

UNIVERSIDADE ESTADUAL PAULISTA “JÚLIO DE MESQUITA FILHO”
FACULDADE DE ENGENHARIA DE PRODUÇÃO DE BAURU
PROGRAMA DE PÓS-GRADUAÇÃO “*STRICTO SENSU*”

ANDERSON ROGÉRIO FAIA PINTO

ALGORITMO GENÉTICO APLICADO AO SEQUENCIAMENTO DE
***PICKING* E FATURAMENTO**

Bauru
2012

ANDERSON ROGÉRIO FAIA PINTO

**ALGORITMO GENÉTICO APLICADO AO SEQUENCIAMENTO DE
PICKING E FATURAMENTO**

Dissertação de Mestrado apresentada ao Programa de Pós Graduação em Engenharia de Produção da Faculdade de Engenharia de Bauru da Universidade Estadual Paulista “Júlio de Mesquita Filho”, como exigência parcial para obtenção do título de Mestre em Engenharia de Produção.

Área de Concentração: Gestão de Operações e Sistemas.

Orientador: Prof.º Dr. Antonio Fernando Crepaldi.

Bauru
2012

Pinto, Anderson Rogério Faia.

Algoritmo genético aplicado ao sequenciamento de *picking* e faturamento / Anderson Rogério Faia
Pinto, 2012

80 f.

Orientador: Antonio Fernando Crepaldi

Dissertação (Mestrado)-Universidade Estadual
Paulista. Faculdade de Engenharia, Bauru, 2012

1. Algoritmos Genéticos. 2. Processo de *Picking*.
3. Sequenciamento de Faturamento. I. Universidade
Estadual Paulista. Faculdade de Engenharia. II.
Título.

ATA DA DEFESA PÚBLICA DA DISSERTAÇÃO DE MESTRADO DE ANDERSON ROGÉRIO FAIA PINTO, DISCENTE DO PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO, DO(A) FACULDADE DE ENGENHARIA DE BAURU.

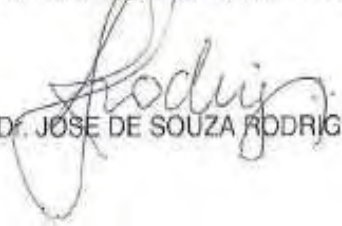
Aos 17 dias do mês de agosto do ano de 2012, às 09:30 horas, no(a) ANFITEATRO DA SEÇÃO DE PÓS-GRADUAÇÃO DA FACULDADE DE ENGENHARIA, reuniu-se a Comissão Examinadora da Defesa Pública, composta pelos seguintes membros: Prof. Dr. ANTONIO FERNANDO CREPALDI do(a) Departamento de Engenharia de Produção / Faculdade de Engenharia de Bauru - UNESP, Prof. Dr. ROGÉRIO ANDRADE FLAUZINO do(a) Departamento de Engenharia Elétrica / Escola de Engenharia de São Carlos - USP, Prof. Dr. JOSE DE SOUZA RODRIGUES do(a) Departamento de Engenharia de Produção / Faculdade de Engenharia de Bauru - UNESP, sob a presidência do primeiro, a fim de proceder a arguição pública da DISSERTAÇÃO DE MESTRADO de ANDERSON ROGÉRIO FAIA PINTO, intitulado "ALGORITMO GENÉTICO APLICADO AO SEQUENCIAMENTO DE PICKING E FATURAMENTO". Após a exposição, o discente foi argüido oralmente pelos membros da Comissão Examinadora, tendo recebido o conceito final: APROVADO. Nada mais havendo, foi lavrada a presente ata, que, após lida e aprovada, foi assinada pelos membros da Comissão Examinadora.



Prof. Dr. ANTONIO FERNANDO CREPALDI



Prof. Dr. ROGÉRIO ANDRADE FLAUZINO



Prof. Dr. JOSE DE SOUZA RODRIGUES

Dedico este trabalho aos meus pais Vagne e Cleide, que estiveram sempre do meu lado me apoiando nos momentos de dificuldades e me fazendo acreditar que era possível.

AGRADECIMENTOS

Agradeço a DEUS por me dar a graça da saúde e do discernimento!

À CAPES pelo apoio financeiro ao conceder a bolsa de estudo para realização desta pesquisa.

À Faculdade de Engenharia de Bauru pela oportunidade de realização do curso de mestrado.

Aos professores do Departamento de Engenharia de Produção pelos conhecimentos transmitidos com os quais deram a base para a realização deste trabalho.

Em especial, ao meu orientador, professor Dr. Antonio Fernando Crepaldi, pela confiança e amizade construída e por sua valorosa atenção nas correções, orientações e sugestões que tanto contribuíram para o enriquecimento e realização deste projeto.

Aos funcionários do Departamento de Engenharia de Produção, da Biblioteca, e da Seção de Pós Graduação da FEB/UNESP pela atenção e apoio administrativo, e a todos os servidores pelo pronto atendimento.

A todas as pessoas que direta ou indiretamente contribuíram para a realização desta pesquisa, familiares, amigos, professores e alunos de mestrado, em particular ao amigo Alex Oliva Borrasca pela ajuda no desenvolvimento do software.

À minha namorada Milene pelo apoio que têm me dado para concluir mais esta fase da minha vida, tanto nas horas difíceis quanto compartilhando dos momentos de conquistas.

E, finalmente, aos meus pais, Vagne e Cleide, e meus irmãos Jefferson e Rafael pelos conselhos e incentivos ao longo desse projeto desafiador, e pelo eterno orgulho de me ver evoluindo ao longo dos anos.

*“Tudo que você vivamente imaginar, ardentemente desejar,
sinceramente acreditar, e entusiasticamente colocar ação,
...inevitavelmente tornar-se-á realidade”*

Paul J. Meyer

RESUMO

PINTO, A. R. F. **Algoritmo genético aplicado ao sequenciamento de *picking* e faturamento**. 2012. 80 f. Dissertação de Mestrado (Mestrado em Engenharia de Produção) - Faculdade de Engenharia de Bauru, Universidade Estadual Paulista "Júlio de Mesquita Filho", Bauru, 2012.

As desordens e incertezas provocadas no decorrer do tempo, face à dinâmica das mudanças e a complexidade dos sistemas que abrangem as organizações, acarretam diversas situações em que os gestores necessitam encontrar soluções das quais seja possível extrair a maximização do resultado empresarial. Logo, o desenvolvimento de ferramentas que possam em dado momento apresentar, de forma ágil, um número mínimo de opções necessárias para investigar a incerteza é uma tarefa necessária em ambientes de negócios. Esta dissertação tem como objetivo a busca por uma solução para o problema do Sequenciamento Ótimo de Faturamento (SOF). A perspectiva adotada para a solução do SOF é o desenvolvimento de um software que automatize o processo de atribuição dos produtos aos pedidos em carteira, denominado como processo de *picking*. O trabalho emprega a Computação Evolucionária como método de adaptação ao problema e utiliza a técnica dos Algoritmos Genéticos (AG) na formulação do modelo de busca de soluções. A concepção do software dar-se-á pela interconexão de um conjunto de dados estáticos que contempla o estoque disponível para venda em um período pré-determinado de tempo t e a carteira de pedidos solicitados em diferentes datas. A representação binária é utilizada para formular a programação das estruturas heurísticas de possíveis soluções e o *Visual Basic for Applications* (VBA) do *Microsoft Office Excel* é empregado como ferramenta computacional para a implementação do modelo proposto. A programação considera as restrições e os parâmetros de decisão de forma que maximização do faturamento seja o resultado otimizado do problema. A implantação do software gera um módulo que automatiza o processo de *picking* e apresenta resultados otimizados para o SOF, o que prove agilidade e aperfeiçoa a tomada de decisão de faturamento. Conclui-se por meio de testes experimentais que o AG desenvolvido é uma opção viável à solução do problema de SOF.

Palavras-Chave: Algoritmos Genéticos; Processo de *Picking*; Sequenciamento de Faturamento.

ABSTRACT

PINTO, A. R. F. **Genetic algorithm applied to the sequencing of picking and billing**. 2012. 80 f. Master Dissertation (Master's Degree in Production Engineering) - Bauru School of Engineering, Universidade Estadual Paulista "Júlio de Mesquita Filho", Bauru, 2012.

The disorders and uncertainties caused in the course of time, given the dynamics of change and systems complexity which include organizations, result in several situations in which managers need to find solutions which can extract the maximization of the entrepreneurial outcome. Therefore, the development of tools that can, at a given time and in an agile way, present a minimum number of options necessary to investigate the uncertainties is a necessary task in business environments. This dissertation aims to search for a solution to the Optimal Sequencing Billing (OSB) problem. The perspective adopted for the solution of the OSB is the development of a software that automates the process of assigning products to backlog, named as “picking process”. The work employs the Evolutionary Computation as a method of adaptation to the problem and uses the technique of Genetic Algorithms (GA) in the formulation of the searching solutions model. The software design will come to be through the interconnection of a set of static data which includes the stock available for sale at a pre-determined period of time t and a backlog requested on different dates. The binary representation is used to formulate the scheduling heuristics structures of possible solutions and Visual Basic for Applications (VBA) in Microsoft Office Excel is a software tool used for the implementation of the proposed model. The program considers the constraints and decision parameters so that maximizing the billing is the result of optimized problem. The implementation of the software generates a module that automates the picking process and presents optimized results for the OSB, which provides agility and improves the decision making for billing. It was concluded by experimental tests that the GA developed is a viable option for solving the problem of OSB.

Keywords: Genetic Algorithms; Picking Process; Billing Sequencing.

LISTA DE FIGURAS

Figura 1 - Fluxograma de desenvolvimento da pesquisa	20
Figura 2 - Estrutura de um algoritmo evolutivo	24
Figura 3 - Diagrama que posiciona os algoritmos evolucionários como técnica de busca	24
Figura 4 - Principais elementos de um algoritmo genético	27
Figura 5 - Fluxograma de funcionamento de um algoritmo genético simples	29
Figura 6 - Esquema de um algoritmo genético simples.....	31
Figura 7 - Representação binária de um indivíduo.....	37
Figura 8 - Exemplo de seleção por roleta proporcional.....	44
Figura 9 - Diagrama do operador de <i>crossover</i> de um ponto	47
Figura 10 - Funcionamento do <i>crossover</i> uniforme	48
Figura 11 - Operador de mutação aleatória	50
Figura 12 - Operador de inversão	50
Figura 13 - Esquema gráfico do elitismo.....	53
Figura 14 - Representação do cromossomo do processo de <i>picking</i>	61
Figura 15 - Interface do <i>software</i>	67
Figura 16 - Exemplo de execução do AG.....	68
Figura 17 - Lista do processo de <i>picking</i>	70
Figura 18 - Lista de pendências a produzir.....	70
Figura 19 - Gráficos de convergência da primeira experimentação	71
Figura 20 - Gráficos de convergência da segunda experimentação	72
Figura 21 - Gráficos de convergência da terceira experimentação	73
Figura 22 - Gráficos de convergência da quarta experimentação	74

LISTA DE TABELAS

Tabela 1 - Exemplo de alfabetos de esquemas e cálculos das <i>strings</i> de <i>bits</i>	33
Tabela 2 - Aptidão de indivíduos	39
Tabela 3 - Exemplo de cálculo da aptidão acumulada	45
Tabela 4 - Estoque de produtos disponíveis para venda e lista de pedidos em carteira	59
Tabela 5 - Resultados estatísticos da primeira experimentação	71
Tabela 6 - Resultados estatísticos da segunda experimentação.....	72
Tabela 7 - Resultados estatísticos da terceira experimentação.....	73
Tabela 8 - Resultados estatísticos da quarta experimentação.....	74

LISTA DE QUADROS

Quadro 1 - Analogia de termos entre algoritmos genéticos e a biologia.....	26
Quadro 2 - Principais representações e os tipos de problemas onde melhor se aplicam.....	35
Quadro 3 - Lógica do algoritmo de implementação da roleta	43
Quadro 4 - Sequência de implementação e funcionamento do AG para o SOF	66

LISTA DE ABREVIATURAS E SIGLAS

- AE: Algoritmos evolucionários
- AG: Algoritmos genéticos
- CE: Computação evolucionária
- CFS: Sistemas classificadores
- NP: Não polinomial
- OG: Operador genético
- PE: Programação evolucionária
- PG: Programação genética
- SOF: Sequenciamento ótimo de faturamento
- VBA: *Visual Basic for Applications*

SUMÁRIO

1 INTRODUÇÃO	15
1.2. Problema de Pesquisa	17
1.3 Motivação e Justificativa	17
1.4. Objetivos.....	18
1.4.1. Objetivo Geral	18
1.4.2 Objetivos Específicos	18
1.5 Delimitação da Pesquisa.....	18
1.6 Método de Pesquisa Adotado	19
1.7 Estrutura da Dissertação	20
2 COMPUTAÇÃO EVOLUCIONÁRIA	21
2.1 Teoria da Evolução Aplicada à Computação	21
2.2 Conceitos Fundamentais da CE.....	22
2.3 Uma Breve História dos AEs.....	22
2.4 Conceitos Básicos dos AEs	23
3 ALGORITMOS GENÉTICOS	25
3.1 Uma Breve História do AG	25
3.2 Conceitos Básicos do AG.....	25
3.3 Funcionamento de um AG.....	29
3.4 Conceitos dos Esquemas	32
3.5 Representação do Cromossomo.....	35
3.5.1 Representação Binária	36
3.6 Geração da População Inicial	37
3.7 Função de Avaliação (Aptidão).....	39
3.8 Operadores Genéticos.....	40
3.9 Operador de Seleção	41
3.9.1 Seleção por Roleta (<i>Roulette Wheel</i>)	43
3.10 Operador de Cruzamento (<i>Crossover</i>).....	45
3.10.1 <i>Crossover</i> de Um Ponto.....	46
3.10.2 <i>Crossover</i> Uniforme	47
3.11 Operador de Mutação	48
3.12 Geração da Nova População (Atualização).....	51

3.12.1 Troca de Toda População (Substituição Geracional)	51
3.12.2 Troca de Toda a População com Elitismo	52
3.13 Parâmetros	53
3.14 Critérios de Parada.....	54
3.15 Verificação de Desempenho.....	54
3.16 Teorema Fundamental dos Esquemas	55
4 IMPLEMENTAÇÃO DO ALGORITMO GENÉTICO PROPOSTO	59
4.1 Representação do Problema.....	59
4.2 Faturamento Máximo (<i>FM</i>)	60
4.3 Representação do Cromossomo.....	61
4.4 Geração da População Inicial	61
4.5 Função de Aptidão.....	62
4.6 Operador de Seleção.....	64
4.7 Operador de <i>Crossover</i> e Mutação	65
4.8 Geração da Nova População.....	65
4.9 Critério de Parada	65
4.10 Metodologia de Implementação	66
5 EXPERIMENTAÇÃO DO SOFTWARE	67
5.1 Experimentação do <i>Software</i>	67
5.2 Relatórios Disponíveis.....	69
5.3 Análise de Desempenho	71
6 CONCLUSÃO.....	76
REFERÊNCIAS	78

1 INTRODUÇÃO

A dinâmica das mudanças em ambientes empresariais, cada vez mais complexos, e a busca por adaptações que proporcionem respostas imediatas ao mercado competitivo, fundamentam a importância de ações que identifiquem o número mínimo de opções necessárias para restringir, de certa maneira, a incerteza. Recentemente, tornou-se comum entre os pesquisadores a inspiração nos mecanismos genéticos evolutivos como uma tentativa de desenvolver ferramentas de mensuração quantitativa que possam, em dado momento, fornecer subsídios, de forma ágil, ao processo de otimização em tomadas de decisões.

No âmbito empresarial, os princípios de genética evolutiva se tornaram de grande importância e de uso crescente pela gestão organizacional como uma maneira de compreender as estratégias de negócios (MITCHELL; TAYLOR, 1999; PHILLIPS; SU, 2009). O avanço tecnológico e a incrível habilidade da evolução biológica em produzir estruturas ótimas motiva os cientistas a simularem estes mecanismos em problemas de otimização e adaptação (RUNARSSON; JONSSON, 1999).

A prática da gestão exige otimização e o desenvolvimento de ferramentas computacionais para esse fim tem se beneficiado muito com os avanços nos estudos da biotecnologia. Em especial, a Computação Evolucionária (CE) tem gerado soluções viáveis quando aplicada em problemas complexos de otimização e tomada de decisões, em uma ampla variedade de atividades e projetos, das diferentes áreas, dos mais diversos tipos de organizações (MITCHELL; TAYLOR, 1999; YANG, 2005; YANG; KOZIEL, 2010). Nesse sentido, a Engenharia de Produção têm se utilizado, cada vez mais, de conceitos e técnicas oriundas da CE.

Um problema relevante e ainda pouco abordado pela Engenharia de Produção é o Sequenciamento Ótimo de Faturamento, que para efeito de estudo é denominado de SOF. Por apresentar um grande número de possíveis combinações que devem ser obrigatoriamente avaliadas, o SOF caracteriza-se por ser um problema de natureza combinatorial complexa. Especificamente, esta complexidade é causada pelas restrições de quantidade e por parâmetros que a empresa define como critérios de decisão de faturamento, que podem ser a data de atendimento do pedido, questões logísticas na entrega, importância do cliente, etc.

Em meio aos diversos paradigmas de otimização computacional existentes, a Programação Dinâmica, por exemplo, é um método eficiente quando utilizado na resolução de problemas de otimização combinatorial e que poderia ser avaliado como uma provável técnica de solução para o SOF. No entanto, diante à natureza do problema, presume-se que o emprego

de uma técnica exata de otimização pode não ser uma opção viável, já que, o SOF pertence ao grupo dos problemas conhecidos pela comunidade científica como intratáveis, os titulados NP-completos, ou seja, aqueles em que ainda não se conhecem algoritmos polinomiais rápidos para resolvê-los de forma exata (LINDEN, 2008; ALMEIDA; VIANA; THOMAZ, 2009; YANG; KOZIEL, 2010).

Um exemplo clássico deste tipo de problema é o do caixeiro viajante, que estando em qualquer n cidade pode partir para qualquer uma das outras $n-1$ cidades para a segunda $n-2$ restantes até chegar à última. Isso resulta em um número de opções igual a $n(n-1)(n-2)\dots(2)(1)=n!$, onde para 100 cidades tem-se 10^{158} opções, o que levaria a um grande número de possíveis combinações a serem testadas. Teoricamente, um tempo igual a 10^{141} anos para encontrar a melhor solução (LINDEN, 2008; YANG; KOZIEL, 2010). Logo os tempos de respostas para modelos que possuam um grande número de variáveis e restrições, principalmente binárias, são ainda impraticáveis pelos métodos tradicionais de busca local. Além disso, o comportamento incerto sempre presente no mundo real dos sistemas faz com que os modelos de otimização sejam afetados por fatores e parâmetros complexos e, na prática, nem sempre é possível encontrar as melhores soluções. Normalmente, soluções viáveis, ou seja, boas o suficiente, e realizáveis em uma escala de tempo razoável são a escolha (YANG, 2005; LINDEN, 2008; ALMEIDA; VIANA; THOMAZ, 2009; YANG; KOZIEL, 2010).

Dentre as técnicas mais utilizadas na obtenção de soluções aproximadas, a Busca Tabu é uma abordagem de destaque na resolução de problemas de decisão sequencial a ser analisada como método de decisão do SOF. Todavia, conforme abordado por Linden (2008), o fato de ainda não haver um algoritmo polinomial rápido para resolver todos os problemas de otimização não-lineares de forma exata faz-se da CE uma opção apropriada. Um método alternativo de adaptação à complexidade pertencente ao campo da CE e muito utilizado pela comunidade científica são os AGs. Em problemas de atribuição os AGs têm se mostrado satisfatórios em comparação a outros métodos na resolução de problemas de natureza semelhante e sua eficácia é comprovada pelo grande número de trabalhos publicados pela comunidade científica das mais diversas áreas (MITCHELL; TAYLOR, 1999; YANG, 2005; LINDEN, 2008; ALMEIDA; VIANA; THOMAZ, 2009; YANG; KOZIEL, 2010).

Embora na investigação bibliográfica realizada não tenha sido encontrado nenhum trabalho em que se fez alusão ao AG para problemas de SOF, a sua aplicação prática nesta pesquisa é, portanto, fundamentada na resolução de problemas semelhantes apresentados na literatura disponível. A perspectiva adotada para a solução do SOF é a aplicação do AG à

automação do processo de *picking*, ou seja, na forma de designar quais produtos e quantidades serão faturados a cada cliente. Esse processo trata especificamente da atribuição ótima dos produtos disponíveis para venda à P_j pedidos visando satisfazer as necessidades da *CP* de forma ágil e eficiente, respeitando determinadas restrições e parâmetros de decisão pré-estabelecidos de forma que maximizar o faturamento seja o resultado otimizado para o SOF.

1.2 Problema de Pesquisa

Essencialmente, o SOF pode ser descrito como um problema de atribuição de n produtos para atender P_j pedidos dado o estoque de produtos disponíveis para a venda (*PA*) e a carteira de pedidos (*CP*) em determinado momento t . Seja, então, um conjunto de n produtos, de forma que o vetor X representa *PA*, $X = \{x_1, x_2, \dots, x_n\}$ e $CP = \{P_1, P_2, \dots, P_j\}$, os produtos devem ser agrupados em j pedidos não vazios e não sobrepostos. Logo, ao considerar que cada pedido pode demandar diferentes produtos e que um mesmo item pode ser requisitado por j pedidos, reduz-se o SOF a um problema de otimização, em que o resultado esperado é a maximização do faturamento, a depender do conjunto específico de parâmetros e critérios de decisão utilizados.

1.3 Motivação e Justificativa

Há uma vasta tecnologia de gestão que tem como objetivo oferecer ferramentas que apresentem um melhor gerenciamento de todos os recursos disponíveis às organizações. No entanto, em casos reais, problemas de SOF são ocasionados por restrições ou falhas na concepção dos sistemas de gestão, que podem ser atribuídas, em sua maioria, às desordens e incertezas provocadas no decorrer do tempo face à dinâmica das mudanças e a complexidade dos sistemas que abrangem as organizações. É, então, inevitável que os modelos de gestão nem sempre consigam prever toda a variabilidade do cenário empresarial ou encontrem soluções aceitáveis em curto espaço de tempo. Em determinadas situações, na medida em que se vai formando a *CP*, é comum a ocorrência de erros, além da emergência e a surpresa irreduzível sempre presente no ambiente de negócios, que afetam as decisões de faturamento e acarretam diversas situações em que se devam encontrar soluções para o SOF.

O fato é que a frequência de pedidos tem aumentado consideravelmente nos últimos anos, já que, objetivando reduções nos estoques, os clientes procuram fazer pedidos cada vez menores e com intervalos de tempo também menores, o que demanda um aumento no volume das operações de várias outras áreas dentro das organizações, em particular as financeiras, logísticas e de produção. Considerando que a carteira de pedidos é ininterruptamente renovada com novas demandas, alterações ou cancelamentos, e que, em dado momento, a

quantidade de produtos pode ser insuficiente para atender os diferentes pedidos, a busca pela melhor solução de um problema de SOF, pode se tornar uma tarefa complexa com a qual os gestores têm de lidar.

Dessa forma, resolver o SOF de forma satisfatória sem o uso de uma ferramenta adequada é, na maioria das vezes, muito difícil, de um modo geral demanda muito trabalho, dado às restrições e critérios de decisão de faturamento específicos a cada caso. Evidentemente que, erros no faturamento, atrasos ou pedidos incompletos, podem ocasionar trabalhos desnecessários, aumento de custos, insatisfação do cliente e perda nas vendas. Ao considerar que os gestores necessitam tomar decisões de forma ágil e que resultem em uma série de ações consistentes, acredita-se ser de grande valia o desenvolvimento de um estudo mais detalhado que ofereça uma solução satisfatória para o problema apresentado. Em síntese, a possibilidade de desenvolver uma ferramenta computacional capaz de oferecer uma solução automática satisfatória diante às eventuais ocorrências do problema apresentado motiva a realização desta pesquisa.

1.4 Objetivos

1.4.1 Objetivo geral

Este trabalho tem como objetivo principal desenvolver um *software* que automatize o processo de *picking* e apresente uma solução otimizada para o problema de SOF.

1.4.2 Objetivos Específicos

Os objetivos específicos deste trabalho são:

1. Aperfeiçoar o processo de tomada de decisão de faturamento;
2. Minimizar os custos operacionais do processo de *picking* e faturamento;
3. Maximizar o faturamento de acordo com os parâmetros pré-estabelecidos;
4. Possibilitar que a área de produção tenha uma visão atualizada das necessidades de demanda.

