

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

**ALGORITMO GENÉTICO COM OPERADOR DE
TRANSGENIA PARA MINIMIZAÇÃO DE MAKESPAN
DA PROGRAMAÇÃO REATIVA DA PRODUÇÃO**

MONIQUE SIMPLICIO VIANA

ORIENTADOR: PROF. DR. ORIDES MORANDIN JUNIOR

São Carlos - SP

Agosto/2016

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

**ALGORITMO GENÉTICO COM OPERADOR DE
TRANSGENIA PARA MINIMIZAÇÃO DE MAKESPAN
DA PROGRAMAÇÃO REATIVA DA PRODUÇÃO**

MONIQUE SIMPLICIO VIANA

Dissertação 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, área de concentração: Inteligência Artificial.

Orientador: Prof. Dr. Orides Morandin Junior

São Carlos - SP

Agosto/2016



UNIVERSIDADE FEDERAL DE SÃO CARLOS
Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovação

Assinaturas dos membros da comissão examinadora que avaliou e aprovou a defesa de Dissertação de Mestrado da candidata Monique Simplicio Viana, realizada em 29/08/2016.

Prof. Dr. Orides Morandin Jr
(UFSCar)

Prof. Dr. Ricardo Augusto Souza Fernandes
(UFSCar)

Prof. Dr. Danilo Sipoli Sanches
(UTFPR)

Certifico que a sessão de defesa foi realizada com a participação à distância do membro Prof. Dr. Danilo Sipoli Sanches. Depois das arguições e deliberações realizadas, o participante à distância está de acordo com o conteúdo do parecer da comissão examinadora redigido no relatório de defesa da aluna Monique Simplicio Viana.

Prof. Dr. Orides Morandin Jr
Coordenador da Comissão Examinadora
(UFSCar)

AGRADECIMENTO

Gostaria de agradecer a todos que, de alguma forma, contribuíram para que este trabalho pudesse ser realizado. Em especial agradeço:

Ao meu orientador, professor Orides, pelo auxílio e discussões, dando o suporte necessário para a realização deste trabalho. Agradeço pelas críticas, pela paciência e pela prestatividade na condução deste trabalho.

A toda minha família, principalmente aos meus pais, pelo imenso amor que recebo todos os dias, pelo apoio, incentivo e carinho.

Ao meu namorado Rodrigo, pelo seu jeito único de ser, por todo seu carinho, atenção, apoio e suas valiosas dicas e revisões matemáticas.

Aos meus amigos e todos aqueles que diretamente ou indiretamente contribuíram para realização do mestrado.

À CAPES, pelo suporte financeiro.

As coisas que um dia eu imaginei como minhas maiores conquistas foram apenas o primeiro passo rumo a um futuro que apenas comecei a visualizar.

Jace Beleren

RESUMO

Nos últimos anos, várias pesquisas vêm sendo realizadas a fim de minimizar o tempo total de produção (*makespan*) em uma programação da produção de algum cenário que representa um sistema de manufatura. O problema da programação da produção é classificado como sendo um problema combinatório pertencente à classe NP-Hard dos problemas computacionais. Além disso, em um sistema de produção real, há muitos eventos inesperados (por exemplo, a revisão da produção, chegada de novos produtos, quebra máquinas, etc.). Para lidar com as interrupções da programação inicial, é preciso realizar outra programação, a qual é denominada de programação reativa da produção. Sendo um problema de recursos combinatórios, é amplamente utilizado meta-heurísticas em sua resolução. Neste trabalho é proposto um método que faz uso de uma meta-heurística evolutiva Algoritmo Genético em conjunto com um operador intitulado Operador de Transgenia, no qual possibilita manipular o material genético dos indivíduos acrescentando características das quais se acredita serem importantes, com a proposta de direcionar alguns indivíduos da população para uma solução mais favorável para o problema sem tirar a diversidade da população com um custo menor de tempo. O Objetivo deste trabalho é utilizando o Algoritmo Genético com Operador de Transgenia obter uma programação reativa em tempo de resposta aceitável, visando minimizar o valor de *makespan*. Resultados experimentais mostraram que algoritmo proposto foi capaz de trazer resultados de *makespan* melhores que os algoritmos comparados e em um menor tempo de processamento, devido ao direcionamento na busca que operador de transgenia proporciona.

Palavras-chave: Programação da Produção, Programação Reativa da Produção, Algoritmo Genético, Operador Transgênico, Sistemas de manufatura, Problemas Combinatórios, Meta-Heurística.

ABSTRACT

In recent years, several studies have been carried out to minimize the production time (makespan) in a production schedule of a scenario that represents a manufacturing system. The problem of production scheduling is classified as a combinatorial problem belongs to the NP-hard class of computational problems. Furthermore, in a real world production system, there are many unexpected events (eg, review of production, entry of new products, breaking machines, etc.). To deal with the interruptions of the initial programming, we need to change any settings, which is called reactive production schedule or, simply, reactive scheduling. As a problem of combinatorial features, meta-heuristics is widely used in its resolution. This paper proposes a method that uses an evolutionary meta-heuristic Genetic Algorithm in conjunction with an operator called "Transgenics", which allows to manipulate the genetic material of individuals adding features which are believed to be important, with the proposal to direct some population of individuals to a more favorable solution to the problem without removing the diversity of the population with a lower cost of time. The objective of this study is to use the Genetic Algorithm with transgenics operator obtain a reactive programming acceptable response time to minimize the makespan value. The objective of this study is to use the Genetic Algorithm with transgenics Operator obtain a reactive programming acceptable response time to minimize the makespan value. Experimental results show the proposed algorithm is able to bring better results than the makespan algorithm and compared in a shorter processing time due to the search direction which provides transgenic operator.

Keywords: Scheduling, Reactive Scheduling, Genetic Algorithm, Transgenic Operator, Manufacturing System, Combinatorial problems, Meta Heuristic.

LISTA DE FIGURAS

Figura 2.1 - Exemplo de pontos de corte (Adaptado de LINDEN, 2012).....	23
Figura 2.2 - Operador de cruzamento com 1 ponto de corte (Adaptado de LINDEN, 2012).....	23
Figura 2.3 - Exemplo de cruzamento com 2 pontos de corte (Adaptado de LINDEN, 2012).....	24
Figura 2.4 - Exemplo de Mutação (Adaptado de LINDEN, 2012).	25
Figura 3.1 Transgenia.....	49
Figura 3.2 Fluxograma do AG com operador Transgênico.....	50
Figura 3.3 Exemplos de Cromossomos de Sequência de Produção.....	54
Figura 3.4 - Passo 1 da Técnica de Cruzamento.....	55
Figura 3.5 Passo 2 da Técnica de Cruzamento.....	55
Figura 3.6 Diagrama de Atividade da Validação.	58
Figura 4.1 - Tela do Software de Geração de Cenários de sistemas de manufatura. ...	60
Figura 4.2 - Saída do software de geração de cenários de sistemas de manufatura.	60
Figura 4.3 – Cromossomo cujo <i>makespan</i> resulta em 4621 u.t.....	73
Figura 4. 4 – Gráfico de convergência do algoritmo proposto.....	76
Figura 4. 5 – Gráfico da média da população em cada iteração do algoritmo proposto.	77
Figura 4.6 – Gráfico de convergência do algoritmo genético sem operador de transgenia.....	77
Figura 4.7 – Gráfico da média da população em cada iteração do algoritmo genético sem operador de transgenia.	78
Figura 4.8 – Gráfico de convergência do algoritmo genético adaptativo.....	78
Figura 4.9 – Gráfico da média da população em cada iteração do algoritmo genético adaptativo.	79
Figura 4. 10 – Gráfico de convergência do algoritmo ACO.	79
Figura 4. 11– Gráfico da média da população em cada iteração do ACO.	80

LISTA DE TABELAS

Tabela 3.1 Configurações do AG Proposto.	52
Tabela 4.1 Produtos e seus respectivos roteiros gerados pelo <i>software</i> criado.	61
Tabela 4.2 Configurações de cenários a serem gerados.	61
Tabela 4.3 Variáveis para o Método Proposto.	65
Tabela 4.4 Dados de Melhorias.	66
Tabela 4.5 Classificação de Genes utilizando Média Ponderada.	66
Tabela 4.6 Classificação de Genes utilizando PCA.	66
Tabela 4.7 Resultados obtidos na execução do cenário com os genes mais significativos determinados pelo método proposto.	67
Tabela 4.7 Tempos de manufatura do cenário.	69
Tabela 4.8 Cenário de produção com 9 máquinas e 9 produtos.	69
Tabela 4.9 Especificação do cenário.	70
Tabela 4.10 Resultados obtidos na execução do cenário.	72
Tabela 4.11 Resumo dos resultados obtidos na execução do cenário.	73
Tabela 4.12 Teste de hipótese do AG Proposto X AG.	74
Tabela 4.13 Teste de hipótese do AG Proposto X AG.	74
Tabela 4.13 Teste de hipótese do AG Proposto X AG Adaptativo.	74
Tabela 4.14 Teste de hipótese do AG Proposto X AG Adaptativo.	75
Tabela 4.15 Teste de hipótese do AG Proposto X ACO.	75
Tabela 4.16 Teste de hipótese do AG Proposto X ACO.	75
Tabela 4.17 Resultados obtidos na execução do cenário 3x8.	81
Tabela 4.18 Resumo dos resultados obtidos na execução do cenário 3x8.	82
Tabela 4.19 Resultados obtidos na execução do cenário 20x8.	83
Tabela 4.20 Resumo dos resultados obtidos na execução do cenário 20x8.	84
Tabela 4.21 Resultados obtidos na execução do cenário 100x40.	84
Tabela 4.22 Resumo dos resultados obtidos na execução do cenário 100x40.	85
Tabela 4.23 Resultados obtidos na execução do cenário 150x40.	85
Tabela 4.24 Resumo dos resultados obtidos na execução do cenário 150x40.	87
Tabela 4.25 Resultados obtidos na execução do cenário 180x80.	87
Tabela 4.26 Resumo dos resultados obtidos na execução do cenário 180x80.	88

LISTA DE ABREVIATURAS E SIGLAS

A*	<i>A star</i>
ACO	<i>Ant Colony Optimization</i>
AG	Algoritmo Genético
AGA	Algoritmo Genético Adaptativo
AGV	Veículo Auto Guiado (<i>Automated Guided Vehicle</i>)
FMS	Sistema Flexível de Manufatura (<i>Flexible Manufacturing System</i>)
GRASP	<i>Greedy Randomized Adaptive search procedure</i>
Meta-RaPS	<i>Meta-heuristics for Randomized Priority Search</i>
NMF	<i>Non-negative matrix factorization</i>
NP	<i>Nondeterministic Polynomial</i>
PCA	<i>Principal Component Analysis</i>
PCP	<i>Planejamento e Controle da Produção</i>
PSO	<i>Particle Swarm Optimization</i>
PH	<i>Partitioning Heuristic</i>
RSA	<i>Restricted Simulated Annealing</i>
SA	<i>Simulated Annealing</i>

SUMÁRIO

CAPÍTULO 1	12
1 INTRODUÇÃO.....	12
1.1 <i>Contextualização.....</i>	12
1.2 <i>Motivação e Justificativa.....</i>	14
1.3 <i>Objetivos</i>	16
1.3.1 <i>Gerais.....</i>	16
1.3.2 <i>Específicos</i>	16
1.4 <i>Delimitações do trabalho</i>	17
1.5 <i>Método de pesquisa e desenvolvimento</i>	17
1.6 <i>Organização do trabalho</i>	18
CAPÍTULO 2	19
2 ESTADO DA ARTE.....	19
2.1 <i>Considerações iniciais.....</i>	19
2.2 <i>Fundamentação teórica</i>	19
2.2.1 <i>Sistemas Flexíveis de Manufatura</i>	19
2.2.2 <i>Programação da Produção</i>	20
2.2.3 <i>Programação Reativa da Produção.....</i>	20
2.2.4 <i>Algoritmos Genéticos</i>	21
2.3 <i>Artigos correlatos.....</i>	26
2.3.1 <i>Trabalhos Relacionados à Programação da Produção</i>	27
2.3.2 <i>Trabalhos Relacionados à Programação Reativa da Produção</i>	34
2.4 <i>Considerações finais.....</i>	46
CAPÍTULO 3	47
3 PROPOSTA DO TRABALHO	47
3.1 <i>Considerações iniciais.....</i>	47
3.2 <i>Algoritmo Genético com Operador de Transgenia.....</i>	48
3.3 <i>Configurações do Algoritmo Genético com Operador de Transgenia</i>	52
3.4 <i>Ambientes de Testes e Validação da Proposta.....</i>	57
3.5 <i>Considerações finais.....</i>	58
CAPÍTULO 4	59

4	APLICAÇÃO DA ABORDAGEM E ANÁLISE DE RESULTADOS	59
4.1	<i>Considerações Iniciais</i>	59
4.2	<i>Programa para Gerar Cenários de Sistemas de Manufatura</i>	59
4.3	<i>Descrição dos métodos para encontrar os genes mais significativos</i>	61
4.3.1	Método utilizando média ponderada de grandezas de estatísticas	62
4.3.2	Método utilizando PCA.....	63
4.3.3	Exemplo de resultados obtidos da execução dos métodos propostos.....	65
4.3.4	Comparação entre o método que utiliza Média Ponderada e o método que utiliza PCA	67
4.4	<i>Avaliação da abordagem</i>	68
4.4.1	Definição dos cenários para a avaliação	69
4.4.2	Resultados	70
4.4.3	Testes em cenários gerados pelo software desenvolvido	80
4.5	<i>Considerações finais.....</i>	88
CAPÍTULO 5	89
5	CONCLUSÃO	89
5.1	<i>Trabalhos futuros.....</i>	90
	REFERÊNCIAS BIBLIOGRÁFICAS.....	92

Capítulo 1

INTRODUÇÃO

1.1 Contextualização

O ambiente de manufatura é altamente ágil e está sempre enfrentando mudanças contínuas de requisições de clientes. O Sistema Flexível de Manufatura, ou FMS, surgiu como um importante sistema de manufatura devido à sua flexibilidade, que é essencial para a competitividade em um ambiente altamente dinâmico (REDDY; RAO, 2011).

FMS é um sistema complexo que consiste de elementos que trabalham simultaneamente com máquinas, sistema de transporte, sistema de armazenamento e recuperação, sistema de controle (REDDY; RAO, 2011) e, normalmente, tem o transporte feito por veículos auto-guiados (AGV - *Automated Guided Vehicle*). Tais sistemas fornecem uma forma de enfrentar o desafio de se obter melhor qualidade, baixo custo e tempos de entregas curtos ao implementar a flexibilidade em empresas de manufatura e linhas de produção (ZAKARIA; PETROVIC, 2012).

As peças presentes em um FMS podem utilizar diferentes máquinas que realizam diferentes operações e que geram uma grande demanda de AGVs (REDDY; RAO, 2011). Para obter melhor proveito dessa flexibilidade e usá-la de forma eficiente é necessário utilizar uma boa programação da produção, que define a ordem e o momento da alocação dos recursos em uma sequência correta. Geralmente, busca-se minimizar o tempo de conclusão de tarefa utilizando alguns critérios conhecidos como *due date*, *lead time* e *makespan*, entre outros.

O uso de FMS trouxe uma série de benefícios, tais como redução de custos, redução de trabalhos em processamento, etc. Há uma série de problemas enfrentados durante o

processo de um FMS. Estes problemas são de projeto, planejamento, programação e controle da produção (SUBBAIAH; RAO; RAO, 2009).

O problema da programação da produção comum segue algumas premissas, a saber:

- Cada trabalho consiste em um número finito de operações;
- O tempo de processamento para cada operação em uma determinada máquina está definido;
- Existe uma sequência pré-definida de operações que têm de ser mantidos para completar cada trabalho;
- Os prazos de entrega dos produtos podem ser indefinidos, se necessário;
- Pode ou não haver custo de *setup* ou custo de atraso;
- Uma máquina pode processar apenas uma tarefa de cada vez;
- Cada trabalho visita cada máquina apenas uma vez;
- Nenhuma máquina pode lidar com mais de um tipo de tarefa;
- O sistema não pode ser interrompido até que cada operação de cada trabalho esteja no estado terminado;
- Nenhuma máquina pode interromper um trabalho e começar outro trabalho antes de terminar o anterior (HASAN; SARKER; ESSAM, 2010).

O objetivo da programação da produção é a minimização do tempo, sendo ele *due date*, *lead time*, *makespan*, entre outros (HASAN; SARKER; ESSAM, 2010). Neste projeto será utilizado o tempo de *makespan*, o qual representa o tempo gasto para completar todas as tarefas em um sistema de fabricação da programação de produção seguindo as premissas descritas anteriormente. Então em uma programação da produção de FMS, as decisões que precisam ser feitas, podem incluir sequenciamento de trabalhos em máquinas e do encaminhamento dos trabalhos através do sistema levando em consideração também algum tipo de medição de tempo (SUBBAIAH; RAO; RAO, 2009).

No entanto, em muitas vezes existem eventos inesperados (a chegada de um pedido urgente, máquinas quebradas, etc.) que podem parar um processo de produção em um sistema de manufatura (HASAN; SARKER; ESSAM, 2010), (MORANDIN et al., 2009) e (TANG; WANG, 2008).

Em muitos ambientes de produções reais, sistemas de programação geralmente operam em ambientes altamente dinâmicos e incertos, no qual existem vários eventos

inesperados que podem impedir a execução da programação da produção exatamente como foram desenvolvidos. A maioria dos FMS opera em ambientes dinâmicos sujeitos a vários eventos em tempo real, o que pode tornar a programação reativa viável. Portanto, a programação reativa é de grande importância para o sucesso da implementação de um sistema do mundo real (ZHANG; GAO; LI, 2013).

Os autores (ABELLO; MICHALEWICZ; BUI, 2012) definem a programação reativa como sendo a busca de uma solução por meio de alguma otimização usando como base uma programação anterior. Essa base é então revisada para se tornar perfeitamente adequada para o novo estado do ambiente, que pode ter sido produzido por algum evento inesperado.

Os autores Zakaria e Petrovic (2012) descrevem três categorias para as abordagens da reprogramação:

1. A programação reativa que produz uma programação ao longo do tempo com alterações de estado;
2. A programação robusta na qual uma programação é produzida de tal forma que ela pode contornar ou resolver as perturbações ocorridas;
3. A programação preditiva-reativa em que uma programação anterior que otimiza o desempenho do chão de fábrica é gerado e modificado sempre que necessário.

1.2 Motivação e Justificativa

Os problemas de programação da produção e programação reativa da programação são descritos na literatura como sendo da classe *NP-Hard* (NIE *et al.*, 2012), (MORANDIN *et al.*, 2009) e (UDHAYAKUMAR; KUMANAN, 2010). Sendo assim, é aconselhada a utilização de algoritmos estocásticos, heurísticos e meta-heurísticos, os quais são recomendados para fazer a otimização de problemas da classe *NP-Hard* (HASAN; SARKER; ESSAM, 2010).

Como dito anteriormente, os problemas de programação da produção e de produção reativa são classificadas como *NP-Hard*. Desta forma, muitos pesquisadores têm utilizado heurísticas e meta-heurísticas para encontrar uma solução aceitável fazendo uso de algum parâmetro para validar sua solução.

Nos últimos anos, foram desenvolvidas pesquisas para solucionar o problema de programação utilizando meta-heurísticas, tais como as pesquisas de Zobolas, Tarantilis e Iannou (2009), Semanco e Modrak (2011), Gomez-Gasquet, Andres e Lario (2012), Roshanaei *et al.* (2009), Kato, Morandin e Fonseca (2010) e Witkowski, Antczak e Antczak (2010). Nestas pesquisas, foi feito uso de algum tipo de meta-heurística, tais como algoritmo genético (AG), procedimento de busca *Greedy Randomized Adaptive Search Procedure* (GRASP), *Ant Colony Optimization* (ACO), etc. Como forma de validação de suas abordagens foi considerada a minimização de *makespan*.

Como no problema de programação da produção, também existem muitas pesquisas desenvolvidas para resolver o problema de programação reativa utilizando meta-heurísticas. Os seguintes trabalhos foram desenvolvidos por meio do algoritmo genético para resolver o problema em questão: Morandin *et al.* (2009), Wu *et al.* (2010), Manchon *et al.* (2011), Zakaria e Petrovic (2012), Hasan, Sarker e Essam (2010), Al-Hinai e Elmekawy (2011), Tanimizu *et al.* (2007) e Nie *et al.* (2012). Tais trabalhos usaram o algoritmo genético para fazer a busca e garantir a otimização do problema da programação reativa.

Os algoritmos genéticos são utilizados para problemas de otimização por fazerem busca global para encontrar as melhores soluções. O conceito de Algoritmo genético (AG) foi introduzido pela primeira vez por John Holland na década de 70 e representa uma meta-heurística baseada nos princípios da seleção natural e da genética natural. O AG tem obtido bastante sucesso em aplicações de aprendizado de máquina e problemas de otimização (GUO; WANG; HAN, 2010).

Em Guo, Wang e Han (2010) é descrito que o algoritmo genético possui três vantagens para problemas de otimização:

1. O AG não tem muitos requisitos matemáticos sobre os problemas de otimização. Devido à sua natureza evolutiva, o AG irá buscar soluções sem levar em conta os mecanismos específicos internos do problema;
2. Os operadores evolutivos do AG fazem uma busca global eficaz, sendo que os principais operadores que fazem com que o AG seja eficiente são os de cruzamento e mutação;
3. AG proporciona uma grande flexibilidade para criar uma hibridização com outras heurísticas para realizar uma implementação mais eficiente em um problema específico.

O AG tem muitas vantagens, mas mesmo que o AG possa localizar a solução em todo o domínio, não resolve os problemas complexos facilmente (GUO; WANG; HAN, 2010). A desvantagem do algoritmo genético é o alto consumo de recursos, isto é, quanto maior o domínio de soluções, maior será o tempo de busca utilizado pelo AG (NIE *et al.*, 2012), (WEI *et al.*, 2010) e (KAZEMI *et al.*, 2012).

Os pesquisadores Amaral e Hruschka (2011) e Amaral e Hruschka (2014) propuseram um operador para o algoritmo genético conhecido como operador transgênico, o qual é inspirado pela engenharia genética, onde é possível manipular o material genético dos indivíduos acrescentando características das quais se acredita serem importantes. Assim, essa abordagem pode ser vista como uma estratégia de elitismo focada em genes específicos. Dessa forma, existe a hipótese de que pode ser usada a meta-heurística do AG com o novo operador genético para reduzir o tempo total de produção e minimizar o *makespan*, isso tudo em um tempo aceitável. Isso porque o operador transgênico deve direcionar alguns indivíduos da população para uma solução mais favorável ao problema, mantendo a diversidade da população e com um custo menor de tempo para realizar a busca de soluções favoráveis.

1.3 Objetivos

1.3.1 Gerais

Como apresentado na seção anterior, os AGs tradicionais são muito utilizados para problemas de otimização, devido à sua busca local ser muito eficiente, porém gasta muito tempo para explorar o espaço de soluções em problemas complexos. Em 2011 foi proposto um novo operador genético que pode auxiliar e direcionar as buscas do AG sem diminuir a sua diversidade populacional.

O objetivo do trabalho é fazer a minimização do *makespan* em uma programação reativa da produção utilizando o operador transgênico do Algoritmo Genético.

1.3.2 Específicos

- Criar mais de um ambiente de testes para fazer a validação do objetivo geral;

- Desenvolver um método ou mais de um para encontrar o gene mais significativo ou genes mais significativos para serem usados no Algoritmo Genético com operador Transgênico;
- Adaptação do algoritmo genético com operador transgênico para o problema de programação reativa da produção.

1.4 Delimitações do trabalho

Nessa seção serão apresentados os delimitadores, itens que não serão abordados no projeto, para que não possa haver dúvida do que será apresentado ou usado durante a pesquisa.

Os delimitadores desse trabalho são:

- A validação não será feita em um cenário real de manufatura, ou seja, os dados que serão utilizados para a validação do projeto não são produzidos por um sistema real de manufatura e sim por um cenário virtual que utiliza características de um sistema de manufatura;
- Os valores de tempo de transporte que serão usados no cálculo de *makespan* são valores médios de transporte, pois no trabalho não serão descritas e detalhadas as rotas de AGV's;
- O tempo de *setup* das máquinas não será levado em conta no cálculo do *makespan*.

1.5 Método de pesquisa e desenvolvimento

O método científico que será usado no projeto de minimização de *makespan* através do AG transgênico será a abordagem quantitativa, o qual é caracterizado pela formação de uma hipótese, definições de variáveis, quantificação na coleta de dados e de informação e uso

de tratamentos estatísticos. O método quantitativo se apoia em dados estatísticos, comprovações ou validações e testes.

1.6 Organização do trabalho

No capítulo 2, são apresentados os principais conceitos envolvidos na proposta, incluindo uma descrição geral do ambiente de produção, assim como, as características do problema em consideração. Os artigos correlatos são apresentados em seguida, buscando apresentar como estão sendo tratados os trabalhos que buscam minimizar o tempo de produção no problema da programação reativa da produção.

O capítulo 3 é dedicado à apresentação da proposta deste trabalho, no qual será descrito o passo a passo de como se pretende desenvolver o projeto para solucionar o problema da programação reativa da produção. Também são apresentados nesse capítulo a forma de validação do problema e o cronograma para a execução da proposta.

No capítulo 4 é apresentado o software desenvolvido para gerar cenários de manufatura, dois métodos elaborados para determinar os genes mais significativos, as definições dos cenários avaliados, resultados obtidos por meio de ensaios e a avaliação da abordagem.

Por fim, no capítulo 5 são apresentadas as conclusões e as propostas para trabalhos futuros.

Capítulo 2

ESTADO DA ARTE

2.1 Considerações iniciais

Neste capítulo serão apresentados e discutidos os principais conceitos desse trabalho, incluindo a caracterização do ambiente de aplicação e do problema considerado. Em seguida, é introduzida a descrição da meta-heurística Algoritmo Genético, servindo como fundamento geral para a proposta do trabalho.

Na seção 2.3, são apresentados alguns trabalhos relacionados à programação da produção e à programação reativa da produção, visando indicar as principais questões quanto ao uso de métodos para o problema de programação e de como suas especificidades têm sido exploradas para a tentativa de resolução do problema.

2.2 Fundamentação teórica

Nesta seção serão apresentados alguns conceitos fundamentais à execução do trabalho.

2.2.1 Sistemas Flexíveis de Manufatura

Um Sistema Flexível de Manufatura (FMS) pode ser caracterizado como um agrupamento de equipamentos, considerados como estações de processamento, interligados por um sistema automatizado de carga e descarga e de movimentação de peças e ferramentas, e um sistema de armazenamento. Este tipo de sistema representa uma configuração de grande flexibilidade no que se refere à inserção de novos produtos, *mix* de produção (variação de produtos), processamento das operações e manipulação de materiais (AGUIRRE, 2007).

Existem inúmeros tipos de problemas inseridos nesse ambiente de produção que envolvem questões de controle e planejamento. Na seção seguinte é descrito o problema de programação da produção.

2.2.2 Programação da Produção

A programação da produção está inserida nas atividades de Planejamento e Controle da Produção (PCP), definida na terceira atividade do PCP, a qual, de posse das informações do planejamento mestre da produção, determina o detalhamento das operações que serão realizadas como: onde as operações dos produtos serão processadas; com quais recursos e em qual tempo de início e término para cada ordem de produção (GROOVER, 2006).

Neste contexto, em uma manufatura existe um conjunto de produtos que deve ser processado por m máquinas. Cada tarefa em uma manufatura consiste de n operações, na qual devem ser processados em diferentes máquinas em dado tempo de processamento. O objetivo é encontrar uma programação da produção que minimize o tempo total de processamento de todos os produtos também conhecido por (*makespan*) (BAI; TANG, 2013). Na subseção seguinte é descrito o problema de programação reativa da produção, o qual é tratado neste trabalho.

2.2.3 Programação Reativa da Produção

Na prática, em ambientes de manufatura, a produção está sujeita a acontecimentos inesperados. Um grande conjunto desses acontecimentos pode afetar o cronograma da programação da produção. Os sistemas produtivos reais não são estáticos, por isso podem ocorrer diversos eventos típicos como falta de matéria prima, quebra de máquina, problemas de transporte, entre outros. Dessa maneira, para manter o fluxo da produção, uma programação reativa da produção deve ser realizada (TOGUYÉNI et al., 2003).

A programação reativa da produção leva em consideração os eventos em tempo real, o que tem atraído grande atenção de muitos pesquisadores. Alguns trabalhos baseiam-se em casos passados como forma de minimizar o impacto do evento ocorrido no sistema produtivo, ou ainda em propostas de monitoramento do sistema para a recuperação na ocorrência de falhas, entre outras abordagens (TANG; WANG, 2008).

