

UNIVERSIDADE FEDERAL DE SÃO CARLOS  
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA  
PROGRAMA DE PÓS-GRADUAÇÃO EM  
CIÊNCIA DA COMPUTAÇÃO

Estratégia de Modelagem por Algoritmo Genético Adaptativo para  
Programação Reativa da Produção de Produtos com uso Simultâneo  
de Máquinas e Sistemas de Transporte em Sistemas de Manufatura

**Danilo Sipoli Sanches**

**São Carlos  
Setembro/2008**

**Ficha catalográfica elaborada pelo DePT da  
Biblioteca Comunitária da UFSCar**

S211em

Sanches, Danilo Sipoli.

Estratégia de modelagem por algoritmo genético adaptativo para programação reativa da produção de produtos com uso simultâneo de máquinas e sistemas de transporte em sistemas de manufatura / Danilo Sipoli Sanches. -- São Carlos : UFSCar, 2008.

88 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2008.

1. Programação da produção. 2. Inteligência artificial. 3. Sistemas automatizados de manufatura. 4. Sistemas de veículos auto-guiados. I. Título.

CDD: 006.3 (20<sup>a</sup>)

# Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

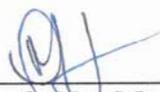
Programa de Pós-Graduação em Ciência da Computação

## “Estratégia de Modelagem por Algoritmo Genético Adaptativo para Programação Reativa da Produção de Produtos com Uso Simultâneo de Máquinas e Sistemas de Transporte em Sistemas de Manufatura”

DANILO SIPOLI SANCHES

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Membros da Banca:



---

Prof. Dr. Orides Morandin Júnior  
(Orientador – DC/UFSCar)



---

Prof. Dr. Edilson Reis Rodrigues Kato  
(Co-Orientador – DC/UFSCar)



---

Prof. Dr. Roberto Hideaki Tsunaki  
(Engenharia Mecânica/USP)



---

Prof. Dr. Sebastián Alberto Urrutia  
(DCC/UFMG)

São Carlos  
Setembro/2008

*Dedico este trabalho aos meus pais,  
Nelson e Márcia, pelo apoio e  
dedicação todos os dias.*

# AGRADECIMENTOS

Primeiro e, principalmente, a Deus por ter me dado tudo o que tenho.

Aos meus pais, Nelson e Márcia, e meu irmão, Murilo, pela ajuda e conselhos em todos momentos.

À Fernanda, pela constante ajuda, incentivo e carinho nos momentos difíceis.

Ao meu orientador, Prof. Dr. Orides Morandin Jr., pela oportunidade oferecida, pelos ensinamentos transmitidos e pelo companheirismo e amizade desenvolvidos durante todo este tempo.

Ao meu co-orientador, Prof. Dr. Edílson Reis Rodrigues Kato, pela disposição, apoio e amizade.

Aos amigos Ageu e Flávio, pela amizade e conhecimentos compartilhados.

Aos amigos da República Eclipse, pela amizade construída e pela compreensão.

Aos professores do Departamento de Computação, da Universidade Federal de São Carlos, pelo apoio e disposição.

A todas as outras pessoas que, direta ou indiretamente, contribuíram para a realização deste trabalho.

## *Resumo*

**SANCHES, Danilo S.** *Estratégia de Modelagem por Algoritmo Genético Adaptativo para Programação Reativa da Produção de Produtos com uso Simultâneo de Máquinas e Sistemas de Transporte em Sistemas de Manufatura. Dissertação de Mestrado; Departamento de Computação, Universidade Federal de São Carlos; Julho de 2008.*

O problema da programação da produção de produtos com uso simultâneo de máquinas e sistemas de transporte em sistemas de manufatura envolvem questões como a modelagem do problema e a técnica utilizada para resolvê-lo. Este tipo de programação é caracterizado pela grande quantidade de soluções possíveis, em que várias pesquisas apontam para o uso de Algoritmos Genéticos Adaptativos como método de busca, uma vez que estes algoritmos possuem a capacidade de percorrer de forma global o espaço da busca, a fim de encontrar boas soluções rapidamente. Neste trabalho, é proposto um método com uso de um algoritmo genético adaptativo para resolver este tipo de problema de programação. O objetivo deste trabalho é obter uma boa programação da produção de produtos, a fim de atingir um bom compromisso entre valores de *makespan* e de tempo de obtenção da resposta. Os valores de *makespan* são obtidos a partir da aplicação do algoritmo genético adaptativo e o tempo de obtenção da resposta é referente ao tempo de processamento do algoritmo genético adaptativo. Os resultados deste trabalho foram validados para cenários pequenos e grandes e comparados com os resultados de outras duas abordagens. Estes resultados são apresentados e discutidos neste trabalho.

**Palavras-chave:** Programação da Produção, Sistemas de Manufatura, Sistemas de Transporte, Algoritmos Genéticos Adaptativos, Veículos Auto-Guiados, Sistemas Automatizados de Manufatura.

## *Abstract*

**SANCHES, Danilo S.** *Modelling Strategy by Adaptive Genetic Algorithm for Production Reactive Scheduling with Simultaneous use of Machines and Transportation Systems in Manufacturing Systems. Dissertação de Mestrado*; Departamento de Computação, Universidade Federal de São Carlos; Julho de 2008.

The production scheduling problem of products with simultaneous use of machines and transportation systems in manufacturing systems involves the system modeling task and the application of a technique to solve it. This scheduling type is characterized by the great amount of possible solutions and several researches indicate the Adaptive Genetic Algorithms as search method to solve this problem, where these algorithms have the capacity of globally explore the search space and to find good solutions quickly. In this dissertation, it is proposed a method that uses an adaptive genetic algorithm to solve this scheduling problem. The aim of this dissertation is to obtain good scheduling of product production, in order to reach good makespan values and response obtaining time. The makespan values are obtained by adaptive genetic algorithm and the response obtaining time is the processing time of the adaptive genetic algorithm. The results were validated in small and large scenarios and compared with the results of two other approaches. These results are presented and discussed in this dissertation.

**Palavras-chave:** Production Scheduling, Manufacturing Systems, Transportation Systems, Adaptive Genetic Algorithm, Automated Guided Vehicle, Automated Manufacturing Systems.

## LISTA DE ABREVIATURAS

A*	Método de Busca Heurística ( <i>A star</i> )
AG	Algoritmo Genético
AGA	Algoritmo Genético Adaptativo
AGV	Veículo Auto Guiado ( <i>Automated Guided Vehicle</i> )
AHA	Algoritmo Heurístico de Ajustamento ( <i>Adjustment Heuristic Algorithm</i> )
C1	Primeiro Cromossomo Pai
C2	Segundo Cromossomo Pai
C/D	Estação de Carga e Descarga
F1	Primeiro Cromossomo Filho
F2	Segundo Cromossomo Filho
FMS	Sistemas Flexíveis de Manufatura ( <i>Flexible Manufacturing Systems</i> )
GD	Algoritmo Genético com Gene Dominante
M	Máquina
P	Produto
PCP	Planejamento e Controle da Produção
PN	Redes de Petri ( <i>Petri Nets</i> )
RV	Veículo Aleatório ( <i>Random Vehicle rule</i> )
STT/D	Menor Tempo de Viagem ( <i>Shortest Travel Time/Distance</i> )
TMF	Tempo de Utilização Final da Máquina
TMI	Tempo de Utilização Inicial da Máquina
TVF	Tempo de Utilização Final do Veículo
TVI	Tempo de Utilização Inicial do Veículo
V	Veículo
WIP	Número de Peças Sendo Processadas em um Ciclo ( <i>Work in process</i> )

## LISTA DE TABELAS

Tabela 1	Produtos e roteiros de fabricação.....	55
Tabela 2	Roteiros do problema 1 .....	70
Tabela 3	Tempos de operação do problema 1.....	70
Tabela 4	Tempos de transporte do problema 1 .....	71
Tabela 5	Roteiros do problema 2 .....	71
Tabela 6	Tempos de operação do problema 2.....	72
Tabela 7	Tempos de transporte do problema 2.....	72
Tabela 8	Resultados obtidos para o problema 1.....	73
Tabela 9	Resultados obtidos para o problema 2.....	74

## LISTA DE FIGURAS

Figura 1	População de cromossomos.....	23
Figura 2	Método de seleção por roleta.....	25
Figura 3	Método de seleção por torneio.....	25
Figura 4	Método de seleção por amostragem universal estocástica .....	26
Figura 5	Cruzamento de um ponto.....	27
Figura 6	Cruzamento de dois pontos.....	27
Figura 7	Cruzamento uniforme .....	28
Figura 8	Operador de mutação .....	28
Figura 9	Fluxograma do algoritmo genético adaptativo .....	52
Figura 10	Representação completa do cromossomo.....	53
Figura 11	Representação simplificada do cromossomo.....	54
Figura 12	Cromossomo .....	55
Figura 13	Cromossomo simplificado.....	55
Figura 14	Cruzamento em relação ao produto $P_{11}$ .....	56
Figura 15	Mutação .....	56
Figura 16	Exemplo de um cromossomo.....	58
Figura 17	Gráfico de Gantt.....	58
Figura 18	Tela inicial do sistema.....	61
Figura 19	Menu cadastro.....	62
Figura 20	Parâmetros do AGA .....	63
Figura 21	Execução da programação .....	63
Figura 22	Resultados armazenados em um arquivo texto.....	64
Figura 23	Resultados armazenados em uma planilha .....	65

Figura 24	Resultado após a execução do teste com o JUnit.....	67
Figura 25	Gráfico de convergência da Proposta para o problema 1 .....	76
Figura 26	Gráfico de convergência da Proposta para o problema 2 .....	77
Figura 27	Gráfico de convergência de R&R para o problema 1 .....	77
Figura 28	Gráfico de convergência de R&R para o problema 2 .....	78
Figura 29	Gráfico de convergência de Morandin para o problema 1 .....	79
Figura 30	Gráfico de convergência de Morandin para o problema 2 .....	79

# SUMÁRIO

<b>1</b>	<b>INTRODUÇÃO</b> .....	13
1.1	Caracterização do problema e do ambiente .....	13
1.2	Problema abordado .....	15
1.3	Objetivos .....	15
1.3.1	Objetivo geral .....	15
1.3.2	Objetivos específicos .....	15
1.4	Resultados alcançados.....	16
1.5	Metodologia.....	16
1.6	Estrutura do trabalho .....	18
<b>2</b>	<b>ALGORITMOS GENÉTICOS</b> .....	20
2.1	Considerações iniciais .....	20
2.2	Algoritmos genéticos .....	20
2.3	Funcionamento .....	22
2.4	Representação dos parâmetros.....	22
2.5	Seleção .....	24
2.5.1	Método da roleta.....	24
2.5.2	Método do torneio .....	25
2.5.3	Método da amostragem universal estocástica .....	25
2.6	Operadores genéticos .....	26
2.7	Operadores genéticos para permutações .....	28
2.8	Controle dos parâmetros genéticos .....	29
2.9	Algoritmos genéticos adaptativos .....	30
2.10	Considerações finais.....	31
<b>3</b>	<b>USO DE ALGORITMOS GENÉTICOS PARA A PROGRAMAÇÃO DA</b>	

<b>PRODUÇÃO .....</b>	<b>32</b>
3.1 <b>Considerações iniciais .....</b>	<b>32</b>
3.2 <b>Uso de algoritmos genéticos tradicionais para a programação da produção. ....</b>	<b>32</b>
3.3 <b>Uso de algoritmos genéticos adaptativos para a programação da produção. ....</b>	<b>38</b>
3.4 <b>Programação simultânea da produção de máquinas e sistemas de transporte .....</b>	<b>43</b>
3.5 <b>Considerações finais.....</b>	<b>47</b>
<b>4 PROPOSTA DE MODELAGEM POR ALGORITMO GENÉTICO ADAPTATIVO PARA PROGRAMAÇÃO REATIVA DA PRODUÇÃO DE PRODUTOS COM USO SIMULTÂNEO DE MÁQUINAS E SISTEMAS DE TRANSPORTE EM SISTEMAS DE MANUFATURA .....</b>	<b>48</b>
4.1 <b>Considerações iniciais .....</b>	<b>48</b>
4.2 <b>Caracterização e restrições da programação da produção proposta.....</b>	<b>49</b>
4.3 <b>Modelagem do algoritmo genético adaptativo .....</b>	<b>51</b>
4.3.1 <b>Codificação do cromossomo .....</b>	<b>53</b>
4.3.1.1 <b>Exemplo de codificação do cromossomo .....</b>	<b>54</b>
4.3.2 <b>Seleção.....</b>	<b>55</b>
4.3.3 <b>Cruzamento.....</b>	<b>56</b>
4.3.4 <b>Mutação.....</b>	<b>56</b>
4.3.5 <b>Função de <i>fitness</i> .....</b>	<b>57</b>
4.3.5.1 <b>Programação da produção dos veículos usando regras de despacho ..</b>	<b>57</b>
4.3.5.2 <b>Exemplo de uma programação da produção de produtos com uso simultâneo de máquinas e veículos.....</b>	<b>58</b>
4.3.5.3 <b>Obtendo o valor de <i>makespan</i> .....</b>	<b>59</b>
4.3.6 <b>Ajuste dinâmico das taxas de mutação e cruzamento .....</b>	<b>60</b>

4.4	Considerações finais.....	61
5	<b>DESENVOLVIMENTO E TESTES DO SISTEMA PARA A PROGRAMAÇÃO DA PRODUÇÃO .....</b>	<b>62</b>
5.1	Considerações iniciais .....	62
5.2	Sistema implementado .....	62
5.3	Aplicação de testes sobre o sistema desenvolvido .....	66
5.4	Considerações finais.....	68
6	<b>RESULTADOS E DISCUSSÕES .....</b>	<b>69</b>
6.1	Considerações iniciais .....	69
6.2	Caracterização do ambiente de testes.....	70
6.3	Testes realizados.....	70
6.4	Resultados obtidos.....	73
6.4.1	Desempenho do algoritmo genético adaptativo .....	76
6.5	Considerações finais.....	80
7	<b>CONCLUSÃO .....</b>	<b>82</b>
7.1	Trabalhos futuros.....	83
	<b>Referências .....</b>	<b>85</b>

# 1 INTRODUÇÃO

## 1.1 Caracterização do Problema e do Ambiente

Nos últimos anos, as indústrias de manufatura têm sofrido uma enorme transformação devido à utilização crescente de novas tecnologias de processo, tais como máquinas ferramentas de controle numérico computadorizadas, robôs industriais, sistemas de manipulação e transporte de materiais (Reddy; Rao, 2006).

Este contexto evidencia uma mudança no enfoque que coloca a manufatura não mais como uma simples área de suporte, mas como integrante do grupo de fatores determinantes para a competitividade. Neste novo ambiente, a manufatura deve ser avaliada em suas dimensões estratégicas, como custo, qualidade, serviço e flexibilidade (Montevechi; Morandin; Miyagi, 2007).

A flexibilidade significa a capacidade de mudar a operação, ou seja, poder alterar o que a operação faz, como faz ou quando faz (Slack et al., 2002). Quando se tem certa flexibilidade no sistema de manufatura, como a possibilidade de fazer um produto por várias rotas diferentes ou de uma máquina fazer várias operações diferentes, podendo ser utilizada para fazer produtos diferentes, espera-se obter uma utilização eficiente dos recursos, em termos de certos critérios de desempenho.

Para tanto, é necessário ter uma programação da produção, em que esta programação é parte do planejamento e controle da produção (PCP), que é a atividade, em um sistema de manufatura, na qual se definem metas e estratégias, formula-se planos para atingi-las, administra recursos humanos e físicos com base nestes planos, direciona-se a ação dos recursos humanos sobre os físicos e acompanha esta ação, permitindo a correção de prováveis desvios (Tubino, 2000). Obter tal programação da produção não é uma tarefa simples, especialmente em ambientes com compartilhamento de recursos.

O problema de programação está na alocação de recursos no tempo e na seqüência correta, de forma que o resultado gerado seja a conclusão do conjunto de tarefas no menor tempo possível. Existem diversos tipos de padrões para o fluxo de produção, conhecido também como roteiro de fabricação, que é o percurso por onde o produto passa até o seu acabamento (Montevechi; Morandin; Miyagi, 2007).

Uma das dificuldades encontradas na programação da produção de um sistema de manufatura com recursos compartilhados é a programação de produtos com uso simultâneo

de máquinas e sistemas de transporte. Dentre os sistemas de transporte existentes, destacam-se os veículos auto-guiados (*Automated Guided Vehicle – AGV*). Os AGVs são veículos pequenos e autônomos, que movem materiais entre vários pontos do chão de fábrica (Slack et al., 2002). Tipicamente, os produtos em um sistema de manufatura com recursos compartilhados visitam diferentes máquinas para diferentes operações, necessitando assim, de um meio de transporte, neste caso o sistema de transporte, para transportar os produtos entre as máquinas do seu roteiro.

O problema da programação é classificado como um problema NP-Difícil, ou seja, existe uma grande concentração de esforço computacional, que cresce exponencialmente com o aumento do tamanho do problema (Chan et al., 2006). Assim, várias abordagens propostas para tratar do problema da programação da produção indicam uma grande eficiência na utilização dos Algoritmos Genéticos (Cavalieri, 1998; Pongcharoen; Hicks; Braiden, 2004; Jeong; Lim; Kim, 2006; Deriz, 2007; Morandin et al., 2008). Outras abordagens fazem uso dos algoritmos genéticos adaptativos a fim de evitar o problema da má escolha dos parâmetros e convergência prematura de um algoritmo genético tradicional (Yu; Cheng, 2004; Zhang; Wang; Zheng, 2006; Chan; Chung; Chan, 2005; sanches et al., 2008). Além disso, existem abordagens que integram, na programação dos produtos, o uso simultâneo de máquinas e sistemas de transporte usando algoritmos genéticos (Ulusoy; Serifoglu; Bilge, 1997; Haq; Karthikeyan; Dinesh, 2003; Jerald et al., 2006; Reddy; Rao, 2006; Sankar et al., 2004).

O grupo de pesquisa do TEAR, Laboratório de Pesquisa e Inovação em Tecnologias e Estratégias de Automação, investiga aplicações em indústrias de manufatura e de processos que buscam a incorporação de questões estratégicas de produção e estratégias competitivas nos projetos. Os temas macro abordados pelo grupo são planejamento reativo de produção, automação da planta industrial e integração entre estratégias, planejamento e chão de fábrica. Os problemas, as oportunidades e as necessidades das manufaturas são o ponto de partida das investigações do grupo, que tem como objetivo a busca por maior eficiência do sistema produtivo.

Uma das linhas de pesquisa na área de planejamento reativo da produção é a programação reativa da produção com técnicas de inteligência artificial. Nesta linha, destacam-se os trabalhos realizados por Maggio (2005) e Morandin et al. (2007a), que usam uma heurística de busca aplicada sobre um modelo de rede de Petri para gerar uma programação da produção. Ainda nesta linha de pesquisa, Deriz (2007), Morandin et al. (2007b) e Morandin et al. (2008) propõem uma forma de modelagem de algoritmo genético

para resolver o problema da programação da produção de sistemas de manufatura com recursos compartilhados, cujo critério para medir a eficiência da programação gerada é o *makespan* e o tempo de obtenção da resposta.

Sendo assim, procurou-se dar continuidade aos trabalhos do grupo na linha de Programação Reativa da Produção com Técnicas de Inteligência Artificial com a utilização de um Algoritmo Genético Adaptativo. Como a programação da produção dos sistemas de transporte não foi tratada por nenhum trabalho do grupo na linha da Programação Reativa da Produção, buscou-se abordar também o problema da programação dos sistemas de transporte, a fim de obter uma programação mais próxima da realidade.

## 1.2 Problema Abordado

A programação dos sistemas de transporte em sistemas de manufatura tem a mesma importância que a programação de máquinas e, por isso, ambos devem ser considerados na avaliação dos ciclos de tempos da produção.

Contudo, a maioria dos pesquisadores tem tratado o problema da programação da produção de produtos e sistemas de transporte como problemas independentes. Somente poucos pesquisadores tem enfatizado a importância da programação da produção de produtos com uso simultâneo de máquinas e os sistemas de transportes (Reddy; Rao, 2006).

## 1.3 Objetivos

### 1.3.1 Objetivo Geral

Em função do problema abordado, é proposto neste trabalho uma programação da produção de produtos com uso simultâneo de máquinas e sistemas de transporte em um sistema de manufatura com recursos compartilhados usando um algoritmo genético adaptativo.

### 1.3.2 Objetivos Específicos

- Propor um método de busca usando Algoritmo Genético Adaptativo (AGA)

para programação da produção em sistemas de manufatura com recursos compartilhados;

- Implementar este método para gerar Programação da Produção segundo o método proposto;
- Testar e avaliar o método proposto;
- Comparar os resultados do método proposto com os resultados de outras abordagens que tratam da programação da produção, sendo estas abordagens os trabalhos propostos por Morandin et al. (2008) e Reddy e Rao (2006).

## 1.4 Resultados Alcançados

Foi elaborado um método para gerar programação de produtos com uso simultâneo de máquinas e sistemas de transporte em sistemas de manufatura com recursos compartilhados, utilizando algoritmos genéticos adaptativos.

Para tanto, foi implementado um sistema a fim de validar a abordagem proposta. O sistema implementado possui algumas características como a configuração de um determinado cenário para a programação da produção.

Os resultados desta abordagem foram testados para uma série de problemas e comparados com os resultados das abordagens propostas por Morandin et al. (2008) e Reddy e Rao (2006). Tais comparações foram realizadas sobre os valores de *makespan* e tempo de execução.

## 1.5 Metodologia

Este trabalho foi desenvolvido baseado em aspectos de três métodos científicos: indutivo, hipotético-dedutivo e dedutivo conforme definido segundo Gil, Lakatos e Marconi (apud SILVA; MENEZES, 2001).

Neste trabalho, é utilizado o método indutivo, pois deriva das observações de casos reais de sistemas de manufatura.

O método hipotético-dedutivo também é utilizado, pois foi criada a hipótese de que uma busca utilizando algoritmos genéticos adaptativos, para a resolução do problema de programação de produtos com uso simultâneo de máquinas e sistemas de transporte para ambientes de manufatura com recursos compartilhados, pode gerar soluções mais eficientes

do que outros métodos de programação da produção. Portanto, pretende-se confirmar a hipótese, e não torná-la falsa, aspecto que remete ao método dedutivo.

Segue, em ordem cronológica, uma lista das atividades que foram desenvolvidas ao longo deste trabalho para obter os resultados finais:

➤ **Levantamento bibliográfico**

- Foi feito um levantamento bibliográfico sobre o que está sendo desenvolvido na área acadêmica e comercial para solução do problema de programação de produtos com uso simultâneo de máquinas e sistemas de transporte em sistemas de manufatura com recursos compartilhados.

➤ **Modelagem e a resolução do problema**

- Foi utilizado um algoritmo genético adaptativo a fim de modelar o problema e, posteriormente, aplicar uma busca sobre este modelo na tentativa de encontrar uma boa solução para o problema proposto.

➤ **Sistema implementado**

- O sistema foi implementado usando a linguagem de programação Java e o Ambiente de desenvolvimento Eclipse.
- Para testar a validade do código e garantir a qualidade do sistema desenvolvido, foi utilizado a ferramenta JUnit para testes unitários.