1.5 Delimitação da Pesquisa

A busca por uma solução otimizada para o problema de SOF utilizando AG é o enfoque dado a esta pesquisa. No entanto, a existência de diferentes organizações, atuando nos mais variados mercados, cada qual com suas peculiaridades e critérios decisórios específicos que mapeiem suas políticas diversas, faz com que o SOF seja um problema de difícil generalização. Em concomitância com os objetivos apresentados, o escopo desta pesquisa é o desenvolvimento de um software que apresente soluções otimizadas para

problemas de SOF específicos às organizações que necessitam realizar o processo de *picking* antes do faturamento.

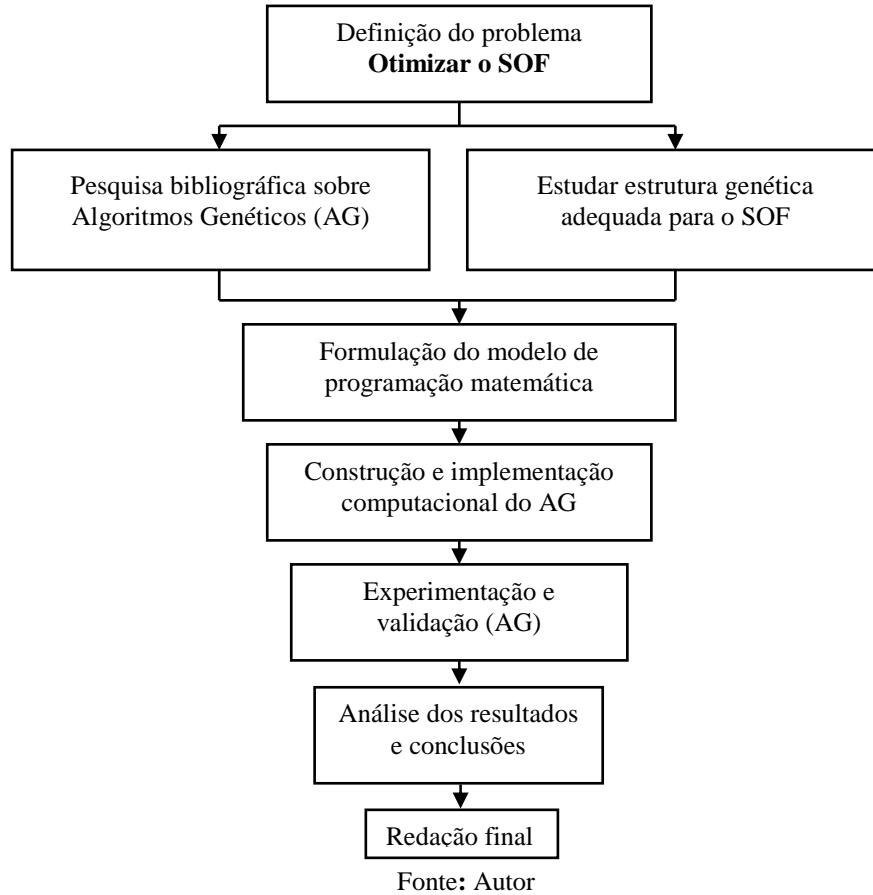
1.6 Método de Pesquisa

Esta dissertação classifica-se como sendo uma pesquisa aplicada no qual emprega o método dedutivo por meio de uma abordagem quantitativa (GIL, 1991; BERTO; NAKANO, 1998; BERTO; NAKANO, 1999; BERTRAND; FRANSOO, 2002; GAVIRA; 2003; MARCONI; LAKATOS, 2003; SILVA; MENEZES, 2005). Em relação à formulação do AG, este segue metodologicamente as fases práticas abordadas no referencial teórico, sendo o *Visual Basic for Applications (VBA)* do *Microsoft Office Excel* a ferramenta utilizada para a programação computacional do modelo proposto. O software será implementado e submetido a testes por meio de um microcomputador com processador Core I5 de 2.3GHz, 8 GByte de RAM e 750 GByte de HD. Na concepção do programa os parâmetros de decisão serão pré-configurados de forma que não poderá haver processo de *picking* de um produto sem saldo de estoque ou de quantidades parciais quando não aceito pelo cliente. O sistema irá fornecer relatórios de *picking* para faturamento e do que é necessário produzir para que todos os clientes sejam atendidos conforme os prazos negociados, sendo que, a preferência de faturamento será por ordem crescente de data de entrega dos pedidos.

Em síntese, o AG é implementado por meio de um conjunto de parâmetros de forma que a obtenção da melhor solução para o SOF está sujeita às soluções otimizadas de P_j pedidos que compõem a CP . Dessa forma, a partir de uma solução inicial não satisfatória, o algoritmo repete ciclos de evolução para encontrar outra solução melhor que a anterior até que se satisfaça o critério de parada. A representação binária aleatória é usada para formular as estruturas heurísticas de atribuição dos produtos aos pedidos, onde o número de gerações é dado pelo usuário para no máximo 200 indivíduos a cada execução. A função aptidão penalizará soluções implausíveis por meio de punições corretivas correspondentes ao valor de faturamento de cada *bit* infrator. O método de seleção adotado é o da roleta proporcional que realizará a seleção de 50% do total de indivíduos gerados. O cruzamento será realizado pelo operador de *crossover* de um ponto, com probabilidade fixada em 100% dos pais selecionados, já o operador de mutação será aplicado a 50% dos filhos resultantes do *crossover* pela técnica de troca aleatória dos *bits*, no qual, o usuário terá a opção de informar a probabilidade desejada. A troca de toda a população com elitismo é a técnica adotada para a substituição da população onde somente o melhor indivíduo é transferido integralmente para a próxima geração. O critério de parada é determinado pelo número de gerações especificadas pelo usuário ou a chegada ao valor ótimo da função de aptidão. As análises da funcionalidade

do software serão obtidas em função das experimentações realizadas e a avaliação final pelo conjunto dos resultados alcançados. A figura 1 apresenta o fluxograma com as etapas do desenvolvimento desta pesquisa.

Figura 1 - Fluxograma de desenvolvimento da pesquisa



1.7 Estrutura da Dissertação

A dissertação está estruturada em capítulos, resumidos da seguinte forma: No capítulo 2 é feita uma breve explanação sobre a Computação Evolucionária (CE) e os Algoritmos Evolucionários (AEs), abordando desde os fatos históricos até seus conceitos e métodos de funcionamento. O capítulo 3 apresenta uma revisão bibliográfica sobre a teoria e aplicação dos Algoritmos Genéticos (AGs) ao qual fornece um arcabouço teórico que provê o suporte à formulação da estrutura genética adequada para o desenvolvimento do modelo e fundamenta os passos necessários à solução do problema que se pretende estudar. O capítulo 4 exprime a modelagem matemática e a implementação computacional do software desenvolvido. O capítulo 5 traz a interface do programa, as simulações experimentais e a análise conjunta dos resultados obtidos a cada execução. O último capítulo apresenta as conclusões relativas ao objetivo do trabalho, as principais contribuições e as sugestões de continuidade para futuros estudos sobre o assunto.

2 COMPUTAÇÃO EVOLUCIONÁRIA

Este capítulo apresenta uma breve explanação sobre a teoria da evolução genética e das pesquisas que inspiraram suas aplicações para o desenvolvimento da Computação Evolucionária (CE), abordando conceitos e fatos históricos desde sua constituição e publicação. Ao longo da seção são apresentados os tipos de Algoritmos Evolucionários (AEs) e suas estruturas funcionais comuns. O objetivo não é detalhar toda a CE e suas técnicas, mas fixar uma base teórica que facilite um melhor entendimento dos Algoritmos Genéticos (AGs).

2.1 Teoria da Evolução Aplicada à Computação

Aceita por praticamente toda a comunidade acadêmica mundial, a moderna teoria da evolução combina a genética e as ideias de Darwin para dominar a biologia, e graças ao progresso científico hoje se sabe que o armazenamento e a transmissão das informações genéticas funcionam em nível molecular. Continuamente, os avanços nos estudos e simulações dos processos de evolução genética enriquecem a visão da ciência sobre a complexidade dos mecanismos moleculares ao explicar a níveis mais profundos porque a evolução acontece (BOWLER, 2004; CARVALHO, 2004; LINDEN, 2008; PHILLIPS; SU, 2009; AYALA, 2010).

Em contrapartida, a recente explosão de conhecimentos e técnicas na área da biotecnologia direcionadas a compreender as características gerais de sistemas adaptativos complexos tem afetado também, já há algum tempo, direções futuras na previsão de novas tecnologias. Os pesquisadores discutem cada vez mais como estas técnicas podem ser aplicadas a outras áreas científicas onde a adaptabilidade às mudanças é observada e necessária ao bom funcionamento do sistema (MITCHELL; TAYLOR, 1999; BOWLER, 2004; PHILLIPS; SU, 2009).

Para os cientistas, os conceitos de genética e seleção natural passaram a não ser mais dirigidos somente a organismos biológicos. Desde que justificados com apoio de uma teoria sólida com evidência empírica, esses conceitos podem ser aplicados a qualquer organismo que se reproduz de modo a envolver tanto a hereditariedade como a variação (MICHALEWICZ, 1996; LINDEN, 2008; GHISELIN, 2009).

A área de inteligência artificial tem se beneficiado muito com a inspiração nestes mecanismos. Em especial as várias abordagens desenvolvidas de forma independente para explorar o ramo da computação, chamadas coletivamente de computação evolucionária (CE), parecem se adequar a capacidade biológica de pesquisar em paralelo todo o espaço de busca para encontrar melhores soluções candidatas a resolução de problemas específicos. A

aplicação de diversos algoritmos evolucionários (AEs) tem proporcionado grandes progressos na resolução de problemas computacionais nas mais variadas áreas (MITCHELL, 1996; MITCHELL; TAYLOR, 1999; YANG, 2005; YANG; KOZIEL, 2010; LINDEN, 2008).

2.2 Conceitos Fundamentais da CE

A CE foi escolhida como o termo geral que engloba todas as áreas de computação inspiradas na natureza evolutiva e biológica. É uma ciência que aplica conceitos da natureza, por meio de paradigmas computacionais dos processos genéticos de seleção natural, como uma ferramenta alternativa para a solução de problemas computacionais complexos do mundo real de forma otimizada (MITCHELL; TAYLOR, 1999; YANG, 2005; LINDEN, 2008).

O objetivo final da CE é criar algoritmos inteligentes que, inspirados na teoria evolutiva e biológica, tenham a capacidade de aprender e se adaptar à necessidade de determinado ambiente do qual são componentes, de forma que ao longo do processo evolutivo seja possível gerar indivíduos que sejam cada vez melhores às soluções de problemas específicos. Segundo Von Zuben (2000), a família de algoritmos que englobam toda a CE são coletivamente denominados de Algoritmos Evolucionários (AEs).

2.3 Uma Breve História da CE

A primeira tentativa de representação da teoria de Darwin por meio de um modelo matemático surgiu em 1930 com o livro de Fisher “*The Genetic Theory of Natural Selection*”. Já na década de 40, muitos cientistas começaram a tentar se inspirar na natureza para criar o ramo da inteligência artificial. No entanto, foi nos anos 50 que diversos cientistas evolucionistas buscavam de forma independente desenvolver sistemas artificiais inspirados em sistemas naturais. Neste período, alguns biólogos usaram computadores com a finalidade de realizar experimentos para simular a evolução, e na área da engenharia eram desenvolvidas ferramentas de otimização para resolver problemas que eram insolúveis computacionalmente até aquela época (GOLDBERG, 1989; OBITKO, 1998; MITCHELL; TAYLOR, 1999; CARVALHO, 2004; YANG, 2005; LINDEN 2008).

Em 1960, Ingo Rechenberg com seu trabalho “*Evolutionsstrategie*” introduziu a Computação Evolucionária (CE). Em termos históricos, três AEs foram desenvolvidos independentemente e formaram a espinha dorsal do campo da CE: Algoritmos genéticos (AG), inventado por Holland em 1962; Programação evolucionária (PE), desenvolvida em 1962 por Fogel, Owens e Walsh, na qual candidatos a soluções de tarefas eram representados como máquinas de estados finitos; Estratégia evolucionária (EE), concebidas por Ingo Rechenberg em 1963, desenvolvidas por Hans-Paul Schwefel, e mais tarde utilizadas por

Peter Bienert para resolver problemas de otimização técnica (GOLDBERG, 1989; BÄCK; SCHWEFEL, 1993; MITCHELL, 1996; OBITKO, 1998; MITCHELL; TAYLOR, 1999; VON ZUBEN, 2000; CARVALHO, 2004; GROSKO; GORSKI; DIAS, 2006; LINDEN 2008).

No ano de 1989 Booker, Goldberg e Holland inventaram um método que combina a adaptação de aprendizagem e evolução que foi denominado como Sistemas de Classificadores (CFS). Em 1992 John Koza usou AGs para gerar e desenvolver automaticamente funções e programas computacionais para realizar certas tarefas e chamou seu método de Programação Genética (PG). Atualmente, existem cinco diferentes paradigmas de computação inspirados na natureza evolutiva: AG, PE, EE, CFS e PG (BÄCK; SCHWEFEL, 1993; MITCHELL, 1996; OBITKO, 1998; VON ZUBEN, 2000; CARVALHO, 2004; LINDEN 2008).

2.4 Conceitos Básicos dos AEs

Os AEs são baseados na ideia central de processos de aprendizagem coletiva, por meio da evolução de uma população de indivíduos potenciais ou candidatos à solução de um determinado problema, usando operadores inspirados na variação genética e seleção natural (GOLDBERG, 1989; BÄCK; SCHWEFEL, 1993; MITCHELL, 1996; MITCHELL; TAYLOR, 1999; VON ZUBEN, 2000; LINDEN 2008).

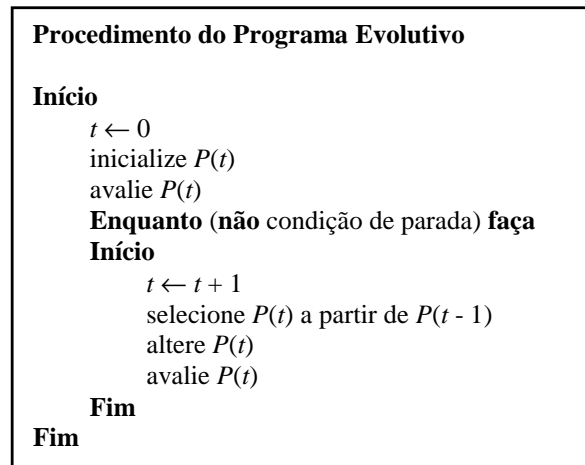
A principal característica dos AEs é a possibilidade de serem aplicados na solução de problemas complexos, via um conjunto de técnicas e procedimentos genéricos adaptáveis, onde muitos destes problemas requerem a busca por um espaço enorme de possibilidades de soluções. A vantagem mais significativa está na sua robustez e flexibilidade, ao possibilitar resolver problemas pela simples descrição matemática do que se quer ver presente na solução. (MITCHELL; TAYLOR, 1999; VON ZUBEN, 2000; LINDEN 2008).

É peculiar na aplicação dos AEs não haver a necessidade de se indicar explicitamente todas as suas etapas de implementação, pois as ferramentas de propósitos gerais são as mesmas para uma ampla gama de problemas, com apenas certas especificidades a cada caso (VON ZUBEN, 2000). Um mesmo AG, por exemplo, pode ser usado para descobrir o máximo de toda e qualquer função de n variáveis sem nenhuma alteração das estruturas de dados e procedimentos adotados, alterando-se, apenas, a sua função de avaliação (LINDEN 2008).

Dessa maneira, apesar de todos os AEs partilharem de uma estrutura conceitual comum, não devem ser considerados “prontos para uso”, mas sim um elenco de procedimentos gerais que podem ser prontamente adaptados a cada contexto de aplicação (VON ZUBEN, 2000; LINDEN 2008). Seja $P(t)=\{x^t_1, \dots, x^t_n\}$ a população de cromossomos

na geração t , na qual cada indivíduo é um candidato à solução do problema em questão, a estrutura básica de busca pela melhor solução de um AE é demonstrada na figura 2 (MICHALEWICZ, 1996).

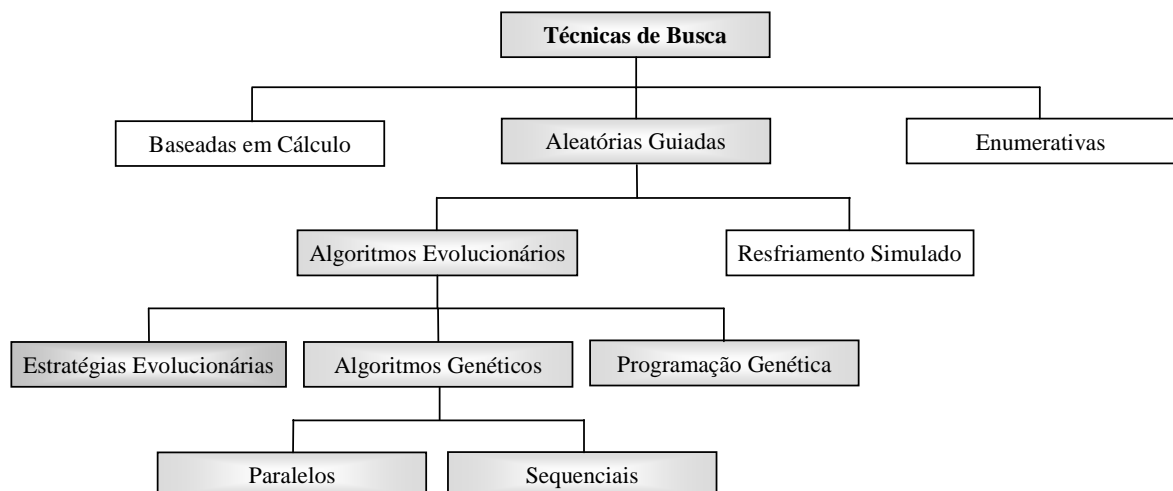
Figura 2 - Estrutura de um algoritmo evolutivo



Fonte: (MICHALEWICZ, 1996)

Em síntese, os AEs são heurísticas de otimização global que fazem parte de um ramo de busca da computação chamado de Técnicas Aleatórias Guiadas. Ainda que sejam métodos fortemente estocásticos, não são totalmente aleatórios, pois usam informações do estado corrente para guiar a pesquisa (LINDEN, 2008). A figura 3 ilustra o diagrama que posiciona os AEs como técnica de busca, e o capítulo 3 descreverá em detalhes todo o funcionamento dos AGs, uma vez que, representa a técnica de solução escolhida para estudo e aplicação ao problema de pesquisa.

Figura 3 - Diagrama que posiciona os algoritmos evolucionários como técnica de busca



Fonte: Adaptado de Linden (2008)

3 ALGORITMOS GENÉTICOS

Este capítulo apresenta uma revisão bibliográfica a respeito da teoria e aplicação dos Algoritmos Genéticos (AGs). Ao longo das seções são discutidas e explicadas as principais técnicas utilizadas em AGs. O objetivo não é esgotar o assunto ou torná-lo extenuante, mas fornecer um embasamento teórico suficiente para o desenvolvimento de um modelo de solução do problema que se pretende pesquisar.

3.1 Uma Breve História do AG

O AG foi criado por John Holland na década de 60 em conjunto com seus alunos e colegas da Universidade de Michigan. Em contraste à EE e à PE, o objetivo de Holland não era desenvolver projetos de algoritmos para resolver problemas específicos. O escopo, até então, era estudar formalmente os fenômenos de adaptação, evolução e seleção natural como ocorre na natureza, e desenvolver formas de modelos heurísticos em que estes mecanismos (*inputs* sensoriais do ambiente) pudessem ser matematicamente formalizados e importados para sistemas computacionais. Ao invés de utilizar valores reais, Holland utilizou cadeias binárias, em que a mutação e o *crossover* se tornaram fontes de variações aleatórias de *bits* na população. Em 1975, ele publicou sua inovação no livro “*Adaptation in Natural and Artificial Systems*”, hoje considerada uma referência bibliográfica de extrema relevância, onde formalizou e fundamentou matematicamente suas técnicas por meio de um firme alicerce teórico baseado em “esquemas”, conforme será abordado na seção 3.4. No mesmo ano De Jong (1975) utilizou pela primeira vez os AGs em otimização de parâmetros, estabelecendo suas bases de aplicações técnicas. Em 1989, David Goldberg populariza os AG’s ao editar “*Genetic Algorithms in Search, Optimization and Machine Learning*”. Hoje em dia, numerosas modificações do AG canônico descrevem algo muito longe da sua concepção original, e são aplicados a muito mais campos conforme indicado por Holland (MITCHELL, 1996; OBITKO, 1998; BÄCK; SCHWEFEL, 1999; MITCHELL; TAYLOR, 1999; VON ZUBEN, 2000; CARVALHO, 2004; LINDEN 2008).

3.2 Conceitos Básicos do AG

Os AGs são um ramo dos AEs, e podem ser definidos como um método genérico adaptativo de busca que imita o processo genético e Darwiniano de evolução natural, por meio da seleção, reprodução e sobrevivência dos mais aptos entre estruturas de *strings*, com uma troca estruturada, ainda que aleatória, de informações (PACHECO, 1999; GOLDBERG, 1989; LINDEN 2008).

Diferente das técnicas convencionais, os AGs são heurísticas de otimização global em que se utiliza um conjunto inicial randômico de soluções chamado de população, onde cada indivíduo desta população é candidato à solução do problema. Assim, a função de otimização representa o ambiente no qual a população inicial se encontra. Os cromossomos artificiais são compostos por cadeias de *bits*, usualmente referidos como genes, e os possíveis valores que um determinado gene pode assumir são denominados alelos (GOLDBERG, 1989; MITCHELL, 1996; GEN; CHENG, 1997; MITCHELL; TAYLOR, 1999; PACHECO, 1999). O quadro 1 apresenta uma analogia entre os termos usados em AGs originados da biologia e o sistema natural.

Quadro 1 - Analogia de termos entre algoritmos genéticos e a biologia

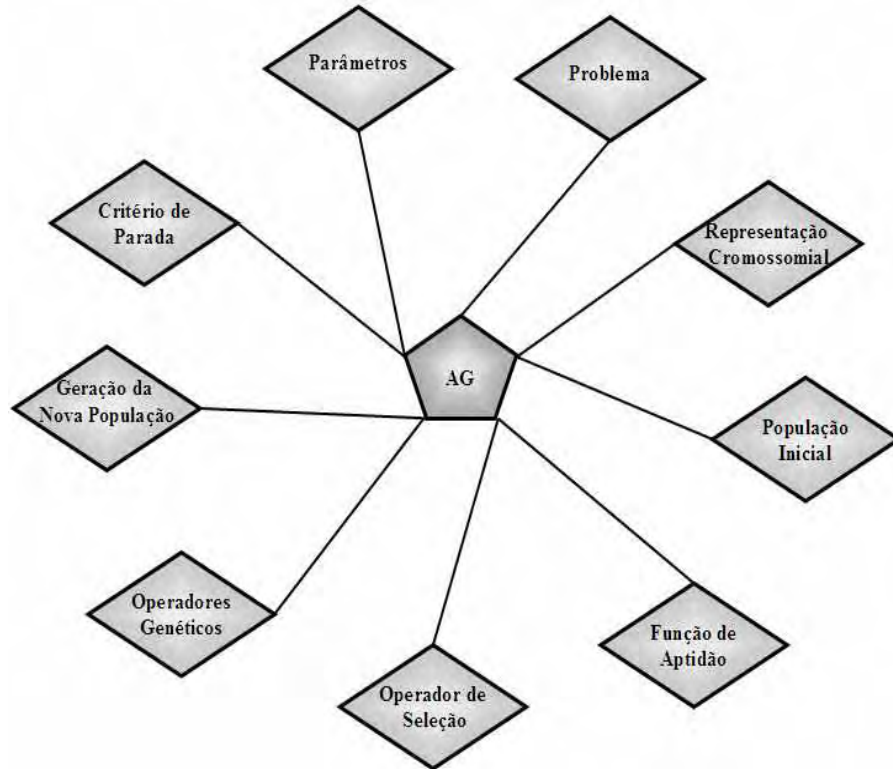
Termo	Biologia	Algoritmos Genéticos
Cromossomo	Conjunto de genes de um organismo que constituem um ser vivo.	Indivíduo, String (de <i>bits</i>) que representa uma solução para o problema (variáveis e parâmetros são combinados).
Gene	Unidade de hereditariedade transmitida pelo cromossomo e que controla as características do organismo.	Parâmetro codificado no cromossomo, ou seja, um elemento da <i>string</i> (podem assumir valores binários, inteiros ou reais).
Genótipo	Composição genética contida no cromossomo.	Informação contida no cromossomo (estrutura).
Fenótipo	Manifestação física de um gene no organismo, por exemplo, pele lisa ou rugosa.	Estrutura construída a partir do genótipo (interação com o ambiente que determina o conjunto de parâmetros).
Alelo	Uma das formas alternativas de um gene.	Valores que o gene pode assumir.
Locus	Local fixo num cromossomo onde está localizado determinado gene ou marcador genético.	Posição onde está localizado um elemento da <i>string</i> , o “gene”.
Geração	Designação do grupo de indivíduos que têm os mesmos progenitores.	Ciclo por meio dos operadores genéticos

Fonte: Adaptado de Lacerda, Carvalho e Ludermir (1999); Pacheco (1999)

A fundamentação teórica do AG baseia-se, então, na analogia de melhorar uma população de soluções de determinado problema por meio do desenvolvimento de modelos de mudanças evolutivas do capital genético dos cromossomos (DE JONG, 1988; BIEGLER; GROSSMANN, 2004; GROSKO; GORSKI; DIAS, 2006). Os principais elementos da

aplicação prática de um AG são apresentados na figura 4 e posteriormente detalhados no decorrer deste capítulo.