2.2.4 Algoritmos Genéticos

Os Algoritmos Genéticos (AGs) são algoritmos de busca e otimização, que utilizam regras baseadas no processo evolutivo proposto por Charles Darwin. O AG é uma técnica de busca estocástica, ou seja, um AG com mesma população inicial e o mesmo conjunto de parâmetros pode gerar soluções diferentes a cada vez que é executado (LINDEN, 2012).

Os AGs foram idealizados na década de 60 por John Holland e outros pesquisadores da Universidade de Michigan. Seu objetivo principal era desenvolver maneiras de utilizar os mecanismos de adaptação natural em sistemas computacionais (MITCHELL, 1998).

Com base nas pesquisas realizadas sobre os AGs, John Holland publicou um livro sobre este assunto, chamado: “*Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*” (HOLLAND, 1975). Após isso, grandes avanços foram feitos na área e os AGs têm sido analisados por diversos pesquisadores e aplicado para vários tipos de problemas nas mais diversas áreas.

Seu funcionamento pode ser resumido algorítmicamente em uma visão de alto nível por meio dos seguintes passos (LINDEN, 2012):

- Inicializar a população de cromossomos;
- Avaliar cada cromossomo na população;
- Selecionar os pais para serem gerados novos cromossomos;
- Aplicar os operadores de cruzamento e mutação de forma a gerar os indivíduos da nova geração;
- Avaliar todos os novos cromossomos e inserir os na população;
- Se o tempo acabar, ou o melhor cromossomo satisfazer os requerimentos, retornar o resultado obtido, caso contrário, voltar para o passo 3.

Nas próximas subseções serão discutidas as técnicas utilizadas na implementação do algoritmo supracitado.

2.2.4.1 Seleção

O método de seleção simula o mecanismo de seleção natural que atua sobre as espécies biológicas, em que os pais mais capazes geram um maior número de filhos, ao

mesmo tempo em que permite que os pais menos aptos também gerem descendentes. Simplificando, este método seleciona os indivíduos que, a partir de suas características, darão origem a uma nova geração da população.

A seleção é feita utilizando-se o valor de uma função de avaliação (*fitness*), a qual é calculada sobre cada indivíduo. Esta função recebe como entrada os valores dos genes do cromossomo e fornece como resultado um valor numérico que representa sua aptidão. Nesse processo, a escolha dos membros da população para o processo de reprodução deve privilegiar os indivíduos de *fitness* mais alto sem, entretanto, desconsiderar os de *fitness* mais baixo. O conceito fundamental é que deve-se privilegiar os indivíduos com função de avaliação alta, sem desprezar completamente aqueles indivíduos com função de avaliação extremamente baixa. Esta decisão é razoável, pois até indivíduos com pior avaliação podem ter características genéticas que sejam favoráveis à criação de um indivíduo que seja a melhor solução para o problema em questão (LINDEN, 2012).

Os métodos de seleção mais usados são: o da Roleta e do Torneio. No método da Roleta, os indivíduos de uma geração são selecionados para a próxima geração utilizando a ideia intuitiva de roleta. Isto é, cada indivíduo da população é representado na roleta por uma fatia proporcional ao seu índice de aptidão. Assim, indivíduos com aptidão alta ocupam fatias maiores da roleta, enquanto indivíduos de aptidão mais baixa ocupam fatias menores. Já no método do Torneio, um número de indivíduos da população é escolhido de modo aleatório e com mesma probabilidade e, em seguida, o cromossomo com maior aptidão dentre todos os cromossomos é selecionado para a população intermediária. Este processo se repetirá até que a população intermediária seja preenchida (LINDEN, 2012).

2.2.4.2 Operador Genético de Cruzamento

Através do cruzamento são criados novos indivíduos agregando características de dois indivíduos selecionados como "pais". Parte das características de um indivíduo são trocados pelo trecho equivalente do outro, o resultado desta operação é um indivíduo que potencialmente pode ter a combinação das melhores características dos indivíduos usados como base.

Um exemplo de um possível tipo de cruzamento é o operador de um ponto, depois de selecionados dois pais pelo módulo de seleção, um ponto de corte é selecionado. Um ponto de

corte constitui uma posição entre dois genes de um cromossomo. Cada indivíduo de n genes contém $n - 1$ pontos de corte, e este ponto de corte é o ponto de separação entre cada um dos genes que compõem o material genético de cada pai. Na Figura 2.1 vê-se um exemplo de pontos de corte presentes em um cromossomo. No caso, o cromossomo é composto de 6 genes e, por conseguinte, tem 5 pontos de corte possíveis.

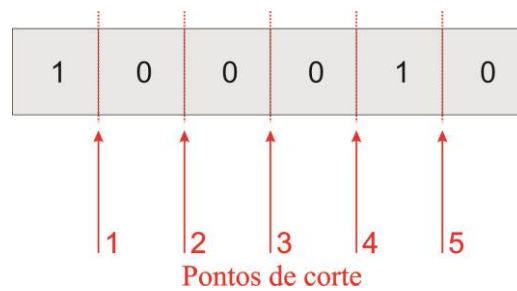


Figura 2.1 - Exemplo de pontos de corte (Adaptado de LINDEN, 2012).

Depois de sorteado o ponto de corte, são separados os pais em duas partes: uma à esquerda do ponto de corte e outra à direita. É importante notar que não necessariamente essas duas partes têm o mesmo tamanho. Por exemplo, se for selecionado o ponto de corte número 2 da Figura 2.1, a parte esquerda de cada pai teria 2 genes enquanto que a parte direita teria 4. A ilustração de um cruzamento entre dois indivíduos utilizando tais configurações pode ser observada na Figura 2.2.



Figura 2.2 - Operador de cruzamento com 1 ponto de corte (Adaptado de LINDEN, 2012).

Assim, tem-se que na Figura 2.2, o primeiro filho é composto pela concatenação da parte do primeiro pai à esquerda do ponto de corte com a parte do segundo pai à direita do

ponto de corte. O segundo filho é composto pela concatenação das partes que sobraram (a metade do segundo pai à esquerda do ponto de corte com a metade do primeiro pai à direita do ponto de corte) (LINDEN, 2012).

Também podem ocorrer situações nas quais mais de um ponto de corte é definido, como apresentado na Figura 2.3.

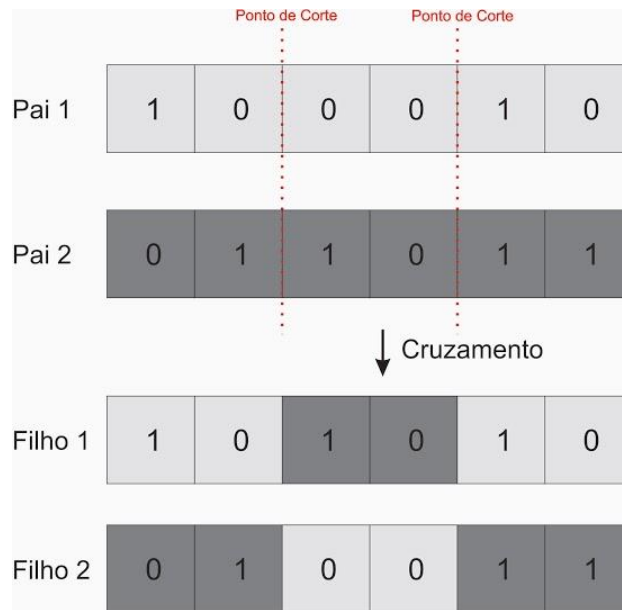


Figura 2.3 - Exemplo de cruzamento com 2 pontos de corte (Adaptado de LINDEN, 2012).

2.2.4.3 Operador Genético de Mutação

O operador de mutação é fundamental para um AG, pois é ele que garante a continuidade da existência de diversidade genética na população. Ele é responsável pela introdução de pequenas mudanças aleatórias nos cromossomos dos filhos, com o objetivo de modificar alguma(s) característica(s) de um indivíduo de forma aleatória. Com isso, é possível explorar o espaço de busca de maneira mais completa e vasta, evitando que o algoritmo fique preso em ótimos locais, por exemplo.

Depois que um cruzamento é realizado, acontece a mutação. A operação de mutação muda aleatoriamente o gene correspondente por um valor presente no alfabeto genético da população em questão, conforme apresentado na Figura 2.4. No caso de uma codificação binária, pode-se mudar aleatoriamente alguns *bits* escolhidos de 1 para 0, ou de 0 para 1.

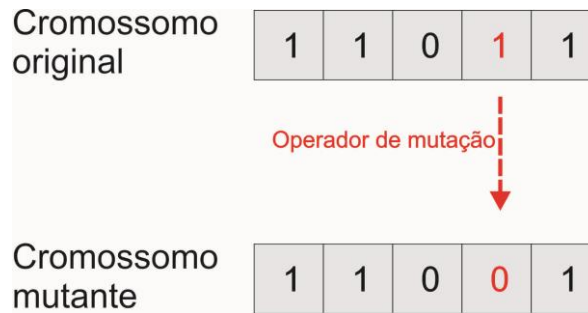


Figura 2.4 - Exemplo de Mutação (Adaptado de LINDEN, 2012).

O conceito fundamental quanto ao valor da probabilidade é que ele deve ser baixo. Se ele for muito alto, o algoritmo genético se parecerá muito com uma técnica chamada *random walk*, na qual a solução é determinada de forma aleatória (simplesmente sorteando-se elementos sem usar informações correntes ou passadas). Se, por exemplo, a taxa de mutação for colocada 100%, o algoritmo se comportará de forma estranha, neste caso todos os *bits* do cromossomo serão invertidos e a qualidade da população degenerará rapidamente e dificilmente o AG convergirá para uma solução (LINDEN, 2012).

2.2.4.4 Parâmetros Genéticos

O desempenho de um Algoritmo Genético é fortemente influenciado pela definição dos parâmetros a serem utilizados, como: tamanho da população, taxa de cruzamento e taxa de mutação. Os principais parâmetros a serem definidos para um AG são (MITCHELL, 1998):

- **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 decair, pois pode não haver indivíduos suficientes para explorar todo espaço de busca, tornando-se incapaz de encontrar boas soluções;
- **Taxa de Cruzamento:** Quanto maior a taxa, mais rapidamente novas estruturas serão introduzidas na população, deste modo podem ocorrer perdas de genes que talvez pudessem ser importantes no processo de busca da solução desejada. Por outro lado, caso esta taxa seja muito pequena, pode-se demorar muito para que o método encontre uma solução ou a busca pode estagnar, uma vez que os melhores indivíduos passarão seus genes com uma frequência

muito baixa e desse modo o AG pode demorar a convergir para o ótimo desejado;

- Taxa de Mutação: Uma baixa taxa de mutação previne que a busca fique estagnada em sub-regiões do espaço de busca, porém se mal ajustado, pode fazer com que o AG se comporte de uma maneira não esperada fazendo com que ele não procure soluções em novas regiões do espaço de busca ou simplesmente tornando a busca aleatória;
- Critério de parada: Diferentes critérios podem ser utilizados para finalizar a execução de um Algoritmo Genético, como, 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.

É muito importante analisar como os parâmetros podem ser utilizados diante das necessidades, pois esses valores são particulares de cada problema. Para a definição de tais valores, podem ser realizados testes com valores variáveis para que se obtenha a ideal definição para a situação a qual deseja tratar (LINDEN, 2012).

2.3 Artigos correlatos

A revisão teórica é o resultado do processo de levantamento e análise do que já foi publicado sobre o tema ou problema de pesquisa escolhido. Assim, o pesquisador terá um mapeamento de quais pesquisadores já escreveram e o que já foi escrito sobre o tema e problema da pesquisa (SILVA; MENEZES, 2005).

Para criar uma revisão teórica, é recomendável que se adote a metodologia de pesquisa bibliográfica, que é aquela baseada na análise da literatura já publicada em forma de periódicos, publicações em eventos e/ou congresso e até por meio de outros trabalhos de monografia, dissertação ou teses (SILVA; MENEZES, 2005).

A revisão de literatura/pesquisa bibliográfica contribuirá para:

- Obter informações sobre a situação atual do tema ou problema pesquisado;

- Conhecer publicações existentes sobre o tema e os aspectos que já foram abordados,
- Verificar as opiniões similares e diferentes a respeito do tema ou de aspectos relacionados ao tema ou ao problema de pesquisa (SILVA; MENEZES, 2005).

Nesta seção, serão apresentados os artigos relacionados à Programação da Produção e Programação Reativa da Produção que fazem uso de meta-heurísticas na resolução de seus problemas.

2.3.1 Trabalhos Relacionados à Programação da Produção

Os pesquisadores Ying, Lee e Lin (2012) descrevem o problema da programação da produção como sendo um conjunto $N = \{P_1, P_2, P_3, \dots, P_n\}$ de n produtos que serão processados em m roteiros de máquinas $M = \{R_1, R_2, R_3, \dots, R_m\}$. Cada produto $i \in N$ tem um tempo de processamento P_{ij} em uma máquina j que deve ser requisitada para o produto i . Além do tempo de processamento, é levado em consideração o tempo de *setup* S_{ijk} , no qual i é o produto processado na máquina k e j é o produto seguinte, com o objetivo de minimizar o tempo de *makespan*.

Pesquisas realizadas para a programação da produção em manufaturas também foram exploradas nos últimos anos. As abordagens que têm sido propostas, incluindo agentes inteligentes e híbridas de algoritmo genético e seus resultados são mostrados por meio de gráficos de Gantt (KHALID; YUSOF; SABUDIN, 2012).

O problema de programação em FMS é visto como sendo fortemente *NP-Hard* e é difícil de encontrar a sua solução ideal. Abordagens diretas, heurísticas e meta-heurísticas são os principais métodos utilizados para resolver o problema de programação de FMS. Muitos pesquisadores têm utilizado as abordagens diretas, tais como técnicas de enumerativos, *branch and bound* e regras de prioridade de despacho para resolver problemas de programação. Abordagens diretas consomem muito tempo e podem não serem capazes de produzirem soluções de qualidade para um tamanho muito grande de variáveis do problema. Por conseguinte, o esforço tem levado a procurar e desenvolver heurísticas e meta-heurísticas eficientes para a resolução de problemas de grande porte (UDHAYAKUMAR; KUMANAN, 2010).

O trabalho de Zobolas, Tarantilis e Ioannou (2009) trata o problema da programação da produção em um *flow shop* com o objetivo de minimizar o seu valor de *makespan*. Os autores usaram um híbrido das meta-heurísticas VNS (*variable neighbourhood search*) e AG, e descrevem que o procedimento VNS está fortemente envolvido no âmbito do AG, ou seja, vizinhanças individuais são pesquisadas antes de uma nova geração genética. Para tratar o problema, os autores consideram os seguintes itens:

- Há um conjunto finito de produtos, de máquinas e de operações, o qual é um conjunto de sequenciamento de máquinas para a criação de um produto;
- Uma máquina pode realizar somente uma operação naquele determinado tempo, ou seja, quando a máquina está processando algo ela só estará livre ao final da operação e não é permitido interrupções;
- Ao final é feito o cálculo do *makespan*, usando também o tempo de *setup* de cada máquina.

Os cenários para testes que foram usados são os 12 conjuntos do *benchmark* desenvolvidos por Taillard (1993). Os resultados obtidos foram comparados com *Simulated Annealing Algorithm* (SAOP) dos autores Osman e Potts (1989), Busca Tabu dos autores Widmer e Hertz (1989), ILS (*Iterated Local Search*) do autor Stützle (1998), duas ACO desenvolvidas por Rajendran e Ziegler (2004), AG desenvolvido pelos autores Ruiz e Maroto (2006) e PSO (*Particle Swarm Optimization*) por Tasgetiren et al. (2007).

A qualidade da solução foi medida, principalmente em termos do *makespan* em relação às melhores soluções conhecidas. Cada instância foi executada dez vezes consecutivas e os desvios da média e melhor *makespan* obtido foram calculados. Com os resultados obtidos os autores relatam que os algoritmos AG por Ruiz e Maroto (2006), PSO e o AG proposto tiveram bons resultados nos testes que foram executados com menores diferenças entre eles.

Mais especificamente, AG tem o menor desvio médio entre as melhores soluções conhecidas e produz os melhores resultados em casos de dimensões 50×20 (50 produtos e 20 máquinas), 100×20 , 200×10 e 500×20 . O PSO apresenta os melhores resultados no grupo de 50×10 e 100×5 , enquanto o AG proposto teve desempenho melhor do que todos os seus concorrentes em 20×5 , 20×10 , 20×20 , 100×10 e 200×20 .

Os autores concluíram que os resultados demonstram a eficiência e eficácia do método, que fornece resultados competitivos em tempos computacionais curtos. Mais importante, o sistema proposto teve uma estrutura simples e poucos parâmetros de ajuste definidos pelo usuário, que são dados fixos ou valores dependentes do tamanho do problema depois de um procedimento de análise de sensibilidade profunda.

No trabalho dos autores Arnaout, Rabadi e Musa (2010) é tratado o problema de programação da produção de máquinas paralelas não relacionadas com as seguintes restrições: cada operação pode ser processada no tempo zero por qualquer máquina não relacionada; cada operação pode ser processada somente em uma máquina e não é permitida interrupção; e os tempos de *setup* das máquinas são colocados também no tempo de processamento de uma operação.

O objetivo do trabalho é usar a meta-heurística ACO para encontrar uma solução de alta qualidade para minimizar o *makespan* a um problema de grande porte, ou seja, para uma instância com um número muito grande de variáveis a serem consideradas no problema. Os autores citam em seu trabalho que, na sua revisão teórica, não foi encontrado nenhum problema de programação da produção com máquinas paralelas não relacionadas que tenha utilizado ACO como meta-heurística.

Para resolver o problema com ACO, os autores dividiram a solução em dois estágios: alocação de recursos que serão usados para processar as operações; e sequenciamento de operações, que são formados por vetores que armazenam as máquinas usadas para produzir cada produto, de forma que um produto pode ter uma ou mais do que uma sequência. A alocação e sequenciamento são baseados no feromônio que será avaliado ou ajustado a cada iteração do algoritmo.

Os resultados do ACO desenvolvido pelos autores foram comparados com os trabalhos de Al-Salem (2004), que usou *Partitioning Heuristic* (PH), com o de Helal, Rabadi e Al-Salem (2006), que usaram Busca Tabu e com Rabadi, Moraga e Al-Salem (2006), que fizeram uso da *Meta-heuristics for Randomized Priority Search* (Meta-RaPS). Os ambientes de testes que foram usados no trabalho são os mesmos dos trabalhos de Al-Salem (2004), Helal, Rabadi e Al-Salem (2006) e Rabadi, Moraga e Al-Salem (2006).

Os autores concluíram que os resultados obtidos pelo ACO de dois estágios foram superiores aos resultados de outras técnicas comparadas, sendo o primeiro estágio responsável

por fazer a alocação de recursos e o segundo estágio utilizado para realizar o sequenciamento. Desta forma, foi demonstrado que a meta-heurística ACO pode ser utilizada para resolver problemas de programação da produção.

Os autores Kato, Morandin e Fonseca (2010) apresentam um trabalho que trata da programação da produção em um ambiente de FMS considerando a flexibilidade de escolha de roteiros para produção, ou seja, a escolha de um conjunto de operações para processar algum tipo de produto, sabendo que um produto pode ter mais de um conjunto de operações ou roteiros para ser produzido.

Para tratar o problema, os autores usaram a meta-heurística ACO, com sistema *Ant* Max-Min que faz sua busca de exploração do espaço, para minimizar o tempo total da produção. A avaliação do método é estabelecida aplicando-a aos cenários de grande escala e em comparação com outra abordagem utilizada para o mesmo problema.

Os autores ainda estabeleceram algumas restrições ao problema abordado:

- Cada máquina processa apenas uma operação por vez;
- Não são permitidas interrupções quando uma operação está sendo processada;
- No tempo zero todas as máquinas estão disponíveis;
- As máquinas não quebram;
- O tempo de transporte de recursos até as máquinas é desprezado;
- O tempo de *setup* é o mesmo para cada máquina;
- As ordens de operações de cada produto são fixas e não podem ser alteradas;
- Cada produto pode ter um ou mais conjuntos de operações.

No trabalho, em cada cenário foram executados 35 testes. O tempo de execução para cada operação foi gerado aleatoriamente seguindo um intervalo de 400 e 500 unidades de tempo (u.t.). Os conjuntos de operações para cada produto também foram gerados aleatoriamente, levando em consideração o número de máquinas e operações para cada cenário.

Os resultados foram comparados com o AG desenvolvido por Morandin et al., (2008a). Os resultados mostram que, para o cenário 3 X 6, ambos obtiveram o mesmo

resultado de *makespan*, porém o tempo para obtenção do resultado do ACO foi de 0,88 e do AG de 1,09. Já para o cenário 5×8 a meta-heurística ACO teve uma média de *makespan* melhor do que o AG.

Os autores concluíram que, com base nos resultados alcançados pelo modelo proposto, é possível observar que a abordagem colaborativa do ACO pode conseguir bons resultados para o problema de programação em um FMS que foi considerado em seu trabalho.

No trabalho dos autores Ying, Lee e Lin (2012) é usada a meta-heurística RSA (*Restricted Simulated Annealing*), pois conforme mencionado pelos autores, o *Simulated Annealing* (SA) tem uma grande fraqueza: o número de soluções de vizinhança gerada cresce muito rápido se o número de operações aumenta, o que significa que encontrar uma solução eficiente exige um grande esforço computacional, por isso foi usado a meta-heurística RSA. O RSA emprega uma estratégia de pesquisa restrita, que faz a eliminação de movimentação de tarefas ineficazes, ou seja, roteiros de operações ineficazes para encontrar a melhor vizinhança da solução, que é a minimização do *makespan* em uma programação da produção.

No trabalho de Ying, Lee e Lin (2012) são apresentadas algumas premissas para que o problema da programação da produção seja tratado:

- As operações estão sempre disponíveis no instante zero e não há alguma restrição de precedência entre eles;
- As máquinas estão continuamente disponíveis e só podem lidar com uma operação de cada vez e sem interrupção;
- Cada operação pode ser processada em qualquer máquina livre e, por no máximo, uma máquina em qualquer dado momento.

Os resultados obtidos pela meta-heurística RSA proposta pelos autores foram comparados com os trabalhos de Al-Salem (2004), que usou *Partitioning Heuristic* (PH), Helal, Rabadi e Al-Salem (2006), que usaram Busca Tabu, Rabadi, Moraga e Al-Salem (2006), que fizeram uso da *Meta-heuristics for Randomized Priority Search* (Meta-RaPS) e com o trabalho de Arnaout, Rabadi e Musa (2010), que usou ACO. Os problemas de teste usados no estudo foram os de Al-Salem (2004), Helal, Rabadi e Al-Salem (2006) e Rabadi, Moraga e Al-Salem (2006). Como resultado, tanto os testes de pequeno e grande porte foram favoráveis, tanto na minimização do *makespan* quanto no tempo de processamento para se

obter a resposta, ao uso da RSA em comparação com demais trabalhos descritos neste parágrafo.

Os autores concluem que a proposta do uso de RSA pode reduzir significativamente o esforço de pesquisa e melhorar a qualidade da solução para os problemas de grande porte. Resultados computacionais indicam que o algoritmo RSA proposto supera algoritmos contrapostos por eles.

Em Gomez-Gasquet, Andres e Lario (2012) é proposto minimizar o *makespan* para o problema da programação da produção dependendo do tempo de *setup* com uma solução baseada no algoritmo genético, no entanto, projetado e desenvolvido com o paradigma de agente de *software*. Ambas as abordagens têm proporcionado resultados bem sucedidos para problemas semelhantes. Ambos são combinados para criar um novo AG com características não vistas no AG tradicional.

Os autores levantaram alguns pontos que serão abordados para fazer as melhorias do MAGSA (*Multi-agent Genetic Scheduling Algorithm*):

- A geração de novos indivíduos baseados na competição local, como ocorre na natureza, e não na competição global como proposto na maioria dos algoritmos genéticos;
- Fortalecer a capacidade de aprendizagem de modo que o ajuste dinâmico de certos parâmetros pode ser feita com base nas circunstâncias de um ambiente em mudança;
- Incentivar a diversidade populacional.

Os conjuntos de testes usados no trabalho de Gomez-Gasquet, Andres e Lario (2012) são obtidos através do uso de uma base de dados publicada originalmente por Vallada, Ruiz e Maroto (2003), e que é uma adaptação do conjunto de dados usado em Taillard (1993). Os autores implementaram dois tipos de MAGSA, o primeiro algoritmo é baseado na tecnologia de multi-agentes, mas não incorpora todas as características de agentes em "sociedade" e, portanto, não obtém vantagem no "trabalho em equipe". No entanto, o segundo tipo de MAGSA presente no trabalho citado incorpora os recursos que o primeiro não incorpora, permitindo, assim, que as vantagens do trabalho em equipe em uma sociedade fossem exploradas.

Os resultados obtidos foram comparados com os trabalhos de Ruiz e Maroto (2006), SAOP dos autores Osman e Potts (1989), Busca Tabu dos autores Widmer e Hertz (1989), o AGH, uma implementação de AG juntamente com heurística NEH, proposta por Reeves (1995) e com a heurística NEH dos autores Nawaz, Enscore e Ham (1983). Os algoritmos AGH e os dois algoritmos MAGSA foram identificados como os mais competitivos para os problemas do tipo $P - 13$ e $P - 3$. Além disso, o segundo algoritmo MAGSA obteve o resultado médio maior em 9 dos 16 conjuntos sempre para os casos mais complexos, enquanto o primeiro algoritmo MAGSA obteve o melhor resultado em apenas três ocasiões das 16.

Outra meta-heurística utilizada para resolver a programação da produção com o objetivo de minimização de *makespan* é o *Self-Organising Migrating Algorithm* (SOMA). A SOMA é da classe de meta-heurística do tipo enxame, a qual tem sido utilizada para resolver problemas de domínio reais. Os autores Davendra et al. (2013) utilizaram tal meta-heurística para solucionar o problema em questão com operações sem tempo de espera.

Os testes foram feitos com os pequenos e médios conjuntos de testes feitos por Taillard (1993). Esses *benchmarks* são compostos por 12 conjuntos de diferentes problemas que variam de 20×5 até 500×20 . Os problemas de pequeno e médio porte podem ser designados a partir de 20×5 até 50×20 , em um total de seis conjuntos de dados e 60 casos de problemas.

Os resultados foram comparados com a meta-heurística F&V desenvolvida por Fink e Vo (2003) e com a meta-heurística híbrida do PSO com VND dos autores Pan, Tasgetiren e Liang (2008). Os resultados mostraram que comparado com a meta-heurística F&V, o SOMA tem uma melhoria significativa para minimizar o *makespan*, perdendo somente em 2 instâncias em um problema de 8. Em comparação ao PSOVND, o SOMA também obteve os melhores resultados, perdendo somente para uma instância do problema.

Em Toader (2014) é proposto minimizar o *makespan* para o problema da programação da produção usando a meta-heurística ACO (Ant Colony Optimization) e PSO (Particle Swarm Optimization), com o objetivo de fazer uma comparação entre as duas técnicas implementadas e avaliar seu desempenho, considerando diferentes cenários de produção.

O trabalho de Toader (2014) apresenta algumas premissas para que o problema da programação da produção seja tratado:

- As máquinas não podem ser substituídas;
- Todas as operações estão disponíveis no momento do tempo zero;
- Cada máquina processa apenas uma operação por vez;
- O tempo de transporte e para preparação da máquina é desconsiderado;
- As restrições em matéria de recursos e espaço de armazenamento, se existir, estão satisfeitos;
- Não é levado em consideração problemas com a máquina.

Os testes foram feitos com cinco diferentes conjuntos de dados. Os resultados experimentais revelaram que PSO foi o melhor método para os cenários de simulação propostos, considerando a simulação de diferentes cenários de produção, tendo em conta a comparação de critérios como: a qualidade da solução (identificado pelo valor da função objetivo) e algoritmo tempo de execução do algoritmo. Os melhores resultados foram obtidos de PSO para os valores do parâmetro de rastro de feromônio que estão mais perto de 0,8 e os valores do parâmetro de velocidade de evaporação de feromônio que são limitados no intervalo de (0,4 ; 0,6).

O autor conclui que, com base nos resultados os dois algoritmos são apropriados para problemas complexos, mas a parte mais importante é identificar os melhores valores para cada parâmetro específico, pois eles podem influenciar muito o desempenho da solução.

2.3.2 Trabalhos Relacionados à Programação Reativa da Produção

Como descrito anteriormente o problema da programação da produção é considerando um problema que contém um conjunto de n produtos $\{P_1, P_2, P_3, \dots, P_n\}$, um conjunto de m máquinas $\{M_1, M_2, \dots, M_m\}$ e um conjunto de operações $O_{n,m}$, tal que cada operação é uma sequência de máquinas para manufaturar um produto P qualquer. Então o problema é encontrar uma permutação dos itens citados para que toda a produção seja feita em um tempo mínimo.