➤ **Testes**

- Os testes foram realizados em alguns cenários propostos, de tamanhos diferentes, no laboratório do TEAR, do Departamento de Computação da Universidade Federal de São Carlos.
- Os cenários ou problemas testados diferem entre si nas quantidades de máquinas da fábrica, tipos de produtos e roteiros de fabricação para cada produto. O número de veículos é fixo, em que foram utilizados dois veículos para todos os cenários testados.

➤ **Análise dos Resultados**

- A partir dos resultados obtidos, buscou-se observar se o trabalho proposto obteve um melhor *makespan* na maioria dos casos testados quando comparado com os resultados obtidos por Reddy e Rao (2006), já que esta

abordagem trata de um problema multi-objetivo. Já, quando comparado com os resultados obtidos por Morandin et al. (2008), buscou-se observar se os valores de *makespan* não ficaram muito distantes, pois esta abordagem não leva em consideração a programação da produção dos sistemas de transporte e em consequência disto, seus valores de *makespan* serão na maioria dos casos menores. Portanto, a partir desta análise, é possível obter um indicativo de que este trabalho possui um bom compromisso entre os valores de *makespan* obtidos.

- Como um dos objetivos deste trabalho é dar continuidade aos trabalhos do grupo de pesquisa TEAR, em especial sobre a linha de Programação Reativa da Produção, foi observado também nos resultados o tempo de obtenção da resposta, já que para uma programação reativa é necessário que o tempo de obtenção da resposta seja baixo. Este tempo foi comparado com ambas às abordagens propostas por Morandin et al. (2008) e Reddy e Rao (2006).
- Outro ponto analisado foi o desempenho da abordagem adaptativa para algoritmo genético, visto que, definir alguns parâmetros como a taxa de cruzamento e mutação não é uma tarefa simples e uma má escolha desses parâmetros pode prejudicar o algoritmo, de modo que o mesmo não explore eficientemente o espaço de busca. Com a utilização do algoritmo genético adaptativo, as taxas de cruzamento e mutação são ajustadas dinamicamente, conforme o desempenho do algoritmo. Assim, buscou-se com o uso do algoritmo genético adaptativo, alcançar outros pontos do espaço de busca e evitar também um problema conhecido como convergência prematura, que é uma estagnação prematura da busca causada pela perda da diversidade genética.

## 1.6 Estrutura do Trabalho

Com base na metodologia empregada, este trabalho está dividido da seguinte maneira:

Para modelar e, posteriormente, aplicar um processo de busca a fim de encontrar uma solução ideal para o problema proposto neste trabalho, foi utilizado um

Algoritmo Genético Adaptativo. Alguns conceitos e funcionalidades sobre os Algoritmos Genéticos e sua abordagem adaptativa são apresentados no Capítulo 2.

A revisão de trabalhos que utilizam algoritmos genéticos para resolver problemas na programação da produção e que direcionaram para a definição das estratégias a serem usadas para resolver o problema em questão, encontram-se no Capítulo 3, que está dividido em três partes. A primeira parte retrata como alguns pesquisadores têm abordado a resolução do problema da programação da produção usando os Algoritmos Genéticos tradicionais. A segunda parte apresenta a utilização dos Algoritmos Genéticos Adaptativos para resolução do problema da programação da produção. A terceira parte aborda a integração da programação de máquinas com a programação dos sistemas de transporte.

Para resolver o problema de programação da produção proposto foi necessário definir um tipo de modelagem usando um algoritmo genético adaptativo e desenvolver um sistema a fim de obter os resultados esperados para problema abordado, respectivamente.

A definição da modelagem do método proposto e as restrições do ambiente de manufatura para a solução deste tipo de problema são apresentados no Capítulo 4. Ainda neste capítulo, são abordados detalhes sobre o algoritmo genético adaptativo utilizado.

As características, funcionalidades e principais interfaces do sistema implementado são mostradas no Capítulo 5. Ainda neste capítulo, é apresentada a técnica utilizada para validar o código desenvolvido.

Para o entendimento de como o sistema foi testado e como os resultados obtidos foram comparados e utilizados na conclusão deste trabalho, o ambiente do Sistema de Manufatura utilizado para os testes é apresentado no Capítulo 6, no qual também são descritas as situações testadas, os critérios usados para comparação e os métodos que foram comparados.

No capítulo 7 são apresentadas as conclusões e as propostas para trabalhos futuros.

## 2 ALGORITMOS GENÉTICOS

### 2.1 Considerações Iniciais

As pesquisas com Computação Evolutiva tiveram seu início na década de 1950. Trata de sistemas para a resolução de problemas e são inspirados na teoria da evolução natural. A Computação Evolutiva por ser dividida em três importantes grupos: Algoritmos Genéticos, Estratégias de Evolução e Programação Genética.

Estes modelos mantêm uma população de indivíduos que representam possíveis soluções para o problema e cada solução é avaliada por uma função que determina quão boa ela é para o problema. Uma nova população será formada com os melhores indivíduos, dentre os quais, alguns serão modificados a fim de gerar novas soluções para o problema. Estes modelos possuem características em comum, mas cada modelo possui diferenças significantes em seu modo de operação (Srinivas; Patnaik, 1994b).

Neste contexto, são apresentados neste capítulo os conceitos básicos do modelo computacional mais conhecido e utilizado da Computação Evolutiva, os Algoritmos Genéticos.

### 2.2 Algoritmos Genéticos

Os Algoritmos Genéticos (AG) são algoritmos de busca baseado nos mecanismos da seleção natural e genética natural (Goldberg, 1989). Segundo Mitchell (1997) os AGs tem sido aplicados com sucesso para uma variedade de tarefas de aprendizado e outros problemas de otimização.

Os AGs foram criados na década de 60 por John Holland e outros pesquisadores da Universidade de Michigan. Com base nas pesquisas realizadas sobre os AGs, John Holland publicou um livro sobre este assunto, chamado de “Adaptation in Natural and Artificial Systems” (Holland, 1975). Após isso, os AGs tem sido analisado por diversos pesquisadores e aplicado para vários tipos de problemas nas mais diversas áreas.

Algoritmos Genéticos partem do pressuposto que, em uma dada população, indivíduos com boas características genéticas têm maiores chances de sobrevivência e de

produzirem indivíduos cada vez mais aptos. Como resultado, os indivíduos menos aptos tenderão a desaparecer. Assim, Algoritmos Genéticos favorecem a combinação dos indivíduos mais aptos (Carvalho; Braga; Ludermir, 2003).

Quando Algoritmos Genéticos são empregados para resolução de problemas do mundo real, cada indivíduo da população, denominado cromossomo, normalmente corresponde a uma possível solução para o problema. Um mecanismo de reprodução, baseado em processo evolutivo, é aplicado sobre a população atual com o objetivo de explorar os espaço de busca e encontrar melhores soluções para o problema.

As técnicas de busca e otimização de problemas segundo Lacerda e Carvalho (1999) geralmente apresentam:

- Um *espaço de busca*, onde se encontram todas as possíveis soluções para o problema;
- Uma *função objetivo* (chamada de *função de aptidão* ou *função de fitness*, quando relacionado aos Algoritmos Genéticos), que é utilizada para avaliar as soluções produzidas, associando a cada uma delas um valor numérico.

As técnicas de busca e otimização tradicionais iniciam seu processamento com um único candidato que, iterativamente, é manipulado utilizando algumas heurísticas diretamente associadas ao problema a ser solucionado. Apesar de esses métodos não serem suficientemente robustos, isto não implica que eles sejam inúteis (Carvalho; Braga; Ludermir, 2003).

Por outro lado, as técnicas de Computação Evolutiva operam sobre uma população de candidatos em paralelo. Assim, elas podem fazer a busca em diferentes áreas do espaço de solução, alocando um número de membros apropriado para a busca em várias regiões. Como resultado, tais técnicas têm maior chance de atingir as áreas mais promissoras do espaço de busca. Os Algoritmos Genéticos diferem dos métodos tradicionais de busca e otimização em quatro aspectos (Goldberg, 1989):

1. trabalham com uma codificação do conjunto de parâmetros e não com os próprios parâmetros;
2. trabalham com uma população e não com um único ponto;
3. utilizam informações de custo ou recompensa e não derivadas ou outro conhecimento auxiliar;

4. utilizam regras de transição probabilísticas e não determinísticos.

## 2.3 Funcionamento

O algoritmo opera através de evoluções, onde em cada evolução a população é atualizada. Esta população é formada por um conjunto aleatório de indivíduos que podem ser vistos como possíveis soluções do problema (Mitchell, 1997).

Durante o processo evolutivo, a população é avaliada: para cada indivíduo, chamado de cromossomo, é dado uma nota, ou índice, refletindo sua habilidade de adaptação a determinado ambiente. Uma quantidade dos cromossomos mais adaptados é mantida, enquanto os outros são descartados. Os cromossomos mantidos pela seleção podem sofrer modificações em suas características fundamentais por meio de mutações e cruzamento (crossover) ou recombinação genética gerando descendentes para a próxima geração. Esse processo, chamado de reprodução, é repetido até que uma solução satisfatória seja encontrada. A seguir é apresentado o fluxo básico dos AGs:

```
1: T = 0;
2: Gerar População Inicial P(0);
3: para todo cada indivíduo i da população atual P(t) faça
4:   Avaliar aptidão do indivíduo i;
5: fim para
6: enquanto Critério de parada não for satisfeito faça
7:   t = t + 1;
8:   Selecionar população P(t) a partir de P(t - 1);
9:   Aplicar operadores de cruzamento sobre P(t);
10:  Aplicar operadores de mutação sobre P(t);
11:  Avaliar P(t);
12: fim enquanto
```

Algoritmo 1 – Algoritmo Genético típico (Carvalho; Braga; Ludermir, 2003).

Cada uma das fases desse ciclo e quais os critérios de parada utilizados para a saída do laço de repetição serão explicados posteriormente.

## 2.4 Representação dos Parâmetros

O Algoritmo Genético processa populações de indivíduos ou cromossomos. O cromossomo é uma estrutura de dados, geralmente vetores ou cadeias de valores binários, que

representa uma possível solução do problema a ser otimizado. Em geral, o cromossomo representa o conjunto de parâmetros da função-objetivo cuja resposta será maximizada ou minimizada. O conjunto de todas as configurações que o cromossomo pode assumir forma o seu *espaço de busca*. Se o cromossomo representa  $n$  parâmetros de uma função, então o espaço de busca é um espaço com  $n$  dimensões. Nos termos biológicos, as características de um indivíduo são definidas como fenótipo, e as codificações genéticas dessas características são definidas como genótipos.

Neste contexto, através dos Algoritmos Genéticos, geralmente o genótipo de um indivíduo é representado por um vetor binário, onde cada elemento de um vetor denota a presença (1) ou ausência (0) de uma determinada característica. Os elementos podem ser combinados formando as características reais do indivíduo, neste caso definido como o seu fenótipo.

A representação do cromossomo pode ser feita através da utilização dos números reais tornando mais fácil à compreensão pelo ser humano e requer menos memória representação através de cadeias de bits.

A utilização de representações em níveis de abstração mais altos tem sido investigada e por serem mais fenotípicas, facilitariam seu uso em determinados ambientes, onde essa transformação “fenótipo – genótipo” é muito complexa. Nesse caso, precisam ser criados operadores específicos para utilizar essas representações (Carvalho; Braga; Ludermir, 2003).

A seguir na Figura 1, são apresentados um conjunto de cromossomos com codificação inteira. Cada posição no cromossomo é definida como gene e o conjunto de cromossomos candidatos à solução é definido como população.

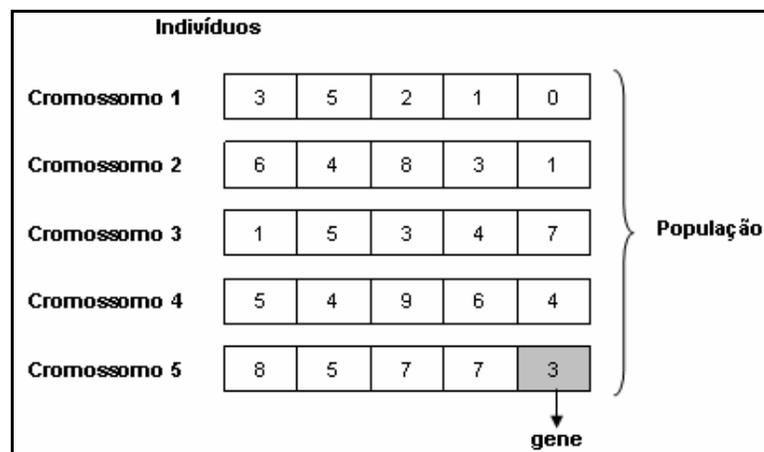


Figura 1 – população de cromossomos

## 2.5 Seleção

O Algoritmo Genético começa com uma população inicial de  $N$  cromossomos. Quando não existe nenhum conhecimento prévio sobre a região do espaço de busca onde se encontra a solução do problema, os cromossomos são gerados aleatoriamente.

Para que o processo de seleção possa dar prioridade aos indivíduos mais aptos, a cada cromossomo da população é atribuído um valor através de uma função  $f_{\text{apt}}$  denominado função de aptidão. Esta função recebe como entrada os valores do gene do cromossomo e fornece como resultado a sua aptidão. A aptidão pode ser vista como uma nota que mede a qualidade da solução codificada por um indivíduo e é baseada no valor da função-objetivo, que é específica para cada problema.

Uma função de aptidão é geralmente uma expressão matemática que mede o quanto uma solução está próxima ou distante da solução desejada. Há problemas de otimização que procuram maximizar o valor da função de aptidão, enquanto outros procuram a minimização.

Após ter associado uma nota ou aptidão a cada indivíduo da população, o processo de seleção escolhe um subconjunto de indivíduos da população atual, gerando uma população intermediária. A seguir três métodos de seleção serão apresentados.

### 2.5.1 Método da Roleta

Neste método, os indivíduos de uma geração (ou população) são selecionados para a próxima geração utilizando uma roleta. Cada indivíduo da população é representado na roleta por uma fatia proporcional ao seu índice de aptidão. Assim, indivíduos com aptidão ocupam fatias maiores da roleta, enquanto indivíduos de aptidão mais ocupam fatias menores.

Na Figura 2 é apresentada a criação de uma roleta a partir dos valores de aptidão dos indivíduos de uma população. Para a seleção dos indivíduos, a roleta é girada  $N$  vezes, onde  $N$  é número da população inicial. A cada vez que a roleta parar de girar, o cromossomo selecionado pelo marcador será copiado para a próxima geração. Cromossomos com maior espaço na roleta terão maior chance de serem selecionados. O algoritmo Roda da Roleta não funciona com aptidões negativas. Além disso, pode gerar também problemas como convergência prematura (Carvalho; Braga; Ludermitz, 2003).

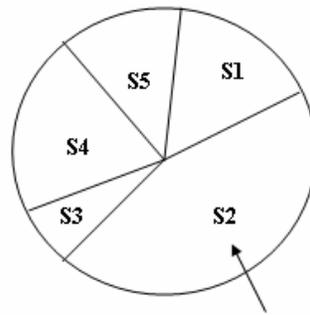


Figura 2 – Método de seleção por roleta

### 2.5.2 Método do Torneio

Quando este método é utilizado, um número de indivíduos  $n$  da população é escolhido de modo aleatório e com mesma probabilidade. O cromossomo com maior aptidão dentre estes  $n$  cromossomos é selecionado para a população intermediária. Este processo se repetirá até que a população intermediária seja preenchida. Na Figura 3 é ilustrado o método de seleção por torneio, em que a partir de uma população composta pelos cromossomos S1, S2, S3, S4 e S5, este método seleciona os cromossomos com maior valor de aptidão.

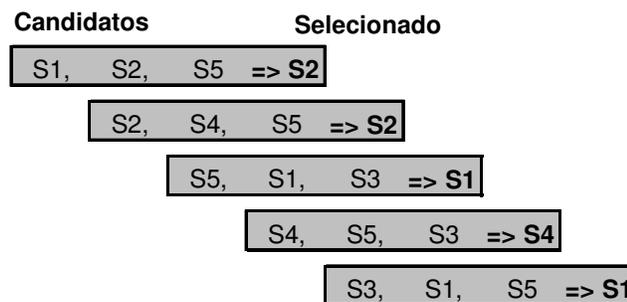


Figura 3 – Método de seleção por torneio

### 2.5.3 Método da Amostragem Universal Estocástica

É semelhante ao método da Roleta, mas com um número de marcadores iguais ao número de indivíduos da população inicial, igualmente espaçados na roleta. Dessa forma, a roleta gira apenas uma vez e cada cromossomo apontado por um marcador é copiado para a próxima geração. Neste método, cromossomos que ocupam pequeno espaço na roleta têm

mais chance de serem selecionados, garantindo a diversidade genética da população. A Figura 4 ilustra este método.

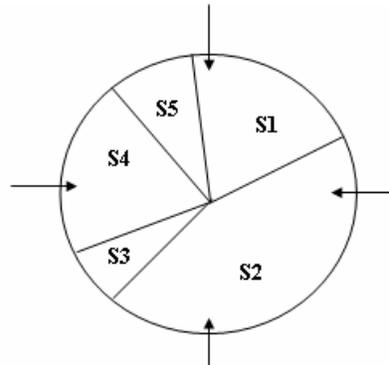


Figura 4 – Método de seleção por Amostragem Universal Estocástica

## 2.6 Operadores Genéticos

Um conjunto de operadores é necessário para que, dada uma população, seja possível gerar populações sucessivas que melhorem sua aptidão com o tempo. Estes operadores são o de cruzamento (crossover) e mutação. Eles são utilizados para assegurar que a nova geração seja totalmente nova, mas possua características de seus pais, sendo necessários para que a população se diversifique e mantenha características de adaptação adquiridas pelas gerações anteriores. Para prevenir que os melhores indivíduos não desapareçam da população pela manipulação dos operadores genéticos, é interessante transferir o melhor cromossomo de uma geração para a outra sem alterações. Esta estratégia é denominada *Elitismo* (Lacerda; Carvalho, 1999).

O cruzamento é o operador responsável pela recombinação de características dos pais durante a reprodução, permitindo que as próximas gerações herdem essas características. Ele é considerado o operador genético predominante, por isso é aplicado com probabilidade dada pela taxa de cruzamento ( $0 \leq P_c \leq 1$ ), que deve ser maior que a taxa de mutação, sendo usualmente  $0,6 \leq P_c \leq 0,99$  (Carvalho; Braga; Ludermir, 2003).

Existem vários operadores de cruzamento, dos quais alguns serão especificados a seguir:

- **Cruzamento de Um-ponto:** um ponto é escolhido aleatoriamente, no qual os dois cromossomos são cortados. O segundo segmento dos dois cromossomos pais são trocados. A partir desta troca, são gerados dois

novos cromossomos definidos como cromossomo filhos. Na Figura 5 é apresentado o funcionamento deste método de cruzamento.



Figura 5 – Cruzamento de um ponto

- **Cruzamento Multipontos:** Através deste operador, são escolhidos  $n$  pontos de cruzamento, dividindo os cromossomos pais em  $n+1$  partes, que podem ser trocadas entre eles para gerar os filhos. O funcionamento deste operador pode ser visto na Figura 6.

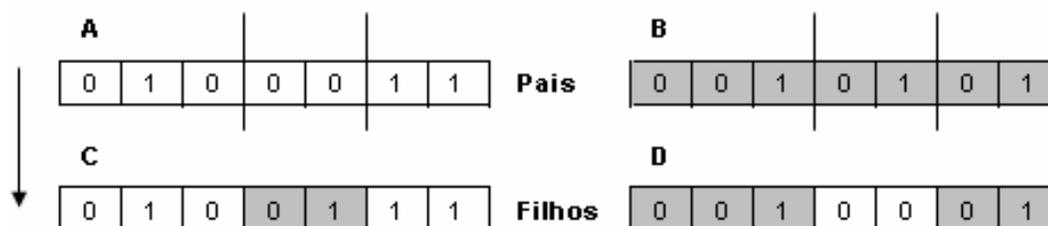


Figura 6 – Cruzamento de dois pontos

- **Cruzamento Uniforme:** Não utiliza pontos de cruzamento, mas determina, através da utilização de uma máscara, quais os genes de cada cromossomo que cada filho herdará. Um valor 1 na máscara indica que o gene correspondente do pai A será herdado pelo filho C, e o gene correspondente do pai B será herdado pelo filho D. Para um valor igual a 0 na máscara, ocorre o inverso. Através da Figura 7 é possível ver um exemplo da troca de informações provocada por este operador.

O operador de mutação é necessário para a introdução e manutenção da diversidade genética da população, alterando arbitrariamente um ou mais componentes de uma estrutura escolhida, o que fornece meios para a introdução de novos elementos na população. Como no cruzamento, existe uma taxa de mutação, que, se for alta demais, pode

tornar o algoritmo desprovido de direção no espaço de busca e se for baixa demais pode tornar o processo de busca lento. Neste sentido, o operador de mutação é aplicado aos indivíduos com uma probabilidade dada pela taxa de mutação ( $0 \leq P_m \leq 1$ ), geralmente se utiliza uma de mutação pequena ( $0,001 \leq P_m \leq 0,1$ ), pois é um operador genético secundário (Carvalho; Braga; Ludermir, 2003). Na Figura 8 é apresentado o funcionamento do operador de mutação.

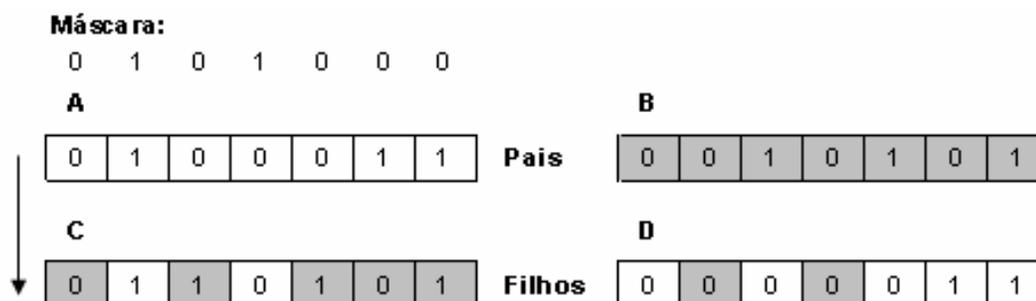


Figura 7 – Cruzamento Uniforme

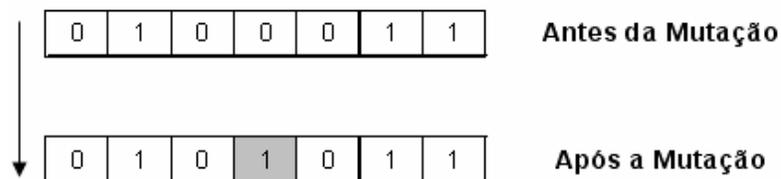


Figura 8 – Operador de Mutação

## 2.7 Operadores Genéticos para Permutações

Existem problemas que dependem da ordem da ordem com que as ações são executadas. Tais problemas têm sido exhaustivamente estudados na literatura de AGs e têm levado a proposta de vários operadores genéticos específicos (Lacerda; Carvalho, 1999).

Uma permutação  $m$  elementos é uma seqüência de  $m$  elementos em que nenhum elemento é repetido. Por exemplo, (A,B,C) e (C,A,B) são exemplos de duas permutações dos elementos A,B e C.

Os operadores de mutação são relativamente simples. Na mutação baseada na posição, dois elementos do cromossomo são escolhidos aleatoriamente e o segundo elemento é colocado antes do primeiro. Na mutação baseada na ordem dois elementos do cromossomo são escolhidos aleatoriamente e suas posições são trocadas. A mutação por embaralhamento

começa escolhendo aleatoriamente dois cortes no cromossomo. Depois os elementos na sub-lista entre os cortes são embaralhados como apresentado a seguir.