Figura 4 - Principais elementos de um algoritmo genético



Fonte: Adaptado de Argoud (2007)

Todos os elementos ilustrados se interagem sequencialmente durante todo o processo de execução do algoritmo, cada qual com suas operações básicas específicas. A busca por soluções é realizada de forma paralela e estruturada utilizando-se de técnicas de transição probabilísticas de forma que é possível guiar a busca em regiões de pontos de “alta aptidão” (melhor solução), e que tem como objetivo encontrar o extremo (mínimo ou máximo) de uma função (GOLDBERG, 1989; CARVALHO, 2004; LINDEN 2008).

Por serem métodos de busca aleatória ignoram o aproveitamento de regiões promissoras do espaço de soluções e não ficam estagnados simplesmente pelo fato de terem encontrado um máximo local, pelo contrário, continuam em busca de indivíduos mais aptos ou soluções que sejam globalmente mais significativas. Segundo Von Zuben (2000) e Linden (2008) estas questões opõem os AGs aos métodos como gradiente (*hill climbing*) que facilmente se prendem a máximos locais sem realizar uma exploração do espaço de busca global.

O AG realiza a busca a partir de um grande número de pontos que representam possíveis soluções dentro de todo o espaço de soluções (universo de soluções), e não de um

ponto único. No entanto, não procura em todos os pontos possíveis, mas sim em um subconjunto destes pontos, chegando a uma sequência de diferentes subpopulações antes de apresentar a melhor solução (GOLDBERG, 1989; OBITKO, 1998; MITCHELL; TAYLOR, 1999; LINDEN, 2008).

Apesar de serem algoritmos probabilísticos e apresentarem etapas não-determinísticas em seu desenvolvimento, não são puramente aleatórios, pois combinam elementos de procura direcionada e estocástica explorando informações históricas dos resultados já obtidos. Ou seja, o AG utiliza informações da função objetivo ou de aptidão para encontrar novos pontos de busca onde são esperados melhores desempenhos (VON ZUBEN, 2000; CARVALHO, 2005; LINDEN, 2008).

É possível concluir que os AGs são sistemas não lineares com comportamento fortemente ecológico que combinam mudanças aleatórias com processos probabilísticos, são, portanto, estocásticos. Iniciando um AG com a mesma população e o mesmo conjunto de parâmetros os resultados encontrados raramente são reproduzíveis ou, necessariamente, os mesmos. A mutação, por exemplo, raramente gera indivíduos iguais, o que dificilmente repete um mesmo resultado de um experimento para outro (TANOMARU, 1995; PACHECO, 1999; LINDEN, 2008).

Dessa forma, por ser uma técnica probabilística, não constitui um algoritmo de busca da solução ótima, o AG pode então encontrar boas soluções, mas não assegura a obtenção do melhor resultado em todas as suas execuções (TANOMARU, 1995; YANG, 2005). Porém, dentre as principais características que fazem do AG uma técnica de busca bem sucedida (DE JONG, 1988; GOLDBERG, 1989; BÄCK; SCHWEFEL, 1993; MICHALEWICZ, 1996; MITCHELL; TAYLOR, 1999; CARVALHO, 2004; LINDEN, 2008) destacam-se:

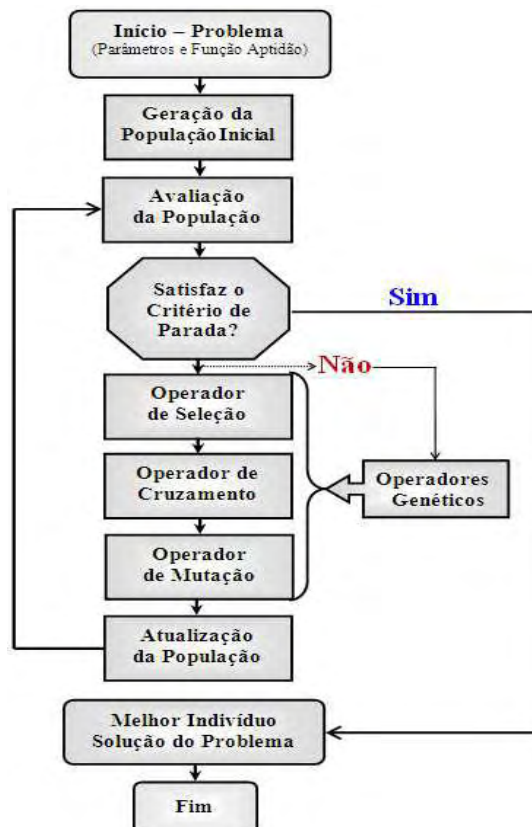
- A simplicidade das operações, facilidade de implementação e um bom desempenho em sua execução;
- Trabalha com a codificação do conjunto de parâmetros em uma *string* e não com os próprios parâmetros;
- São muito flexíveis, aceitam sem grandes dificuldades uma infinidade de alterações e permitem fácil hibridização;
- Não são totalmente aleatórios, fazem uma busca paralela de forma dirigida e simultânea;

- São extremamente eficientes no seu objetivo de varrer o espaço de soluções na busca da região onde provavelmente encontra-se o máximo ou o mínimo global;
- São capazes de lidarem com funções discretas e contínuas sem serem afetados por descontinuidades na sua função ou em suas derivadas;
- Fornecem um conjunto de heurísticas de pesquisa eficiente apresentando uma melhoria significativa sobre os métodos tradicionais sem a necessidade de incorporação de conhecimentos altamente específicos.

3.3 Funcionamento de um AG

O funcionamento do AG pode ser descrito como um processo contínuo que repete ciclos de evolução controlados por parâmetros e critérios de parada pré-definidos pelo usuário (PACHECO, 1999). Por ser baseada na evolução biológica, cada iteração de todo este processo é chamado de geração e todo o conjunto de sucessivas gerações a serem testadas por uma função de aptidão, visando encontrar uma solução (adaptação) satisfatória aos critérios de parada, é denominado de execução. A figura 5 esboça o fluxograma de funcionamento de um AG simples.

Figura 5 – Fluxograma de funcionamento de um algoritmo genético simples



Fonte: Autor

Na figura 5 é possível observar que quando a condição de parada é atendida o algoritmo para e retorna o melhor indivíduo como a solução do problema, quando não, o algoritmo irá executar um incremento de iterações que consiste na aplicação dos OGs à atual população, na qual irá gerar uma nova solução que, do mesmo modo que a anterior, será submetida à função de avaliação, e esse processo irá se repetir, enquanto necessário, durante todo o processo de execução do algoritmo.

Conforme demonstrado, a essência do funcionamento de um AG é toda fundamentada na técnica de geração e teste, utilizando-se de informações de custo ou recompensa para realizar uma procura contínua por melhores resultados. Inicialmente, fundamentadas por um conjunto pré-determinado de parâmetros, as informações sobre o problema são representadas nos cromossomos de acordo com o esquema de codificação escolhido. A partir daí, são gerados aleatoriamente n indivíduos que irão compor uma população inicial (OBITKO 1998; LACERDA; CARVALHO; LUDERMIR, 1999; MITCHELL; TAYLOR, 1999; YANG, 2005; LINDEN, 2008).

Cada indivíduo gerado recebe, então, uma avaliação denominada de aptidão, que é uma quantificação da sua qualidade como solução do problema em questão. O operador de seleção preconiza os melhores indivíduos, ou seja, todo o processo depende da aptidão da *string* de forma a simular a sobrevivência dos mais aptos. Uma porcentagem dos cromossomos mais adaptados é, então, mantida, enquanto os restantes são descartados (MITCHELL, 1996; PACHECO, 1999; CARVALHO, 2004; LINDEN, 2008).

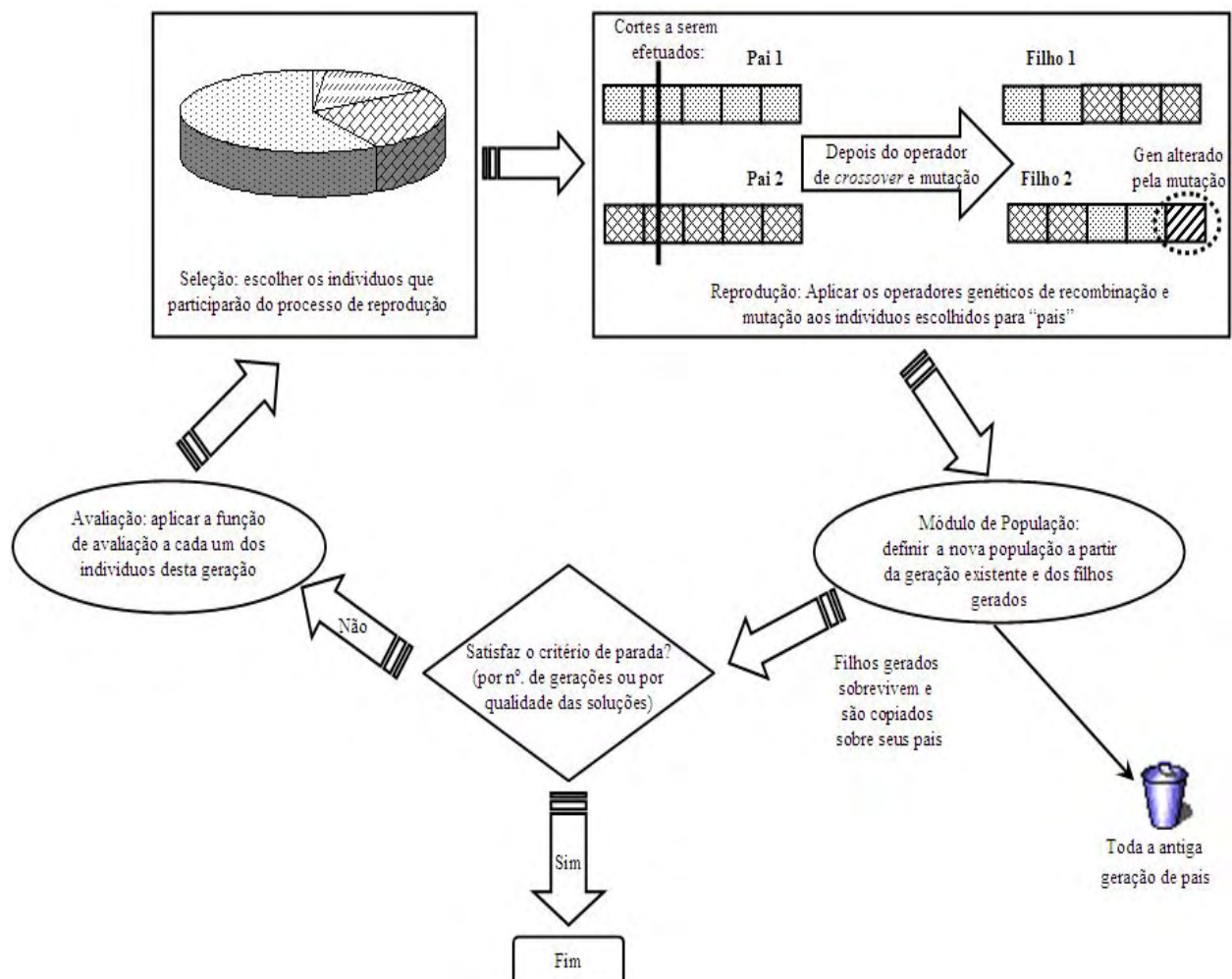
Baseado no processo de avaliação, os cromossomos selecionados são, posteriormente, submetidos a um processo evolucionário de modificação genética por meio de operadores genéticos (OGs), tais como: recombinação sexual (*crossover*) e mutação. A reprodução é equivalente à sexuada e a mutação corrobora para que haja diversidade, evitando que haja convergência prematura (BÄCK; SCHWEFEL, 1993; PACHECO, 1999; VON ZUBEN, 2000; BIEGLER; GROSSMANN, 2004; GRSKO; GORSKI; DIAS, 2006; LINDEN, 2008).

Esse processo de avaliação se encarregará de fazer com que a população evolua, já que, os indivíduos com melhor grau de adaptação terão maiores chances de sobreviver e repassar o seu material genético para as próximas gerações. Assim, em cada geração de uma nova população, um novo conjunto de *strings* artificiais é criado usando *bits* e partes aptas da geração anterior de forma a semear a população inicial da geração seguinte com as melhores soluções encontradas anteriormente (LACERDA; CARVALHO; LUDERMIR, 1999; CARVALHO, 2004; YANG, 2005; ZUKHRI; OMAR, 2006; ROSA; LUZ, 2009).

Ocasionalmente a nova população de indivíduos e sua eficácia na resolução do problema é novamente testada pelo processo de avaliação, que deverá eventualmente gerar um novo indivíduo no qual resultará na mais adequada (a sobrevivente), se não a ótima solução para o problema (OBITKO 1998; MITCHELL; TAYLOR, 1999; YANG, 2005; LINDEN, 2008).

O término do algoritmo é estabelecido por um critério de parada. Após um número de gerações, a condição de parada deve ser atendida, a qual geralmente indica a existência, na população, de um indivíduo que represente, se não a melhor, uma solução aceitável para o problema (BÄCK; SCHWEFEL, 1993; MICHALEWICZ, 1996; PACHECO, 1999; VON ZUBEN, 2000; GRSKO; GORSKI; DIAS, 2006; LINDEN, 2008). Todo o diagrama de funcionamento básico de um AG canônico, conforme descrito anteriormente, é demonstrado na figura 6.

Figura 6 - Esquema final de um algoritmo genético simples



Fonte: Adaptado de Linden (2008)

3.4 Conceitos dos Esquemas

A teoria dos esquemas (*schema*) foi formulada por Holland (1975) como uma forma de explicar, por meio de um modelo matemático, como e por que os AGs funcionam. O *schema* é demonstrado como um padrão genético capaz de ilustrar todo o processo de como um conjunto de cromossomos que apresentam similaridades em certas posições consegue, de forma simultânea, realizar uma varredura que explora todo o espaço de busca presente em determinado ambiente de soluções (GOLDBERG, 1989; PACHECO, 1999; WHITLEY, 1994 *apud* GROSKO; GORSKI; DIAS, 2006).

O *schema* é uma *string* $s = \{s_1 s_2 \dots s_n\}$, de comprimento n , em que as posições pertencem ao conjunto Γ (alfabeto) utilizado na representação, mais o símbolo $\{*\}$, que significa “não-importa” (*don't care* ou *wildcard*). Cada posição da *string* dada por $s_k \neq *$ é chamada de especificação, enquanto que um *wildcard* demonstra o fato de que aquela posição pode assumir qualquer elemento dentro do conjunto Γ , e que o valor ali representado não terá relevância para a solução (MITCHELL, 1996; MICHALEWICZ, 1996; LINDEN, 2008).

Em síntese, um esquema consiste em um gabarito (“*template*”) que representa um subconjunto dentro o conjunto de todos os indivíduos possíveis, descrevendo quais genomas são idênticos no espaço de busca. No caso binário, o Γ pode ser descrito como um modelo dado por $\{0, 1, *\}$ que tem cardinalidade igual a 3, onde o símbolo $\{*\}$ denota a possibilidade do gene poder assumir tanto o valor 0 como o 1 (HOLLAND, 1975; MITCHELL, 1996; MICHALEWICZ, 1996; PREBYS, 1999; LINDEN, 2008).

Em AG, para representar o *schemata* (plural de *schema*) utiliza-se o símbolo de somatório “ Σ ”. Matematicamente emprega-se o H como um padrão que descreve todos os cromossomos do espaço de busca; K para representar o número de símbolos do alfabeto e L o comprimento do cromossomo (DIAS; BARRETO, 1998; PACHECO, 1999; GROSKO; GORSKI; DIAS, 2006; LINDEN, 2008). Assim, o esquema $H \in \{0, 1, *\}^L$ é uma descrição de um modelo de similaridade ou um hiperplano L -dimensional do espaço de *bits*. (BÄCK; SCHWEFEL, 1993; MITCHELL, 1996).

No Γ binário $\{0, 1, *\}$, sendo $L = 2$, o espaço de busca é igual a K^L , então, se tem exatamente $2^2 = 4$ soluções (pontos) possíveis. Considerando que para cada uma das L posições de um indivíduo define-se um *schema* diferente usando o símbolo $*$, para um espaço de busca representado por K^L , existem, então, $(K+1)^L$ *schemata*, portanto, um total de $(2+1)^2 = 9$ esquemas. Dessa forma, para o exemplo dado têm-se os seguintes *schemata*: 00, 01, 10, 11, 1*, 0*, *1, **, *0 (GOLDBERG, 1989; PACHECO, 1999; LINDEN, 2008).

Isto implica que, quando é usada a representação binária, um esquema que tenha comprimento L com m posições, incluindo o símbolo *, terá m graus de liberdade e representará até 2^m indivíduos diferentes da atual população. Se o alfabeto K contiver n símbolos, e o esquema m posições com *, então o esquema será representado por $(n-1)^m$ indivíduos. Observe que a retirada de 1 elemento $(n-1)$ provém do fato de que o * não entrará na composição dos cromossomos (GOLDBERG, 1989; PACHECO, 1999; LINDEN, 2008).

O número de esquemas presente em um indivíduo é, então, dependente do comprimento da *string* e do número de opções contidas no Γ de codificações. Dessa forma, um indivíduo pertence a um *schema* se para todas as L posições o símbolo do indivíduo é igual ao símbolo do *schema*, exceto nas posições onde o símbolo do *schema* é “*don't care*”, ou seja, um *schema* possui $2^{L-O(H)}$ indivíduos que são quantificados por duas importantes propriedades (HOLLAND, 1975; GOLDBERG, 1989; BÄCK; SCHWEFEL, 1993; MITCHELL, 1996; PACHECO, 1999; PREBYS, 1999; LINDEN, 2008):

- $O(H)$ - denota a ordem de H , ou seja, a quantidade de posições (genes) que assumem valores 0's e 1's, ou seja, diferentes de * ;
- $\delta(H)$ - denota a definição de comprimento, ou seja, representa a distância entre a primeira e a última posição fixa em termos de números de corte existentes em Σ , ou o número de pontos de corte, diferente de *, dentro do esquema.

Para que se possa transmitir um melhor entendimento de como e porque os AGs funcionam, a tabela 1 apresenta um exemplo matemático para três esquemas baseados na representação binária. A seguir é explicado todo o procedimento de cálculo do esquema $H = 1*0*1$, que, de acordo com Pacheco (1999), pode ser chamado de instância do esquema H , já que representa um modelo onde todos os cromossomos se iniciam e terminam com 1.

Tabela 1 - Exemplo de alfabetos de esquemas e cálculos das *strings* de bits

Esquemas (Σ)	Indivíduos do <i>Schema</i> (Quantidade e Representação)		Propriedade do <i>Schema</i>		Graus de Liberdade m	Espaço de Soluções K^L	<i>Schemata</i> por Indivíduo (Quantidade e Representação)	
	$2^{L-O(H)}$	Representação	$O(H)$	$\delta(H)$			$(K+1)^L$	Representação
1*	2	10, 11	1	0	1	4	9	00, 01, 10, 11, 1*, 0*, *1, *0, **
1*0*1	4	10001, 10011, 11001, 11011	3	4	2	32	243	0000, 1000, 1100, 1110, ..., ****
0	4	000, 010, 100, 110	1	0	2	8	27	000, 100, 110, 111, 010, ..., **

Fonte: Adaptado de Pacheco (1999) e Linden (2008)

De acordo com a tabela 1, o esquema $H = 1*0*1$ terá $4=(2^{5-3})$ possíveis representantes que podem ou não estarem presentes em determinada geração. A sua ordem é 3, já que existem três elementos diferentes de * na *string* H . O seu comprimento é 4, pois contém 4

pontos de corte entre a primeira posição fixa e a última. Possui 2 graus de liberdade, que é referente ao número de ocorrências de *. O espaço de busca K^L para cada um dos 4 indivíduos que representam o *schema* é de $32=(2^5)$ pontos de possíveis de soluções, sendo capaz de gerar $(2+1)^5 = 243$, dado que se têm três elementos (0,1,*) para 5 posições diferentes na *string* H .

Conforme demonstrado, cada *string* na população é um exemplo de diferentes esquemas contidos uns nos outros, e se cada esquema gera certa quantidade de indivíduos, conseqüentemente, este número previsto será superior ao tamanho da população. Logo, se a população é suficientemente grande e inicialmente escolhida ao acaso, diversos esquemas podem ser manipulados em paralelo, o que permite ao AG explorar similaridades em codificações arbitrárias. Assim, o processo de busca por melhores soluções gera outras possíveis soluções à medida que a evolução acontece (HOLLAND, 1975; LINDEN, 2008).

Portanto, o paralelismo implícito nos AGs não está apenas no fato de que uma população contendo vários indivíduos é manipulada simultaneamente. Existe paralelismo também embutido no fato de que para cada elemento da população de um AG canônico processam-se e manipulam-se, simultaneamente, dezenas, ou mesmo centenas de esquemas com características negativas ou positivas, e que podem levar a uma boa ou má avaliação. O AG calcula explicitamente a avaliação de n indivíduos da população corrente, mas, implicitamente, calcula a avaliação de um número maior de esquemas que são instanciados por cada indivíduo da população (MITCHELL, 1996; LINDEN, 2008).

Numa população de n indivíduos, onde cada indivíduo representa 2^L *schemata*, há entre 2^L e $n \cdot 2^L$ *schemata*, dependendo da diversidade da população. O número de *schemata* processados a cada geração é, então, proporcional a n^3 , enquanto avaliam n indivíduos. Nota-se que há mais informações nos *schemata* para guiar a busca do que simplesmente nos indivíduos. Isso quer dizer que o importante é o esquema e não o indivíduo, se este morrer o esquema que o torna bom tende a se proliferar e continuar na população (HOLLAND, 1975; PACHECO, 1999; LINDEN, 2008).

A teoria dos esquemas corrobora que os AGs constituem uma classe de métodos de busca de propósito geral e que por meio de heurísticas poderosas têm como principal recurso a capacidade de identificar, explorar e acumular informações sobre um espaço de pesquisa inicialmente desconhecido, e que podem convergir rapidamente para soluções de alta qualidade em espaços complexos. Em síntese, a principal particularidade dos AGs esta na

busca para ajustar o equilíbrio ou tensão entre dois objetivos aparentemente conflitantes (HOLLAND, 1975; DE JONG, 1988; GOLDBERG, 1989; BÄCK; SCHWEFEL, 1993; MICHALEWICZ, 1996; MITCHELL; TAYLOR, 1999; LINDEN, 2008):

- *Exploration* (exploração): exploração do espaço de busca no sentido de investigar novas áreas (adaptações) de regiões ainda desconhecidas;
- *Exploitation* (aproveitamento): obter o máximo proveito do conhecimento adquirido e das melhores soluções em pontos mais promissores já visitados do espaço de busca (adaptações feitas até a atual geração).

3.5 Representação do Cromossomo

A representação genotípica corresponde a uma descrição de cada indivíduo da população por meio de uma lista ordenada (cromossomo) de atributos descritos a partir de um alfabeto finito. Cada atributo desta cadeia representa um pedaço indivisível equivalente a um gene que por sua vez armazena informações de mensuração quantitativa nos alelos para representar uma das possíveis soluções do problema (PACHECO, 1999; VON ZUBEN, 2000; LINDEN, 2008).

O tamanho da lista está diretamente associado ao número mínimo de atributos necessários para descrever cada indivíduo da população, geralmente têm tamanho único, embora existam abordagens que permitem o tratamento de cromossomos de tamanho variável (PACHECO, 1999; VON ZUBEN, 2000; LINDEN, 2008).

A representação cromossomial é completamente arbitrária e consiste em traduzir a informação do problema em uma maneira viável de ser tratada pelo computador. Existem diversos tipos e formas de codificações que podem variar de acordo com a estrutura de dados imaginada pelo programador (OBIKO, 1998; PACHECO, 1999; ARGOUD, 2007; LINDEN, 2008). O quadro 2 apresenta as principais representações e os tipos de problemas onde melhor se aplicam.