Na prática, em ambientes de manufatura, a produção está sujeita a acontecimentos inesperados. Um grande conjunto desses acontecimentos pode afetar o cronograma da programação da produção. Alguns dos eventos inesperados são: máquinas quebradas, chegada

de pedidos importantes, falta de matéria prima, entre outros tipos (WANG; WANG; WANG, 2011), (MORANDIN et al., 2009), (NIE et al., 2012), (ZHANG; GAO; LI, 2013).

A maioria dos sistemas de manufatura opera em ambientes dinâmicos sujeitos a esses eventos inesperados em tempo real. Portanto, a programação reativa da produção é de grande importância para contornar esses tipos de eventos, para que a produção não pare ou que a produção não sofra um atraso muito grande para a produção total de produtos em um ambiente de manufatura (ZHANG; GAO; LI, 2013).

A programação reativa leva em consideração os eventos em tempo real, o que tem atraído grande atenção de muitos pesquisadores e engenheiros. Os problemas de programação da produção, como já citado anteriormente, são considerados um problema *NP-hard*. Devido à sua característica dinâmica, os problemas de programação reativa da produção são muito mais complexos do que os problemas de programação da produção estática. Portanto, devem ser um problema fortemente *NP-hard* (ZHANG; GAO; LI, 2013).

Para Aissani, Beldjilali e Trentesaux (2009) a programação reativa é capaz de lidar com as incertezas ambientais. Soluções reativas ligam a programação da produção com recursos em tempo real. Na verdade, o processo de alocação de recursos evolui, tornando mais informações disponíveis e, assim, permitindo que as decisões sejam tomadas em tempo real. Naturalmente, a solução reativa não é uma simples função objetivo.

No trabalho dos pesquisadores Morandin et al. (2008a) propõe-se minimizar o *makespan* para o problema da programação reativa da produção em um FMS usando a meta-heurística AG. No trabalho são apresentadas as características do problema que são abordadas, sendo elas:

- As máquinas em tempos determinados e previamente conhecidos;
- O tempo de transporte é considerado no tempo de operação;
- Há *buffers* de armazenamento intermediário com capacidade infinita entre cada máquina;
- O tempo de *setup* está incluso nos tempos de processamento;
- Uma operação não pode ser interrompida depois de seu início.

Os testes foram executados para o problema abordado no trabalho e os resultados encontrados foram comparados com o método de busca A* de Morandin et al. (2007) e com

os trabalhos de Reyes et al. (2002), Yu et al. (2003a) e Yu et al. (2003b), os critérios de comparações são *makespan* e tempo de execução para realizar a busca.

O teste de Wilcoxon foi aplicado para fazer a comparação dos resultados obtidos pelo método proposto pelos autores e com os métodos selecionados para fazer a comparação sobre o *makespan*. Para cada problema foram realizados 50 testes, ou seja, o sistema executou 50 vezes para as mesmas configurações de um problema.

Foram gerados três cenários de teste com tamanhos 3x6, 5x8 e 9x9. Os valores de tempo de produção foram gerados aleatoriamente com variação de 400 – 500 u.t. e os roteiros de produção também foram produzidos aleatoriamente.

As configurações do AG proposto pelos autores foram:

- População de tamanho entre 30 – 40;
- Taxa de cruzamento de 0,8 (80%);
- Taxa de mutação de 0,05 (5%).

Os valores acima foram escolhidos por apresentar bons resultados nos trabalhos de Yeung e Moore (2000), Pongcharoen, Hicks e Braiden (2004) e Pongcharoen et al. (2002).

Para o problema de dimensões 3×6 , os autores constataram que 78% dos resultados obtidos foram iguais ou menores que a média. Também foi constatada uma tendência de melhoria em 88% em comparação aos resultados obtidos por Morandin et al. (2007), e em 78% dos resultados em relação aos resultados obtidos em Reyes et al. (2002), Yu et al. (2003a) e YU et al. (2003b).

Para o problema de dimensões 5×8 , foi possível constatar que 60% dos resultados obtidos foram iguais ou menores do que o valor da média, e que em 92% dos casos, os *makespans* obtidos foram 18% superiores aos apresentados em Morandin et al. (2007). No entanto, a média do tempo de execução do método proposto foi de 0,87 segundos, com um desvio padrão de 0,53 segundos, o qual é 300 vezes menor do que o tempo de execução de Morandin et al. (2007). Os outros métodos não resolveram o problema em tempo viável.

Finalmente, para o problema de dimensões 9×9 , foi observado pelos autores do trabalho que 52% dos resultados obtidos foram iguais ou menores do que o valor médio, embora não tenha sido feita a análise estatística sobre esta observação. A média do tempo de execução foi de 2,05 segundos, com um desvio padrão de 1,23 segundos, o tempo máximo de

7 segundos e tempo mínimo de 0,9 segundos. Foi possível notar que os outros métodos não resolveram o problema em tempo de execução viável.

Com base dos resultados, os autores Morandin et al. (2008a) concluíram que a Programação da Produção com base em algoritmo genético proposto por eles pode ser aplicado para a solução de problemas de manufaturas com pequeno e grande porte, em relação às máquinas e produtos, apresentando soluções com tempo de resposta aceitável, o que não aconteceu com os outros métodos comparados no trabalho.

No trabalho de Kato, Morandin e Fonseca (2009) é apresentada uma modelagem e análise de um sistema de produção específico, incluídos na classe de *job shop*, no qual o objetivo foi tratar o problema de programação da produção reativa, usando o ACO para minimizar o valor de *makespan* em um curto espaço de tempo.

Para tratar o problema os autores descreveram algumas restrições para o problema que foi considerado em seu artigo:

- Cada máquina processa somente uma operação por vez;
- Não é permitido interrupções durante o processamento de um trabalho;
- Todas as máquinas estão disponíveis no tempo zero;
- O tempo do transporte de recurso para as máquinas não é considerado;
- O tempo de *setup* está incluso nos tempos de processamento;
- A execução de cada produto é fixa e não pode ser alterada.

Para fazer a validação da proposta, os resultados do ACO proposto por Kato, Morandin e Fonseca (2009) foram comparados com os resultados encontrados pelo AG produzido pelos autores Morandin et al. (2008a). Para cada cenário de *job shop*, foram feitos 35 testes, ou seja, a abordagem foi executada 35 vezes em cada cenário.

Os cenários propostos no trabalho correspondem a duas instâncias do problema, um com três postos de trabalho e seis máquinas (3×6) e outra instância, com nove empregos e nove máquinas (9×9). Na primeira instância do problema (3×6), cada produto pode ter 2 ou 3 roteiros diferentes que podem conter no máximo cinco máquinas para serem utilizadas. No segundo, cada produto tem exatamente dois roteiros de produção e pode ter de 3 a 7 máquinas para serem produzidos. Ambos os cenários foram gerados aleatoriamente.

Os tempos de produções também foram gerados aleatoriamente com valores entre 400 – 500 u.t.. Foram criados dois tipos de configurações para o ACO, sendo que a primeira configuração usou como parâmetros os seguintes dados:

- Número de formigas: 15;
- Valores de alfa e beta 5 e 0,7 respectivamente;
- Taxa de evaporação: 0,07;
- Número de elites: 3.

Os resultados para essa configuração do ACO se comparado com o AG mostraram que o ACO não foi melhor que o AG considerando o valor da média do *makespan*. Em contrapartida, o tempo para encontrar a solução para o *job shop* 9×9 foi melhor do que o tempo gasto pelo AG.

A segunda configuração do ACO teve como parâmetro os seguintes dados:

- Número de formigas: 20;
- Valores de alfa e beta: 1 e 0,7 respectivamente;
- Taxa de evaporação: 0,1,
- Número de elites: 3.

Com os resultados dessa configuração do ACO os autores mostraram que o estabelecimento de uma baixa influência da feromônios com uma evaporação rápida, os resultados tendem a melhorar e aproximando-se aos resultados obtidos pelo AG.

O trabalho de Morandin et al. (2009) aborda o problema de programação reativa da produção em sistemas de manufatura com recursos compartilhados e utilização simultânea de máquinas e AGVs envolvendo decisões sobre a produção e alocação de recursos de AGVs ao longo do tempo, bem como a escolha de roteiros para a fabricação de cada lote de um produto, definindo o momento da execução de cada estágio. Um critério comum para medir o desempenho da produção é o valor *makespan*. O trabalho apresenta, como meta-heurística de busca, uma abordagem por meio do AG Adaptativo. No trabalho são apresentadas algumas restrições do problema que será abordado, são elas:

- As máquinas em tempos determinados e previamente conhecidos;
- O tempo de transporte é considerado no tempo de operação;

- Há buffers de armazenamento intermediário com capacidade infinita entre cada máquina;
- O tempo de setup está incluso nos tempos de processamento;
- Uma operação não pode ser interrompida depois de seu início.

Os AGVs são usados para transportar os produtos entre as máquinas. O sistema deve adotar algumas regras de despacho para decidir qual AGV vai realizar o trabalho. No trabalho Morandin et al. (2009) foram utilizadas duas regras:

- RV (*Random Vehicle*): usado quando todos os veículos estão disponíveis em *Load / Unload* estação;
- SST / D (*Shortest Travel Time/Distanc*): usado quando há mais de um AGV disponível e eles não estão em *Load / Unload* estação.

Este método foi proposto por Morandin et al. (2008b), o qual envolve algumas regras que ajustam dinamicamente a taxa de mutação e cruzamento de acordo com o desempenho dos operadores genéticos. Os possíveis ajustes de taxas do operador genético são apresentados a seguir:

1. Se o percentual de melhoria entre a média de *fitness* dos filhos e a média de *fitness* dos pais for igual ou maior que 10%, a probabilidade da ocorrência do operador de cruzamento deve ser aumentada para mais 0,05 e 0,005 para as operações de mutação;
2. Se a percentagem de degradação entre a média de *fitness* dos filhos e a média de *fitness* dos pais for igual ou maior do que 10%, a probabilidade de ocorrência de um cruzamento deve ser diminuída de 0,05 e 0,005 para as operações de mutação;
3. Se a porcentagem de melhora / piora entre média de *fitness* dos filhos e a média de *fitness* dos pais tem amplitude modular de menos de 10%, a probabilidade de ocorrer um cruzamento/mutação não deve ser alterada;
4. As taxas devem estar entre 0,5 e 1,0 para operador de cruzamento e entre 0 e 0,10 para operador de mutação.

Assim, o teste de Wilcoxon foi utilizado para comparar os resultados obtidos no trabalho, já que os resultados não pertenciam a uma distribuição normal, sobre a abordagem proposta por Reddy e Rao (2006). O teste de Wilcoxon não foi aplicado sobre o método proposto por Morandin et al. (2008a), porque essa abordagem não considera a programação

AGVs e, portanto, têm valores mais baixos de *makespan* na maioria dos casos. Assim, uma análise comparativa foi realizada a fim de verificar que os resultados não foram muito distantes, quando comparado com Morandin et al. (2008a).

O cenário para teste foi o seguinte 9×9 , onde cada produto tem duas rotas de fabricação possíveis com variações de 5 a 7 máquinas cada uma. Os roteiros de produções para cada produto foram gerados aleatoriamente, bem como os tempos de operação, que variavam entre 400 e 500 u.t.. Por fim, foram utilizados dois AGVs.

Como resultados, os autores constataram que a média do *makespan* encontrado pela proposta do AG adaptativo foi 5572 u.t., com desvio padrão de 148,48. O valor mínimo encontrado foi de 5342 u.t. e o valor máximo foi de 5779 u.t. Para este problema, houve uma tendência de melhoria de 71% se comparado com os resultados obtidos por Reddy e Rao (2006). Assim, através da comparação de valores do *makespan*, a abordagem proposta por Morandin et al. (2009) apresentou resultados melhores e estatisticamente diferentes dos resultados obtidos por Reddy e Rao (2006), com 95% de confiança de acordo com o teste de Wilcoxon.

Quando comparados com os resultados de Morandin et al. (2008a), foi possível observar que a média obtida foi de 10,45% mais elevada, enquanto que a distância máxima em relação ao valor mínimo do *makespan* não era superior a 23,67%, com uma distância média de 19,24% do valor mínimo do *makespan* nos testes.

Os autores concluíram que o método proposto obteve um melhor *makespan* na maioria dos casos testados, quando comparados com resultados de Reddy e Rao (2006). Quando comparado com Morandin et al. (2008a) os resultados não foram muito distantes, como não foi considerado os tempos de transporte e, com resultado de *makespan* inferiores na maioria dos casos. Assim, a proposta dos autores chegou ao objetivo esperado.

No artigo de Nie et al. (2012), foi proposta uma heurística para funcionar sobre uma programação reativa para um *job shop* flexível, onde os trabalhos chegam ao longo do tempo, e propor uma abordagem usando *Gene Expression Programming* (GEP) para a construção de políticas para problemas de roteamento e sequenciamento em uma programação reativa.

Para tratar o problema os autores Nie et al. (2012) listaram algumas restrições que foram seguidas em seu trabalho:

- O tempo de *setup* é incluído no tempo de processamento e é independente da sequência de produção;
- Cada operação não pode sofrer interrupções depois de iniciada;
- Todas as máquinas estão avaliadas todo o tempo;
- Cada operação de um trabalho pode ser processada apenas em uma máquina de cada vez. Cada máquina pode, além disso, processar, no máximo, uma operação de cada vez;
- Há *buffers* infinitos para cada máquina;
- Não é considerado o tempo de transporte entre as máquinas; os produtos estão disponíveis para o processamento de uma máquina imediatamente depois de completar o processamento na máquina anterior;
- Cada produto tem data de lançamento e data de vencimento,
- A ordem de operações para cada produto é predefinida e invariável.

O objetivo da programação é atribuir uma sequência de tarefas atribuídas às máquinas, de modo a minimizar um critério de desempenho específico, desde que satisfaçam todas as restrições adequadas. No trabalho de Nie et al. (2012) os critérios são *makespan*, *arrive date* e *due date*.

Os ambientes testados no trabalho têm como características ser um *job shop f%* implica que, no máximo, $f\%$ do número total de máquinas na loja estão disponíveis para processar uma operação. Nos ambientes de testes de Nie et al. (2012) são utilizadas três configurações de flexibilidade de 100, 50 e 20%. Os números de produto e máquina são 50×10 . O número de operações para cada produto é uniformemente distribuídos entre os valores 4 e 10.

Os tempos de produção são construídos a partir da distribuição uniforme de 1 a 100 u.t.. Os tempos de lançamentos foram gerados usando uma distribuição exponencial. Três níveis de utilização da manufatura são testados 60, 75 e 90%. Foi feito um total de 27 grupos experimentais de simulação.

Os resultados do GEP foram comparados com as seguintes técnicas FIFO (*First In First out*), SPT (*Shortest Processing Time*), EDD (*Earliest Due Date*), SL (*Slack Time*) e MDD (*Modified Due-Date*). Os resultados encontrados, considerando o critério da média do *makespan*, vê-se que GEP surge como sendo o melhor dentre todas as regras. Desta forma,

seu desempenho para busca é significativamente melhor do que a das outras regras. Para níveis de flexibilidade baixa, os resultados encontrados pelo GEP são inferiores aos encontrados pelo SL. No entanto, quando os níveis de flexibilidade são altos, os resultados do GEP são superiores aos demais.

Considerando o critério do fluxo de tempo, o GEP foi significativamente melhor se comparado com as demais técnicas. Os resultados do experimento mostraram que a abordagem baseada em GEP pode construir políticas de programação reativas mais eficientes para problemas de programação complexas em comparação com as outras técnicas.

O trabalho de Nie et al. (2012) trata o problema da programação reativa em um *job shop* flexível com perturbações de máquinas. O objetivo do trabalho foi estabelecer um algoritmo com uma codificação do cromossomo especial e pode fazer uma nova programação da produção (reativa) quando há perturbações de máquina, minimizando o *makespan* ou reparando a programação inicial da produção com o mesmo *makespan*.

Os autores tratam o problema como sendo um *job shop* flexível $n \times m$ onde n é o número de produtos e m o número de máquinas. Tendo que satisfazer as seguintes restrições:

- Cada produto tem uma ordem de processo que especifica uma ordenação em que as operações devem ser executadas e essa ordem não pode ser modificada;
- Existe, pelo menos, uma máquina que pode processar cada operação, às vezes com custos diferentes em diferentes máquinas;
- Uma máquina pode processar apenas uma operação de cada vez;
- Uma máquina continuará processando a operação até que a mesma esteja terminada, ou seja, não são permitidas interrupções durante o processo.

Uma perturbação de máquina é um período de tempo quando um determinado equipamento não está disponível para as operações do processo. Tal perturbação pode vir de muitas causas: quebra da máquina, manutenção, entre outras. Quando uma interrupção ocorre, as operações atualmente programadas na máquina sofrem uma interrupção.

Os autores Nie et al. (2012) descrevem que quando uma interrupção ocorre, um processo de reprogramação ou programação reativa é usado para reparar a programação. A

programação reparada deve evitar a perturbação, ainda seguindo todas as restrições iniciais, minimizando o *makespan* reparado.

Dois algoritmos de referência serão utilizados para comparação: RSR (*Right-Shift Rescheduler*) e *Prescheduler*, o qual reconstrói programações desde o início em cada um dos cenários que sofreram perturbações. Os testes foram feitos em 10 instancias de *job shops*, a saber: 10×6 , 10×6 , 15×8 , 15×8 , 15×4 , 10×15 , 20×5 , 20×10 , 20×10 e 20×15 . Nas 4 primeiras instâncias, os cenários sofreram de 2 a 4 perturbações; os cenários de 5 a 7 sofreram de 4 a 6 perturbações; e os demais cenários sofreram de 2 a 5 perturbações. O algoritmo proposto foi executado 100 vezes em cada instância do problema. As perturbações foram geradas aleatoriamente pelos autores.

Em quase todos os resultados, o AG proposto superou o RSR e o *Prescheduler* na maioria dos cenários, tendo o pior resultado para apenas um tipo de cenário de perturbação, a saber, o cenário 8.

No trabalho de Tuma, Morandin e Caridá (2013) são apresentados dois objetivos de pesquisa, o primeiro consiste na verificação da hipótese de que a hibridação de uma pesquisa global Algoritmo Genético com uma pesquisa local Busca Tabu, podem alcançar melhores resultados no problema de programação reativa Produção. O segundo consiste na investigação de uma nova estrutura do cromossoma AG para o algoritmo ter maior correlação com a variável de minimização do problema, o *makespan*.

Para tratar o problema os autores Tuma, Morandin e Caridá (2013) listaram algumas restrições que foram seguidas em seu trabalho:

- Cada máquina opera apenas um produto de cada vez e cada produto somente pode ser operado apenas por uma máquina de cada vez;
- Um AGV carrega apenas um produto de cada vez;
- Os tempos de produção e transporte são determinados e conhecidos antecipadamente;
- Os tempos de *setup* são incluídos no tempo da produção;
- Um trabalho começa transportar a matéria-prima a partir do setor de carga para o buffer de entrada da primeira máquina de o seu encaminhamento;
- Máquinas e AGVs não param e nem ficam com defeito;

- 2 AGVs são considerados;
- Cada máquina tem um buffer de entrada de tamanho 1 e buffer de saída de tamanho grande o suficiente para suportar qualquer requerimento.

Para a validação da proposta foram usadas seis variantes para comparação e verificação dos resultados. Na primeira e segunda aplicação é usado Algoritmo Genético replicado do trabalho de Reddy e Rao (2006) e Algoritmo Genético Adaptativo replicado do trabalho Morandin et al. (2009). A terceira e quarta implementação é utilizado o Algoritmo Genético e Algoritmo Genético Adaptativo com o cromossomo modificado para provar a hipótese de que um cromossomo com mais aderências ao problema pode proporcionar melhores resultados. O quinto é a implementação da adaptação Algoritmo Genético do trabalho de Morandin et al. (2009) com a assistência da Busca Tabu, para provar a hipótese de que uma pesquisa global com a pesquisa local proporciona melhores resultados para o problema. O sexto é a implementação do Algoritmo Genético Adaptativo com cromossomo modificado e com Busca Tabu mostrando que mesmo com um híbrido, procurar o cromossomo com mais aderências para o problema obtém melhores resultados.

Muitos trabalhos na literatura que utilizam o Algoritmo Genético para a resolução do problema da programação de produção, como no trabalho Morandin et al. (2009), é usado a configuração de que cada cromossomo que mapeia um possível solução, onde cada par de genes indicam uma produção de um produto utilizando um específico roteamento.

O novo cromossomo proposto por Tuma, Morandin e Caridá (2013) é dividido em duas partes, A e B. Parte A indica o roteiro para a produção de cada produto. Já a segunda parte do cromossomo (parte B) indica a sequência de operações que serão executados de acordo com a programação.

Quatro baterias de testes foram realizados, cada execução com 35 resultados. O experimento utilizou um cenário de 9 produtos, com 9 máquinas, 2 roteiros para cada produto e 2 AGVs.

A média do *makespan* encontrada na implementação I (AG usando cromossomo tradicional de Reddy e Rao (2006)) foi de 5686 u.t .. A média do *makespan* encontrada na implementação II (AGA usando cromossomo tradicional de Morandin et al. (2009)) foi de 5572 ut. A média do *makespan* encontrada na implementação III (AG usando o novo

cromossomo) foi 5054,31 ut, com desvio padrão de 170. O valor mínimo encontrado foi de 4697 ut e o máximo de 5512 u.t .. A média do *makespan* encontrada na implementação IV (AGA usando o novo cromossomo) foi 5015,74 ut, com desvio padrão de 152. O valor mínimo encontrado foi de 4689 ut e o máximo de 5191 u.t .. A média do *makespan* encontrada na implementação V (AGA com busca tabu usando cromossomo tradicional) foi 5094,16 ut, com desvio padrão de 155. O valor mínimo encontrado foi 4737 u.t. e o máximo foi de 5262 u.t .. O *makespan* média encontrada na implementação VI (AGA com busca tabu usando o novo cromossomo) foi 4914,97 ut, com desvio padrão de 159. O valor mínimo encontrado foi 4645 u.t. e o máximo foi de 5177 u.t ..

Para obter mais confiabilidade dos resultados foi utilizado o teste de Wilcoxon, para os conjuntos de resultados pode-se dizer que:

- GA (Reddy e Rao (2006)) obteve o pior resultado;
- A combinação de pesquisa global com a pesquisa local (AGA + TS) obtiveram melhores resultados do que a global buscando sozinha (GA ou AGA), em ambos os casos: cromossomo tradicional ou cromossomo novo;
- GA e AGA tanto com o novo cromossomo, ficaram estatisticamente amarrados;
- Comparando os mesmos algoritmos, mas apenas mudando o tipo de cromossomo, nos três casos, os algoritmos com novo cromossomo eram melhor;
- O teste de Wilcoxon confirma que AGA + TS com o novo cromossomo é o melhor de todos os experimentos.

Considerando os resultados de média do *makespan* encontrado em experimentos é possível verificar que a primeira hipótese é válida. A segunda hipótese também apoiada pelos resultados alcançados pelos algoritmos com o novo cromossomo, que foi melhor em todos os testes. O busca global com busca local (AGA + TS) obtém melhores resultados do que busca global sozinha (GA e AGA).

2.4 Considerações finais

Nota-se através dos temas discutidos neste capítulo que os problemas de programação da produção e programação reativa da produção têm atraído a atenção de vários pesquisadores pelo fato de ter um comportamento combinatório e classificado como sendo *NP-Hard*.

Todos os pesquisadores citados utilizaram algum tipo de meta-heurística como: ACO, PSO, GEP e em muitos casos algum tipo de AG modificado, para alcançar melhores resultados, ou mesmo o AG tradicional. Em alguns casos foram utilizados os 12 conjuntos do *benchmark* para *job shops* desenvolvidos por (TAILLARD, 1993) e instâncias do problema para FMS de tamanho 9x9 encontrados no trabalho de MORANDIN et al. (2008a).

Também é possível notar que cada autor cria suas restrições para o problema, na maioria dos casos as restrições são:

- Cada máquina deve processar apenas uma operação por vez;
- Quando é dado o início do processo em uma máquina, a mesma não pode sofrer interrupções;
- A sequência de produção é fixa e não pode ser alterada;
- As máquinas no tempo zero estão sempre disponíveis;
- Os tempos de *setup* estão inclusos no tempo de processamento.

Os trabalhos apresentados neste capítulo levam em consideração a minimização do *makespan* como critério de desempenho para a solução da programação da produção ou reativa da produção. Alguns trabalhos usaram o teste de Wilcoxon, teste de hipótese estatístico para comparar dois tipos de técnicas para problemas que não representam a distribuição normal.

PROPOSTA DO TRABALHO

3.1 Considerações iniciais

Neste capítulo, será descrita a técnica proposta para cumprir os objetivos gerais, os quais são caracterizados pela minimização do *makespan* em um problema de programação reativa da produção usando algoritmo genético com operador transgênico; e os objetivos específicos, os quais estão compreendidos entre desenvolver diferentes ambientes de teste e validação, desenvolver métodos capazes de calcular os genes mais significativos e, finalmente, a implementação completa do *software* da proposta. Desta forma, neste capítulo são apresentadas duas seções que dizem respeito ao algoritmo evolutivo usado no trabalho; uma seção responsável por apresentar o domínio de testes e a metodologia de validação dos resultados obtidos pela aplicação da técnica proposta.

O assunto tratado neste trabalho é o de programação reativa da produção em um sistema de manufatura, no qual há um conjunto de n produtos $\{P_1, P_2, P_3, \dots, P_n\}$ que são produzidos pela manufatura, e tais produtos fazem uso compartilhado de um conjunto de m máquinas $\{M_1, M_2, M_3, \dots, M_m\}$. Para ser manufaturado, cada produto precisa ser processado por um arranjo específico de máquinas, formando, assim, um roteiro de produção. Neste trabalho, é estabelecido que qualquer produto P_i apresente pelo menos um roteiro de produção, sendo que podem existir R_i roteiros diferentes, no qual i corresponde ao índice do produto manufaturado. O objetivo consiste em encontrar uma permutação desses arranjos de forma que toda a produção seja manufaturada em um menor tempo possível (*makespan*).

Em sistemas de manufatura reais é possível que ocorram eventos inesperados que podem atrasar a produção. Para minimizar tal problema, é feito uma reprogramação da produção inicial ou uma programação reativa da produção. Esse problema é reconhecido pela

literatura como sendo um problema combinatório da classe *NP-Hard*, o qual leva um tempo muito elevado para ser solucionado. Em vista disso é recomendado o uso de heurísticas ou meta-heurísticas de busca, entre outros para resolvê-lo eficientemente.

Desta forma, a proposta consiste em fazer uso da meta-heurística conhecida como algoritmo genético com operador transgênico a fim de minimizar o *makespan* de uma programação reativa da produção dado um sistema de manufatura, tendo como hipótese a possibilidade de encontrar um *makespan* satisfatório em um espaço de tempo aceitável.

3.2 Algoritmo Genético com Operador de Transgenia

Como visto nos capítulos anteriores, as meta-heurísticas vem sendo utilizadas para minimizar o valor de *makespan* para programação da produção e programação reativa da produção, uma dessas meta-heurísticas é o AG que usa como inspiração a Teoria da Evolução de Charles Darwin apresentada em sua obra prima “A Origem das Espécies”.

As vantagens de se utilizar um algoritmo evolutivo, segundo Rotshtein e Rakytyanska (2012) e Guo et al. (2010) são:

- (i) O sistema de busca é muito eficiente para encontrar máximos locais e globais, ao fazer uso de operadores de cruzamento e de mutação;
- (ii) AG não necessita de formulações matemáticas rigorosas, devido ao uso de conceitos de Evolução Natural;
- (iii) O AG provê grande flexibilidade para hibridização e uso de operadores e técnicas evolucionárias adicionais.

A principal desvantagem do algoritmo genético encontrada na literatura consiste no alto consumo de recursos computacionais (NIE *et al.*, 2012), (WEI *et al.*, 2010) e (KAZEMI *et al.*, 2012).

Fazendo uso da vantagem (iii) citada anteriormente, os pesquisadores Amaral e Hruschka (2011) e Amaral e Hruschka (2014) desenvolveram um algoritmo evolutivo que utiliza abordagens de transgenia a fim de encontrar resultados satisfatórios em menor tempo. Tal operador foi desenvolvido inspirado na engenharia genética, na qual é possível manipular o material genético de indivíduos adicionando-se características das quais se julga serem as mais importantes. Essa mesma atividade feita por engenheiros genéticos foi acoplada ao AG,

sendo necessário identificar o gene ou os genes mais importantes e, então, obter esses genes do melhor indivíduo e passá-los para um determinado número de outros indivíduos da população, como pode ser observado na Figura 3.1, onde há 6 indivíduos, sendo que o primeiro é classificado como melhor e seus genes mais significativos, representados na cor vermelha, são passados para outros 3 indivíduos.