**Cromossomo:**      A   B | C   D   F | E   G

**Após a mutação:**    A   B | F   C   D | E   G

Existem vários operadores de cruzamento para permutação. Alguns deles como: OBX (Order-Based Crossover), PBX (Position Based Crossover), PMX (Partially Matched Crossover), CX (Cycle Crossover) e OX (Order Crossover) são discutidos com mais detalhe em Lacerda e Carvalho (1999).

## 2.8 Controle dos Parâmetros Genéticos

O desempenho de um Algoritmo Genético é fortemente influenciado pela definição dos parâmetros a serem utilizados. Segundo Srinivas e Patnaik (1994b) a escolha dos parâmetros pode tornar-se um problema complexo de otimização e, além disso, a escolha destes parâmetros esta relacionada com a natureza da função objetivo. Portanto, é importante analisar como os parâmetros podem ser utilizados diante das necessidades do problema e dos recursos disponíveis. A seguir serão apresentados alguns critérios para escolha dos principais parâmetros:

- **Tamanho da População:** O tamanho da população afeta o desempenho global e a eficiência dos Algoritmos Genéticos. Em uma população pequena, o Algoritmo Genético é mais rápido, mas seu desempenho pode cair, pois fornece uma pequena cobertura do espaço de busca.
- **Taxa de Cruzamento:** Quanto maior a taxa, mais rapidamente novas estruturas serão introduzidas na população, mas pode perder indivíduos mais aptos. Caso esta taxa seja muito pequena, a busca pode estagnar.
- **Taxa de Mutação:** Uma baixa taxa de mutação previne que a busca fique estagnada em sub-regiões do espaço de busca. Além disso, possibilita que qualquer ponto do espaço de busca seja atingido. A taxa muito alta torna a busca aleatória.

- **Critério de Parada:** Diferentes critérios podem ser utilizados para terminar a execução de um Algoritmo Genético, por exemplo, após um dado número de gerações, quando a aptidão média ou de melhor indivíduo parar de melhorar ou quando as aptidões dos indivíduos de uma população se tornarem muito parecidas.

## 2.9 Algoritmos Genéticos Adaptativos

Como visto anteriormente, a escolha de valores inadequados para os parâmetros genéticos pode causar uma série de problemas como a *convergência prematura*, causada pela perda de diversidade na população.

Segundo Zhou e Xin (2006), a maioria dos AGs simples da literatura utilizam parâmetros inalteráveis e conseqüentemente convergem prematuramente ou ficam estagnados em um ponto do espaço de busca e além disso, estes AGs simples podem não alcançar um ponto do espaço de busca que possui soluções boas para um determinado problema.

Um grande número de experimentos para descobrir valores ótimos para tais parâmetros são reportados em Grefenstette (1986), onde os valores ideais encontrados para estes parâmetros permanecem fixos durante todo o processo de busca e estão dentro de um intervalo relativamente grande que, dependendo do valor escolhido, podem levar a uma significativa diferença no desempenho do algoritmo. Além disso, alguns valores podem ser eficientes para um problema específico, mas não para outro.

Através destas observações, percebe-se a necessidade da variação dinâmica de parâmetros analisando o comportamento interno de um algoritmo genético. Nos trabalhos encontrados na literatura sobre ajuste dinâmico de valores de parâmetros, o motivo para tal ajuste é quase o mesmo na maioria dos trabalhos encontrados, onde o objetivo é evitar a convergência prematura, melhorando a forma com que o algoritmo explora o espaço de busca. Neste contexto, destacam-se os trabalhos de Srinivas e Patnaik (1994a) e Jerald et al. (2006), onde Srinivas e Patnaik (1994a) propõem uma alteração dinâmica das taxas de cruzamento e mutação a fim de alcançar duas características. A primeira característica é a capacidade de convergir para uma solução ótima após alcançar uma região contendo boas soluções dentro do espaço de busca. A segunda característica é a capacidade de explorar novas regiões do espaço de soluções na busca. Já Jerald et al. (2006), propõem uma alteração dinâmica das taxas de cruzamento e mutação a fim aumentar a ocorrência do operador quando ele produz melhores

cromossomos filhos. Caso o operador sempre produza cromossomos filhos piores que os cromossomos pais, a probabilidade de ocorrência deste operador será reduzida.

## **2.10 Considerações Finais**

Neste capítulo, foram apresentados os conceitos e fundamentos básicos sobre os Algoritmos Genéticos, que serão suficientes para entender seu funcionamento no contexto deste trabalho.

No próximo capítulo será realizado um levantamento de trabalhos propostos para a resolução do problema de programação da produção, em especial para a resolução do problema da programação de produtos com uso simultâneo de máquinas e sistemas de transporte usando algoritmos genéticos.

## **3 USO DE ALGORITMOS GENÉTICOS PARA A PROGRAMAÇÃO DA PRODUÇÃO**

### **3.1 Considerações Iniciais**

Atualmente, os algoritmos genéticos são aplicados para muitos tipos de problemas de otimização, em especial para problemas de otimização relacionados com a área de manufatura.

Neste contexto, será apresentado o uso dos algoritmos genéticos para vários tipos de problemas na área da programação da produção. Este capítulo está dividido em 3 partes: a primeira parte fala sobre como alguns pesquisadores tem abordado a resolução do problema da programação da produção usando os algoritmos genéticos tradicionais, já a segunda parte é caracterizada pela utilização dos algoritmos genéticos adaptativos para tal problema, e por fim na terceira parte, é abordado a integração da programação da produção de máquinas com a programação da produção dos sistemas de transporte.

### **3.2 Uso de Algoritmos Genéticos Tradicionais para a Programação da Produção**

Cavalieri (1998) propõe uma estratégia de otimização de desempenho que leva em conta a seqüência de programação para atividades desempenhadas por cada máquina do sistema e o *work in process* (WIP) em cada ciclo de produção, isto é, o número de peças em cada ciclo de produção. Para tanto, utiliza modelagem por meio de Redes de Petri (*Petri Nets* – PN), com tempo de operação associado às transições. Os ciclos de produção são modelados, ligando a transição que corresponde ao último processo de cada ciclo de produção à transição que corresponde ao primeiro processo do mesmo ciclo de produção por meio de um lugar. Nesta abordagem, a seqüência de lugares e transições que representam atividades desempenhadas por uma máquina forma o circuito de comando daquela máquina. O número de marcas presente em cada circuito de comando é igual a 1, pois uma máquina só processa um trabalho por vez. Desta forma, o *throughput* (número de peças processadas por unidade de tempo) do sistema é limitado pela máquina que tem maior tempo de processamento. Para

tanto, Cavalieri (1998) utiliza um algoritmo genético (AG) para explorar o espaço de seqüência de programação para as atividades desempenhadas por cada máquina e utiliza um algoritmo heurístico de ajustamento (AHA), que determina o mínimo WIP em cada ciclo de produção para cada cenário de produção e retorna este valor como valor de *fitness*.

Os cromossomos são codificados por um vetor de tamanho  $n \times m$  de números inteiros, agrupados em  $m$  subvetores, onde  $m$  é o número de máquinas do sistema (número de circuitos de comando) e  $n$  é o número máximo de atividades processadas por cada máquina. Cada subvetor diz respeito a uma máquina específica e cada inteiro está associado a transições contidas em cada circuito de comando. A operação de cruzamento é feita selecionando dois cromossomos pais e, para um subvetor correspondente em ambos, forma-se o subvetor dos filhos, alternando entre elementos de ambos os pais. A operação de mutação é feita pela troca de dois elementos, aleatoriamente escolhidos em cada filho. A busca pára quando a população converge.

O autor apresenta um exemplo de aplicação do algoritmo no qual, iniciando com um grafo, o AG encontra uma solução ótima que apresenta menor valor de WIP, mas não apresenta resultados de desempenho em comparação com outros métodos.

Pongcharoen et al. (2002) propõem um experimento deste AG aplicado a uma manufatura de bens de capital, para identificar valores dos parâmetros do AG, que produzem os melhores resultados com um número total de cromossomos gerados fixo. O objetivo é identificar os valores dos parâmetros do AG mais eficientes para resolver um problema grande e computacionalmente custoso, com tempo de execução específico. Foram testadas três combinações de valores de tamanho da população e número de gerações, respectivamente, sendo eles: 20 e 60; 40 e 30; 60 e 20. Os melhores resultados foram obtidos com tamanho da população igual a 60 cromossomos e 20 gerações. O processo de reparo dos cromossomos teve impacto crítico na geração de programações factíveis.

Os resultados das programações geradas pelo AG e passadas pelo processo de reparo foram comparados com programações geradas aleatoriamente e passadas pelo processo de reparo. As programações geradas pelo AG foram melhores que as geradas aleatoriamente.

O AG foi aplicado a um problema grande e custoso computacionalmente, com população de 60 cromossomos, 20 gerações, taxa de mutação de 18% e probabilidade de cruzamento de 0,6 e 0,9 e os resultados foram comparados com a programação da companhia. A programação da companhia resultou em atraso na entrega e custos de multa por atraso. A programação resultante do AG gerou entrega pontualmente, mas gerou multa de menor valor, por adiantamento, relacionada a custos de armazenamento. No problema testado, o melhor

resultado foi obtido com probabilidade de cruzamento de 0.6, não gerando atraso na entrega e atingindo uma redução de custos de 63% em relação à programação feita pela companhia. A probabilidade de cruzamento, embora tenha sido estatisticamente insignificante anteriormente, teve impacto neste caso particular, o que indica o estudo, para trabalhos futuros, da relação entre parâmetros do AG e o tamanho do problema.

Pongcharoen, Hicks e Braiden (2004) propõem um algoritmo genético modificado em relação ao funcionamento geral dos AGs, para programação da produção de produtos complexos com estrutura de produto de múltiplos níveis, considerando restrições de recursos e relações de precedência entre seqüências de operações e montagens.

Neste algoritmo, após a aplicação dos operadores genéticos, há um processo de reparo nos cromossomos, que corrige programações não factíveis que possam ser geradas pela aplicação do cruzamento e mutação. Os cromossomos são codificados como cadeias alfanuméricas, onde cada gene tem três partes: um identificador da estrutura do produto, um identificador da instância do produto e o número da operação. Os genes são aleatoriamente seqüenciados para formar cada cromossomo da população. Cada cromossomo é subdividido em  $n$  subcromossomos, que representam seqüências de operações para  $n$  recursos.

As operações genéticas são aplicadas em cada subcromossomo para gerar seqüência de operações para cada máquina. Nos cromossomos selecionados aleatoriamente, é aplicado cruzamento de um ponto, que consiste em selecionar de maneira aleatória um ponto que divide os cromossomos pais em duas partes. A primeira parte do primeiro pai é diretamente copiada para o primeiro filho. Os genes remanescentes são obtidos do segundo pai. O processo é repetido em ordem reversa para produzir o segundo filho. Mutação inversa é aplicada dentro de cada subcromossomo, respeitada a probabilidade, que consiste em selecionar aleatoriamente dois pontos no subcromossomo e realocar os genes localizados entre esses dois pontos em ordem reversa.

Após as operações genéticas, é aplicado um processo de reparo, pois os cromossomos gerados podem representar programações não factíveis. O processo de reparo tem quatro estágios: ajustamento da operação precedente, ajustamento da precedência das peças, temporização da tarefa e considerações de capacidade finita e, por último, ajustamento de *deadlock*.

A função de *fitness* considera os custos de multas, tanto por adiantamento, que tem custo de armazenamento (de matéria prima, de produtos em processo e de produtos acabados) associado, como por atraso nas entregas (somente de produtos acabados), que gera multas. Em uma programação ideal, estes custos, ou seja, o valor de *fitness* deve ser igual a

zero. A função de *fitness* considera o custo total como sendo a soma do custo de adiantamento para todos os componentes, do custo de adiantamento para todos os produtos e do custo de atraso para todos os produtos.

A seleção dos indivíduos para a próxima geração é feita pelo método da roleta, onde a probabilidade de sobrevivência e o número de réplicas de um indivíduo na próxima geração é proporcional ao seu valor de *fitness*. Em experimentos aplicados a três problemas com tamanhos diferentes com relação à quantidade de operações de usinagem e montagem e à quantidade de recursos, verificou-se que o número de cromossomos da população e o número de gerações determinam o tempo de execução, além disso, juntamente com as probabilidades de cruzamento e de mutação, influenciam na convergência e na qualidade das soluções encontradas.

O fator principal que influencia nos custos de multas é o tamanho do problema e a qualidade da solução é inversamente proporcional ao tamanho do problema. Obtiveram-se melhores resultados uma população composta de 40 cromossomos e um número de gerações igual a 40. Outro fator importante foi à probabilidade de mutação, os melhores resultados foram obtidos com valores entre 0,06 e 0,1. A probabilidade de cruzamento não foi estatisticamente significativa para os testes realizados, com valores entre 0,6 e 0,9. O processo de reparo é um fator crítico para corrigir programações não factíveis. Todos os cromossomos em cada geração são corrigidos.

As programações geradas pelo AG mostraram custos de multas muito mais baixos do que os métodos tradicionais usados nas companhias, mostrando mais de 80% de redução nos custos.

Jeong, Lim e Kim (2006) propõem um método híbrido, combinando AG e simulação para resolver o problema da programação da produção, sendo o AG configurado para decidir programações ótimas para minimizar o *makespan* e o modelo de simulação utilizado para encontrar o tempo de execução das programações fixadas pelo AG.

O ambiente é composto de  $m$  máquinas e  $n$  trabalhos, com rotinas de processamento específicas (operações) a serem seguidas. Cada operação de cada trabalho tem uma precedência e leva um tempo determinístico em uma determinada máquina. O tempo de início de operação de cada trabalho está sujeito ao tempo disponível e à data devida do trabalho.

Uma solução factível é uma programação que satisfaz às restrições de precedência de operações em um mesmo trabalho e a restrição de que uma máquina só pode processar uma operação de cada vez, ou seja, processar um trabalho de cada vez, dentre os

trabalhos que estão na fila de espera. Para um problema com  $n$  trabalhos e  $m$  máquinas, um cromossomo é do tamanho  $n \times m$ .

A representação do cromossomo é baseada em operações, sendo uma programação codificada como uma seqüência de operações, uma seqüência de números naturais, de 1 até  $n \times m$ , onde cada gene representa uma operação. A operação de cruzamento é feita selecionando uma cadeia de genes aleatoriamente de um cromossomo pai, encontrando sua posição no outro pai e copiando os itens restantes para o segundo cromossomo pai, na ordem em que eles aparecem no primeiro cromossomo pai. A mutação é feita selecionando duas posições em um cromossomo e trocando seus conteúdos. A seleção de indivíduos para cruzamento usa um mecanismo baseado em categoria, no qual a probabilidade de seleção não está diretamente ligada ao valor de *fitness* do indivíduo, mas é uniformemente curva, evitando que bons indivíduos dominem a evolução precocemente. A função objetivo é o mínimo *makespan*.

O *makespan* de cada cromossomo é produzido pelo processo que atribui operações para as máquinas, pela seqüência de cada trabalho, examinando o cromossomo da esquerda para a direita. O *makespan* objetivo é selecionado por comparação dos desempenhos das programações.

Para simular o comportamento dos sistemas de manufatura em caso real, neste modelo de simulação, são incorporadas características dinâmicas do sistema, como quebra e reparo de máquinas, enfileiramento e espera de produtos. Neste caso, são distribuídos tempos médios de quebra e reparos a máquinas aleatoriamente. O tempo de conclusão é o tempo de processamento no modelo de simulação, ou seja, o tempo total de simulação gasto no sistema para processar todas as operações, baseado na programação da produção do AG. Em seguida, o tempo de operação do AG é ajustado pelos resultados da simulação e o modelo AG gera novas programações pelo tempo de operação ajustado. Este processo segue até que a taxa de diferença entre o tempo da simulação precedente e o tempo da simulação corrente seja pequeno o suficiente para ser aceitável.

Nas abordagens propostas por Deriz (2007) e Morandin et al. (2008), é apresentada uma forma de modelagem de algoritmo genético para resolver o problema da programação da produção de sistemas de manufatura com recursos compartilhados. Os problemas de programação de um sistema de manufatura com compartilhamento de recursos envolvem decisões na alocação dos recursos de produção ao longo do tempo, bem como a escolha dos roteiros de fabricação para cada lote de peças na obtenção de um tipo de produto, definindo o momento de execução de cada etapa.

Em relação à modelagem do sistema, foram dotadas algumas restrições neste trabalho, sendo elas: as máquinas nunca falham e seus tempos de operação são determinísticos e previamente conhecidos, tempos de transporte são considerados como parte dos tempos de operação, existem áreas de armazenamento intermediárias com capacidade infinita entre as máquinas e os tempos de setup de máquinas estão incluídos nos tempos de operação.

Neste trabalho, um cromossomo é codificado de maneira a indicar uma programação da produção, ou seja, uma solução completa para o problema. A solução encontrada é 'traduzida' para uma programação da produção através da leitura dos genes do cromossomo. Um cromossomo é representado por um conjunto de valores inteiros de tamanho  $n+n*t$ , dividido em  $n$  subconjuntos de genes de tamanho  $t+1$ , onde  $n$  é o número de produtos a serem produzidos e  $t$  o número máximo de operações em todos os possíveis roteiros de fabricação para os  $n$  produtos. Cada subconjunto representa um produto a ser processado, da seguinte forma: o primeiro gene do subconjunto é o índice que identifica o produto, e os  $t$  genes seguintes representam as máquinas do roteiro de fabricação escolhido para aquele produto.

O cruzamento é feito entre dois cromossomos pais  $C1$  e  $C2$ , trocando o subconjunto referente ao mesmo produto  $Pip$ , em cada um deles, gerando dois cromossomos filhos  $C3$  e  $C4$ , que irão para a nova população. A mutação é realizada em um cromossomo, obtido aleatoriamente da população, trocando o roteiro de fabricação de um produto  $Pip$ , também aleatoriamente escolhido, por um dos possíveis roteiros para  $Pip$ .

A função de fitness usada para avaliar as soluções para programação da produção considera o mínimo *makespan*, ou seja, cromossomos com menor valor de *makespan* são considerados mais aptos do que os cromossomos com maior valor de *makespan*. Foram realizados testes computacionais com 3 tipos de problemas, 3 produtos e 6 máquinas, 5 produtos e 8 máquinas e 9 produtos e 9 máquinas.

Os resultados encontrados foram comparados com outras abordagens propostas, variantes do método de busca  $A^*$ , tendo como critérios de comparação o *makespan* obtido e o tempo de execução da busca. Para os problemas 1 e 2, o *makespan* ficou próximo, sendo que em alguns testes foi melhor quando comparado com as outras abordagens, destacando-se a média do tempo de execução do método proposto que foi 300 vezes menor que as demais propostas. Por último, para o problema 3, observou-se que 52% dos resultados obtidos foram iguais ou menores que o valor da média, embora não tenha sido feita a análise estatística sobre estes resultados comparando-os com as outras propostas, já que os métodos

derivados do A\* são inviáveis para problemas com grande espaço de busca, devido ao tempo de execução, onde ficaram em execução por mais de um dia e não chegaram em nenhum resultado.

### **3.3 Uso de Algoritmos Genéticos Adaptativos para a Programação da Produção**

Embora o algoritmo genético tenha ganhado muitos campos de aplicação, é relatado que um algoritmo genético simples sofre frequentemente problemas de convergência prematura e dependência de parâmetros (Zhang; Wang; Zheng, 2006).

Neste contexto, Yin, Yiu e Cheng (2004) propõem um algoritmo genético adaptativo para problemas de programação em flowshop onde as probabilidades dos operadores podem ser ajustadas dinamicamente conforme o valor fitness dos cromossomos. O problema de programação de flowshop é definido no trabalho como um conjunto de  $n$  tarefas e  $m$  máquinas, sendo que o tempo de cada tarefa em uma determinada máquina é determinístico. Algumas suposições foram apontadas onde a ordem das tarefas em uma determinada máquina devem ser preemptivo e não existem buffers intermediários entre as seqüências de máquinas.

O cromossomo é definido como uma cadeia de números inteiros e cada gene representa uma tarefa. O cromossomo completo é definido como a programação completa do flowshop. O tradicional operador de cruzamento de dois pontos é usado para gerar os cromossomos filhos e o operador de mutação é definido através da troca de posição entre dois genes no cromossomo.

Neste trabalho é utilizado um algoritmo genético adaptativo a fim de aumentar a velocidade de convergência do algoritmo, para isso, as probabilidades de cruzamento e mutação devem ser dinamicamente ajustadas conforme a situação atual do processo de evolução. Para realizar este ajuste, foi definida uma função  $C(fi)$  cujo valor varia entre (0,1) apresentando-se os indivíduos na população que estão possuindo grandes variações com o valor de *fitness*. Quando o valor de  $C(fi)$  é grande, significa que a diferença entre o *fitness* dos indivíduos e o *fitness* da população é grande. Nesse caso as taxas de mutação e cruzamento devem ser aumentadas, do contrário, se a diferença entre os indivíduos e a população é pequena, a taxa deve ser reduzida. Com isso, o algoritmo procura produzir boas populações,

sendo que esta estratégia adaptativa pode prevenir a convergência prematura e acelerar a velocidade da busca.

Para validar a proposta o algoritmo genético adaptativo foi comparado com um algoritmo genético simples e ambos algoritmos foram implementados no Matlab 6.1. Como o objetivo deste trabalho foi minimizar o *makespan*, ambos os algoritmos utilizaram o *makespan* como critério de desempenho. A partir dos testes realizados, foi possível concluir que o algoritmo genético simples obteve um *makespan* superior, sendo que convergiu prematuramente e o tempo de execução aumenta quando o tamanho do problema aumenta, já o algoritmo genético adaptativo obteve um *makespan* inferior, atingindo o objeto de minimização e seu tempo de execução não sofreu grandes variações quando o tamanho do problema aumenta. Portanto, o algoritmo genético adaptativo obteve melhores resultados quando comparado com o algoritmo genético simples.

Chan, Chung e Chan (2005) propõem um algoritmo genético adaptativo com genes dominantes (GD) para o problema de programação distribuída de ambientes com várias fábricas e vários produtos. Para este problema de programação distribuída, há três fabricas e cada fabrica possui quatro máquinas. Existe um total de seis tipos de produtos e cada tipo de produto pode ser produzido em qualquer uma dessas fabricas em uma diferente seqüência de produção e tempo de operação. Portanto, o problema neste tipo de programação é determinar a alocação dos produtos para as fabricas e determinar a programação da produção em cada fabrica.

O objetivo neste trabalho é minimizar o *makespan* total da rede de fabricas. Neste trabalho, cada cromossomo representa uma solução correspondente para: (i) a alocação de produtos para as fábricas e (ii) a prioridade de produção de cada produto em cada máquina na rede de fábricas. Um cromossomo é codificado da seguinte maneira: cada gene consiste de cinco números inteiros que representam, respectivamente, fábrica, máquina, produto, operação e dominação (1 para dominante e 0 para normal).

Um cromossomo contém vários genes referentes a várias operações de vários produtos. A prioridade de programação dos trabalhos nas máquinas é definida pela ordem dos genes no cromossomo: maior prioridade à esquerda, menor prioridade à direita. A idéia de Genes Dominantes é proposta para melhorar o desempenho da busca e sua função é representar os genes fortes no cromossomo.