Quadro 2 - Principais representações e os tipos de problemas onde melhor se aplicam

Representações	Problemas	Exemplos
Binária	Inteiros	(100110011)
Números Reais	Numéricos	(43.2 -33.1... 0.0 89.2)
Permutação de Símbolos	Baseados em Ordem	(E4 E3 E7 E2 E6 E1 E5)
Lista de Regras	Grupamento	(R1 R2 R3... R22 R23)

Fonte: Adaptado de Pacheco (1999) e Linden (2008)

Apesar do AG ser uma heurística poderosa para lidar com problemas complexos, a representação cromossomial é uma das etapas mais críticas e fundamentais para sua definição e desempenho. Quanto mais adequada ao problema, maior a qualidade dos resultados obtidos, quando não, pode levar a problemas de convergência prematura ou exigir um esforço computacional intensivo, tornando-o ineficaz ou intratável. Portanto, se houver soluções proibidas ao problema estas não devem ter uma representação, se o problema impuser condições de algum tipo, estas devem estar implícitas dentro da representação (DE JONG, 1988; OBITKO, 1998; PACHECO, 1999; LINDEN, 2008).

Dessa forma, embora arbitrária, a escolha do tipo de representação a ser usada deve ser realizada em função do problema que se quer resolver, do mecanismo de *feedback* e do que essencialmente se deseja manipular geneticamente. Em geral, as regras para a escolha da melhor forma de representação são: deve ser a mais simples possível e capaz de representar todo o espaço de busca do problema; deve utilizar um alfabeto mínimo que permita a expressão natural do que se deseja resolver e deve prestigiar a formação de *schemata* curtos e de baixa ordem (DE JONG, 1988; OBITKO, 1998; PACHECO, 1999; ARGOUD, 2007; LINDEN, 2008).

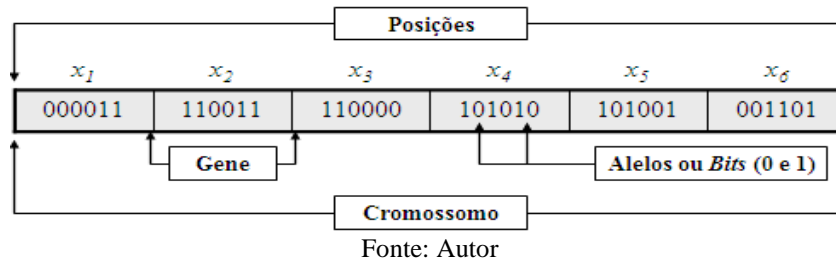
3.5.1 Representação Binária

A representação binária, proposta por Holland (1975), é a mais utilizada pela maioria dos pesquisadores, e por ter sido mais frequentemente usada, apresenta um histórico de pesquisas que servem de modelos referenciais e que permitiram um avanço aos estudos e aplicação deste tipo de codificação. Embora apresente algumas desvantagens, quando aplicada a problemas com alta dimensionalidade de variáveis contínuas e alta precisão numérica, é a mais simples e a mais bem executável por um AG canônico. As principais vantagens deste tipo de representação resumem-se na facilidade de ser transformada em números inteiros ou números reais e de ser tratada pelos OGs; na capacidade de manipular esquemas de forma eficiente, oferecendo o número máximo de *schemata* e maximizando o paralelismo implícito; e, ainda por facilitar a prova de alguns teoremas (HOLLAND 1975; DE JONG, 1988; BÄCK; SCHWEFEL, 1993; PACHECO, 1999; MICHALEWICZ, 1996; LINDEN, 2008).

Na codificação binária, um indivíduo x é representado por uma sequência de *bits* que pode assumir valores 0 ou 1, onde cada alelo é um *bit*. Esta sequência é representada por $s = [b_n \dots b_2 b_1]$,

onde n é o número de *bits* necessários para representar x e cada $b_i \in 0,1$ (TANOMARU, 1995; MICHALEWICZ, 1996; PACHECO, 1999). A figura 7 demonstra a representação binária de um indivíduo composto por seis genes representados pelas variáveis x_1, x_2, \dots, x_6 , onde os primeiros b_i ($i=1$ a 6) *bits* representam x_1 , os b_i ($i=7$ a 12) *bits* seguintes representam x_2 e assim por diante.

Figura 7 - Representação binária de um indivíduo



Fonte: Autor

De acordo com a figura 7, cada um dos genes é mensurado por seis alelos ou *bits* que determinam um valor de solução entre os vários possíveis do espaço de busca deste problema. Segundo Linden (2008) é normal que a quantidade de alelos para os x_n sejam iguais. Assim, $b_{x1} = b_{x2} = \dots = b_{xn}$, mas é possível que as necessidades de precisão, para cada número que estes representam, sejam diferentes, o que faz com que esta igualdade não seja obrigatória.

A definição do tamanho do cromossomo sempre dependerá da precisão requerida para cada variável. Deste modo, para representar números reais como números binários é preciso se determinar dois parâmetros que em conjunto definem quantos *bits* por variável devem ser usados, são eles: a faixa de operação de cada uma das variáveis e a precisão desejada (TANOMARU, 1995; MICHALEWICZ, 1996; LINDEN, 2008).

3.6 Geração da População Inicial

Um AG tem início a partir da execução do operador denominado inicialização. Tal operador consiste na criação de uma população inicial para o primeiro ciclo de solução do algoritmo. A população de cromossomos consiste de uma matriz com duas dimensões, podendo ser representada como $N_{pop} \times N_{bits}$, onde N_{pop} é a quantidade de indivíduos de uma determinada população e N_{bits} é o tamanho de cada cromossomo (PACHECO, 1999; HAUPT, RANDY; HAUPT, SUE, 2004; ROSA; LUZ, 2009).

A literatura apresenta várias formas de geração da população inicial, sendo que, a mais comum e simples é a geração randômica para cada indivíduo. Embora nenhum conhecimento de domínio seja necessário, existem também formas determinísticas, em que os cromossomos são gerados de acordo com uma função heurística (BÄCK; SCHWEFEL, 1993; HAUPT, RANDY; HAUPT, SUE, 2004; LINDEN, 2008; ROSA; LUZ, 2009).

Em geral, para uma evolução mais rápida, se busca semear os bons cromossomos, quando se conhece, a priori, o valor das boas “sementes” com certos níveis de desempenho. O ideal é que a população inicial contenha cromossomos representativos de todo o espaço de soluções para que se aumente a diversidade genética, garantindo assim, uma boa distribuição das soluções e maior alcance do espaço de busca. (DE JONG, 1988; LACERDA; CARVALHO; LUDERMIR, 1999; ARGOUD, 2007; LINDEN, 2008; ROSA; LUZ, 2009).

O problema é que a capacidade de armazenamento demandada por um AG é exponencial e proporcional ao número de indivíduos presentes na população, ou seja, o total de avaliações em um experimento será igual ao tamanho da população vezes número de gerações a executar. A análise de desempenho do AG é, então, extremamente sensível ao tamanho da população, e isso faz com que o processo de dimensionamento do tamanho ideal incorra em duas situações complexas e conflitantes (DIAS; BARRETO, 1998; GROSKO; GORSKI; DIAS, 2006; ARGOUD, 2007; LINDEN, 2008).

Isto ocorre porque, caso a população seja muito pequena não haverá espaço para que se tenha variedade genética, o que pode levar à convergência prematura e tornar o AG incapaz de encontrar boas soluções. Em contrapartida, quanto maior a quantidade de cromossomos maior a chance de encontrar melhores resultados, porém, se essa for grande demais, maior será o tempo de processamento, o que pode se aproximar de uma busca exaustiva. Logo, a demora em conseguir uma resposta pode fazer com que o AG deixe de ser um método interessante (DIAS; BARRETO, 1998; GROSKO; GORSKI; DIAS, 2006; ARGOUD, 2007; LINDEN, 2008).

Evidentemente, há um limite superior para o tamanho da população onde se verifica a melhora no desempenho do AG. No entanto, a literatura disponível não especifica um número ideal, em geral, a maioria dos trabalhos publicados utiliza 100 indivíduos inicialmente, mas isto irá depender do tipo de problema, da experiência e heurística utilizada pelo usuário e da realização de testes (LACERDA; CARVALHO; LUDERMIR, 1999; ARGOUD, 2007; LINDEN, 2008; ROSA; LUZ, 2009).

Para Tanomaru (1995) uma população entre 50-200 indivíduos é suficiente para resolver a maioria dos problemas. De acordo com Cole (1998) *apud* Argoud (2007) o tamanho ideal, especificamente em problemas de agrupamento, pode variar entre 10 e 1000. Já Linden (2008) sugere como tentativa inicial (extremamente imprecisa) uma quantidade de 40 cromossomos para qualquer tipo de problema. O mesmo autor apresenta ainda, como uma ideia para minimizar este tipo de problema, algumas estratégias de tamanho variável, cuja população aumente ou diminua com o tempo.

3.7 Função de Avaliação (Aptidão)

A função de avaliação, de custos ou de adaptabilidade é a maneira utilizada pelos AGs para se determinar uma medida de qualidade ou habilidade de adaptação dos indivíduos gerados ao ambiente específico. A aptidão de um cromossomo é uma forma de diferenciar (métrica de qualidade) entre as boas e as soluções sub-ótimas, deixando claro qual delas está mais próxima, ou quão apto determinado cromossomo está da solução procurada (MITCHELL, 1996; PACHECO, 1999; CARVALHO, 2004; LINDEN, 2008; ROSA; LUZ, 2009).

O cálculo da função de aptidão é o único elo não genérico entre o AG e o problema proposto e seu valor é posteriormente utilizado pelo operador de seleção como um parâmetro que irá dirigir o processo de busca para solucionar o problema em questão. Uma vez que o critério de seleção atua sobre a aptidão, em vez de valores da função objetivo, os valores de aptidão são utilizados como resultado do processo de avaliação (BÄCK; SCHWEFEL, 1993; PACHECO, 1999; GROSKO; GORSKI; DIAS, 2006; LINDEN, 2008; ROSA; LUZ, 2009).

Nota-se que a formulação desta função é uma ação complexa que depende totalmente do problema a resolver. Logo, deve ser representada por meio de uma função capaz de identificar os objetivos específicos e embutir todas as restrições do problema (proporcional e gradual à sua gravidade) além de aplicar, quando necessário, punições aos cromossomos que as desrespeitarem (PACHECO, 1999; CARVALHO, 2004; GROSKO; GORSKI; DIAS, 2006; LINDEN, 2008; ROSA; LUZ, 2009).

A função de aptidão atribui, então, uma nota ou um índice a cada indivíduo da população corrente no qual representa um valor numérico que reflete quão bem determinado indivíduo resolve o problema (GROSKO; GORSKI; DIAS, 2006; LINDEN, 2008; ROSA; LUZ, 2009). A tabela 2 apresenta um exemplo dado por Pacheco (1999) cuja função de avaliação dada por $f(x) = x^2$ mede a aptidão de cada indivíduo. De acordo com essa função, a avaliação deve ser tal que, se o cromossomo C_1 representa uma solução melhor do que o cromossomo C_2 , então a avaliação de C_1 deve ser maior do que a avaliação de C_2 (LINDEN, 2008).

Tabela 2 - Aptidão de indivíduos.

	Cromossomo	x	f(x)
C_1	001001	9	81
C_2	000100	4	16

Fonte: Pacheco (1999)

Conforme o exemplo dado na tabela 2, ao se aplicar a função de aptidão ao número inteiro correspondente ao cromossomo binário, o resultado demonstra que C_1 é um indivíduo mais apto que C_2 , comprovando, assim, que a função de avaliação é para um AG o que o meio ambiente é para um ser vivo, ou seja, quando determinado indivíduo não se encontra apto para sobreviver ali, tende à extinção (LINDEN, 2008; ROSA; LUZ, 2009).

Portanto, a avaliação simula o papel da pressão exercida pelo ambiente sobre o indivíduo, por meio do processo Darwiniano de seleção e sobrevivência dos mais aptos, sendo também usado para selecionar os cromossomos para reprodução. Ao manter uma porcentagem dos cromossomos mais aptos, estes terão maior chance de passar seu material genético para as próximas gerações para que se obtenham melhores descendentes (GOLDBERG, 1989; MITCHELL 1996; MICHALEWICZ, 1996; OBITKO, 1998; PACHECO, 1999; CARVALHO, 2004; GROSKO; GORSKI; DIAS, 2006; LINDEN, 2008).

É, então, possível afirmar, que o valor exato fornecido por uma função de avaliação (*fitness is evaluation*) é a característica inerente à causa da convergência genética, visto que, pode fazer com que o desempenho do AG se degenere caso ocorra o “super-indivíduo” ou a “competição próxima” (MICHALEWICZ, 1996; LINDEN, 2008).

O super-indivíduo se dá quando a avaliação de um cromossomo apresenta uma aptidão muito superior à média dos outros membros da população, que em casos extremos são próximos do ótimo global. A competição próxima ocorre quando os indivíduos apresentam valores de aptidão que diferem muito pouco, percentualmente, tornando a seleção praticamente aleatória, no qual bons esquemas podem deixar de ser selecionados (GOLDBERG, 1989; TANOMARU, 1995; LACERDA; CARVALHO; LUDERMIR, 1999; PACHECO, 1999; LINDEN, 2008).

3.8 Operadores Genéticos (OGs)

Em AG, os OGs são os principais mecanismos de busca em regiões desconhecidas do espaço de soluções (LACERDA; CARVALHO; LUDERMIR, 1999). Os OGs consistem em um processo evolucionário que se consolida pelo uso de técnicas de randomização, baseadas na aptidão dos cromossomos, obtidas por meio de iterações computacionais que mimetizam fenômenos biológicos de seres vivos, no qual novos indivíduos são criados a partir da troca de material genético (HOLLAND, 1975; PACHECO, 1999; LINDEN, 2008).

A função dos OGs é fazer com que a população inicial sofra sucessivas transformações a cada nova geração e se diversifique sem que ocorra a perda das características genéticas de adaptação adquiridas pelas gerações anteriores (ambos os genitores). A ideia é fazer com que

o AG melhora com o tempo, gerando indivíduos cada vez mais aptos, o que estende o espaço de busca e contribui para que as populações evoluam no decorrer das gerações até que se chegue a um resultado satisfatório (GOLDBERG, 1989; PACHECO, 1999; ROSA; LUZ, 2009).

A ação dos OGs modifica as características dos cromossomos de forma a gerar uma nova descendência de diferentes indivíduos. Após vários ciclos de evolução a população deverá conter indivíduos mais aptos, que por sua vez incluirão mais descendentes, e terão uma maior chance de perpetuarem seus códigos genéticos nas próximas gerações (GOLDBERG, 1989; PACHECO, 1999).

Existem diversos OGs com diferentes técnicas e específicos a cada caso, no qual suas aplicações estão intimamente relacionadas com as decisões da forma de representação que seja a mais adequada ao tipo de problema que se quer resolver (DE JONG, 1988). Os OGs classificam-se em (PACHECO, 1999; YANG, 2005; LINDEN, 2008):

- Operadores de seleção (seleciona os indivíduos para o *crossover* e a mutação);
- Operadores de recombinação (reprodução sexuada - *crossover*);
- Operadores de mutação (mutação genética);
- Operadores específicos ao domínio do problema (quaisquer outros que a imaginação dos programadores consiga produzir).

É importante frisar que não há uma probabilidade que seja adequada para os operadores de *crossover* e mutação durante toda a execução do algoritmo. Outra característica relevante é que estes operadores são independentes e, portanto, suas percentagens não precisam somar 100%. Estudos mais avançados permitem a ação destes operadores com probabilidade variáveis, no qual é possível fazer dois sorteios com aplicações independentes e com percentagens não relacionadas permitindo, assim, explorar de forma mais eficiente o espaço de soluções (PACHECO, 1999; LINDEN, 2008).

Normalmente, no início do AG, se busca executar muita reprodução e pouca mutação e após um grande número de gerações os indivíduos tendem a apresentar, na sua maioria, características muito similares, podendo ocorrer a convergência genética. A partir daí, torna-se extremamente interessante que o operador de mutação seja escolhido mais frequentemente para dispersar a população, trazendo novo material genético para a formação de melhores indivíduos (PACHECO, 1999; LINDEN, 2008).

3.9 Operador de Seleção

A seleção é executada logo após o cálculo de aptidão dos cromossomos gerados. É por meio deste operador que os indivíduos mais aptos da população são selecionados

(cromossomos pais) com base em suas avaliações para a reprodução, e que irão gerar descendentes (cromossomos filhos) por meio da aplicação dos operadores de cruzamento (*crossover*) e mutação (TANOMARU, 1995; MITCHELL, 1996; PACHECO, 1999; ARGOUD, 2007; ROSA; LUZ, 2009).

Este operador simula o mecanismo de seleção natural, portanto, é baseado na competição e no favorecimento dos indivíduos com maiores valores de aptidão. A função deste operador é muito importante, se não houvesse o processo de seleção, além de o AG perder grande parte do caráter evolutivo, o mesmo seria um processo ineficiente e similar a uma busca aleatória (DE JONG, 1988; TANOMARU, 1995; YANG, 2005; ROSA; LUZ, 2009).

A seleção pode ser qualquer função crescente na qual cada estrutura da população tenha uma aptidão associada a uma avaliação orientada (DE JONG, 1988; PREBYS, 1999). A seleção do AG canônico enfatiza uma regra de sobrevivência probabilística atrelada a um valor de aptidão para produzir mais ou menos descendentes. Portanto, conforme o princípio Darwiniano de seleção, quanto melhor a avaliação do cromossomo, maior é a sua probabilidade de permanecer na nova geração e ser selecionado mais vezes para reprodução (BÄCK; SCHWEFEL, 1993; MITCHELL, 1996; LINDEN, 2008).

Dessa forma, ao permitir aos pais mais aptos a possibilidade de gerar mais filhos, espera-se que as boas características predominem dentro das novas populações. É interessante notar que o fato do operador de seleção canônico utilizar-se de uma regra probabilística não impede que pais menos aptos também possam ser selecionados e gerar descendentes, porém em menor escala. Assim, é possível que nesse processo haja também certo nível de diversificação positiva, visto que, até os piores cromossomos podem conter certas características genéticas que em conjunto com outros gerem indivíduos ainda melhores (MITCHELL, 1996; PACHECO, 1999; YANG, 2005; LINDEN, 2008; ROSA; LUZ, 2009).

O principal método de seleção utilizado em AG é a roleta, no entanto, para que se tenham ferramentas alternativas que se adéquem aos diversos problemas, existem vários outros métodos de seleção presentes na literatura. O fato é que ainda não há um consenso em relação a qual método é melhor, é uma questão em aberto, vai depender muito do tipo de problema que se quer resolver (MITCHELL, 1996; LINDEN, 2008).

Porém, para qualquer mecanismo de seleção utilizado, o final do processo de escolha dos indivíduos é sempre o mesmo da roleta. As principais técnicas de seleção comumente usadas são: roleta acumulada e proporcional, *ranking*, torneios, truncamento, normalização

linear e normalização exponencial, amostragem estocástica uniforme e por seleção local (BLICKLE, 1996; PACHECO, 1999; ARGOUD, 2007; LINDEN, 2008).

Um mecanismo de seleção é caracterizado pela pressão seletiva ou intensidade de seleção que o mesmo introduz no AG (PACHECO, 1999). A definição de intensidade de seleção empregada em genética é a variação na aptidão média da população induzida pelo método de seleção ou a razão entre o cromossomo de maior aptidão e a aptidão média da população (BLICKLE, 1996; LACERDA; CARVALHO; LUDERMIR, 1999).

Se a pressão seletiva é muito baixa a aptidão é praticamente igual para toda a população e o AG apresenta um comportamento aleatório devido à competição próxima. Se a pressão seletiva é alta os super-indivíduos terão vários descendentes nas próximas gerações, enquanto que os cromossomos com baixo valor de avaliação poderão ser extintos. Consequentemente, o AG tenderá a convergir rapidamente e provavelmente para um máximo local (TANOMARU, 1995; LACERDA; CARVALHO; LUDERMIR, 1999).

3.9.1 Seleção por Roleta (*Roulette Wheel*)

O método de seleção dos indivíduos pela Roleta (*Roulette Wheel*) foi inicialmente utilizado no AG clássico apresentado por Holland (1975). Este método é baseado no valor relativo de aptidão de cada indivíduo em relação à soma da avaliação de todos os cromossomos da população. Assim, a seleção dos indivíduos ocorre de forma proporcional ao seu valor de aptidão (MITCHELL, 1996; MICHALEWICZ, 1996; PACHECO, 1999; ROSA; LUZ, 2009).

É um método simples e fácil de se entender, por isso é muito adotado. Neste método, cria-se uma roleta dividida em n faixas, onde n é o número de cromossomos representados. Cada faixa é rotulada de 1 a n , e o comprimento da circunferência do arco é proporcional ao espaço representado pelo valor de aptidão que cada cromossomo ocupa na roleta, que somadas as frações totalizam 100% (TANOMARU, 1995; ARGOUD, 2007; LINDEN, 2008; ROSA; LUZ, 2009). O quadro 3 descreve a lógica do algoritmo de implementação da roleta.

Quadro 3 - Lógica do algoritmo de implementação da roleta

- | |
|---|
| <p>(a) Some todas as avaliações para uma variável soma
 (b) Ordene todos os indivíduos em ordem crescente de avaliação (opcional)
 (c) Selecione um número s entre 0 e soma (Não incluídos)
 (d) $i=1$
 (e) $aux=$avaliação do indivíduo 1
 (f) enquanto $aux < s$
 (g) $i = i + 1$
 (h) $aux=aux+$avaliação do indivíduo i
 (i) fim enquanto</p> |
|---|

Fonte: Linden (2008)

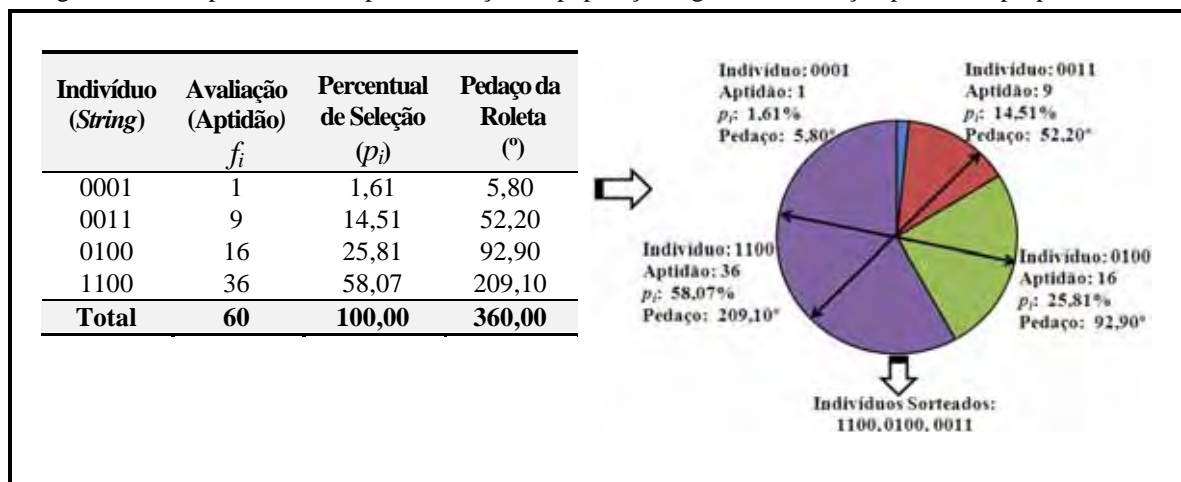
De acordo com o quadro 3, o índice do indivíduo escolhido é dado pela variável i , e o algoritmo tem parada garantida, visto que aux tende, no limite em que i é igual ao número de elementos da população, ao somatório das avaliações dos indivíduos e s é menor que esta soma (LINDEN, 2008).

Dessa forma, na seleção por roleta a probabilidade de um indivíduo ser selecionado é simplesmente proporcional ao seu valor de aptidão. De acordo com a equação 1, p_i é a probabilidade de seleção de um indivíduo i , f_i é a aptidão do mesmo e N é o tamanho da população (PACHECO, 1999):

$$p_i = \frac{f_i}{\sum_{i=1}^N f_i} \quad (1)$$

A figura 8 apresenta um modelo de cálculo de seleção pela roleta proporcional para quatro indivíduos hipotéticos, e também esboça, graficamente, o resultado desta simulação por meio de uma figura gráfica da roleta. De acordo com o exemplo da figura 8, após a roleta ser girada, irá parar em um ponto aleatório determinando o cromossomo a ser selecionado. O indivíduo “1100” destaca-se por ter uma chance de 58,06% de ser selecionado, em contrapartida o cromossomo “0001” terá apenas 1,61%.

