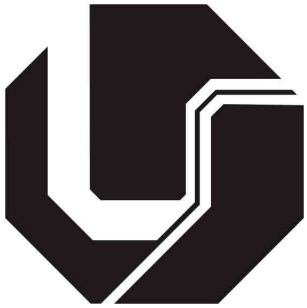
Celso Gonçalves Camilo Junior



Um Algoritmo Auxiliar Paralelo inspirado na Fertilização in Vitro para melhorar o desempenho dos Algoritmos Genéticos

Uberlândia - MG

Março - 2010

Celso Gonçalves Camilo Junior

Um Algoritmo Auxiliar Paralelo inspirado na Fertilização in Vitro para melhorar o desempenho dos Algoritmos Genéticos

Tese apresentada à Coordenação do Programa de Pós-graduação em Engenharia Elétrica da Universidade Federal de Uberlândia para a obtenção do título de Doutor em Ciências na área de concentração Processamento da Informação e linha de pesquisa Inteligência Artificial

Orientador:
Prof. Dr. Keiji Yamanaka

Uberlândia – MG

Março - 2010

Dados Internacionais de Catalogação na Publicação (CIP)

C183a Camilo Junior, Celso Gonçalves, 1980-
Um algoritmo auxiliar paralelo inspirado na fertilização in vitro para melhorar o desempenho dos algoritmos genéticos [manuscrito] / Celso Gonçalves Camilo Junior. - 2010.
216 f. : il.

Orientador: Keiji Yamanaka.

Tese (doutorado) – Universidade Federal de Uberlândia, Programa de Pós-Graduação em Engenharia Elétrica.
Inclui bibliografia.

1. Algoritmos genéticos - Teses. 2. Otimização combinatória - Teses. I. Yamanaka, Keiji . II. Universidade Federal de Uberlândia. Programa de Pós-Graduação em Engenharia Elétrica. IV. Título.

CDU: 681.3.06

Elaborado pelo Sistema de Bibliotecas da UFU / Setor de Catalogação e Classificação

Tese de Doutorado sob o título *"Um Algoritmo Auxiliar Paralelo análogo à Fertilização in Vitro para melhorar o desempenho dos Algoritmos Genéticos"*, para a obtenção do título de Doutor em Ciências na área de concentração Processamento da Informação, defendida por Celso Gonçalves Camilo Junior e aprovada em 05 de março de 2010, em Uberlândia, Minas Gerais, pela banca examinadora constituída pelos doutores:

- Prof. Dr. Keiji Yamanaka, Faculdade de Engenharia Elétrica, UFU, Orientador;
- Prof. Dr. Alexsandro Santos Soares, Departamento de Ciência da Computação, UFG;
- Prof. Dr. Luciano Vieira Lima, Faculdade de Engenharia Elétrica, UFU;
- Prof. Dr. Marcone Jamilson Freitas Souza, Departamento de Computação, UFOP;
- Profa. Dra. Rita Maria da Silva Julia, Faculdade de Computação, UFU.

Prof. Dr. Keiji Yamanaka
Faculdade de Engenharia Elétrica - UFU
Orientador

Prof. Dr. Alexandre Cardoso
Faculdade de Engenharia Elétrica - UFU
Coordenador do Programa de Pós-Graduação em
Engenharia Elétrica

Dedico esta tese à Deus e a minha família, principalmente aos meus pais, meus irmãos, minha esposa e meus filhos, pelo apoio incondicional, incentivo e pela inspiração provocada.

Agradecimentos

Ao professor doutor Keiji Yamanaka, pela orienatação, incentivo e parceria na realização deste trabalho.

Aos professores Prof. Dr. Alexsandro Santos Soares, Profa. Dra. Gina Maira Barbosa de Oliveira, Prof. Dr. Luciano Vieira Lima, Prof. Dr. Marcone Jamilson Freitas Souza e Profa. Dra. Rita Maria da Silva Julia, por participarem das bancas de avaliação, contribuindo decisivamente com esse trabalho.

À todos os colegas da UFU, pelo incentivo e troca de conhecimento durante as longas conversas.

Às senhoras Marli e Cinara, funcionárias da UFU, pelo carinho e atenção dispensada.

Aos meus familiares e amigos, que direta ou indiretamente me estimularam e ajudaram.

Aos meus pais, pela educação e o apoio que me foram dados ao longo de toda a minha vida.

Aos meus irmãos, que sempre me motivaram a ser exemplo.

À minha esposa, pelo amor, respeito, dedicação, generosidade e compreensão.

À minha filha, pelos sorrisos que tornaram os problemas mais simples e que deram sentido à minha dedicação.

Quanto maior o conhecimento, melhor é a percepção da ignorância.
Assim, todo amante do saber tem a humildade
como consequência de sua formação.

Resumo

Várias são as técnicas aplicadas em problemas de otimização. No entanto, poucas alcançam desempenho satisfatório quando o problema é complexo, por exemplo multimodal ou multiobjetivo. Entre as técnicas para otimização estão as metaheurísticas, algoritmos heurísticos de base empírica que não garantem a ótimo global mas, normalmente, encontram boas soluções.

Várias são as metaheurísticas, como exemplo: *Simulated Annealing*, Busca Tabu, GRASP, VND, VNS e Colônia de Formigas. Entre as metaheurísticas, os algoritmos da Computação Evolucionária são muito usados, dado a eficácia e as características modular e adaptativa. Estratégia Evolutiva, Programação Genética e Programação Evolutiva, são exemplos de Algoritmos Evolucionários. No entanto, o mais popular na literatura é o Algoritmo Genético, apesar das dificuldades de convergência em alguns casos.

Algoritmos Genéticos são métodos de otimização e busca inspirados nos mecanismos de evolução de população de seres vivos. Os algoritmos, baseados nesta técnica, seguem o princípio da seleção natural e sobrevivência do mais apto de Charles Darwin.

Analisando a evolução do algoritmo, onde várias gerações são produzidas, uma a cada iteração, e considerando que, a cada nova geração, a anterior é descartada, em parte ou na totalidade, percebe-se que os AGs podem estar eliminando informações relevantes, presentes nos indivíduos descartados, que não foram transmitidas ou mesmo avaliadas pelo algoritmo, causando assim uma perda de informação.

Por isso e considerando os poucos trabalhos que abordam a melhoria no aproveitamento das informações, além da necessidade de apresentar soluções evolucionárias de aplicabilidade mais ampla, esse trabalho propõe o Algoritmo Auxiliar Paralelo (AAP), que objetiva auxiliar os AGs com bons indivíduos a partir de um melhor tratamento das estruturas presentes nas populações de pais.

O AAP é um algoritmo auxiliar executado em um fluxo paralelo aos AGs e que recombina cromossomos para maximizar o aproveitamento das informações presentes nos indivíduos. Como resultado, o módulo pode gerar indivíduos artificiais mais aptos, que são inseridos na nova geração e manipulados pelos AGs na iteração seguinte.

Inspirado na Fertilização in Vitro e no *Preimplantation Genetic Diagnosis*, que analisa e seleciona bons pré-embriões para serem transferidos à mãe, o AAP segue um fluxo de Coleta, Manipulação, Seleção e Transferência de bons indivíduos.

Para testar o desempenho do algoritmo proposto (AAP) e de seus operadores quando acoplados aos AGs, optou-se por dois problemas *benchmark*. O primeiro, de minimização da função *Rastrigin*, por ter um grande espaço de busca, ser não linear e ter um alto grau de multimodalidade e, o segundo, o da Mochila Multidimensional (*Multidimensional Knapsack Problem*), por ser um problema altamente combinatório e multidimensional. Com

isso, foi possível medir o desempenho da proposta em dois tipos diferentes de problema: otimização de função multimodal não restritiva e otimização combinatória restritiva.

O AAP foi testado e comparado com o AG canônico, identificando a melhoria de desempenho com o acréscimo da proposta, e com um algorítmo híbrido AG-BT, que tem características de busca local e global.

Foram construídos 39 cenários dos problemas abordados para os testes. Os resultados apresentados mostram o AAP com uma boa ferramenta para auxiliar o AG a melhorar o desempenho de convergência. Percebe-se, também, que houve uma considerável melhoria na velocidade de convergência sem prejudicar a qualidade da solução final.

Dados os resultados e a estrutura modular do AAP, que permite outras variações e novos operadores, acredita-se que o AAP pode ser útil em várias aplicações e aplicável a outras heurísticas populacionais.

Palavras-chave: Velocidade de Convergência, Algoritmos Genéticos, Computação Evolucionária, Metaheurística, Otimização.

Abstract

There are several techniques applied to optimization problems. However, few achieve satisfactory performance when the problem is complex, for example multi-modal or multi-objective. The metaheuristics, examples of optimization techniques, are heuristic algorithms that can not guarantee the global optimum, but usually find good solutions.

There are several metaheuristics, as an example: Simulated Anneling, Tabu Search, GRASP, VND, VNS and Ant Colony. Among the metaheuristics, the algorithms of Evolutionary Computation is one of the most used, since the effectiveness and modular/adaptive features. Evolution strategy, Genetic programming and Evolutionary programming are examples of evolutionary algorithms, however, the pioneer and the most popular in the literature is the Genetic Algorithm, despite the difficulties of convergence in some cases.

Genetic Algorithms (GA) are optimization and search methods inspired by the evolution of living beings population. Algorithms, based on this technique, follow the principle of natural selection and survival of the fittest (Charles Darwin).

When we analyze the GA, where several generations are produced, one on each iteration, and considering that each new generation the old one is partially or totally discarded, the GA may be removing relevant information within individuals discarded, which were not transmitted or evaluated by the algorithm.

Therefore, and considering the few studies that address the improvement in the use of the information, and the need to present evolutionary solutions with wider applicability, this paper proposes the Algoritmo Auxiliar Paralelo (AAP), which aims to assist the GA with good individuals from better treatment of the structures present in parents populations.

The AAP is an auxiliary module running on a parallel flow to the GA and that recombine chromosomes to maximize the use of the information present in individuals. As a result, the module can generate artificial fittest individuals, which are inserted in the new generation and manipulated by the GA in the next iteration.

Inspired by In Vitro Fertilization and the Preimplantation Genetic Diagnosis, which reviews and selects good pre-embryos to be transferred to the mother, the AAP following a flow of Collection, Manipulate, Select and Transfer of good individuals.

To test the performance of the proposed algorithm (AAP), and their operators, when linked to the GA, we chose two benchmark problems. The first, Rastrigin function by having a large search space, non-linear and have a high degree of multimodality. The second, the Multidimensional Knapsack Problem, as a multidimensional and highly combinatorial problem. Therefore, it was possible to measure the performance of the proposal in two different types of problem: non-restrictive multimodal function and restrictive combinatorial optimization.

The AAP has been tested and compared with the canonical GA, identifying the performance of the proposal, and with a hybrid algorithm GA-TS (Tabu Search), which has characteristics of global and local search.

We constructed 39 sets of the addressed problems for the tests. The results show the AAP as a good tool to assist the GA to improve the convergence performance. It is noticed also that there was a considerable improvement in the speed of convergence without affecting the quality of the final solution.

Given the results and the modular structure of AAP, allowing other changes and new operators, we believed that the AAP may be useful in various applications and applicable to other populations heuristics.

Keywords: Speed of Convergence, Genetic Algorithms, Evolutionary Computation, Metaheuristic, Optimization.

Lista de Figuras

Lista de Tabelas

Lista de Siglas

AAP	Algoritmo Auxiliar Paralelo	24
AC	*Artificial Chromosome Generating Mechanism*	74
ADF	*Additively Decomposable Function*	34
AEs	Algoritmos Evolucionários	22
AG	Algoritmo Genético	22
AG-BT	Algoritmo Genético Busca Tabu	80
AGD	Algoritmo Genético Dinâmico	46
AR	*Assisted Recombination*	88
BAI	Baixo Aproveitamento de Informação	60
BT	Busca Tabu	80
CMGA	*Cure Mechanism Genetic Algorithm*	69
CP	Condição de Parada	97
CPE	*Clearing Procedure with Elitist*	79
CX	*Cycle Crossover*	39
DCGA	*Dynastic Changes Mechanism Genetic Algorithm*	70
DM	*Displacement Mutation*	41
DMG	Divisão do Material Genético	97
EAR	*Exploratory Assisted Recombination*	91
EDA	*Estimation of Distribution Algorithm*	80
EM	*Exchange Mutation*	42
FISH	*Flourescent In Vitro Hybridization*	85
FIV	Fertilização In Vitro	84
HFC-I	*Implicit Hierarchical Fair Competition*	79

IGA	*Immune Genetic Algorithm*	69
ISM	*Insertion Mutation*	42
IVM	*Inversion Mutation*	42
LT	Lista Tabu	82
Max	Máximo	97
Md	Média	97
MFO	Menor valor de Função encontrado	97
MG	*Mining Gene structures*	73
MGSPGA	*Mining Gene structures on Sub-Population Genetic Algorithm*	73
Min	Mínimo	97
NNRC	*Nearest Neighbors Replacement Crowding*	79
OBX	*Order-Based Crossover*	38
OD	Operador de Duplicação	71
OF	Operador de Fabricação	71
PCR	*Polymerase Chain Reaction*	85
PF	Função de Participação	81
PGD	*Preimplantation Genetic Diagnosis*	84
PI	Perda de Informação	60
PME	Problema de Mínimo Enganador	63
PMM	Problema da Mochila Multidimensional	116
PMX	*Partially Matched Crossover*	39
Pop	População	97
PS	Pressão de Seleção	57
PSO	*Particle Swarm Optimization*	124
QA	Quantidade de Avaliações	97
QG	Quantidade de Gerações	97
QtdG	Quantidade de Genes no cromossomo	97

RA	Reprodução Assistida	84
SCT	*Species Conservation Technique*	79
SGA	*Simple Genetic Algorithm*	27
SIM	*Simple Inversion Mutation*	42
SM	*Scramble Mutation*	41
SPGA	*Sub-Population Genetic Algorithm*	73
SSAG	*Steady-State* Algoritmo Genético	80
TMGS	*Threshold Mining Gene Structures*	74
Ts	Tempo em segundos	97
TSP	*Travel Salesman Problem*	70
TxS	Taxa de Sucesso	97
UMDA	*Univariate Marginal Distribution Algorithm*	82
UOBX	*Uniform Order-Based Crossover*	38
WMGS	*Weighting Mining Gene Structures*	74

Sumário

1 Introdução

Várias são as técnicas aplicadas em problemas de otimização. No entanto, poucas alcançam desempenho satisfatório quando o problema é complexo, por exemplo multimodal ou multiobjetivo. Por exemplo, alguns algoritmos da Programação Matemática, que utilizam como guia de busca o gradiente, tem grandes dificuldades e, quase sempre, não atingem o ótimo global em problemas multimodais. Já as metaheurísticas[1] apresentam bons resultados e, por isso, são bastante utilizadas para esses cenários, apesar de não garantirem o ótimo global.

Podem-se dividir as metaheurísticas, quanto à estratégia de busca, em dois grupos, sendo o primeiro de busca populacional e, o segundo, não populacional. A estratégia populacional inicia a busca com vários pontos no espaço de busca e, por meio da interação desses, tenta levá-los para um ponto de ótimo da função objetivo, a cada iteração. Essa, portanto, explora o espaço de busca em vários pontos simultaneamente, fazendo assim um paralelismo na busca. Já a não populacional baseia-se em um único ponto para efetuar a exploração do espaço, quase sempre munida de técnicas para fugir de ótimos locais. Ambas as estratégias demonstram bons resultados; dependendo do problema uma ou outra estratégia é mais adequada.

Como exemplos de metaheurística, podem-se citar: *Simulated Annealing*, Busca Tabu, GRASP, VND, VNS, Colônia de Formigas e Algoritmos Genéticos. Entre esses, os Algoritmos Evolucionários (AEs), especialmente os Algoritmos Genéticos (AGs), apresentam ótimos resultados, por isso um dos métodos mais populares entre os pesquisadores.

Algoritmos Genéticos são métodos de otimização e busca inspirados nos mecanismos de evolução de população de seres vivos. Os algoritmos baseados nesta técnica seguem o princípio da seleção natural e sobrevivência do mais apto de Charles Darwin (GOLDBERG,

[1]Metaheurística, introduzido em Glover (1986), representa um conjunto de algoritmos heurísticos genéricos estudados desde a década de 1970. Estes métodos se baseiam em ideias de diversas fontes para realizar a busca da solução para problemas de otimização (FREITAS et al., 2009). São geralmente aplicadas a problemas que não se conhece um algoritmo eficiente.

1989).

Uma das vantagens de um algoritmo genético é a simplificação na formulação e solução de problemas de otimização. AGs simples normalmente trabalham com descrições de entrada formadas por cadeias de bits de tamanho fixo. Outros tipos de AGs podem trabalhar com cadeias de bits de tamanho variável, como, por exemplo, AGs usados para Programação Genética (RODRIGUES, 2003; FERREIRA, 2001).