Na população inicial, alguns genes são aleatoriamente designados como DGs. Cada cromossomo pode conter mais de um DG. Durante evoluções, apenas aqueles DGs sofrem cruzamento em cada par de pais para gerar um par de filhos. Cada filho preserva a

maior parte dos genes de um dos pais e herda apenas dos DGs do outro pai. Se estes DGs herdados tornam o filho mais forte do que o pai (com melhor valor de *fitness*), eles se tornarão DGs no filho, senão, eles se tornarão genes normais. Esta idéia assegura que os melhores genes serão passados ao filho. Há dois operadores de mutação: no primeiro, um par de genes no mesmo cromossomo é aleatoriamente selecionado e trocado, alterando, assim, as prioridades de programação das operações dos trabalhos. No segundo tipo de mutação, alguns genes são aleatoriamente selecionados, neste caso o parâmetro “fábrica” ou o parâmetro “máquina”, e mutados com o propósito de aumentar a diversidade genética. Em ambos os tipos de mutação, se o cromossomo, após a mutação, é mais forte que antes, o gene mutado torna-se DG.

É utilizada uma técnica de elitismo para impedir que o melhor cromossomo se perca. O melhor cromossomo é identificado e gravado. Caso ele se perca ou se torne fraco, depois da evolução, ele será inserido na população intermediária para a próxima evolução. Uma estratégia adaptativa é aplicada neste trabalho, onde as taxas de cruzamento e mutação são ajustadas dinamicamente. A taxa de cruzamento dependerá do número de genes dominantes e a taxa de cruzamento é ajustada conforme o desempenho do algoritmo genético na busca. Para fazer este ajuste dinâmico das taxas, em cada geração o melhor cromossomo é salvo, e posteriormente comparado com os melhores cromossomos das gerações futuras. A partir disso, se nas gerações futuras não forem encontradas soluções melhores, significará que o algoritmo genético encontrou um ótimo local. Portanto, nesse caso, as taxas devem ser aumentadas.

Esta abordagem adaptativa usando a idéia de genes dominantes apresentou melhores resultados quando comparado com outras abordagens, sendo algumas delas um algoritmo genético tradicional e um algoritmo genético adaptativo.

Zhang, Wang e Zheng (2006) propõem um algoritmo genético adaptativo com múltiplos operadores para problemas de programação em flowshop, sendo o AG configurado para decidir as melhores seqüências de tarefas a fim de minimizar o *makespan*. O ambiente é composto de  $m$  máquinas e  $n$  tarefas, sendo que as  $n$  tarefas devem ser processadas seqüencialmente nas  $m$  máquinas. O tempo de processamento é dado e cada máquina pode processar mais de uma tarefa e cada tarefa pode ser processada em mais de uma máquina.

A seqüência com que as tarefas são processadas é a mesma para cada máquina. A representação do cromossomo é baseada em tarefas, sendo uma programação codificada como uma seqüência de tarefas, uma seqüência de números naturais, de 1 até  $n \times m$ , onde cada gene representa uma tarefa. Inicialmente a população inicial é gerada aleatoriamente e

em seguida é aplicada à seleção através de um mecanismo baseado na ordem. A probabilidade de seleção de um indivíduo é determinado segundo a posição relativa do indivíduo na população, sendo a população ordenada do melhor para o pior.

Como este trabalho é baseado nas permutações das tarefas, foram testados quatro tipos de operadores de cruzamento para permutações, sendo eles: LOX, PMX, CX e NABEL.

O operador LOX inicia com dois pontos de corte escolhidos aleatoriamente, depois os símbolos que aparecem na seção de cruzamento do primeiro cromossomo pai são removidas a partir do segundo cromossomo pai, deixando alguns “buracos” no cromossomo. Feito isto, os buracos são empurrados das extremidades para o centro até alcançarem a seção de cruzamento. Finalmente estes “buracos” são substituídos com as respectivas posições dos cromossomos pais formando assim os cromossomos filhos.

O operador PMX inicia com dois pontos de corte escolhidos aleatoriamente, que definem uma sublista. Em seguida, este operador realiza trocas no sentido do primeiro cromossomo pai para o segundo cromossomo pai e depois no sentido inverso, isto é, do segundo cromossomo pai para o primeiro cromossomo pai, para evitar cromossomos inválidos.

O operador CX começa copiando o primeiro elemento do primeiro cromossomo pai para o primeiro cromossomo filho (alternativamente, pode-se começar copiando um elemento de qualquer posição da lista). Para evitar a duplicação de um gene no segundo cromossomo filho, é requerido que o gene do primeiro cromossomo pai seja copiado para o primeiro cromossomo filho. Na etapa final, as posições que ficaram em branco, são preenchidas por simples troca de elementos entre o primeiro cromossomo pai e o segundo cromossomo pai. O operador NABEL é um operador criado pelo grupo teórico não abeliano.

Para mutação foram utilizados três operadores, sendo eles: SWAP, INV e INS. O operador SWAP faz a troca de dois genes aleatoriamente. No operador INV a subsequência entre duas diferentes posições aleatórias são invertidas, já no operador INS duas posições são aleatoriamente selecionadas e novos valores são inseridos nestas posições. Neste algoritmo genético adaptativo cada operador tem sua proporção de utilização ajustada dinamicamente conforme seu desempenho durante o processo de busca. Assim os operadores mais fortes são geralmente os mais utilizados.

A fim de testar e validar a proposta, este trabalho foi comparado com um algoritmo genético simples que possui apenas um operador de cruzamento e um operador de mutação. Os resultados mostraram a eficiência da aplicação simultânea de múltiplos

operadores, em especial, a eficiência do controle adaptativo de utilização dos operadores.

Ainda no contexto dos algoritmos genéticos adaptativos, mas considerando também os sistemas de transporte na programação, em especial os veículos autoguiados (AGV – *Automated Guided Vehicle*), Jerald et al. (2006) propõem uma técnica de programação simultânea de peças e AGVs utilizando um algoritmo genético adaptativo.

O objetivo da proposta é minimizar o custo de penalidade para as datas de entrega atrasadas e o tempo de máquina parada. Para isso, quando as peças e os AGVs são bem programados, o tempo de máquina parada pode ser minimizado e sua utilização maximizada. Algumas questões como controle do tráfego, congestionamento, falhas de máquina e despachos de veículos para troca de bateria foram ignorados e deixados para serem tratadas durante o controle em tempo real.

Neste trabalho, é proposto um algoritmo genético adaptativo para encontrar uma ótima seqüência de tarefas. Para cada tarefa, existe uma tabela contendo informações como o número da máquina, tempo de processamento, data devida e custo da penalidade. Os cromossomos são codificados por uma seqüência aleatória de tarefas com um determinado AGV associado a esta tarefa.

É proposto um novo esquema para os operadores genéticos onde é permitido alterar as taxas de cruzamento e mutação conforme o desempenho dos operadores genéticos durante o processo de busca. Se a porcentagem de melhora do fitness da maioria dos cromossomos filhos é maior ou igual a 10% referente aos cromossomos pais, então a probabilidade de ocorrência do operador genético será aumentada em 0.05 para operações de cruzamento e 0.005 para operações de mutação. Se a porcentagem de piora do fitness dos cromossomos filhos é menor que 10% referente aos cromossomos pais, então a probabilidade de ocorrência do operador genético será reduzida em 0.05 para operações de cruzamento e 0.005 para operações de mutação. Se a porcentagem de melhora ou piora do fitness da maioria dos cromossomos filhos estiver dentro de  $\pm 10\%$  dos valores fitness dos seus respectivos cromossomos pais, então a probabilidade de ocorrência dos operadores não será alterada. O ajuste das taxas deve estar entre 0.5 e 1.0 se o operador genético for o cruzamento e 0.00 e 0.10 se o operador genético for mutação. Esta variação estimula os operados com bom desempenho a produzirem mais filhos enquanto isso reduz as chances de operadores com fraco desempenho de destruírem os indivíduos em potencial durante o processo de recombinação.

A função de *fitness* tem como objetivo minimizar o custo de penalidade e tempo de máquina parada, onde informações como o tempo de transporte é conhecido. Os

resultados obtidos indicaram que a abordagem proposta é eficiente na geração de boas soluções para o problema de programação de FMS quando comparado com um algoritmo genético tradicional.

Sanches et al. (2008) propõem o uso de um algoritmo genético adaptativo para uma programação reativa da produção. Neste trabalho, a modelagem do algoritmo genético utilizado, em especial a codificação do cromossomo, é a mesma proposta por Deriz (2007) e Morandin et al. (2008), mas com uma diferença nas taxas de cruzamento e mutação, que são ajustadas dinamicamente durante o processo de busca do algoritmo genético.

O ajuste dinâmico das taxas de cruzamento e mutação foi utilizado a fim de encontrar bons parâmetros para o algoritmo genético e evitar o problema de convergência prematura. Este trabalho se baseia no esquema de ajuste dinâmico proposto por Jerald et al. (2006), sendo que este esquema envolve algumas regras que ajustam as taxas de cruzamento e mutação de acordo com o desempenho dos operadores genéticos.

Os resultados deste trabalho foram comparados com um algoritmo genético tradicional, em que foram observados os valores de *makespan* e o tempo de execução da busca. A partir da análise feita sobre os resultados obtidos, foi possível observar que este método pode ser aplicado para soluções de problemas de programação reativa da produção, a fim de atingir um bom compromisso de *makespan* em um baixo tempo de execução.

### **3.4 Programação Simultânea da Produção de Máquinas e Sistemas de Transporte**

Conforme apresentado por Jerald et al. (2006) e em outros trabalhos que serão discutidos a seguir, existem várias abordagens fazendo uso simultâneo de máquinas e sistemas de transportes para a programação de produtos usando Algoritmos Genéticos. Este tipo de abordagem têm apresentado resultados promissores relacionado aos critérios comumente utilizados como o *makespan*.

Ulusoy, Sivrikaya e Bilge (1997) propõem uma programação simultânea de máquinas e AGVs em um FMS com o objetivo de minimizar o *makespan*. Neste trabalho os tipos e números de máquinas são conhecidos e existe um espaço suficiente no buffer de entrada e saída de cada máquina. Assume-se que o número de AGVs é dado e possuem as mesmas características como a velocidade. O arranjo físico dos caminhos entre as máquinas é

dado e os tempos de viagem em cada segmento do caminho são conhecidos. O tempo de processamento das operações são gerados aleatoriamente entre um intervalo de 3-15 minutos. Situações de *deadlock* não são consideradas, mas cromossomos que representam situações de *deadlock* são penalizados com um valor de *makespan* muito alto.

Para resolver o problema proposto, um algoritmo genético foi utilizado, de modo que a codificação do cromossomo representa a seqüência das operações e os AGVs designados, ou seja, cada gene é composto por uma operação e um AGV associado a esta operação. Neste caso, a operação é representada por um caractere *ASCII* enquanto o AGV é representado por um numero inteiro.

Um operador especial de cruzamento é utilizado para criar um cromossomo filho a partir de dois cromossomos pais. Para isso, iniciando a partir das primeiras operações dos cromossomos pais, um dos cromossomos pais são selecionados aleatoriamente e a próxima operação que não foi selecionada torna-se a próxima operação no cromossomo filho. Se o AGV selecionado para aquela operação é o mesmo em ambos os cromossomos pais, então este AGV será selecionado para o cromossomo filho. Se não for o mesmo AGV, então um dos AGVs dos cromossomos pais será selecionado aleatoriamente. Este operador favorece ao cromossomo filho herdar boas soluções contidas nos cromossomos pais.

Dois operados de mutação são utilizados, um para o AGV e outro para a operação. O operador para a operação escolhe duas operações aleatoriamente no cromossomo pai e efetua a troca entre estas operações, já para o operador referente ao AGV, a mutação ocorrerá segundo uma taxa de mutação especificada para este operador. Se para um determinado gene à taxa indicar que deve ocorrer a mutação, um dos AGVs devem ser escolhidos aleatoriamente e designado para operação daquele gene. Com isto, pode resultar na alocação do mesmo AGV para um gene em particular, e o objetivo deste operador é evitar a perda das boas alocações dos AGVs. A função de *fitness* definida neste trabalho é encontrar o mínimo *makespan*.

Testes realizados comparados com a complexidade de limite inferior (*lower bound*) mostraram que a proposta obteve um bom desempenho, sendo que a maioria das soluções encontradas pelo AG foi igual ao *lower bound* implicando que a solução é ótima.

Neste enfoque, Haq, Karthikeyan e Dinesh (2003) propõem uma programação integrada de FMS, em que a programação das máquinas é adaptada com a programação do AGV. Um algoritmo heurístico proposto por Giffler e Thompson (1960) com seis diferentes

regras de despacho é utilizado a fim de minimizar o *makespan*. As informações obtidas através do algoritmo heurístico são enviadas como entrada para a programação dos AGVs. Esta programação é feita através do algoritmo genético, que procura minimizar dois objetivos, à distância percorrida e o número de retornos do AGV.

Inicialmente é fornecida uma matriz de P peças e M máquinas e uma matriz de incidência indicando o número de seqüência de operação de cada peça em uma determinada máquina segundo o roteiro cadastrado. Após isso, através da heurística de Giffler e Thompson com as regras de prioridade de despacho SPT, LPT, LOR, MOR, LWR e MWR é possível obter o melhor roteiro das tarefas com o mínimo *makespan*, sendo que este roteiro será fornecido como entrada para a programação do AGV.

Para a programação do AGV um AG é aplicado para a minimizar a distância percorrida e o número de retornos do AGV permitindo encontrar uma seqüência ótima de máquinas. O cromossomo é codificado, nesta abordagem, como uma seqüência de roteiros para uma determinada peça onde, este roteiro foi obtido anteriormente através da heurística de Giffler e Thompson.

O tempo de transporte depende da distância entre as máquinas e a velocidade do AGV, para isso, duas matrizes são fornecidas possuindo ambas as informações. A função de *fitness* minimiza a distância percorrida e o número de retornos do AGV. Testes realizados demonstraram que a proposta quando comparada com o algoritmo Simulated Annealing obteve um melhor desempenho.

Sankar et al. (2004) propõem uma programação simultânea de tarefas/máquinas e AGVs utilizando um algoritmo genético. O objetivo da proposta é encontrar uma seqüência ótima de peças para a produção, diminuindo os tempos de esperas das peças a serem produzidas devido à indisponibilidade de alguns recursos, resultando em uma minimização geral do *makespan*.

Neste trabalho, o número de tarefas simultâneas permitidas no sistema é limitado e o tempo de processamento total é constante. O problema de programação integrada é dividido em dois subproblemas: Um gerador de programação, que gera a programação das tarefas através de um algoritmo genético e um simulador de eventos discretos que é utilizado para simular os AGVs no sistema. Ambos os subproblemas são ligados através de uma estrutura iterativa, que facilita a busca para uma boa solução. Após um certo número de iterações o procedimento terminará com a melhor solução encontrada para este número de iterações. Para cada solução obtida através do subproblema 1, as operações do FMS serão simuladas e o *makespan* será encontrado a partir do subproblema 2. Em seguida, o valor do

*makespan* será retornado para o subproblema 1. Este processo se repete até cumprir um certo número de iterações.

Os cromossomos são codificados para o subproblema 1 através de uma seqüência aleatória de peças. O cruzamento é feito através de um par de cromossomos pais, que são selecionados segundo uma dada probabilidade, e os pontos de cruzamento são selecionados aleatoriamente, em que as seqüências das peças após o ponto de cruzamento do primeiro cromossomo pai são reorganizadas de modo que fiquem no mesmo modo que aparecem no segundo cromossomo pai. Este processo se repete para o segundo cromossomo pai, em que as seqüências das peças após o ponto de cruzamento do segundo cromossomo pai são reorganizadas de modo que fiquem no mesmo modo que aparecem no primeiro cromossomo pai. Para a operação de mutação, um cromossomo é selecionado de acordo com a taxa de mutação, sendo que dois genes destes cromossomos são selecionados aleatoriamente e trocados entre si.

O modelo de simulação do sistema de eventos discretos que é utilizado no subproblema 2 incorpora a maioria dos eventos e atividades em tempo real, que incluem: AGVs e Máquinas bloqueadas; regras de despacho para os AGVs; atrasos devido a não disponibilidade dos AGVs e outras condições como o roteiro definido para cada peça, capacidades limite para o buffer de entrada/saída e velocidade de transporte do AGV.

Os resultados obtidos indicaram que a abordagem proposta é eficiente na geração de soluções boas para o problema de programação de FMS e o número de iterações requeridos pelo algoritmo genético são muito inferiores. Os resultados de *makespan* obtidos foram menores quando comparado com os resultados pelo algoritmo *Kangaroo*.

Como a maioria dos pesquisadores tem tratado a programação de máquinas e veículos como problemas independentes e as pesquisas tem enfatizado somente um único objetivo de otimização. Reddy e Rao (2006) propõem uma programação simultânea de máquinas e AGVs no FMS com o objetivo de minimização do *makespan*, *flow time* (tempo de fluxo) e *tardiness* (atraso absoluto) usando um híbrido algoritmo genético com soluções não dominadas. As soluções não dominadas são utilizadas para abordagens com mais de um objetivo ou multi-objetivo em que, as soluções quando comparadas com todas as outras, serão melhores ou piores em um ou mais objetivos.

O híbrido algoritmo genético proposto é uma combinação de um algoritmo genético junto com uma técnica heurística reduzindo assim o espaço de busca, minimizando as restrições na busca e aumentando a eficiência da busca genética. O algoritmo heurístico é

embutido dentro do AG, sendo utilizado durante o processo de cálculo da função de *fitness*, em que verifica-se o status do AGV e das tarefas, calculando a disponibilidade do AGV para a tarefa designada. O cromossomo é representado como a seqüência de operações, onde cada operação significa o processamento de uma peça específica em uma determinada máquina.

A função de *fitness* é a minimização dos três objetivos *makespan*, *flow time* e *tardiness*. Para a operação de cruzamento é feita inicialmente uma seleção aleatória de uma tarefa nos cromossomos pais onde as operações da tarefa selecionada são copiadas diretamente para as respectivas posições dos cromossomos filhos. A operação de mutação seleciona dois genes aleatoriamente no cromossomo e as operações associadas com estas posições são trocadas. Após o AG fornecer uma ótima seqüência de operações, as tarefas são encaminhadas para a programação do AGV.

Os autores testaram o desempenho do algoritmo proposto para apenas um objetivo, neste caso, a minimização do *makespan*. Para isso, utilizaram os problemas reportados no trabalho de Ulusoy, Sivrikaya e Bilge (1997). Os resultados gerados foram superiores na maioria dos casos ou no mínimo iguais aos resultados propostos por Ulusoy, Sivrikaya e Bilge (1997).

### **3.5 Considerações finais**

Neste capítulo foi apresentado um levantamento de trabalhos propostos para a resolução do problema de programação da produção, em especial para a resolução do problema da programação de produtos com uso simultâneo de máquina e sistemas de transporte usando algoritmos genéticos.

No próximo capítulo será apresentada a modelagem e o processo de busca para a resolução do problema proposto neste trabalho, que é a obtenção de uma programação de produtos com uso simultâneo de máquina e sistemas de transporte em sistemas de manufatura com recursos compartilhados usando algoritmos genéticos adaptativos.

## **4 PROPOSTA DE MODELAGEM POR ALGORITMO GENÉTICO ADAPTATIVO PARA PROGRAMAÇÃO REATIVA DA PRODUÇÃO DE PRODUTOS COM USO SIMULTÂNEO DE MÁQUINAS E SISTEMAS DE TRANSPORTE EM SISTEMAS DE MANUFATURA**

### **4.1 Considerações Iniciais**

Sistemas de manufatura com compartilhamento de recursos, como máquinas e Sistemas de Transporte têm sido utilizados para automação da produção e permitem flexibilidade condizente com as necessidades de uma boa parcela de manufaturas do mercado. Esta flexibilidade ocorre, pois as máquinas podem ser utilizadas para várias operações diferentes e os produtos podem ser processados por várias máquinas diferentes. O problema é que esta flexibilidade exige um esforço considerável para definir a programação da produção e utiliza os recursos disponíveis ao longo do tempo, com o objetivo de satisfazer certos critérios de desempenho.

A maioria dos pesquisadores tem tratado o problema da programação da produção de produtos e Sistemas de Transporte como problemas independentes para estes tipos de sistema. Entretanto, alguns pesquisadores têm enfatizado a importância da programação simultânea de produtos e Sistemas de Transporte (Reddy; Rao, 2006).

No grupo de trabalho do TEAR, uma busca baseada em algoritmos genéticos para a programação da produção de sistema de manufatura com recursos compartilhados foi proposta por Deriz (2007). O problema é modelado através do algoritmo genético e um cromossomo é codificado de maneira a indicar uma programação da produção, ou seja, uma solução completa para o problema. A solução encontrada é 'traduzida' para uma programação da produção através da leitura dos genes do cromossomo. O critério de desempenho escolhido é o *makespan*. O próprio autor indica tanto a utilização de algoritmos genéticos adaptativos para melhorar o desempenho do algoritmo genético na busca como a utilização do transporte no problema de programação, a fim de tornar este sistema mais próximo da realidade.

Sendo assim, neste trabalho é proposta uma modelagem por algoritmo genético adaptativo para programação da produção de produtos com uso simultâneo de máquinas e sistemas de transporte em um sistema de manufatura com recursos compartilhados.

Neste capítulo, serão definidas as características e restrições do sistema e será exibida a proposta de modelagem do algoritmo genético adaptativo para a solução do

problema de programação simultânea de produtos e sistemas de transporte.

## **4.2 Caracterização e restrições da Programação da Produção Proposta**

O problema da programação de produtos com uso simultâneo de máquinas e sistemas de transporte em um sistema de manufatura com compartilhamento de recursos envolvem decisões na alocação dos recursos de produção e sistemas de transporte ao longo do tempo, bem como a escolha dos roteiros de fabricação para cada lote de peças na obtenção de um tipo de produto, definindo o momento de execução de cada etapa. Um dos critérios comumente utilizado para definir o desempenho da produção é o valor de *makespan*.

Assim, dado um sistema de manufatura, supõe-se que a meta seja fabricar certo número de produtos em um determinado tempo de produção. Para isso, dispõe-se de certos recursos de produção cujas operações sobre diferentes materiais, nos diferentes estágios da obtenção de um produto, precisam ser determinadas. O problema da programação da produção desse sistema será, no corrente trabalho, definido como a tarefa de se determinar previamente quais operações deverão ser feitas e em quais instantes, de forma que todos os produtos estejam finalizados no menor tempo possível.

O presente trabalho trata o problema da programação da produção de uma fábrica, onde há várias máquinas e vários tipos de produtos a serem produzidos. Cada tipo de produto pode ter vários roteiros de fabricação. Um roteiro de fabricação é uma seqüência de operações que deve ser seguida na ordem em que se encontra, onde cada operação é uma etapa da fabricação do produto executada em uma determinada máquina e, ao final da seqüência, o produto estará pronto. Pode haver mais de uma máquina possível para executar uma mesma operação, o que leva um produto a poder ter mais de um roteiro de fabricação. Assim, os roteiros de fabricação de cada produto são representados pelas máquinas através das quais ele deve passar para ser produzido.

O problema da programação dos veículos encontra-se no momento em que uma determinada peça precisa ser transportada até outra máquina através de um veículo. Nesta etapa, mais de um veículo pode estar disponível e o problema será qual veículo deve ser selecionado. Para tratar este problema, este trabalho utiliza duas regras de despacho, a regra STT/D (*Shortest Travel Time/Distance* - Menor Tempo de Viagem) que é apresentada por Benincasa (2003) e RV (*Random Vehicle rule* – Veículo Aleatório) que é apresentada por Egbelu e Tanchoco (*apud* BENINCASA, 2003).