Figura 8 - Exemplo de cálculo para avaliação da população e gráfico da seleção por roleta proporcional



Fonte: Adaptado de Linden (2008)

Conforme ilustrado no exemplo, os cromossomos que possuem um alto valor de aptidão terão maior chance de serem selecionados se comparados aos cromossomos que representam menores frações na roleta ou avaliações inferiores. Esse processo é repetido quantas vezes forem necessárias até que se tenha o número suficiente de pares para a aplicação dos operadores de cruzamento e mutação (LACERDA; CARVALHO; LUDERMIR, 1999; LINDEN, 2008; ROSA; LUZ, 2009).

A expressão que demonstra a intensidade da pressão seletiva I é dada pela equação 2 e representa a razão entre o desvio padrão das aptidões (σ) e a média das mesmas (M) (BLICKLE, 1996):

$$I = \frac{\sigma}{M} \quad (2)$$

Nota-se que, exatamente como na natureza, os mais fortes têm preferência para a reprodução, mas os mais fracos ainda possuem alguma chance. De acordo com Holland (1975) a utilização da seleção proporcional otimiza o *trade-off* entre *exploration* x *exploitation*. No entanto, este método é o que mais apresenta os problemas da existência de super-indivíduos e competição próxima (PACHECO, 1999; LINDEN, 2008).

Outro método de seleção que utiliza a probabilidade de seleção p_i , conforme demonstrado na equação 1, é a roleta por aptidão acumulada. De acordo com Argoud (2007), após ordenar os cromossomos de forma decrescente de aptidão gera-se um número aleatório r entre $[0, \text{Aptidão Acumulada}]$, o cromossomo selecionado será aquele com aptidão acumulada maior que r , e este procedimento é repetido até se completar o número de indivíduos para reprodução. A tabela 3 traz um exemplo de cálculo da seleção por aptidão acumulada para alguns indivíduos fictícios.

Tabela 3 - Exemplo de cálculo da aptidão acumulada

Posição na Ordenação i	Cromossomo (Indivíduo)	Avaliação (Aptidão) f_i	Aptidão Acumulada $\sum_{k=1}^i f_k$	Percentual de Seleção (p_i)	Pedaço da Roleta ($^{\circ}$)
1	11001000	2,00	2,00	11,59	41,74
2	00101011	1,75	3,75	21,74	78,26
3	00011101	1,50	5,25	30,43	109,57
4	10011011	1,00	6,25	36,24	130,43
Total	-	6,25	17,25	100,00	360,00

Fonte: Adaptado de Lacerda, Carvalho e Ludermir (1999)

A expressão da intensidade de seleção I para esse método é dada pela equação 3, onde M é a aptidão média da população, M^* é o valor esperado da aptidão média após a seleção, e σ é o desvio padrão dos valores de aptidão da população antes da seleção (BLICKLE, 1996):

$$I = \frac{M^* - M}{\sigma} \quad (3)$$

3.10 Operador de Cruzamento (*Crossover*)

O operador de *crossover* realiza a manipulação do material genético da população e é aplicado após a seleção dos genitores. Esse processo consiste na troca de segmentos entre

pares de cromossomos por meio de permutação aleatória de elementos do vetor ou quaisquer combinações lineares de dois pais para a reprodução dos genes que serão recombinados para originar os novos indivíduos. Em geral, o resultado são dois cromossomos filhos que virão a formar a população da geração seguinte (PACHECO, 1999; BIEGLER; GROSSMANN, 2004; HAUPT, RANDY; HAUPT, SUE, 2004; ARGOUD, 2007; LINDEN, 2008; ROSA; LUZ, 2009).

A literatura não especifica uma regra que determine a escolha dos pares que irão cruzar, a seleção das duplas pode ser feita de forma aleatória ou por qualquer outro critério escolhido pelo programador que melhor represente o problema que se quer resolver. Definido este critério, o cruzamento pode ocorrer ou não, dependendo de uma taxa de probabilidade p_c que pode ser pré-definida por meio da programação ou determinada pelo usuário à cada execução do AG. Se não ocorrer, os cromossomos filhos serão cópias exatas de seus pais, e se a taxa for baixa a convergência será mais lenta, pois é preciso mais tempo para que haja a melhora do material (soluções) genético (GOLDBERG, 1989; BÄCK; SCHWEFEL, 1993; MICHALEVICZ, 1996; PACHECO, 1999; PREBYS, 1999; YANG, 2005; LINDEN, 2008).

No final do processo de *crossover* a população geralmente permanece do mesmo tamanho da anterior, na qual novos indivíduos irão gerar descendentes diferentes de seus pais, mas com características genéticas de ambos os genitores. O *crossover*, somado ao módulo de seleção, é, então, o responsável pelo fato de um AG não ser comparado a uma busca aleatória. A sua ideia principal é que se consiga propagar as características positivas dos indivíduos selecionados originando indivíduos mais aptos (BÄCK; SCHWEFEL, 1993; PACHECO, 1999; PREBYS, 1999; HAUPT, RANDY; HAUPT, SUE, 2004; ARGOUD, 2007; LINDEN, 2008; ROSA; LUZ, 2009).

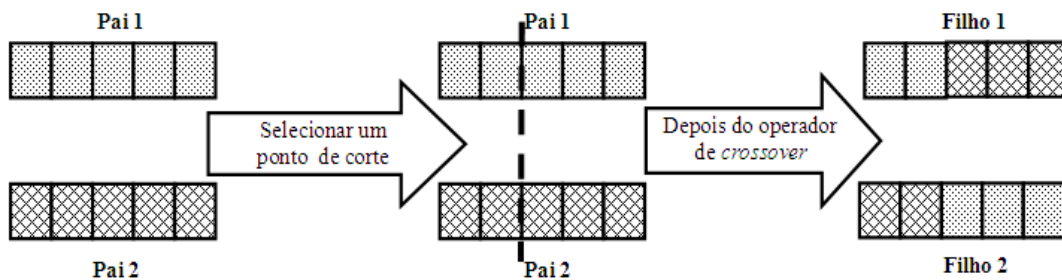
Considerada sua importância, este operador apresenta diversas variações, os principais tipos são o *crossover* de um ponto e o *crossover* uniforme (PACHECO, 1999; VON ZUBEN, 2000; LINDEN, 2008). No entanto, não há nenhuma evidência teórica clara para decidir qual operador de *crossover* é mais adequado ou apresenta um desempenho superior aos demais, cada tipo é particularmente eficiente/ineficiente para uma determinada classe de problemas. A escolha de qual utilizar irá depender da função de aptidão, da codificação utilizada ou outros detalhes do AG (BÄCK; SCHWEFEL, 1993; MITCHELL, 1996; VON ZUBEN, 2000).

3.10.1 *Crossover* de Um Ponto

Depois de selecionados os pais pelo módulo de seleção, o *crossover* de um ponto (*one-point crossover*) corta os genitores em uma posição aleatoriamente escolhida. Um ponto de

corte constitui uma posição entre dois genes de um cromossomo, no qual, cada indivíduo de n genes contém $n-1$ pontos de corte. A partir desse ponto o material genético (*bits* à direita) dos pais são trocados dando origem a dois novos cromossomos descendentes, formados pela combinação das características genéticas dos pais, como demonstrado no diagrama da figura 9 (GOLDBERG, 1989; MICHALEVICZ, 1996; PACHECO, 1999; YANG, 2005; ARGOUD, 2007; LINDEN, 2008; ROSA; LUZ, 2009).

Figura 9 - Diagrama do operador de *crossover* de um ponto



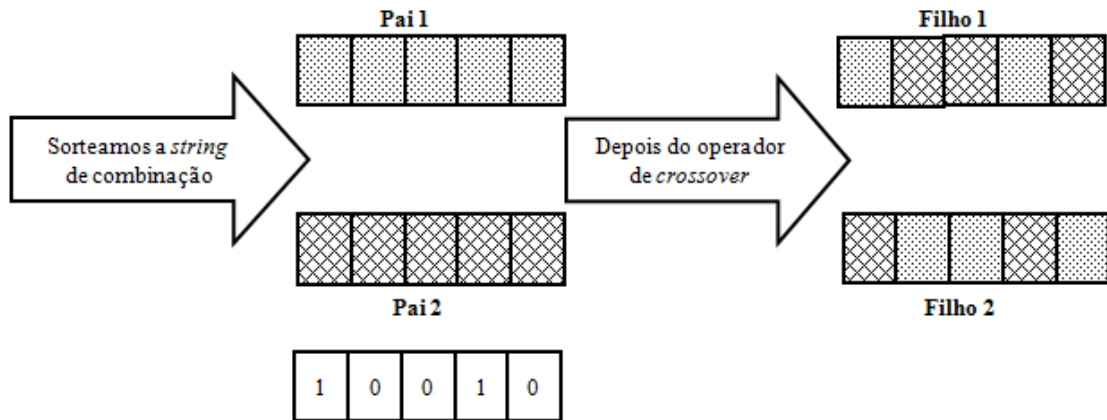
Fonte: Adaptado de Linden (2008)

Conforme observado na figura 9, após a seleção dos pontos de corte os pais são separados em duas partes, que não necessariamente precisam ter o mesmo tamanho, uma à esquerda do ponto de corte e outra à direita. O primeiro filho é composto por meio da concatenação da parte esquerda do primeiro genitor com a parte direita do segundo. O segundo descendente é composto por meio da concatenação das partes que sobraram, ou seja, a metade esquerda do segundo pai com a metade à direita do primeiro (PACHECO, 1999; LINDEN, 2008; ROSA; LUZ, 2009).

3.10.2 *Crossover* Uniforme

O *crossover* uniforme é o tipo de cruzamento que causa maiores mudanças em um AG, além de gerar as mesmas soluções, e ainda novas combinações que os outros tipos não são capazes de criar, ele proporciona um número maior de esquemas que podem ser efetivamente transferidos aos descendentes, como também, tende a conservar esquemas longos com a mesma probabilidade que preserva esquemas de menor comprimento, desde que ambos tenham a mesma ordem. Ou seja, o *crossover* uniforme determina o ponto de corte e realiza o cruzamento em uma posição que preserva os bons esquemas (PACHECO, 1999; LINDEN, 2008).

Este operador utiliza máscaras de cromossomos binários como base para a técnica de cruzamento, que são escolhidos aleatoriamente para designar os *bits* selecionados em cada genitor na criação dos descendentes (PACHECO, 1999; ROSA; LUZ, 2009). A figura 10 demonstra graficamente o funcionamento do *crossover* uniforme.

Figura 10 - Funcionamento do *crossover* uniforme

Fonte: Linden (2008)

A figura 10 mostra a máscara gerada com valores 0 ou 1, sorteados aleatoriamente para cada gene. Se a primeira posição da máscara sorteada possuir o valor 1, então o gene da primeira posição do pai 1 é copiado para o filho 1 e o gene da primeira posição do pai 2 é copiado para o filho 2. Se o valor sorteado for zero, as atribuições dos pais serão invertidas para os filhos (ARGOUD, 2007; LINDEN, 2008).

Devido ao fato de fazer um sorteio para cada posição, o *crossover* uniforme apresenta um poder maior de destruição se comparado aos outros tipos de *crossovers*. Considerando a grande possibilidade de estragar todo e qualquer esquema, sua utilização deve ser, portanto, em ambientes altamente elitistas como na reprodução parcial da população (*steady state*), que garantem a permanência dos melhores indivíduos (PACHECO, 1999; LINDEN, 2008).

3.11 Operador de Mutação

O operador de mutação foi proposto por Holland (1975) para prevenir a baixa diversidade de uma população. Este operador é executado logo após a composição dos filhos gerados pelo processo de cruzamento (*crossover*) e tem por objetivo realizar modificações em determinadas propriedades genéticas dos indivíduos por meio da ação de uma variável aleatória que modifica os elementos de um vetor alterando a posição dos *bits* de uma *string* (PACHECO, 1999; BIEGLER; GROSSMANN, 2004; ARGOUD, 2007; LINDEN, 2008; ROSA; LUZ, 2009).

O operador de mutação requer somente um indivíduo e o processo inicia-se com a escolha de um ponto (*bit*) aleatório deste cromossomo, depois é aplicada uma taxa de probabilidade p_m de troca alterando o valor deste *bit* por um outro *bit* selecionado. Assim como na natureza, a taxa de mutação é aplicada de forma menos frequente que a recombinação. Em geral, a p_m é extremamente baixa, seguindo a inspiração biológica esse valor deve acometer uma pequena parcela da população, que varia da ordem de 0,1% à 1%, e

é informada pelo usuário ou baseada em algum critério previamente definido pelos parâmetros do AG (GOLDBERG, 1989; MITCHELL, 1996; MICHALEWICZ, 1996; PACHECO, 1999; ARGOUD, 2007; LINDEN, 2008; ROSA; LUZ, 2009).

No entanto, não existe nenhuma exigência matemática para o uso das probabilidades, a maioria dos trabalhos publicados na área utiliza um valor predeterminado de 0,5% ou 1% devido a razões históricas dos primeiros artigos terem conseguido bons resultados (LINDEN, 2008). Contudo, na prática pode-se trabalhar com taxas bem maiores, dependendo do objetivo, topologia da superfície de erro e até mesmo se será ou não empregado o elitismo no processo de busca. A eficácia deste operador é, no entanto, restrita a problemas que utilizam a representação genotípica baseada em ordem (GOLDBERG, 1989; MICHALEWICZ, 1996; LINDEN, 2008).

Evidentemente, a operação de mutação mostra-se importante e indispensável para um AG, uma vez que a única maneira de gerar novos valores ou material genético para um gene que não está presente em nenhum outro indivíduo da população. Isso possibilita à população atual obter propriedades genéticas que não existiam ou eram encontradas em baixa porcentagem na população anterior. O operador de mutação permite, assim, uma busca maior no espaço de soluções do problema, de forma a possibilitar que o algoritmo realize a busca em regiões ainda não exploradas no espaço de soluções (DE JONG, 1988; GROSKO; GORSKI; DIAS, 2006; ARGOUD, 2007; ROSA; LUZ, 2009).

Em outras palavras, a mutação substitui aleatoriamente a posição dos genes de um cromossomo introduzindo material genético que pode não estar presente em nenhum outro indivíduo da população de tal modo que se permita o surgimento de diversidade genética, o que faz com que, de certa forma, se evite a convergência prematura. Deste modo, ao contrário do operador de *crossover*, a mutação é o único meio de fazer com que um indivíduo obtenha diferentes valores em alguma posição de sua cadeia de *bits*.

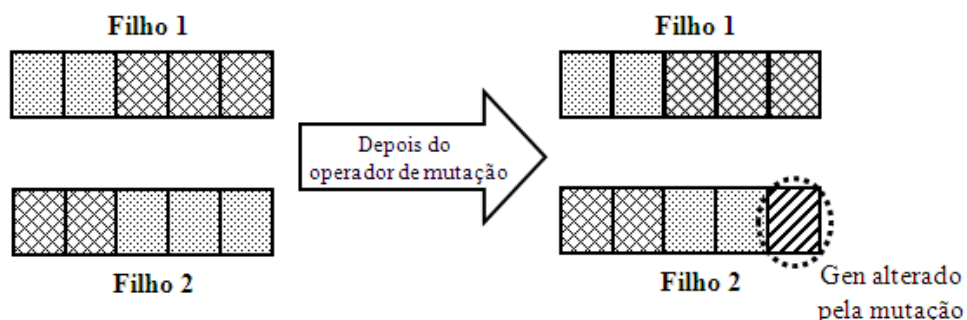
Diante do exposto, é importante citar que a mutação pode ocasionar duas situações críticas que são abordadas pela comunidade científica como sendo o dilema fundamental da mutação. Normalmente, na parte final de qualquer execução do AG se quer pouca mutação para não destruir o alto desempenho dos bons esquemas encontrados, porém, se esta probabilidade for baixa demais, poderá haver uma amostragem global insuficiente no qual ocasionará uma rápida perda de diversidade populacional. Evidentemente, o contrário também se aplica, se a taxa de mutação for alta demais, a fim de superar os efeitos da convergência genética prematura, os indivíduos gerados pouco se assemelharão aos seus pais, o que pode

destruir a informação contida no cromossomo e, portanto, destruir boas soluções. Para resolver este tipo de problema, uma solução razoável é utilizar uma taxa de mutação que varie com o desenrolar da evolução do algoritmo (DE JONG, 1988; ARGOUD, 2007; LINDEN, 2008; ROSA; LUZ, 2009).

Em síntese, a ideia intuitiva por trás do operador de mutação apresentado por Holland (1975) é criar uma variabilidade extra na população, mas sem destruir o progresso já obtido (VON ZUBEN, 2000). A literatura disponível apresenta varias técnicas de se efetuar mutação, dentre elas, as principais são a mutação aleatória e a inversão (GOLDBERG, 1989; OBITKO, 1998; YANG, 2005; LINDEN, 2008; ROSA; LUZ, 2009).

No processo de mutação aleatória, mediante a codificação binária, o operador de mutação simplesmente sorteia um *bit* do cromossomo (0 ou 1) e atua sobre ele alterando-lhe o valor aleatoriamente. Esse processo ocorre de acordo com uma probabilidade pré-definida e é aplicado aos filhos que compõem a população (HOLLAND, 1975; MITCHELL, 1996; LINDEN, 2008). Na figura 11 é apresentado um exemplo da operação de mutação aleatória.

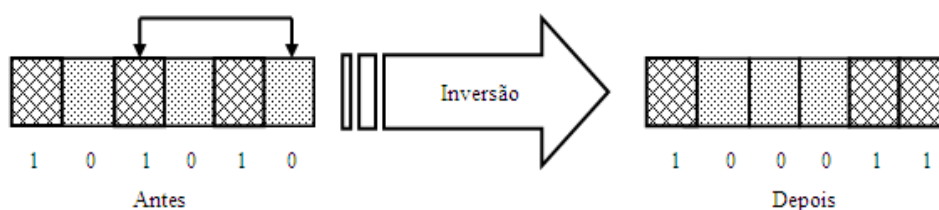
Figura 11 - Operador de mutação aleatória



Fonte: Adaptado de Linden (2008)

Já o operador genético denominado de inversão realiza a troca de posição de n pares de uma *string* ou valores de dois genes, que, da mesma forma que a mutação aleatória, são estocasticamente escolhidos para alteração, e esse processo é também repetido à todos os filhos que compõem a população (GOLDBERG, 1989; MICHALEWICZ, 1996; OBITKO, 1998; PACHECO, 1999; YANG, 2005). A figura 12 apresenta um exemplo da operação de inversão.

Figura 12 - Operador de inversão



Fonte: Autor

Os exemplos apresentados nas figuras 11 e 12 demonstram em detalhes que a mutação é um operador de heurística exploratória que tem por objetivo repor ou acrescentar um novo material genético inexistente, de forma a criar ou aumentar a diversidade na atual população, ao contrário do operador de *crossover* (PACHECO, 1999; GROSKO; GORSKI; DIAS, 2006; ARGOUD, 2007).

3.12 Geração da Nova População (Atualização)

O mecanismo de se criar novas populações, troca da antiga por uma nova, é chamado de geração ou atualização. Esse processo consiste em aplicar técnicas de substituição de populações (cromossomos) para a próxima geração, sendo que os pais têm que ser substituídos conforme os filhos vão nascendo de forma a simular um ambiente de recursos limitados. Isso é feito incorporando os cromossomos filhos à população, gerados pelo cruzamento dos indivíduos selecionados da população anterior (LACERDA; CARVALHO; LUDERMIR, 1999; PACHECO, 1999; YANG, 2005; ROSA; LUZ, 2009).

O módulo de população é responsável pelo controle e inserção dos descendentes na população. Essa não pode crescer, e então, a cada atuação dos OGs criam-se dois filhos que vão sendo armazenados em um espaço auxiliar (vetor de tamanho constante), até que o número de filhos criados sejam iguais ao tamanho da população. Neste ponto o módulo de população entra em ação, todos os pais são então descartados e os filhos copiados sobre suas posições de memória, indo tornar-se os pais da nova geração (GOLDBERG, 1989; MICHALEWICZ, 1996; LINDEN, 2008).

Existem várias alternativas para a concepção das novas populações que permitem melhor explorar as qualidades da atual geração em prol de um progresso nas próximas proles. A maioria dos métodos utiliza cromossomos de comprimento fixo, ou seja, o tamanho das estruturas de codificação dos indivíduos não se altera, embora haja substancial quantidade de investigação sobre a sequência de comprimento variável. As técnicas geralmente usadas no processo de geração são a troca de toda população com ou sem elitismo (LACERDA; CARVALHO; LUDERMIR, 1999; PACHECO, 1999; YANG, 2005; LINDEN, 2008).

3.12.1 Troca de Toda População (Substituição Geracional)

Na substituição geracional, toda a população da atual geração é substituída por um número igual de descendentes a cada nova geração. Assim, $N/2$ pares são escolhidos para o cruzamento, gerando N filhos que substituem N pais. Dessa forma, a população anterior não

convive com a próxima população, perdendo-se, então, as boas soluções encontradas (PACHECO, 1999; ARGOUD, 2007; LINDEN, 2008; ROSA; LUZ, 2009).

3.12.2 Troca de Toda a População com Elitismo

A seleção elitista ou elitismo foi introduzido por Kenneth De Jong em 1975 e pode ser usada em conjunto com vários outros métodos de seleção. Esta técnica consiste de uma pequena alteração no módulo de população que quase não altera o tempo de processamento, o que garante certa percentagem dos melhores indivíduos na geração seguinte, de forma que o desempenho do AG melhore a cada ciclo (TANOMARU, 1995; MITCHELL, 1996; LINDEN, 2008).

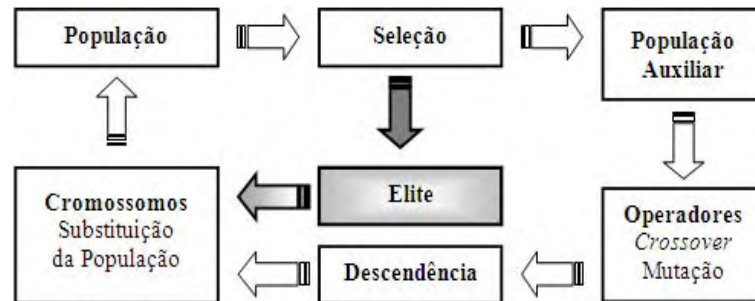
Por ser de simples aplicação e apresentar vantagens significativas para o desempenho dos AG é o método mais utilizado pelos programadores. Esta técnica assegura que os melhores indivíduos passarão para a geração seguinte, já que, transfere os cromossomos responsáveis pelas boas avaliações diretamente para a nova população sem que estes sofram a ação dos operadores de seleção, *crossover* e mutação. Dessa forma, o elitismo proporciona a condição de que o melhor indivíduo da geração t conviva na população da geração $t+1$, o que garante que este indivíduo t será pelo menos igual ao melhor indivíduo das gerações seguintes (ARGOUD, 2007; LINDEN, 2008).

A ideia básica do elitismo, além de aumentar o componente de memória do AG, é impedir ou minimizar o risco de que os n melhores indivíduos (alto grau de adaptação) de cada geração sejam destruídos pela aplicação da mutação ou então não selecionados para o método de *crossover*. Assim, o elitismo obriga o AG a reter um determinado número (percentagem) de melhores (mais aptos) indivíduos da população corrente o que garante que seus materiais genéticos (genomas) sejam preservados (MITCHELL, 1996; PACHECO, 1999; ARGOUD, 2007; LINDEN, 2008).

A sua amostragem pode ser direta, escolhendo-se simplesmente os melhores, ou por sorteio, escolhendo-se os melhores entre os melhores da população. Geralmente essa percentagem é muito pequena para que não ocorra o risco de uma convergência prematura. Após esse processo, os cromossomos são transferidos integralmente para a nova população, o que dará a eles uma nova chance de se reproduzir. Portanto, parte dos indivíduos da população anterior convivem com a população formada por seus filhos (MITCHELL, 1996; PACHECO, 1999; ARGOUD, 2007; LINDEN, 2008; ROSA; LUZ, 2009). O esquema gráfico do elitismo é ilustrado na figura 13, onde é possível visualizar que uma parte dos

cromossomos mais aptos da população corrente será selecionada e constituirá um conjunto (elite) que comporá a nova população dispensando os processos de *crossover* e mutação.