Os AGs são indicados para a solução de problemas complexos de otimização que, por exemplo, envolvem um grande número de variáveis e, consequentemente, espaços de soluções de dimensões elevadas. No entanto, em alguns casos o desempenho, relação tempo/qualidade da solução, não é satisfatório (GEN; CHENG, 1997; MICHALEWICZ, 1996), por isso alguns trabalhos são desenvolvidos com o intuito de melhorá-lo (PARK et al., 2000; RONG-LONG; KOZO, 2005; RAJAN; MOHAN; MANIVANNAN, 2002; WU et al., 2004; RUTTKAY; EIBEN; RAUE, 1995; YEN et al., 1998; YANG; DOUGLAS, 1998; MUSIL; WILMUT; CHAPMAN, 1999; CHAINATE; THAPATSUWAN; PONGCHAROEN, 2007). Desses, alguns trabalhos focam na velocidade; no entanto, a maioria busca a eficácia na qualidade, onde a manutenção de diversidade genética para evitar convergência prematura é um ponto bastante pesquisado (TACKETT; CARMI, 1994; MAHFOUD, 1992; SHIMODAIRA, 2002; MAHFOUD, 1995).

1.1 Objetivo

Os AGs são divididos, normalmente, em 6 partes: geração da população inicial, avaliação da população, a seleção de pais, o cruzamento, a mutação e a substituição da população. Exceto a geração da população inicial, todas as outras partes estão em um fluxo cíclico que gera uma nova população a cada iteração.

Analisando a evolução do algoritmo, onde várias gerações são produzidas, uma a cada iteração, e considerando que, a cada nova geração a anterior é descartada em parte ou na totalidade, percebe-se que os AGs podem estar eliminando informações relevantes, presentes nos indivíduos descartados, que não foram transmitidas ou mesmo avaliadas pelo algoritmo, causando assim uma perda de informação.

Além da perda, pode-se ter, dependendo do operador de seleção e cruzamento utilizado, um baixo aproveitamento das informações presentes nos indivíduos, já que a seleção (*selection pressure*) pode restringir o acesso a determinadas informações e o cruzamento pode ignorar algumas combinações.

Em vista disso, qualquer contribuição que visa à melhoria dos AGs é benéfica. Por isso, esse trabalho, dando continuidade aos trabalhos CAMILO e YAMANAKA (2007, 2009), tem como objetivo investigar a possibilidade de diminuir a perda e melhorar o aproveitamento das informações presentes nos indivíduos, por meio de um melhor tratamento das estruturas presentes nas populações de pais.

1.2 Justificativa

Podem-se apresentar duas principais justificativas para o estudo desta tese: a pouca quantidade de estudos (CHANG; CHEN; FAN, 2008; CHANG et al., 2008) propondo o melhor aproveitamento, e/ou a minimização da perda das informações nos AGs e a necessidade de criar, ou melhorar, as soluções da computação evolucionária de aplicabilidade mais ampla.

Várias são as alterações propostas que provocam melhorias nos AGs para as mais diversas aplicações (CHANG; CHEN; FAN, 2008; MATHIAS; WHITLEY, 1992); no entanto, estas modificações são direcionadas a uma aplicação ou, no máximo, a uma classe de problemas. Por isso, e segundo Mitchell e Forrest (1994), devem-se estudar novos operadores ou modificações que, de preferência, tenham uma maior abrangência e, assim, possam ser largamente utilizados, assim como os próprios AGs.

1.3 Proposta e contribuições

Com o intuito de minimizar a perda e melhorar o aproveitamento das informações, este trabalho propõe o Algoritmo Auxiliar Paralelo (AAP), que é um algoritmo auxiliar executado em um fluxo paralelo aos AGs e que recombina cromossomos para maximizar o aproveitamento das informações presentes nos indivíduos, oriundos das populações criadas pelo AG ou gerados pelo operador do AAP. Como resultado, o módulo pode gerar indivíduos artificiais mais aptos, que são inseridos na nova geração e manipulados pelo AG na iteração seguinte.

Inspirado e análogo à Fertilização in Vitro e ao *Preimplantation Genetic Diagnosis*, que analisa e seleciona bons pré-embriões para serem transferidos à mãe, o AAP segue um fluxo de Coleta, Manipulação, Seleção e Transferência de bons indivíduos.

Diferentemente de alguns trabalhos na literatura (SINGH; DEB, 2006; RAJAN; MOHAN; MANIVANNAN, 2002; YANG; DOUGLAS, 1998) que, para melhorar o desempenho, modifi-

cam os operadores dos AGs ou hibridizam com outras metaheurísticas, o AAP propõe um módulo auxiliar que não altera as estruturas funcionais dos AGs e que trata da perda e do aproveitamento das informações.

Minimizando a perda e melhorando o aproveitamento das informações espera-se melhoria no desempenho dos AGs, que poderão contar com mais informações relevantes e com uma maior velocidade de evolução.

1.4 Organização da tese

O restante deste trabalho está dividido em capítulos, sendo o capítulo 2 sobre os AGs, descrição e operadores; o capítulo 3 sobre a convergência e desempenho dos AGs, onde são apresentados alguns mecanismos de funcionamento e problemas difíceis para os AGs; o capítulo 4 mostra modificações que melhoram o desempenho dos AGs, onde algumas técnicas evolucionárias mais recentes são apresentadas e exemplos de hibridismo e nichos são descritos, além da definição da proposta desse trabalho, o AAP; o capítulo 5 contém os experimentos para testar o AAP quanto à sua eficiência e eficácia; e, por fim, o capítulo 6 apresenta a conclusão e trabalhos futuros.

2 Algoritmos Genéticos

Nos últimos anos a computação inspirada na natureza tem atraído muito atenção. A natureza tem servido como uma rica fonte de conceitos, princípios e mecanismos para o projeto de sistemas computacionais artificiais. Entre esses sistemas estão os algoritmos evolucionários, como exemplo: Programação Genética (PG), Programação Evolucionária e os Algoritmos Genéticos (AGs).

O interesse dos pesquisadores da área de computação em biologia é consequência do bom desempenho de estruturas biológicas na resolução de problemas difíceis, inerentes a vida e a sobrevivência. Para alguns biólogos, um dos mecanismos que leva a estas proezas notáveis de solução de problemas é a seleção natural (Charles Darwin). Dada a grande quantidade de problemas difíceis, os pesquisadores da área de computação aplicam os bem sucedidos mecanismos encontrados na natureza para solucionar esses problemas.

Como exemplos de problemas de otimização de difícil solução (MICHALEWICZ, 1996), podem-se citar: otimização de funções matemáticas, otimização combinatória, otimização de planejamento, problemas de roteirização, otimização de layout de circuitos, otimização de distribuição e otimização em negócios. Observa-se que grande parte desses problemas modelam aplicações reais.

Credita-se a grande aplicabilidade dos AGs ao bom desempenho e à fácil adaptação aos problemas, dada a estrutura evolutiva básica e modular desenvolvida.

Por se tratar de uma técnica amplamente utilizada, que tem bom desempenho e ser base para esse trabalho, este capítulo discorre sobre os AGs. Na seção 2.1 definem-se os conceitos; na seção 2.2 apresentam-se algumas formas de representar e codificar a solução; na seção 2.3 discorre-se sobre a população inicial, na seção 2.4 discorre-se sobre as formas de avaliação, as seções 2.5, 2.6, 2.7 e 2.8 apresentam os métodos de seleção, cruzamento, mutação e reprodução, respectivamente; a seção 2.9 apresenta os diversos parâmetros dos AGs; a seção 2.10 apresenta uma análise da importância e uso dos operadores de cruzamento e mutação; e, por fim, a seção 2.11 discorre sobre o hibridismo e as várias

formas.

2.1 Conceitos

Desenvolvido por John Holland e popularizado por David Goldberg (GOLDBERG, 1989), os AGs consistem em métodos de busca e otimização inspirados em princípios da genética de G. Mendel e na teoria da evolução natural das espécies de Darwin (HAUPT; HAUPT, 2004a).

Segundo Darwin, os indivíduos mais aptos têm, em condições iguais de ambiente, maior chance de reproduzirem e, assim, ter mais descendentes, e propagarem seus códigos genéticos (cromossomos) para as próximas gerações. Portanto, podem-se afirmar que as boas informações genéticas perpetuam ao longo do tempo, ajudando o melhoramento da espécie.

A primeira tentativa de representar em um modelo matemático as teorias de Darwin foi apresentada no livro *The Genetical Theory of Natural Selection* (FISHER, 1930). Assim como a aprendizagem, a evolução é uma forma poderosa de adaptação. No entanto, ao invés de ser um processo de vida, a evolução é um processo de gerações.

Após a publicação de Fisher, John Holland iniciou seu estudo sobre processos naturais adaptáveis que culminou na criação do *Simple Genetic Algorithm* (SGA). No início o objetivo era estudar o fenômeno de adaptação natural para propor modelos computacionais análogos. Após alguns refinamentos, Holland publicou em 1975 seu livro *Adaptation in Natural and Artificial Systems*, considerada a primeira referência de AGs, e uma das mais importantes.

Segundo Holland (1992), o SGA poderia solucionar qualquer problema que apresentasse as mesmas características da evolução. O algoritmo trabalha com uma população de algumas cadeias de bits (0s e 1s), denominadas cromossomos, e tem uma execução cíclica para simular as gerações. Análogo à natureza, os AGs evoluem o código genético da população durante as gerações para adaptar-se (resolver) a um problema específico, mesmo sem ter informações detalhadas do problema. Assim, pode-se afirmar que os AGs, na busca por soluções para o problema, empregam um processo adaptativo, já que a informação corrente influencia a busca futura, e paralelo, pois várias soluções são consideradas ao mesmo tempo. Uma analogia entre os AGs e a natureza é representada na Tabela 2.

Tabela 2: Termologia usada pelos AGs dada a analogia com a Natureza

Natureza	**Algoritmo Genético**
Cromossomo	Estrutura de dados que representa a solução
Indivíduo	Mesmo que cromossomo
Gene	Característica (variável que compõe o cromossomo)
Alelo	Valor da característica
Locus	Posição do gene no cromossomo
Genótipo	Cromossomo codificado
Fenótipo	Cromossomo decodificado
Populacão	Conjunto de soluções
Geracão	Ciclo

Como pode-se observar no pseudo-código (SOARES, 1997) do SGA, definido pelo algoritmo 2.1.1, o processo evolutivo do SGA parte de um conjunto de soluções, também chamadas de indivíduos ou cromossomos, candidatas para o problema abordado. Em seguida, esses indivíduos (cromossomos) são analisados por uma função de avaliação que determina quão bom o indivíduo é como solução potencial para o problema tratado. Posteriormente, são aplicados operadores genéticos de seleção, cruzamento e mutação sobre os indivíduos anteriores, objetivando a criação de novas soluções. Após varias gerações (iterações), espera-se que os novos indivíduos possuam características melhores que as de seus progenitores, ou seja, que as novas soluções encontradas sejam superiores as anteriores.

Algoritmo 2.1.1: ALGORITMO GENÉTICO SIMPLES()

```
Definindo {
        função desempenho
        formação do indivíduo e tamanho da população
        probabilidade dos operadores
}
INICIALIZAR POPULAÇÃO ALEATÓRIA();
Enquanto não alcançar critério de convergência faça{
        AVALIAÇÃO()
        SELEÇÃO()
        CRUZAMENTO()
        MUTAÇÃO()
}
```

A Figura 1 apresenta a estrutura básica dos AGs e o fluxo cíclico, onde a cada iteração os operadores criam uma nova geração.

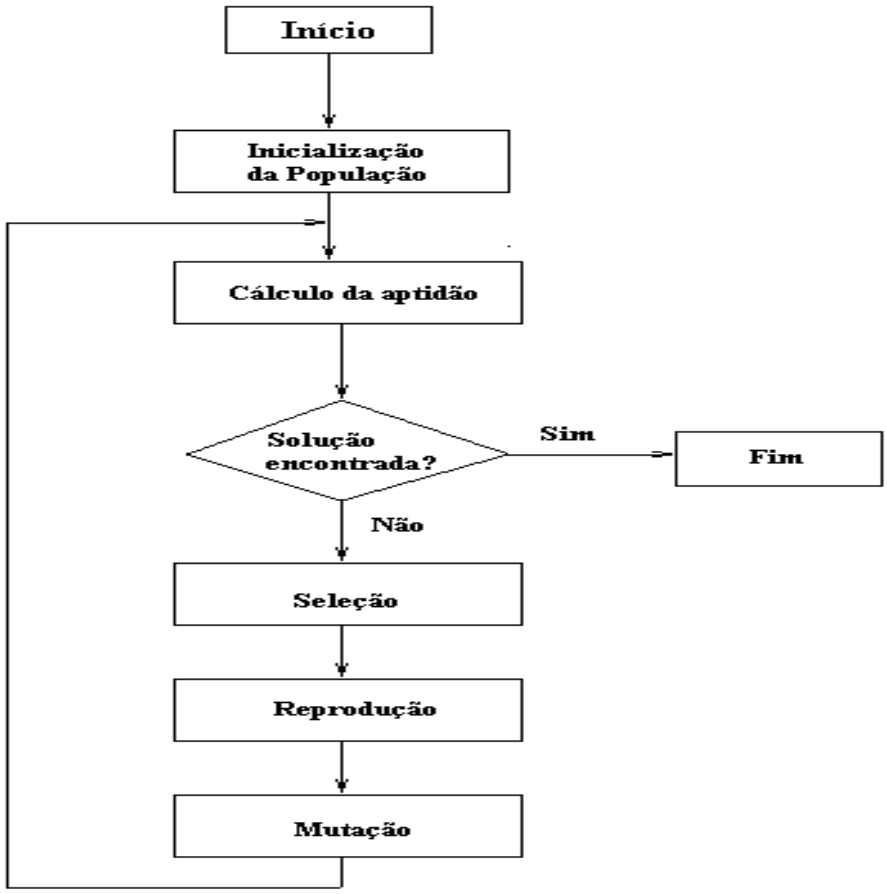


Figura 1: Fluxo de execução dos AGs

Os AGs diferem-se dos métodos tradicionais determinísticos de otimização, pois utilizam meios probabilísticos para encontrar o resultado, trabalham com parâmetros codificados e avaliam cada indivíduo isoladamente, possuindo um paralelismo implícito (FREITAS et al., 2007).

Algumas das vantagens de se utilizarem AGs são (HAUPT; HAUPT, 2004b):

- A capacidade deles de otimizar variáveis contínuas ou discretas;
- O fato de não requerem informações de derivadas;
- A habilidade deles de efetuarem busca simultânea a partir de um grande número de pontos;
- O fato deles lidarem com um grande número de variáveis;
- A boa adaptabilidade deles para ambientes de processamento paralelo;

- Facilidade para fugir de mínimos locais, podendo trabalhar com variáveis codificadas;
- O fato deles proporcionarem um ótimo conjunto de soluções e não apenas uma única;
- A otimização é feita por meio da codificação de variáveis;
- A propriedade que eles apresentam de trabalhar com dados gerados numericamente, dados experimentais, ou funções analíticas.

Apesar disso, os AGs não devem ser considerados como a melhor alternativa para resolução de todos os problemas, apesar de serem empregados com sucesso para solucionar uma vasta variedade deles.

AGs são mais eficientes na resolução de problemas complexos, com grande número de variáveis e espaço de busca não linear. São empregados também com sucesso na resolução de problemas onde não há conhecimento prévio sobre a resolução do problema ou quando não se tem conhecimento de qualquer outra técnica específica para solucionar o problema.

Nas seções seguintes são apresentados os procedimentos, operadores e parâmetros que definem os AGs.

2.2 Representação

Uma das primeiras decisões que deve-se tomar no projeto dos AGs é qual representação será usada. A representação das possíveis soluções do espaço de busca de um problema define a estrutura do cromossomo a ser manipulado pelo algoritmo.

A representação do cromossomo depende do tipo de problema e do que se deseja manipular geneticamente. Os principais tipos de representação são apresentados pela Tabela 3.

Tabela 3: Principais tipos de representação

Representação	**Problemas**
Binária	Numéricos, Inteiros
Números Reais	Numéricos
Permutação de Símbolos	Baseados em Ordem
Símbolos Repetidos	Grupamento

A escolha da representação do cromossomo é uma definição importante para o sucesso da busca dos AGs, pois influencia no espaço de busca a ser varrido pelo algoritmo e na definição dos operadores de crossover e mutação que serão utilizados (BARRA, 2007; ROTHLAUF, 2002). Apesar da independência de execução dos operadores, os projetos devem ser definidos juntos, para garantir um bom funcionamento do algoritmo.

São apresentadas, nas seções seguintes, três das principais formas de representação de soluções nos AGs. Na seção 2.2.1 a representação binária, na seção 2.2.2 a representação GRAY e na seção 2.2.3 a representação Real.

2.2.1 Representação Binária

Nessa representação os cromossomos são formados por uma cadeia de bits {0,1}. Essa é uma das representações mais usadas na literatura, por ser de fácil implementação e por apresentar bons resultados em alguns problemas.

A Figura 2 ilustra um cromossomo representado através da codificação binária. Nesse exemplo, os três primeiros genes representam a variável X e os dois últimos a variável Y.

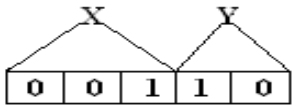


Figura 2: Representação de um indivíduo - Codificação binária

2.2.2 Representação Gray

Tal como na representação binária, um cromossomo representado através da representação Gray é formado por uma cadeia de bits {0,1}. Este tipo de representação mantém sempre em um bit a distância entre números adjacentes (FOGEL; BÄCK; MICHALEWICZ, 2000), auxiliando na busca local.