Assim, é importante definir quais as características e restrições do problema a ser tratado. Estas características são dadas a seguir:

- (i) Inicialmente nenhum produto estará sendo produzido;
- (ii) Um tipo de produto será produzido seguindo um único roteiro de fabricação, ou seja, se houver vários roteiros possíveis, deverá ser selecionado apenas um;
- (iii) Um determinado produto deverá passar por todas as máquinas do roteiro de fabricação escolhido, sem alteração da ordem definida;
- (iv) Uma operação será executada em apenas uma máquina, em um determinado tempo, ou seja, um produto passará por apenas uma máquina em determinado tempo;
- (v) Uma máquina processará apenas um produto por vez;
- (vi) Tempos de operação dos produtos nas máquinas são determinísticos e previamente conhecidos;
- (vii) Máquinas nunca falham;
- (viii) Uma vez que uma determinada operação é iniciada (que um determinado produto passa por uma máquina), ela não é interrompida até que seja concluída;
- (ix) O número de veículos é dado e os veículos são iguais em relação à velocidade e características de transporte de cargas;
- (x) Os veículos partem inicialmente de um ponto caracterizado como estação de Carga/Descarga;
- (xi) Um veículo carrega somente um produto por vez;
- (xii) Os tempos de transporte para cada ponto do chão de fábrica são conhecidos;
- (xiii) Existem áreas de armazenamento intermediário, com capacidade infinita entre as máquinas;
- (xiv) Tempos de *setup* de máquinas estão incluídos nos tempos de operação.

Neste trabalho, os critérios adotados para definir o desempenho da produção são baseados na obtenção do mínimo *makespan* e o tempo de obtenção da resposta.

### 4.3 Modelagem do Algoritmo Genético Adaptativo

A modelagem do problema através do uso do algoritmo genético adaptativo será a mesma modelagem (somente as etapas de codificação do cromossomo, cruzamento e mutação) proposta por Morandin et al. (2008) e Deriz (2007). Nestas abordagens, um cromossomo é codificado de modo a indicar quais os produtos e seus respectivos roteiros de fabricação devem ser processados. A partir disso, o algoritmo genético utilizado é encarregado de realizar toda a programação da produção, ou seja, definir em que momento e em qual máquina tais produtos serão produzidos.

Segundo Morandin et al. (2008) e Deriz (2007), a população inicial é gerada de acordo com os produtos que necessitam ser produzidos pelos possíveis roteiros de fabricação para cada produto. Feito isso, cada cromossomo será avaliado de acordo com o seu *makespan* e um valor de *fitness* que será atribuído a ele. No presente trabalho, durante a etapa do cálculo do *fitness* é realizada a programação da produção e os produtos são atribuídos às máquinas segundo os roteiros cadastrados. Ainda, os veículos são programados com regras de despacho, cuja finalidade é transportar um determinado produto para uma determinada máquina ou estação de carga e descarga. Com isso, é possível encontrar o valor de *makespan* e o valor de aptidão para um determinado cromossomo, respectivamente. Este processo será abordado com mais detalhe adiante.

Em seguida, serão selecionados cromossomos para se aplicar cruzamento e mutação a fim de gerar novos indivíduos para a próxima população. No presente trabalho, o elitismo escolhido difere do elitismo utilizado por Morandin et al. (2008), que aplicaram o elitismo de modo que o melhor cromossomo de cada geração seja selecionado para a população intermediária para sofrer cruzamento e mutação e gerar descendentes para a próxima geração. Neste trabalho, o elitismo utilizado copia o melhor cromossomo de uma geração para outra sem alterações, ou seja, o processo de cruzamento e mutação não são aplicados nesta solução, evitando assim a perda desta boa solução.

No presente trabalho, o algoritmo genético adaptativo utilizado é o mesmo utilizado em Sanches et al. (2008), em que as taxas de cruzamento e mutação são ajustadas dinamicamente conforme o desempenho deste algoritmo na busca. Tais ajustes serão abordados com mais detalhes neste capítulo.

O critério de parada é quando um determinado número de gerações é atingido ou quando o algoritmo converge. Na Figura 9, é possível observar um fluxograma do

algoritmo genético adaptativo utilizado, em que uma população inicial de cromossomos é gerada, sendo que esta população é avaliada e cada cromossomo recebe uma nota de aptidão, que representa a qualidade de sua solução. Em geral, os cromossomos mais aptos são selecionados para a próxima geração e os menos aptos são descartados. O método de seleção deve priorizar cromossomos com valores de aptidão mais altos, mas sem comprometer a diversidade genética da população. Uma parte selecionada dos cromossomos pode sofrer modificações através de operadores de cruzamento e mutação, de modo que após a utilização de cada operador, os valores de *fitness* da população devem ser salvos. Feito isso, as taxas de cruzamento e mutação são ajustadas conforme o desempenho do algoritmo genético adaptativo. Este processo é repetido até que o critério de parada seja atingido.

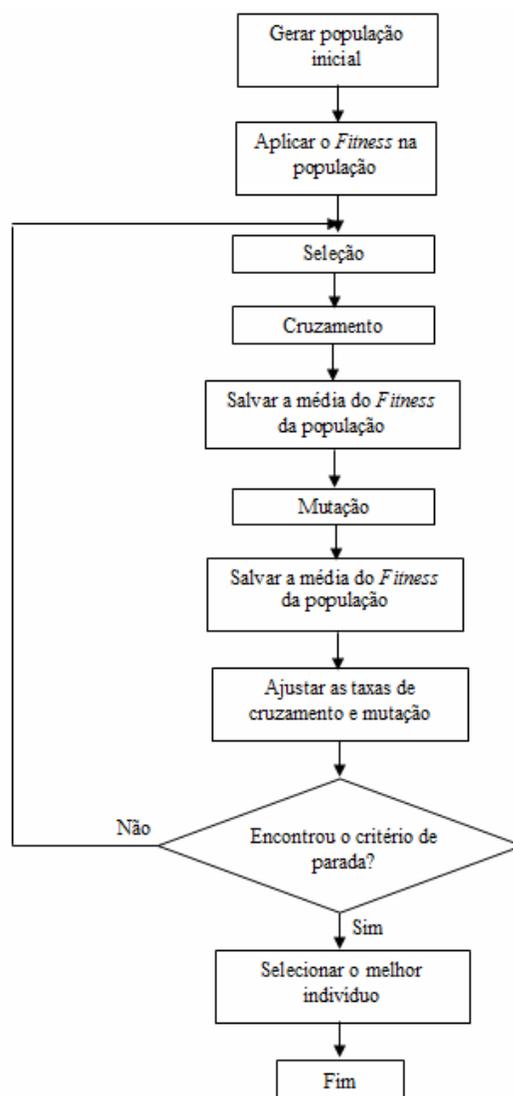


Figura 9: Fluxograma do algoritmo genético adaptativo.

Nos próximos tópicos, serão apresentados mais detalhes dessa modelagem junto com a programação da produção do sistema de transporte e a abordagem adaptativa para o algoritmo genético.

#### 4.3.1 Codificação do cromossomo

Segundo Morandin et al. (2008), a codificação do cromossomo é definida da seguinte maneira: seja  $n$  o número de produtos a serem produzidos,  $s$  o número máximo de roteiros possíveis para cada produto, e  $t$  o número máximo de operações em todos os possíveis roteiros de fabricação para os  $n$  produtos. Um cromossomo é representado por um conjunto de valores inteiros de tamanho  $n+n*t$ , subdividido em  $n$  conjuntos de genes de tamanho  $t+1$ .

Cada conjunto de genes representa um produto a ser processado seguido de seu roteiro de fabricação, em que o primeiro gene deste conjunto é o produto e os  $t$  genes seguintes representam as máquinas do roteiro de fabricação escolhido para aquele produto.

Defina-se  $P_{i_p}$  como o produto a ser processado e  $M_{i_p, K_{i_p}}$  como uma máquina do roteiro de fabricação deste produto, onde  $K_{i_p}$  representa a ordem da máquina no roteiro do produto, ou seja, a  $K_{i_p}$ -ésima máquina pela qual o produto  $P_{i_p}$  deverá passar. Sendo que,  $\{ i_1, i_2, i_3, \dots, i_p \} \in \{ 1, 2, \dots, n \}$  &  $i_g \neq i_j, g \neq j$ , e  $K_{i_p} \in \{ 1, 2, \dots, t \}$  &  $K_{i_p_g} \neq K_{i_p_j}, g \neq j$ . Assim, temos uma representação completa do cromossomo na Figura 10.

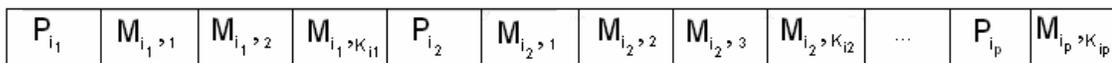


Figura 10 – Representação completa do cromossomo (Morandin et al., 2008).

Para que cada conjunto apresente tamanho fixo  $t+1$ , caso o conjunto represente um produto com  $K_{i_p} < t$ , há a necessidade de inserção de genes vazios, sendo preenchidos com o valor 0 entre o gene da última máquina do roteiro e o indicador do próximo produto no cromossomo.

A fim de facilitar a modelagem e a implementação da proposta, os autores redefinem a modelagem do cromossomo de um modo mais simplificado, de forma que cada conjunto de genes contenha o produto a ser produzido e o índice do roteiro escolhido,

conforme apresentado na Figura 11, em que  $P_{i_p}$  representa o produto a ser processado e  $R_{i_p, j}$  representa o roteiro escolhido para o produto  $P_{i_p}$ , entre os roteiros possíveis. Sendo que  $\{i_1, i_2, i_3, \dots, i_p\} \in \{1, 2, \dots, n\}$  &  $i_g \neq i_j, g \neq j, e j \in \{1, 2, \dots, s\}$ .

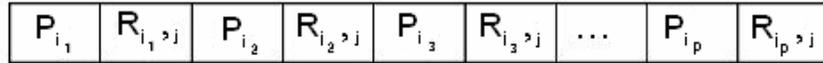


Figura 11 – Representação simplificada do cromossomo (Morandin et al., 2008).

Como é retratado na Figura 10, os produtos são processados inicialmente da esquerda para a direita, em que primeiro é alocada a máquina  $M_{i_1, 1}$  para  $P_{i_1}$ , em seguida é alocada  $M_{i_2, 1}$  para  $P_{i_2}$ , sendo que este processo é realizado até o último produto do cromossomo, em que é alocada  $M_{i_p, k_{i_p}}$  para  $P_{i_p}$ . Quando um produto finaliza uma operação em uma máquina, é alocada a este produto a próxima máquina cadastrada em seu roteiro de fabricação. Esta etapa de alocação de máquinas é realizada de acordo com a disponibilidade das máquinas, ou seja, um produto deverá esperar a liberação de uma máquina caso a mesma se encontre ocupada. Quando uma máquina se torna disponível, o cromossomo é percorrido novamente na ordem da esquerda para a direita até encontrar um produto que necessite ser processado nesta máquina. Todo este procedimento entra em um processo de iteração até que todas as máquinas de todos os roteiros contidos no cromossomo tenham sido alocadas para os respectivos produtos.

#### 4.3.1.1. Exemplo de codificação do cromossomo

Um cromossomo representa quais os produtos que devem ser produzidos, seguido dos respectivos roteiros de fabricação. Após isso, durante a etapa de cálculo do *fitness*, os produtos serão alocados para suas respectivas máquinas com o uso de um veículo, em um determinado momento, a fim de obter uma programação da produção.

Morandin et al. (2008) e Deriz (2007) apresentam um exemplo de codificação do cromossomo, em que consideram uma fábrica com seis máquinas  $M_1, M_2, M_3, M_4, M_5, M_6$  e três tipos de produtos  $P_1, P_2, P_3$ .

Na Tabela 1 estão relacionados os roteiros de fabricação possíveis para os produtos.

Tabela 1 – Produtos e roteiros de fabricação (Morandin et al., 2008).

<b>Produto</b>	<b>Roteiros de Fabricação</b>
P <sub>1</sub>	R <sub>11</sub> (M <sub>1</sub> , M <sub>2</sub> , M <sub>6</sub> )
	R <sub>12</sub> (M <sub>4</sub> , M <sub>5</sub> , M <sub>6</sub> )
P <sub>2</sub>	R <sub>21</sub> (M <sub>1</sub> , M <sub>2</sub> , M <sub>5</sub> , M <sub>6</sub> )
	R <sub>22</sub> (M <sub>3</sub> , M <sub>4</sub> , M <sub>5</sub> , M <sub>6</sub> )
P <sub>3</sub>	R <sub>31</sub> (M <sub>4</sub> , M <sub>3</sub> , M <sub>2</sub> )
	R <sub>32</sub> (M <sub>4</sub> , M <sub>1</sub> , M <sub>5</sub> )

Na Figura 12 é apresentada a codificação completa de um cromossomo C<sub>1</sub>, gerado a partir dos produtos e roteiros de fabricação relacionados na Tabela 1. O primeiro gene indica o produto 1 e os 4 genes seguintes indicam as máquinas para o produto 1. O sexto gene indica o produto 3 e o décimo primeiro gene indica o produto 2. Na aplicação, os genes ‘vazios’ são codificados com o valor 0 (zero).

C <sub>1</sub>	1	1	2	6	0	3	4	1	5	0	2	1	2	5	6
----------------	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Figura 12 – Cromossomo (Morandin et al., 2008).

O mesmo cromossomo C<sub>1</sub> é apresentado com a codificação simplificada (Figura 13), em que o primeiro e o segundo genes indicam que o produto P<sub>1</sub> será produzido pelo roteiro R<sub>11</sub>, o terceiro e o quarto genes indicam que o produto P<sub>3</sub> será produzido pelo roteiro R<sub>32</sub> e quinto e o sexto genes indicam que o produto P<sub>2</sub> será produzido pelo roteiro R<sub>21</sub>.

C <sub>1</sub>	1	1	3	2	2	1
----------------	---	---	---	---	---	---

Figura 13 – Cromossomo simplificado (Morandin et al., 2008).

#### 4.3.2. Seleção

Diferente da abordagem proposta por Morandin et al. (2008) e Deriz (2007), o método de seleção por torneio foi utilizado neste trabalho, onde uma população inicial composta por 30 indivíduos é gerada aleatoriamente. Os indivíduos passam por um processo de torneio para de serem copiados para a população intermediária. Então, cinco candidatos

são selecionados aleatoriamente através da população e o melhor indivíduo, baseado no valor da função de *fitness*, é copiado para a população intermediária. A seleção por torneio é repetida até que toda a população intermediária seja preenchida.

#### 4.3.3. Cruzamento

Segundo Morandin et al. (2008), o cruzamento é feito entre dois cromossomos pais  $C_1$  e  $C_2$ , que trocam o conjunto de genes referente ao mesmo produto  $P_{i_p}$  em cada um dos cromossomos, e geram dois cromossomos filhos  $F_1$  e  $F_2$ , que irão para a nova população (Figura 14).

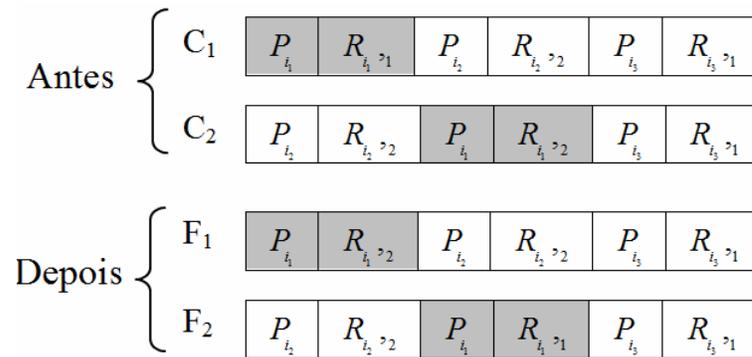


Figura 14: Cruzamento em relação ao produto  $P_{i_1}$ .

#### 4.3.4. Mutação

A mutação (Figura 15) é realizada em um cromossomo, em que um determinado produto  $P_{i_p}$  é escolhido aleatoriamente e o seu roteiro de fabricação é trocado por outro possível roteiro (Morandin et al., 2008). Através deste processo, é possível observar que ocorre perda de material genético e novas características são introduzidas ao cromossomo, garantindo maior diversidade na população.

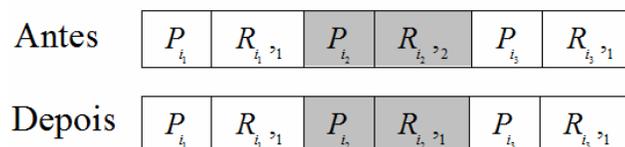


Figura 15: Mutação.

#### 4.3.5. Função de *fitness*

A função de *fitness* é baseada no valor de *makespan* de cada cromossomo e tem a finalidade de atribuir uma determinada nota aos indivíduos, de acordo com sua aptidão. Para obter o valor de *makespan*, é necessário definir como o veículo será programado para transportar os produtos entre as máquinas.

##### 4.3.5.1. Programação da Produção dos Veículos usando Regras de Despacho

A programação dos sistemas de transporte é utilizada para transportar os produtos entre seus roteiros de fabricação, que são obtidos através da leitura do cromossomo. A partir disto, as decisões de escolha de qual veículo realizará a tarefa são baseadas em regras de despacho adotadas pelo sistema.

Neste trabalho, foram utilizadas duas regras:

- RV: Utilizado quando todos os sistemas de transporte estão disponíveis na estação Carga/Descarga.
- STT/D: Utilizado quando existe mais de um veículo disponível e estes veículos não se encontram na estação de Carga/Descarga.

Após definir quais as regras de despacho que serão utilizadas, é necessário definir como estas regras serão aplicadas durante o processo de alocação de produtos às máquinas. Para isso, os passos que representam a lógica de execução do veículo são apresentados a seguir:

1. Mover todos os produtos para a estação Carga/Descarga de acordo com a ordem fornecida pelo cromossomo, que é da esquerda para a direita;
2. Verificar qual o produto deve ser transportado segundo sua prioridade;
  - 2.1. Verificam-se quais os veículos encontram-se disponíveis e se aplica a regra de despacho apropriada;
  - 2.2. Mover o veículo a partir do ponto corrente onde ele se encontra até o ponto onde esta sendo solicitado. O veículo pega e move o produto para a próxima máquina designada conforme cadastrado no roteiro do produto;

- 2.3. Após transportar um determinado produto, o veículo fica estacionado no ponto de estacionamento da máquina ou no ponto de estacionamento da estação Carga/Descarga em que foi entregue o produto;
3. Voltar à etapa 2. Este processo repete-se até que todos os produtos tenham sido produzidos e entregues a estação Carga/Descarga.

#### 4.3.5.2. Exemplo de uma Programação da Produção de Produtos com uso Simultâneo de Máquinas e Veículos

Para ilustrar como será o resultado final da aplicação da programação da produção de produtos com uso simultâneo de máquinas e sistemas de transporte, um cromossomo subdividido em três conjuntos de genes, em que cada conjunto é composto por cinco genes, é apresentado na Figura 16 para que a partir dele possa ser aplicado todo o processo de programação descrito anteriormente. Após isso, é possível observar o gráfico de Gantt (Figura 17), que representa a programação da produção obtida. Neste exemplo, foram utilizados 6 máquinas, 3 produtos e 2 sistemas de transporte, em que V indica o veículo escolhido, M a máquina utilizada e P o produto processado.



Figura 16 – Exemplo de um cromossomo.

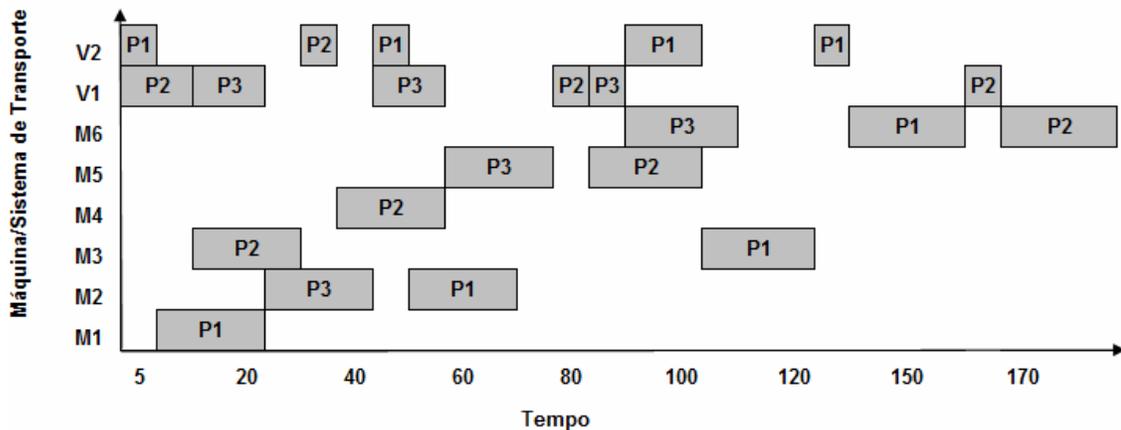


Figura 17 – Gráfico de Gantt.

#### 4.3.5.3. Obtendo o valor de *Makespan*

Para calcular o valor de *makespan* e obter um valor de aptidão para cada cromossomo, é necessário encontrar o tempo de conclusão de uma determinada operação de um produto, que é dado por (1).

$$O_{ij} = T_{ij} + P_{ij} + E_{ij} \quad (1)$$

Em que:

$O_{ij}$  = Tempo de conclusão da operação, em que  $j$  é o  $j$ -ésima operação do produto  $i$

$T$  = Tempo de transporte,  $P$  = Tempo de processamento da operação e  $E$  = Tempo de espera

Após encontrar o tempo de conclusão de uma determinada operação de um produto, é possível encontrar o tempo total para a obtenção do produto  $P_i$ , dado em (2).

$$P_i = \sum_{j=1}^n O_{ij} \quad (2)$$

Na equação (3) é possível encontrar o *makespan*,  $mkp$ , baseado no tempo de obtenção do último produto.

$$mkp = \text{Max}(P_i) \quad (3)$$

Os cromossomos com menor valor de *makespan* são considerados mais aptos do que cromossomos com valor maior. Para obter o valor *fitness*  $f_i$ , em que  $i$  é um determinado cromossomo, uma operação é realizada nos valores de *makespan* da geração corrente, dado em (4).

$$f_i = mkp_{\max} - mkp_i + mkp_{\min} \quad (4)$$

Onde,  $mkp_{\max}$  é o maior valor de *makespan* da geração atual,  $mkp_{\min}$  é o menor valor de *makespan* da geração atual e  $mkp_i$  é o valor de *makespan* do cromossomo  $i$  na geração atual. Esta operação atribui valores de *fitness* mais altos a indivíduos com valores de

*makespan* mais baixos na geração, conforme proposto por Chiu e Fu (1997). Essa expressão é usada para calcular o valor de *fitness* de um cromossomo, de tal forma que um cromossomo com valor menor de *makespan* (mais adaptado), obtenha um valor maior de *fitness* e tenha maior probabilidade de ser selecionado para gerar descendentes para a próxima geração.

#### 4.3.6. Ajuste dinâmico das taxas de mutação e cruzamento

A fim de encontrar bons parâmetros para o AG e evitar a convergência prematura do mesmo, este trabalho utiliza um método para ajustar dinamicamente os parâmetros do AG durante a busca. O método é baseado na abordagem proposta por Jerald et al. (2006) e utilizado em Sanches et al. (2008).