Figura 13 - Esquema gráfico do elitismo



Fonte: Autor

3.13 Parâmetros

Os parâmetros são valores definidos pelo usuário e que têm como função controlar as variáveis do processo evolucionário. Usualmente, por funcionarem bem na maioria dos trabalhos reportados na literatura, utilizam-se valores reais para sua mensuração, embora já existam pesquisas direcionadas ao desenvolvimento de estratégia para alterar a forma de tratar estes valores objetivando melhorar seu desempenho. Os principais parâmetros de um AG são (DE JONG, 1988; MITCHELL, 1996; PACHECO, 1999; ARGOUUD, 2007):

- Tamanho da População: define o total de cromossomos de uma população a cada ciclo;
- Número de Gerações: determina o total de ciclos de evolução ou total de tentativas de solução em um experimento;
- Função Aptidão: define o critério para medir a aptidão de todos os indivíduos da população em relação àquele ambiente específico;
- Forma de Seleção: método para selecionar os indivíduos mais aptos definidos pela função de aptidão e que sofrerão ação dos operadores de *crossover* e mutação;
- Taxa e Tipo de *Crossover*: número de indivíduos selecionados para reprodução (p_c de aplicação) e quantos pontos de cruzamento haverá em cada cromossomo;
- Taxa e Tipo de Mutação: probabilidade (p_m de aplicação) do conteúdo de uma posição/alelo do cromossomo e a forma que cada ponto deve ser alterado;
- Geração da Nova População: determina a forma de troca da antiga população por uma nova por meio da interação, controle e inserção de novos descendentes.

Em geral, o AG é extremamente sensível à introdução ou combinação de técnicas e de parâmetros empregados, visto que, durante o processo de evolução, alguns aspectos que determinam as novas populações estão fortemente relacionados (convergência, aptidão, taxas dos operadores e os parâmetros). E embora possa ser uma escolha empírica e obtida a partir da execução de testes, os parâmetros devem estar em sintonia com o problema (PACHECO, 1999; LINDEN, 2008).

3.14 Critérios de Parada

Os critérios de parada tratam da forma de término do AG e são determinados por meio de parâmetros que controlam a repetição de ciclos definindo o momento em que se deve parar o processo de evolução e adotar o cromossomo com maior aptidão como a melhor solução (PACHECO, 1999; ARGOUD, 2007). As condições de parada devem ser definidas pelo usuário de maneira que melhor se adéquem ao tipo de problema que se quer resolver. De acordo com Argoud (2007), a chegada ao valor ótimo da função de aptidão pode ser adotada como critério de parada quando o mesmo é conhecido, quando não, os critérios comumente usados são (PACHECO, 1999; ARGOUD, 2007; LINDEN, 2008):

- Número de Gerações: o AG termina quando se atingiu um número de gerações especificadas pelo usuário;
- Perda da Diversidade: o AG termina quando uma porcentagem alta da população, geralmente acima de 90%, representa a mesma solução;
- Critério da Convergência: o AG pára quando não há melhoramento no cromossomo de maior aptidão por determinado número de gerações;
- Tempo de Execução: o AG termina quando se atingiu o tempo de execução especificado pelo usuário.

3.15 Verificação de Desempenho

Os AGs são difíceis em termos de análise, pois sua estrutura é probabilística por natureza. Assim sendo, seu comportamento durante as interações não é previsível, já que, devido ao fato de que raramente geram indivíduos iguais, dificilmente repetem resultados de um experimento para outro. Dessa forma, indivíduos muito diferentes podem ter funções de avaliação muito parecidas, portanto, a análise da variabilidade deve ser feita, se possível, com base no genótipo dos indivíduos e não pela função de avaliação (GOLDBERG, 1989; PACHECO, 1999; LINDEN, 2008).

A maioria das provas e teoremas sobre o AG se baseia no seu comportamento ao longo do tempo. Ou seja, o desempenho de um AG é medido pelo resultado médio de vários

experimentos e pelo grau de evolução alcançado durante todo o processo. As principais medidas de desempenho são (GOLDBERG, 1989; PACHECO, 1999; LINDEN, 2008).

- Curva da média da aptidão dos melhores cromossomos a cada ciclo em vários experimentos: apresenta o desempenho médio de um AG e serve como base de ajuste de parâmetros;
- Curva *on-line* da média da avaliação de todos os indivíduos até um determinado instante t em um experimento: mede a velocidade com que o AG consegue produzir boas soluções para o consumo “*on-line*” das soluções;
- Curva *off-line* da média da avaliação do melhores indivíduos até um instante t em um experimento: mede o grau de convergência do AG na criação de soluções mais aptas, geradas *off-line* em relação ao problema.

3.16 Teorema Fundamental dos Esquemas

Nesta última parte, as implicações que a ação conjunta dos operadores de seleção, *crossover* e mutação exercem sobre os esquemas (*schemata*) são analisadas e explicadas em função do Teorema Fundamental do AGs e da Hipótese dos Blocos Construtivos. Esta análise permite visualizar o que acontece, ciclo a ciclo, com os representantes daqueles indivíduos que possuem o padrão H . Considerando uma geração de indivíduos que possuem avaliações positivas, a expressão matemática que agrupa os operadores de seleção, *crossover* e mutação, é dada pela equação 4 (GOLDBERG, 1989; BÄCK; SCHWEFEL, 1993; MITCHELL, 1996; MICHALEWICZ, 1996; PACHECO, 1999):

$$m(H, t+1) \geq m(H, t) * \frac{f(H)}{\langle f \rangle} * \left[1 - p_c * \frac{\delta(H)}{L-1} \right] * [1 - p_m * O(H)] \quad (4)$$

Onde:

$f(H)$ Valor médio da adaptação do esquema H na geração t ;

$m(H, t)$ Número de representações de um esquema H na geração t .

L Número de *bits* em uma string (tamanho de cada indivíduo);

$\langle f \rangle$ Valor médio da avaliação (aptidão) da população na geração t ;

p_m Variável que representa a probabilidade de ocorrência da mutação;

p_c Variável que representa a probabilidade de ocorrência do cruzamento;

$O(H)$ Ordem de um esquema H (quantidade de genes que assumem valores diferentes de *);

$\delta(H)$ Comprimento de um esquema H (distância entre a primeira e a última posição fixa diferente de *).

A interpretação desta equação levou Holland (1975) a descrever o Teorema Fundamental dos Esquemas, no qual afirma que *schemata* curtos e de baixa ordem tendem a se proliferar ou desaparecer nas gerações seguintes, de acordo com a aptidão média. Evidentemente, o operador de seleção não introduz novos esquemas na população, no entanto, combinado ao operador de cruzamento e mutação pode destruir ou criar esquemas (GOLDBERG, 1989; MITCHELL; HOLLAND; FORREST, 1994; MITCHELL, 1996; LINDEN, 2008). Para melhor compreender o teorema apresentado por Holland faz-se necessário discretizar as variáveis da equação 4, assim, é possível analisar o efeito que a ação individual dos operadores de seleção, *crossover* e mutação exercem sobre os esquemas.

O primeiro termo da equação, denominada de equação do crescimento reprodutivo (*reproductive schema growth equation*), calcula o número provável de representantes ou cópias esperadas de um esquema H na próxima geração. A equação 5 reflete o efeito da seleção, demonstrando que este operador tende a favorecer esquemas cujas avaliações estão acima da média da população. Isto significa que o desempenho médio de H em relação à aptidão é maior que o desempenho médio da população. Logo, quando a razão é maior que 1, o número de bons representantes de H em sucessivas gerações $m(H, t+1)$ cresce durante a evolução se proliferando mais frequentemente nas próximas gerações. Em contrapartida, esquemas com aptidão abaixo da média tendem a desaparecer (MICHALEWICZ, 1996; MITCHELL, 1996; PACHECO, 1999):

$$m(H, t+1) = m(H, t) * \frac{f(H)}{\langle f \rangle} \quad (5)$$

Já o operador de *crossover*, quando aplicado em um esquema pode destruí-lo para sempre, exceto quando os pares de indivíduos pais contenham um esquema idêntico a este depois da posição de corte. Portanto, quanto maior o $\delta(H)$, maior o número de pontos de corte dentro de H , logo, maior será a probabilidade de que o *crossover* venha a quebrá-lo, rompendo, possivelmente, suas boas características, ou ainda, retardando a descoberta ou o ajuste de melhores esquemas nas próximas gerações. Seja $p_d(H)$ a probabilidade de destruição de um *schema* H , ou seja, a probabilidade deste esquema não transmitir boas características aos seus descendentes é dado pela equação 6 (MITCHELL; HOLLAND; FORREST, 1994; MICHALEWICZ, 1996; MITCHELL, 1996; PACHECO, 1999):

$$p_d(H) = \frac{\delta(H)}{L-1} \quad (6)$$

Em contrapartida, a probabilidade de um esquema sobreviver (se manter intacto) ao *crossover* é maior se esse esquema for pequeno. Um *schema* $O(H)=1$ e $\delta(H)=0$ nunca poderá

ser destruído, não importa onde o operador de *crossover* faça o corte. Portanto, quanto maior a avaliação do esquema e menor o seu tamanho, mais cópias ele terá na próxima geração. Assim, a probabilidade de sobrevivência de H é calculada conforme a equação 7 (MITCHELL; HOLLAND; FORREST, 1994; MICHALEWICZ, 1996; MITCHELL, 1996; PACHECO, 1999):

$$p_s(H) = 1 - \frac{\delta(H)}{L-1} \quad (7)$$

Considerando que um par de pais genitores pode recuperar parte de um esquema destruído, a possibilidade de transferência de características positivas a um dos seus descendentes é dada pela equação 8 (BÄCK; SCHWEFEL, 1993; MICHALEWICZ, 1996; MITCHELL, 1996; PACHECO, 1999):

$$p_s \geq 1 - p_c * \frac{\delta(H)}{L-1} \quad (8)$$

O efeito da mutação também poderá ser destrutivo caso ocorra em uma posição em que o esquema possua um valor diferente de “*”. Dessa forma, quanto maior $O(H)$, maior será o número de *bits* sorteados para sofrer a mutação, e assim, maiores serão as chances deste *schemata* ser corrompido. A probabilidade p_s de sobrevivência de um esquema após a mutação será $p_s = (1 - p_m)^{O(H)}$. Logo, taxas de mutação com $p_m < 1$ resultam em $p_s \approx 1 - p_m * O(H)$. Isso significa que cromossomos de baixa ordem terão maiores chances de não serem destruídos pela mutação (BÄCK; SCHWEFEL, 1993; MITCHELL; HOLLAND; FORREST, 1994; MICHALEWICZ, 1996; MITCHELL, 1996; PACHECO, 1999).

Nota-se que qualquer ação dos OGs é potencialmente destrutiva, mas se encaixa na categoria de exploração, a busca por indivíduos de avaliação melhor que seus pais. O fato é que, os OGs exercem efeito positivo sobre aqueles esquemas que apresentam comprimento pequeno, constituídos de *bits* que trabalham bem em conjunto e tendem a melhorar a aptidão de um indivíduo, e que são denominados por Goldberg (1989) de “Blocos Construtivos” (BÄCK; SCHWEFEL, 1993; MITCHELL, 1996; MICHALEWICZ, 1996; LINDEN, 2008).

Segundo Goldberg (1989) os Blocos Construtivos (*building blocks*) e suas subsequências de combinações representam os principais critérios de *design* e o mecanismo de trabalho mais importante para a aplicação de um AG canônico a um determinado problema. Goldberg (1989, p. 41) formulou ainda a hipótese dos Blocos Construtivos, no qual descreve que: “[...] um algoritmo genético busca desempenho próximo do ótimo por meio da justaposição de esquemas curtos, de baixa ordem e alto desempenho”.

A hipótese dos Blocos Construtivos pressupõe que um AG funciona bem em esquemas bons e curtos, aqueles que conferem aptidão elevada ao indivíduo e que podem ser recombinados com outros blocos construtivos no decorrer das gerações para formar esquemas maiores que conferem maior aptidão ao mesmo, obtendo assim, indivíduos de alta aptidão (BÄCK; SCHWEFEL, 1993; MITCHELL; HOLLAND; FORREST, 1994).

Basicamente, isto pode ser traduzido como o conceito de que o AG, intrinsecamente, procura, então, criar soluções incrementalmente melhores por meio da aplicação dos OGs em esquemas de baixa ordem e alta função de avaliação. Em especial, quando sua representação interna incentiva a emergência de blocos construtivos que podem ser combinados posteriormente com os outros blocos para melhorar seu desempenho (DE JONG, 1988; LINDEN, 2008)

Dessa forma, uma vez que uma instância de um esquema de ordem superior é descoberta (*schemata* acima da média), sua aptidão elevada permite que este esquema se prospere rapidamente e cresça exponencialmente nas gerações seguintes. Isso se baseia no princípio de que pais de boa avaliação tendem a gerar filhos com avaliações tão boas quanto ou melhores que as suas (BÄCK; SCHWEFEL, 1993; MITCHELL; HOLLAND; FORREST, 1994; MITCHELL, 1996; MICHALEWICZ, 1996; PACHECO, 1999; LINDEN, 2008).

Os mecanismos de seleção natural vão fazer com que os melhores esquemas acabem reproduzindo mais e permanecendo mais tempo na população. Isto justifica a ideia de que o modo ótimo de explorar os possíveis pontos do espaço de busca é alocar aos indivíduos a oportunidade de reprodução em proporção direta à sua aptidão relativa ao resto da população. Assim, os *schemata* propagam os bons esquemas para toda a população durante a sua execução, encontrando os melhores dentre todos (HOLLAND, 1975; BÄCK E SCHWEFEL, 1993).

Em síntese, o teorema dos esquemas formaliza a existência de blocos construtivos em um AG e justificam o seu melhor funcionamento (MITCHELL, 1996). Para Linden (2008, p. 109) a forma final do teorema dos esquemas é o seguinte: “o AG tende a preservar com o decorrer do tempo aqueles esquemas com maior avaliação média e com menores ordem e tamanho, combinando-os como blocos de armar de forma a buscar a melhor solução”.

4 IMPLEMENTAÇÃO DO ALGORITMO GENÉTICO PROPOSTO

Este capítulo aborda a formulação e a metodologia de implementação do AG proposto para o SOF em estudo. Inicialmente, faz-se a análise detalhada do problema e nas seções subsequentes os elementos do AG são expressos por meio de modelagem matemática.

4.1 Representação do Problema

Esta seção detalha a representação do problema de SOF de forma que seja possível identificar as variáveis, estabelecer restrições e especificar os parâmetros que mapeiem o objeto de pesquisa. Para efeito de estudo a tabela 4 apresenta um conjunto de dados hipotéticos que representam o estoque de produto acabado (*PA*) disponível para venda e a carteira de pedidos (*CP*) em determinado momento *t*. Esses dados serão utilizados como base para a implementação do software.

Tabela 4 -Estoque de produtos disponíveis para venda e lista de pedidos em carteira.

ESTOQUE (<i>PA</i>)		CARTEIRA DE PEDIDOS (<i>CP</i>)							
Registro do Produto (Rg.)	Quantidade Estoque (x_i)	Número do Pedido	Registro do Produto (Rg.)	Quantidade Pedida (q_j)	Preço Venda Unitário (pr_j)	Faturamento Total (FT_{CP})	Código do Cliente (C)	Data de Atendimento	Aceita Prod. Parcial
32417	1	100	372300	30	540,00	16.200,00	10	07/03/2012	Sim
38638	2	100	276618	30	464,81	13.944,30	10	07/03/2012	Sim
98152	4	200	1166149	1	6.380,65	6.380,65	20	07/03/2012	Sim
98160	5	300	372300	30	540,00	16.200,00	30	07/03/2012	Sim
98830	1	300	276618	5	464,81	2.324,05	30	07/03/2012	Sim
137539	2	300	1166149	3	6.380,65	19.141,95	30	07/03/2012	Sim
137620	2	300	726422	3	338,00	1.014,00	30	07/03/2012	Sim
154517	1	300	851337	2	420,34	840,68	30	07/03/2012	Sim
186106	2	300	98830	1	1.089,82	1.089,82	30	07/03/2012	Sim
260349	3	400	372300	4	540,00	2.160,00	40	07/03/2012	Não
276618	5	400	726422	6	338,00	2.028,00	40	07/03/2012	Não
372300	30	500	98152	4	740,52	2.962,08	50	07/03/2012	Não
408658	0	500	726422	4	338,00	1.352,00	50	07/03/2012	Não
580282	1	600	98152	1	740,52	740,52	60	08/03/2012	Não
726422	7	600	98160	3	624,50	1.873,50	60	08/03/2012	Não
851337	2	700	98152	3	740,52	2.221,56	70	08/03/2012	Sim
1158608	1	700	98160	2	624,50	1.249,00	70	08/03/2012	Sim
1166149	1	800	186106	2	624,95	1.249,90	80	09/03/2012	Sim
1169441	3	800	408658	1	1.272,16	1.272,16	80	09/03/2012	Sim
		900	98160	4	624,50	2.498,00	90	10/03/2012	Não
		900	137620	2	2.298,63	4.597,25	90	10/03/2012	Não
		1000	1166149	1	6.380,65	6.380,65	100	10/03/2012	Não
Total	73	Totais	-	142	-	107.720,07	-	-	-

Fonte: Autor

Seja *PA* o conjunto de *n* produtos, onde *p* é o registro do produto e *q* a quantidade unitária para cada *p*, o conjunto de objetos x_i é dado por um vetor $X = \{x_1, x_2, \dots, x_n\}$ que representa a quantidade máxima de cada tipo de produto disponível para a venda em determinada data *d*. Já a *CP* é constituída pelo conjunto de *P* pedidos, $CP = \{P_1, P_2, \dots, P_j\}$, onde *j* é o número de pedidos atualizados dia a dia. Considerando que q_i é a quantidade de um

tipo de produto requerido por P_j , então, a soma de q_i , para determinado i , por todos os j de P_j gera a demanda total Q_i da CP a ser comparada com a oferta x_i .

Nota-se que na CP a coluna “Aceita Prod. Parcial” é configurada de acordo com os parâmetros de decisão pré-definidos para cada P_j e especifica se o cliente, denominado de C , aceita ou não faturamentos de quantidades parciais, assim, denomina-se C_{sim} para “Sim” e $C_{n\tilde{a}o}$ quando “Não”. Assim, a representação do AG deve contemplar o conjunto específico de parâmetros de decisão e as respectivas similaridades existentes entre todos os P_j . Logo, q_i deve ser atribuído à P_j pedidos não sobrepostos (disjuntos), onde a quantidade de q_i é definida como a pertinência do i -ésimo produto ao j -ésimo pedido, tal que: $P_1 \cup P_2 \dots P_j \leq x_i$.

4.2 Faturamento Máximo (FM)

O Faturamento Máximo (FM) representa o valor máximo possível de faturamento que se pode obter a partir do conjunto de dados existente no momento t . Assim, o processo de *picking* tende a aperfeiçoar o mecanismo de atribuição, utilizando o FM como a melhor solução a ser obtida pelo SOF. Dessa forma, o FM tem como propósito final servir tanto de verificador da real necessidade de se executar o AG como também de um parâmetro que define o objetivo de busca. Considerando que Q_i é a soma de cada produto q_i solicitado por diferentes P_j que compõem a CP , então se $Q_i > x_i$, ou seja, quando a demanda por determinado produto excede a quantidade em estoque, haverá y_i restrições para o i -ésimo produto. Sendo $w_i = (Q_i - y_i)$ a atribuição parcial de q_i , e se cada produto i tem um preço de venda pr_{ij} que pode variar conforme o pedido j , devido as diversas questões que influenciam a negociação com o cliente, pode-se definir:

$$\text{se } Q_i > x_i \Rightarrow pr_{ij} * w_i = f_{w_i} \text{ ou se } Q_i \leq x_i \Rightarrow pr_{ij} * Q_i = f_{Q_i} \quad (9)$$

Reduzindo-se a expressão anterior é possível reescrever o cálculo do FM , conforme apresentado na equação 10:

$$FM = \sum_i f_{w_i} + \sum_i f_{Q_i} \quad (10)$$

Onde:

FM Valor máximo possível de faturamento no momento t ;

f_{w_i} Faturamento de todos os produtos cuja demanda não será totalmente atendida por motivo de falta de estoque;

f_{Q_i} Faturamento de todos os produtos cujo estoque é mais que suficiente para o atendimento da demanda apresentada.

Tomando a equação 10 como um parâmetro de decisão, o software realizará a verificação da real necessidade de se executar o AG. Adotando a notação FT_{CP} como o lucro total da CP , esse processo é realizado da seguinte forma: se $x_i \geq Q_i$, então $FM = FT_{CP}$, portanto realiza o faturamento de todos os itens constantes na CP . Caso contrário, subentende-se que há y_i restrições para x_i , logo, o usuário será então informado por meio de uma *message box* que execute o AG para que o programa encontre, dentre as possíveis alternativas de faturamento, uma solução que busca a otimização para o SOF.

4.3 Representação do Cromossomo

A forma escolhida para representar as informações do processo de *picking* na estrutura de uma *string* é um cromossomo binário proposto por Zuhri e Omar (2006) em que cada posição do cromossomo é uma solução s representada por uma estrutura de dados do tipo vetor de uns e zeros, onde $s \in \{0,1\}$. Adaptando-se esta representação ao SOF, o mecanismo de atribuição para o processo de *picking* será dado pela variável binária s , no qual o item i pode ser ou não fornecido para o j -ésimo pedido. Assim, para a seleção binária s_{ij} tem-se:

$$s_{ij} = \begin{cases} 1, \text{ então } x_i \in P_j \\ 0, \text{ caso contrário} \end{cases}$$

Portanto, se a seleção binária for $s_{ij} = 1$, haverá processo de *picking*, ou seja, determinada quantidade de um produto x_i será atribuída a um pedido P_j , caso contrário, $s_{ij} = 0$. A utilização desta estrutura torna possível a parametrização das condições do SOF que devem estar implícitas dentro do cromossomo a ser processado, já que, um cromossomo CP se dividirá em P_j genes, no qual o j -ésimo gene representará o j -ésimo pedido, onde q_i é um alelo. Para um melhor entendimento, a representação do cromossomo é ilustrada graficamente na figura 14.

Figura 14 - Representação do cromossomo do processo de *picking*

Produtos - Demanda (Q)																							
q_i	.	.	.	q_{ij}	q_i	.	.	q_{ij}	q_i	q_{ij}	q_i	.	.	q_{ij}	q_i	q_{ij}	q_i	.	.	q_{ij}			
P_1 Pedido				P_2 Pedido				P_3 Pedido				P_4 Pedido				P_5 Pedido				P_j Pedido			
Carteira de Pedidos (CP) - Cromossomo																							

Fonte: Adaptado de Zuhri e Omar (2006)

4.4 Geração da População Inicial

A população inicial será gerada de forma randômica por meio do gerador de números aleatórios $Rand()$ do *Excel*, que gera números aleatórios uniformes com valores entre 0 e 1 para cada *string*, onde a função *Value* arredonda estes números para o inteiro mais próximo.

Assim, caso o número gerado seja menor 0,5 o alelo assumira o valor 0 e caso seja maior ou igual a 0,5 o alelo terá o valor 1. Empregando a notação lógica proposta por Haupt, Randy e Haupt, Sue (2004), uma matriz N representará o tamanho da população, onde cada coluna N_{pop} é um cromossomo e N_{bits} é o número de *bits* na *string*. No AG proposto o usuário é quem define N e há restrições quanto a este número máximo permitido, que é de 200 indivíduos. Para facilitar a operação do *crossover*, já que este operador atua por meio da troca de segmentos entre pares de cromossomos, N sempre deverá ser um número par. Após a execução do operador de inicialização denominar-se-á FO como o total de faturamento obtido em N_{pop} , conforme demonstrado na equação 11:

$$FO = \sum_{i=1} \sum_{j=1} s_{ij} Q_i pr_{Q_{ij}} \quad \forall s_{ij} = \{0,1\} \quad (11)$$

4.5 Função de Aptidão

A função de aptidão irá medir o grau de adequação de cada cromossomo como solução do SOF e será aplicada em todas as gerações de forma que seja possível avaliar as relações de um indivíduo com todo o restante da população. A formulação desta função apresentará certas particularidades, visto que, em virtude da aleatoriedade implícita à inicialização da população, é possível pressupor que em determinadas situações o cálculo do FO contempla a geração de *bits* inválidos. Para que estes eventos não comprometam a solução do problema bem como a evolução do algoritmo a função de aptidão penalizará soluções inviáveis.