Enquanto a representação binária pode ampliar o espaço de busca por soluções a ser explorado pelos AGs, a representação Gray ajuda na convergência dos AGs. Por isso, pode favorecer a precisão da solução, mas pode levar a um ótimo local (MOGNON, 2004). A Tabela 4 demonstra a representação binária e a Gray para números de 0 a 15.

Tabela 4: Representação Gray

Decimal	Binário	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

2.2.3 Representação Real

Pelo fato de trabalhar diretamente com números reais, a representação real facilita o trabalho com variáveis continuas. Nesse tipo de representação, os alelos dos genes são números reais, o que modifica a implementação dos operadores genéticos de cruzamento e mutação. Por isso, faz-se necessário a implementação de operadores específicos para esse tipo de representação.

Além das apresentadas, outras formas de representações são usadas com sucesso em algumas aplicações, como exemplo nos trabalhos Achiche, Baron e Balazinski (2003), Freitag, Hildebrand e Moraga (1999). Mais detalhes e comparações de desempenho entre os diversos tipos de codificação apresentadas podem ser vistas nos trabalhos de Rowe et al. (2004), Hajela e Lin (2000), Rothlauf (2002), Chakraborty e Janikow (2003).

2.3 População Inicial

A população inicial é o ponto de partida dos AGs, por isso, tem grande influência na convergência e no desempenho do algoritmo (TOGAN; DALOGLU, 2008). Uma boa população inicial deve apresentar diversidade, para dar aos AGs o maior número de matéria-prima possível. Dessa forma os AGs iniciam a busca com conhecimento amplo do espaço

de busca e, por isso, com maior probabilidade de encontrar um bom resultado (HILL, 1999).

Existem várias abordagens para gerar a população inicial dos AGs, são elas:

Geração Aleatória Atribui-se a cada gene do cromossomo um valor gerado aleatoriamente dentro do domínio estabelecido. Essa é a maneira mais usada de geração da população inicial, dada a simplicidade de implementação (SYBERFELDT; PERSSON, 2009).

Geração por otimização anterior Nesse método soluções encontradas por otimizações anteriores do problema são inseridas na população inicial (LOUIS; LI, 2000). Essas soluções podem ser fornecidas pelos próprios AGs, em execuções anteriores, ou por outros algoritmos de otimização.

Geração por Conhecimento Avançado Nesse método usa-se o conhecimento avançado associado ao problema para criar uma ou mais soluções que serão inseridas na população inicial (ZUBEN, 2000). A grande desvantagem é a necessidade de conhecimento avançado prévio do problema e a dificuldade de criar manualmente essas soluções (SYBERFELDT; PERSSON, 2009).

Geração Heurística Baseado em uma função guia e nas informações disponíveis, são geradas e/ou melhoradas as soluções que irão compor a população inicial. Essa abordagem foi tratada por Grefenstette (1987), que já entendia que o acréscimo de bom material genético na população inicial poderia auxiliar o algoritmo a encontrar melhores resultados.

Geração Complementar Em cromossomos com representação binária, gera-se metade da população aleatoriamente e a outra metade recebe a inversão dos bits da primeira metade. Essa técnica garante que toda posição na cadeia tem 0 e 1. Espera-se com esse método criar uma diversidade genética a partir da metade aleatória.

Geração Híbrida Gera-se a população inicial a partir da combinação de duas ou mais abordagens citadas.

Uma preocupação que se deve ter em todos os métodos é que não se gere indivíduos inválidos, no caso de problemas restritivos.

2.4 Avaliação

Cada indivíduo da população é avaliado pela função de avaliação para determinar a aptidão do mesmo para o problema. Assim, pode-se considerar a função de avaliação como o elo de ligação entre os AGs e o problema a ser resolvido (MOLE, 2002), pois é através dela que os AGs qualificam a solução para o problema. Portanto, pode-se afirmar que a função de avaliação é o guia dos AGs na busca.

Apesar de ser altamente dependente do problema, a função de avaliação não é necessariamente completamente determinada pela definição do problema. Por exemplo, os AGs podem tratar a inviabilidade das soluções pela inclusão de uma função de penalização na função de avaliação (LINDEN, 2006; DIJK, 2001). Nesse caso, a solução infactível seria penalizada durante a avaliação, minimizando assim a propagação desse material genético.

Na literatura encontra-se algumas formas de classificar a função de avaliação (ver detalhes em (JANSEN, 1999)). Entre essas, um tipo muito usado de função de avaliação é a *additively decomposable function* (ADF). Nesse, a função é dividida e o resultado é a soma das partes, sendo que cada parte depende de alguns genes. Caso as partes dependam de genes diferentes, a ADF é chamada de separável. Caso as partes dependam dos mesmos genes, a função é chamada de uniformemente escalado (DIJK, 2001).

2.5 Seleção

O operador de seleção tem como objetivo selecionar, dentre toda a população, indivíduos com boa aptidão para gerarem descendentes durante o processo de evolução. Na natureza, acontece processo similar, onde os indivíduos competem entre si pelo recurso limitado. Os mais adaptados conseguem ter os recursos e tem maior chance de gerar descendentes. Isso é chamado de pressão de seleção.

Quanto mais pressão de seleção maior é intensificação da busca em torno da região das melhores soluções, e quanto menos pressão maior a diversificação da busca. A pressão de seleção pode variar de zero, onde os indivíduos são escolhidos aleatoriamente, até a pressão máxima, onde os melhores indivíduos são sempre escolhidos, sendo que um balanço entre esses dois extremos é desejável. Com a pressão muito baixa teremos o *genetic drift* dominando a convergência, definido como uma busca aleatória com barreiras de absorção (DIJK, 2001; ASOH; MÜHLENBEIN, 1994; GOLDBERG; SEGREST, 1987). Já com a pressão muito alta, causa-se a convergência prematura, devido a rápida diminuição da diversidade

genética, e o *hitchhiking*, dado a não evolução dos bons indivíduos que carregam elementos ruins no cromossomo (MITCHELL; FORREST; HOLLAND, 1991).

Existem diversos métodos que podem ser utilizados pelos AGs para realizar a seleção, dentre eles: Seleção por *Ranking*, Seleção Bi-classista, Seleção por Diversidade, Seleção Local, Seleção Truncada (MüHLENBEIN; SCHLIERKAMP-VOOSEN, 1993), Seleção por Bando (TACKETT; CARAMI, 1994), Seleção por Gênero (SANCHEZ-VELAZCO; BULLINARIA, 2003), Seleção por Torneio e Seleção por Roleta (GOLDBERG, 1989), sendo esse último o mais utilizado (PAPPA, 2002; MARIANO, 2007; LINDEN, 2006). Os trabalhos Blickle e Thiele (1995, 1996) apresentam comparações entre alguns tipos de seleção.

No método de Seleção por Roleta, cada individuo da população ocupa um espaço na roleta proporcional ao valor da sua aptidão, sendo assim os indivíduos mais aptos possuem mais chances de serem selecionados. A Figura 3 ilustra um exemplo do método da Roleta. A equação (2.1) define a probabilidade de seleção p_i, de um individuo i, com aptidão $f(x_i)$, onde N representa o total de indivíduos:

Cromossomo		X	Aptidão (x^2)
A	100100	36	1296
B	010010	18	324
C	010110	22	484
D	000001	1	1

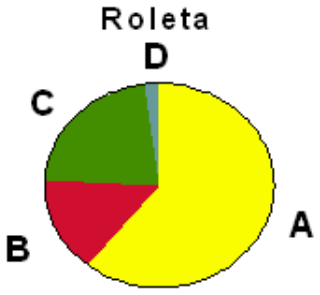


Figura 3: Exemplo do método da Roleta

$$p_i = \frac{f(x_i)}{\sum_{k=1}^{N} f(x_k)} \tag{2.1}$$

O método da Roleta, apesar de ser o método mais usado, tem problemas. No caso de uma população inicial com indivíduos consideravelmente superiores, o método tende a convergência prematura. Já no fim da execução, quando o valores ficam similares, a pressão de seleção diminui e o algoritmo não consegue bom desempenho (DIJK, 2001).

Uma alternativa, muito usada também, é o método do Torneio. Nesse, um grupo de indivíduos é selecionado aleatoriamente e o mais apto é selecionado como progenitor. Esse método tem pressão de seleção constante, pois seleciona a partir do *ranking* de alguns indivíduos. Dessa forma, evita o problema de oscilação de pressão de seleção encontrado no método da roleta. A pressão da seleção nesse método é determinado pelo tamanho

do torneio (número de indivíduos que participam do torneio), sendo que quanto maior o tamanho maior a pressão de seleção (DIJK, 2001).

2.6 Cruzamento

Após a seleção, são criados os descendentes a partir da recombinação das características genéticas dos progenitores. Assim, espera-se combinar bons pedaços de material genético e criar bons indivíduos.

Considerando que no início do algoritmo os bons blocos de material genético estão dispersos, o operador de cruzamento tem a função de juntá-los em um único indivíduo e, assim, produzir o melhor indivíduo. No entanto, durante essas recombinações o operador deve-se preocupar em não destruir as boas combinações já feitas ou pré-existentes. Ou seja, é necessário fazer o *mixing* (THIERENS; GOLDBERG, 1993) mas evitando o *disruption* dos bons blocos (DIJK, 2001).

Para o processo de recombinação, indivíduos progenitores são selecionados e recombinados com a probabilidade p_c, chamada de probabilidade de cruzamento. Para isso, um número uniforme e randômico r é gerado. Caso $r \leq p_c$, os indivíduos selecionados completam a recombinação. Caso $r > p_c$, a recombinação não acontece e os filhos são cópias dos pais.

Como já mencionado, a escolha do operador de cruzamento é influenciada pela codificação estabelecida no algoritmo. A junção da codificação e dos operadores, que modificam o código genético dos indivíduos, constituem os fatores mais importantes para o sucesso ou fracasso do algoritmo (BOOKER et al., 1997).

Existem vários métodos que podem ser utilizados para realizar essa recombinação nos algoritmos evolucionários (GOLDBERG, 1989; BOOKER et al., 1997; SPEARS, 1997). Apesar da maioria desses serem dependentes do problema, indicados em casos mais específicos, existem bons operadores de cruzamento de caráter mais geral. Entre esses, cita-se:

Cruzamento de n pontos O cruzamento de n pontos é a generalização do cruzamento de ponto único, onde $n = 1$. Inspirado na natureza, que troca pedaços de cromossomos, no cruzamento de ponto único é escolhido aleatoriamente um ponto de corte, o qual divide um individuo em duas partes, e troca-se as partes após esse ponto entre os pais (HOLLAND, 1992), ver exemplo na Figura 4. O cruzamento de 2 pontos ($n = 2$) é similar, no entanto, utiliza dois pontos de corte e a parte do cromossomo

que fica entre os pontos é trocada (JONG, 1975; SPEARS; JONG, 1991), ver exemplo na Figura 5. Esses são os métodos mais utilizados de operadores de cruzamento (GOLDBERG, 1989).

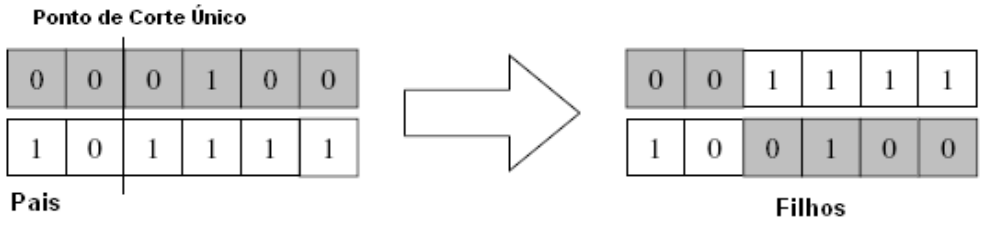


Figura 4: Exemplo de Cruzamento de 1 ponto

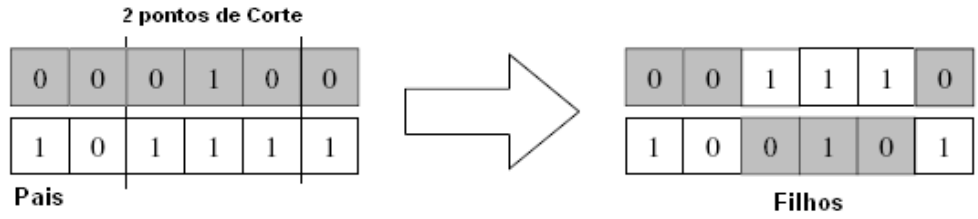


Figura 5: Exemplo de Cruzamento de 2 pontos

Cruzamento Uniforme Nesse método todos os alelos são trocados com uma certa probabilidade, p_e, conhecido como probabilidade de troca (*swapping probability*). Usualmente, é atribuído a essa probabilidade o valor de $0,5$ (SYWERDA, 1989; SPEARS, 1997), ver exemplo na Figura 6.

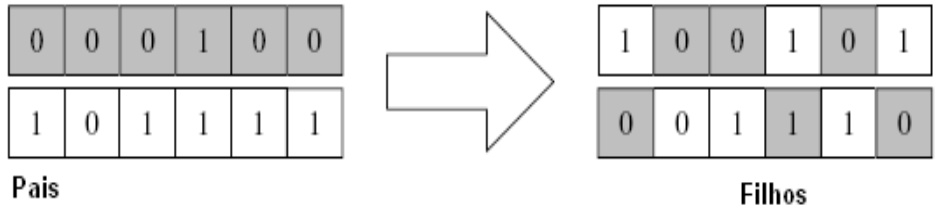


Figura 6: Exemplo de Cruzamento Uniforme

Os métodos apresentados são muito utilizados e têm bom desempenho, no entanto, não podem ser usados com a codificação real e não são indicados para problemas de busca com permutação de características, como por exemplo o problema do Caixeiro Viajante, e nos problemas de escalonamento, dado que esses métodos geram filhos que representam soluções inviáveis para esses problemas.

Uma opção para o uso dos métodos citados é o uso do operador de reparação específico para o problema, que tem a função de transformar os indivíduos inválidos em

soluções factíveis. Outra alternativa, é o uso de operadores de cruzamento desenvolvidos especificamente para problemas combinatoriais. Entre as várias opções (LARRAñA et al., 1999), cita-se:

Cruzamento Uniforme Baseado em Ordem Conhecido como *Uniform Order-Based Crossover* (UOBX), esse seleciona 2 pais (P_1 e P_2) aleatoriamente e gera randomicamente uma máscara binária. O filho C_1 recebe os genes do Pai P_1 quando o alelo da máscara for 1. Os genes de P_1 correspondentes as posições com alelos 0 na máscara são ordenados segundo a ordem em que aparecem no P_2 e copiados para o C_1 nas posições vazias. O C_2 é criado da mesma forma, invertendo somente os papeis de P_1 e P_2. A Figura 7 ilustra esse operador (SASTRY; GOLDBERG; KENDALL, 2005).

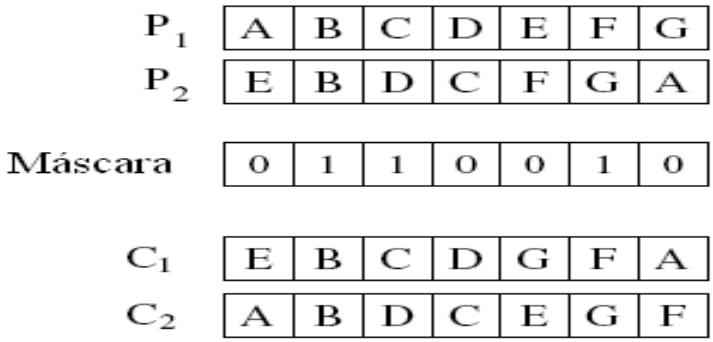


Figura 7: Exemplo de Cruzamento Uniforme Baseado em Ordem

Cruzamento Baseado em Ordem Chamado de *Order-Based Crossover* (OBX), esse operador é uma variante do método anterior. Dados os 2 pais (P_1 e P_2), gera-se aleatoriamente 2 pontos de corte. Os genes entre esses pontos de corte são copiados para os filhos (C_1 e C_2), sendo a carga genética de P_1 para C_1 e a do P_2 para C_2. Finalizada essa primeira fase, a segunda fase consiste em preencher os espaços vazios. Para tal e começando do segundo ponto de corte, copia-se os genes que não existem do pai que gerou o outro filho, na ordem que aparecem (DAVIS, 1985). A Figura 8 ilustra esse operador (SASTRY; GOLDBERG; KENDALL, 2005).

