovianos com probabilidades imprecisas*. Dissertação (Dissertação de mestrado) — Universidade de São Paulo, 2010. Citado 2 vezes nas páginas 25 e 39.

- FERNÁNDEZ, C. *Interfaces de Comunicação para o software CPN Tools*. [S.l.], 2010. Citado na página 22.
- FESTO. *Logistics League Robocup 2012 Rulebook*. [S.l.], 2012. Citado 4 vezes nas páginas 8, 15, 32 e 33.
- GROOVER, M. P. *Automation, production systems, and computer-integrated manufacturing*. [S.l.]: Prentice Hall Press, 2007. Citado na página 34.
- HUBER, P. et al. Reachability trees for high-level petri nets. *Theoretical Computer Science*, 1986. Elsevier, v. 45, p. 261–292, 1986. Citado 3 vezes nas páginas 8, 47 e 48.
- JENSEN, K. A brief introduction to coloured petri nets. In: *Tools and Algorithms for the Construction and Analysis of Systems*. [S.l.]: Springer, 1997. p. 203–208. ISBN 3-540-06790-1. Citado 3 vezes nas páginas 16, 23 e 24.
- JENSEN, K.; KRISTENSEN, L. M.; WELLS, L. Coloured petri nets and cpn tools for modelling and validation of concurrent systems. *International Journal on Software Tools for Technology Transfer*, 2007. Springer, v. 9, n. 3-4, p. 213–254, 2007. Citado na página 24.
- LAROUSSE, C. Grande enciclopédia.(1998). *São Paulo: Nova Cultural Ltda*, 1998. 1998. Citado 2 vezes nas páginas 14 e 15.
- LIN, C.; MARINESCU, D. C. Reachability trees for high level petri nets with marking variables. 1989. 1989. Citado na página 47.
- LITTMAN, M. L.; DEAN, T. L.; KAELBLING, L. P. On the complexity of solving markov decision problems. In: MORGAN KAUFMANN PUBLISHERS INC. *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*. [S.l.], 1995. p. 394–402. Citado 2 vezes nas páginas 15 e 26.
- MATCOVSCHI, M.-H.; MAHULEA, C.; PASTRAVANU, O. Petri net toolbox for matlab. In: *11th IEEE Mediterranean Conference on Control and Automation MED'03*. [S.l.: s.n.], 2003. Citado 2 vezes nas páginas 76 e 89.
- MATOS, R. et al. Planejamento de capacidade de sistemas através de cadeias de markov. *Anais do ERCEMAPI 2011 - Escola Regional de Computação de Ceará, Maranhão e Piauí*, 2011. 2011. Citado na página 22.
- MEMMI, G.; FINKEL, A. An introduction to fifo nets—monogeneous nets: A subclass of fifo nets. *Theoretical Computer Science*, 1985. Elsevier, v. 35, p. 191–214, 1985. Citado na página 51.
- MOLINA, J. R. C. *Uma abordagem híbrida para o controle de sistemas de manufatura baseada na teoria de controle supervisorio e nas redes de Petri coloridas*. Dissertação (Master's thesis) — PUC-PR, 2007. Citado 2 vezes nas páginas 40 e 84.
- MORAES, C. C. D.; CASTRUCCHI, P. B. de L. *Engenharia de automação industrial*. [S.l.]: Livros Tecnicos e Científicos, 2001. Citado na página 46.
- MUNIZ, J. I. R. Uso da tecnologia de grupos na automação industrial: Estudando problemas de job-shop. 2009. 2009. Citado 2 vezes nas páginas 34 e 36.

- MURATA, T. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 1989. IEEE, v. 77, n. 4, p. 541–580, 1989. Citado 6 vezes nas páginas 16, 17, 18, 19, 25 e 39.
- PELLEGRINI, J.; WAINER, J. Processos de decisão de markov: um tutorial. *Revista de Informática Teórica e Aplicada*, 2007. v. 14, n. 2, p. 133–179, 2007. Citado 4 vezes nas páginas 15, 26, 27 e 28.
- PETRI, C. A. *Kommunikation mit Automaten*. [S.l.], 1962. Citado 2 vezes nas páginas 16 e 17.
- PINEDO, M. *Scheduling: theory, algorithms, and systems*. [S.l.]: Springer Science+Business Media, 2012. Citado 3 vezes nas páginas 15, 26 e 50.
- RAMADGE, P. J.; WONHAM, W. M. The control of discrete event systems. *Proceedings of the IEEE*, 1989. IEEE, v. 77, n. 1, p. 81–98, 1989. Citado 2 vezes nas páginas 15 e 33.
- REIS, J. Uma introdução ao scheduling. *Relatório Interno ISCTE - DCTI*, 1996. 1996. Citado 2 vezes nas páginas 26 e 35.
- ROUCAIROL, G. Fifo-nets. In: *Proceedings of an advances course in Petri nets: Central Models and Their Properties*. [S.l.]: Springer, 1987. p. 436–459. ISBN 978-0136042594. Citado na página 51.
- RUSSELL, S. J. et al. *Artificial intelligence: a modern approach*. [S.l.]: Prentice hall Englewood Cliffs, 2010. Citado na página 46.
- SALES, A. H. C. de. Um estudo sobre redes de petri estocásticas generalizadas. 2002. 2002. Citado na página 22.
- SILVA, D. B. *Aplicação da teoria de controle supervisorio no projeto de controladores para sistemas de rota variável centrado em robô PPGEPS*. Dissertação (Master's thesis) — PUC-PR, 2007. Citado 2 vezes nas páginas 39 e 84.
- SILVA, M.; VALETTE, R. Petri nets and flexible manufacturing. In: *Advances in Petri nets 1989*. [S.l.]: Springer, 1990. p. 374–417. Citado 5 vezes nas páginas 15, 33, 37, 38 e 39.
- THE MATHWORKS. *SimEvents 3 Getting Started Guide*. EUA, 2005–2010. Citado na página 41.
- VEUNG, D. et al. Fuzzy coloured petri nets in modelling flexible manufacturing systems. In: IEEE. *ISAI/IFIS 1996. Mexico-USA Collaboration in Intelligent Systems Technologies. Proceedings*. [S.l.], 1996. p. 100–107. Citado na página 16.