Este esquema envolve algumas regras que ajustam as taxas de cruzamento e mutação dinamicamente de acordo com o desempenho dos operadores genéticos. Os passos do algoritmo genético adaptativo utilizado e os possíveis ajustes das taxas dos operadores genéticos são apresentados a seguir:

1. Gerar a população inicial aleatoriamente e calcular o valor de *fitness* de cada indivíduo;
2. Aplicar o método de seleção na população corrente;
3. Os indivíduos selecionados devem passar pelo processo de cruzamento proposto de acordo com a taxa de cruzamento. Após isso, verificar a média do *fitness* da população;
4. Salvar a média do *fitness* da população após o processo de cruzamento;
5. Aplicar o processo de mutação de acordo com a taxa de mutação e verificar a média do *fitness* da população;
6. Salvar a média do *fitness* da população após o processo de cruzamento;
7. Ajustar as taxas de cruzamento e mutação através da seguinte regra:
  - Se a porcentagem de melhora da média do *fitness* dos cromossomos filhos gerados for maior ou igual a 10% do que os cromossomos pais, então a probabilidade de ocorrência do operador genético deve ser aumentado de 0.05 para operações de cruzamento e 0.005 para operações de mutação.
  - Se a porcentagem de piora da média do *fitness* dos cromossomos filhos gerados for maior ou igual a 10% do que os cromossomos pais, então a probabilidade de ocorrência do operador genético deve ser reduzida de

- 0.05 para operações de cruzamento e 0.005 para operações de mutação.
- Se a porcentagem de melhora/piora da média do *fitness* dos cromossomos filhos estiverem dentro de  $\pm 10\%$  da média do *fitness* dos cromossomos pais, então a probabilidade de ocorrência do operador genético não deve ser alterada.
  - As taxas devem estar entre 0.5 e 1.0 para o operador de cruzamento e entre 0.00 e 0.10 para o operador de mutação.
8. Verificar a condição de parada. Se a condição de parada é satisfeita, selecionar o melhor cromossomo da última geração como a solução final para o problema. Caso contrário, gerar a próxima população, onde a população antiga é substituída pela nova população. Feito isso, retornar a etapa 2.

#### **4.4 Considerações finais**

Neste capítulo, foram abordados a modelagem do problema com uso de um algoritmo genético adaptativo e o processo de programação da produção do veículo com a utilização de regras de despacho. Com isso, foi apresentado um método para a resolução do problema da programação da produção de produtos com uso simultâneo de máquina e sistemas de transporte em sistemas de manufatura com recursos compartilhados.

No capítulo 4 serão mostradas as etapas de desenvolvimento do sistema implementado e suas principais características, bem como a técnica utilizada para validação do código desenvolvido.

## **5 DESENVOLVIMENTO E TESTES DO SISTEMA PARA A PROGRAMAÇÃO DA PRODUÇÃO**

### **5.1 Considerações Iniciais**

Para avaliar os resultados esperados neste trabalho, um sistema para a programação da produção foi desenvolvido e as principais interfaces deste sistema serão apresentadas neste capítulo. Para garantir a qualidade do código do sistema desenvolvido, o mesmo passou por um processo de teste usando a ferramenta JUnit que será apresentada com mais detalhe neste capítulo.

### **5.2 Sistema Implementado**

O sistema foi desenvolvido usando a linguagem Java e o ambiente de desenvolvimento utilizado foi o Eclipse 3.2. A partir da Figura 18 é possível ver a tela inicial do sistema.



Figura 18 – Tela Inicial do sistema

Neste sistema é possível cadastrar um determinado cenário através do menu Cadastro, em que deve ser informado o tempo de operação, máquinas, produtos, sistemas de transportes e roteiros. Já através do menu Parâmetros AG, é possível informar quais os parâmetros iniciais para o Algoritmo Genético Adaptativo (AGA), tais parâmetros são: tamanho da população, taxas iniciais de cruzamento e mutação. Na Figura 19 é possível ver a partir do menu de Cadastro as opções para cadastro e na Figura 20 é possível ver a tela de inclusão dos parâmetros para o AGA.



Figura 19 – Menu Cadastro

Na Figura 21 é possível observar a tela de geração da programação. Através desta tela o usuário pode informar o número de vezes que este sistema será executado e ainda pode definir qual o formato da saída. As possíveis saídas são: planilha ou arquivo texto, ou ambos. Após o usuário definir o número de testes e um tipo de saída para os resultados, o mesmo deve pressionar o botão Executar para que inicie a execução da programação. Feito isso, é possível ver o melhor cromossomo e o seu *makespan* para cada teste durante a execução. Como um exemplo, foi definido um número de teste igual a 15 e foram selecionados os dois tipos de saídas, sendo que os formatos dessas saídas serão discutidos

posteriormente.

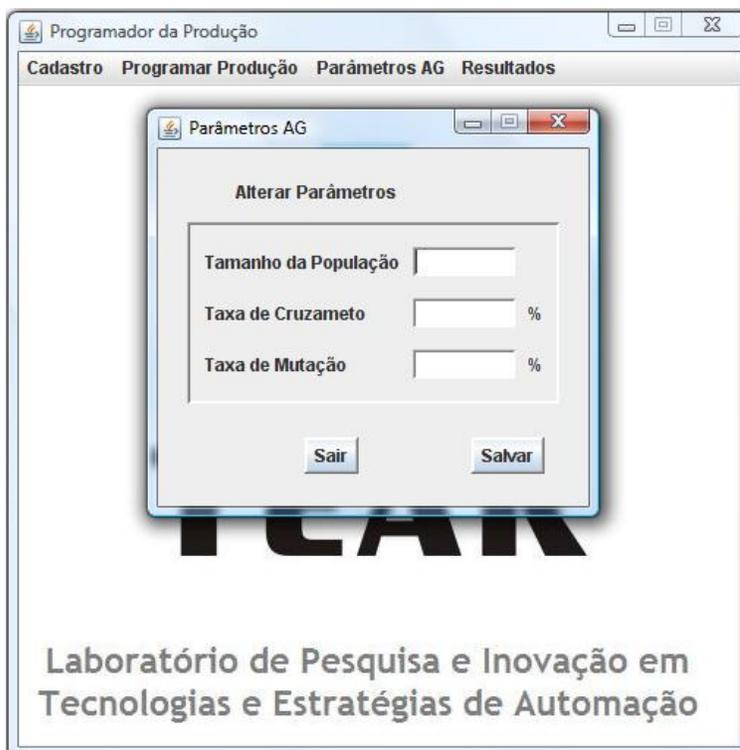


Figura 20 – Parâmetros do AGA

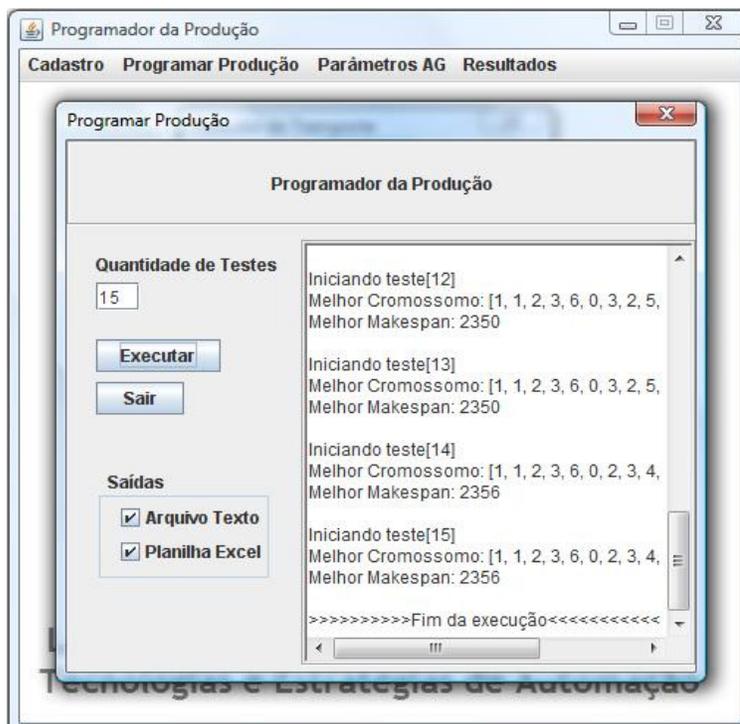


Figura 21 – Execução da Programação

A partir do menu Resultados é possível ver os resultados em dois tipos de formatos, arquivo texto ou planilha, estas opções encontram-se como submenus do menu Resultados. Quando selecionado o submenu Arquivo Texto, conforme ilustrado na Figura 22, são apresentadas informações referentes sobre o melhor cromossomo e *makespan* obtido em cada teste, o tempo de execução de cada teste em milissegundos e a programação da produção detalhada de acordo com o cromossomo obtido em um determinado teste, em que:

**P** - Número do Produto;

**M** - Número da Máquina e **C/D** - Estação de Carga e Descarga;

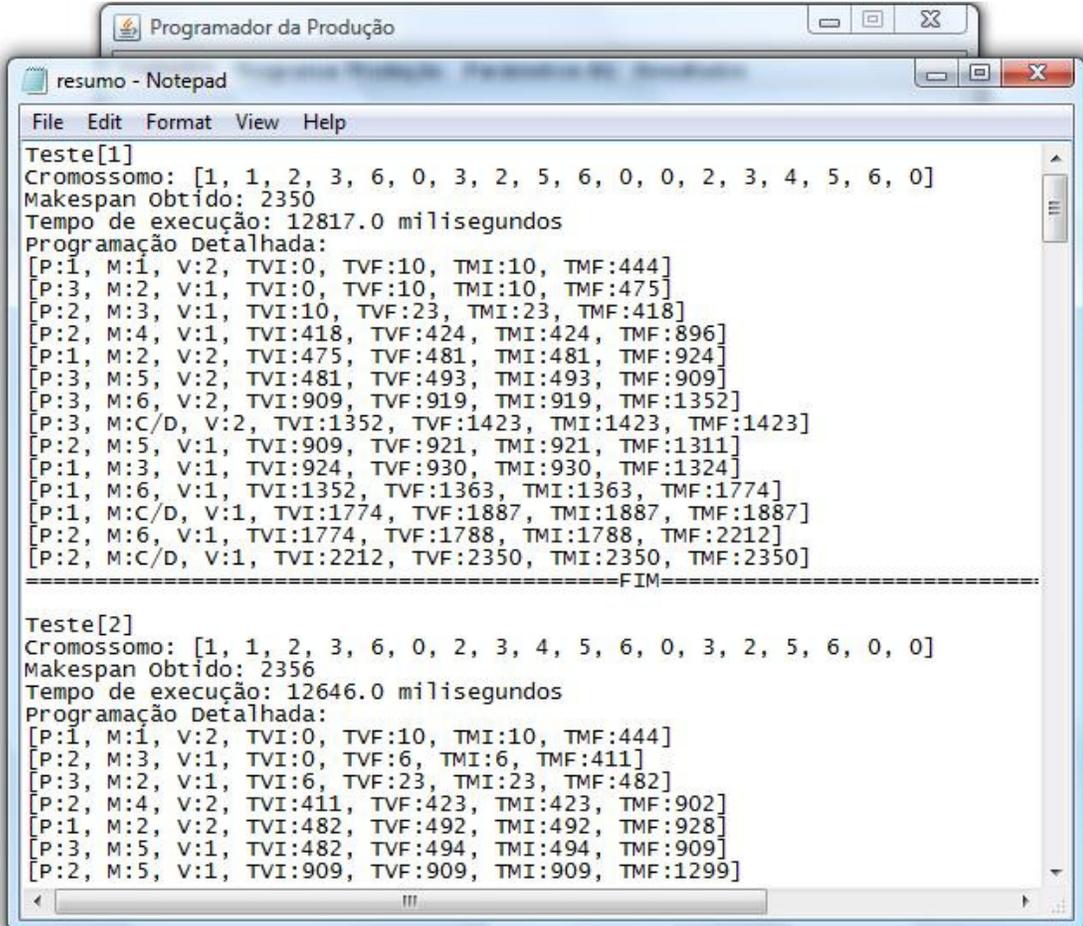
**V** - Número do Veículo;

**TVI** - Tempo de Utilização Inicial do Veículo;

**TVF** - Tempo de Utilização Final do Veículo;

**TMI** - Tempo de Utilização Inicial da Máquina;

**TMF** - Tempo de Utilização Final da Máquina.



```
Programador da Produção
resumo - Notepad
File Edit Format View Help
Teste[1]
Cromossomo: [1, 1, 2, 3, 6, 0, 3, 2, 5, 6, 0, 0, 2, 3, 4, 5, 6, 0]
Makespan Obtido: 2350
Tempo de execução: 12817.0 milissegundos
Programação Detalhada:
[P:1, M:1, V:2, TVI:0, TVF:10, TMI:10, TMF:444]
[P:3, M:2, V:1, TVI:0, TVF:10, TMI:10, TMF:475]
[P:2, M:3, V:1, TVI:10, TVF:23, TMI:23, TMF:418]
[P:2, M:4, V:1, TVI:418, TVF:424, TMI:424, TMF:896]
[P:1, M:2, V:2, TVI:475, TVF:481, TMI:481, TMF:924]
[P:3, M:5, V:2, TVI:481, TVF:493, TMI:493, TMF:909]
[P:3, M:6, V:2, TVI:909, TVF:919, TMI:919, TMF:1352]
[P:3, M:C/D, V:2, TVI:1352, TVF:1423, TMI:1423, TMF:1423]
[P:2, M:5, V:1, TVI:909, TVF:921, TMI:921, TMF:1311]
[P:1, M:3, V:1, TVI:924, TVF:930, TMI:930, TMF:1324]
[P:1, M:6, V:1, TVI:1352, TVF:1363, TMI:1363, TMF:1774]
[P:1, M:C/D, V:1, TVI:1774, TVF:1887, TMI:1887, TMF:1887]
[P:2, M:6, V:1, TVI:1774, TVF:1788, TMI:1788, TMF:2212]
[P:2, M:C/D, V:1, TVI:2212, TVF:2350, TMI:2350, TMF:2350]
=====FIM=====
Teste[2]
Cromossomo: [1, 1, 2, 3, 6, 0, 2, 3, 4, 5, 6, 0, 3, 2, 5, 6, 0, 0]
Makespan Obtido: 2356
Tempo de execução: 12646.0 milissegundos
Programação Detalhada:
[P:1, M:1, V:2, TVI:0, TVF:10, TMI:10, TMF:444]
[P:2, M:3, V:1, TVI:0, TVF:6, TMI:6, TMF:411]
[P:3, M:2, V:1, TVI:6, TVF:23, TMI:23, TMF:482]
[P:2, M:4, V:2, TVI:411, TVF:423, TMI:423, TMF:902]
[P:1, M:2, V:2, TVI:482, TVF:492, TMI:492, TMF:928]
[P:3, M:5, V:1, TVI:482, TVF:494, TMI:494, TMF:909]
[P:2, M:5, V:1, TVI:909, TVF:909, TMI:909, TMF:1299]
```

Figura 22 – Resultados armazenados em um arquivo texto

Já através do submenu Planilha, é possível observar os valores referentes ao *makespan* e o tempo de execução para cada teste, conforme ilustrado na Figura 23.

	Makespan	Tempo (ms)
1	2350	12817
2	2356	12646
3	2350	12432
4	2356	12466
5	2356	12285
6	2356	12371
7	2356	12550
8	2350	12474
9	2350	12498
10	2356	12445
11	2350	12438
12	2350	12774
13	2350	12425
14	2356	12326
15	2356	12329
16		
17		
18		

Figura 23 – Resultados armazenados em uma planilha

Tais formatos de arquivos foram necessários para a etapa de validação dos resultados, em que a partir da planilha gerada, foi possível tratar os dados e posteriormente realizar comparações entre as outras abordagens testadas. A partir do arquivo texto gerado, um gráfico de Gantt pode ser construído, já que são fornecidas as informações sobre quais produtos foram processados e em quais máquinas, segundo um determinado tempo.

### 5.3 Aplicação de testes sobre o sistema desenvolvido

Para garantir a qualidade do software, ele deve passar por uma seqüência de

testes. O objetivo do teste neste trabalho é garantir que a proposta implementada esteja livre de eventuais erros nas principais operações do sistema desenvolvido.

Para isso, foram aplicados testes em nível unitário, que segundo Lourida (2005), verifica a implementação de um módulo ou função do software. Assim, é possível garantir que a lógica do programa esteja completa e correta.

Como a atividade de executar testes é uma tarefa na maioria das vezes exaustiva, uma sugestão seria utilizar a automatização de teste. O uso de testes automatizados permite ao desenvolvedor verificar se mudanças no código fonte não se propagam para outras classes e outros requisitos. Para auxiliar no uso de testes automatizados neste trabalho, a ferramenta JUnit na versão 4.4 foi utilizada.

JUnit foi desenvolvido por Erich Gamma e Kent Bech, onde esta ferramenta possibilita a criação de testes automatizados utilizando a linguagem Java.

Portanto, o JUnit foi utilizado neste trabalho para validar as seguintes e mais importantes etapas do sistema:

- Função *Fitness*
- Operação de Cruzamento
- Operação de Mutação

Para verificar o sucesso na aplicação do teste, o JUnit utiliza *assertions*, que são métodos embutidos dentro da ferramenta e que verificam se o resultado correto é o mesmo obtido após a execução do teste. Se os resultados forem idênticos, o módulo testado está livre de erros; caso contrário, é falho (Louridas, 2005).

Segundo Louridas (2005), os testes realizados individualmente são chamados de Casos de Teste (*Test cases*), mas estes testes podem ser agrupados formando um Grupo de Teste (*Test suite*). Neste contexto, este trabalho fez uso do *Test suite* para testar as classes e métodos referentes ao cálculo da função de *fitness*, operação de cruzamento e mutação. Para este teste, foram informadas as saídas desejadas para cada etapa testada e as comparações entre os resultados foram realizadas através das *assertions* discutidas anteriormente.

Na Figura 24 é possível observar a tela de exibição do resultado após execução do JUnit no ambiente Eclipse. Nesta tela os campos *Errors* e *Failures* estão com valores iguais a zero, indicando que nenhum erro é encontrado após a execução do teste. Ainda nesta tela, uma tarja que está indicada por uma seta é apresentada. Esta tarja, quando possui cor verde, indica que o teste foi bem sucedido, mas quando a cor é vermelha, indica que existe algum erro no código, sendo que mais detalhes sobre este erro são apresentados no campo

“Failure Trace”, na parte inferior esquerda da tela.

## 5.4 Considerações finais

Neste capítulo foram apresentadas as principais telas do sistema implementado para a programação de produtos com uso simultâneo de máquinas e sistemas de transporte. Outro ponto observado foi como os testes foram aplicados a este sistema através da ferramenta JUnit.

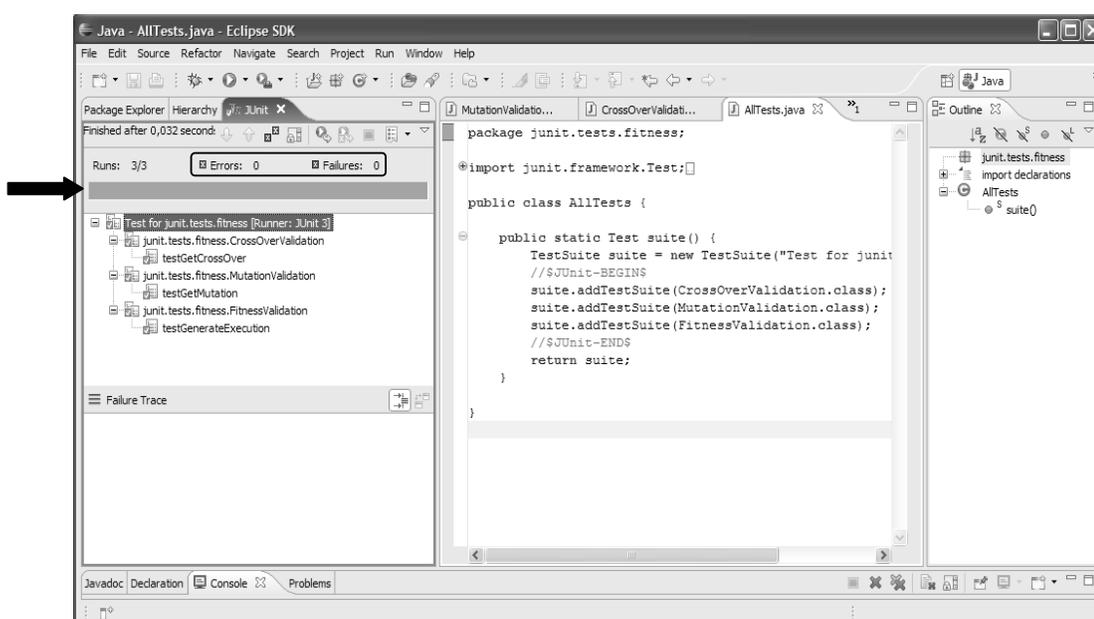


Figura 24 – Resultado após a execução do teste com o JUnit

Os resultados e as respectivas discussões sobre o método proposto em relação as outras propostas comparadas serão discutidas no capítulo 6, onde serão realizadas algumas análises sobre os resultados de *makespan*, tempo de resposta da execução e o desempenho do algoritmo genético adaptativo na busca.

## 6 RESULTADOS E DISCUSSÕES

### 6.1 Considerações Iniciais

Neste capítulo, serão apresentados os resultados obtidos a partir da proposta deste trabalho. Serão realizadas algumas análises sobre os resultados de *makespan*, tempo de resposta da execução e o desempenho do algoritmo genético adaptativo na busca. Os resultados destas análises serão comparados com os resultados de outras abordagens testadas neste trabalho.

Experimentos foram realizados e os resultados encontrados foram comparados com a abordagem proposta por Reddy e Rao (2006) e Morandin et al. (2008), pois ambos utilizam Algoritmo Genético como método de modelagem e busca para o problema de programação. No corrente trabalho, a abordagem proposta foi apelidada de “Proposta” enquanto a abordagem proposta por Reddy e Rao (2006) foi apelidada de “R&R” e a abordagem proposta por Morandin et al. (2008) de “Morandin”.

Dois tipos de cenários foram utilizados para a validação do método proposto e, para cada cenário, foi definida uma quantidade de 35 testes, que foram submetidos a testes estatísticos. Devido às características não determinístico da busca com o algoritmo genético adaptativo, os resultados variam em cada teste.

Como os dados obtidos nos testes não pertenceram a uma distribuição normal, pôde-se aplicar um teste não-paramétrico para a análise dos resultados obtidos. Dentre os testes possíveis, foi utilizado o teste de Wilcoxon (Martins, 2005) para verificar o grau de confiança dos resultados obtidos em relação às outras abordagens testadas.

No teste de Wilcoxon, uma amostra A1 é submetida a um tratamento T1 e tem seu efeito medido. Posteriormente, essa mesma amostra, chamada agora de A2 é submetida a um segundo tratamento T2, e tem seu efeito medido pela mesma variável usada no primeiro tratamento. Comparando-se o efeito dos dois tratamentos em cada elemento da amostra, pode-se verificar se o efeito aumentou, diminuiu ou permaneceu o mesmo. Através do teste de Wilcoxon, pode-se analisar se os resultados obtidos pelos dois tratamentos aplicados à mesma amostra têm diferenças estatisticamente relevantes, levando-se em conta a magnitude do aumento ou da diminuição dos resultados e não apenas a direção da variação para mais ou para menos (Campos, 2008).