Cruzamento Parcial Conhecido como *Partially Matched Crossover* (PMX), esse operador sempre gera indivíduos válidos, além de preservar a ordenação dentro do cromossomo (GOLDBERG; LINGLE, 1985). No PMX, dados os pais (p_1 e p_2), dois pontos de corte são selecionados aleatoriamente. Os filhos f_1 e f_2 herdam integralmente, preservando a ordem e a posição de cada gene, as seqüências parciais entre os dois pontos de corte respectivamente de p_2 e p_1. Cada gene de f_1, ainda não definido,

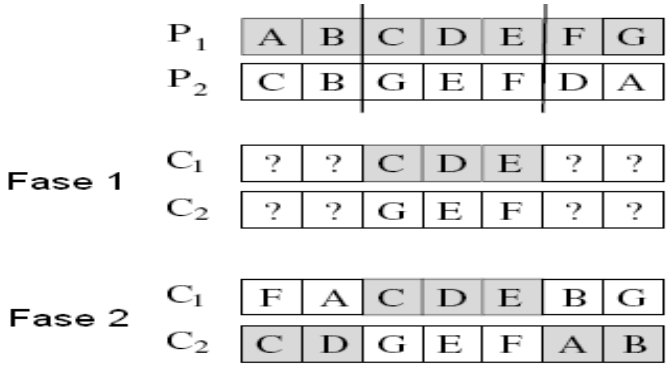


Figura 8: Exemplo de Cruzamento Baseado em Ordem

é preenchido a partir dos respectivos genes do seu pai p_1, e o f_2 da mesma forma que f_1 trocando o p_1 pelo p_2. Como exemplo, na Figura 9 o f_1 herda o elemento 1, que já está na sequência parcial de f_1. Para evitar a solução infactível, o alelo 1 é trocado pelo alelo 3, segundo definição no mapeamento.

$$\left.\begin{aligned} p_1 &= (1, 2, \mid 3, 4, 5, \mid 6) \\ p_2 &= (6, 3, \mid 1, 4, 5, \mid 2) \end{aligned}\right\} \rightarrow \left\{\begin{aligned} f_1 &= (3, 2, \mid 1, 4, 5, \mid 6) \\ f_2 &= (6, 1, \mid 3, 4, 5, \mid 2) \end{aligned}\right.$$

Figura 9: Exemplo do PMX

Cruzamento Cíclico Chamado de *Cycle Crossover* (CX), esse operador gera filhos que preservam a posição absoluta dos elementos provenientes dos cromossomos pais (OLIVER; SMITH; HOLLAND, 1987). Para exemplificar o funcionamento, considere 2 pais (P_1 e P_2) conforme a Figura 10 (SASTRY; GOLDBERG; KENDALL, 2005). Para começar a gerar o C_1, o primeiro gene do P_1 é copiado para C_1, alelo 9. Esse gene mapeia o alelo 1 em P_2, assim copia-se o alelo 1 em C_1 na mesma posição que aparece em P_1. O gene com alelo 1 em P_1 mapeia o alelo 4 em P_2, assim copia-se o alelo 4 em C_1 na mesma posição que aparece em P_1. O gene com alelo 4 em P_1 mapeia o alelo 6 em P_2, assim copia-se o alelo 6 em C_1 na mesma posição que aparece em P_1. O gene com alelo 6 em P_1 mapeia o alelo 9 em P_2, no entanto o alelo 9 já foi incluído no C_1, fechando assim o ciclo. Os espaços vazios de C_1 são completados pelos genes de P_2, respeitando a ordem e a posição. Para criar o C_2 inverte-se os pais de P_1 e P_2.

A variedade de operadores de cruzamento demonstra que não existe o melhor operador, mas sim, operadores que se adaptam melhor a determinados tipos de problemas.

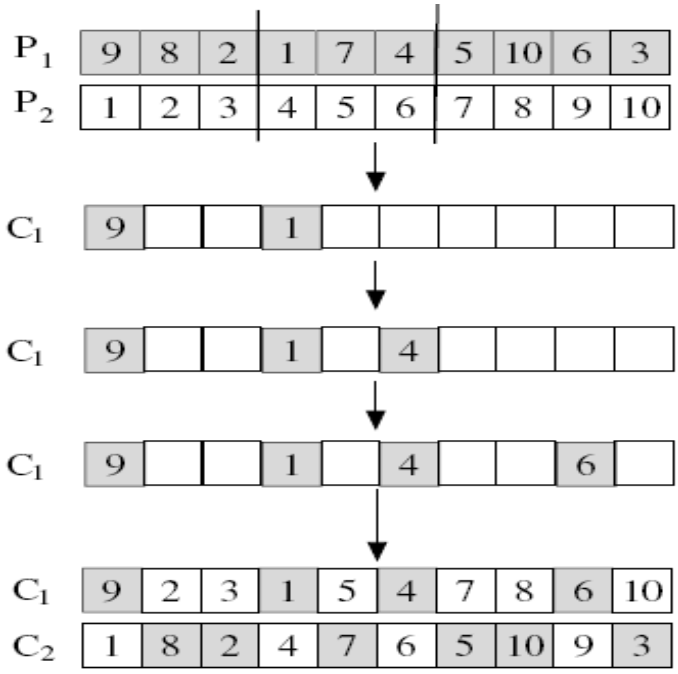


Figura 10: Exemplo do Cruzamento Cíclico

2.7 Mutação

O operador de mutação tem como objetivos introduzir novas características genéticas na população e restaurar características perdidas ao longo do processo evolutivo.

Com isso, promove a diversidade genética entre os indivíduos da população e induz o algoritmo a explorar novas regiões dentro do espaço de busca a procura de possíveis soluções para o problema abordado.

Entre os operadores de mutação para codificação binária, cita-se:

Mutação Clássica: esse operador de mutação varre todos os genes do cromossomo e a cada gene muta-se o valor com a probabilidade p_m (taxa de mutação). A mutação é feita pela inversão do gene, onde for 0 altera-se para 1 e onde for 1 altera-se para 0. Deve-se procurar um valor de p_m que permita um balanço entre a descoberta de novas soluções e, ao mesmo tempo, que não provoque excessiva destruição dos bons blocos de material genético já descobertos. Sugere-se que p_m seja $1/t$, onde t é o número de genes do cromossomo (BäCK, 1993; MUNETOMO; GOLDBERG, 1999). A Figura 11 demonstra um exemplo deste operador.

Mutação Uniforme: similar ao cruzamento uniforme, nesse método de mutação cria-se uma máscara indicando os genes a serem mutados no cromossomo, como demostrado

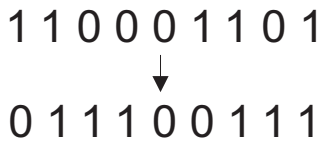


Figura 11: Operador de mutação clássico para codificação binária

na Figura 12.

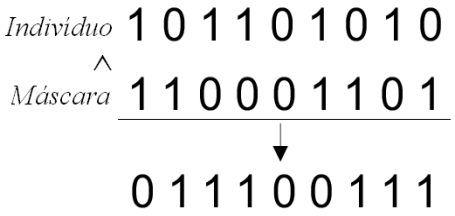


Figura 12: Operador de mutação uniforme para codificação binária

Assim como os operadores de cruzamento, os operadores de mutação também são dependentes da codificação. Portanto, além dos operadores citados que são utilizados na codificação binária, existem outros operadores de mutação (LARRAñA et al., 1999) frequentemente utilizados na codificação real e na codificação por permutação, como exemplo:

Mutação por Deslocamento Chamado de *Displacement Mutation* (DM), esse operador seleciona um subconjunto de genes aleatoriamente, remove-o da estrutura do cromossomo e insere-o em uma posição definida aleatoriamente (MICHALEWICZ, 1996), ver Figura 13.

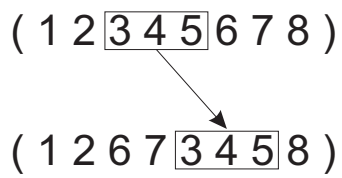


Figura 13: Operador de mutação *Displacement Mutation* (DM) (LARRAñA et al., 1999)

Mutação por Mistura Chamado de *Scramble Mutation* (SM) e desenvolvido por Syswerda (1991), o SM seleciona aleatoriamente um subconjunto de genes para ser misturado, como exemplo a Figura 14.

Mutação por Troca Chamado de *Exchange Mutation* (EM) e desenvolvido por Banzhaf (1990), o EM seleciona aleatoriamente 2 genes do cromossomo e permuta seus alelos. Esse operador também é conhecido como: *swap mutation, point mutation, reciprocal*

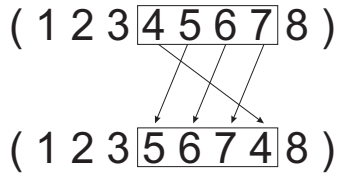


Figura 14: Operador de mutação *Scramble Mutation* (SM) (LARRAñA et al., 1999)

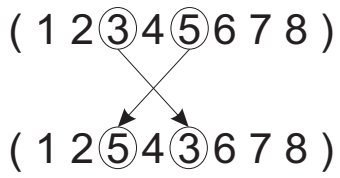


Figura 15: Operador de mutação *Exchange Mutation* (EM) (LARRAñA et al., 1999)

exchange mutation e *order based mutation* (LARRAñA et al., 1999). A Figura 15 ilustra esse operador.

Mutação por Inserção Chamado de *Position based mutation* ou *Insertion Mutation* (ISM), nesse método escolhe-se um gene do cromossomo aleatoriamente e sorteia-se nova posição a qual o gene escolhido irá ocupar, conforme mostra a Figura 16.

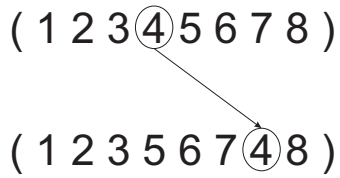


Figura 16: Operador de mutação *Insertion Mutation* (ISM) (LARRAñA et al., 1999)

Mutação por Inversão Simples: Chamado de *Simple Inversion Mutation* (SIM), nesse sorteia-se 2 pontos do cromossomo e inverte-se a ordem dos genes contidos entre esses pontos, como demonstrado na Figura 17.

Mutação por Inversão Chamado de *Cut-Inverse Mutation* ou *Inversion Mutation*(IVM), o IVM é semelhante ao DM, pois seleciona um subconjunto de genes, remove-o do cromossomo e o insere em ordem inversa após uma posição escolhida aleatoriamente. Este processo é demostrado na Figura 18.

2.8 Reprodução

Após criar os descendentes com os operadores genéticos, é necessário definir os indivíduos que farão parte da próxima geração. Para tal, aplica-se as técnicas de reprodução que considera todos os filhos gerados e a população progenitora corrente.

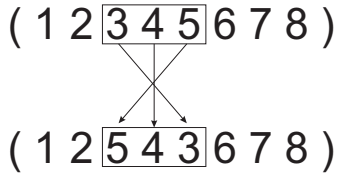


Figura 17: Operador de mutação *Simple Inversion Mutation* (SIM) (LARRAñA et al., 1999)

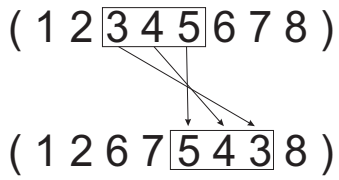


Figura 18: Operador de mutação *Inversion Mutation* (IVM) (LARRAñA et al., 1999)

Existem várias técnicas para substituir a população corrente de pais pela nova população, entre elas cita-se:

Substituição Total Nessa, todos os membros da população de pais correntes são eliminados e substituídos pelos novos indivíduos criados. Essa é a técnica mais popular dada a sua facilidade de implementação e a ausência de parâmetros.

Steady-state Nessa técnica elimina-se n pais correntes e substitui-os por n novos indivíduos. Um parâmetro que deve ser definido é quantos e quais pais serão substituídos. Entre as possibilidades, uma opção é substituir os n piores pais pelos n novos filhos, outra opção é substituir os pais que foram utilizados no cruzamento, ou ainda, substituir aleatoriamente os n pais pelos n filhos.

***Steady-state* sem duplicatas** Semelhante ao *Steady-state*, substitui-se parcialmente os pais mas com a restrição de não permitir cromossomos iguais na nova geração. Apesar de aumentar o custo computacional, essa técnica pode melhorar a exploração do espaço de busca.

Elitismo Uma das técnicas mais usadas na literatura e uma variação do *Steady-state*, o elitismo preserva o melhor indivíduo da população de pais corrente e substitui o restante por novos indivíduos (SHIZEN; YANG, 2004; MOGNON, 2004).

Exceto a técnica de Substituição Total, todos as outras técnicas de substituição preservam indivíduos correntes para a próxima geração com o intuito de preservar o bom material genético encontrado.

2.9 Parâmetros dos AGs

Ao elaborar um AG, o projetista deve considerar alguns parâmetros que influenciam diretamente no funcionamento do mesmo. São eles:

Tamanho da População: O parâmetro n_{pop} influência diretamente na exploração do espaço de busca por possíveis soluções para o problema abordado a ser explorado pelos AGs, no tempo de execução e na demanda por recursos computacionais. Uma população pequena possui amostragem insuficiente do espaço de busca. Uma população grande, apesar de ter maior representação do espaço, leva uma convergência mais lenta, levando a uma necessidade de mais recursos computacionais ou no aumento do tempo necessário para execução do algoritmo.

Taxa de Cruzamento: Esse parâmetro p_c controla a frequência com a qual o operador de cruzamento é aplicado. A cada nova geração provavelmente $p_c * n_{pop}$ serão realizados. Um valor de p_c baixo significa pouco aproveitamento da informação existente, já um alto valor de p_c pode provocar convergência prematura.

Taxa de Mutação: Esse parâmetro p_m define a probabilidade de um indivíduo ter seus genes alterados pelo operador de mutação. A escolha de um valor muito baixo de p_m pode não satisfazer a necessidade de exploração e levar o algoritmo à estagnação. Por outro lado, um alto valor de p_m conduz a uma busca aleatória.

Critério de Parada: O Critério de Parada determina o método para finalizar a execução dos AGs, existem diversas formas de se determinar qual é o momento exato para os AGs interromperem a busca, entre elas: atingir o ótimo ou um valor conhecido, homogeneidade da população (analisada normalmente pela média da população), ausência de melhorias após n gerações, número de chamadas à função de avaliação e número de gerações, essas duas últimas as mais usadas na literatura.

A escolha ideal dos parâmetros é um problema não linear e depende do tipo de problema tratado. Por isso, não é possível encontrar uma boa configuração para generalizar a execução de qualquer tipo de problema.

Entre as sugestões de valores, Jong (1975) sugere $n_{pop} = 50$, $p_c = 0,60$ e $p_m = 0,001$. Grefenstette (1986) analisou vários conjuntos de valores de parâmetros e apresentou sugestões para satisfazer as medidas de desempenho *on-line* e *off-line performance*. Sugere

$n_{pop} = 30$, $p_c = 0,95$ e $p_m = 0,01$, no caso *on-line*, e $n_{pop} = 80$, $p_c = 0,45$ e $p_m = 0,01$, no caso *off-line*.

Schaffer, após um extenso trabalho experimental (SCHAFFER et al., 1989), sugere $n_{pop} = 20$ a 30, $p_c = 0,75$ a $0,95$ e $p_m = 0,005$ a $0,01$. Goldberg (GOLDBERG, 1989) sugere uma fórmula (ver 2.2) para o tamanho da população em função do tamanho (*lenght*) do cromossomo:

$$n_{pop} = 1,65 * 2^{0,21*lenght} \tag{2.2}$$

Além desses, outros trabalhos sugerem como valor de p_c entre 0.6 e 0.9, podendo ser seguramente atribuído 1 quando se usa elitismo (DIJK, 2001). Apesar das sugestões, normalmente o valor é atribuído a partir de testes experimentais, dada a dificuldade de generalização.

Outra forma de definir os parâmetros dos AGs é dinamicamente. Nesse caso os parâmetros são alterados durante a execução do algoritmo de acordo com uma avaliação (adaptativa) ou uma regra pré-estabelecida.

Entre as várias técnicas de parametrização dinâmica, cita-se: Adaptação dinâmica por indivíduo (SRINIVAS; PATNAIK, 1994), Adaptação dinâmica baseada na média (VASCONCELOS; SALDANHA, 1997), Adaptação baseada em *cluster* (ZHANG; CHUNG; ZHONG, 2005), Adaptação baseado no modelo de nuvens (DAI; ZHU; CHEN, 2006), Adaptação baseada em matrizes (LAW; SZETO, 2007), Adaptação baseada em Lógica *Fuzzy* (HERRERA; LOZANO, 1996).

Alguns trabalhos (ANGELINE, 1995; HINTERDING; MICHALEWICZ; EIBEN, 1997; SMITH; FOGARTY, 1997) sugerem classificações para as técnicas, entre eles Eiben et al. (2000) define uma taxonomia (ver Figura 19) que divide as técnicas, segundo o método de alteração, em duas grandes classes: Afinação de parâmetros (pré-estabelecidas antes da execução) e Controle de parâmetros (durante a execução), sendo essa última subdividida em Determinística, Adaptativa e Auto-adaptativa. Nesse mesmo trabalho, sugere-se uma outra dimensão de classificação segundo o componente do algoritmo a ser tratado, podendo ser: a representação, a função de avaliação, os operadores, seleção, substituição e população.

Além do ajuste dinâmico dos parâmetros, uma possibilidade é a adaptabilidade do tipo de operador que será executado. Um AG tradicional usa um operador de cruzamento e um operador de mutação durante a execução. Sabe-se que a escolha desse operador, como já citado, é crítico para o sucesso dos AGs. No entanto, diferentes tipos de operadores podem ser indicados para diferentes problemas e momentos distintos da evolução, dificultando

Figura 19: Taxonomia de classificação das técnicas de parametrização dos AGs (EIBEN et al., 2000)

assim a escolha que normalmente é feita por tentativa-e-erro.