APÊNDICE A – Definição da expressão de cálculo da quantidade de estados do sistema

O objetivo deste apêndice é mostrar a dedução da expressão utilizada para determinação da quantidade de estados gerada em um FMS modelado em RdP, principalmente se consideradas filas FIFO.

De acordo com o apresentado na seção 3.2.3, a representação de filas FIFO faz com que alguns estados não sejam atingidos. Ao entrar numa fila, uma determinada peça ocupa a primeira posição livre da mesma, o que se repete até que a fila esteja totalmente cheia. Se a máquina seguinte a esta fila está ociosa, a peça que ocupa a primeira posição da fila é enviada para processamento, e todo o conteúdo restante da fila passa a ocupar uma posição à frente. Assim, estados do sistema que tenham posições livres da fila entre posições ocupadas não são atingíveis, tal como mostrado na tabela 9. Assim, parte-se do princípio que todas as filas do sistema são preenchidas a partir da primeira para a última posição. A figura 55 esquematiza uma fila de 6 posições e o sentido percorrido pelas entidades que passam por ela.

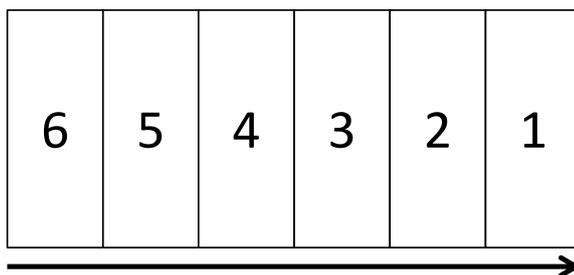


Figura 55 – Fila de 6 posições

Outro fator que deve ser considerado é a quantidade de cores admitidas em um determinado lugar da rede. Esta informação é importante pois cada combinação de produtos em uma fila gera um estado diferente do sistema. De forma a mostrar como a capacidade da fila e os diferentes tipos de cores admitidos no FMS influenciam na quantidade de esta-

dos produzidos por um lugar da RdP e então deduzir a expressão 3.2, sejam consideradas duas filas com três posições, a primeira admitindo apenas cor A e a segunda cores A e B .

Na figura 56, partindo do princípio que a fila sempre é ocupada da primeira para a última posição e que a fila vazia é sempre o estado inicial do lugar que a representa, são listados os 4 estados possíveis. Estando a fila vazia, pode-se apenas chegar uma unidade de A , que ocupa a primeira posição. Após isso, caso a fila não seja esvaziada, apenas chegarão peças A até que a fila fique cheia. Assim, haverá apenas um novo estado para cada nova peça que é adicionada à fila.

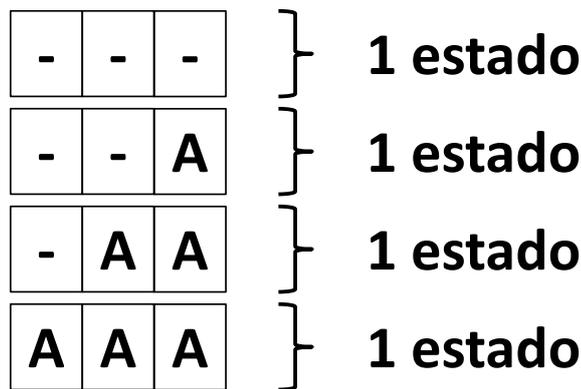


Figura 56 – Fila de 3 posições para produto A

Já a figura 57 mostra todos os estados formados pela mesma fila de 3 posições, porém admitindo peças tipo A e B . Agora, cada nova peça que entra na fila pode ser A ou B . Partindo da fila vazia (1 estado), podem haver dois estados subsequentes (para A ou B que entre na fila). Estando a primeira posição da fila ocupada, a próxima peça a entrar na fila pode novamente ser A ou B , porém agora, para que sejam definidos todos os estados possíveis, deve-se fazer a combinação da nova peça com a peça já presente na fila. Assim, existem 2 estados possíveis caso A entre na fila e 2 estados caso B seja adicionada.

O total de estados gerados pela fila é a soma das combinações feitas para cada nova posição da fila que é ocupada e para cada cor admitida no lugar. Assim, no caso da figura 57, a soma é:

$$2^0 + 2^1 + 2^2 + 2^3 = 1 + 2 + 4 + 8 = 15$$

Desta forma a dedução da expressão é intuitiva, considerando a capacidade da fila e a quantidade de cores admitida por ela, o que resulta em:

$$S_{l_i} = \sum_{j=0}^{n_pos_i} n_c_i^j$$



Figura 57 – Fila de 3 posições para produtos *A* e *B*

onde S_{l_i} é a quantidade de estados gerada pelo lugar i , n_{c_i} é a quantidade de cores admitida no lugar i e n_{pos_i} é a capacidade do lugar. Vale lembrar que esta expressão determina a quantidade de estados gerados por apenas uma fila. No caso de todo o sistema, este total é definido pelo produto da quantidade de estados gerada por todos os lugares da rede.