Portanto, o teste de Wilcoxon foi utilizado para comparar os resultados de *makespan* obtidos neste trabalho em relação à abordagem proposta por Reddy e Rao (2006), com o objetivo de verificar se tais valores são menores na maioria dos casos comparados, visto que a abordagem proposta por Reddy e Rao (2006) procura minimizar mais de um objetivo. O teste de Wilcoxon não foi aplicado sobre a abordagem proposta por Morandin et al. (2008), pois esta abordagem não leva em consideração a programação da produção do sistema de transporte e, com isso, terá valores de *makespan* mais baixos na maioria dos casos. Neste caso, uma análise comparativa foi realizada com o objetivo de verificar se os resultados não ficaram muito distantes quando comparado com Morandin et al. (2008).

## 6.2 Caracterização do ambiente de testes

Cenários gerados aleatoriamente e de dois tamanhos diferentes foram avaliados em relação à quantidade de produtos e máquinas. Os roteiros de fabricação dos produtos foram gerados aleatoriamente, assim como os tempos de operação, que são determinísticos, sendo que, uma vez gerados, permaneceram fixos.

Para cada problema testado, foi considerada a programação de um produto de cada tipo, o que pode corresponder também a um lote de produtos de tamanho padrão, se for considerado que os tempos de operação de cada produto em cada máquina também correspondem aos tempos de um lote de tamanho padrão do produto.

## 6.3 Testes realizados

O primeiro problema refere-se a um cenário que contém seis máquinas e três produtos, em que o produto 1 possui dois possíveis roteiros de fabricação e os demais produtos possuem três possíveis roteiros de fabricação contendo três a cinco máquinas cada um (Tabela 2). Os roteiros de fabricação dos produtos foram gerados aleatoriamente, assim como os tempos de operação, que variaram entre 400 e 500 unidades de tempo (u.t.) (Tabela 3).

Além disso, foram calculados os tempos de transporte para o cenário do problema 1, que são os tempos que o veículo leva para alcançar uma máquina ou estação de carga/descarga a partir da sua posição atual (Tabela 4).

O segundo problema testado refere-se a um cenário que contém nove máquinas

e nove produtos, em que cada produto possui dois possíveis roteiros de fabricação contendo cinco a sete máquinas cada um (Tabela 5). Os roteiros de fabricação dos produtos foram gerados aleatoriamente, assim como os tempos de operação, que variaram entre 400 e 500 unidades de tempo (u.t.) (Tabela 6). Os tempos de transporte podem ser vistos na Tabela 7. Em ambos os problemas, foram utilizados dois veículos.

Tabela 2: Roteiros do Problema 1

<b>Produtos</b>	<b>Roteiros</b>	
<b>P<sub>1</sub></b>	R <sub>11</sub>	M <sub>1</sub> M <sub>2</sub> M <sub>3</sub> M <sub>4</sub> M <sub>5</sub>
	R <sub>12</sub>	M <sub>1</sub> M <sub>2</sub> M <sub>3</sub> M <sub>6</sub>
<b>P<sub>2</sub></b>	R <sub>21</sub>	M <sub>1</sub> M <sub>4</sub> M <sub>5</sub> M <sub>6</sub>
	R <sub>22</sub>	M <sub>2</sub> M <sub>4</sub> M <sub>5</sub> M <sub>6</sub>
	R <sub>23</sub>	M <sub>3</sub> M <sub>4</sub> M <sub>5</sub> M <sub>6</sub>
<b>P<sub>3</sub></b>	R <sub>31</sub>	M <sub>1</sub> M <sub>5</sub> M <sub>6</sub>
	R <sub>32</sub>	M <sub>2</sub> M <sub>5</sub> M <sub>6</sub>
	R <sub>33</sub>	M <sub>3</sub> M <sub>4</sub> M <sub>5</sub> M <sub>6</sub>

Tabela 3: Tempos de Operação do Problema 1

<b>Máquinas /Produtos</b>	<b>P<sub>1</sub></b>	<b>P<sub>2</sub></b>	<b>P<sub>3</sub></b>
<b>M<sub>1</sub></b>	434	458	472
<b>M<sub>2</sub></b>	452	443	465
<b>M<sub>3</sub></b>	400	405	469
<b>M<sub>4</sub></b>	472	485	459
<b>M<sub>5</sub></b>	460	402	432
<b>M<sub>6</sub></b>	421	435	445

Tabela 4: Tempos de Transporte do Problema 1

	Carga (C)	M1	M2	M3	M4	M5	M6	Descarga (D)
C	0	4	6	8	14	12	10	6
M1	10	0	3	5	11	9	7	14
M2	12	15	0	3	9	7	9	5
M3	14	17	15	0	7	9	11	8
M4	8	11	9	7	0	3	5	10
M5	6	9	7	9	15	0	3	7
M6	4	7	9	11	17	15	0	14
D	4	7	9	11	17	15	0	0

Tabela 5: Roteiros do Problema 2

Produtos	Roteiros	
<b>P1</b>	R11	M1 M2 M4 M5 M7 M9
	R12	M3 M4 M5 M6 M8 M9
<b>P2</b>	R21	M1 M2 M3 M4 M5 M6 M7
	R22	M2 M3 M5 M7 M8 M9
<b>P3</b>	R31	M4 M5 M6 M7 M8
	R32	M2 M3 M7 M8 M9
<b>P4</b>	R41	M2 M3 M4 M6 M7
	R42	M1 M5 M6 M8 M9
<b>P5</b>	R51	M4 M5 M7 M8 M9
	R52	M1 M2 M3 M5 M6
<b>P6</b>	R61	M2 M4 M5 M6 M7 M8 M9
	R62	M1 M3 M6 M7 M8 M9
<b>P7</b>	R71	M1 M2 M4 M5 M6 M9
	R72	M1 M2 M3 M7 M8 M9
<b>P8</b>	R81	M4 M5 M6 M7 M8 M9
	R82	M3 M4 M5 M7 M8 M9
<b>P9</b>	R91	M3 M5 M6 M7 M8 M9
	R92	M2 M4 M6 M7 M8 M9

Tabela 6: Tempos de Operação do Problema 2

Máquinas /Produtos	P <sub>1</sub>	P <sub>2</sub>	P <sub>3</sub>	P <sub>4</sub>	P <sub>5</sub>	P <sub>6</sub>	P <sub>7</sub>	P <sub>8</sub>	P <sub>9</sub>
M <sub>1</sub>	428	439	453	403	481	446	414	491	458
M <sub>2</sub>	423	433	474	436	440	495	457	419	486
M <sub>3</sub>	459	487	417	410	477	474	452	435	416
M <sub>4</sub>	433	405	447	410	442	448	426	491	454
M <sub>5</sub>	467	447	486	400	450	469	493	495	452
M <sub>6</sub>	461	497	496	468	468	408	408	452	438
M <sub>7</sub>	464	495	459	489	436	454	457	477	484
M <sub>8</sub>	455	469	489	439	486	424	497	452	435
M <sub>9</sub>	418	439	480	457	435	482	445	408	416

Tabela 7: Tempos de Transporte do Problema 2

	C	M1	M2	M3	M4	M5	M6	M7	M8	M9	D
C	0	14	5	10	12	14	10	13	11	9	13
M1	11	0	11	6	5	6	5	9	11	6	11
M2	6	9	0	7	5	6	10	6	9	11	13
M3	10	7	5	0	11	5	10	12	14	8	6
M4	7	13	8	9	0	10	11	6	14	7	9
M5	9	10	8	11	10	0	7	13	9	12	9
M6	14	11	13	7	14	6	0	12	13	14	8
M7	10	7	9	13	5	8	10	0	8	10	13
M8	11	6	9	6	8	10	12	13	0	9	7
M9	6	5	6	8	9	10	9	5	4	0	8
D	6	5	11	8	10	11	8	5	14	13	0

## 6.4 Resultados obtidos

Para a realização dos testes com o método proposto, foram utilizados os seguintes valores para os parâmetros do algoritmo genético adaptativo: tamanho da população igual a 30, taxa de cruzamento iniciando em 0,6 (60%) e taxa de mutação iniciando em 0,005 (0.5%), com os valores de taxa de cruzamento entre 0.5 e 1.0 e taxa de mutação entre 0.005 e 0.1. O critério de parada utilizado foi de 100 gerações ou quando o algoritmo converge.

Na abordagem proposta por Morandin et al. (2008) foram utilizados os seguintes valores: tamanho da população = 40, taxa de cruzamento 0,8 (80%) e taxa de mutação 0,2 (20%) enquanto na abordagem proposta por Reddy e Rao (2006) foram definidos os valores como: tamanho da população = 200, taxa de cruzamento = 0,8 (80%) e taxa de

mutação = 0,4 (40%).

Nas Tabelas 8 e 9 podem ser vistos os resultados dos testes para cada problema. As colunas indicam o número do teste (de 1 a 35), o *makespan* obtido e o tempo de resposta, calculados em minutos para cada teste, respectivamente. As próximas colunas representam os valores de *makespan* obtidos nos demais métodos testados e seus tempos de resposta, respectivamente. Os valores de *makespan* foram contados em unidades de tempo (u.t.) e os tempos de execução em segundos.

Tabela 8: Resultados obtidos para o Problema 1.

#	Proposta <i>Makespan</i>	Proposta Tempo(min)	R&R <i>Makespan</i>	R&R Tempo(min)	Morandin <i>Makespan</i>	Morandin Tempo(min)
1	2356	1,13	2373	1,04	2198	0,023
2	2350	1,12	2373	0,99	2198	0,018
3	2350	1,13	2373	1,07	2198	0,018
4	2350	1,15	2373	1,09	2228	0,018
5	2350	1,17	2373	0,98	2198	0,018
6	2383	1,31	2373	0,96	2198	0,018
7	2356	1,17	2373	0,99	2219	0,018
8	2350	1,20	2373	0,99	2198	0,018
9	2350	1,23	2373	1,01	2198	0,017
10	2356	1,21	2373	0,98	2198	0,018
11	2348	1,25	2373	0,97	2219	0,017
12	2349	1,27	2373	0,98	2198	0,017
13	2350	1,28	2373	0,97	2219	0,017
14	2356	1,29	2373	0,99	2198	0,018
15	2350	1,36	2373	1,05	2250	0,019
16	2350	1,35	2373	0,99	2198	0,017
17	2350	1,34	2373	1,00	2198	0,017
18	2356	1,37	2373	1,00	2198	0,017
19	2350	1,41	2373	1,02	2214	0,017
20	2350	1,26	2373	1,02	2198	0,017
21	2350	1,43	2373	1,00	2198	0,017
22	2350	1,47	2373	1,01	2198	0,017
23	2356	1,47	2373	1,01	2219	0,017
24	2378	0,56	2373	1,02	2198	0,017
25	2356	1,53	2373	1,04	2198	0,018
26	2350	1,54	2373	1,03	2198	0,018
27	2356	1,64	2373	1,04	2198	0,018
28	2350	1,81	2373	1,03	2198	0,017
29	2350	1,80	2373	1,04	2214	0,017
30	2356	1,63	2373	1,06	2198	0,017
31	2356	1,60	2373	1,04	2198	0,018
32	2350	1,90	2373	1,08	2198	0,017
33	2350	1,69	2373	1,08	2198	0,017
34	2349	1,69	2373	1,09	2219	0,017
35	2356	1,86	2373	1,11	2198	0,017
<b>Média</b>	<b>2354</b>	<b>1,39</b>	<b>2373</b>	<b>1,02</b>	<b>2204</b>	<b>0,018</b>

A média do *makespan* encontrada para o método proposto foi 2354 u.t., com

desvio padrão de 7,3. O valor mínimo encontrado foi 2348 u.t. e o valor máximo foi 2383 u.t. Observou-se que houve tendência de melhora em 94% dos resultados obtidos com relação aos resultados obtidos por R&R. Devido à utilização de uma população com um tamanho grande, neste caso composta de 200 indivíduos, e um baixo número de soluções possíveis no cenário testado, a abordagem proposta por R&R apresentou o mesmo *makespan* nos 35 testes. Portanto, a partir da comparação destes valores de *makespan*, conclui-se que o método proposto apresentou resultados melhores e diferentes estatisticamente que os resultados obtidos por R&R, com 95% de confiança segundo o teste de Wilcoxon.

Quando comparado com os valores obtidos por Morandin observou-se que a média obtida foi superior em 6,81%, enquanto que a distância máxima em relação ao menor valor de *makespan* encontrado, não foi maior que 8,42%, se distanciando em média 7,10% do mínimo *makespan* encontrado nos testes.

Com relação ao tempo de execução do método proposto, sua média foi 1,39 minutos, que é muito próximo ao tempo de execução de R&R. O tempo gerado (0,018 minutos) ficou abaixo quando comparado com tempo de execução de Morandin, devido a não utilização da programação da produção do sistema de transporte.

Tabela 9: Resultados obtidos para o Problema 2.

#	Proposta <i>Makespan</i>	Proposta Tempo(min)	R&R <i>Makespan</i>	R&R Tempo(min)	Morandin <i>Makespan</i>	Morandin Tempo(min)
1	5342	2,14	5677	4,37	4755	0,135
2	5779	2,12	5693	4,37	4975	0,136
3	5342	1,89	5693	4,39	5083	0,133
4	5573	1,87	5677	4,37	4901	0,141
5	5579	1,87	5693	4,39	4972	0,145
6	5593	2,05	5691	4,33	4676	0,148
7	5563	1,81	5677	4,32	5096	0,150
8	5585	2,09	5692	4,31	5187	0,159
9	5755	1,86	5677	4,34	5147	0,141
10	5681	1,86	5689	4,34	5123	0,160
11	5738	2,10	5688	4,30	4901	0,142
12	5596	1,83	5685	4,34	4990	0,146
13	5594	1,86	5677	4,31	4973	0,151
14	5767	1,91	5680	4,32	5175	0,141
15	5402	1,86	5691	4,33	4673	0,138
16	5743	1,91	5677	4,32	4975	0,135
17	5726	2,01	5677	4,31	4754	0,142
18	5349	2,02	5689	4,39	5082	0,160
19	5455	1,90	5677	4,32	5121	0,153
20	5556	1,99	5677	4,31	4676	0,135
21	5595	2,04	5677	4,30	5082	0,132
22	5360	1,86	5692	4,29	5082	0,129
23	5358	2,13	5681	4,32	4942	0,127
24	5563	1,86	5677	4,32	5170	0,136

25	5750	1,85	5693	4,32	5502	0,135
26	5346	1,92	5688	4,30	5181	0,134
27	5663	1,37	5696	4,32	5003	0,130
28	5566	2,08	5689	4,34	5188	0,135
29	5725	1,86	5677	4,33	5082	0,134
30	5420	1,87	5703	4,33	5082	0,130
31	5580	1,34	5696	4,33	5302	0,137
32	5358	1,97	5680	4,32	4983	0,135
33	5524	2,05	5696	4,34	5115	0,134
34	5766	1,38	5693	4,32	5119	0,132
35	5739	1,81	5693	4,32	5518	0,136
<b>Média</b>	<b>5572</b>	<b>1,89</b>	<b>5686</b>	<b>4,33</b>	<b>5054</b>	<b>0,140</b>

Em relação aos resultados apresentados na Tabela 9, a média do *makespan* encontrada para o método proposto foi 5572 u.t., com desvio padrão de 148,48. O valor mínimo encontrado foi 5342 u.t. e o valor máximo encontrado foi 5779 u.t. Para este problema houve tendência de melhora em 71% dos resultados obtidos com relação aos resultados obtidos por R&R. Portanto, a partir da comparação destes valores de *makespan*, conclui-se que o método proposto apresentou resultados melhores e diferentes estatisticamente que os resultados obtidos por R&R, com 95% de confiança segundo o teste de Wilcoxon.

Quando comparado com os valores obtidos por Morandin observou-se que a média obtida foi superior em 10,45%, enquanto a distância máxima em relação ao menor valor de *makespan* encontrado, não foi maior que 23,67%, se distanciando em média 19,24% do mínimo *makespan* encontrado nos testes.

Com relação ao tempo de execução do método proposto, a média foi de 1,90 minutos, quase 2,5 vezes menor que o tempo de execução de R&R, que foi de 4,33 minutos. A partir dos resultados obtidos para o problema 2, pôde-se observar que a média do tempo de execução obtida aumentou somente 45,03% enquanto a média do tempo de execução obtida por R&R aumentou em 324,51% quando comparado com os valores obtidos no problema 1. Já os valores de tempo de execução obtidos por Morandin foram novamente baixos. Isto ocorre devido à não utilização da programação da produção do sistema transporte conforme discutido anteriormente.

#### 6.4.1 Desempenho do Algoritmo Genético Adaptativo

Com a finalidade de verificar o comportamento do algoritmo genético adaptativo durante as gerações em busca de uma boa solução, serão apresentados os gráficos

de convergência referente ao algoritmo genético adaptativo utilizado neste trabalho, e o algoritmo genético tradicional utilizado por R&R e Morandin. Neste tipo de gráfico, foram comparados o melhor valor de *fitness* e a média do *fitness* da população durante 100 gerações. Estes valores são referentes aos *makespans* obtidos para um determinado teste obtido em cada um dos problemas testados (Tabela 8 e 9).

Nas Figuras 25 e 26, são apresentados os desempenhos do algoritmo genético adaptativo utilizados neste trabalho. É possível observar que através do ajuste dinâmico das taxas de cruzamento e mutação, o algoritmo genético adaptativo explora outros pontos do espaço de busca convergindo para uma região do espaço da busca que contém soluções melhores. Este processo é apresentado na Figura 26, em que um cenário grande foi testado (problema 2) possuindo um grande número de soluções.

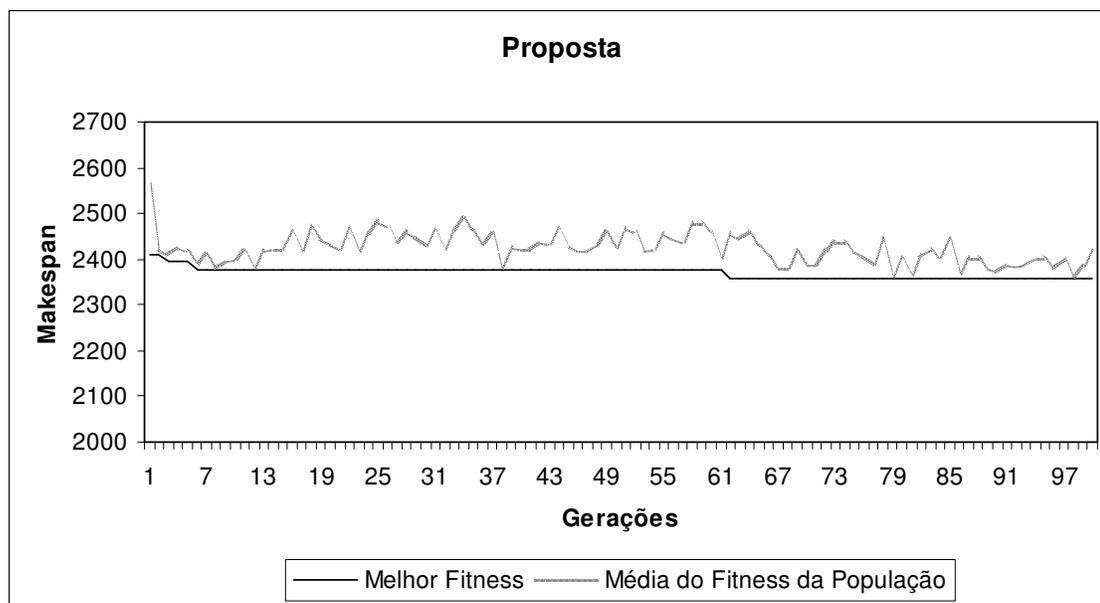


Figura 25: Gráfico de Convergência da Proposta para o Problema 1

Nas Figuras 27 e 28, são apresentados os desempenhos do algoritmo genético utilizado por R&R, onde é possível observar que, a partir da utilização de uma população com um tamanho grande (neste caso com 200 indivíduos), o algoritmo genético demora para alcançar uma região com soluções melhores dentro do espaço de busca. Outro ponto a ser observado é que quanto maior o tamanho do problema maior será o tempo de execução deste algoritmo e maior será a demora para o algoritmo genético alcançar melhores regiões no espaço de busca.

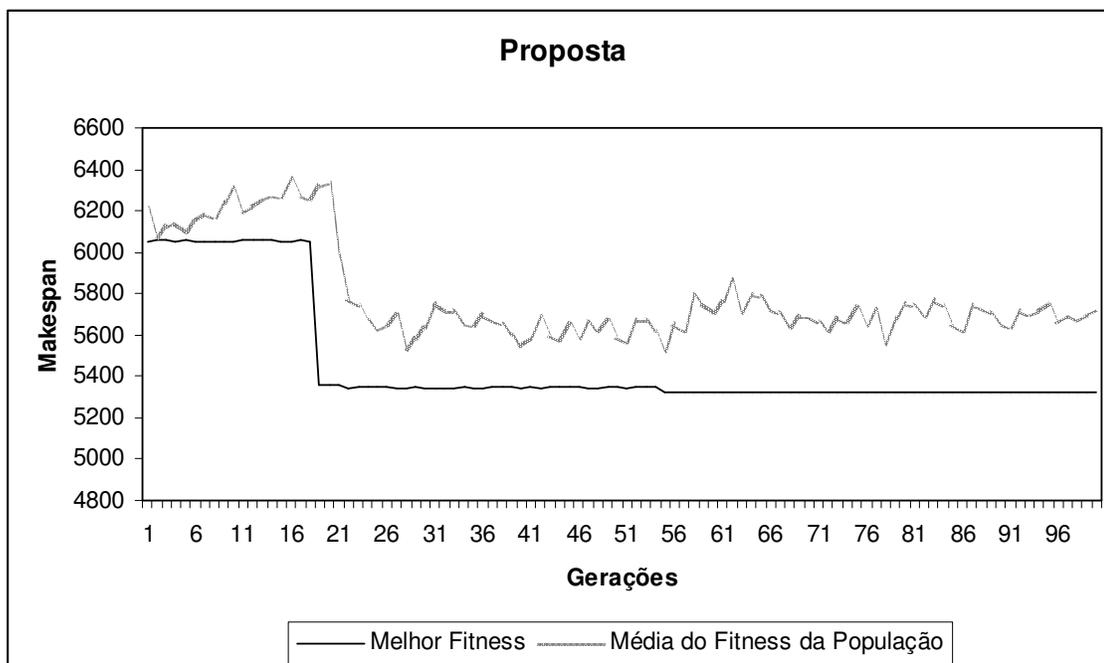


Figura 26: Gráfico de Convergência da Proposta para o Problema 2

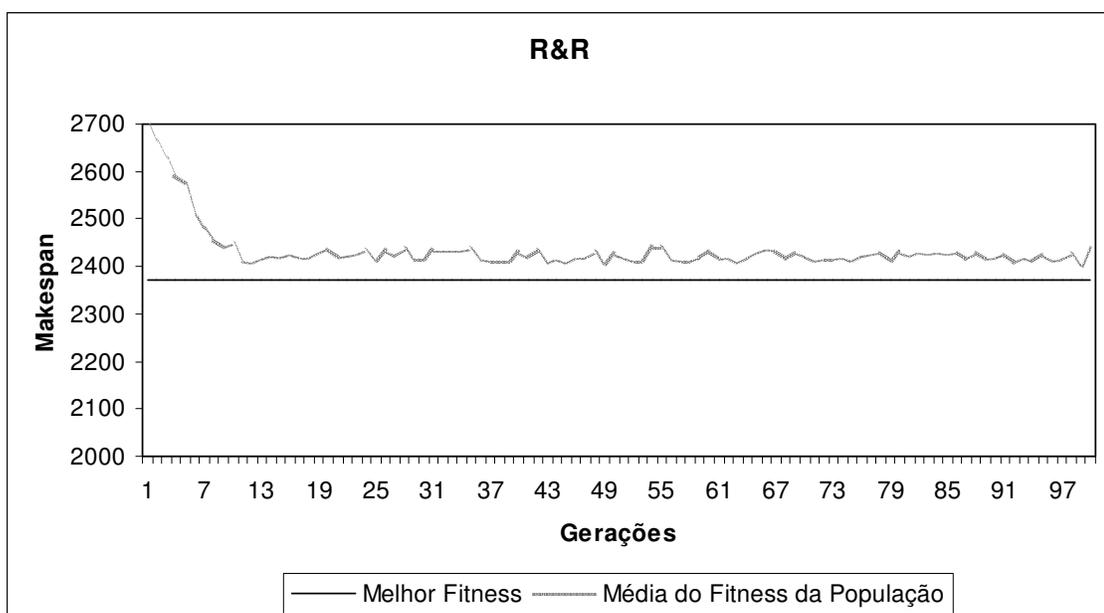


Figura 27: Gráfico de Convergência de R&R para o Problema 1

Nas Figuras 29 e 30, são apresentados os desempenhos do algoritmo genético utilizado por Morandin, em que é possível observar que a melhor solução durante a busca é perdida e as variações, tanto do melhor valor de *fitness* como as médias de *fitness* das populações, são muito grandes. Isso ocorre porque o elitismo utilizado nesta abordagem copia

a melhor solução da geração para a população intermediária e permite que a melhor solução passe pelo processo de cruzamento e mutação; com isso, esta boa solução pode ser perdida.