Por isso, o trabalho Hong et al. (2002) propõe Algoritmo Genético Dinâmico (AGD) que pode usar diferentes operadores durante a execução. Cada operador é utilizado segundo uma taxa, alterada durante a evolução pelo desempenho dos filhos gerados pelo mesmo. Dessa forma, o AGD tenta o operador mais adequado ao problema e ao momento da evolução.

2.10 Mutação versus Cruzamento

Duas das áreas mais pesquisadas dos AGs são os operadores de cruzamento e mutação, dado a importância desses para o sucesso do algoritmo. A vários anos, inúmeros trabalhos são desenvolvidos para investigar o comportamento desses operadores e analisar o impacto durante a evolução. L. Fogel apresentou, em 1966, uma demonstração do uso da mutação e da seleção para evoluir autômatos (FOGEL; OWENS; WALSH, 1966). Nesse viés, Rechenberg analisou a Estratégia Evolutiva que também usa a mutação como chave do processo evolutivo e operador principal (RECHENBERG, 1973). O uso da adaptação dinâmica dos parâmetros na mutação também foi analisado e alcançou-se resultados melhores (SCHWEFEL, 1981; BäCK; HOFFMEISTER; SCHWEFEL, 1991). Confirmando a importância da mutação, o trabalho Schaffer et al. (1989) analisou a força desse operador para a evolução. Por outro lado, outros trabalhos como Jong (1975) mostram, através de testes experimentais, a força do cruzamento.

Dada a ideia inicial de que a mutação seria um operador secundário e o cruzamento o principal (HOLLAND, 1992; JONG, 1975; SPEARS; ANAND, 1991; VOSE; LIEPINS, 1991),

Fogel e Atmar (1990) afirmou que no geral o cruzamento não tem vantagens sobre a mutação. Já Schaffer e Eshelman (1991) concluiu que a mutação sozinha não é suficiente para todos os casos. Conclusão elaborada também por Iclanzan (2007), que analisou diferentes algoritmos e encontrou, para um conjunto de funções definido no trabalho, vantagens fundamentais nos algoritmos com recombinação quando comparados aos que não usam recombinação. Além desses, outros trabalhos analisaram a performance da mutação e diferentes formas de AGs (JANSEN; WEGENER, 1999; BAUM; BONEH; GARRETT, 1995; RABINI; RABINOVICH; SINCLAIR, 1998; GARNIER; KALLEL; SCHOENAUER, 1999).

Considerando que as comparações e os resultados foram analisados após testes experimentais, uma possibilidade levantada por Spears (1992) é que a implementação interferiria nos resultados e, por isso, sugeriu uma análise teórica comparativa, focando especialmente na capacidade de rompimento e construção dos operadores. Esse trabalho concluiu que a mutação tem maior poder de rompimento do que o cruzamento, que por sua vez tem maior poder de construção do que a mutação.

O trabalho de Sastry e Goldberg (2007) analisou os operadores *BB-wise mutation* e *uniform BB-wise crossover*, segundo as classes de problemas determinísticos e estocásticos aditivamente separáveis, e concluiu que para problemas deterministos e exponencialmente escalados a mutação é mais eficiente, agilizando o algoritmo em $o(l \log l)$, onde l é o tamanho do problema. Já para os problemas de escala exponencial e com ruído dominante, o cruzamento agiliza o algoritmo em $o(l)$.

Comparando a eficiência, o trabalho Jansen e Wegener (2005) propõe um conjunto de funções em que os AGs com cruzamento tem tempo esperado de otimização polinomial e os algoritmos com base na mutação (sem cruzamento) tem tempo esperado de otimização exponencial.

Nota-se que não existe uma definição geral sobre qual dos operadores é o melhor ou se algum dos operadores é dispensável. Assim, percebe-se que dependendo do problema existe uma configuração mais adequada, que eventualmente pode eliminar um dos operadores ou diminuir o seu impacto no algoritmo.

2.11 Hibridismo

O hibridismo consiste em combinar diferentes técnicas para que com a soma das boas características dessas se tenha uma técnica melhor. Pesquisadores da Computação Evolucionária têm usado com frequência o hibridismo para melhorar o desempenho em

problemas reais. Combina-se, por exemplo, Algoritmos Evolucionários (AEs) com técnicas da Pesquisa Operacional e/ou do Aprendizado de Máquinas, formando assim um híbrido normalmente conveniente ao problema a ser tratado.

Nessa combinação de boas características, os AEs são bons em explorar o espaço de busca e encontrar regiões promissoras, mas são lentos na busca dos melhores pontos dessas regiões (JONG, 2005; PREUX; TALBI, 1999). Por outro lado, as buscas locais (como os métodos baseados em gradiente) e os métodos específicos para o problema (como as heurísticas) são melhores para essa busca de granulação mais fina e apresentam uma dificuldade na busca por regiões promissoras. Assim, pode-se dizer que a combinação dos AEs com a busca local traz, quando requisitado pelo problema, melhoria de desempenho na busca (GOLDBERG; VOESSNER, 1999; MOSCATO, 1989). A junção de AEs com busca local tem sido referenciado na literatura como Algoritmos Meméticos (MOSCATO, 1989) ou Algoritmos de Busca Local Genético.

Com o objetivo de organizar as técnicas já desenvolvidas e ajudar na previsão de futuras, o trabalho Sinha e Goldberg (2003) propõe a seguinte taxonomia para classificar os AEs: Propósito da Hibridização, Arquitetura do Hibridismo e o Método Secundário. As classes não são mutualmente exclusivas, por isso um algoritmo pode estar em mais de uma classe.

Na classificação por Propósito, deve-se procurar a motivação para a incorporação da busca local. Por isso, de acordo com o objetivo da hibridização, essa classe é dividida nas 4 seguintes subclasses.

Exploração Essa classe consiste dos híbridos que usam a busca local como agente de agilização da convergência para o ótimo, dada uma região promissora delimitada pelo AE. Tem-se usado nessa classe os híbridos com *hill-climbers, Simulated Anneling* e *Tabu Search.*

Reparação O uso da busca local para a geração de soluções factíveis e a reparação de soluções infactíveis (ORVOSH; DAVIS, 1993) são os objetivos dos algoritmos dessa classe. Esses algoritmos são úteis em problemas altamente restritivos onde, normalmente, os operadores de cruzamento e mutação produzem infactibilidade.

Otimização de Parâmetros Segundo Sinha e Goldberg (2003), nessa classe os AEs são usados para otimizar os parâmetros de uma segunda técnica. Como exemplo, o uso de AGs para treinar Redes Neurais (BELEW; MCINERNEY; SCHRAUDOLPH, 1991;

MONTANA; BERANEK; INC, 1995) ou AGs para gerar conjunto de regras ou funções de pertinência da Lógica *Fuzzy*.

Substituição Funcional ou Aperfeiçoamento no AE Nessa classe, os métodos secundários são usados para atuar como funções do AE, ou melhorá-lo pelo melhor controle. A segunda técnica pode executar a função de avaliação, cruzamento, mutação ou outras. Como exemplo, usa-se a Rede Neural como modelo de aproximação da função de avaliação e, assim, como avaliadora dos indivíduos. Espera-se, com isso, acelerar a busca em problemas em que a função de avaliação é custosa e complexa (JIN, 2005). Acrescenta-se nessa classificação o uso da segunda técnica para otimizar os parâmetros dos AEs (ELMIHOUB et al., 2006). Como exemplo, usa-se busca local para determinar o tamanho da população do AE (ESPINOZA; MINSKER; GOLDBERG, 2003; ELMIHOUB et al., 2004) ou a Lógica *Fuzzy* para melhorar o desempenho dos AGs (RICHTER; PEAK, 2002).

Dada a forma modular e aberta dos AEs, é possível incorporar outras técnicas em diferentes partes (SCHWEFEL, 1997). Por isso a classe Arquitetura, da taxonomia citada, agrupa os algoritmos de acordo com a disposição da segunda técnica no AE. Essa é subdividida em Alocação Híbrida, Híbridos Assíncronos, Híbridos Hierárquicos e Híbridos Embutidos, como definido a seguir:

Alocação Híbrida Essa classe é caracterizada por dois estágios sequenciais distintos de execução, sendo que em um desses o AE é o ator. Dentro dessa classe existem 3 subclasses: Pré-processamento, onde o AE fornece uma região promissora dentro do espaço de busca e o segundo algoritmo dá sequencia na busca até o termino; Pós-processamento, onde o método secundário é usado somente para obter uma boa solução inicial para o AE, como exemplo, o uso de *Tabu Search* para a inicialização (FLEURENT; FERLAND, 1993; SENTINELLA; CASALINO, 2009); e Intercalado, onde várias iterações acontecem com o AE e método secundário interpolados. Normalmente, cada filho gerado pelo cruzamento sofre uma busca local, por um tempo delimitado, antes de ser introduzido na população. Outra opção é a aplicação da busca local nos pais antes do cruzamento. Essa arquitetura é muito utilizada na literatura (MATHIAS et al., 1994; SILVA et al., 2009; HAMZAçEBI, 2008). A Figura 20 ilustra essa classe.

Híbridos Assíncronos O algoritmo dessa classe tem a cooperação assíncrona entre os métodos, ou seja, valores encontrados pelos métodos podem ser usados pelo ou-

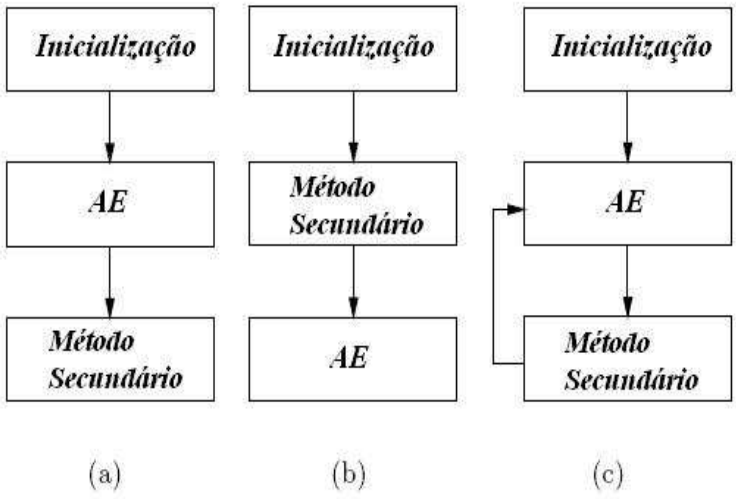


Figura 20: As subclasses Pré-processamento (a), Pós-processamento (b) e Intercalado (c) da classe Posicionamento Híbrido (SINHA; GOLDBERG, 2003)

tro método durante a execução. Normalmente, ambos os métodos usam memória, possibilitando assim um retorno ao estágio de melhor solução quando a busca é deteriorada.

Híbridos Hierárquicos Na forma hierárquica, o algoritmo tem funções com vários níveis de otimização que utilizam diferentes tipos de técnicas, sendo que uma dessas é AE (ROGERS, 1991).

Híbridos Embutidos O algoritmo dessa classe tem o método secundário embutido em um módulo do AE. Essa classe pode ser subdividida de acordo com o estágio da inserção. Podendo ser na Inicialização, na Avaliação, no Cruzamento, na Mutação ou como Operador Especial.

Por fim, é possível classificar os algoritmos híbridos segundo o método empregado junto ao AE. Encontra-se na literatura diversos trabalhos com diferentes tipos de técnicas associadas ao AE, desde da Programação Matemática até o Aprendizado de Máquinas. Assim, uma forma de classificar os algoritmos híbridos é pelo método empregado junto com o AE (SINHA; GOLDBERG, 2003). Como exemplo: Métodos de Busca Local (ZHENG et al., 2007; HAMZAçEBI, 2008; YUN, 2006; HARADA; IKEDA; KOBAYASHI, 2006; DENGIZ; ALTIPARMAK, 1997), Recozimento Simulado (HUNG, 2008; CORDóN; ANEGóN; ZARCO, 2002; HWANG; HE, 2006; HE; HWANG, 2006), Rede Neural (CERAVOLO; FELICE; PIZZUTI, 2009; MISHRA et al., 2009; HUANG; HUNG, 2008; LI; SHI; ZHANG, 2007; GüLER; POLAT; ERGüN, 2005), Lógica Fuzzy (LAU et al., 2009; OH; PEDRYCZ; ROH, 2009; JIN; LI, 2008; LO et al., 2007), Busca Tabu (ZHANG; SHI; GAO, 2008; YANG; ZHANG; BAI, 2008; CHIU et al.,

2007), Árvore de Decisão (CARVALHO; FREITAS, 2004; KIM et al., 2004; FU, 2002), Sistemas Especialistas (SKOLNICK; TONG, 1991), Programação Dinâmica (DUNKER; RADONS; WESTKAMPER, 2005; YAGIURA; IBARAKI, 1996), Raciocínio baseado em Casos (CHANG; LAI; LAI, 2006), Programação Lógica por Restrição (DERIS et al., 1999), *Branch and Bound* (FRENCH; ROBINSON; WILSON, 2001; NAGAR; HERAGU; HADDOCK, 1996).

Além da taxonomia de Sinha e Goldberg (2003), outras são referenciadas na literatura, como os trabalhos Krasnogor e Smith (2005), Talbi (2002).

3 *Convergência e Desempenho dos AGs*

Os AGs são algoritmos que vêm sendo usados em vários áreas de aplicação, pela sua adaptabilidade e eficácia. No entanto, essa relação adaptação/eficácia não é simples, pois são objetivos normalmente conflitantes e, em alguns casos, antagônicos. Usualmente, quando se consegue um bom desempenho, não se tem um algoritmo facilmente adaptável a outros problemas. Por isso, alguns algoritmos (heurísticas) são construídos de forma a privilegiar a eficácia/eficiência e menos a adaptabilidade.

Dada essa vasta aplicabilidade dos AGs, inclusive em problemas reais, torna-se necessário estudos sobre o fluxo de funcionamento do algoritmo para que novos algoritmos sejam propostos e o desempenho seja melhorado, já que em algumas aplicações, principalmente as reais, os AGs apresentam problemas de eficiência.

Apesar de ter uma estrutura simples para implementar, os AGs têm um funcionamento complexo, já que durante o fluxo evolutivo vários fatores e módulos podem interferir no resultado. Muitos desses fatores estão interligados, de forma que a mudança em um perturba e altera o outro, aumentando ainda mais a complexidade e o entendimento dos AGs.

Por isso, esse capítulo apresenta os mecanismos de convergência (fluxo evolutivo de funcionamento) e alguns fatores que podem prejudicar o desempenho dos AGs.

3.1 *Exploration versus Exploitation*

Os AEs pertencem a família dos algoritmos de busca estocásticos gerar-e-testar. Esses algoritmos têm basicamente duas estratégias de busca: *exploration* e *exploitation*, que são a origem do poder de busca desses.

Pode-se dividir a ocorrência da *exploration* e da *exploitation* em três níveis: no nível

de indivíduo, no nível de sub-indivíduo e no nível de um único gene. No primeiro nível, o indivíduo é atômico (indivisível), não permitindo uma visão interna desse. Assim, as informações são limitadas à avaliação do todo. No segundo nível, cada indivíduo é analisado como uma instância de 2^l genes, onde l é o tamanho do indivíduo. Nessa, as informações tratadas e herdadas são de parte do indivíduo. Já o terceiro nível é um caso especial do segundo, onde um único gene forma a unidade a ser tratada e herdada (EIBEN; SCHIPPERS, 1998). Assim, cada operador pode ser projetado para exercer as estratégias nos níveis desejáveis.

Os termos *exploration* e *exploitation* são importantes para descrever o funcionamento dos AEs, os operadores de busca (mutação e recombinação) e a seleção. Cada elemento do algoritmo pode ser caracterizado pela respectiva contribuição para a *exploration* ou a *exploitation*.

Entre as visões existentes, a mais comum é que a seleção é a fonte da *exploitation*, enquanto a *exploration* é feita pelos operadores de recombinação e mutação (ESHELMAN; CARUANA; SCHAFFER, 1989). Desta forma, o uso de material genético caracterizaria *exploitation*. Outra visão, restringe a *exploitation* ao bom uso da informação (HANDA et al., 1997). Para Beyer (1998), a *exploitation* é a habilidade do algoritmo em caminhar na direção do melhoramento desejado.

Outra visão largamente defendida é que a *exploration* e a *exploitation* são forças opostas, dessa forma, quando se acresce uma decresce a outra. Por isso, deve-se buscar uma boa mistura entre elas para se obter bons resultados na busca (KENNETH; SPEARS, 1992; TAN et al., 2009). Das várias técnicas usadas para balanciar as estratégias destaca-se a diminuição da pressão de seleção (ver seção 3.2.2), usada por exemplo no trabalho Sá et al. (2008), e o uso de variação adaptativa dos operadores (TAN et al., 2009).

Outra decisão para se considerar é o momento para enfatizar os esforços na *exploitation* ou na *exploration* (TAN; LEE; KHOR, 2001; TAN et al., 2006). Uma das opções é o uso da *exploration* no início do algoritmo, para extrair a maior quantidade de informação do espaço de busca, e depois o uso da *exploitation*, para fazer a "sintonia fina". Essa divisão de estágios é conhecida na literatura como *explore first, exploit later*. Alguns trabalhos como Abbass (2002), Bambha et al. (2004) demonstram a vantagem de usar essa abordagem.