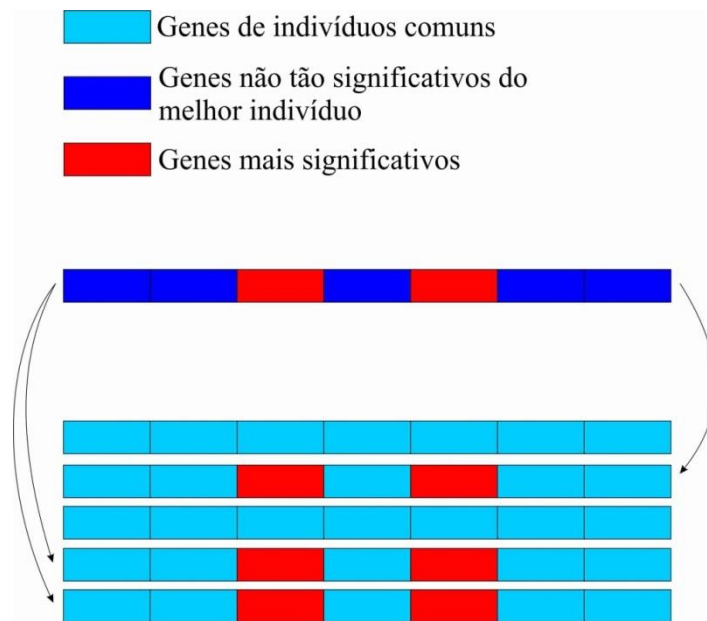


Figura 3.1 Transgenia.

Os passos do algoritmo genético com operador transgênico são semelhantes aos do AG tradicional, exceto pelo processo de transgenia que é adicionado à técnica canônica. No fluxograma da Figura 3.2 são descritas as etapas do AG transgênico, sendo que as atividades em azul são referentes ao operador transgênico.

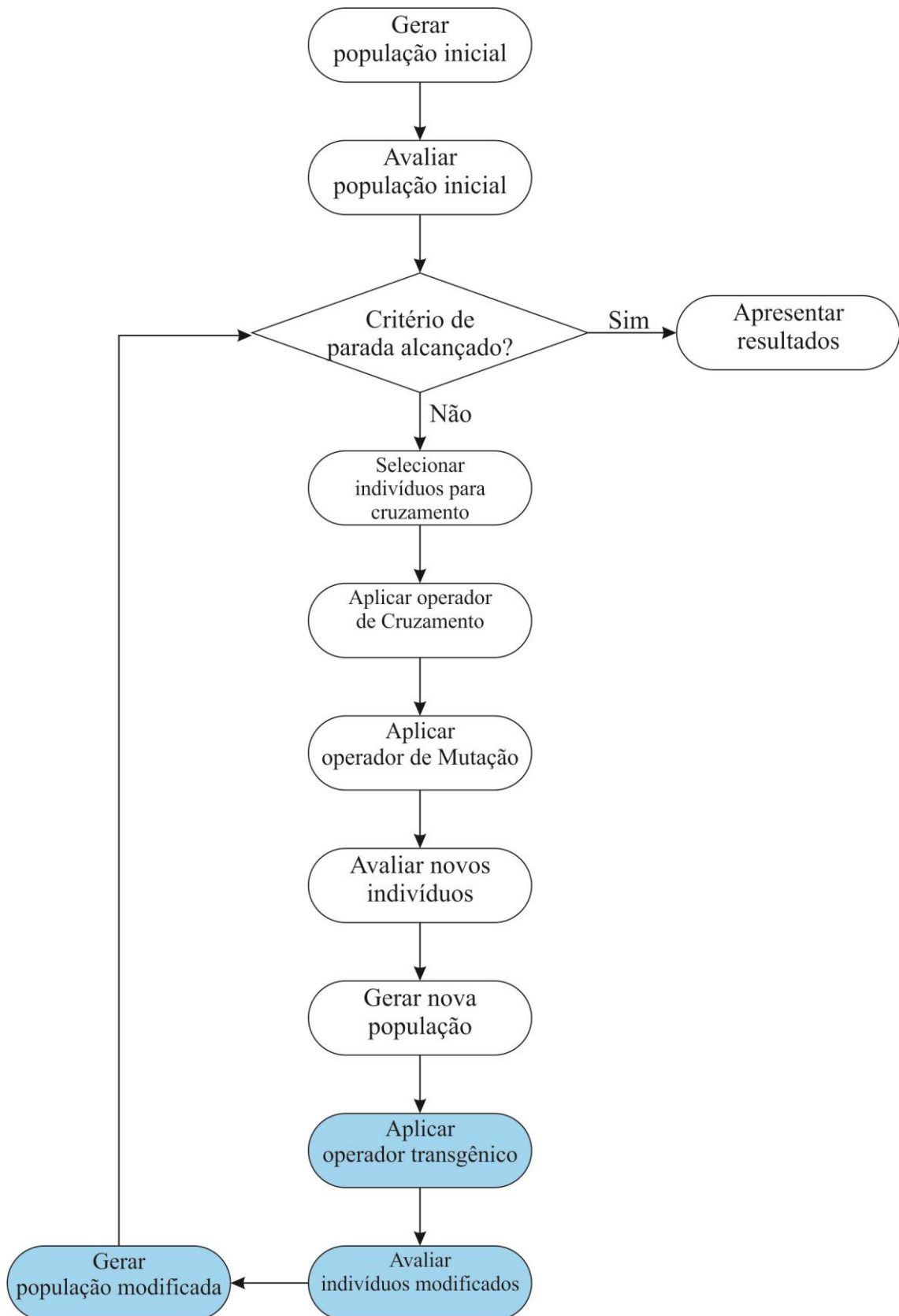


Figura 3.2 Fluxograma do AG com operador de Transgenia.

A seguir, é descrita a construção do AG com operador transgênico de forma que as suas primeiras oito etapas formam o desenvolvimento de um AG canônico (LINDEN, 2012) e os passos subsequentes são responsáveis pela adição do operador transgênico de Amaral e Hruschka (2011) e Amaral e Hruschka (2014):

- Passo 1. Cria-se a população inicial: Neste passo é criado aleatoriamente um número pré-estabelecido de indivíduos para formar o conjunto inicial de elementos a serem avaliados;
- Passo 2. Avalia-se a população inicial: Avaliam-se os indivíduos da população inicial, ou seja, executa-se o cálculo do *fitness* de cada um dos indivíduos gerados no Passo 1;
- Passo 3. Analisa-se um argumento de parada: Neste passo é realizada uma avaliação sobre um argumento de parada. Tal análise pode ser feita baseada em um ponto específico da geração, ou fazendo uso do grau de convergência da população, ou ainda avaliando o grau de diversidade apresentado pela população. Deste modo, o argumento é dependente do problema a ser otimizado. Caso a condição de parada seja alcançada, o algoritmo se encerra e mostra ao usuário o resultado da busca;
- Passo 4. Selecionam-se indivíduos para cruzamento: Neste passo é feita a seleção de indivíduos para o cruzamento, geralmente levando-se em consideração o valor apresentado individualmente pela função de avaliação dos indivíduos presentes na população;
- Passo 5. Executa-se o cruzamento: Nesta etapa do algoritmo é feito o cruzamento entre os indivíduos selecionados no passo anterior.
- Passo 6. Aplica-se o operador de Mutação: Nesta etapa é selecionado um número k de pontos de mutação, na qual $k \in \{1, 2, \dots, N_G - 1\}$ e N_G representa o número de genes a serem modificados por outros valores;
- Passo 7. Avaliam-se os Novos Indivíduos: Logo após o cruzamento e a mutação é necessário que seja feita uma avaliação sobre os indivíduos gerados nos passos anteriores;
- Passo 8. Gera-se uma nova População: Nesse passo é criada uma população, na qual os novos indivíduos são inseridos na população anterior, em alguns casos chamada população de pais. Para realizar tal inserção, existem duas técnicas mais utilizadas, que são:

- a. Inserção Balanceada: Nesta técnica são inseridos na população os melhores N filhos no lugar dos N piores indivíduos da população de pais;
 - b. Inserção Elitista: Nesta técnica só permanecem na nova população os melhores N indivíduos, sejam eles advindos da população antiga ou dos filhos gerados.
- Passo 9. Aplica-se a Transgenia: Neste operador genético é selecionado o “melhor” indivíduo da população intermediária e os seus melhores genes são passados para uma porcentagem da população;
 - Passo 10. Avaliam-se os indivíduos modificados: Após ter sido aplicada transgenia em parte da população é preciso reavaliar os indivíduos que foram modificados, para se obter o conhecimento de melhora ou não do indivíduo;
 - Passo 11. Gera-se a População Modificada: A população modificada conterá os N melhores indivíduos da população anterior e, além disso, será mantida a diversidade da população e uma porção de prováveis soluções ótimas. Ao fim do Passo 11 o algoritmo é redirecionado ao Passo 3.

3.3 Configurações do Algoritmo Genético com Operador de Transgenia

Como mencionado anteriormente, para tornar a execução do AG possível é necessário fixar algumas variáveis que configurem o modo com o qual o AG é executado. Tais variáveis podem ser vistas na Tabela 3.1.

Tabela 3.1 Configurações do AG Proposto.

Variáveis do Algoritmo Genético Proposto ¹	
Tamanho da população	30 – 300
Taxa de cruzamento	80% - 100%
Taxa de mutação	0,05% – 5%
Número de Individuo para Transgenia	10% - 40% da população
Número de Gerações	Conforme critério de condição de parada.

¹ Esses itens serão testados individualmente para encontrar a melhor configuração final.

Além das configurações dadas na Tabela 3.1, é preciso escolher as técnicas para condições de parada, seleção de indivíduos para cruzamento, técnicas de cruzamento, técnicas de mutação e população intermediária.

As técnicas que serão testadas no AG proposto são¹:

- Condições de parada: Neste projeto serão averiguadas as condições de parada por convergência mínima da população e taxa de perda de diversidade. Cada uma dessas técnicas será testada para verificar qual melhor técnica pode ser utilizada para o problema;
- Técnicas de seleção de indivíduos para cruzamento: As técnicas que irão ser testadas são as conhecidas como Técnica da Roleta e Técnica do Torneio;
- Técnicas de mutação: Serão utilizadas neste projeto as técnicas que fazem modificações a partir de um ponto de referência presente no cromossomo e técnicas que fazem uso de K pontos de referência;
- Técnicas para montar população intermediária: No projeto proposto serão testadas a técnica balanceada e a técnica de elitismo.

Todas as técnicas listadas anteriormente serão testadas a fim de encontrar as melhores configurações para serem utilizadas no AG proposto para o problema abordado no projeto.

A codificação do cromossomo do algoritmo genético será composta através de N genes, na qual N representa o número de produtos que a manufatura produz. Tais genes são pares ordenados de produtos e roteiros de produção dos mesmos. É apresentado na Figura 3.3 exemplos de cromossomos de uma manufatura com 7 produtos.

$P_1R_{1,2}$	$P_5R_{5,2}$	$P_2R_{2,3}$	$P_7R_{7,1}$	$P_6R_{5,2}$	$P_3R_{3,1}$	$P_4R_{4,2}$
$P_4R_{4,3}$	$P_7R_{7,2}$	$P_1R_{1,1}$	$P_5R_{5,1}$	$P_3R_{3,3}$	$P_6R_{5,1}$	$P_2R_{2,1}$
$P_7R_{7,3}$	$P_1R_{1,2}$	$P_5R_{5,1}$	$P_3R_{3,1}$	$P_6R_{5,1}$	$P_2R_{2,3}$	$P_4R_{4,2}$
$P_3R_{3,1}$	$P_6R_{5,3}$	$P_4R_{4,3}$	$P_1R_{1,1}$	$P_7R_{7,2}$	$P_2R_{2,2}$	$P_5R_{5,3}$

Figura 3.3 Exemplos de Cromossomos de Sequência de Produção.

Dada a configuração do cromossomo apresentada na Figura 3.3, a técnica de cruzamento utilizada neste trabalho apresenta duas etapas. Em primeiro lugar, é feita a cópia dos genes que representam o produto P_1 até o gene que representa o produto com o número do ponto de cruzamento do *Pai 1* para o cromossomo do *Filho 1*, o mesmo é feito para o *Pai 2* e *Filho 2* como mostra a Figura 3.4. Isto é, inicialmente são transferidas para o i -ésimo filho as posições ocupadas no i -ésimo pai pelos produtos P_1, P_2, \dots, P_k , em que k representa o “ponto de corte” do cruzamento e $i \in \{1, 2\}$.

Em seguida são copiados os genes do *Pai 1* que não foram passados para *Filho 1* para o *Filho 2* e os genes do *Pai 2* que não foram copiados para o *Filho 2* são passados para o *Filho 1*, como mostra a Figura 3.5. Isto é, neste passo do cruzamento são preservadas no i -ésimo filho a ordem com a qual os produtos $P_{k+1}, P_{k+2}, \dots, P_N$ estão dispostos no j -ésimo pai, em que $i \neq j$ e $i, j \in \{1, 2\}$.

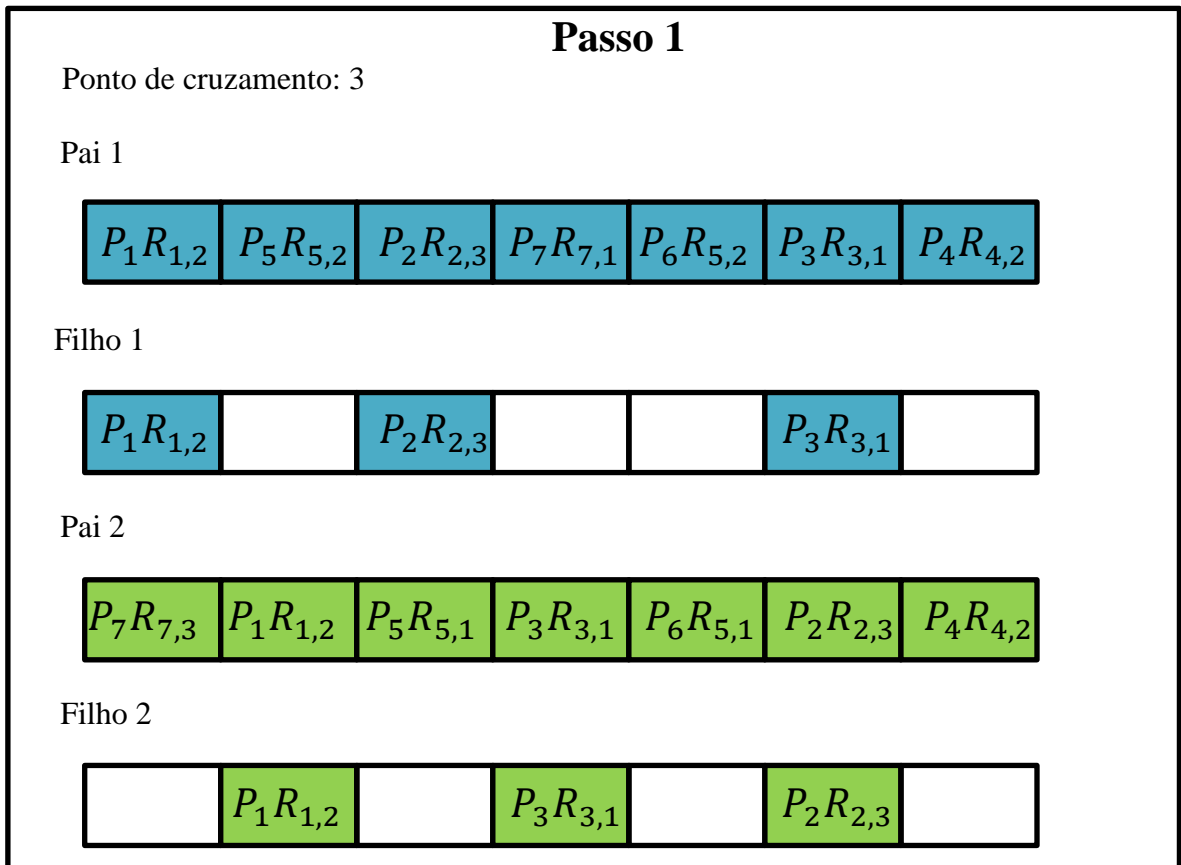


Figura 3.4 - Passo 1 da Técnica de Cruzamento.

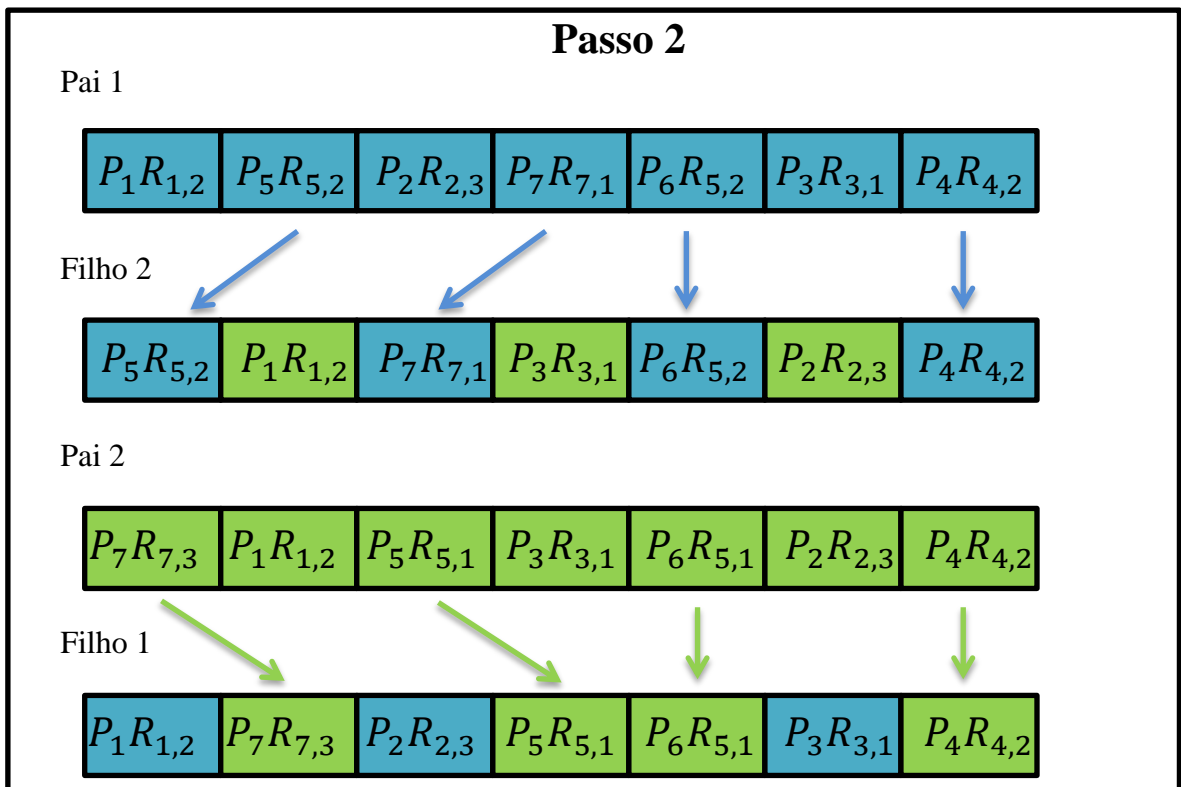


Figura 3.5 Passo 2 da Técnica de Cruzamento.

Uma vez definidas as variáveis, as técnicas e a configuração do cromossomo será definida a função de avaliação (*fitness*) do Algoritmo Genético com operador transgênico. Como já descrito anteriormente, o *fitness* do AG proposto será o *makespan*.

Para calcular o valor do *makespan* é necessário saber o tempo de conclusão de uma operação determinada (O), em que pode ser encontrado por Equação 3.1, na qual O é igual ao tempo de conclusão da operação, j é o índice que representa a máquina e i representa o produto que pertence a operação. T_{ij} é o tempo de transporte do produto i para a máquina j , $T_{P_{ij}}$ é o tempo que o produto P_i leva para ser processado na máquina j e E_{ij} é o tempo de espera que o produto i leva para poder utilizar a máquina j (MORANDIN et al., 2008a).

$$O_{ij} = T_{ij} + T_{P_{ij}} + E_{ij}. \quad (3.1)$$

Além do tempo de conclusão da operação em um produto i é preciso saber o tempo de produção total de um produto i , para isso é preciso somar todas as operações de P_i , como mostra a Equação 3.2 (MORANDIN et al., 2008a).

$$P_i = \sum_{j=1}^m O_{ij}. \quad (3.2)$$

Finalmente, é possível encontrar o *makespan* fazendo uso da Equação 3.3 (MORANDIN et al., 2008a). Observa-se que, neste projeto, o melhor indivíduo é aquele que apresenta o valor *Mkp* mínimo.

$$Mkp = \text{Min}(P_1, P_2, P_3, \dots, P_n) \quad (3.3)$$

3.4 Ambientes de Testes e Validação da Proposta

Foi apresentada nas seções anteriores a técnica que será utilizada para minimizar o *makespan* neste trabalho, a saber, o Algoritmo Genético Transgênico. Foram apresentadas também as configurações desta técnica. Agora serão apresentados os ambientes de testes e a validação da proposta.

Serão utilizados os ambientes de teste propostos nos trabalhos de Morandin et al. (2008a), Morandin et al. (2008b), Morandin et al. (2009), Kato, Morandin e Fonseca (2009) e Kato, Morandin e Fonseca (2010).

Além dos ambientes de teste citados no parágrafo anterior, será implementado um algoritmo que criará ambientes de teste que terão entre 3 – 180 produtos e de 8 – 80 máquinas. Cada ambiente terá uma variação no tempo de produção entre 400 – 500 u.t. (Unidades de tempo) que serão gerados aleatoriamente.

Os resultados encontrados pela proposta por meio do algoritmo genético com operador transgênico serão comparados pelos resultados encontrados pelas técnicas dos trabalhos de Morandin et al. (2008a), Morandin et al. (2008b) e Kato, Morandin e Fonseca (2010).

A validação será feita como mostra a Figura 3.6. As seções em amarelo representam a criação dos ambientes de teste já descritos anteriormente. Nas seções em verde está a execução das técnicas do Algoritmo Genético de Morandin et al. (2008a), Algoritmo Genético Adaptativo, onde as regras de adaptação são encontradas no trabalho Morandin et al. (2008b), ACO de Kato, Morandin e Fonseca (2010) e o Algoritmo Genético com operador transgênico, descrito nas seções anteriores. A seção em azul é referente à obtenção e análise dos dados obtidos através das técnicas em verde. Para tal, serão obtidas as médias, desvios padrões e variâncias de cada teste e em seguida será feito um teste de hipótese para fazer a validação dos resultados obtidos.

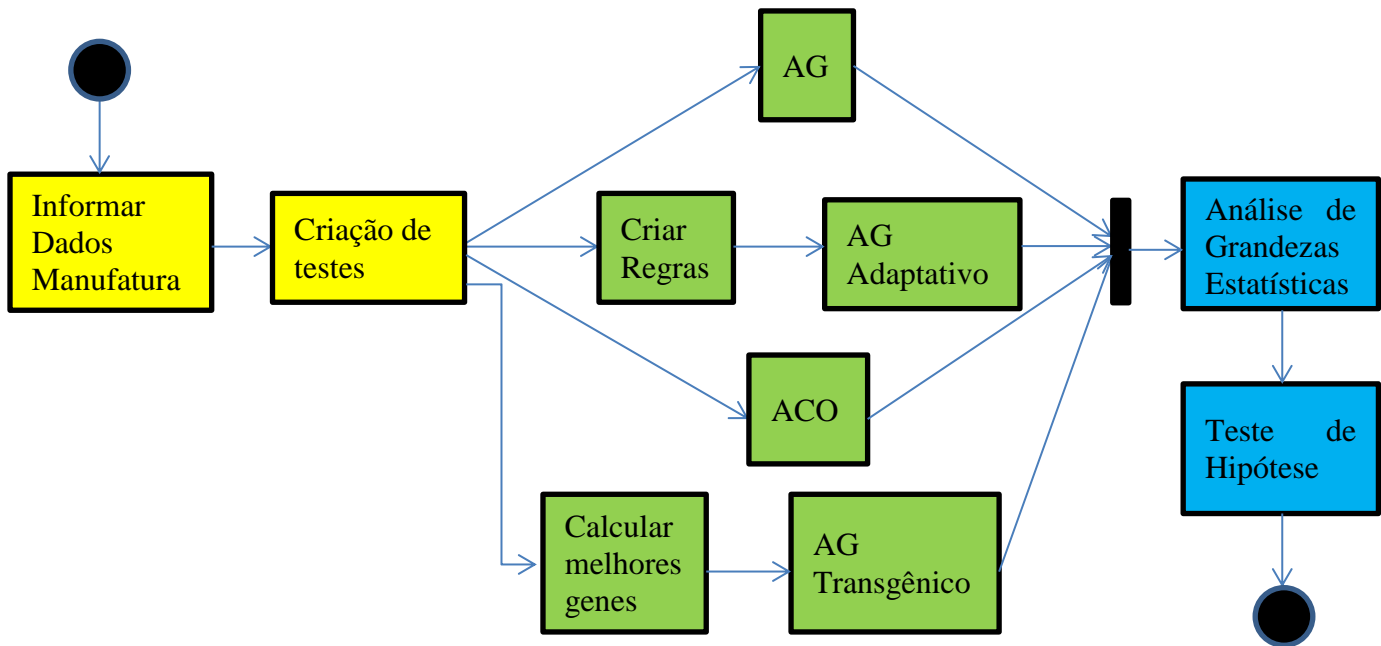


Figura 3.6 Diagrama de Atividade da Validação.

O teste de hipótese de Wilcoxon será utilizado neste projeto para realizar a validação dos dados, o qual também foi utilizado nos trabalhos de Morandin et al. (2008a), Morandin et al. (2008b), Morandin et al. (2009), Kato, Morandin e Fonseca (2009) e Kato, Morandin e Fonseca (2010).

Foi escolhido o teste de hipótese de Wilcoxon pelo fato dos dados não necessariamente apresentarem o comportamento de uma distribuição normal. O teste de hipótese de Wilcoxon é uma das técnicas não paramétricas que assumem pouca ou nenhuma hipótese sobre a distribuição de probabilidade da população no qual se retira os dados.

3.5 Considerações finais

Neste capítulo foi abordada a proposta de minimização de *makespan* da programação reativa da produção usando Algoritmo Genético Transgênico. Com essa proposta, espera-se obter uma minimização das taxas de *makespan* de um FMS proposto em relação a um FMS que não utilize um sistema otimizado por Algoritmo Genético Transgênico.

No próximo capítulo, serão apresentados os resultados obtidos através da proposta comparada a outras técnicas utilizadas na literatura, assim como a técnica responsável por encontrar os genes mais significativos e o software criado para gerar cenários de manufatura.

Capítulo 4

APLICAÇÃO DA ABORDAGEM E ANÁLISE DE RESULTADOS

4.1 Considerações Iniciais

Neste capítulo serão detalhados os procedimentos descritos no capítulo anterior. Para isso, inicialmente é apresentado o programa de gerar cenários construídos neste trabalho, o método proposto para encontrar os genes mais significados e, por fim, a avaliação da abordagem incluindo a descrição dos cenários que foram testados, que correspondem a uma instância do problema como forma de ilustrar cada procedimento, as análises realizadas e a comparação com outras abordagens para o problema em consideração.

4.2 Programa para Gerar Cenários de Sistemas de Manufatura

Para a validação da proposta desta dissertação é necessário realizar testes para que os mesmos sejam analisados e comparados. Os testes que serão usados neste trabalho são sistemas de manufatura que contêm um número de N produtos e M máquinas, e também possuem uma tabela de tempos de produção, além dos conjuntos de roteiros de cada produto.

Para a criação dos cenários de testes, foi desenvolvido um *software* na linguagem C#, apresentado na Figura 4.1. Esse programa recebe como entrada as seguintes informações: quantidade de produto e máquinas, variações do tempo de produção, número de conjuntos de roteiros e número de máquinas para cada roteiro.