Na avaliação, cada indivíduo será penalizado caso não atenda às restrições e aos parâmetros pré-determinados como critérios de decisão impostos pelo SOF. As penalidades serão aplicadas por meio da atribuição de um peso que corresponderá ao valor de faturamento $f_{q_{ij}}$ de cada *bit* infrator, o que afetará diretamente a aptidão do indivíduo, fazendo com que seu processo evolutivo seja alterado. As penalidades que expressam a função de aptidão são definidas a seguir:

1. *Não poderá haver processo de picking de um produto sem saldo no PA:*
Considerando que a ação prescrita para a ocorrência do *picking* é dada por s_{ij} , a penalidade de saldo PE_s que garante que a quantidade de x_i não seja ultrapassada é dada pela equação 12:

$$PE_s = \sum_{j=i}^{N_{bits}} f_{q_{ij}} \text{ se } s_{ij} = 1 | x_i = 0 \rightarrow pr_{ij} * q_{ij} = f_{q_{ij}} \quad (12)$$

2. Não poderá haver processo de picking de quantidades parciais quando não aceito pelo cliente: A ocorrência deste evento implica na penalidade denominada de PE_w e dar-se-á quando $x_i < Q_i$ e C assume-se $C_{n\tilde{a}o}$, assim, o algoritmo irá punir o eventual *bit*. A expressão que define PE_w é dada pela equação 13:

$$PE_w = \sum_{j=i}^{N_{bits}} f_{q_{ij}} \text{ se } w_i \wedge C_{n\tilde{a}o} \rightarrow pr_{q_{ij}} * q_{ji} = f_{q_{ij}} \quad (13)$$

Para se evitar que ocorram avaliações negativas, as penalidades só incidirão uma única vez sobre o valor de cada *bit*, independentemente se o *bit* em questão tenha infringido mais de um critério factível de penalidade, e serão aplicadas satisfazendo a seguinte sequência:

$$PE_s = \begin{cases} f_{q_{ij}} & \text{se } s_{ij} = 1 \mid x_i = 0 \\ 0 & \text{se } s_{ij} = 0 \end{cases}$$

$$PE_w = \begin{cases} f_{q_{ij}} & \text{se } PE_s = 0 \\ f_{q_{ij}} & \text{se } w_i \wedge C_{n\tilde{a}o} \\ 0 & \text{se } s_{ij} = 0 \end{cases} \quad (14)$$

Definidas as condições de penalidade, o AG inicia a verificação da sequência de fornecimento de *picking* por ordem crescente de data de atendimento dos pedidos. Dessa forma, em casos onde $x_i \geq Q_i$, os P_j pedidos referentes à data de atendimento d_1 terão preferência de faturamento em relação à d_2 . Logo, se x_i é atribuído para d_1 não poderá ser atribuído à d_2 , e assim por diante, exceto em casos que w_i seja atribuído a C_{sim} . Portanto, para que seja possível dirigir o processo de busca, de forma que o critério de atendimento ao cliente ocorra por ordem de data de entrega dos pedidos, o AG aplicará uma penalidade de data PE_d , que ao contrário das penalidades anteriores, ocorrerá após a ação do operador de seleção e poderá incidir sobre um *bit* que já tenha sido punido anteriormente.

Evidentemente, a utilização desta regra implica no fato de que, em certas situações, determinados *bits* $s_{ij} = 1$ poderão sofrer duplas penalizações, o que pode vir a acarretar um valor de FO negativo. Neste caso, para se evitar problemas com o operador de seleção, atribui-se R\$ 1,00 ao valor de FO para $FO \leq 1$, conforme equação 15:

$$PE_d = \text{se } x_i \geq Q_i \wedge s_{ij} = 0 \forall P_j^d \mid \text{se } s_{ij} = 1 \forall P_j^{d+1} \rightarrow pr_{q_{ij}}^{P_j^{d+1}} * q_{ji} = f_{q_{ij}} \quad (15)$$

Nota-se que as penalidades caracterizam-se por serem punições corretivas que têm como objetivo permitir o advento de descendentes mais aptos, por meio de uma contínua

evolução das soluções a cada nova geração, em substituição àqueles concebidos de forma indevida. Tomando FO como base de avaliação do cromossomo, a função de aptidão que avalia o processo de *picking* é dada pela equação 16:

$$F_{\text{aptidão}} = \begin{cases} FO - \sum_{j=1}^{N_{\text{bits}}} (PE_s + PE_w + PE_d) \\ \text{se } FO - \sum_{j=1}^{N_{\text{bits}}} (PE_s + PE_w + PE_d) < 1 \rightarrow F_{\text{aptidão}} = \text{R\$ } 1,00 \end{cases} \quad (16)$$

Onde:

$F_{\text{aptidão}}$ Função aptidão (grau de adequação do indivíduo como solução do SOF);

PE_s ; Penalidade de saldo de estoque aplicada a N_{bits} ;

PE_y ; Penalidade de sobreposição de produto aplicada a N_{bits} ;

PE_w ; Penalidade de *picking* parcial de produto aplicada a N_{bits} .

PE_d ; Penalidade de sequenciamento de data aplicada a N_{bits} .

De acordo com a equação 16, os indivíduos mais aptos serão aqueles que obtiverem maior faturamento após a aplicação das penalidades. Assim, os cromossomos de maior $F_{\text{aptidão}}$ terão maior probabilidade de serem selecionados a participar das novas gerações ou ainda serem escolhidos como solução do problema. Em síntese, considerando que o valor de aptidão de um indivíduo varia de 1 a $F_{\text{aptidão}}$, a função aptidão determina quantitativamente se tal cromossomo atende ou não às condições impostas pelo problema e contribui para que a maximização do faturamento seja o resultado otimizado do problema.

4.6 Operador de Seleção

O método de seleção adotado é o da roleta proporcional, que irá operar aleatoriamente de acordo com a probabilidade de seleção obtida por cada *string*, onde N_{pop} é a posição de cada indivíduo equivalente a uma determinada faixa da roleta. Sendo $p_{N_{\text{pop}}}$ a probabilidade de seleção de um indivíduo N_{pop} e N o tamanho da população, o cálculo da seleção é definido pela equação 17:

$$P_{N_{\text{pop}}} = \frac{F_{\text{aptidão}_{N_{\text{pop}}}}}{\sum_{N_{\text{pop}}=1}^N F_{\text{aptidão}_{N_{\text{pop}}}}} \quad (17)$$

Considerando que na equação 17 a avaliação de N_{pop} é proporcional ao seu $F_{\text{aptidão}}$, as soluções que obtiverem faturamento superior às demais terão maior probabilidade de seleção,

pois representarão maiores faixas da roleta. Em relação à quantidade de N_{pop} a ser selecionada para participar do processo de *crossover* e mutação, optou-se por fixar um percentual de 50% ($N/2$) do total de cromossomos representados em N .

4.7 Operador de *Crossover* e Mutação

O operador de *crossover* de um ponto, onde a posição de corte é fixada em 50% de N_{bits} , executará o cruzamento entre os pares de todos os cromossomos selecionados como pais. Já o operador de mutação será aplicado a todos os filhos resultantes do *crossover* pela técnica de troca aleatória dos *bits*, no qual, o usuário terá a opção de informar a probabilidade de troca desejada, que pode variar de 0% até 100%, de acordo com a necessidade de evolução das soluções.

4.8 Geração da Nova População

A troca de toda a população com elitismo é o método adotado para a substituição da população, onde N , após definido pelo usuário, nunca varia de uma geração para a outra. No AG proposto somente o melhor indivíduo de cada ciclo será transferido integralmente para a próxima geração, eliminando-se o risco desta *string* não ser selecionada ou ainda de ser destruída pela ação do OGS. Assim, é possível garantir que o faturamento máximo obtido em cada geração seja no mínimo igual ao faturamento da geração anterior. No entanto, conforme definido na seção 4.5, a $F_{aptidão}$ do melhor indivíduo pode diminuir de uma geração para outra em virtude da aplicação de uma penalidade de data. Assim sendo, considerando N sempre fixo a cada nova geração, onde o elitismo retém o melhor indivíduo da população corrente e a roleta seleciona 50% dos cromossomos que irão participar do processo de *crossover* e mutação, o restante dos indivíduos $N - (N/2+1)$ que completarão a nova população serão gerados aleatoriamente. Esse processo é repetido a todas as gerações do AG.

4.9 Critério de Parada

No software desenvolvido para o problema de SOF o usuário terá a possibilidade de especificar o número de gerações desejadas para o término do processo de evolução e parada do algoritmo. A chegada ao valor ótimo da função de aptidão também é adotada como critério de parada. Dessa forma, independente do número de ciclos, se o cromossomo gerado satisfaz o objetivo requerido encontrando o valor máximo de aptidão (FM) em qualquer uma das gerações, a execução é encerrada, quando não, o AG realizará todos os ciclos até atingir o número de gerações especificadas pelo usuário.

4.10 Metodologia de Implementação

A metodologia de implementação do AG para a solução do SOF segue as fases práticas abordadas no referencial teórico e utilizar-se-á o *Visual Basic for Applications* (VBA) do *Microsoft Office Excel* como ferramenta para a programação computacional do modelo proposto. O software será implementado e submetido a testes por meio de um microcomputador com processador Core I5 de 2.3GHz, 8 GByte de RAM e 750 GByte de HD. O quadro 4 apresenta uma síntese de toda a sequência de etapas da implementação do AG proposto.

Quadro 4 - Sequência de implementação e funcionamento do AG para o SOF

- Etapa 1** - Representação do Problema: Analisar o problema de forma a identificar as variáveis, estabelecer restrições e especificar os parâmetros que mapeiem o objeto de pesquisa.
- Etapa 2** - Verificação: Comparar os dados de entrada (quantidades da *CP* com o *PA*). Se a soma das quantidades de um produto k_i solicitada nos diferentes P_j da *CP* for maior que a quantidade deste mesmo produto no estoque de *PA* (x_i), então executa o AG, se não, realiza faturamento (FIM).
- Etapa 3** - Codificação: Codificar a estrutura genética adequada à parametrização dos critérios de decisão que devem estar implícitas dentro do cromossomo a ser processado.
- Etapa 4** - Inicializar $P(t)$: Seja $P(t)$ a população de cromossomos na geração t gere aleatoriamente uma população $P(t) = \{x^t_1, \dots, x^t_n\}$ durante a iteração t ($t \leftarrow 0$) de n indivíduos.
- Etapa 5** - Avaliar $P(t)$: Calcular o valor de avaliação (aptidão) para avaliar a adequação $f(x)$ de cada cromossomo x^t_i da população durante o processo evolutivo.
- Etapa 6** - Seleção: De acordo com a aptidão, transfira a melhor solução (Elitismo) para a nova população. Para o restante dos indivíduos selecione 50% de $P(t)$ a partir de $P(t + 1)$ dos cromossomos pais em ordem decrescente de aptidão usando a roleta proporcional.
- Etapa 7** - Operadores genéticos: Aplique os OGs a todos os pais selecionados modificando suas características de forma a gerar uma nova descendência de diferentes indivíduos:
- a. *Crossover*: Aplique o *crossover* sobre $P(t)$ cromossomos sorteados para formar a nova geração;
 - b. *Mutação*: Com a p_m aplique a mutação sobre $P(t)$ alterando os cromossomos da nova geração nos *loci* para criar diversidade.
- Etapa 8** - Nova população: Adicione todos os descendentes gerados à nova população corrente. Substitua o restante da população apagando os velhos cromossomos e inserindo novos (iteração $t + 1$) até que a atual esteja completa.
- Etapa 9** - Critério de parada: Se o cromossomo gerado satisfaz o objetivo requerido (*FM*), pare, e retorne a melhor solução da população atual, caso contrário, enquanto o critério de parada não for satisfeito volte ao passo 4 e faça $t \leftarrow t + 1$. Repita o procedimento até que o critério de finalização (número de gerações) seja atingido (FIM ENQUANTO).
- Etapa 10** - Repita: Vá ao passo 4 para novas execuções.

Fonte: Autor

5 EXPERIMENTAÇÃO DO SOFTWARE

Este capítulo apresenta a interface do software e as análises dos resultados obtidos nos experimentos realizados. As simulações ocorreram por meio de testes controlados, nos quais o melhor resultado já era conhecido. As análises foram realizadas em função do resultado médio obtido nos vários experimentos e a avaliação final pelo conjunto dos resultados alcançados.

5.1 Experimentação do Software

Esta seção apresenta os testes de verificação do programa proposto e sua interface com o usuário. Os dados utilizados nos processos de simulação contemplam as ocorrências apresentadas na tabela 4. Será realizado um total de 4 experimentos onde se utilizará uma população inicial de 10 para até no máximo 30 indivíduos, conforme a necessidade de convergência do algoritmo. Todos os experimentos simularão 50 gerações a cada execução, empregando uma taxa de mutação que varia de 5% a 10%. A figura 15 ilustra a interface do software.

Figura 15 - Interface do software

ALGORITMO GENÉTICO APLICADO AO SEQÜENCIAMENTO DE PICKING E FATURAMENTO - SOF

Verifica Picking

Estoque de Prod. Acabado (PA)		Carteira de Pedidos (CP)	
Rg.	Qtid. (x.)	Preço	Total
32417	1	540,00	16.200,00
38638	2	464,81	13.944,30
98152	4	6.380,65	6.380,65
98160	5	540,00	16.200,00
98830	1	464,81	2.524,08
137539	2	6.380,65	19.141,95
137620	2	338,00	1.014,00
154517	1	420,34	840,68
186106	2	1.059,82	1.059,82
260349	3	540,00	2.160,00
276618	5	338,00	2.028,00
372300	30	740,52	2.962,08
408658	0	338,00	1.352,00
580282	1	740,52	740,52
726422	7	624,50	1.873,50
851337	2	740,52	2.221,56
1158608	1	624,50	1.249,00
1166149	1	624,95	1.249,90
1169441	3	1.272,16	1.272,16
		624,50	2.498,00
		2.298,65	4.697,25
		6.380,65	6.380,65

Valor da Carteira de Pedidos (R\$) **107.720,07**

Valor Máximo de Faturamento (R\$) **41.132,93**

Executar AG? Sim Não **VERIFICAR**

Algoritmo Genético

Parâmetros

Tamanho da População:

Número de Gerações:

Taxa de Cruzamento:

Taxa de Mutação:

Crítério de Parada

Número de Gerações

Máximo Faturamento

EXECUTAR **TEMPO** 00:00:03

Módulo População

Método de Geração

Aleatória

Determinística

Troca da População

Troca de Toda População

Troca da População com Elitismo

Número de Indivíduos:

Taxa de Transferência:

Operadores Genéticos

Operador de Seleção

Roda da Proporcional

Roda por Ação Acumulada

Operador de Crossover

Crossover Uniforme

Crossover de Um-Ponto

Operador de Mutação

Mutação por Troca

Inversão por Inversão

Solução do Problema

Melhor Indivíduo:

Pop. do Melhor Indiv.:

Faturamento Obtido (R\$) **36.878,38**

Relatórios

PRODUÇÃO

FATURAMENTO

Estatísticas

Total de Gerações: Total de Mutações:

Total de Indivíduos: Taxa de Seleção:

Total de Elitismo: Taxa de Cruzamento:

Total de Cruzamentos: Taxa de Mutação:

UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
DEPARTAMENTO DE ENGENHARIA DE PRODUÇÃO DE BAURU
PROGRAMA DE PÓS-GRADUAÇÃO "STRICTO SENSU"

 Aluno: Anderson Rogério Faia Pinto
Orientador: Prof. Dr. Antonio Fernando Crepaldi
Software de Pesquisa - Versão: 01 - Bauru: 2012

Fonte: Autor

Para melhorar o entendimento de como o AG funciona, é demonstrado um exemplo de execução com 4 indivíduos para 2 gerações com uma taxa de mutação de 10%. Conforme

ilustrado na figura 15, a primeira operação do programa é o cálculo do FM em que se verifica a necessidade de se utilizar o algoritmo. Dessa forma, o usuário é informado pelo software se precisa ou não executar o AG. Quando necessário, o algoritmo irá operar de acordo com os parâmetros de entrada selecionados pelo usuário no *toolbox* do programa. Esses parâmetros definem: o tamanho da população, o número de gerações e a taxa de mutação. A figura 16 ilustra a criação aleatória de N cromossomos (POPULAÇÃO 1), nos quais se utiliza cores para diferenciar as características dos N_{bits} de cada N_{pop} . Assim, definiu-se a cor preta para o fornecimento de quantidades totais q_i , a verde corresponde às atribuições parciais w_i , e vermelho os *bits* penalizados.

Figura 16 - Exemplo de execução do AG

POPULAÇÃO 1				POPULAÇÃO 2				MELHOR INDIVÍDUO		
Indivíduo 1	Indivíduo 2	Indivíduo 3	Indivíduo 4	Indivíduo 1	Indivíduo 2	Indivíduo 3	Indivíduo 4	Indiv. 1 / Pop. 2	Qtd.	Faturamento
1	1	0	1	1	0	0	0	1	20	R\$ 16.200,00
1	1	1	0	0	0	1	1	0	0	R\$ -
0	1	1	1	1	0	0	0	1	1	R\$ 6.380,66
1	1	1	1	1	0	0	1	1	0	R\$ -
0	0	1	0	0	1	1	1	0	0	R\$ -
0	0	1	1	1	1	1	1	1	0	R\$ -
0	1	1	0	0	0	1	1	0	0	R\$ -
1	1	0	0	0	0	1	1	0	0	R\$ -
0	1	0	0	1	1	1	1	0	0	R\$ -
0	0	0	1	1	0	1	0	1	0	R\$ -
1	0	1	1	1	0	1	1	1	6	R\$ 2.028,00
0	0	0	1	1	1	1	1	1	4	R\$ 2.962,08
1	0	1	1	1	1	1	1	1	1	R\$ 333,00
0	0	0	0	0	1	0	1	0	0	R\$ -
1	0	0	1	1	1	1	1	1	3	R\$ 1.873,50
1	1	0	0	0	0	0	0	0	0	R\$ -
1	1	1	1	1	0	0	0	1	2	R\$ 1.219,00
1	0	0	1	1	1	1	0	0	2	R\$ 1.249,90
1	0	1	0	0	0	1	0	0	0	R\$ -
1	1	1	1	1	0	0	0	0	0	R\$ -
0	0	0	1	1	0	0	0	0	1	R\$ 4.897,24
1	0	1	1	1	0	0	0	1	0	R\$ -
R\$ 14.177,49	R\$ 2.899,18	R\$ 839,45	R\$ 28.182,94	R\$ 28.182,94	R\$ 1,00	R\$ 1,00	R\$ 264,92	R\$ 36.878,38	R\$ 21,00	R\$ 36.878,38
0,307544015	0,062890219	0,018209699	0,611356067	0,990617880	0,000035150	0,000035150	0,009311821			
X	X		Melhor	Melhor						

CROSSOVER E MUTAÇÃO			
Indivíduo 1	Indivíduo 2	Indivíduo 2	Indivíduo 1
0	0	0	0
0	1	1	0
0	0	0	0
0	0	1	0
1	1	1	1
1	1	1	0
0	1	1	0
0	1	1	0
1	1	1	1
0	1	1	0
0	1	1	0
1	1	1	1
1	1	1	1
1	0	1	1
1	1	1	1
0	0	0	0
0	0	0	0
1	0	0	0
0	1	0	1
0	0	0	1
0	0	0	0
0	1	1	0

Fonte: Autor

Conforme demonstrado, cada *string* N_{pop} criada é submetida à ação da função aptidão, que penalizará soluções impróprias fazendo com que seu FO seja reduzido. Por conseguinte, o operador de seleção é que selecionará 50% do total da população avaliada pelo $F_{aptdão}$. Nota-se

que o melhor indivíduo, neste caso o de número 4, não participará da seleção, já que, será copiado direto para a próxima geração (POPULAÇÃO 2) por meio do elitismo, onde se tornará o primeiro indivíduo da próxima geração. Logo, o percentual de transferência segundo a regra do elitismo é dado por $1/N = 50\%$. O *crossover* de um ponto irá, então, atuar em 100% dos pais, tomados aos pares pela seleção, realizando 2 cruzamentos e gerando 2 novos indivíduos (amarelo), assim, a cada rodada 50% da população é modificada pela ação deste operador. Depois de finalizado o *crossover*, o operador de mutação irá operar sobre todos os filhos criados, alterando aleatoriamente n bits (azul) de cada N_{pop} , de acordo com a p_m definida. Após finalizar a mutação, todos os descendentes gerados são transferidos para a nova população (POPULAÇÃO 2) indo se juntar ao melhor indivíduo resultante do elitismo. O restante de N_{pop} necessários para se completar a nova população $N-(N/2+1)$ será gerada aleatoriamente, sendo assim, toda a antiga população será substituída pela que acaba de ser criada. Feita esta atualização, o $F_{aptidão}$ avaliará cada N_{pop} de forma a verificar se a primeira condição de parada (FM) foi alcançada. Nota-se que, o fato de já haver encerrado o número de gerações especificadas faz com que o N_{pop} de maior $F_{aptidão}$ seja, então, adotado como solução do problema.

Evidentemente, existe a opção de aumentar tanto a população quanto o número de gerações nas próximas execuções. Neste caso, resultados que não sejam o valor ótimo de aptidão serão desprezados e o processo recomeça por meio de novos e sucessivos ciclos a serem testados. Esse processo é repetido até se chegar a melhor alternativa ou ao valor do limite máximo de gerações. Ao final de todas as gerações necessárias às condições de parada, o usuário pode acompanhar, por meio da figura 15, a solução do problema, onde é demonstrado o melhor indivíduo, a população ao qual faz parte e o total de faturamento obtido. No item Estatísticas, da figura 15, é possível visualizar os dados referentes a cada execução, que compreendem as seguintes informações: o total de gerações realizadas, o total de indivíduos da população, a taxa de seleção utilizada, a taxa e o total dos melhores indivíduos transferidos para a próxima geração por meio do elitismo e o número de vezes e a taxa em que os operadores de *crossover* e mutação foram aplicados.

5.2 Relatórios Disponíveis

O software desenvolvido possibilita, fundamentado na solução encontrada, gerar dois tipos de relatórios que permitem uma melhor gestão e facilita o processo de tomada de decisões. O primeiro relatório consiste no resultado do problema, ou seja, no processo de

picking, o segundo informa os itens solicitados pela *CP*, mas que não foram faturados por falta de estoque. A figura 17 ilustra o processo de *picking* e a figura 18 demonstra a lista de pendências a produzir.

Figura 17 - Lista do processo de *picking*

Pedido	Cliente	Data	RG	QTDE	Preço Unit.	VR. Fat.
100	10	05.11.2011	372300	30	540,00	16.200,00
200	20	05.11.2011	1166149	1	6.380,65	6.380,65
400	40	05.11.2011	726422	6	338,00	2.028,00
500	50	05.11.2011	98152	4	740,52	2.962,08
500	50	05.11.2011	726422	1	338,00	338,00
600	60	06.11.2011	98160	3	624,50	1.873,50
700	70	06.11.2011	98160	2	624,50	1.249,00
800	80	07.11.2011	186106	2	624,95	1.249,90
900	90	09.11.2011	137620	2	2.298,63	4.597,25

Fonte: Autor

Figura 18: Lista de pendências a produzir

RG	Qtde. Produzir
372300	34,000
276618	30,000
1166149	4,000
372300	4,000
1166149	3,000
726422	6,000
726422	3,000
98152	4,000
98160	4,000
98160	1,000
408658	1,000

Fonte: Autor

Dessa forma, uma cópia do processo de *picking* poderá ser enviada ao departamento logístico para que os operadores realizem todo o trâmite de separação física e expedição e a outra é destinada à área responsável pela emissão das notas fiscais de fatura. Já a lista de pendências a produzir será direcionada à área de produção que proporcionará uma visão geral, em tempo real, da necessidade de demanda. Assim, é possível aperfeiçoar o cálculo das necessidades de produção e o alinhamento entre data de fabricação e a entrega do produto ao cliente, de forma mais rápida e precisa, o que proporciona à gestão de demanda condições de flexibilidade e adaptação às mudanças.

5.3 Análise de Desempenho

As análises de desempenho serão realizadas em relação aos resultados médios obtidos nos vários experimentos. Além de estimar o grau de evolução médio alcançado durante todas as gerações para cada execução, avalia-se também o número de vezes em que se atingiu o valor máximo da função de aptidão. Em todos os experimentos são demonstrados os gráficos de convergência referente aos valores de aptidão do melhor indivíduo de cada geração para todas as execuções realizadas. A tabela 5 apresenta os resultados obtidos no primeiro experimento, e a figura 19 esboça graficamente o grau de convergência do algoritmo.