Para Spears (1992), e em outros trabalhos, o operador da *exploration* é a mutação, com o papel de provocar diversificação, e o da *exploitation* é o cruzamento, com o papel de acelerar o afunilamento genético. Apesar de que para alguns autores, como Fogel, o

cruzamento seja um fator de terceira ordem, que não é necessário para todos os problemas, dado que também não ocorre sempre na natureza (ATMAR, 1992).

Dada a diversidade de visões, o trabalho de Eiben e Schippers (1998) apresenta, após analisar outros trabalhos, um sumário. Percebe-se que são várias as visões, sendo que em alguns casos complementares e em outros conflitantes. Baseado nesse estudo, os autores enumeram algumas hipóteses e perguntas para serem investigadas:

1. Independentemente se as informações são boas ou se são usadas corretamente, o uso de informações conhecidas é igual a *exploitation*. Alternativamente, considera-se *exploitation* somente quando se faz o bom uso da informação.

2. Dado que a escolha de bons indivíduos leva a escolha de boas informações, a seleção é a fonte da *exploitation*.

3. Os operadores mutação e recombinação são puramente *exploration*, dado que o objetivo desses é criar variação.

4. *Exploitation* e *Exploration* são forças opostas que necessitam de balanço.

5. Qual o nível de informação disponível e que parte dessa informação é explorada?

6. Qual a influência da representação? Percebe-se em alguns casos uma grande influência na construção da informação.

7. Em alguns casos os alelos, ou a combinação desses, são explorados pela geração de novos indivíduos contendo esses alelos. Em outros casos, de forma mais sutil, os valores dos genes são determinados somente pela influência dos seus antepassados.

8. Existe diferença da *Exploitation* e da *Exploration* entre os diferentes tipos de AEs.

9. Na maioria das vezes, a *exploitation* nos AEs são feitas de forma implícita, ou seja, o valor de aptidão do indivíduos determina se aquele indivíduo será explorado. Alguns trabalhos tornam a *exploitation* mais explícita, a fim de melhorar o desempenho dos AEs. Mas essa forma não estaria em conflito com a ideia original de um algoritmo elegante, simples e bom?

10. Quanto tempo o material genético pode ou deve ser explorado?

Encontrar a melhor combinação da *exploration* com a *exploitation* e esclarecer o funcionamento das partes que compõem e contribuem para essas estratégias são os grandes

desafios para o aprimoramento das técnicas de busca. Por isso, o assunto foi e continua sendo amplamente pesquisado.

3.2 Convergência

Caracteriza-se como convergência evolutiva o processo de afunilamento genético para um ponto determinado pelos melhores indivíduos da população. Assim, a convergência é um importante sinal de evolução.

O algoritmo que converge demonstra capacidade de evoluir na direção apontada pela função guia. Já um algoritmo sem convergência torna-se puramente aleatório. Por isso, todo algoritmo evolucionário deve lidar com a convergência.

Durante o processo de convergência alguns fatores, como a Diversidade Genética, a Pressão de Seleção, a Convergência Prematura e os Critérios de Convergência, devem ser considerados e tratados para contribuir com a melhoria de desempenho do algoritmo.

3.2.1 Diversidade Genética

Pode-se definir a diversidade genética como uma medida de biodiversidade da variabilidade genética dentro de uma população, ou seja, procura-se observar a quantidade de genes que tendem a variar seus alelos entre si. A variabilidade é fundamental para a boa evolução das espécies, pois acrescenta novas informações na população, evitando assim uma homogenização e um empobrecimento genético. Essa homogenização causa uma maior vulnerabilidade da população às doenças, dado que indivíduos geneticamente semelhantes são igualmente sucetíveis as mesmas doenças, caracterizando assim a variabilidade genética como um importante fator de sobrevivência (WIKIPEDIA.ORG, 2009).

A variabilidade genética é responsável também pela adaptação dos indivíduos, dado que com a mudança do ambiente novas características devem ser inseridas na população para que haja melhor adaptabilidade ao ambiente.

Nos AEs a diversidade é fundamental para a exploração do espaço de busca (YUAN; LI; LI, 2008a), dado que com mais informações amplia-se a área de cobertura do algoritmo e, por consequência, aprimora-se a decisão de priorização de determinada região promissora. A Figura 21 ilustra a cobertura de uma população com alto grau de diversidade e outra com pouca diversidade. Sem a diversidade o algoritmo perde muito da sua força e acaba fixando-se em mínimos locais, além de enfrentar problemas como a Convergência Prema-

tura (ver seção 3.2.4). No entanto, com muita diversidade o algoritmo torna-se puramente aleatório. Por isso, deve-se buscar adequação da diversidade ao problema abordado.

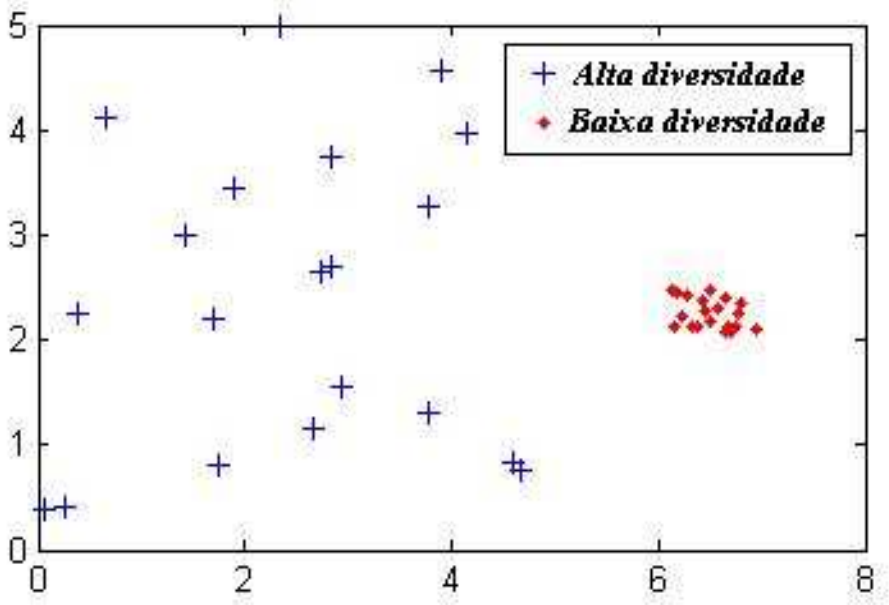


Figura 21: Cobertura do espaço de busca feita por populações com níveis diferentes de diversidade

Os AEs precisam da diversidade também para explorarem o "paralelismo implícito", uma das vantagens dos AEs sobre alguns outros métodos de busca. O paralelismo só ocorre porque o AE trabalha com pontos, em geral, distintos. Caso os pontos fossem idênticos não haveria paralelismo.

Durante a evolução da população novos indivíduos são criados a cada geração. Dada a priorização dos melhores indivíduos como progenitores, o processo evolutivo tende a convergir para indivíduos semelhantes. Assim, a diversidade genética na população tende normalmente a diminuir. Por isso, os AEs tem operadores responsáveis por acrescentar ou manter diversidade durante o processo de evolução, simulando o processo natural de interação genética entre espécies ou a mutação (MAHFOUD, 1992; GLICKMAN; SYCARA, 2000; ZHU; LIU, 2004). Entre as formas de aumentar a diversidade, cita-se o acréscimo da taxa de mutação e o aumento da população com indivíduos geneticamente diferentes dos existentes. Quando se aumenta a taxa de mutação o número de genes alterados aumenta e, assim, os indivíduos tendem a um afastamento um dos outros na mesma razão do aumento da taxa (BARCELLOS, 2000).

Dos métodos usados para medir a distância (diversidade) entre dois indivíduos da população (KIM; MOON, 2004; BURKE; GUSTAFSON; KENDALL, 2004), cita-se Hamming (FREDERICK; SEDLMEYER; WHITE, 1993), a distância euclidiana (YUAN; LI; LI, 2008a),

o desvio padrão dos valores de aptidão e o identificador ancestral (BURKE; GUSTAFSON; KENDALL, 2004). Desses, os mais usados são Hamming e a distância euclidiana. O primeiro faz uma medida genética, pois calcula o número de bits diferentes entre dois cromossomos. Já o segundo faz uma medida fenótipa, pois calcula a diferença entre os valores de aptidão dos cromossomos.

3.2.2 Pressão de Seleção

Define-se como Pressão de Seleção (PS) o grau de priorização dos indivíduos com maior valor de aptidão para a sobrevivência e reprodução. Como a aptidão é avaliada de acordo com o ambiente, pode-se dizer que a PS é a influência que o meio ambiente exerce na seleção genética. Ou seja, a PS são características do meio ambiente imposta à população e que direcionará a priorização de determinados genes que melhor adapta o indivíduo.

Nos AEs a PS está relacionada à velocidade e a direção da busca, pois quanto maior a pressão, mais rápido e focado fica o algoritmo. No entanto, quanto maior a PS, maior é a perda da diversidade genética, pois com o aumento da PS reduz-se o número de indivíduos selecionados para reprodução e, com isso, aumenta-se a semelhança entre os indivíduos da população. Assim, afirma-se que a diversidade genética e a PS são duas forças antagônicas e importantes para busca evolucionária.

Define-se a PS como alta quando as probabilidades de alguns indivíduos serem selecionados para a reprodução em relação a outros diferem muito. Por outro lado, quando as probabilidades de sobrevivência entre os indivíduos são semelhantes, diz-se que a PS é baixa. Quando a PS é nula - todos os indivíduos com a mesma probabilidade de sobrevivência - a busca torna-se puramente aleatória.

A PS determina a velocidade pois interfere na convergência. Quanto maior a pressão, mais rápida a convergência. Caso os indivíduos favorecidos estiverem na direção (próximos) do ponto ótimo global ou de um ponto satisfatório, a PS alta é benéfica pois reduzirá o tempo da busca. Caso os indivíduos favorecidos estiverem distantes do ponto satisfatório, a alta da PS é prejudicial pois pode levar a população a uma convergência prematura para um ponto de ótimo local, além da perda da diversidade, o que provocará uma dificuldade para os AGs em encontrarem um novo ponto dentro do espaço (CAMARGO, 2006).

Controla-se a PS pelo método de seleção ou por um controle de descendentes de determinados indivíduos. Entre os métodos de controle da PS, cita-se o Ranking Linear,

Ranking Geométrico, Corte σ, Nicho e Compartilhamento, Escalamento Linear e a Normalização (CAMARGO, 2006).

3.2.3 Critérios de Convergência

Os algoritmos estocásticos, como os AEs, não possuem mecanismos que indique se a solução encontrada é a melhor possível. Dessa forma, encontrar o momento adequado de parada do algoritmo é também um problema a ser tratado. Caso contrário, o algoritmo pode parar antes de encontrar uma boa solução ou executar durante várias gerações sem necessidade, acarretando em um desperdício de recurso computacional.

Assim, uma opção é parar o algoritmo, ou executar outra ação de melhoria, quando se detectar que a população convergiu para um ponto do espaço, já que a busca após esse momento normalmente não teria uma boa probabilidade de encontrar uma solução melhor.

Para identificar a convergência da população usa-se alguns critérios (VASCONCELOS; SALDANHA, 1997; TRAUTMANN et al., 2008). Entre os mais usados cita-se: o número máximo de gerações, onde predefine-se o limite de gerações e retorna-se como solução o melhor indivíduo encontrado até então, e a convergência do melhor indivíduo, onde analisa-se o melhor indivíduo da geração corrente e o melhor indivíduo da geração anterior segundo a equação (3.1), caso a condição aconteça n vezes, considera-se que o algoritmo convergiu.

$$f_{max\{atual\}} - f_{max\{anterior\}} \leq \varepsilon \tag{3.1}$$

Além desses, usa-se o número de gerações sem melhoria, interrompendo-se o algoritmo após n gerações sem melhora do melhor indivíduo encontrado até então; e a convergência da população, onde analisa-se a proximidade do valor médio de aptidão da população (f_{med}) em relação ao melhor valor de aptidão (f_{max}). Quanto mais próximo f_{med} está de f_{max}, mais homogênea está a população e, consequentemente, menor a diversidade genética, ver Figura 22. Assim, pode-se interromper o algoritmo quando a relação $\frac{f_{med}}{f_{max}}$ atinge determinado fator de homogeneidade f_{hom}, como visto na equação (3.2).

$$1 - \frac{f_{med}}{f_{max}} \leq f_{hom} \tag{3.2}$$

A escolha do melhor critério de convergência não é uma tarefa fácil, por isso requer

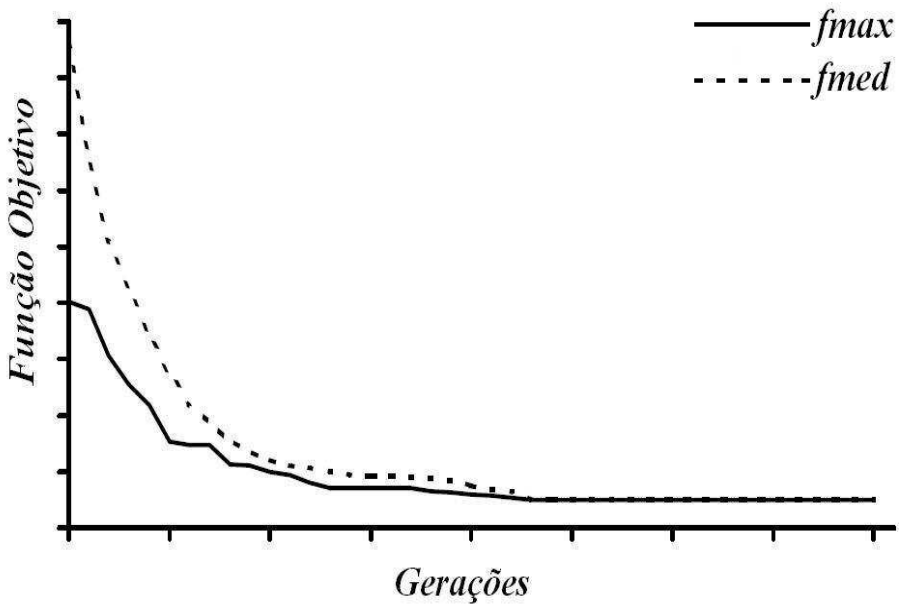


Figura 22: Curva de convergência de um problema de minimização

análise do problema e dos operadores, sendo necessário, em alguns casos, o uso de mais de um critério.

3.2.4 Convergência Prematura

Sempre que se tenta agilizar o processo evolutivo dos AGs uma preocupação fica eminente, a convergência prematura (GOLDBERG, 1989; SULTAN; MAHMUD; S., 2007), definida como a perda de diversidade genética na população de forma acelerada e prejudicial à solução final. Apesar de ser considerada um grande problema quando a convergência é para um ótimo local, distante ou diferente do ótimo global, é desejável caso a convergência seja para o ótimo global.

A discussão sobre a convergência prematura passa por outros conceitos como *exploration* e *exploitation* (apresentadas na seção 3.1), pois pode-se definir convergência prematura como pouca *exploration* e prematura *exploitation*. Os algoritmos que usam das duas estratégias, normalmente, no início usam da busca global para identificar regiões promissoras e, posteriormente, a busca local para encontrar a melhor solução nas regiões promissoras.

Os AGs são algoritmos que começam sua execução privilegiando a estratégia *exploration*, quando sua diversidade genética é alta, e passam a privilegiar a *exploitation* no momento em que uma representativa parcela dos indivíduos tem similaridades.

Para evitar a convergência prematura para ótimos locais é importante fazer uma boa

exploration. No caso dos AGs é feita pela mutação, com baixa probabilidade, e pela recombinação de soluções geradas pelo início do processo evolutivo, normalmente com alta diversidade genética. No entanto, este processo é lento, pois são necessárias várias gerações para se priorizar umas das regiões promissoras.

Fazendo os AGs aproveitarem melhor seus indivíduos e outros novos acredita-se que a velocidade pode ser maior sem aumentar as chances de convergência prematura para um ótimo local.

Uma das principais características do AAP, proposto neste trabalho (ver seção 4.4), é a capacidade de melhor aproveitar as informações presentes na população, gerando bons indivíduos que podem agilizar o processo evolutivo dos AGs.

3.3 A perda e o baixo aproveitamento da informação

O processo evolutivo dos AGs é composto de um fluxo cíclico, em que a cada iteração uma nova geração é criada. A cada nova geração, indivíduos são gerados e introduzidos na população em substituição a outros existentes, que são descartados. No entanto, muitos desses indivíduos eliminados contêm informações nos seus genes importantes para a busca; apesar disso são descartados sem passar pelo cruzamento. Ou seja, foram gerados e eliminados e não contribuíram para a evolução. Por isso, pode-se dizer que a cada iteração várias informações relevantes são perdidas.

Apesar deste processo imitar a evolução das espécies, onde um indivíduo pode nascer e morrer sem gerar descendentes, é notória a perda de informação. A informação perdida pode não voltar para a população, causando, em alguns casos, uma redução da eficácia e caracterizando a perda de oportunidade; ou voltar pelo processo da mutação, que normalmente tem baixa probabilidade e, por isso, pouca eficiência, pois é necessário um tempo até que a informação retorne e seja aproveitada.