A partir dos resultados apresentados neste capítulo, é possível observar que a abordagem proposta consegue obter um melhor *makespan* na maioria dos casos testados quando comparado com os resultados obtidos por R&R. Quando comparado com os resultados obtidos por Morandin, é possível observar que o *makespan* obtido não ficou muito distante, visto que tal abordagem não leva em consideração a programação da produção dos sistemas de transporte.

A partir da comparação dos tempos de execução obtidos, é possível observar que esta proposta pode ser aplicada tanto para problemas de programação com cenários pequenos (problema 1) como para cenários grandes (problema 2), e atinge um bom compromisso de *makespan* e um tempo de resposta baixo, que é uma característica muito importante quanto se trata de uma programação reativa.

Outro ponto a ser observado, é que a busca com algoritmo genético adaptativo obteve um bom desempenho sobre o problema abordado e, a partir do ajuste dinâmico das taxas de cruzamento e mutação, o algoritmo genético explorou várias regiões do espaço de busca convergindo para regiões que possuem melhores soluções.

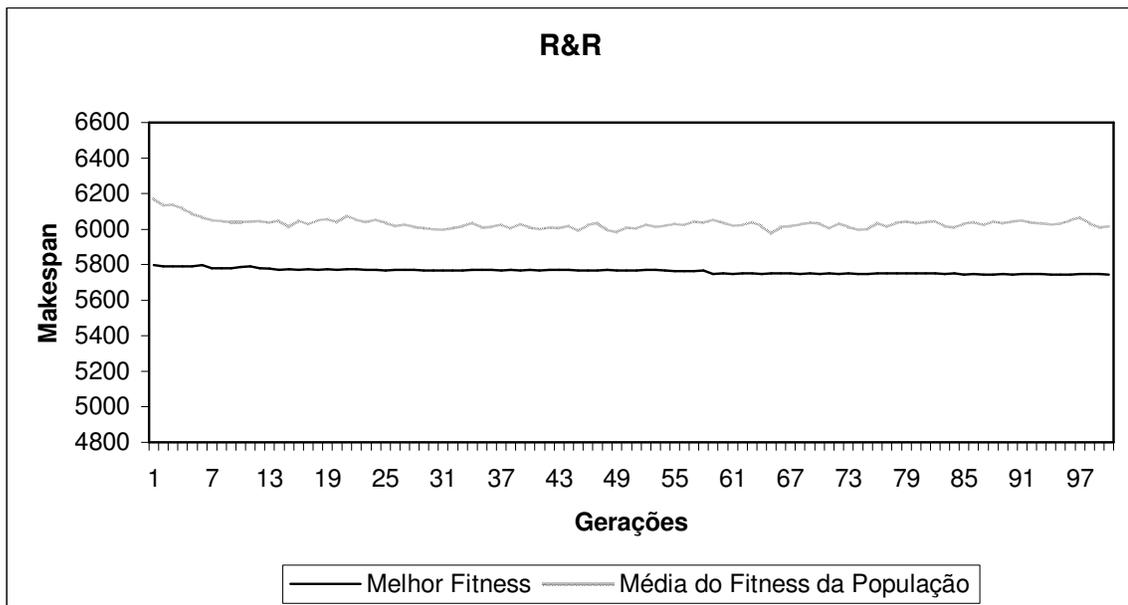


Figura 28: Gráfico de Convergência de R&R para o Problema 2

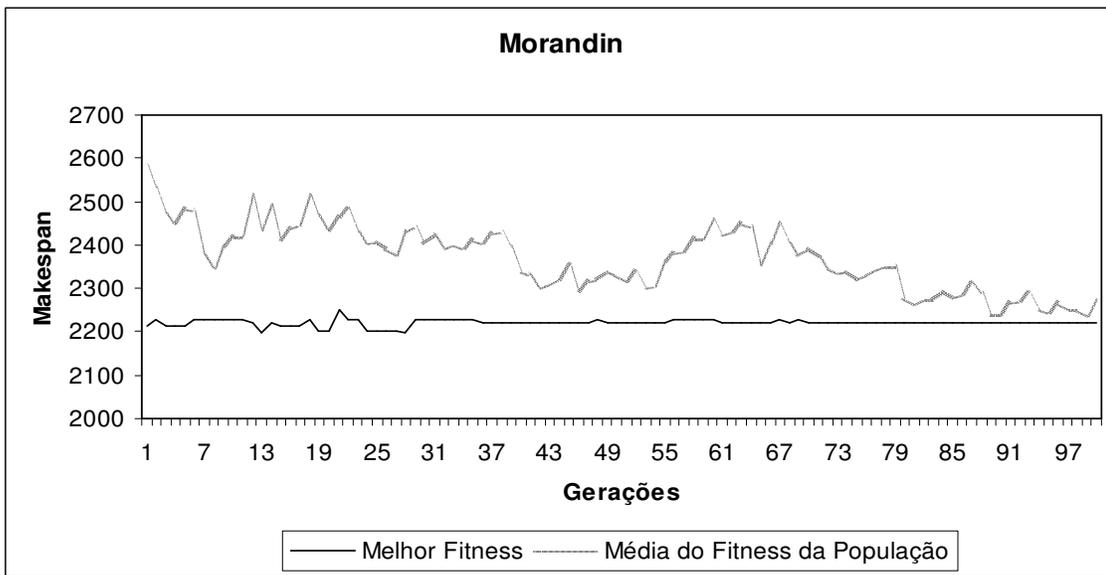


Figura 29: Gráfico de Convergência de Morandin para o Problema 1

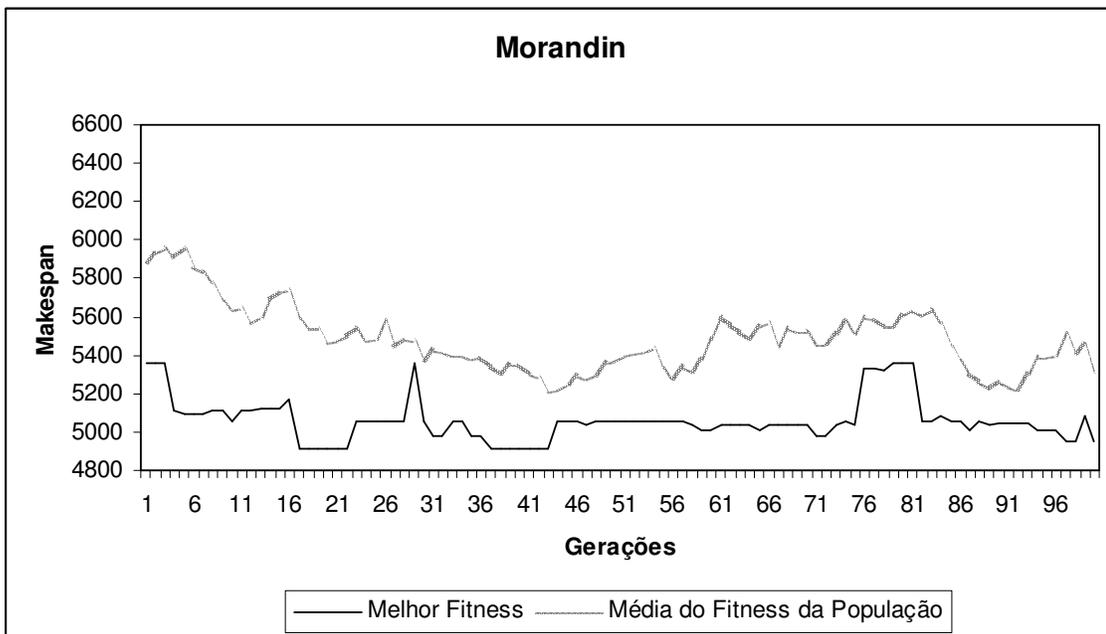


Figura 30: Gráfico de Convergência de Morandin para o Problema 2

### 6.5 Considerações finais

Neste capítulo, os problemas (cenários) foram definidos e testados para a avaliação do método proposto e os resultados foram comparados com outras abordagens propostas por Reddy e Rao (2006) e Morandin et al. (2008), tendo como critério de

desempenho o valor de *makespan* e o tempo de execução obtido.

As conclusões sobre o método proposto, os testes realizados e possíveis alterações sobre o algoritmo genético adaptativo utilizado, com o objetivo melhorar seu desempenho e suas soluções, serão propostas e discutidas no capítulo 7, assim como indicações para trabalhos futuros.

## 7 CONCLUSÃO

A programação de produtos com uso simultâneo de máquinas e sistemas de transporte é caracterizada pela grande quantidade de soluções possíveis, em especial para problemas de grande escala, que possuem um número considerável de máquinas e produtos para a produção. Por isso, as características dos algoritmos genéticos se tornam apropriados para tal tarefa, uma vez que estes algoritmos possuem a capacidade de percorrer de forma global os espaços da busca com as soluções possíveis para o problema proposto, e assim, encontram uma solução ótima ou quase ótima rapidamente.

O principal objetivo deste trabalho foi propor um método com um algoritmo genético adaptativo para gerar programação de produtos com uso simultâneo de máquinas e sistemas de transporte em sistemas de manufatura com recursos compartilhados.

O método foi testado para uma série de problemas e os resultados foram comparados com outras abordagens propostas por Morandin et al. (2008) e Reddy e Rao (2006), cujo critério de desempenho foi o mínimo *makespan* e o tempo de resposta obtido.

Os resultados foram obtidos a partir de um sistema que foi implementado para facilitar a captura dos resultados e permitir um acompanhamento visual em tempo de execução para cada teste realizado.

A partir dos resultados obtidos, é possível observar que a abordagem proposta consegue obter um melhor *makespan* na maioria dos casos testados quando comparada com os resultados obtidos por Reddy e Rao (2006), já que esta abordagem trata de um problema multi-objetivo. Quando comparada com os resultados obtidos por Morandin et al. (2008), os resultados não ficaram muito distantes, pois não leva em consideração os tempos de transporte e em consequência disto, seus resultados de *makespan* são na maioria dos casos menores. Dessa forma, a proposta alcançou o objetivo esperado.

Os tempos de obtenção das respostas apresentaram-se baixos após a execução para os dois tipos de cenários testados, um pequeno e outro grande. Esta característica é muito importante quando se trata de uma programação reativa.

Quando se trata da busca com algoritmo genético, o comportamento da busca é influenciado ou controlado por alguns parâmetros, como o tamanho da população e as taxas de cruzamento e de mutação, os quais podem melhorar ou piorar o desempenho da busca. Por isso, este trabalho utilizou um algoritmo genético adaptativo com o ajuste dinâmico das taxas

de cruzamento e mutação.

A partir do uso deste algoritmo genético adaptativo pôde-se obter um bom desempenho sobre o problema abordado, além de explorar outros pontos do espaço de busca pelo ajuste dinâmico das taxas de cruzamento e mutação que levou a convergência para uma região com boas soluções.

Com base nestes resultados, observa-se que a abordagem proposta neste trabalho pode ser aplicada para a solução de problemas de manufatura com compartilhamento de recursos de pequena e grande escala, a fim de atingir um bom compromisso entre os valores de *makespan* e um baixo tempo de obtenção da resposta.

## 7.1 Trabalhos Futuros

Com base nos resultados obtidos, sugere-se a inclusão de outro método de busca para trabalhar juntamente com o algoritmo genético adaptativo, com o objetivo de melhorar o desempenho da busca. Neste caso, pode ser incluso o método de busca Tabu, que é apresentado em vários trabalhos com seu uso agregado ao algoritmo genético.

Outra sugestão é a inclusão de um sistema *fuzzy* para trabalhar juntamente com as regras de despacho utilizadas para a programação do sistema de transporte. Neste caso, outras regras de despacho deveriam ser testadas a fim de obter um melhor desempenho para a programação do sistema de transporte e, com isso, alcançar melhores valores de *makespan*.

Alteração da função objetivo para outros parâmetros utilizados como medida de desempenho nas manufaturas, como a taxa de utilização de máquinas e cumprimento de data devida podem ser realizadas. Além disso, pode-se utilizar o algoritmo genético para mais de um objetivo.

Em relação ao algoritmo genético adaptativo, este pode ser testado com outros tipos de operadores de cruzamento, como os operadores de cruzamento para permutações OBX, PBX, PMX, CX e OX, que são eficientes para problemas de otimização combinatória.

Sugere-se também que seja alterado o operador de mutação, de maneira que ele opere sobre a alteração da ordem dos produtos, sendo que estes produtos serão escolhidos aleatoriamente no cromossomo e trocados entre si. Após isso, serão trocados os roteiros de fabricação dos produtos escolhidos. A partir disso, o cromossomo terá uma nova ordem de prioridades dos produtos na programação, inserindo mais diversidade genética na população.

Estas sugestões necessitam ser implementadas isoladamente e testadas para os

problemas em questão. Feito isso, deve-se aplicar uma comparação entre os novos resultados e os obtidos neste trabalho, com a finalidade de verificar se houve melhora nas soluções encontradas.

## Referências

- BENINCASA, A. X. **Um modelo de sistemas Fuzzy para despacho de veículos autoguiados em manufatura integrada**. Dissertação de Mestrado em Ciência da Computação. Universidade Federal de São Carlos, São Carlos, 2003.
- CAMPOS, M.C. **Estatística prática para docentes e pós graduados**. Disponível em: <[http://www.forp.usp.br/restauradora/gmc/gmc\\_livro/gmc\\_livro.html](http://www.forp.usp.br/restauradora/gmc/gmc_livro/gmc_livro.html)>. Acesso em: 5 jun. 2008.
- CARVALHO, A. C. P. L. F.; BRAGA, A.P.; LUDERMIR, T.B. Computação Evolutiva In: REZENDE, S. O. **Sistemas Inteligentes**. 1 ed. São Paulo: Editora Manole, 2003, p. 225-248.
- CAVALIERI, S. Petri nets and genetic algorithms to increase productivity in FMS. In: **Second International Conference on Knowledge-Based Intelligent Electronic Systems**, Adelaide, Australia, 1998. p. 134-142.
- CHAN, F. T. S.; CHUNG, S. H.; CHAN, P. L. Y. An adaptive genetic algorithm with dominated genes for distributed scheduling problems. **Expert Systems with Applications**, v 29, p. 364-371, 2005.
- CHAN, F. T. S.; CHUNG, S. H.; CHAN, P. L. Y.; FINKE, G; TIWARI, M. K., Solving distributed FMS scheduling problems subject to maintenance: Genetic algorithms approach. **Robotics and Computer-Integrated Manufacturing**, v 22, p. 493-504, 2006.
- CHIU, Y. F.; FU, L. C. A. GA embedded dynamic search algorithm over a petri net model for an FMS scheduling. In: **Proceedings of the 1997 IEEE International Conference on Robotics and Automation**, Albuquerque, New Mexico, April, 1997. v. 1 p.513-518.
- DERIZ, A. C. **Um Método de Busca usando Algoritmo Genético para Programação Reativa da Produção de Sistemas de Manufatura com Recursos Compartilhados**. Dissertação de Mestrado em Ciência da Computação. Universidade Federal de São Carlos, São Carlos, 2007.
- GOLDBERG, D. E. **Genetic Algorithms in Search, Optimization and Machine Learning**. Addison-Wesley, 1989.
- GRFENSTETTE, J.J. Optimizations of control parameters for genetic algorithms. **IEEE Trans. Systems, Man, and Cybernetics**, v 16, p. 122-128, 1986.
- HAQ, A. N.,; KARTHIKEYAN, T.; DINESH, M. Scheduling Decisions in FMS using a Heuristic Approach. **International Journal Advanced Manufacturing Technology**, , 2003, v.22, p. 374-379.

- HOLLAND, J. H.. **Adaptation in Natural and Artificial Systems**. University of Michigan Press, 1975.
- JEONG, S. J.; LIM, S. J.; KIM, K. S. Hybrid approach to production scheduling using genetic algorithm and simulation. **International Journal of Advanced Manufacturing Technology, London**, v. 28, p. 129 – 136, 2006.
- JERALD, J.; ASOKAN, P.; SARAVANAN, R.; RANI, A. D. C., Simultaneous scheduling of parts and automated guided vehicles in an FMS environment using adaptive genetic Algorithm. **International Journal Advanced Manufacturing Technology**, v.29, p. 584-589, 2006.
- LACERDA, E. G. M.; CARVALHO, A. C. P. L. F., Introdução aos algoritmos genéticos. In: **Anais XIX Congresso Nacional da Sociedade Brasileira de Computação**, Rio de Janeiro, Julho 1999. p. 51-126.
- LOURIDAS, P. JUnit: Unit Testing and Coding in Tandem. **IEEE Software**, v. 22(4), p. 12-15, 2005.
- MAGGIO, E. G. R. **Uma heurística para a programação da produção de sistemas flexíveis de manufatura usando modelagem em redes de petri**. Dissertação de Mestrado em Ciência da Computação. Universidade Federal de São Carlos, São Carlos, 2005.
- MARTINS, G. A. **Estatística geral e aplicada**. 3. Ed. São Paulo: Atlas, 2005.
- MITCHELL, T.M. **Machine Learning**. McGraw-Hill Science Engineering, 1997.
- MONTEVECHI, J.A.B.; MORANDIN JR., O.; MIYAGI, P.E. Sistemas de manufatura In: AGUIRRE, L.A. **Enciclopédia de Automática, Controle & Automação**. 1 ed. São Paulo: Editora Blucher, 2007, p. 247-287.
- MORANDIN JR, O.; KATO E. R. R.; MAGGIO, E. G. R.; SANCHES, D. S.; DERIZ, A. C. Heuristic based on Petri nets modeling for FMS scheduling problem of makespan minimization, **33rd Annual Conf. of the IEEE Industrial Electronics Society**, 2007a, Taipei – Taiwan, p. 2683-2688.
- MORANDIN JR, O.; KATO E. R. R.; DERIZ, A. C.; SANCHES, D. S. Uma modelagem para programação da produção de sistemas de manufatura com recursos compartilhados utilizando algoritmos genéticos, **XXVII Encontro Nacional de Engenharia de Produção**, 2007b, Foz de Iguaçu – Brasil.
- MORANDIN JR, O.; DERIZ, A. C.; SANCHES, D. S.; KATO E. R. R. A Search Method using Genetic Algorithm for Production Reactive Scheduling Manufacturing Systems. **IEEE International Symposium on Industrial Electronics**, 2008, Cambridge, UK, June, in press.

- PONGCHAROEN P.; HICKS C.; BRAIDEN P. M.; STEWARDSON D.J., Determining optimum genetic algorithm parameters for scheduling the manufacturing and assembly of complex products. **International Journal of Production Economics**. n. 78, p. 311–322, 2002.
- PONGCHAROEN, P.; HICKS, C.; BRAIDEN, P.M. The development of genetic algorithms for the finite capacity scheduling of complex products, with multiple levels of product structure. **European Journal of Operational Research**. n. 152, p. 215–225, 2004.
- REDDY, B. S. P.; RAO, C. S. P. A hybrid multi-objective GA for simultaneous scheduling of machines and AGVs in FMS. **International Journal Advanced Manufacturing Technology**, v.31, p. 601-613, 2006.
- SANKAR, S.S; PONNAMBALAM S. G.; RAJKUMAR R.; GURUMARIMUTHU M., An Intelligent Integrated Scheduling Model for Flexible Manufacturing System, **Proceedings of the 2004 IEEE Conference on Robotics, Automation and Mechatronics**, Singapore, December, 2004, p.1095-1100.
- SANCHES, D. S.; MORANDIN JR, O.; DERIZ, A. C.; KATO E. R. R.; TSUNAKI, R. H. An Adaptive Genetic Algorithm Based Approach for Production Reactive Scheduling of Manufacturing Systems. **34rd Annual Conf. of the IEEE Industrial Electronics Society**, Orlando – USA, November, 2008, p.1461-1466.
- SILVA, E. L.; MENEZES, E. M. **Metodologia da pesquisa e elaboração de dissertação**. 3 ed. Florianópolis: Laboratório de Ensino à Distância da UFSC, 2001
- SLACK, N.; CHAMBERS S.; JOHNSTON R., **Administração da Produção**. São Paulo: Atlas, 2002.
- SRINIVAS, M.; PATNAIK, L.M. Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms, **IEEE Transactions on Systems, Man and Cybernetics**, v. 24, n. 4, p. 656-667, April, 1994a.
- SRINIVAS, M.; PATNAIK, L.M. Genetic Algorithms: A Survey. **IEEE Computer**, vol. 27, no. 6, p. 17-26, June 1994b.
- TUBINO, D. F. **Manual de planejamento e controle da produção**. 2 ed. São Paulo: Atlas, 2000.
- ULUSOY, G.; SIVRIKAYA, F.; BILGE, U. A Genetic Algorithm Approach to the Simultaneous Scheduling of Machines and Automated Guided Vehicles. **Computers & Operations Research**, vol. 24, pp. 335-351, 1997.
- YIN, Y.; YIU, J.; CHENG, Z. A Genetic Algorithm Based Approach to Flowshop Scheduling, **Proceedings of the 5<sup>th</sup> World Congress on Intelligent Control and Automation**, Hangzhou, P.R. China, June, 2004.

ZHANG, L.; WANG, L.; ZHENG, D. An Adaptive Genetic Algorithm with multiple operators for Flowshop Scheduling, **International Journal Advanced Manufacturing Technology**, v. 27, p. 580-587, 2006.

ZHOU, L.; XIN, S.S. A Self-Adaptive Genetic Algorithm for TskS Scheduling in Multiprocessor System, **Proceedings of the 2006 IEEE International Conference on Communications, Circuits and Systems**, China, June, 2006, p.2098-2101.