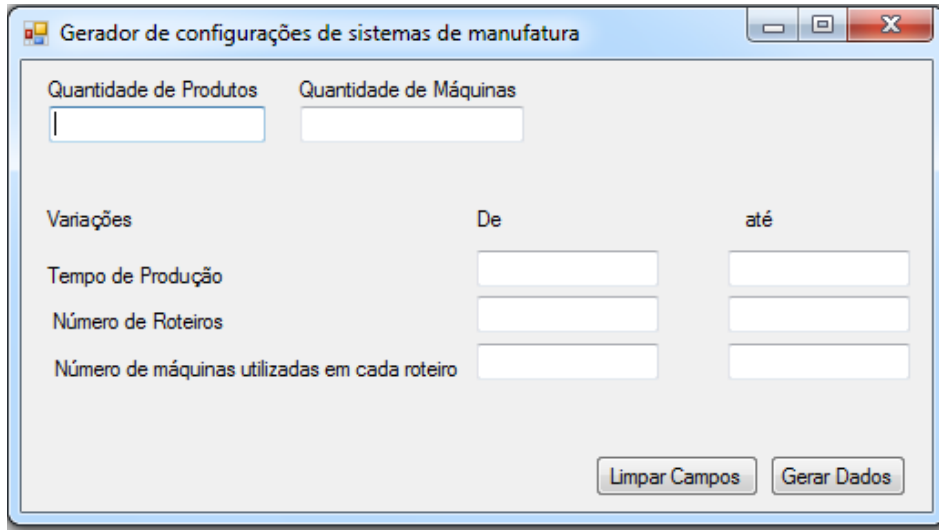
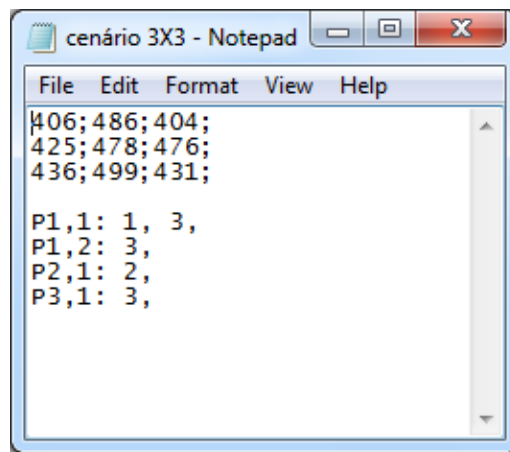


Figura 4.1 - Tela do Software de Geração de Cenários de sistemas de manufatura.

A saída do programa é um arquivo de texto contendo a tabela com tempo de produção de cada produto em uma respectiva máquina, as linhas representam os produtos e as colunas representam as máquinas, por exemplo, o produto P_1 na máquina 1 leva 406 u.t. para ser processado; e os roteiros de produção de cada produto que contém a ordem e quais máquinas devem ser utilizadas para processar determinado produto, como apresentado na Figura 4.2.



```
File Edit Format View Help
406; 486; 404;
425; 478; 476;
436; 499; 431;

P1,1: 1, 3,
P1,2: 3,
P2,1: 2,
P3,1: 3,
```

Figura 4.2 - Saída do software de geração de cenários de sistemas de manufatura.

No caso da Figura 4.2, a tabela com o tempo de produção de cada produto seria a matriz J dada pela Equação 4.1, cujas linhas representam os produtos e as colunas representam as máquinas, e os roteiros de cada produto seriam os apresentados na Tabela 4.1.

Tabela 4.1 Produtos e seus respectivos roteiros gerados pelo *software* criado.

Produtos	Roteiros
P_1	M_1M_3 M_3
P_2	M_2
P_3	M_3

$$J = \begin{bmatrix} 406 & 486 & 404 \\ 425 & 478 & 476 \\ 436 & 499 & 431 \end{bmatrix} \quad (4.1)$$

O programa foi executado para gerar cinco tipos de sistemas de manufatura com os seguintes dados apresentados na Tabela 4.2.

Tabela 4.2 Configurações de cenários a serem gerados.

Cenário	Produtos	Máquinas	Tempo de Produção	Número de Roteiros	Número de Máquinas em cada Roteiro
1	3	8	400 – 500	2 – 5	5 – 7
2	20	8	400 – 500	2 – 5	5 – 7
3	100	40	400 – 500	2 – 5	5 – 7
4	150	40	400 – 500	2 – 5	5 – 7
5	180	80	400 – 500	2 – 5	5 – 7

Os resultados dos cenários gerados podem ser encontrados no link (<https://drive.google.com/file/d/0B4P13Mg3z2sMQV81QIRsQ3ZmRTA/view?usp=sharing>). Assim, o programa apresentado nesta seção cumpriu com o objetivo de gerar os sistemas de manufatura descritos na Figura 4.2. Sendo assim, tal aplicativo é muito útil para o projeto de dissertação descrito neste trabalho.

4.3 Descrição dos métodos para encontrar os genes mais significativos

Como visto no capítulo anterior, para a execução do algoritmo genético com operador transgênico é preciso saber quais são os genes que terão maior importância para o problema, pois esses genes serão transmitidos do melhor indivíduo para outros indivíduos pertencentes à

população, como apresentado na Figura 3.1, visando aprimorar os indivíduos da população e assim, possivelmente, encontrar um melhor resultado. Deste modo, é proposto um método que encontre os melhores genes para o problema abordado neste trabalho, sendo que para cada cenário de manufatura é necessário encontrar outros genes que terão a maior influência para aquele cenário.

Os métodos desenvolvidos neste trabalho simulam o comportamento do algoritmo genético transgênico, no qual cada gene separadamente é simulado como sendo o mais significativo pelo operador transgênico e a simulação é realizada num certo número de gerações N_g .

A cada geração é salva a melhoria dos indivíduos modificados, que é a diferença entre a aptidão de um indivíduo antes da utilização do transgênico e depois a utilização do transgênico. Assim, na i -ésima geração é construído um vetor $g_i \in \mathbb{R}^{N_g}$ que armazena a melhoria média entre os valores dos *fitness* dos N_C indivíduos antes e depois da aplicação de transgenia. Ao fim de todas as gerações da simulação do gene fixado, define-se a matriz de diferenças G , definida pela Equação (4.2) a seguir.

$$G = \begin{array}{l} \text{geração 1} \rightarrow \\ \vdots \\ \text{geração } m \rightarrow \end{array} \begin{bmatrix} | & | & \dots & | \\ g_1 & g_2 & \dots & g_n \\ | & | & \dots & | \end{bmatrix} \quad (4.2)$$

De posse da matriz de dados apresentada na Equação 4.2 é executado um dos dois métodos propostos para determinar os genes mais significativos. Nos tópicos a seguir são apresentados os dois métodos com tal objetivo, o primeiro utiliza uma média ponderada e o segundo utiliza a técnica PCA (*Principal Component Analysis*).

4.3.1 Método utilizando média ponderada de grandezas de estatísticas

Dada a matriz de melhorias G (Equação 4.2) descrita na subseção anterior, extrai-se de cada coluna g_i a média (μ), o desvio padrão (σ) e a maior diferença (amplitude λ) dentre os valores de tal coluna. Esses valores serão incluídos em uma média ponderada (M_p), dada pela Equação 4.3, que é um vetor que apresenta uma coordenada para cada gene presente no cromossomo. Assim, é feita uma classificação dos genes mais importantes do problema abordado.

$$M_p(g_i) = \frac{\sigma(g_i) + (\mu(g_i) \cdot 2) + (\lambda(g_i) \cdot 3)}{6} \quad (4.3)$$

Cada variável na Equação 4.3 foi escolhida com um objetivo, o desvio padrão (σ) foi utilizado no cálculo a fim de verificar a rapidez da convergência das melhorias, uma vez que um desvio padrão alto significa que o mesmo teve uma influência muito significativa em um intervalo de poucas gerações, entretanto um desvio padrão baixo não representa necessariamente uma baixa influência de um determinado gene, sendo que um gene que melhora constantemente uma população pode ser um dos genes mais importantes. A variável média (μ) foi escolhida para contornar tal deficiência, sendo-lhe atribuído, inclusive, um maior peso para realizar este feito. Desta forma, a média avalia se tal gene influencia em uma alta ou baixa melhora no decorrer das gerações. Além disso, é muito importante avaliar a variável “maior diferença” (λ), pois ela é responsável por discriminar genes que corroboram para uma grande melhoria instantânea, isto é, tal grandeza representa o quão influente foi um gene em uma determinada geração.

Deve-se analisar também a quantidade de genes mais significativos que devem ser utilizados. Por meio de testes empíricos, essa quantidade foi determinada por \sqrt{N} , sendo N o número de genes que há no cromossomo. No problema abordado, o número de genes é o número de produtos que devem ser fabricados. Por exemplo, no caso de um problema 9×9 no qual há 9 genes que representam 9 produtos, o número de genes mais significativos a serem considerados são $\sqrt{9} = 3$. Já em um problema 20×10 onde há 20 genes, que são 20 produtos, o número de genes mais significativos a serem considerados são aproximadamente $\sqrt{20} \approx 4.4 \approx 4$. Para chegar a uma conformidade sobre essa estimativa com relação à quantidade, foram feitos testes alternando os números de genes significativos e comparando os resultados obtidos, com isso foi determinada uma quantidade que apresentou bons resultados.

4.3.2 Método utilizando PCA

O PCA, descrito detalhadamente em Jolliffe (2012), é um método de redução de dimensionalidade que funciona por meio de análise de variância. Ele pode ser utilizado para

quantizar influência de indivíduos em uma população. A técnica de PCA analisa quais componentes (atributos de observações) são detentoras das maiores variâncias e realiza uma projeção a um subespaço de dimensão menor. No caso, pode-se dizer que as componentes principais extraídas da matriz G (equação 4.2) via PCA representam os elos surgidos na “evolução artificial”, isto é, a técnica de PCA projeta os dados de G em Y via uma matriz de transformação A . Tal matriz é construída em função da covariância dos atributos dos indivíduos da matriz G a fim de identificar quais atributos são menos redundantes entre si. Deste modo, as componentes principais (Y) de G são obtidas através da multiplicação apresentada na equação (4.4).

$$Y = A \cdot G \quad (4.4)$$

Após obter a projeção Y de G é feito uso da equação (4.5) para estabelecer um grau de influência de cada gene g_j na determinação das componentes principais Y .

$$c_{i,j} = \frac{1}{n-1} \frac{(Y_i^t g_j)^2}{\lambda_i}, \quad (4.5)$$

sendo n o número de indivíduos observados, g_j a j -ésima amostra, Y_i a i -ésima componente principal e λ_i a variância do i -ésimo atributo. $c_{i,j}$ é responsável por descrever o quanto a observação g_j está contribuindo na composição da componente Y_i , ou seja, $c_{i,j}$ representa o quanto o gene g_j contribui com o surgimento do elo de evolução Y_i . Estipula-se como sendo os genes mais representativos, os genes que apresentarem valores de contribuição (equação (4.5)) de maiores magnitudes.

As componentes principais obtidas por meio da técnica de PCA estão diretamente relacionadas às variâncias dos atributos da população avaliada. Desta forma, quanto mais variância um determinado atributo “acumular” em sua disposição, maior será a probabilidade de tal atributo definir mais significativamente uma componente. Para analisar a porcentagem da variância presente em uma determinada componente Jolliffe (2012) associa à componente Y_i o valor $\frac{\lambda_i}{\sum_{j=1}^n \lambda_j}$. Deste modo, caso seja necessário utilizar apenas as componentes que, juntas, acumulam, pelo menos, 85% da variância total da população, então deve-se fazer uso de N componentes, com N dado na equação (4.6).

$$N = \min \left\{ k : \frac{\sum_{i=1}^k \lambda_i}{\sum_{i=1}^n \lambda_i} > 0.85 \right\}. \quad (4.6)$$

Sendo assim, pode-se utilizar tal conceito apresentado na equação (4.6) para se determinar quantos genes devem ser utilizados como mais representativos de forma que o método proposto funcione de maneira satisfatória.

4.3.3 Exemplo de resultados obtidos da execução dos métodos propostos

Os dois métodos descritos na seção 4.3 foram testados em um FMS específico de tamanho 9x9 que foi encontrado nos trabalhos de Morandin et al. (2008a), Morandin et al. (2008b), Morandin et al. (2009) e Kato, Morandin e Fonseca (2010). Além de definir qual era o cenário de teste, foram definidas também as variáveis necessárias para o uso de um dos métodos de encontrar os genes mais significativos, sendo que a representação do cromossomo é a mesma apresentada na Figura 3.3. Na Tabela 4.3, são apresentadas as variáveis do método.

Tabela 4.3 Variáveis para o Método Proposto.

Variável	Valor
Número de gerações testadas	60 gerações
Método de Seleção	Roleta viciada
Método de Cruzamento	Método de cruzamento descrito na Seção 3.3
Taxa de Cruzamento	80%
Taxa de Mutação	5%
Método de geração de nova população	Técnica Balanceada
Taxa de pais selecionados para transgenia	70% da população
Tamanho da população	300
Número de Genes Testados	9 (cada um representa um produto do FMS)

Os resultados foram encontrados utilizando o cenário descrito anteriormente, tal como as configurações do AG e foi utilizado o Método de Média Ponderada e o Método de PCA para classificar os genes e encontrar os mais influentes. O problema de programação que foi testado é um FMS de nove produtos e nove máquinas. Assim, o cromossomo tem nove genes e cada gene foi testado com 60 gerações.

A cada iteração ou geração é feita a seleção de 210 indivíduos que receberam o gene do melhor indivíduo da geração testada. Com ele foi possível construir os vetores com a melhoria de cada gene. Para cada gene foram gerados 12.600 registros e o que gera um total de 113.400 registros. Estes dados podem ser encontrados neste link (<https://drive.google.com/file/d/0B4P13Mg3z2sMS19kUnd4U2hVWG8/view?usp=sharing>).

Na Tabela 4.4 são apresentados os valores de média, desvio padrão e a maior diferença entre melhores valores do vetor de melhoria (v_i), isso para cada gene que represente um produto. Analisando-se apenas a média, pode-se observar que os melhores resultados seriam decorrentes de gene que representam o produto P_5 , P_8 e P_7 . Agora, analisando-se apenas o desvio-padrão, os melhores resultados em ordem decrescente seriam os genes que representam P_5 , P_8 e P_7 . Finalmente, analisando-se apenas a diferença de melhoria entre os melhores valores da amostra, então os genes escolhidos seriam os que representam os produtos P_7 , P_3 e P_5 .

Tabela 4.4 Dados de Melhorias.

Genes	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9
Média	84,770	100,061	101,213	104,598	141,068	107,655	115,971	128,665	109,022
Desvio padrão	166,392	165,343	183,946	183,598	225,869	202,735	203,037	210,687	194,813
Melhores valores	0	0	120	8	39	0	180	0	0

Ressalta-se que não seria interessante que o melhor gene ou gene mais significativo fosse escolhido levando em consideração somente uma variável (média, desvio padrão ou melhor valor). Por isso, utiliza-se um dos métodos propostos para definir com um maior embasamento os resultados. Desta forma, analisando os resultados da Tabela 4.5 obtidos por meio do método da Média Ponderada, pode-se observar que os melhores resultados são decorrentes dos genes que representam o emprego P_7 , P_3 e P_5 . Analisando os resultados da Tabela 4.6 obtidos por meio do método de PCA, pode-se observar que os melhores resultados são decorrentes dos genes que representam o emprego P_8 , P_4 e P_7 .

Tabela 4.5 Classificação de Genes utilizando Média Ponderada.

Genes	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9
Média ponderada	65,232	70,096	127,948	79,221	114,549	80,937	163,776	89,708	79,632

Tabela 4.6 Classificação de Genes utilizando PCA.

Genes	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9
Contribuição na 1ª componente	1,2927	0,4085	0,4310	2,6651	1,4334	1,4424	1,6440	2,7039	1,4185

Portanto, utilizando o algoritmo genético transgênico para resolver o problema de programação reativa para o cenário descrito neste trabalho, levando-se em consideração o

menor valor de *makespan*, podem-se utilizar os genes que representam o emprego P_3 , P_5 e P_7 ou P_8 , P_4 e P_7 .

Conclui-se que os métodos propostos cumprem com o objetivo de encontrar o gene mais significativo para o problema descrito neste trabalho e representa um método genérico para busca de genes mais influentes. Como uma vantagem, os métodos que buscam os melhores genes fazem uso da simulação do algoritmo genético transgênico com a utilização isolada de cada gene para criar a base de dados para análise. Assim, podemos dizer que os dados gerados podem corresponder com a realidade do uso de algoritmo genético transgênico para o problema de programação reativa da produção.

4.3.4 Comparação entre o método que utiliza Média Ponderada e o método que utiliza PCA

Os resultados obtidos da execução dos métodos propostos foram comparados com a finalidade de escolher o mais adequado para a utilização na validação da proposta deste trabalho. Na Tabela 4.7 são apresentados os valores de *makespan* obtidos pelo AG com operador de transgenia, foram utilizados como parâmetro de genes mais significativos os resultados de classificação encontrados através de cada método proposto.

Tabela 4.7 Resultados obtidos na execução do cenário com os genes mais significativos determinados pelo método proposto.

ID do teste	Média Ponderada	PCA	Genes aleatórios
	<i>Makespan</i>	<i>Makespan</i>	<i>Makespan</i>
1	4649	4635	4632
2	4635	4635	4659
3	4670	4669	4670
4	4632	4632	4632
5	4670	4629	4693
6	4632	4670	4632
7	4635	4670	4635
8	4632	4670	4632
9	4632	4640	4635
10	4646	4635	4673
11	4650	4654	4635
12	4632	4635	4677
13	4677	4635	4635
14	4632	4635	4632
15	4635	4654	4632
16	4659	4632	4654
17	4632	4655	4687
18	4632	4635	4693
19	4632	4621	4649
20	4632	4635	4670
21	4632	4693	4640

22	4632	4635	4632
23	4670	4635	4692
24	4633	4683	4797
25	4635	4632	4649
26	4635	4632	4635
27	4632	4635	4649
28	4635	4635	4677
29	4635	4632	4691
30	4635	4632	4670
31	4635	4621	4632
32	4632	4632	4735
33	4635	4632	4735
34	4632	4632	4650
35	4632	4635	4632
Média	4639,6000	4642,0571	4659,1429

Como pode ser observada, a média obtida pelo método que utiliza PCA foi de 4642,0571 com um desvio padrão de 17,4995. O valor mínimo encontrado foi de 4621 e o máximo 4693. Vale ressaltar que nenhuma outra técnica obteve menor *makespan*, o que sugere que os genes responsáveis por gerar tal valor foram considerados como mais significativos. Estes resultados estão muito semelhantes aos do método que utiliza Média Ponderada com a ressalva do menor *makespan* ter sido obtido pela abordagem com PCA, sendo que sua média e desvio padrão são, respectivamente, 4639,6 e 13,2448.

Os valores obtidos através da fixação aleatória dos melhores genes não superam os valores obtidos pela abordagem com PCA e nem os valores obtidos pela com Média Ponderada, devido ao fato da técnica ter apresentado resultado de melhor *makespan* pior que o das técnicas propostas e ter apresentado resultado de pior *makespan* superior ao das técnicas propostas. Além disso, o desvio padrão apresentado por essa abordagem é 35,2188 e a média é 4659,1429.

4.4 Avaliação da abordagem

Para a avaliação da abordagem proposta foram realizados experimentos, e os resultados foram comparados com a abordagem proposta em Morandin *et al.*, (2008a), sendo esta baseada em Algoritmos Genéticos (AG), com a abordagem proposta em Morandin *et al.*, (2008b), sendo baseada em Algoritmo Genético Adaptativo e com a abordagem proposta em Kato, Morandin e Fonseca, (2010), sendo baseada em Otimização por Colônias de Formigas (ACO), todas as propostas foram aplicados ao mesmo problema de programação da produção.

Todas as abordagens foram codificadas utilizando o *software* Matlab e os testes foram realizados em um computador *Core i7* com 2.4GHz e 16GB de memória.

4.4.1 Definição dos cenários para a avaliação

Para a avaliação do comportamento da abordagem proposta, foram executados testes em diversos tamanhos de cenários. Foi utilizado um FMS específico de tamanho 9×9 que foi encontrado nos trabalhos de Morandin et al. (2008a), Morandin et al. (2008b), Morandin et al. (2009) e Kato, Morandin e Fonseca (2010) e foram utilizados mais cinco cenários de tamanhos variados que foram gerados automaticamente pelo software desenvolvido neste trabalho.

Os dados do cenário 9×9 como tempo de produção e roteiros são apresentados na Tabela 4.7 e na Tabela 4.8.

Tabela 4.7 Tempos de manufatura do cenário.

Produtos/Máquinas	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9
P_1	428	423	459	433	467	461	464	455	418
P_2	439	433	487	405	447	497	495	469	439
P_3	453	474	417	447	486	496	459	489	480
P_4	403	436	410	410	400	468	489	439	457
P_5	481	440	477	442	450	468	436	486	435
P_6	446	495	474	448	469	408	454	424	482
P_7	414	457	452	426	493	408	457	497	445
P_8	491	419	435	491	495	452	477	452	408
P_9	458	486	416	454	452	438	484	435	416

Tabela 4.8 Cenário de produção com 9 máquinas e 9 produtos.

Produtos	Roteiros de Produção							
P_1	R_{11}	M_1	M_2	M_4	M_5	M_7	M_9	
	R_{12}	M_3	M_4	M_5	M_6	M_8	M_9	
P_2	R_{21}	M_1	M_2	M_3	M_4	M_5	M_6	M_7
	R_{22}	M_2	M_3	M_5	M_7	M_8	M_9	
P_3	R_{31}	M_4	M_5	M_6	M_7	M_8		
	R_{32}	M_2	M_3	M_7	M_8	M_9		
P_4	R_{41}	M_2	M_3	M_4	M_5	M_6	M_7	
	R_{42}	M_1	M_5	M_6	M_8	M_9		
P_5	R_{51}	M_4	M_5	M_6	M_8	M_9		
	R_{52}	M_1	M_2	M_3	M_5	M_6		
P_6	R_{61}	M_2	M_4	M_5	M_6	M_7	M_8	M_9
	R_{62}	M_1	M_3	M_6	M_7	M_8	M_9	
P_7	R_{71}	M_1	M_2	M_4	M_5	M_6	M_9	
	R_{72}	M_1	M_2	M_3	M_7	M_8	M_9	
P_8	R_{81}	M_4	M_5	M_6	M_7	M_8	M_9	
	R_{82}	M_3	M_4	M_5	M_7	M_8	M_9	
P_9	R_{91}	M_3	M_5	M_6	M_7	M_8	M_9	
	R_{92}	M_2	M_4	M_6	M_7	M_8	M_9	

Os demais cenários foram gerados seguindo as especificações da Tabela 4.6. Os tempos de processamento de cada operação foram gerados aleatoriamente dentro de uma faixa de valores. Também foram gerados os roteiros de todos os produtos. Os cenários gerados foram os que representam os tamanhos 3X8, 20X8, 100X40, 150X40 e 180X80.

Tabela 4.9 Especificação do cenário.

Cenário	Produtos	Máquinas	Número de Roteiros	Números de Operações em cada Roteiro	Tempo de Produção de cada Operação
1	$P_i, i = 1:3$	$M_k, k = 1:8$	{2, 3, 4, 5}	{5, 6, 7}	{400, 401, ..., 500}
2	$P_i, i = 1:20$	$M_k, k = 1:8$	{2, 3, 4, 5}	{5, 6, 7}	{400, 401, ..., 500}
3	$P_i, i = 1:100$	$M_k, k = 1:40$	{2, 3, 4, 5}	{5, 6, 7}	{400, 401, ..., 500}
4	$P_i, i = 1:150$	$M_k, k = 1:40$	{2, 3, 4, 5}	{5, 6, 7}	{400, 401, ..., 500}
5	$P_i, i = 1:180$	$M_k, k = 1:80$	{2, 3, 4, 5}	{5, 6, 7}	{400, 401, ..., 500}

Desta forma, por exemplo, para o cenário 2, para cada produto P_i , com $i = 1, 2, \dots, 20$, tem-se seus respectivos roteiros $R_{i,j}$, nos quais $j = 1, \dots, N_i$ e $N_i \in \{2, 3, 4, 5\}$. Sendo que cada roteiro $R_{i,j}$ apresenta 5, 6 ou 7 operações e cada operação dura de 400 a 500 unidades inteiras de tempo (u.i.t.).

4.4.2 Resultados

Para o primeiro cenário avaliado foi utilizado um FMS específico de tamanho 9x9 que foi encontrado nos trabalhos de Morandin et al. (2008a), Morandin et al. (2008b), Morandin et al. (2009) e Kato, Morandin e Fonseca (2010), e foi definida uma quantidade de 35 testes. Essa quantidade de testes foi definida de acordo com o utilizado nos trabalhos comparados, os quais citam que determinaram essa quantidade baseados em corolários do Teorema do Limite Central, o qual informa que conforme o tamanho da amostra aumenta, a distribuição amostral da sua média aproxima-se cada vez mais de uma distribuição normal. Em geral é considerado que o tamanho da amostra maior do que trinta produz uma boa aproximação. Foram aplicadas as quatro abordagens sem considerar a ocorrência de eventos no cenário. Na abordagem proposta por Morandin *et al.* (2008a) foram utilizados os seguintes valores para os parâmetros: tamanho da população = 30, taxa de cruzamento = 0,8 (80%), taxa de mutação = 0,05 (5%) e 100 gerações como critério de parada.

Na abordagem proposta por Morandin *et al.* (2008b) foram utilizados os mesmos valores de parâmetros que a abordagem de Morandin *et al.* (2008a), a diferença é que nesta abordagem foram introduzidas algumas regras que ajustam dinamicamente a taxa de mutação e cruzamento de acordo com o desempenho dos operadores genéticos. Os possíveis ajustes de taxas do operador genético são apresentados a seguir:

1. Se o percentual de melhoria entre a média de *fitness* dos filhos e a média de *fitness* dos pais for igual ou maior que 10%, a probabilidade da ocorrência do operador de cruzamento deve ser aumentada para mais 0,05 e 0,005 para as operações de mutação;
2. Se a percentagem de degradação entre a média de *fitness* dos filhos e a média de *fitness* dos pais for igual ou maior do que 10%, a probabilidade de ocorrência de um cruzamento deve ser diminuída de 0,05 e 0,005 para as operações de mutação;
3. Se a porcentagem de melhora / piora entre média de *fitness* dos filhos e a média de *fitness* dos pais tem amplitude modular de menos de 10%, a probabilidade de ocorrer um cruzamento/mutação não deve ser alterada;
4. As taxas devem estar entre 0,5 e 1,0 para operador de cruzamento e entre 0 e 0,10 para operador de mutação.

Na abordagem proposta por Kato, Morandin e Fonseca (2010) foram utilizados os seguintes valores para os parâmetros: influência do valor de feromônio = 1, influência do valor heurístico = 2, taxa de evaporação = 0,02, número de formigas = 50 e número de iterações = 75.

Na abordagem proposta neste trabalho foram utilizados os seguintes valores para os parâmetros: tamanho da população = 300, taxa de cruzamento = 0,8 (80%), taxa de mutação = 0,05 (5%), taxa de transgenia = 0,4 (40%), produtos fixos para transgenia (genes mais significativos) P_3 , P_5 e P_7 , e convergência da população como critério de parada. Os produtos fixos para transgenia foram selecionados através do método de encontrar os genes mais significativos, e os demais valores foram estabelecidos através da análise de resultados de vários testes.

Na Tabela 4.9 são apresentados os resultados de cada teste. As colunas indicam o número do teste, o *makespan* obtido e o tempo de resposta calculado em segundos por cada abordagem. Os valores de *makespan* estão expressos em unidades de tempo (u.t).

Tabela 4.10 Resultados obtidos na execução do cenário.

ID do teste	Proposta		MORANDIN (2008a)		MORANDIN (2008b)		KATO (2010)	
	Makespan	Tempo (seg.)	Makespan	Tempo (seg.)	Makespan	Tempo (seg.)	Makespan	Tempo (seg.)
1	4679	2,6406	4945	3,0938	5073	4,7969	4945	10,6094
2	4679	2,2188	4946	3,0781	4962	4,7188	4702	10,0625
3	4635	3,0625	5063	3,1094	5030	5,2344	4640	11,7656
4	4635	2,3125	5035	3,0625	5020	4,7813	4693	12,1563
5	4635	2,5938	4693	3,1250	5107	4,7188	4635	12,2813
6	4632	2,8281	4976	3,0938	4959	4,7344	4735	12,1094
7	4635	1,9219	4972	3,0781	4997	4,7188	4704	12,5781
8	4638	2,7813	5068	3,1094	4741	4,7344	4674	12,5000
9	4635	3,3594	4860	3,0625	4970	4,7813	4670	12,3125
10	4632	2,6875	4677	3,0781	4693	4,7500	4710	12,0469
11	4632	3,0156	4705	3,0625	5063	4,7813	4704	13,0938
12	4632	2,3438	5020	3,0469	5075	4,7813	4657	11,9844
13	4632	3,4844	4879	3,0469	4635	4,8281	4736	12,5781
14	4654	3,6719	4946	3,0313	4934	4,6875	4692	12,2500
15	4635	3,1719	4674	3,0625	4651	4,7656	4704	12,1406
16	4635	3,2969	4944	3,0781	4710	5,0469	4632	12,1563
17	4632	2,8906	4929	3,0938	4967	4,6875	4677	11,8906
18	4672	3,7813	5160	3,0781	4945	4,7188	4688	12,0469
19	4635	2,9063	4946	3,3281	4988	4,6875	4688	12,1250
20	4632	2,0781	4848	3,0625	5070	4,7656	4632	12,9063
21	4632	2,7031	5010	3,1094	4677	4,7656	4670	12,9219
22	4635	2,6875	5066	3,0625	5033	4,7344	4688	12,5156
23	4635	2,8125	5177	3,0781	5124	4,7500	4674	12,6563
24	4632	2,2031	4984	3,0938	4977	4,7188	4677	12,7031
25	4632	3,8281	4988	3,0469	4791	4,6875	4686	12,5000
26	4621	3,1094	4929	3,0938	5000	4,6875	4668	12,8906
27	4635	2,9531	4848	3,0781	4848	4,7031	4649	12,8906
28	4670	3,3125	4903	3,0469	4981	4,8750	4937	12,7813
29	4635	2,8438	4977	3,0938	4677	4,7813	4714	12,7031
30	4691	3,6563	4984	3,0781	4944	4,7500	4649	12,5000
31	4705	3,6875	5145	3,0625	4763	4,7500	4692	12,6719
32	4693	2,8125	4697	3,0781	4980	4,7813	4681	15,0781
33	4673	3,0313	4993	3,0938	4934	4,8438	4752	13,6719
34	4632	3,7969	4945	3,1250	4945	4,7813	4688	12,5156
35	4635	2,3438	4958	3,0938	5068	4,7344	4746	12,5625
Média	4645,2000	2,9383	4939,7143	3,0861	4923,7714	4,7737	4699,6857	12,4330

Como pode ser observado, a média obtida pela abordagem proposta foi de 4645,2000 com um desvio padrão de 19,5919. O valor mínimo encontrado foi de 4621 e o máximo 4705.