É necessário diferenciar a perda de informação (PI) do baixo aproveitamento da informação (BAI). No primeiro, a informação é eliminada da população e depende de algum mecanismo de reinjeção para voltar. Já no BAI, a informação está na população mas não é manipulada e nem analisada pelos operadores. Nesse último caso, há um desperdício de esforço, pois quanto mais rápido e melhor os AGs utilizarem de suas bases de conhecimento disponíveis (população), mais ágil serão as evoluções.

Feita a distinção, é necessário identificar a influência dos operadores nestes fatores (PI

e BAI). Três são os operadores identificados que podem provocar o baixo aproveitamento da informação:

Seleção Durante a seleção alguma estratégia, por exemplo a "Pressão da Seleção", define os pais que irão transmitir seus genes. Neste momento muitos indivíduos com informações relevantes e de baixa aptidão, ou outro critério definido pelo operador, podem não ser selecionados e, assim, não serem aproveitados.

Cruzamento O operador de cruzamento utilizado na combinação entre os cromossomos pode não ser eficaz e, assim, produzir material de baixa qualidade. Uma baixa taxa de cruzamento também causa pouca combinação e aproveitamento.

Substituição Considerando que este operador define quantos novos indivíduos gerados farão parte da geração seguinte, o aproveitamento das informações e dos indivíduos criados pelo cruzamento depende da taxa de substituição. Quanto maior o número de indivíduos introduzidos na população mais relevante será o trabalho de aproveitamento de informações feito pelo cruzamento e pela seleção. Uma baixa taxa de substituição pode prejudicar o aproveitamento das informações.

Já para a perda de informação, identifica-se três operadores capazes de interferir:

Cruzamento A cada nova estrutura criada pelo cruzamento pode-se estar eliminando outras presentes nos pais. Considerando a codificação binária e a informação como sendo os alelos e/ou a ordem destes no cromossomo (schemata), a cada cruzamento um schemata[1] pode estar sendo destruído.

Mutação A mutação é o operador responsável por inserir informação (diversidade) genética na população, no entanto, a inserção elimina outra informação. Por isso, uma alta taxa de mutação pode provocar perda de informação, apesar de que aumenta a chance da reintrodução das informações perdidas.

Substituição Caso a taxa de substituição seja alta há um alto rodízio dos indivíduos na população, o que provoca uma alta taxa de perda de informações.

[1]Plural de schema, que é um padrão de similaridades que descreve um subconjunto do espaço de soluções de um AG (GOLDBERG, 1989). Assim, em um AG binário com l posições o schema é representado como um elemento do conjunto $\{0, 1, *\}^l$. Por exemplo, considere os cromossomos "010" e "011", ambos são considerados instâncias do schema "01*".

Analisando as características anteriores, percebe-se um dilema entre os agravantes da PI e do BAI. Enquanto uma baixa taxa de cruzamento pode ser prejudicial, considerando o BAI, pode ser eficaz para evitar a PI, pois evita a quebra dos schemata. Na substituição, uma alta taxa de novos indivíduos agrava a PI, no entanto, minimiza o BAI, pois faz com que as informações descobertas e aproveitadas pelos operadores anteriores sejam consideradas.

Além do cruzamento e da substituição, a mutação e a seleção exercem, respectivamente, influência na PI e no BAI. A seleção, como já explicado, é fundamental no processo de aproveitamento das informações. Sem um método que consiga diversificar os selecionados, os AGs sofrem de convergência prematura, mesmo tendo na sua base de conhecimento (população) material genético capaz de evitar o mínimo local.

Já a mutação, tem a função de aumentar a diversidade genética da população, fator este, além da PI e o BAI, que interfere no desempenho dos AGs. Quanto maior a diversidade, maior a capacidade de análise do espaço de busca e menor é o desperdício de processamento, já que minerar uma população com informações muito similares prejudica a eficácia do algoritmo. Apesar disso, a taxa de mutação não deve ser muito alta, evitando assim o excesso de PI, causada pela substituição da informação existente por uma nova.

Assim, pode-se concluir que o ideal é um algoritmo que tenha pouca PI, que gere informações diversificadas e que tenha um bom aproveitamento destas. Com o intuito de auxiliar os AGs a equilibrarem estes fatores, este trabalho propõe, e define na seção 4.4, o Algoritmo Auxiliar Paralelo (AAP). O uso do módulo de auxílio em uma estrutura paralela facilita a dissociação dos fatores que, como visto anteriormente, estão interligados. Desta forma, é possível intensificar, por exemplo, a recombinação em uma população paralela e não interferir na PI da população original.

3.4 Problemas difíceis para o AGs

Como apresentado, vários fatores podem influenciar o comportamento dos AGs, inclusive o problema a ser tratado. Alguns desses problemas apresentam características que dificultam a convergência para um ponto de ótimo global ou reduzem a eficiência do algoritmo.

Entre as características que dificultam a resolução do problema (NAUDTS; KALLEL, 2000), citam-se: a decepção (enganadores), a epistasia, o isolamento e a multimodalidade, apresentadas nas seções seguintes.

Na tentativa de medir a dificuldade dos AGs no problema, algumas medidas preditivas foram propostas, entre elas: correlação de distância da aptidão, correlação entre tamanho e operador, variância da epistasia, variância de schema e classificação de hiperplanos. No entanto, estudos sugerem que nenhuma dessas medidas preditivas de dificuldade são confiáveis (NAUDTS; KALLEL, 2000).

3.4.1 Enganadores

Por ser uma das piores formas de confundir os AGs, o problema enganador é um dos mais abordados e tratados na literatura. Considerando que a avaliação guia a seleção por melhores indivíduos e, consequentemente, a busca do algoritmo, qualquer interferência que confunda ou altere o processo de apontamento para o ótimo prejudica muito o desempenho dos AGs.

O problema enganador ocorre quando a aptidão dos schemata instanciados pelo ponto ótimo é menor do que a aptidão de outros schemata competidores primários (em que as posições fixadas são as mesmas) deste schema ótimo. Esta situação pode levar a busca para regiões consideradas promissoras mas que não incluam o ponto ótimo global, ou seja, para ótimos locais possivelmente distantes do ótimo global (CHEN et al., 2008).

Muitos trabalhos na literatura abordam o problema enganador, sendo que alguns definem ou classificam esses (VOSE; LIEPINS, 1991; WHITLEY, 1991), outros constroem exemplos de funções enganadoras (GOLDBERG, 1992; LIEPINS; VOSE, 1990; WHITLEY, 1991), e outros investigam alterações nos AGs para tentar resolver esses (NOVKOVIC; SVERKO, 1998; LIEPINS; VOSE, 1990; LI; LI, 2005; CHEN et al., 2008).

No problema enganador a confusão ocorre pois indivíduos representantes de um ótimo local têm aptidão maior que indivíduos representantes do ponto ótimo global. Assim, os AGs privilegiam e convergem para uma região do espaço que pode não levar para a melhor solução possível.

Para entender o que causa dificuldade para os AGs, Goldberg definiu o Problema Mínimo Enganador (PME), onde o cromossomo binário assume 2 bits e a relação entre os schemata são estabelecidas de forma que instâncias do ótimo local sejam melhores do que instâncias do ótimo global, como ilustrado na Figura 23.

Embora cause engano, os PMEs não são difíceis para um AG resolver, dado que seu nível de engano é limitado. Por isso, foi criada uma classificação para os tipos de engano: *fully deceptive, consistently deceptive* e *partially deceptive.*

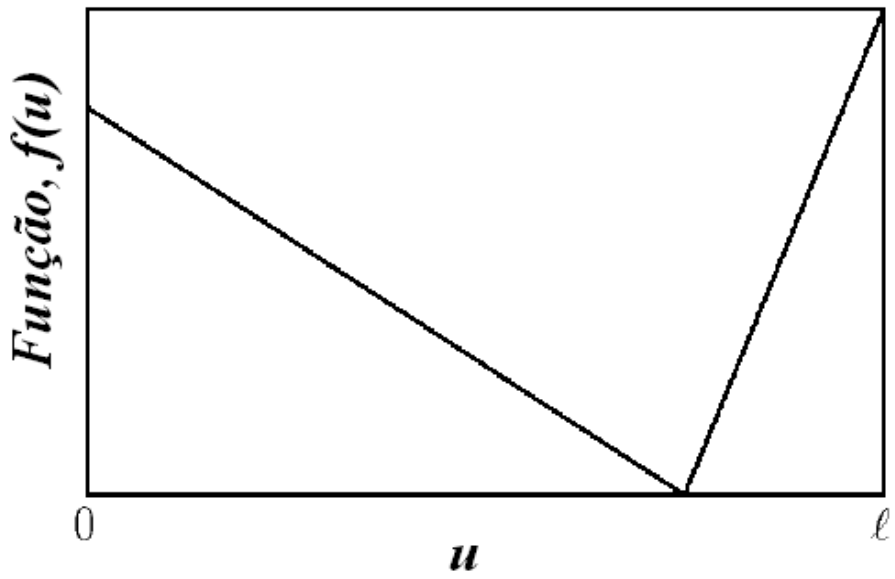


Figura 23: Problema Mínimo Enganador (GOLDBERG; DEB; HORN, 1992)

Considera-se que um problema é *fully deceptive* quando todas as competições primárias de ordem inferior à ordem do problema são vencidas por schemata que são instanciados por um mesmo ponto diferente do ótimo. Rotula-se esse ponto como o atrator decepcionante, que em representações binárias é sempre o complemento do ponto ótimo. Por exemplo, um schema "**1**1**1" que instancia o ponto ótimo global tem como complemento e atrator decepcionante o schema "**0**0**0" (WHITLEY, 1991).

Já o *consistently deceptive* ocorre quando todas as competições primárias de ordem inferior à ordem do problema são vencidas por schemata que não instanciam o ponto ótimo global e pelo menos uma das competições seja vencida pelo atrator decepcionante. Assim, um problema *fully deceptive* é sempre um *consistently deceptive*, apesar da recíproca não ser verdadeira (VENTURINI, 1995).

Quando um problema tem algum grau de engano, mas não é nem *fully deceptive* nem *consistently deceptive*, ele é considerado *partially deceptive* (WHITLEY, 1992).

Dessa forma, percebe-se que nem toda o engano é um problema que pode prejudicar o desempenho dos AGs. Por isso, é necessário analisar o grau para identificar a melhor forma de abordar o problema.

3.4.2 Epistasia

Oriundo da biologia, o termo epistasia significa interação funcional entre genes não alélicos. Ou seja, o quanto que um gene que define uma característica de um indivíduo

influencia o resultado de outra característica.

A ideia de epistasia foi introduzido nos AGs por Davidor (DAVIDOR, 1990) para analisar a interferência da interação funcional genética nos AGs e, assim, apresentá-la como medida de avaliação da dificuldade dos AGs em alguns problemas (QU; XU, 2006).

Na representação binária, a epistasia é caracterizada pela ocorrência da influência de um *bit* em outro do cromossomo. Essa dependência genética interfere na convergência dos AGs, pois obriga os algoritmos a terem schemata de maior ordem, já que somente schema com todos os genes dependentes entre si têm significância evolutiva.

Considerando os níveis de dependência, é possível classificar os problemas segundo o grau da epistasia:

Problemas sem epistasia são problemas com os genes linearmente independentes e, por isso, não apresentam dificuldades para os AGs (ROCHET, 1997). Dada a simplicidade, técnicas gulosas e de menor custo computacional são mais indicadas.

Problemas com epistasia média são problemas com alguns genes interdependentes e, normalmente, dispostos de tal forma que não comprometa muito o cromossomo. Esses é o grau mais indicado para os AGs, pois não é tão simples e nem tão complexo.

Problemas com epistasia alta são problemas com vários genes interdependentes e espalhados no cromossomo, criando um contexto de busca aleatória para os AGs (ROCHET, 1997). Nesse cenário, os AGs não conseguem analisar schemata de menor ordem e direcionar o algoritmo para regiões promissoras, comprometendo a convergência.

Entre os problemas mais usados para abordar a epistasia, estão: as funções NK (KAUFFMAN, 1989) e a *Royal Road* (MITCHELL; FORREST; HOLLAND, 1992), essa última dividida em dois tipos: RR1 e RR2. Outras funções também são usadas, como nos trabalhos Manela e Campbell (1992), Rowe e East (1992).

3.4.3 Multimodalidade

Como sugerido pelo nome, a multimodalidade caracteriza-se pela presença de vários picos e vales, sendo que a maioria desses são ótimos locais. Considerando que cada ótimo local é um atrator da busca, a multimodalidade na otimização de problemas normalmente causa dificuldade em encontrar o ótimo global para qualquer algoritmo de otimização.

Para os AGs não é diferente, a multimodalidade também causa normalmente uma dificuldade para encontrar o ótimo global, pois indivíduos que são ótimos locais aumentam seus descendentes e, com isso, atraem o algoritmo para a região em que se encontram.

Entre as várias funções multimodais, cita-se as funções $f6$, *Rastrigin*, *Ackley* e *Griewangk*. Todas essas são difíceis para os AGs, dado o grande número de ótimos locais e o tamanho dos vales que separam os ótimos. Por isso, também são usadas para testar o desempenho dos algoritmos de otimização.

A função $f6$ é uma versão invertida da função $F6$ de Schaffer (DAVIS, 1991), definida para duas variáveis pela equação (3.3). Esta função possui vários ótimos locais próximos entre si no $\Re^3$ e um único ótimo global em $(x, y) = (0, 0)$ com $f(x, y) = 1$, ver Figuras 24 e 25 . A cada par de ótimos locais, há um ponto de mínimo intercalando-os, por isso um problema difícil para as técnicas mais simples e também para os AGs.

$$f(x, y) = 0,5 + \frac{(\sin\sqrt{x^2 + y^2})^2 - 0,5}{(1 + 0,001(x^2 + y^2))^2} \tag{3.3}$$

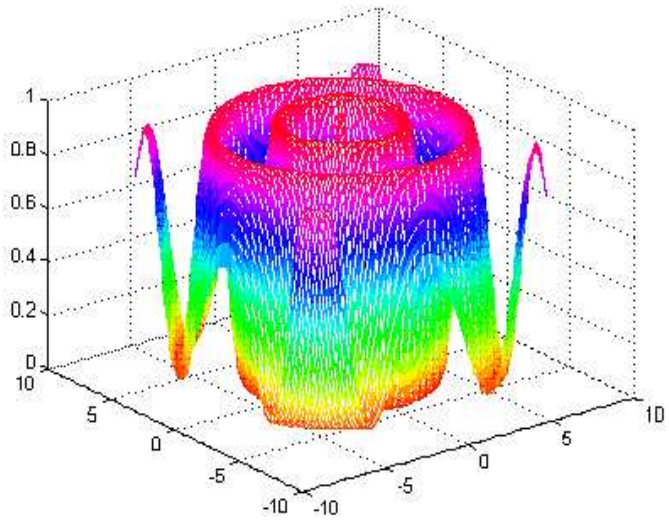


Figura 24: Função multimodal $f6$ (ICA, 2009)

A função *Ackley*, ver Figura 26, a ser minimizada, foi criada inicialmente para duas dimensões (ACKLEY, 1987) mas generalizado para N dimensões por Back (1996). O problema é formulado pela busca de um vetor $\vec{x} = \{x_1, x_2, \ldots, x_N\}$, com $x_i \in [-32.768; 32.768]$, que minimize a equação (3.4). O valor ótimo é o vetor $x^* = (0, ..., 0)$ com $f(x^*) = 0$.