Tabela 5 - Resultados estatísticos da primeira experimentação

RESULTADOS ESTATÍSTICOS DA PRIMEIRA EXPERIMENTAÇÃO											
Seqüência de Execuções	Quantidade Indiv.(N)	Número Gerações	Indivíduos Selecionados	Indivíduos Transferidos	Total de Cruzamentos	Taxa de Mutação	Total de Mutações	População Melhor Indiv.	Melhor Indivíduo	Total ($F_{aptidão}$) Faturamento	Tempo Minutos
1	10	50	250	49	245	5%	490	39	3	40.043,11	00:03:34
2	10	50	250	49	245	5%	490	24	10	40.043,11	00:03:48
3	10	50	250	49	245	5%	490	44	3	39.042,35	00:03:31
4	10	50	250	49	245	5%	490	38	8	40.043,11	00:03:55
Média	10	50	250	49	245	5%	490	-	-	39.792,92	00:03:42

Fonte: Autor

Figura 19 - Gráficos de convergência da primeira experimentação



Fonte: Autor

Conforme ilustrado na figura 19, o número de indivíduos utilizado é insuficiente para a convergência do algoritmo. É possível constatar que, até certo número de gerações o algoritmo obteve um nível de evolução esperado, no entanto, a partir de um determinado ponto não há melhora nos resultados. Nota-se ainda que, em virtude de se usar o elitismo, o melhor resultado de cada geração se repete na geração seguinte quando não há melhora nas

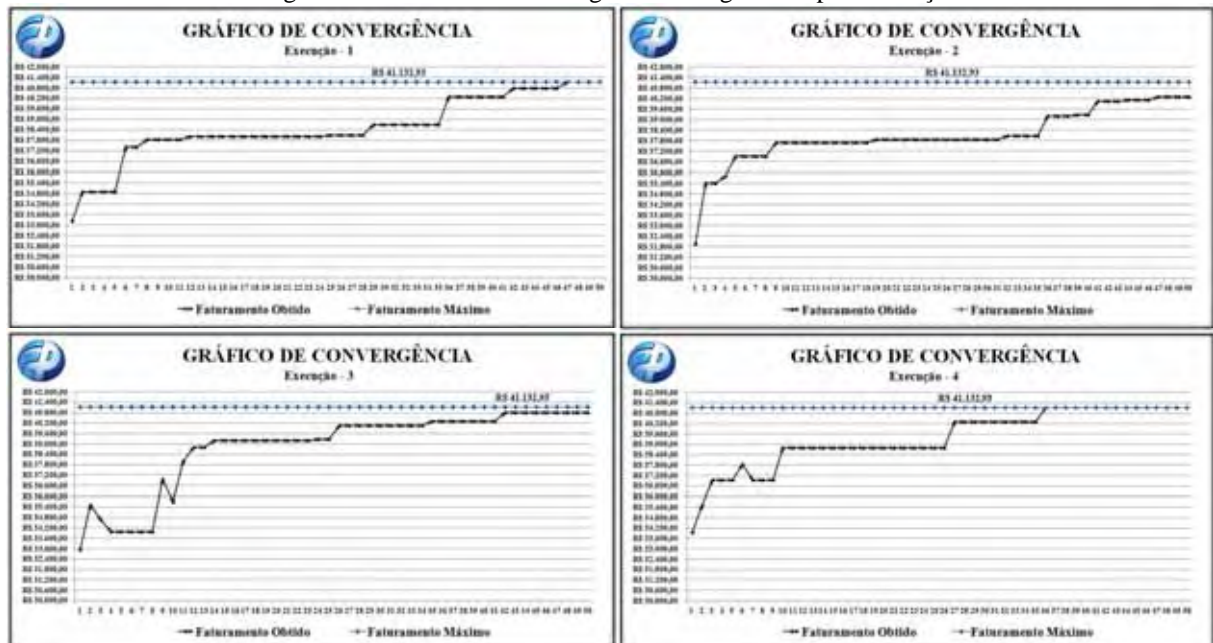
soluções. Entretanto, é possível visualizar que de uma geração para outra podem existir situações de regressão no grau de evolução do faturamento. Isto ocorre devido à ação da penalidade de data, no qual aplica punições aos melhores indivíduos que desrespeitam as condições de sequenciamento de atendimento impostas pelo problema. Portanto, é comum a ocorrência destes eventos que também podem ser observados em outras figuras gráficas de todos os quatro experimentos realizados. A fim de buscar o dimensionamento do tamanho ideal onde se verifica o melhor desempenho do AG, no próximo experimento, demonstrado na tabela 6 e ilustrado na figura 20, utilizar-se-á uma quantidade de 16 indivíduos para uma taxa de mutação de 8%.

Tabela 6 - Resultados estatísticos da segunda experimentação

RESULTADOS ESTATÍSTICOS DA SEGUNDA EXPERIMENTAÇÃO											
Seqüência de Execuções	Quantidade Indiv.(N)	Número Gerações	Indivíduos Selecionados	Indivíduos Transferidos	Total de Cruzamentos	Taxa de Mutação	Total de Mutações	População Melhor Indiv.	Melhor Indivíduo	Total ($F_{aptidão}$) Faturamento	Tempo Minutos
1	16	46	368	45	368	8%	736	46	16	41.132,93	00:03:37
2	16	50	400	49	392	8%	784	44	7	40.118,93	00:07:50
3	16	50	400	49	392	8%	784	42	6	40.043,11	00:13:54
4	16	35	280	34	280	8%	560	35	7	41.132,93	00:12:32
Média	16	45,25	362	44,25	358	8%	716	-	-	40.606,98	00:09:28

Fonte: Autor

Figura 20 - Gráficos de convergência da segunda experimentação



Fonte: Autor

No segundo conjunto de simulações observa-se que com um pequeno aumento no número de indivíduos houve uma melhora de desempenho do algoritmo. A figura 20 permite verificar que o AG encontrou o valor de máximo faturamento na primeira e quarta execução. Logo, o resultado médio obtido não é satisfatório, visto que, o algoritmo obteve um resultado

desejável em apenas 50% das execuções, podendo ainda haver o risco de o algoritmo ficar restrito a uma solução local para outras execuções. Evidentemente, um aumento no tamanho da população aumenta a probabilidade de encontrar melhores resultados, porém, tem-se também uma ampliação considerável no tempo de processamento computacional que em média triplicou para 09:28 minutos. Embora não seja o caso, há de se considerar que em situações reais de SOF, onde os gestores necessitam tomar decisões de forma ágil, a demora em conseguir uma resposta, em virtude de uma busca na qual demandaria um exaustivo esforço computacional, inviabilizaria o software proposto, fazendo com que o AG deixe de ser um método de solução viável ao problema em estudo. Para o terceiro experimento utilizar-se-á uma quantidade de 20 indivíduos para uma taxa de mutação de 10%.

Tabela 7 - Resultados estatísticos da terceira experimentação

RESULTADOS ESTATÍSTICOS DA TERCEIRA EXPERIMENTAÇÃO											
Seqüência de Execuções	Quantidade Indiv. (N)	Número Gerações	Indivíduos Selecionados	Indivíduos Transferidos	Total de Cruzamentos	Taxa de Mutação	Total de Mutações	População Melhor Indiv.	Melhor Indivíduo	Total ($F_{\text{aptidão}}$) Faturamento	Tempo Minutos
1	20	28	280	27	280	10%	1.120	28	10	41.132,93	00:02:19
2	20	44	440	43	440	10%	1.760	44	13	41.132,93	00:04:16
3	20	40	400	39	400	10%	1.600	40	10	41.132,93	00:03:13
4	20	33	330	32	330	10%	1.320	33	20	41.132,93	00:03:12
Média	20	36,25	363	35,25	362,50	10%	1.450	-	-	41.132,93	00:03:15

Fonte: Autor

Figura 21 - Gráficos de convergência da terceira experimentação



Fonte: Autor

No terceiro experimento é possível visualizar que com um incremento no número de indivíduos na população, o AG encontrou o máximo faturamento em todas as execuções. Este

cenário se reflete claramente nos gráficos da figura 21 onde se torna visível que, embora a mutação contribua para que haja mudança nas características genéticas dos cromossomos, a variedade populacional é essencialmente obtida pela variação no número de indivíduos utilizados a cada execução. Nota-se que a média do valor de faturamento do melhor indivíduo na primeira geração aumenta consideravelmente quando se aumenta o tamanho da população, o que torna mais fácil para o AG encontrar a solução para o máximo faturamento possível. É importante explicar que, embora na terceira execução o AG tenha encontrado uma combinação ótima de faturamento na geração de número 36, esta não respeita o critério de data, dessa forma, o resultado é desprezado e o processo continua até a geração 40 onde se origina um indivíduo apto ao SOF. No quarto experimento as simulações são realizadas com 30 indivíduos permanecendo com uma taxa de mutação de 10%.

Tabela 8 - Resultados estatísticos da quarta experimentação

RESULTADOS ESTATÍSTICOS DA TERCEIRA EXPERIMENTAÇÃO											
Sequência de Execuções	Quantidade Indiv. (N)	Número Gerações	Indivíduos Selecionados	Indivíduos Transferidos	Total de Cruzamentos	Taxa de Mutação	Total de Mutações	População Melhor Indiv.	Melhor Indivíduo	Total (F _{aptidão}) Faturamento	Tempo Minutos
1	30	5	75	4	75	10%	300	5	8	41.132,93	00:07:54
2	30	6	90	5	90	10%	360	6	26	41.132,93	00:05:34
3	30	11	165	10	165	10%	660	11	24	41.132,93	00:07:18
4	30	13	195	12	195	10%	780	12	12	41.132,93	00:03:40
Média	30	8,75	131	7,75	131,25	10%	525	-	-	41.132,93	00:06:07

Fonte: Autor

Figura 22 - Gráficos de convergência da quarta experimentação



Fonte: Autor

Analisando os resultados das simulações para 30 indivíduos, ao final das gerações verifica-se que a frequência de convergência do algoritmo para o maior valor de aptidão foi

rápida, 9 gerações, para um tempo médio de 06:07 minutos. Isso demonstra que os parâmetros bem como o total de indivíduos utilizados se mostraram suficientes para representar todo o conjunto de dados empregados nas simulações. Portanto, é a partir deste ambiente que se verifica a melhora no desempenho do AG, onde tamanho da população proporcionou um número de pontos do espaço de busca capaz de produzir diversidade suficiente para capturar todas as informações do problema e permitir a convergência da população.

A análise final referente ao conjunto dos resultados alcançados proporciona um amplo entendimento do comportamento do AG proposto no que diz respeito ao tamanho ideal da população, a parametrização dos OGs e os limites da complexidade das análises. De forma geral, o FM reflete o total de faturamento em função das restrições de PA enquanto que o ótimo global da $F_{\text{aptidão}}$ traduz o valor máximo de faturamento que se pode obter face às restrições e parâmetros de decisão pré-determinados. O fato é que a chegada ao valor ótimo da $F_{\text{aptidão}}$ esta condicionada ao dimensionamento ideal da população. Dessa forma, se a cada ciclo de evolução, o tamanho da população for pequeno, o efeito do cruzamento e da mutação também será pequeno e não haverá variedade genética para que o AG encontre uma solução satisfatória. Em relação aos OGs, se aplicado somente o operador de *crossover* a população provavelmente se estagnaria em torno de uma solução local. Já a taxa de mutação se for baixa, as mudanças ocorrerão de forma lenta, se for alta, os traços desejados não serão mantidos. Em síntese, quanto maior o tamanho da CP e a complexidade dos parâmetros de decisão utilizados maior será a quantidade de indivíduos necessários ao bom comportamento do AG, logo, maior será o tempo de processamento computacional. Particularmente, o microcomputador usado nas experimentações realizadas apresentou desempenho satisfatório em relação aos tempos de execução. No entanto, estes tempos podem variar de acordo com a capacidade de processamento dos recursos computacionais utilizados, o que pode vir a causar variações nos tempos de execução para um mesmo conjunto de dados.

6 CONCLUSÃO

Esta dissertação apresentou a formulação de um software para a automatização do processo de *picking* de forma que o resultado apresentasse uma combinação otimizada para o SOF. A pesquisa empregou o AG como método de busca de possíveis soluções, utilizando o *Visual Basic for Applications (VBA)* como ferramenta de implementação computacional do sistema proposto. O software foi desenvolvido sobre a plataforma de um conjunto de dados estáticos representados pelo *PA* e a *CP* em determinado momento t , configurados para atender as restrições e parâmetros de decisão pré-definidos de forma a maximizar o faturamento.

Nos experimentos realizados os resultados obtidos se demonstraram satisfatórios e coerentes quanto ao escopo da pesquisa. Além de fácil operacionalização e razoável *overhead* computacional, ou seja, um razoável custo de programação e tempo de execução, o software possui um potencial de otimização capaz de automatizar e prover agilidade e flexibilidade ao processo de *picking*, aperfeiçoando as tarefas de planejamento e tomada de decisões de SOF. Embora não foram realizadas pesquisas sobre outras ferramentas de SOF que fosse capaz de permitir uma comparação entre o sistema apresentado e outros métodos existentes, conclui-se que, o AG proposto é uma opção viável às empresas que se deparam com o tipo de problema em estudo.

O software se auto-organiza por meio de diversos parâmetros de decisão de forma a aperfeiçoar e prover eficiência ao processo de *picking* e a tomada de decisão de faturamento. A partir do resultado obtido as variáveis de custos de diferentes departamentos relacionados ao planejamento e aspectos operacionais do SOF são minimizadas, o fluxo de informações logísticas também se torna mais rápido, já que, a listagem de *picking* permite agilidade na identificação e separação dos produtos. Os relatórios fornecidos pelo software proporcionam ao departamento de produção uma visão geral em tempo real da *CP*, o que proporciona mais rapidez e precisão à programação do processo produtivo. De forma geral, o tamanho da população e a taxa de mutação utilizada proporcionou um número de pontos do espaço de busca capaz de produzir diversidade genética suficiente para capturar todas as informações do problema e permitir a convergência da população.

Em síntese, a solução do SOF minimiza problemas reais e reduz custos, além de proporcionar condições de flexibilidade e adaptação às mudanças, o que de certa forma corrobora para a eficiência de uma série de processos que envolvem tempo e negociação. No entanto, é importante frisar que o trabalho desenvolvido não trata a margem de lucro perante o resultado otimizado do SOF. Portanto, o gestor deve se atentar ao fato de que as soluções são obtidas em função de parâmetros pré-definidos e que estes devem estar em consonância

com as atividades operacionais e a estratégia organizacional de forma que os objetivos corporativos sejam alcançados.

Por fim, esta pesquisa apresentou uma ferramenta de apoio gerencial que inspirada na inteligência artificial e na evolução natural e biológica é capaz de propor a melhor decisão de faturamento. Evidentemente, abre-se um pressuposto para que diferentes estudos venham a contribuir com novas ferramentas para problemas desta natureza. Algumas sugestões para futuras pesquisas que complementariam o software proposto são:

- É recomendável nas próximas pesquisas tentar aumentar o desempenho do AG por outros parâmetros, operadores ou representações;
- Realizar um estudo com um conjunto de dados relativamente grande e obtidos diretamente de alguma organização, de forma que fosse possível comparar mais efetivamente os processos existentes com aquele resultante da aplicação do AG para o SOF;
- Uma pesquisa relevante seria a avaliação do SOF para a margem de lucro e aspectos operacionais dinâmicos, como por exemplo, *lead time* de produção e as quantidades de estoque em processo, aumentando assim o grau de probabilidade para adaptação em resposta ao ambiente sempre variável;
- Incluir ao programa um critério para minimizar a distância percorrida na separação física dos produtos apontados pelo processo de *picking* para o roteamento de múltiplos caixeiros viajantes em grandes armazéns.

REFERÊNCIAS

- ARGOUD, A. R. T. T. **Procedimento para projeto de arranjo físico modular em manufatura através de algoritmo genético de agrupamento**. 2007. 328f. Tese (Doutorado em Engenharia de Produção) - Programa de Pós-Graduação em Engenharia de Produção - Escola de Engenharia de São Carlos - Universidade de São Paulo, USP, São Carlos, SP, 2007.
- ALMEIDA, F. W. C.; VIANA, G. V. R.; THOMAZ, A. C. F. Algoritmo Genético Para a Solução de Problemas de Programação Linear Inteira. In: ENCONTRO REGIONAL DE PESQUISA OPERACIONAL DO NORDESTE, 3. 2009, Fortaleza. **Anais...** Fortaleza: UECE/SOBRAPO, 2009. 1 CD-ROM.
- AYALA, F. J. Darwin's explanation of design: From natural theology to natural selection. **Infection, Genetics and Evolution** v. 10, n. 6, p. 840-843, aug. 2010.
- BÄCK, T.; SCHWEFEL, H. P. An Overview of evolutionary algorithms for parameter optimization. **Evolutionary Computation**, v. 1, n. 1, p. 1-23, 1993.
- BERTO, R. M. V. S.; NAKANO, D. N. Metodologia da pesquisa e a Engenharia de Produção. In: ENCONTRO NACIONAL DE ENGENHARIA DE PRODUÇÃO, 18. 1998, Niterói. **Anais...** Niterói: UFF/ABEPRO, 1998. 1 CD-ROM.
- BERTO, R. M. V. S.; NAKANO, D. N. A produção científica nos anais do encontro nacional de engenharia de produção: um levantamento dos métodos e tipos de pesquisa. In: ENCONTRO NACIONAL DE ENGENHARIA DE PRODUÇÃO, 19. 1999, Rio de Janeiro. **Anais...** Rio de Janeiro: UFRJ/ABEPRO, 1999. 1 CD-ROM.
- BERTRAND, J. W. M.; FRANSOO, J. C. Modelling and Simulation: Operations Management Research Methodologies Using Quantitative Modeling. **International Journal of Operations & Production Management**, v. 22, n. 02, p. 241-264, 2002.
- BIEGLER, L. T.; GROSSMANN, I. E. Retrospective on optimization. **Computers and Chemical Engineering**, v. 28, p. 1169-1192, jul. 2004.
- BLICKLE, T. **Theory of evolutionary algorithms and application to system synthesis**. 1996, 272f. Ph.D. dissertation (Doctor Of Technical Sciences), Swiss Federal Institute of Technology, ETH, Zurich, Switzerland, n°. 11894, 1996.
- BOWLER, P. J. Evolution, History of. **International Encyclopedia of the Social & Behavioral Sciences**, p. 4986-4992, 2004.
- CARVALHO, A. P. L. F. de. **Tutorial introdutório sobre Algoritmos Genéticos**. In: Instituto de Ciências Matemáticas e de Computação, ICMC, Universidade de São Paulo. São Carlos: USP, 2004. Disponível em: <<http://www.icmc.usp.br/~andre/research/genetic/index.htm>> Acesso em: 06 fev. 2011.
- DIAS, J. S.; BARRETO, J. M. **Algoritmo genético: inspiração biológica na solução de problemas - uma introdução**. Revista Marítima Brasileira - Suplemento Especial, Pesquisa Naval, n° 11, p. 105-128, 1998.
- DE JONG, K. Learning with genetic algorithms: An overview. **Machine Learning**, v. 3, n. 2-3, p. 121-138, may, 1988.

- GAVIRA, M. O. **Simulação computacional como uma ferramenta de aquisição de conhecimento**. 2003. 146p. Dissertação (Mestrado em Engenharia de Produção) - Programa de Pós-Graduação em Engenharia de Produção - Escola de Engenharia de São Carlos - Universidade de São Paulo, USP, São Carlos, SP, 2003.
- GEN, M.; CHENG, R. **Genetic algorithms & engineering design**. New York: John Wiley & Sons, Inc., 1997.
- GHISELIN, M. T. Darwin and the evolutionary foundations of society **Journal of Economic Behavior & Organization**, v. 71, p. 4-9, July, 2009.
- GIL, A. C. **Como elaborar projetos de pesquisa**. 3. ed. São Paulo: Atlas, 1991. 159 p.
- GOLDBERG, D. E. **Genetic Algorithms in Search, Optimization, and Machine Learning**. 2. ed. USA: Addison-Wesley Co. Massachusetts, 1989.
- GROSKO, A. P.; GORSKI, J. R.; DIAS, J. S. Algoritmo Genético: Revisão Histórica e Exemplificação. In: CONGRESSO BRASILEIRO DE INFORMÁTICA EM SAÚDE, 10, 2006, Florianópolis. **Anais...** Florianópolis: CBIS, 2006. 1 CD-ROM.
- HAUPT, Randy, L.; HAUPT, Sue, E. **Practical Genetic Algorithms**, 2 ed. New York: John Wiley & Sons, Inc. 2004.
- HOLLAND, J. H. **“Adaptation in Natural and Artificial Systems”** University of Michigan Press Ann Arbor, 1975.
- LACERDA, E. G. M.; CARVALHO, A. C. P. L. F.; LUDERMIR, T. B. Um tutorial sobre algoritmos genéticos. **Revista de Informática Teórica e Aplicada**, v. 4, n. 2, p.109-139, 1999.
- LINDEN, R. **Algoritmos Genéticos: Uma Importante Ferramenta da Inteligência Computacional**. 2. ed. Rio de Janeiro: Brasport, 2008.
- MARCONI, M. de A.; LAKATOS, E. M. **Fundamentos de metodologia científica**. 5. ed. SÃO PAULO: Atlas, 2003. 311p.
- MICHALEWICZ, Z. **“Genetic algorithm + data structures = evolution programs”**, 3º ed. Springer-Verlag, 1996.
- MITCHELL, M., **An Introduction to Genetic Algorithms**. MIT, Cambridge: 1996.
- MITCHELL, M.; HOLLAND, J. H.; FORREST, S. When will a genetic algorithm outperform hill climbing? In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, 6, 1994, San Mateo, **Electronic annals...** San Mateo, CA: Morgan Kaufmann, 1994, p. 51-58. Disponível em: <<http://web.cecs.pdx.edu/~mm/nips93.pdf>> Acesso em: 05 mai. 2011.
- MITCHELL, M.; TAYLOR, C. E. Evolutionary computation: An overview. **Annual Review of Ecology and Systematics**, v. 30, p. 593-616, 1999.
- OBITKO, M. **Uma introdução aos Algoritmos Genéticos com Java applets**, 1998, traduzido para o Português do Brasil por Hermelindo Pinheiro Manoel em 2004. Disponível em: <<http://www.professor.webizu.org/ga/>>. Acesso em: 10 mar. 2011.

PACHECO, M. A. C. **Algoritmos Genéticos: Princípios e Aplicações**. In: Laboratório de Inteligência Computacional Aplicada, ICA. Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro, Versão 1 em 14/07/1999 Disponível em: <<http://www.ica.ele.puc-rio.br/Downloads/38/CE-Apostila-Comp-Evol.pdf>> Acesso em: 10 jan. 2011.

PHILLIPS, F.; SU, Y.S. Advances in evolution and genetics: Implications for technology strategy. **Technological Forecasting and Social Change**. v. 76, p. 597-607, 2009.

PREBYS, E. K. The Genetic Algorithm in Computer Science. **MIT Undergraduate Journal of Mathematics**, v. 1, p. 165–170, jun., 1999.

ROSA, T. O.; LUZ, H. S. Conceitos Básicos de Algoritmos Genéticos: Teoria e Prática. In: ENCONTRO DE ESTUDANTES DE INFORMÁTICA DO TOCANTINS, 11., 2009, Palmas. **Anais Eletrônicos...** Palmas: CULP, 2009. p. 27-37. 1 CD-ROM.

RUNARSSON, T. P.; JONSSON, M. T. Genetic production systems for intelligent problem solving. **Journal of Intelligent Manufacturing**, v. 10, n. 2, p. 181-186, may. 1999.

SILVA, E. L.; MENEZES, E. M. **Metodologia da Pesquisa e Elaboração de Dissertação**. Florianópolis: UFSC, 2005. 4. ed. 138p. Disponível em: <<http://www.posarq.ufsc.br/download/metPesq.pdf>>. Acesso em: 01 jun. 2010.

TANOMARU, J. Motivação, fundamentos, e aplicações de algoritmos genéticos. In: CONGRESSO BRASILEIRO DE REDES NEURAIS, 2. 1995, Curitiba. **Anais...** Curitiba: COPEL/CBRN. v. 1, p. 373 - 403, out. 1995. 1 CD-ROM.

YANG, X. S. Biology-derived algorithms in engineering optimization. **Handbook of Bioinspired Algorithms and Applications**. cap. 32 p. 585-595, 2005.

YANG, X. S.; KOZIEL, S. Computational optimization, modelling and simulation – a paradigm shift. **Procedia Computer Science**. v. 1, n. 1, p. 1291-1294, may, 2010.

VON ZUBEN, F. J. Computação Evolutiva: Uma Abordagem Pragmática. In: JORNADA DE ESTUDOS EM COMPUTAÇÃO DE PIRACICABA E REGIÃO, 1. 2000, Piracicaba. **Anais...** Piracicaba: JECOMP, v. 1, p. 25-45, 2000. 1 CD-ROM.

ZUKHRI, Z.; OMAR, K. Implementation of Genetic Algorithms to Cluster New Students Into Their Classes. In: SEMINAR NASIONAL APLIKASI TEKNOLOGI INFORMASI, 3., 2006, Yogyakarta. **Electronic annals...** Yogyakarta: SNATI, 2006, p. A101-A104, jun. 2006. Disponível em: <<http://journal.uui.ac.id/index.php/Snati/article/view/1462/1233>> Acesso em: 10 mar. 2011.