A média obtida por Morandin et al. (2008a) foi de 4939,7143, com um desvio padrão de 129,6313 devido à característica menos direcionada do método de busca utilizado, e os valores variam entre o mínimo de 4674 e o máximo de 5177.

A média obtida por Morandin et al. (2008b) foi de 4923,7714, com um desvio padrão de 144,1819, e os valores variam entre o mínimo de 4635 e o máximo de 5124.

A média obtida por Kato, Morandin e Fonseca (2010) foi de 4699,6857, com um desvio padrão de 67,4204, e os valores variam entre o mínimo de 4635 e o máximo de 4945.

A proposta apresentou melhores resultados que os demais trabalhos propostos por Morandin et al. (2008a), Morandin et al. (2008b) e Kato, Morandin e Fonseca (2010), tanto no valor mínimo e máximo, quanto no desvio padrão e no tempo médio de execução do algoritmo (Tabela 4.10).

Tabela 4.11 Resumo dos resultados obtidos na execução do cenário.

Algoritmo utilizado	Média	Desvio Padrão	Menor Valor	Maior Valor	Tempo médio de execução
Proposta	4645,2000	19,5919	4621	4705	2,9383
MORANDIN (2008a)	4939,7143	129,6313	4674	5177	3,0861
MORANDIN (2008b)	4923,7714	144,1819	4635	5124	4,7737
KATO (2010)	4923,7714	67,4204	4635	4945	12,4330

Finalmente, tem-se como indivíduo “vencedor” o cromossomo formado pelos seguintes genes: $P_{9,1}$, $P_{7,1}$, $P_{3,2}$, $P_{8,1}$, $P_{4,1}$, $P_{6,2}$, $P_{5,2}$, $P_{2,1}$ e $P_{1,1}$. Isto é, o cromossomo (Figura 4.3) do qual se obtém o *makespan* de 4621 u.t..

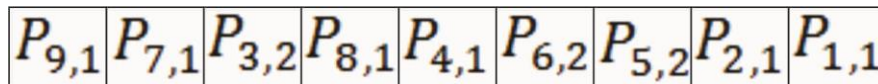


Figura 4.3 – Cromossomo cujo *makespan* resulta em 4621 u.t..

Foi escolhido o teste de hipótese de Wilcoxon pelo fato dos dados não necessariamente apresentarem o comportamento de uma distribuição normal. O teste de hipótese de Wilcoxon é uma das técnicas não paramétricas que assumem pouca ou nenhuma hipótese sobre a distribuição de probabilidade da população no qual se retira os dados.

Foram testadas duas hipóteses, na primeira foi testado se o algoritmo proposto comparado individualmente com cada algoritmo AG, AG Adaptativo e ACO apresenta resultados maiores (Figura 4.4, Figura 4.6 e Figura 4.8) e posteriormente se apresenta resultados iguais (Figura 4.5, Figura 4.7 e Figura 4.9). Com o primeiro teste de hipótese pode ser concluído que o algoritmo proposto não apresenta resultados maiores e com o segundo teste de hipótese pode ser concluído que ele não possui resultados iguais, então conclui-se que ele possui resultados menores.

Aplicando o teste de Wilcoxon, é possível concluir com 95% de confiança, que o método proposto apresenta resultados estatisticamente melhores do que os resultados obtidos pelos outros métodos testados.

Tabela 4.12 Teste de hipótese do AG Proposto X AG.

Variáveis do Algoritmo Genético Proposto	
V	0
P-valor	1
Hipótese nula	1
Método	Wilcoxon signed rank test
(Pseudo) Mediana	-307,5
Intervalo de Confiança - Limite Inferior	-336,5

Tabela 4.13 Teste de hipótese do AG Proposto X AG.

Variáveis do Algoritmo Genético Proposto	
V	0
P-valor	5,82076609134675E-11
Hipótese nula	0
Método	Wilcoxon signed rank test
(Pseudo) Mediana	-307,5
Intervalo de Confiança	95%
Limite Inferior	-343,5
Limite Superior	-262,5

Tabela 4.13 Teste de hipótese do AG Proposto X AG Adaptativo.

Variáveis do Algoritmo Genético Proposto	
V	0
P-valor	0,999999881507812
Hipótese nula	1
Método	Wilcoxon signed rank test
(Pseudo) Mediana	-293,676070939136
Intervalo de Confiança - Limite Inferior	-337,499959824489

Tabela 4.14 Teste de hipótese do AG Proposto X AG Adaptativo.

Variáveis do Algoritmo Genético Proposto	
V	0
P-valor	2,58649971906392E-07
Hipótese nula	0
Método	Wilcoxon signed rank test
(Pseudo) Mediana	-293,676070939136
Intervalo de Confiança	95%
Limite Inferior	-345,499959859046
Limite Superior	-221,500033581154

Tabela 4.15 Teste de hipótese do AG Proposto X ACO.

Variáveis do Algoritmo Genético Proposto	
V	30
P-valor	0,999998542471206
Hipótese nula	1
Método	Wilcoxon signed rank test
(Pseudo) Mediana	-46,0000174623007
Intervalo de Confiança - Limite Inferior	-58,0000462738533

Tabela 4.16 Teste de hipótese do AG Proposto X ACO.

Variáveis do Algoritmo Genético Proposto	
V	22
P-valor	4,01972439800288E-06
Hipótese nula	0
Método	Wilcoxon signed rank test
(Pseudo) Mediana	-49,0000622812156
Intervalo de Confiança	95%
Limite Inferior	-63,5000540063563
Limite Superior	-35,0000198048552

Foram construídos gráficos de convergência do algoritmo proposto e dos três algoritmos testados, sendo que no eixo y estão dispostos os valores *makespan* alcançados e no

eixo x é apresentado o número de iterações. O objetivo desses gráficos é demonstrar o quão rápido ou lento o algoritmo apresenta-se para encontrar uma solução ideal. Como pode ser observado nos gráficos de convergência do algoritmo, o AG com operador transgênico apresentado na Figura 4.4 por ser direcionado através da aplicação de transgenia dos genes mais significativos, ele tem uma convergência com menos iterações se comparado a um simples AG apresentado na Figura 4.6, a um AG Adaptativo apresentado na Figura 4.8 e ao ACO apresentado na Figura 4.10. Com isso nota-se a vantagem em uma convergência mais rápido que um AG com operador de transgenia pode oferecer.

Foram construídos também, gráficos que apresentam a média da população em cada iteração feita pelo algoritmo, com o objetivo de apresentar a evolução da população do algoritmo conforme é passada cada iteração, o AG com operador de transgenia apresentado na Figura 4.5, o AG apresentado na Figura 4.7, o AG Adaptativo apresentado na Figura 4.9 e o ACO apresentado na Figura 4.11.

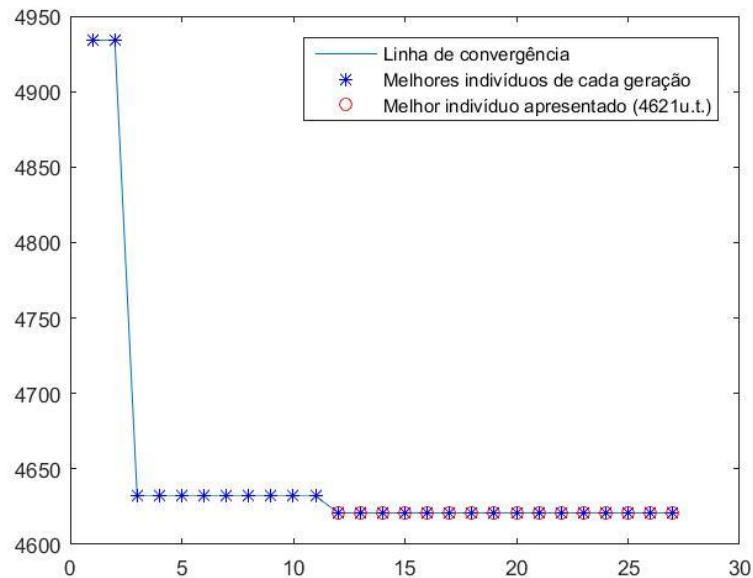


Figura 4. 4 – Gráfico de convergência do algoritmo proposto.

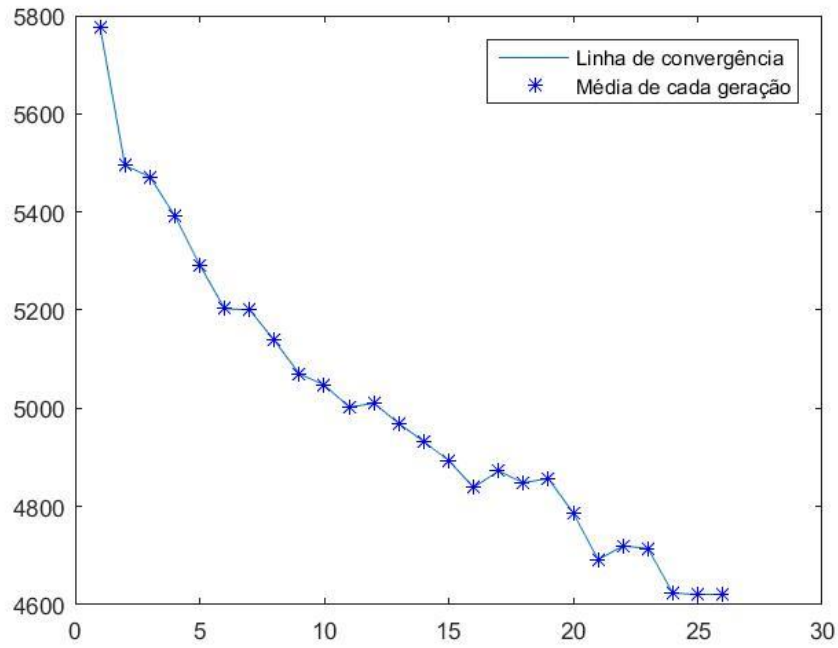


Figura 4.5 – Gráfico da média da população em cada iteração do algoritmo proposto.

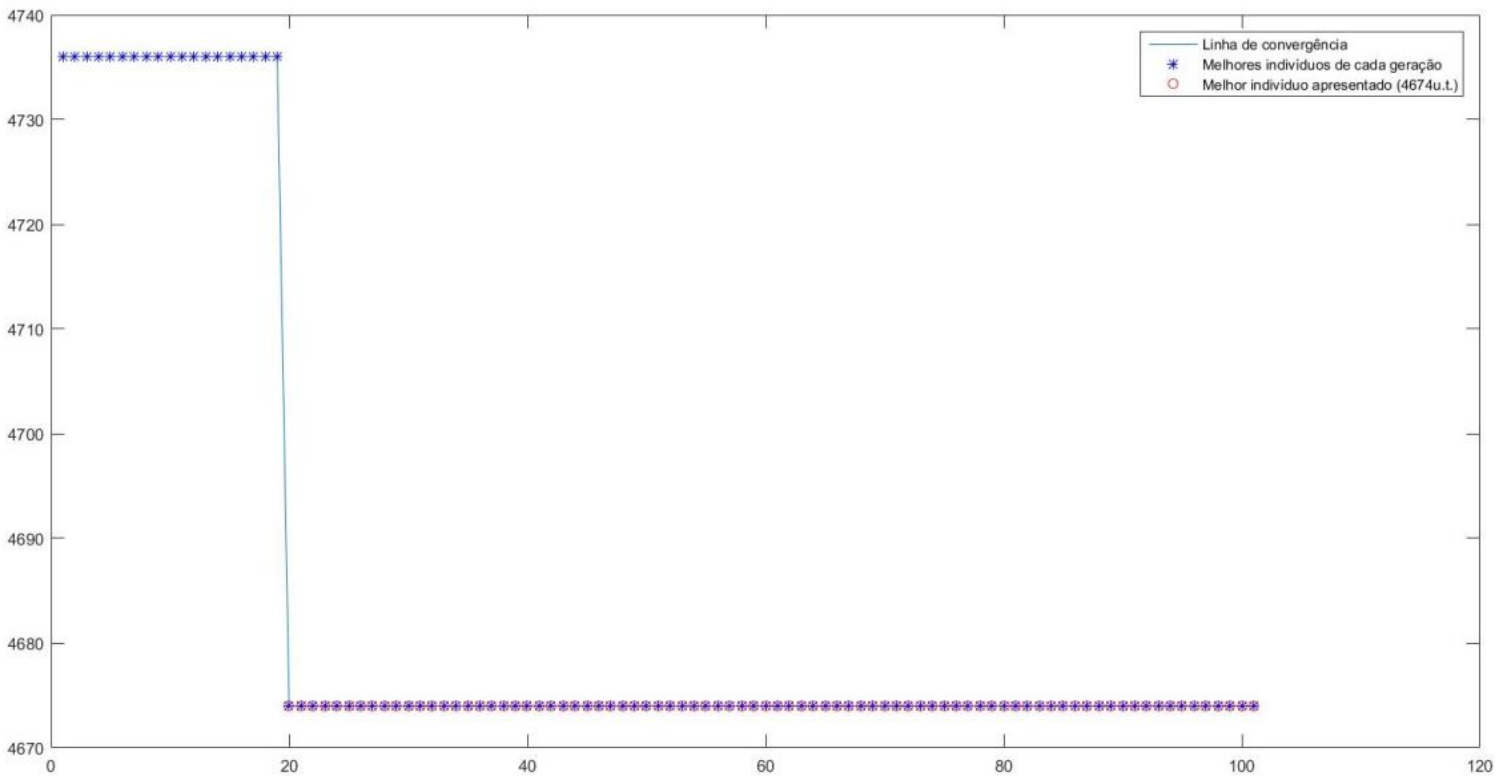


Figura 4.6 – Gráfico de convergência do algoritmo genético sem operador de transgenia.

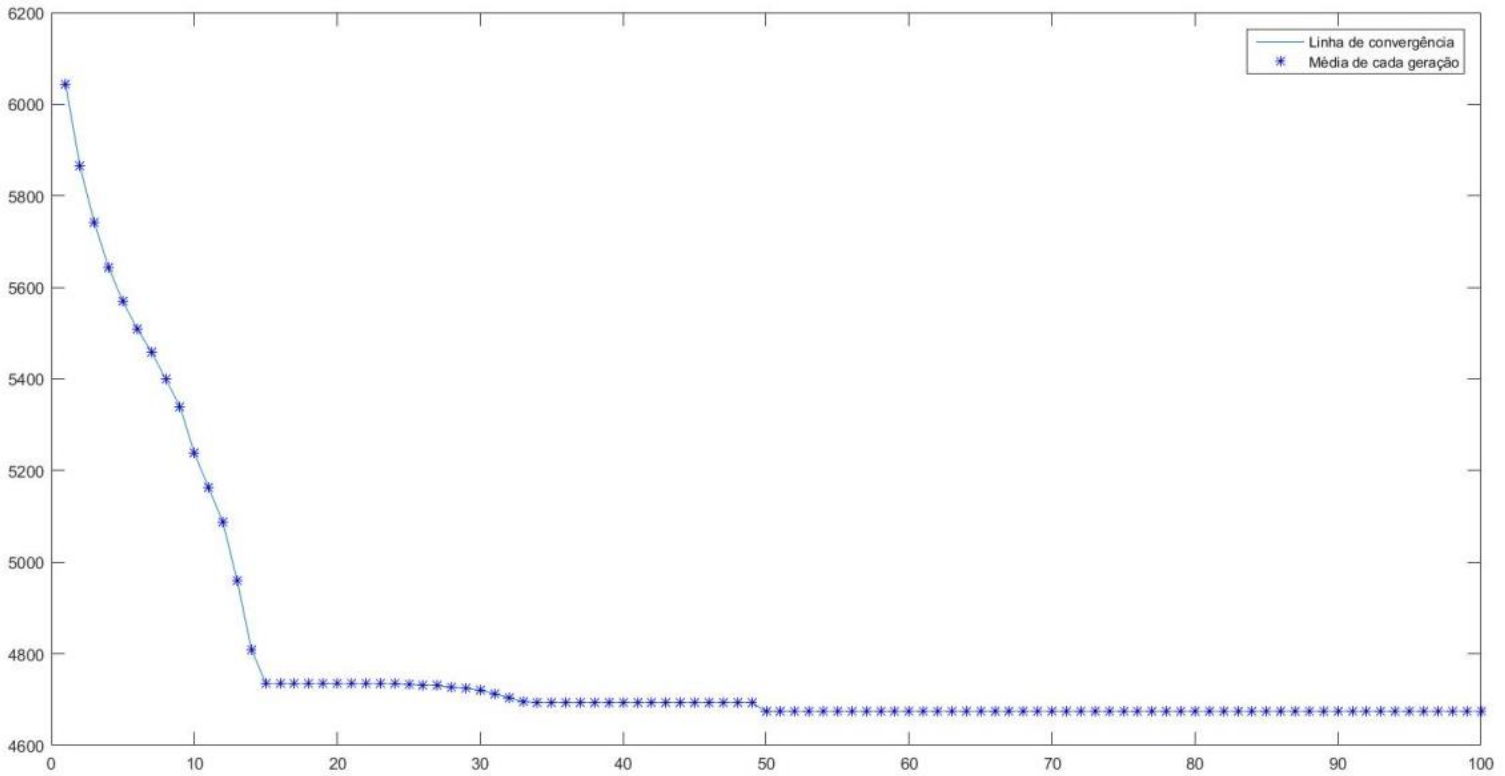


Figura 4.7 – Gráfico da média da população em cada iteração do algoritmo genético sem operador de transgenia.

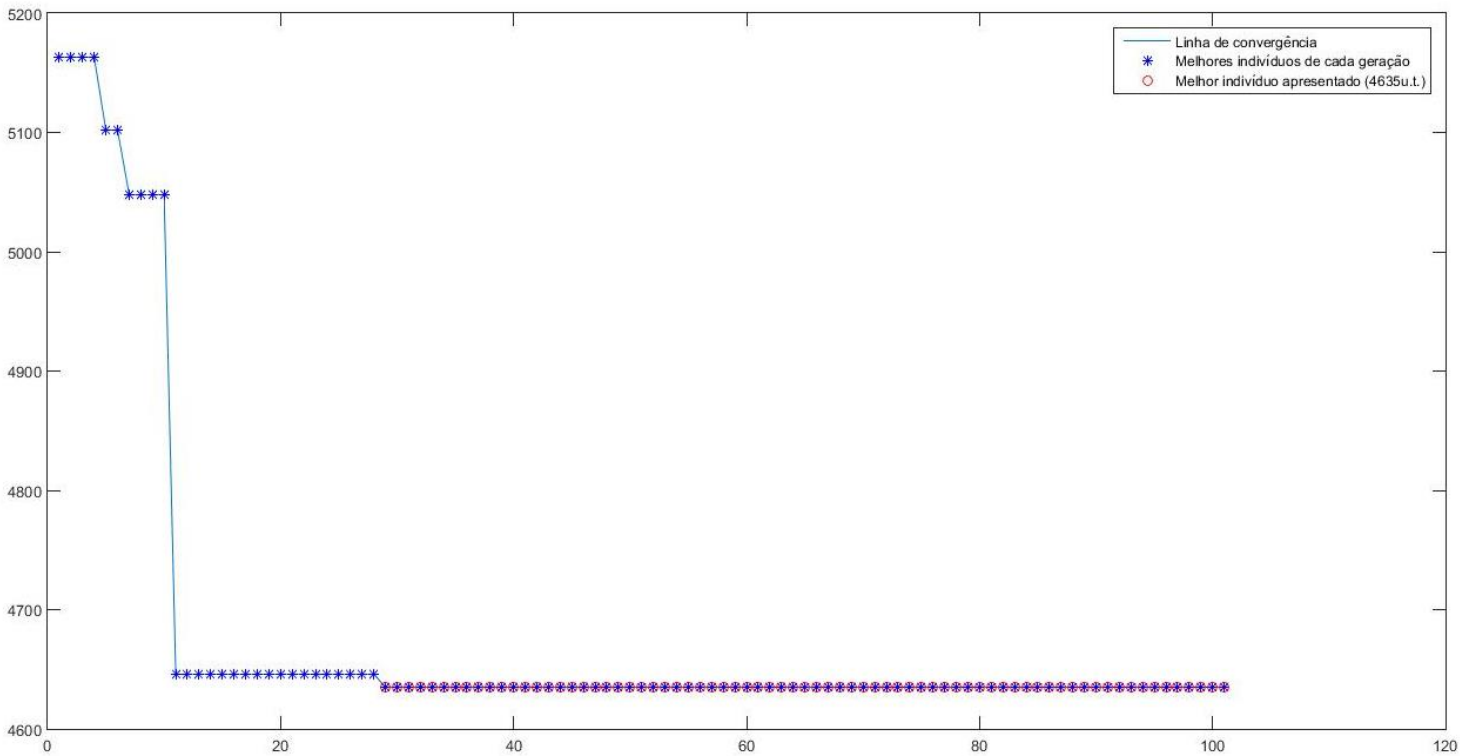


Figura 4.8 – Gráfico de convergência do algoritmo genético adaptativo.

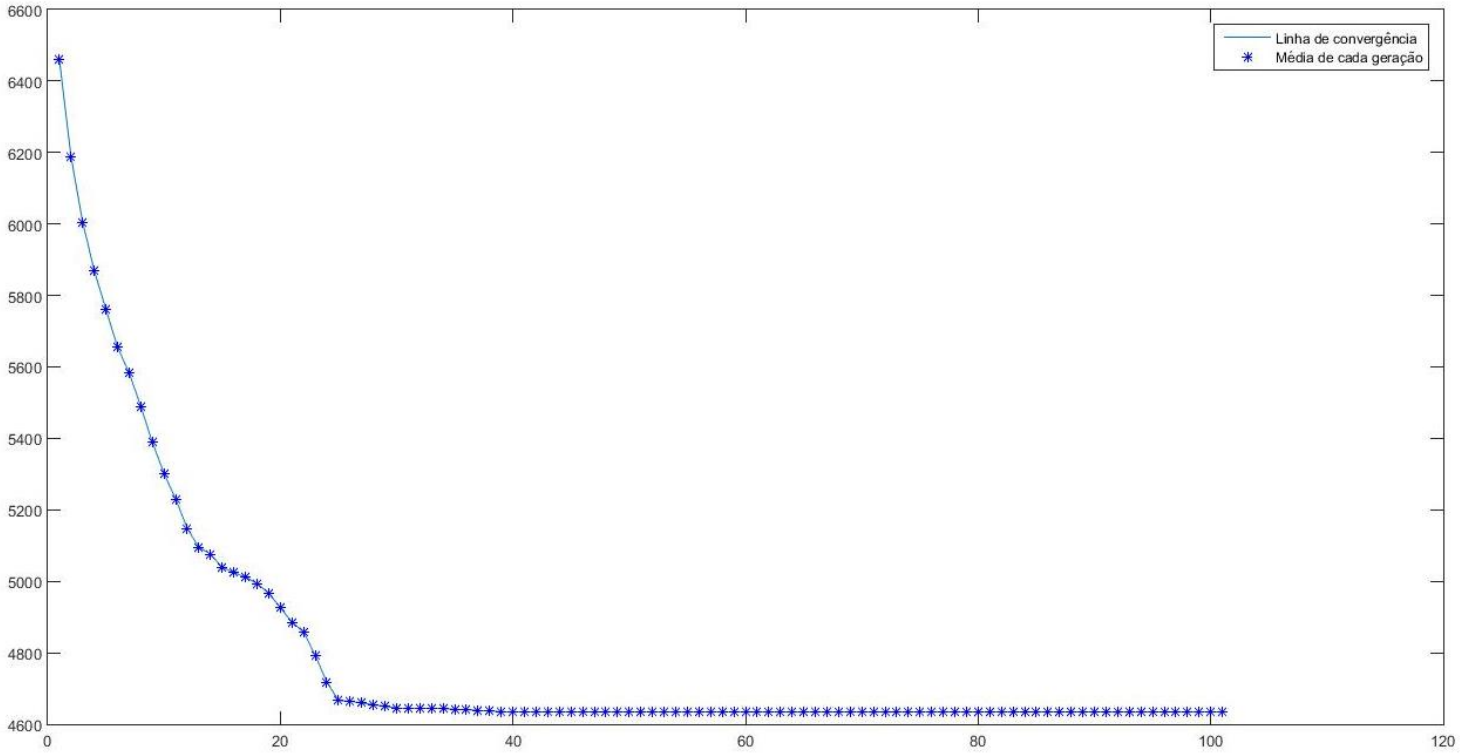


Figura 4.9 – Gráfico da média da população em cada iteração do algoritmo genético adaptativo.

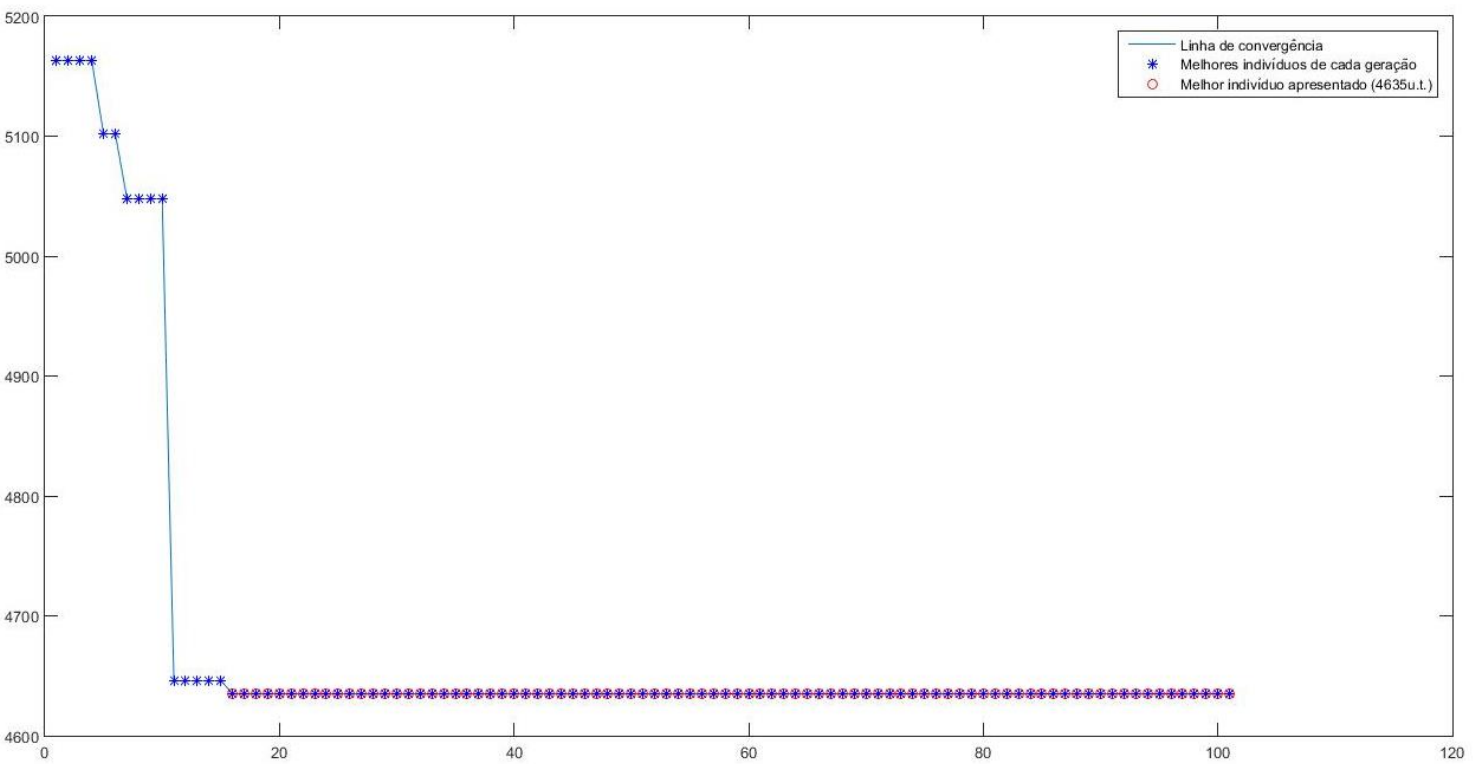


Figura 4.10 – Gráfico de convergência do algoritmo ACO.

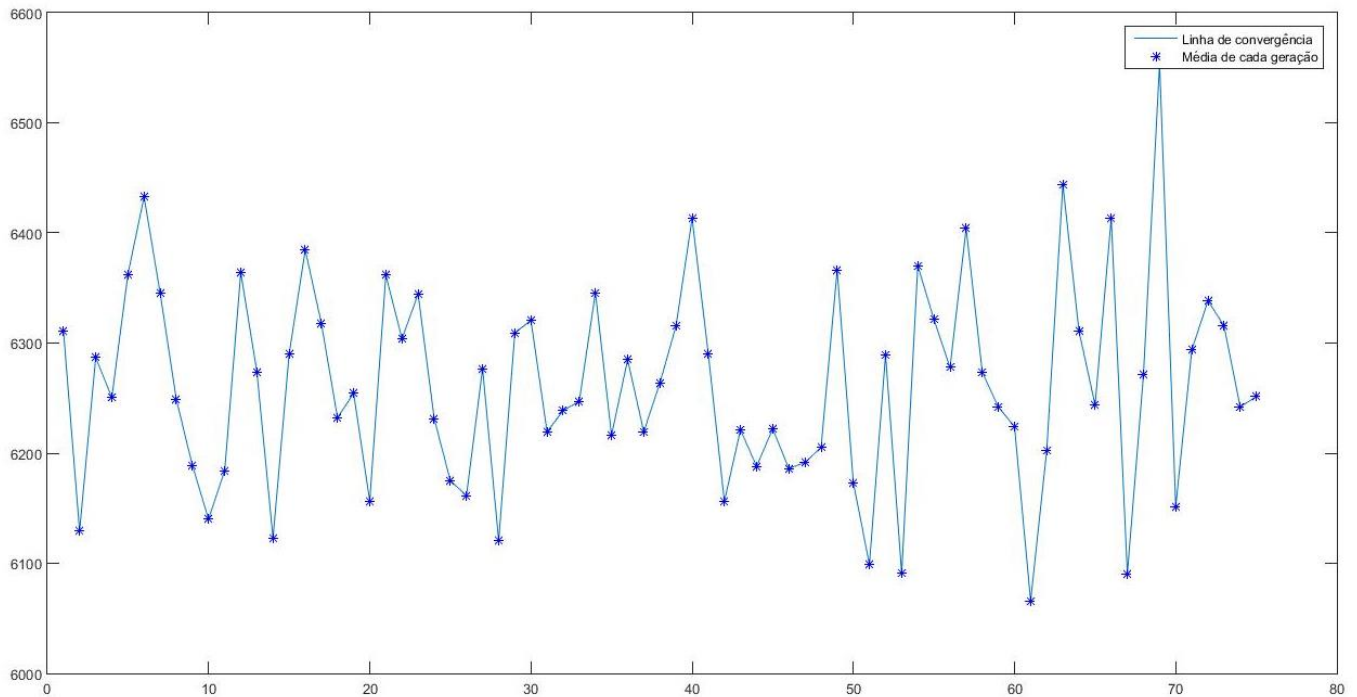


Figura 4. 11– Gráfico da média da população em cada iteração do ACO.

Analisando os gráficos de convergência, nota-se que o algoritmo genético com operador transgênico apresenta uma convergência com menor número de iterações se comparado com um AG tradicional, devido ao seu direcionamento através dos genes mais significativos. É constatado também que o algoritmo proposto apresenta uma boa evolução da população conforme as gerações.

4.4.3 Testes em cenários gerados pelo software desenvolvido

Foram feitos testes em outros cenários de tamanhos variados para verificar o comportamento do algoritmo em bases de tamanhos menores e maiores, no que se diz respeito ao direcionamento da busca e sua convergência. Como as demais bases testadas não foram retiradas de trabalhos publicados na literatura e sim criadas para este projeto, então houve a necessidade de determinar os parâmetros dos algoritmos para todos os cenários testados. Os parâmetros utilizados nos trabalhos Morandin et al. (2008a) e Morandin et al. (2008b) que usam como o algoritmo de busca o AG, receberam para testes os mesmo parâmetros que o algoritmo proposto (AG com operador de transgenia) e os parâmetros do trabalho de Kato, Morandin e Fonseca (2010) foram determinados por testes empíricos considerando uma boa parametrização para cada cenário testado.

No segundo cenário foram executados testes utilizando a abordagem proposta neste trabalho e nos trabalhos Morandin et al. (2008a) e Morandin et al. (2008b), foram utilizados os seguintes valores para os parâmetros: tamanho da população = 10, taxa de cruzamento = 0,8 (80%), taxa de mutação = 0,05 (5%), taxa de transgenia = 0,4 (40%), produtos fixos para transgenia (genes mais significativos) P_3 e a convergência da população como critério de parada. Os produtos fixos para transgenia foram selecionados através do método de encontrar os genes mais significativos, e os demais valores foram estabelecidos através da análise de resultados de vários testes. E para a abordagem de Kato, Morandin e Fonseca (2010), foram utilizados os seguintes valores para os parâmetros: influência do valor de feromônio = 1, influência do valor heurístico = 2, taxa de evaporação = 0,02, número de formigas = 50 e número de iterações = 75.

Para determinar tais parâmetros foram feitos testes empíricos e analisado quais parâmetros adequavam-se melhor ao AG e ACO para determinada base.

Para os demais cenários, nos trabalhos que utilizam AG foram alterados apenas o parâmetro referente ao tamanho da população inicial em cada cenário para 50, no algoritmo proposto foi necessário para cada cenário encontrar quais genes eram os mais significativos para usar na etapa de transgenia e para o de ACO foram alterados o parâmetro número de formigas e número de iterações.

Tabela 4.17 Resultados obtidos na execução do cenário 3x8.

ID do teste	Proposta		MORANDIN (2008a)		MORANDIN (2008b)		KATO (2010)	
	Makespan	Tempo (seg.)	Makespan	Tempo (seg.)	Makespan	Tempo (seg.)	Makespan	Tempo (seg.)
1	3148	0,1250	3148	0,1875	3148	0,2031	3148	0,0625
2	3148	0,1250	3168	0,2031	3148	0,3125	3194	0,0469
3	3148	0,0938	3148	0,1719	3231	0,2656	3148	0,0781
4	3148	0,0938	3148	0,1406	3194	0,3281	3148	0,0625
5	3148	0,1250	3200	0,1406	3148	0,2656	3148	0,0625
6	3148	0,1094	3200	0,1406	3148	0,2969	3194	0,0313
7	3148	0,1250	3148	0,2188	3148	0,2813	3148	0,0313
8	3148	0,0938	3168	0,1875	3194	0,1719	3200	0,0313
9	3148	0,0938	3148	0,1250	3194	0,2969	3148	0,0625
10	3148	0,0938	3148	0,1719	3148	0,2344	3148	0,0469
11	3148	0,1250	3148	0,1406	3148	0,2656	3148	0,0313
12	3148	0,1094	3148	0,1406	3148	0,2344	3168	0,0625
13	3148	0,1250	3148	0,2188	3148	0,2188	3200	0,0313
14	3148	0,1719	3194	0,1406	3148	0,2656	3148	0,0625
15	3148	0,0625	3194	0,1563	3194	0,2500	3148	0,0313
16	3168	0,1563	3148	0,1719	3148	0,2031	3148	0,0625
17	3148	0,0938	3168	0,2031	3194	0,2969	3148	0,0313
18	3148	0,1250	3148	0,2031	3148	0,3438	3194	0,0625
19	3168	0,1094	3148	0,2500	3148	0,2500	3168	0,0625
20	3148	0,1094	3148	0,2656	3148	0,3438	3168	0,0313
21	3194	0,1875	3148	0,1719	3168	0,1719	3148	0,0469

22	3200	0,1563	3148	0,1719	3148	0,2031	3148	0,0313
23	3148	0,1250	3148	0,2031	3194	0,3281	3148	0,0313
24	3148	0,0781	3194	0,1875	3148	0,2813	3148	0,0625
25	3148	0,1719	3168	0,2031	3148	0,2500	3148	0,0313
26	3148	0,1250	3148	0,1563	3200	0,2813	3148	0,0313
27	3148	0,1094	3200	0,2188	3148	0,2656	3148	0,0625
28	3148	0,1250	3148	0,1250	3148	0,2344	3148	0,0313
29	3148	0,0938	3148	0,1875	3148	0,3438	3148	0,0469
30	3194	0,1406	3322	0,2188	3148	0,2813	3148	0,0313
31	3148	0,1094	3148	0,2031	3148	0,3281	3148	0,0625
32	3168	0,1094	3148	0,1719	3194	0,2656	3168	0,0625
33	3148	0,0781	3148	0,2500	3148	0,2813	3148	0,0313
34	3148	0,0938	3148	0,1719	3148	0,2656	3168	0,0469
35	3148	0,1094	3148	0,1406	3148	0,4531	3148	0,0313
Média	3153,8286	0,1170	3163,6571	0,1817	3161,6286	0,2750	3157,771	0,0464

Como pode ser observado, a média obtida pela abordagem proposta foi de 3153,8286 com um desvio padrão de 14,2941. O valor mínimo encontrado foi de 3148 e o máximo 3200.

A média obtida por Morandin et al. (2008a) foi de 3163,6571, com um desvio padrão de 33,3237 devido à característica menos direcionada do método de busca utilizado, e os valores variam entre o mínimo de 3148 e o máximo de 3322.

A média obtida por Morandin et al. (2008b) foi de 3161,6286, com um desvio padrão de 23,2025, e os valores variam entre o mínimo de 3148 e o máximo de 3231.

A média obtida por Kato, Morandin e Fonseca (2010) foi de 3157,7714, com um desvio padrão de 17,5014, e os valores variam entre o mínimo de 3148 e o máximo de 3200.

A proposta apresentou melhores resultados que os trabalhos propostos por Morandin et al. (2008a) e Morandin et al. (2008b), comparado ao ACO de Kato, Morandin e Fonseca (2010) o valor mínimo e máximo encontrados foram iguais, no desvio padrão e na média o algoritmo teve resultados melhores, porém no tempo médio de execução do algoritmo ACO foi melhor (Tabela 4.18).

Tabela 4.18 Resumo dos resultados obtidos na execução do cenário 3x8.

Algoritmo utilizado	Média	Desvio Padrão	Menor Valor	Maior Valor	Tempo médio de execução
Proposta	3153,8286	14,2941	3148	3200	0,1170
MORANDIN (2008a)	3163,6571	33,3237	3148	3322	0,1817
MORANDIN (2008b)	3161,6286	23,2025	3148	3231	0,2750
KATO (2010)	3157,7714	17,5014	3148	3200	0,0464

Tabela 4.19 Resultados obtidos na execução do cenário 20x8.

ID do teste	Proposta		MORANDIN (2008b)		MORANDIN (2008a)		KATO (2010)	
	Makespan	Tempo (seg.)	Makespan	Tempo (seg.)	Makespan	Tempo (seg.)	Makespan	Tempo (seg.)
1	7127	2,3594	7206	2,4219	6843	3,3125	7600	11,8281
2	6732	2,2344	6861	1,9063	7162	4,0469	7700	11,2656
3	7135	2,9219	7170	2,4375	7024	4,5781	7554	12,8125
4	7213	1,9375	7370	1,9688	7285	3,1875	7734	10,8594
5	7109	2,5625	6852	2,3438	7198	3,8594	7661	10,4531
6	7068	1,8125	7221	1,4219	7304	4,8750	7545	12,0313
7	7261	2,4375	7175	1,7500	7176	3,1719	7680	13,2969
8	6881	2,2344	7186	3,3594	7191	3,7969	7590	12,0313
9	7234	0,9844	7155	2,5469	7225	3,9063	7606	9,4375
10	7229	3,0781	7218	2,7344	7178	4,4375	7897	13,6875
11	7135	2,7813	7123	3,9219	7108	4,4219	7663	10,3906
12	7125	3,4844	7048	2,4219	6793	4,2188	7485	11,9375
13	7131	3,7031	7163	3,2969	7094	3,7813	7635	12,1875
14	6841	3,7656	7096	2,6563	7538	4,8906	7811	13,4688
15	7064	2,5313	7174	3,8281	7241	4,2031	7577	10,5938
16	7193	2,5313	7187	2,9531	6729	5,6406	7550	12,8750
17	7189	2,0156	7204	2,4531	6778	5,1250	7235	11,6875
18	7053	3,9844	7182	2,5313	7196	3,3594	7559	13,3125
19	7081	2,9063	7062	2,9219	7471	5,2031	7628	13,2344
20	6698	3,0781	7143	3,0313	7006	4,5781	7853	12,5781
21	7152	2,2656	7157	4,0313	7404	3,6406	7764	12,9688
22	7172	2,3906	7163	2,6250	7262	3,7813	7153	10,2031
23	7211	1,7188	7208	3,2813	7144	5,1563	7600	10,9219
24	7085	2,2656	7257	2,9688	7183	5,6406	7520	12,3281
25	7124	2,0000	7116	2,2188	7072	4,2188	7603	11,0000
26	7144	2,5313	7135	2,8438	7200	3,4688	7628	12,3750
27	7072	3,9531	7163	1,9063	7193	5,6563	7730	10,7656
28	7232	2,2656	7201	1,8750	7145	4,5781	7579	11,7188
29	7120	3,7031	7245	2,2188	7225	5,6406	7614	11,0313
30	7301	2,1563	7026	3,1406	6915	4,8906	7510	11,0313
31	7115	2,6250	7216	2,5000	7154	5,0313	7548	10,6406
32	7099	2,8125	7311	2,4531	7173	4,4063	7442	13,1875
33	7179	2,1094	7239	4,5000	7190	5,6563	7527	11,0156
34	7276	2,5938	7097	2,3438	7449	4,8906	7614	10,3594
35	7229	2,7969	7181	2,4219	7207	3,9531	7673	13,0313
Média	7114,5714	2,6156	7157,4571	2,6928	7155,8857	4,4348	7601,9429	11,7870

Como pode ser observado, a média obtida pela abordagem proposta foi de 7114,5714 com um desvio padrão de 137,2279. O valor mínimo encontrado foi de 6698 e o máximo 7301.

A média obtida por Morandin et al. (2008a) foi de 7157,4571, com um desvio padrão de 101,0784 devido à característica menos direcionada do método de busca utilizado, e os valores variam entre o mínimo de 6852 e o máximo de 7311.

A média obtida por Morandin et al. (2008b) foi de 7155,8857, com um desvio padrão de 183,0339, e os valores variam entre o mínimo de 6729 e o máximo de 7538.

A média obtida por Kato, Morandin e Fonseca (2010) foi de 7601,9429, com um desvio padrão de 142,9887, e os valores variam entre o mínimo de 7153 e o máximo de 7897.

A proposta apresentou melhores resultados que os demais trabalhos propostos por Morandin et al. (2008a), Morandin et al. (2008b) e Kato, Morandin e Fonseca (2010) tanto no valor mínimo e máximo, quanto no desvio padrão e no tempo médio de execução do algoritmo (Tabela 20).

Tabela 4.20 Resumo dos resultados obtidos na execução do cenário 20x8.

Algoritmo utilizado	Média	Desvio Padrão	Menor Valor	Maior Valor	Tempo médio de execução
Proposta	7114,5714	137,2279	6698	7301	2,6156
MORANDIN (2008a)	7157,4571	101,0784	6852	7311	2,6928
MORANDIN (2008b)	7155,8857	183,0339	6729	7538	4,4348
KATO (2010)	7601,9429	142,9887	7153	7897	11,7870

Tabela 4.21 Resultados obtidos na execução do cenário 100x40.

ID do teste	Proposta		MORANDIN (2008a)		MORANDIN (2008b)		KATO (2010)	
	Makespan	Tempo (seg.)	Makespan	Tempo (seg.)	Makespan	Tempo (seg.)	Makespan	Tempo (seg.)
1	8689	11,8125	8824	9,4375	8540	25,8594	8637	60,2188
2	8579	9,9375	8586	18,0000	8710	24,0469	8604	57,7031
3	8548	11,1875	8315	11,7813	8938	21,9531	8644	60,7813
4	8569	14,1875	8616	12,1250	8539	28,8906	8715	47,8750
5	8609	9,2656	8659	6,7656	8607	21,9531	8716	53,0156
6	8205	11,7969	8324	10,6875	8543	21,1719	8558	48,7500
7	8571	15,9063	8652	8,9063	8563	27,6719	8581	50,0156
8	8573	11,1719	8814	16,4375	8561	15,3594	8708	52,7813
9	8270	16,0469	8890	10,2969	8681	18,2656	8979	40,0313
10	8554	14,2656	8253	16,0625	8514	34,1094	8778	81,7969
11	8589	16,0156	8384	11,2344	8580	23,9219	8533	60,6563
12	8590	18,4063	8689	11,6094	8549	27,6875	8566	43,2969
13	8588	9,8594	8889	11,7031	8682	13,1719	8537	45,3438
14	8602	11,1875	8587	10,2813	8609	20,4375	8568	54,5469
15	8618	12,9688	8125	21,7813	8255	21,1406	8681	56,1406
16	8196	17,5000	8677	19,7656	8602	21,9219	8648	61,0469
17	8607	10,3750	8658	13,8906	8461	20,3594	8596	45,1406
18	8622	10,3750	8450	20,0781	8571	21,8281	8558	40,1406
19	8118	14,9531	8526	20,1875	8633	23,9531	8686	49,7344
20	8662	20,2188	8494	11,4219	8446	24,5938	8512	49,9688
21	8613	13,7969	8655	12,4375	8614	18,0469	8608	51,4063
22	8450	7,8438	8555	12,9219	8556	25,5156	8782	53,0469
23	8632	13,1563	8634	16,2031	8539	19,1250	8535	41,6094
24	8570	15,0625	8639	9,6719	8543	23,0469	8500	53,0469
25	8827	8,4531	8531	15,1875	8601	28,5938	8464	40,2500
26	8538	12,0313	8716	12,7969	8590	19,6719	8468	38,5000
27	8586	14,9688	8550	10,2344	8735	21,1094	8460	40,0625
28	8503	12,0781	8634	14,2656	8709	21,0938	8533	52,9844
29	8378	9,6563	8686	12,0313	8198	19,6719	8847	49,8125
30	8560	10,8906	8533	16,1875	8653	29,5156	8937	35,1875
31	8551	13,2969	8522	9,1563	8239	26,7656	8743	33,9375
32	8473	10,9219	8481	16,4375	8682	22,5000	8472	46,7500

33	8564	11,5313	8557	14,5156	8639	22,3594	8620	48,5000
34	8619	10,2656	8690	15,0313	8589	20,3906	8562	67,5625
35	8621	15,0000	8244	11,3594	8591	26,8750	8576	83,7656
Média	8538,4000	12,7540	8572,5429	13,4545	8573,2000	22,9321	8626,0571	47,6276

Como pode ser observado, a média obtida pela abordagem proposta foi de 8538,4000 com um desvio padrão de 144,4668. O valor mínimo encontrado foi de 8118 e o máximo 8827.

A média obtida por Morandin et al. (2008a) foi de 8572,5429, com um desvio padrão de 175,6454 devido à característica menos direcionada do método de busca utilizado, e os valores variam entre o mínimo de 8125 e o máximo de 8890.

A média obtida por Morandin et al. (2008b) foi de 8573,2000, com um desvio padrão de 137,7299, e os valores variam entre o mínimo de 8198 e o máximo de 8938.

A média obtida por Kato, Morandin e Fonseca (2010) foi de 8626,0571, com um desvio padrão de 127,5364, e os valores variam entre o mínimo de 8460 e o máximo de 8979.

A proposta apresentou melhores resultados que os demais trabalhos propostos por Morandin et al. (2008a), Morandin et al. (2008b) e Kato, Morandin e Fonseca (2010) tanto no valor mínimo e máximo, quanto no desvio padrão e no tempo médio de execução do algoritmo (Tabela 4.22).

Tabela 4.22 Resumo dos resultados obtidos na execução do cenário 100x40.

Algoritmo utilizado	Média	Desvio Padrão	Menor Valor	Maior Valor	Tempo médio de execução
Proposta	8538,4000	144,4668	8118	8827	12,7540
MORANDIN (2008a)	8572,5429	175,454	8125	8890	13,4545
MORANDIN (2008b)	8573,2000	137,7299	8198	8938	22,9321
KATO (2010)	8626,0571	127,5364	8460	8979	47,6276

Tabela 4.23 Resultados obtidos na execução do cenário 150x40.

ID do teste	Proposta		MORANDIN (2008a)		MORANDIN (2008b)		KATO (2010)	
	Makespan	Tempo (seg.)	Makespan	Tempo (seg.)	Makespan	Tempo (seg.)	Makespan	Tempo (seg.)
1	11678	9,8281	11993	8,5000	11686	14,0938	12218	44,5156
2	11865	6,6875	11558	8,1563	11911	10,7031	12072	59,4688
3	11804	8,5469	11721	7,9219	11701	17,9844	12040	60,9063
4	11839	7,5313	11869	9,1094	11971	11,9688	11647	60,5781
5	11863	8,7500	11638	12,3438	11960	14,5625	12259	33,7031
6	11798	7,1875	11604	9,6719	11979	12,6250	11854	53,9219
7	11664	14,5000	11974	12,6250	11802	14,0625	12049	48,3125
8	11873	11,5938	11951	7,2500	11826	11,9375	12073	60,5156
9	12086	5,8906	11911	7,2656	11572	14,2500	11845	40,0000
10	11706	5,7031	11733	9,3906	11832	15,4844	12072	40,6406
11	11843	9,2344	11782	7,9063	11609	14,5313	12175	46,0469
12	11903	5,8594	11955	6,3906	11789	19,0938	12011	44,9688

13	11789	5,8750	11907	6,9844	11934	18,3594	11882	37,9688
14	11602	7,9531	11997	9,0156	11900	13,6563	11772	45,1250
15	11677	9,1406	11926	9,9375	11875	13,5781	11526	45,1406
16	11921	8,3438	11353	9,9531	11757	20,4063	12217	38,6719
17	11986	5,0313	11704	8,1719	11986	21,2969	12206	36,9531
18	11771	5,9219	12062	8,4688	12103	13,1094	11838	50,8438
19	12004	6,8750	12215	10,2500	11793	15,6406	11839	33,2969
20	12100	7,1250	11972	6,3906	11555	14,7344	12158	46,5156
21	11798	7,1250	11838	10,1250	11921	13,1250	12173	48,5469
22	11834	10,3594	11894	14,4219	11636	18,8750	12140	59,6250
23	11917	7,5156	11769	10,2813	12130	7,3438	11832	50,9219
24	11655	6,3125	12022	7,3281	11731	11,2969	11637	40,5313
25	12148	7,5625	12080	7,3438	11963	13,6719	11957	56,9375
26	11299	6,2656	11547	10,2031	11991	12,2656	11742	53,5313
27	11850	6,8906	11762	6,0625	11916	21,3438	12532	44,1406
28	11641	9,9844	11947	9,3750	11453	19,5000	12047	62,4531
29	12089	9,5938	11429	7,6094	11962	13,5781	11732	50,9375
30	11651	6,7031	11971	6,6563	11836	10,2188	11882	36,5625
31	11287	6,6875	12114	8,7813	12050	11,6563	11833	36,5000
32	11842	6,7188	11768	5,4844	11854	15,0938	12087	40,7031
33	12085	8,3438	11784	10,5313	11973	17,6406	12242	54,3125
34	11721	9,3906	12099	5,4688	11711	11,7344	11974	43,3281
35	11653	5,8594	12094	8,1719	11752	13,5625	11719	58,3750
Média	11806,9143	7,7973	11855,514	8,6732	11840,571	14,6580	11979,4857	47,5861

Como pode ser observado, a média obtida pela abordagem proposta foi de 11806,9143 com um desvio padrão de 195,6512. O valor mínimo encontrado foi de 11287 e o máximo 12148.

A média obtida por Morandin et al. (2008a) foi de 11855,5143, com um desvio padrão de 200,8834 devido à característica menos direcionada do método de busca utilizado, e os valores variam entre o mínimo de 11353 e o máximo de 12215.

A média obtida por Morandin et al. (2008b) foi de 11840,5714, com um desvio padrão de 159,2086, e os valores variam entre o mínimo de 11453 e o máximo de 12130.

A média obtida por Kato, Morandin e Fonseca (2010) foi de 11979,4857, com um desvio padrão de 216,9361, e os valores variam entre o mínimo de 11526 e o máximo de 12532.

A proposta apresentou melhores resultados que os demais trabalhos propostos por Morandin et al. (2008a), Morandin et al. (2008b) e Kato, Morandin e Fonseca (2010) tanto no valor mínimo e máximo, quanto no desvio padrão e no tempo médio de execução do algoritmo (Tabela 4.24).

Tabela 4.24 Resumo dos resultados obtidos na execução do cenário 150x40.

Algoritmo utilizado	Média	Desvio Padrão	Menor Valor	Maior Valor	Tempo médio de execução
Proposta	11806,9143	195,6512	11287	12148	7,7973
MORANDIN (2008a)	11855,5143	200,8834	11353	12215	8,6732
MORANDIN (2008b)	11840,5714	159,2086	11453	12130	14,6580
KATO (2010)	11979,4857	216,9361	11526	12532	47,5861

Tabela 4.25 Resultados obtidos na execução do cenário 180x80.

ID do teste	Proposta		MORANDIN (2008a)		MORANDIN (2008b)		KATO (2010)	
	Makespan	Tempo (seg.)	Makespan	Tempo (seg.)	Makespan	Tempo (seg.)	Makespan	Tempo (seg.)
1	8790	12,0938	8594	10,0781	8905	21,6875	9052	65,5000
2	8881	10,0938	8754	12,3125	8406	24,7813	8817	64,1406
3	8659	12,8750	8617	19,4375	8559	25,9063	8935	65,0938
4	8562	11,2500	8644	12,2188	8611	25,6094	8760	66,1563
5	8621	11,9531	8902	13,4063	8441	24,7031	8692	66,1563
6	8836	11,2813	8664	17,4219	8919	20,7813	8572	65,9688
7	8630	14,0313	8729	14,8438	8563	29,0156	8855	66,9219
8	8662	12,7969	8613	12,1875	8676	40,2344	8914	66,7188
9	8687	12,6719	8777	9,0469	8511	25,7031	8791	65,9219
10	8636	13,3281	8649	9,2031	8720	24,0469	8727	64,4531
11	8669	13,3438	8538	13,7344	8975	20,7969	8643	65,7031
12	8742	11,9375	8562	13,2344	8792	23,4375	8650	65,3750
13	8516	14,5156	8857	15,3438	8963	24,5781	8628	65,2656
14	8579	14,7344	8429	19,1406	8865	21,3750	8958	64,3281
15	8555	14,8125	8747	20,5625	8769	32,8438	8926	65,3594
16	8645	10,5625	8803	16,9844	8661	18,9063	8616	65,8438
17	8598	12,1875	8742	16,3594	8927	22,3125	8676	65,9219
18	8574	11,2969	8790	27,6250	8631	31,2656	8921	66,3125
19	8594	10,6094	8403	23,5938	8533	29,9375	8616	66,1406
20	8818	16,9375	8631	20,4844	8504	26,6406	8576	66,8125
21	8556	17,4688	8635	31,7500	8985	21,4219	8613	64,2656
22	8602	12,9063	8590	11,9375	8776	36,3438	9099	64,9063
23	8658	14,2656	8631	19,1719	8867	30,8906	8731	65,1094
24	8925	11,9844	8893	14,8125	8769	19,1094	8666	58,6563
25	8700	14,0313	8694	16,3594	8982	26,5313	8924	52,5000
26	8620	13,5469	8902	16,8906	8653	23,8750	8954	52,4688
27	8725	13,3594	8599	12,5000	8681	20,8125	8568	52,9219
28	8330	11,2500	8806	14,7500	8632	20,5313	8491	52,4375
29	8595	19,5938	8562	19,4375	8814	22,3906	9001	52,0156
30	8642	12,6719	8528	14,3281	8506	25,7031	8934	52,3750
31	8620	21,8125	8683	18,7969	8601	21,3750	8807	52,3750
32	8626	10,5781	8551	20,6875	8654	22,3906	8925	52,3750
33	8713	14,7500	8950	16,0938	8976	31,5938	8647	52,4375
34	8704	15,6719	8674	15,0938	8643	32,6094	9028	52,4531
35	8644	14,0156	8543	17,6406	8928	20,5625	8644	52,4219
Média	8654,6857	13,4642	8676,7429	16,4991	8725,6571	25,448	8781,6286	61,2236

Como pode ser observado, a média obtida pela abordagem proposta foi de 8654,6857 com um desvio padrão de 109,5466. O valor mínimo encontrado foi de 8330 e o máximo 8925.

A média obtida por Morandin et al. (2008a) foi de 8676,7429, com um desvio padrão de 133,1596 devido à característica menos direcionada do método de busca utilizado, e os valores variam entre o mínimo de 8403 e o máximo de 8950.

A média obtida por Morandin et al. (2008b) foi de 8725,6571, com um desvio padrão de 137,7299, e os valores variam entre o mínimo de 8406 e o máximo de 8985.

A média obtida por Kato, Morandin e Fonseca (2010) foi de 8781,6286, com um desvio padrão de 165,5300, e os valores variam entre o mínimo de 8491 e o máximo de 9099.

A proposta apresentou melhores resultados que os demais trabalhos propostos por Morandin et al. (2008a), Morandin et al. (2008b) e Kato, Morandin e Fonseca (2010) tanto no valor mínimo e máximo, quanto no desvio padrão e no tempo médio de execução do algoritmo (Tabela 4.26).

Tabela 4.26 Resumo dos resultados obtidos na execução do cenário 180x80.

Algoritmo utilizado	Média	Desvio Padrão	Menor Valor	Maior Valor	Tempo médio de execução
Proposta	8654,6857	109,5466	8330	8925	13,4642
MORANDIN (2008a)	8676,7429	133,1596	8403	8950	16,4991
MORANDIN (2008b)	8725,6571	137,7299	8406	8985	25,4487
KATO (2010)	8781,6286	165,5300	8491	9099	61,2236

Conforme apresentado nos resultados apresentados nos demais cenários, o algoritmo proposto apresenta bons resultados tanto em bases menores, quanto maiores. Em todas as bases ele conseguiu trazer melhores resultados de *makespan* em menor tempo de processamento. Em posse desses resultados é possível concluir que o algoritmo proposto se adapta bem para diversos cenários.

4.5 Considerações finais

Neste capítulo foram detalhados os procedimentos da abordagem proposta e em seguida definidos os cenários para aplicação e avaliação da abordagem. Os resultados obtidos foram comparados com Morandin *et al.* (2008a), Morandin *et al.* (2008b) e Kato, Morandin e Fonseca (2010), sendo considerado o valor de *makespan* obtido e o tempo de resposta com critérios de avaliação.

Capítulo 5

CONCLUSÃO

O objetivo do trabalho foi desenvolver uma abordagem que utiliza uma meta-heurística para redução de *makespan* em uma programação reativa da produção. A proposta foi testada e os resultados obtidos foram comparados a outras abordagens propostas em trabalhos relacionados (Morandin et al. (2008a), Morandin et al. (2008b) e Kato, Morandin e Fonseca (2010)), utilizando como critérios de avaliação a minimização do valor de *makespan* e o tempo para obtenção da resposta.

Como contribuição principal deste trabalho destaca-se a adaptação de uma versão alternativa do operador de transgenia proposto por Amaral e Hruschka (2011) e Amaral e Hruschka (2013) ao problema da programação reativa da produção. Também foram desenvolvidos dois métodos que determinam quais são os genes mais significativos para cada cenário. Além disso, para realizar certas avaliações foi elaborado um *software* que a partir de determinados dados de entrada gera possíveis cenários de manufatura.

Foram testados em cenários de tamanhos diversos, desde uma configuração de tamanho considerado pequeno de 3×9 , que representa um cenário de 3 produtos e 9 máquinas, a um de 180×80 , que representa um cenário de 180 produtos e 80 máquinas. Algumas variáveis de grandezas estatísticas foram utilizadas para uma comparação breve dos resultados. Sendo que a média foi utilizada para verificar a tendência do conjunto de resultados obtidos e o desvio padrão foi utilizado para verificar a dispersão dos valores individuais em torno da média.

Os resultados obtidos pelo algoritmo proposto foram comparados com três trabalhos presentes no estado da arte da literatura especializada, sendo que dois utilizam AG como algoritmo de busca para o problema de programação reativa da produção (Morandin et al. (2008a) e Morandin et al., (2008b)) e um utiliza ACO (Kato, Morandin e Fonseca (2010)). Os dois primeiros foram escolhidos por usarem AG e, com isso, apresentar nos resultados a vantagem que se tem em utilizar o operador de transgenia e a desvantagem em não usar. Já o

terceiro trabalho foi escolhido por utilizar uma meta-heurística diferente do Algoritmo Genético e por ser muito utilizada em trabalhos similares.

O algoritmo com operador transgênico aplicado no problema da programação da produção apresenta uma convergência mais rápida do que um AG sem operador de transgenia, diminuindo o tempo de resposta dos resultados.

Por meio da comparação dos valores de *makespan* obtidos para o primeiro cenário testado, houve uma tendência de melhoria do Algoritmo Proposto em 100% dos casos em comparação com os resultados obtidos com o Algoritmo Genético e com o Algoritmo Genético adaptativo. Em comparação com o ACO, houve uma tendência de melhoria do Algoritmo Proposto em 88% dos casos. A média do tempo de execução do algoritmo proposto foi 2,9383 segundos, enquanto a AG foi 3,0861, o AG Adaptativo foi 4,7737 e o ACO foi 12,4330, ou seja, houve uma redução no tempo de processamento do algoritmo proposto em 5% com relação ao AG, em 62% com relação ao AG Adaptativo e em 323% com relação ao ACO. Além disso, por meio da comparação dos valores de *makespan* obtidos para o problema testado, é possível concluir com 95% de confiança, que o método proposto terá resultados melhores do que os resultados obtidos por AG, AG Adaptativo e ACO.

Ao analisar os resultados obtidos é constatado que o algoritmo proposto alcança o objetivo estabelecido no trabalho e apresenta-se como um algoritmo viável e aplicável em outros problemas combinatórios. Para os cenários analisados, o algoritmo proposto foi capaz de reduzir o valor de *makespan* e o tempo de processamento.

5.1 Trabalhos futuros

Para trabalhos futuros propõe-se um estudo mais detalhado sobre a influência da quantidade de genes que devem ser transmitidos na etapa de transgenia, uma vez que foi verificado nos experimentos que tal variação tem grande impacto nos resultados. Também deve ser investigada mais profundamente a porcentagem ideal de transgenia que deve ser aplicada para melhores resultados sem renunciar à diversidade da população.

Conforme apresentado na literatura, o AG adaptativo aplicado ao problema da programação reativa da produção traz resultados mais relevantes que um AG canônico. Dessa

forma, seria interessante avaliar o efeito da adição de regras adaptativas no AG com operador de transgenia a fim de obter melhoras nos resultados.

O algoritmo genético com operador de transgenia se mostrou promissor na resolução de problemas de programação reativa da produção. Sendo assim, é conveniente que em trabalhos futuros o algoritmo proposto seja aplicado em problemas similares ao problema de programação da produção, uma vez que o AG com operador de transgenia obteve resultados mais significativos quando comparado a outras meta-heurísticas. Dessa forma, é possível que funcione de forma equivalente quando aplicado em outros problemas combinatórios.

Com base nas indicações da literatura, diversos trabalhos atuais apontam para o uso de algoritmos multiobjetivo, neste trabalho é abordada apenas uma variável, o *makespan*, outras variáveis que também tem grau significativo de importância para o problema de programação da produção, por exemplo, *tardiness* e *earliness* poderiam ser abordadas.

Outra sugestão para trabalhos futuros é verificar a possibilidade de criar novos métodos de determinar os genes mais significativos utilizando outros algoritmos de análise matemática de dados, por exemplo, a técnica conhecida como NMF (*Non-negative Matrix Factorization*), que é uma técnica não determinística de análise de dados que realiza manipulações com grandes matrizes cujos elementos são não negativos. Tal técnica pode ser utilizada para cálculo de influência (LEE et al., 2009) gerando resultados muito bons.

REFERÊNCIAS BIBLIOGRÁFICAS

ABELLO, M. B.; MICHALEWICZ, Z.; BUI, L. T. **A reactive-proactive approach for solving dynamic scheduling with time-varying number of Tasks.** 2012 IEEE Congress on Evolutionary Computation (CEC), 2012. 1-10.

AGUIRRE, L. A. **Enciclopédia de Automática Controle e Automação.** Vol. 1. São Paulo: Editora Blucker, 2007.

AISSANI, N.; BELDJILALI, B.; TRENTESAUX, D. Dynamic scheduling of maintenance tasks in the petroleum industry: A reinforcement approach. **Engineering Applications of Artificial Intelligence**, v. 22, p. 1089-1103, 2009.

AL-HINAI, N.; ELMEKKAWY, T. Robust and stable flexible job shop scheduling with random machine breakdowns using a hybrid genetic algorithm. **International Journal of Production Economics**, v. 132, n. 2, p. 279–291, 2011.

AL-SALEM, A. Scheduling to minimize makespan on unrelated parallel machines with sequence dependent setup times. **Engineering Journal of the University of Qatar**, v. 17, p. 177–187, 2004.

AMARAL, L. R.; HRUSCHKA, E. R. **Transgenic, an operator for evolutionary algorithms.** 2011 IEEE Congress on Evolutionary Computation (CEC), [S.l.]: [s.n.]. 2011. p. 1308-1314.

AMARAL, L. R.; HRUSCHKA, E. R. Transgenic: An evolutionary algorithm operator. **Neurocomputing**, v. 127, p. 104-113, 2014.

ARNAOUT, J.-P.; RABADI, G.; MUSA, R. A two-stage Ant Colony Optimization algorithm to minimize the makespan on unrelated parallel machines with sequence-dependent setup times. **Journal of Intelligent Manufacturing**, v. 21, n. 6, p. 693-701, 2010. ISSN ISSN: 0956-5515 DOI: 10.1007/s10845-009-0246-1. Disponível em: <<http://dx.doi.org/10.1007/s10845-009-0246-1>>.

BAI, D.; TANG, L. Open shop scheduling problem to minimize makespan with release dates. **Applied Mathematical Modelling**, v. 37, n. 4, p. 2008-2015, 2013. ISSN ISSN: 0307-904X

DOI: 10.1016/j.apm.2012.04.037. Disponível em:
<<http://www.sciencedirect.com/science/article/pii/S0307904X12002685>>.

DAVENDRA, D., ZELINKA, I., BIALIC-DAVENDRA, M., SENKERIK, R.; JASEK, R. Discrete Self-Organising Migrating Algorithm for flow-shop scheduling with no-wait makespan. **Mathematical and Computer Modelling**, v. 57, n. 12, p. 100-110, 2013. ISSN 0895-7177 DOI: 10.1016/j.mcm.2011.05.029. Disponível em:
<<http://www.sciencedirect.com/science/article/pii/S0895717711002998>>. **Mathematical and Computer Modelling in Power Control and Optimization**.

FINK, A.; VO, S. Solving the continuous flow-shop scheduling problem by metaheuristics. **European Journal of Operational Research**, v. 151, n. 2, p. 400-414, 2003. ISSN 0377-2217 DOI: 10.1016/S0377-2217(02)00834-2. Disponível em:
<<http://www.sciencedirect.com/science/article/pii/S0377221702008342>>. **Meta-heuristics in combinatorial optimization**.

GOMEZ-GASQUET, P.; ANDRES, C.; LARIO, F.-C. An agent-based genetic algorithm for hybrid flowshops with sequence dependent setup times to minimise makespan. **Expert Systems with Applications**, v. 39, n. 9, p. 8095-8107, 2012. ISSN 0957-4174 DOI: 10.1016/j.eswa.2012.01.158. Disponível em:
<<http://www.sciencedirect.com/science/article/pii/S0957417412001789>>.

GROOVER, M. P. **Fundamentals of Modern Manufacturing: Materials, Processes, And Systems**. 3 ed. Editora John Wiley & Sons, 2006.

GUO, P.; WANG, X.; HAN, Y. **The enhanced genetic algorithms for the optimization design**. Biomedical Engineering and Informatics (BMEI), 2010 3rd International Conference on. [S.l.]: [s.n.], 2010. p. 2990-2994.

HASAN, S. M. K.; SARKER, R.; ESSAM, D. **Evolutionary scheduling with rescheduling option for sudden machine breakdowns**. Evolutionary Computation (CEC), 2010 IEEE Congress on. [S.l.]: [s.n.], 2010. p. 1-8.

HELAL, M.; RABADI, G.; AL-SALEM, A. A tabu search algorithm to minimize the makespan for the unrelated parallel machines scheduling problem with setup times. **International Journal of Operations Research**, v. 3, n. 3, p. 182-192, 2006.

HOLLAND, J. H. **Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence**. Michigan - USA: University of Michigan Press, 1975.

JOLLIFFE, Ian. **Principal component analysis**. John Wiley & Sons, Ltd, 2002.

JONG, K. A. DE. **An Analysis of the Behavior of a Class of Genetic Adaptive Systems**, 1975. University of Michigan.

KATO, E. R. R.; MORANDIN, O.; FONSECA, M. A. S. **Ant colony optimization algorithm for reactive production scheduling problem in the job shop system**. *Systems, Man and Cybernetics*, 2009. SMC 2009. IEEE International Conference on. [S.l.]: [s.n.]. 2009. p. 2199-2204.

KATO, E. R. R.; MORANDIN, O.; FONSECA, M. A. S. **A Max-Min Ant System modeling approach for production scheduling in a FMS**. *Systems Man and Cybernetics (SMC)*, 2010 IEEE International Conference on. [S.l.]: [s.n.]. 2010. p. 3977-3982.

KAZEMI, A. et al. An Improved Power Quality Monitor Placement Method Using MVR Model and Combine Cp and Rp Statistical Indices. **Journal of Electrical Review**, v. 88, p. 205-209, 2012.

KHALID, M. N. A.; YUSOF, U. K.; SABUDIN, M. **Solving flexible manufacturing system distributed scheduling problem subject to maintenance using harmony search algorithm**. 2012 4th Conference on Data Mining and Optimization (DMO). [S.l.]: [s.n.]. 2012. p. 73-79.

Lee, J. H., Park, S., Ahn, C. M., & Kim, D. (2009). Automatic generic document summarization based on non-negative matrix factorization. **Information Processing & Management**, 45(1), 20-34.

LINDEN, R. **Algoritmos Genéticos: Uma importante ferramenta da Inteligência Computacional**. 3ª ed. Rio de Janeiro - Brazil: Brasport, 2012.

MANCHON, U., HO, C., FUNK, S., & RASHEED, K. **GART: A genetic algorithm based real-time system scheduler**. *Evolutionary Computation (CEC)*, 2011 IEEE Congress on. [S.l.]: [s.n.]. 2011. p. 886-893.

MITCHELL, M. **An introduction to genetic algorithms**. Fifth ed. London - England: MIT Press, 1998.

MORANDIN, O., KATO, E. R. R., MAGGIO, E. G. R., SANCHES, D. S., DERIZ, A. C. **A Heuristic based on Petri Nets modeling for FMS Scheduling problem of makespan minimization.** Industrial Electronics Society, 2007. IECON 2007. 33rd Annual Conference of the IEEE. [S.l.]: [s.n.]. 2007. p. 2683-2688.

MORANDIN, O., KATO, E. R., DERIZ, A. C., SANCHES, D. S. **A search method using genetic algorithm for production reactive scheduling of manufacturing systems.** 2008 IEEE International Symposium on Industrial Electronics. ISIE 2008.. [S.l.: s.n.], 2008a. 1843–1848.

MORANDIN, O., SANCHES, D. S., 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, 2008b.

MORANDIN, O., KATO, E. R., SANCHES, D. S., MUNIZ, B. D. **Modeling Strategy by Adaptive Genetic Algorithm for Production Reactive Scheduling with Simultaneous use of Machines and AGVs.** Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics, San Antonio, Outubro 2009. 704 - 709.

NAWAZ, M.; ENSCORE, E. E.; HAM, I. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. **Omega**, v. 11, n. 1, p. 91-95, 1983. ISSN ISSN: 0305-0483 DOI: 10.1016/0305-0483(83)90088-9. Disponível em: <<http://www.sciencedirect.com/science/article/pii/0305048383900889>>.

NIE, L., GAO, L., LI, P., & LI, X. A GEP-based reactive scheduling policies constructing approach for dynamic flexible job shop scheduling problem with job release dates. **Journal of Intelligent Manufacturing**, p. 1-12, 2012. Disponível em: <<http://dx.doi.org/10.1007/s10845-012-0626-9>>.

OSMAN, I.; POTTS, C. Simulated annealing for permutation flow-shop scheduling. **Omega**, v. 17, n. 6, p. 551-557, 1989. ISSN ISSN: 0305-0483 DOI: 10.1016/0305-0483(89)90059-5. Disponível em: <<http://www.sciencedirect.com/science/article/pii/0305048389900595>>.

PAN, Q.-K.; TASGETIREN, M. F.; LIANG, Y.-C. A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem. **Computers & Operations Research**, v. 35, n. 9, p. 2807-2839, 2008. ISSN ISSN: 0305-0548 DOI:

10.1016/j.cor.2006.12.030. Disponível em:
<<http://www.sciencedirect.com/science/article/pii/S0305054806003170>>. Part Special Issue:
Bio-inspired Methods in Combinatorial Optimization.

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** , v. 78, n. 3, p. 311-322, 2002. ISSN ISSN: 0925-5273 DOI: 10.1016/S0925-5273(02)00104-4. Disponível em:
<<http://www.sciencedirect.com/science/article/pii/S0925527302001044>>.

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** , v. 152, n. 1, p. 215-225, 2004. ISSN ISSN: 0377-2217 DOI: 10.1016/S0377-2217(02)00645-8. Disponível em:
<<http://www.sciencedirect.com/science/article/pii/S0377221702006458>>.

RABADI, G.; MORAGA, R.; AL-SALEM, A. Heuristics for the Unrelated Parallel Machine Scheduling Problem with Setup Times. **Journal of Intelligent Manufacturing**, v. 17, n. 1, p. 85-97, 2006. ISSN ISSN: 0956-5515 DOI: 10.1007/s10845-005-5514-0. Disponível em:
<<http://dx.doi.org/10.1007/s10845-005-5514-0>>.

RAJENDRAN, C.; ZIEGLER, H. Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs. **European Journal of Operational Research** , v. 155, n. 2, p. 426-438, 2004. ISSN ISSN: 0377-2217 DOI: 10.1016/S0377-2217(02)00908-6. Disponível em:
<<http://www.sciencedirect.com/science/article/pii/S0377221702009086>>. Financial Risk in Open Economies.

REDDY, B. S. P.; RAO, C. S. P. A hybrid multi-objective GA for simultaneous scheduling of machines and AGVs in FMS. **The International Journal of Advanced Manufacturing Technology**, v. 31, n. 5-6, p. 602-613, 2006. ISSN ISSN: 0268-3768 DOI: 10.1007/s00170-005-0223-6. Disponível em: <<http://dx.doi.org/10.1007/s00170-005-0223-6>>.

REDDY, B.; RAO, C. Flexible manufacturing systems modelling and performance evaluation using AutoMod. **International Journal of Simulation Modelling**, v. 10, n. 2, p. 78-90, 2011.

- REEVES, C. R. A genetic algorithm for flowshop sequencing. **Computers & Operations Research** , v. 22, n. 1, p. 5-13, 1995. ISSN ISSN: 0305-0548 DOI: 10.1016/0305-0548(93)E0014-K. Disponível em: <<http://www.sciencedirect.com/science/article/pii/0305054893E0014K>>. Genetic Algorithms.
- REYES, A., YU, H., KELLEHER, G., LLOYD, S. Integrating Petri Nets and hybrid heuristic search for the scheduling of FMS. **Computers in Industry**, v. 47, n. 1, p. 123-138, 2002. ISSN ISSN: 0166-3615 DOI: 10.1016/S0166-3615(01)00124-5. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0166361501001245>>.
- ROSHANAELI, V., NADERI, B., JOLAI, F., KHALILI, M. A variable neighborhood search for job shop scheduling with set-up times to minimize makespan. **Future Generation Computer Systems** , v. 25, n. 6, p. 654-661, 2009. ISSN ISSN: 0167-739X DOI: 10.1016/j.future.2009.01.004. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0167739X09000028>>.
- ROTSCHTEIN, A. P.; RAKYTYANSKA, H. B. **Fuzzy evidence in identification, forecasting and diagnosis**. [S.l.]: Springer, v. 275, 2012.
- RUIZ, R.; MAROTO, C. A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. **European Journal of Operational Research** , v. 169, n. 3, p. 781-800, 2006. ISSN ISSN: 0377-2217 DOI: 10.1016/j.ejor.2004.06.038. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0377221705001451>>.
- SEMANCO, P.; MODRAK, V. **Hybrid GA-based metaheuristics for production planning and scheduling optimization in intelligent flow-shop manufacturing systems**. Intelligent Engineering Systems (INES), 2011 15th IEEE International Conference on. [S.l.]: [s.n.], 2011. p. 381-385.
- SILVA, E. L.; MENEZES, E. M. Metodologia da pesquisa e elaboração de dissertação. **Florianópolis, UFSC**, 2005.
- STÜTZLE, T. **Applying iterated local search to the permutation flow shop problem**. FG Intellektik, TU Darmstadt, Darmstadt, Germany, 1998.
- SUBBAIAH, K.; RAO, M. N.; RAO, K. N. Scheduling of AGVs and machines in FMS with makespan criteria using sheep flock heredity algorithm. **International Journal of Physical Sciences**, v. 4, n. 2, p. 139-148, 2009.

TAILLARD, E. Benchmarks for basic scheduling problems. **European Journal of Operational Research**, v. 64, n. 2, p. 278-285, 1993. ISSN: 0377-2217 DOI: 10.1016/0377-2217(93)90182-M. Disponível em: <<http://www.sciencedirect.com/science/article/pii/037722179390182M>>. Project Management and Scheduling.

TANG, L.; WANG, X. A predictive reactive scheduling method for color-coating production in steel industry. **The International Journal of Advanced Manufacturing Technology**, v. 35, n. 7-8, p. 633-645, 2008. ISSN: 0268-3768 DOI: 10.1007/s00170-006-0740-y. Disponível em: <<http://dx.doi.org/10.1007/s00170-006-0740-y>>.

TANIMIZU, Y., MIYAMAE, T., SAKAGUCHI, T., IWAMURA, K., & SUGIMURA, N. Multi-objective Reactive Scheduling Based on Genetic Algorithm. In: **Towards Synthesis of Micro-/Nano-systems**. [S.l.]: Springer London, 2007. p. 65-70. ISBN: 978-1-84628-558-5 DOI: 10.1007/1-84628-559-3_10. Disponível em: <http://dx.doi.org/10.1007/1-84628-559-3_10>.

TASGETIREN, M. F., LIANG, Y. C., SEVKLI, M., GENCYILMAZ, G. A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem. **European Journal of Operational Research**, v. 177, n. 3, p. 1930-1947, 2007. ISSN: 0377-2217 DOI: 10.1016/j.ejor.2005.12.024. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0377221705008453>>.

TOADER, Florentina Alina. Production scheduling by using ACO and PSO techniques. In: **2014 International Conference on Development and Application Systems (DAS)**. IEEE, 2014. p. 170-175.

TOGUYÉNI, A. K. A. BERRUET, P. CRAYE, E. Models and Algorithms for Failure Diagnosis and Recovery in FMSs. **International Journal of Flexible Manufacturing Systems**, v. 15 p. 57 – 85, 2003.

TUMA, Carlos CM; MORANDIN, Orides; CARIDÁ, Vinicius F. Minimizing the makespan for the problem of reactive production scheduling in a FMS with AGVs using a new structure of chromosome in a hybrid GA with TS. In: **2013 IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA)**. IEEE, 2013. p. 1-6.

UDHAYAKUMAR, P.; KUMANAN, S. Sequencing and scheduling of job and tool in a flexible manufacturing system using ant colony optimization algorithm. **The International Journal of Advanced Manufacturing Technology**, v. 50, n. 9-12, p. 1075-1084, 2010. ISSN ISSN: 0268-3768 DOI: 10.1007/s00170-010-2583-9. Disponível em: <<http://dx.doi.org/10.1007/s00170-010-2583-9>>.

VALLADA, E.; RUIZ, R.; MAROTO, C. **Synthetic and Real Benchmarks for Complex Flow-shops Problems**. Grupo de Investigación Operativa (GIO), Universitat Politècnica de València. València, Espanha. 2003.

WANG, X.-Y.; WANG, M.-Z.; WANG, J.-B. Flow shop scheduling to minimize makespan with decreasing time-dependent job processing times. **Computers & Industrial Engineering**, v. 60, n. 4, p. 840-844, 2011. ISSN ISSN: 0360-8352 DOI: 10.1016/j.cie.2011.02.003. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0360835211000581>>.

WEI, G. Z. LI, N., LI, H., SHI, N. **A New Replica Selection Strategy Based on Combination Algorithm**. 2010 International Conference on Internet Technology and Applications. [S.l.]: [s.n.]. 2010. p. 1-5.

WIDMER, M.; HERTZ, A. A new heuristic method for the flow shop sequencing problem. **European Journal of Operational Research**, v. 41, n. 2, p. 186-193, 1989. ISSN ISSN: 0377-2217 DOI: 10.1016/0377-2217(89)90383-4. Disponível em: <<http://www.sciencedirect.com/science/article/pii/0377221789903834>>.

WITKOWSKI, T.; ANTCZAK, P.; ANTCZAK, A. **Solving the Flexible Open-Job Shop Scheduling Problem with GRASP and Simulated Annealing**. Artificial Intelligence and Computational Intelligence (AICI), 2010 International Conference on. [S.l.]: [s.n.]. 2010. p. 437-442.

WU, L. H. et al. **A genetic algorithm for reactive scheduling based on real-time manufacturing information**. Responsive Manufacturing - Green Manufacturing (ICRM 2010), 5th International Conference on. [S.l.]: [s.n.]. 2010. p. 375-381.

YEUNG, W. H. R.; MOORE, P. R. Genetic algorithms and flexible process planning in the design of fault tolerant cell control for flexible assembly systems. **International Journal of Computer Integrated Manufacturing**, v. 13, n. 2, p. 157-168, 2000. ISSN DOI:

10.1080/095119200130009. Disponível em:
<<http://www.tandfonline.com/doi/abs/10.1080/095119200130009>>.

YING, K.-C.; LEE, Z.-J.; LIN, S.-W. Makespan minimization for scheduling unrelated parallel machines with setup times. **Journal of Intelligent Manufacturing**, v. 23, n. 5, p. 1795-1803, 2012. ISSN ISSN: 0956-5515 DOI: 10.1007/s10845-010-0483-3. Disponível em: <<http://dx.doi.org/10.1007/s10845-010-0483-3>>.

YU, H., REYES, A., CANG, S., LLOYD, S. Combined Petri net modelling and AI based heuristic hybrid search for flexible manufacturing systems part 1. Petri net modelling and heuristic search. **Computers & Industrial Engineering**, v. 44, n. 4, p. 527-543, 2003a. ISSN ISSN: 0360-8352 DOI: 10.1016/S0360-8352(02)00212-7. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0360835202002127>>.

YU, H., REYES, A., CANG, S., LLOYD, S. Combined Petri net modelling and AI-based heuristic hybrid search for flexible manufacturing systems part II. Heuristic hybrid search. **Computers & Industrial Engineering**, v. 44, n. 4, p. 545-566, 2003b. ISSN ISSN: 0360-8352 DOI: 10.1016/S0360-8352(02)00213-9. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0360835202002139>>.

ZAKARIA, Z.; PETROVIC, S. Genetic algorithms for match-up rescheduling of the flexible manufacturing systems. **Computers & Industrial Engineering**, v. 62, n. 2, p. 670-686, 2012. ISSN ISSN: 0360-8352 DOI: 10.1016/j.cie.2011.12.001. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0360835211003664>>.

ZHANG, L.; GAO, L.; LI, X. A hybrid intelligent algorithm and rescheduling technique for job shop scheduling problems with disruptions. **The International Journal of Advanced Manufacturing Technology**, v. 65, n. 5-8, p. 1141-1156, 2013. ISSN ISSN: 0268-3768 DOI: 10.1007/s00170-012-4245-6. Disponível em: <<http://dx.doi.org/10.1007/s00170-012-4245-6>>.

ZOBOLAS, G. I.; TARANTILIS, C. D.; IOANNOU, G. Minimizing makespan in permutation flow shop scheduling problems using a hybrid metaheuristic algorithm. **Computers & Operations Research**, v. 36, n. 4, p. 1249-1267, 2009. ISSN ISSN: 0305-0548 DOI: 10.1016/j.cor.2008.01.007. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0305054808000129>>.