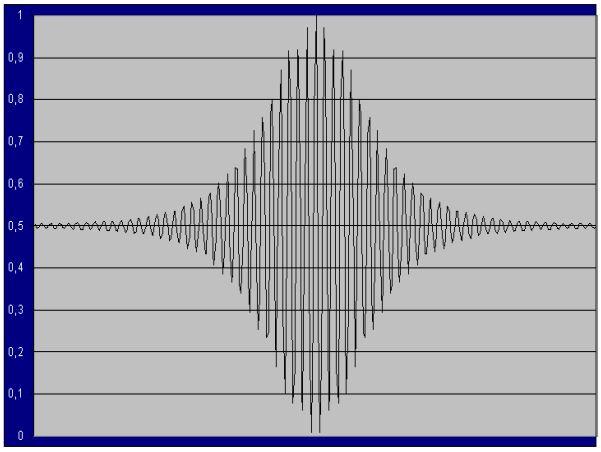


Figura 25: Função multimodal $f6$ em duas dimensões (ICA, 2009)

$$F(\vec{x}) = -20 \cdot exp(-0.2\sqrt{\frac{1}{n} \cdot \sum_{i=1}^{n} x_i^2}) - exp(\frac{1}{n} \cdot \sum_{i=1}^{n} \cos(2\pi x_i)) + 20 + exp(1) \tag{3.4}$$

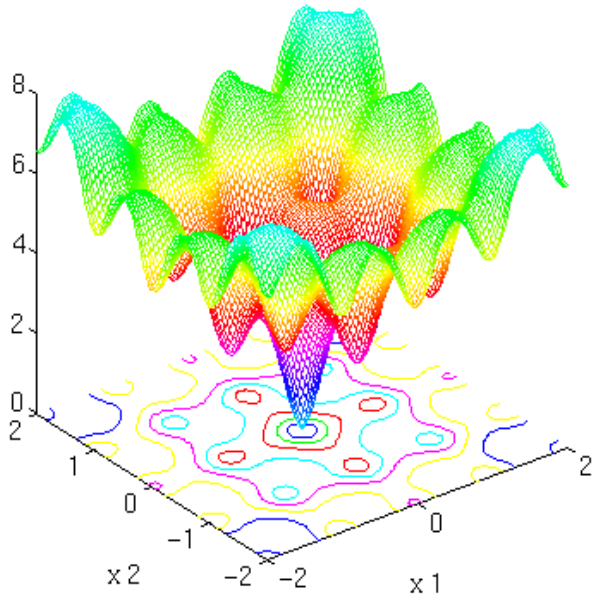


Figura 26: Função multimodal *Ackley* (TRACER, 2009a)

A função *Griewangk*, a ser minimizada, ver Figura 27, foi definida no trabalho Torn e Zilinskas (1989). O problema é formulado pela busca de um vetor $\vec{x} = \{x_1, x_2, \ldots, x_N\}$, com $x_i \in [-600; 600]$, que minimize a equação (3.5).

$$F(\vec{x}) = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}}) \tag{3.5}$$

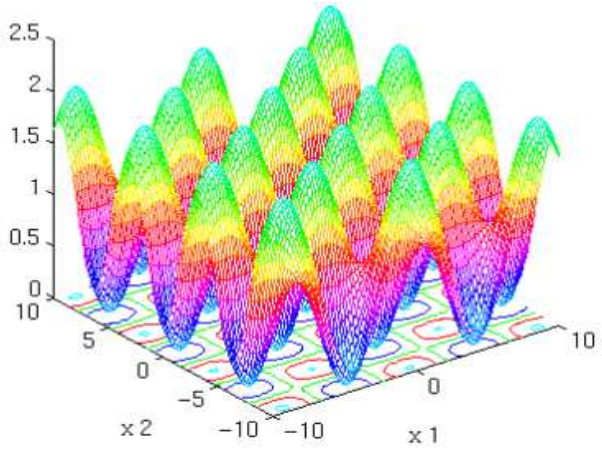


Figura 27: Função multimodal *Griewangk* (TRACER, 2009b)

A função *Rastrigin*, definida pela equação (3.6), foi proposta inicialmente por Rastrigin (TORN; ZILINSKAS, 1989) como um função bidimensional, e posteriormente generalizada por Muhlenbein, Schomisch e Born (1991). A função tem vários ótimos locais, é não linear e tem um grande espaço de busca, por isso considerada uma das funções mais difíceis de otimizar e um excelente *benchmark*. O ótimo global é definido pelo vetor $x^* = (0, ..., 0)$ com $f(x^*) = 0$. A variável A controla a superfície do problema pela amplitude das ondas, ver Figura 28.

$$f(x) = nA + \sum_{i=1}^{n}(x_i^2 - A\cos(2\pi x_i)); \forall i \in [1..n], x_i \in [-5,12; 5,12] \tag{3.6}$$

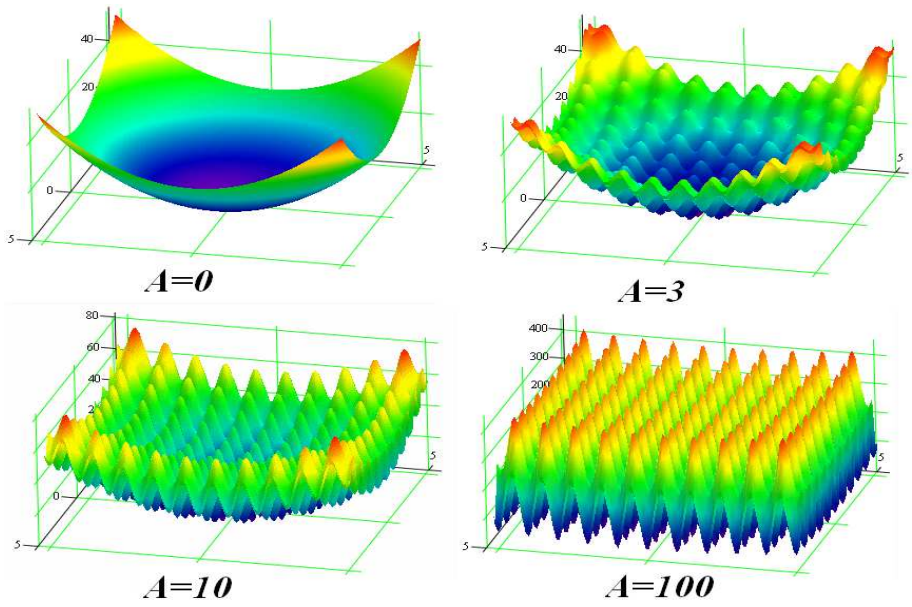


Figura 28: Função multimodal *Rastrigin* (RASTRIGIN, 2009)

4 *Modificações nos AGs*

Dada a quantidade e a importância dos problemas de otimização, alguns algoritmos são desenvolvidos para resolvê-los ou encontrar melhores soluções. Devido ao bom desempenho e à versatilidade, os AGs são muito utilizados para otimização. No entanto, eles apresentam, em alguns problemas, dificuldades de convergência, como por exemplo, convergência prematura, baixa qualidade de solução, e baixa velocidade de convergência, devido ao aspecto degenerativo dos operadores.

Por isso, modificações nos AGs são propostas com o intuito de melhorar a velocidade de convergência e/ou a eficácia do algoritmo. Muitas são as inspirações que norteiam as propostas, já que os autores se baseiam normalmente em modelos existentes para construir as teses. Modelos naturais, como por exemplo o IGA - *Immune Genetic Algorithm* (JIAO; WANG, 2000), ou artificiais, como por exemplo o CMGA - *Cure Mechanism Genetic Algorithm* (WANG et al., 2005), servem de inspiração para os pesquisadores.

Assim, neste capítulo serão apresentados algumas modificações feitas nos AGs para aprimorar a funcionalidade de busca. Entre as modificações, destacam-se: as novas técnicas evolucionárias ou alterações em operadores (seção 4.1), as técnicas de nicho (seção 4.2), as técnicas híbridas (seção 4.3) e o Algoritmo Auxiliar Paralelo, sendo, este último, proposta deste trabalho (seção 4.4).

4.1 Técnicas evolucionárias e novos operadores

Um dos métodos usados para melhorar o desempenho da busca é o uso de conhecimento para direcioná-la, evitando assim a busca cega ou lenta no espaço de busca. Entre os algoritmos que usam do conhecimento estão: os Algoritmos Culturais (REYNOLDS; SVERDLIK, 1994), onde o conhecimento é armazenado em um espaço de crença e usado para promover regiões promissoras; os AGs baseados na imunidade (IGA) (JIAO; WANG, 2000), onde a vacina é considerada um pedaço de informação e a vacinação é a inoculação

(direcionamento) da informação no indivíduo para melhorá-lo; o CMGA (WANG et al., 2005), um AG baseado no mecanismo de cura da medicina chinesa tradicional, que funciona com a montagem dinâmica de um operador de cura que substitui genes normais pelos eugênicos, com a operação "bu", e substitui genes anormais por genes normais, com a operação "xie", guiando assim a evolução em função do conhecimento; os AGs baseados na teoria da terapia genética (GTGA) (WANG et al., 2006), onde o conhecimento é armazenado usando o mecanismo de *gene pool*, que é utilizado pelos operadores da terapia genética para transferir conhecimento e guiar a busca; e o modelo evolutivo humano (HEM) (MONTIEL et al., 2007), que usa o conhecimento de especialistas para inferir os parâmetros mais adequados para alcançar a evolução.

4.1.1 AG baseado nos mecanismos de mudança das dinastias

Baseado também no uso do conhecimento para guiar a busca pelo espaço de soluções, o trabalho Wang et al. (2008) propõe o DCGA (*Dynastic Changes Mechanism Genetic Algorithm*), que se inspira nas mudanças de dinastias das nações para desenvolver os operadores que guiarão a busca.

Historicamente é possível perceber que durante a evolução da sociedade humana as nações tiveram muitas dinastias, e cada dinastia demonstram um comportamento similar. No início, um grande crescimento e prosperidade; já no fim, uma decadência seguida do surgimento de uma nova dinastia. A nova dinastia surgida não era uma simples repetição da anterior, mas sim uma evolução que aprendeu com a experiência da antecessora e guiou-se por esse conhecimento. Percebe-se assim um comportamento cíclico de revezamento entre as dinastias.

Fazendo uma analogia com os AGs, a configuração da dinastia é semelhante à geração da população inicial, o desenvolvimento da dinastia é semelhante ao desenvolvimento da população e a extinção da dinastia é semelhante à convergência da população.

Assim, o DCGA faz uma analogia ao processo de revezamento entre as dinastias. No DCGA o processo evolutivo é dividido em fases e a cada fase evolui-se uma nova população. Quando uma população fixa-se em um ótimo local ela é descartada e uma nova população é criada com o conhecimento adquirido pela população anterior.

Instâncias do problema do caixeiro viajante (TSP) foram usadas para o teste. Os resultados demonstram o bom desempenho do DCGA para evitar a convergência prematura e a estagnação evolucionária nos problemas combinatórios. A Figura 29 ilustra a evolução

das populações do DCGA (WANG et al., 2008). Destacam-se as mudanças de fases, onde há uma queda abrupta da aptidão entre elas, e a tendência de crescimento da aptidão ao longo das fases.

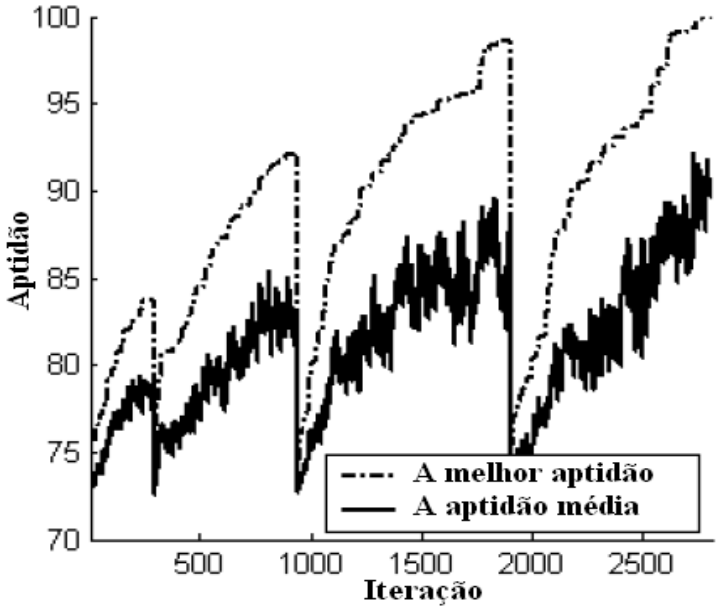


Figura 29: Um exemplo da curva evolutiva do DCGA (WANG et al., 2008)

4.1.2 Operador de Fabricação e Duplicação

Baseado na importância de analisar a inserção de alguns mecanismos biológicos nos AGs, apontada por Mitchell (1998), e a necessidade de criar operadores menos dependentes do problema (MITCHELL; FORREST, 1994), o trabalho Chang, Wang e Liu (2005) apresenta o Operador de Fabricação (OF) e o Operador de Duplicação (OD) para melhorar a taxa de convergência e a qualidade da solução dos AGs (CHANG; WANG; LIU, 2005).

O OD, baseado na tecnologia de clonar, copia o melhor indivíduo para gerar os filhos. Já o OF tem o objetivo de extrair a boa parte da estrutura cromossômica para fabricar novos filhos. Estes operadores são inseridos no AG logo após a seleção, como observado no fluxograma do AG modificado ilustrado na Figura 30.

O OD é um procedimento fácil de implementar e rápido computacionalmente. Similarmente ao Elitismo, o OD copia o melhor indivíduo na população. No entanto, o OD copia o melhor cromossomo em $d\%$ da população, já o Elitismo copia uma única vez. Com o aumento de cópias do melhor indivíduo, cresce a probabilidade da busca ir para um ótimo local. Por isso, os autores indicam o uso moderado das cópias para manter a

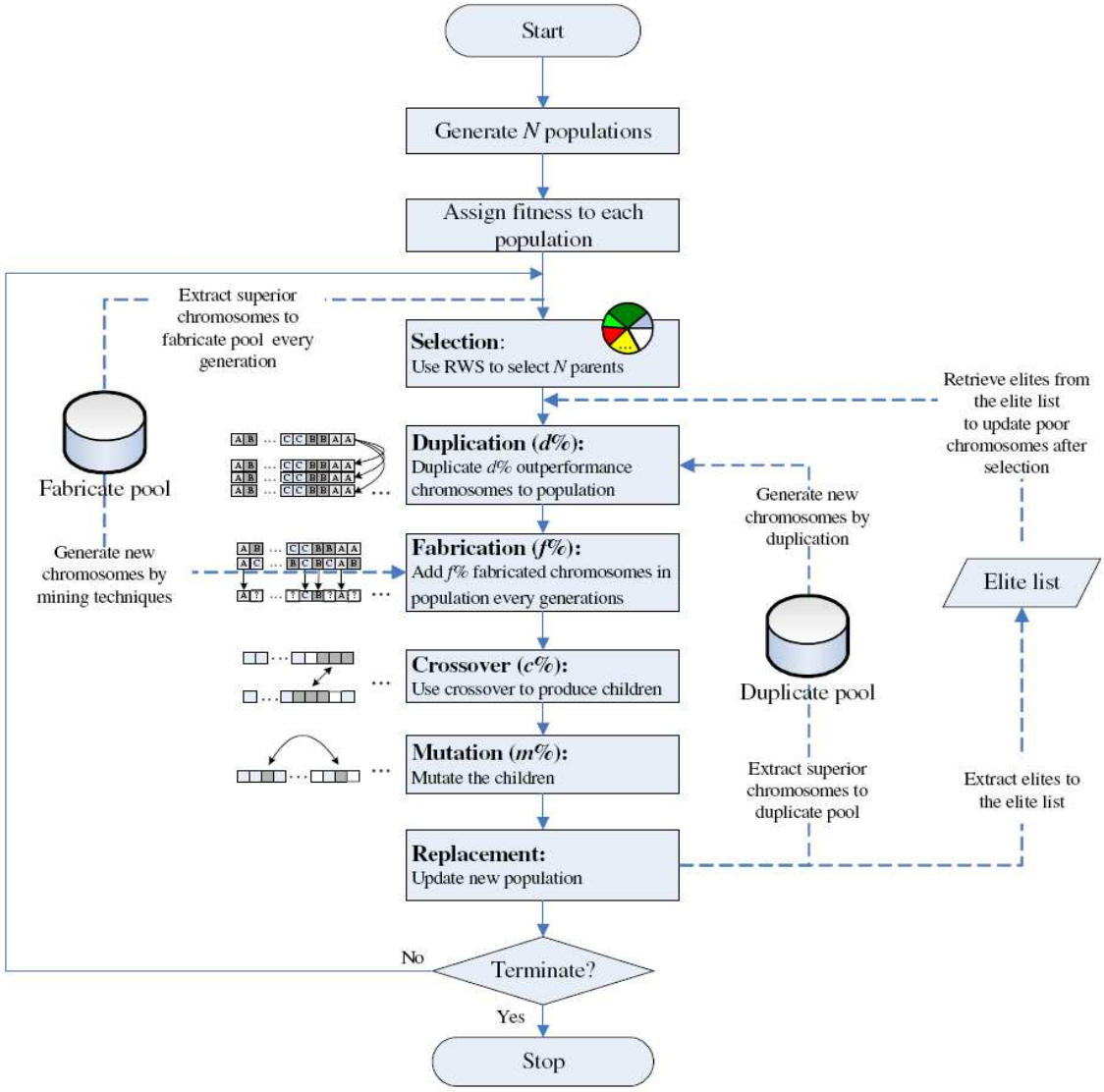


Figura 30: Fluxo do AG com os operadores de Fabricação e Duplicação (CHANG; WANG; LIU, 2005)

diversidade na população.

O OF é um procedimento para construir um conjunto de novos indivíduos baseados nos indivíduos elite. Segundo os autores, existe bastante informações não aproveitadas no conjunto de cromossomos elite. Os AGs tradicionais reutilizam somente 20% dos cromossomo gerados.

Por isso, o OF percorre os genes de cada cromossomo contabilizando a quantidade de vezes que um determinado valor aparece em cada gene. A matriz formada pela contabilização recebe o nome de matriz de dominância M_{ij}. Esse processo é chamado de votação e acontece a cada x iterações, sendo que a matriz é zerada a cada execução. Baseado nessa, fabrica-se o cromossomo artificial pela atribuição do gene com maior frequência

em cada posição. Alguns elementos não são preenchidos, dado a igualdade da frequência, e devem ser definidos em um processo de escolha randômico entre os valores possíveis. Assim, os filhos são fabricados a partir do cromossomo artificial para substituir $f\%$ da população. Um exemplo é apresentado na Figura 31.

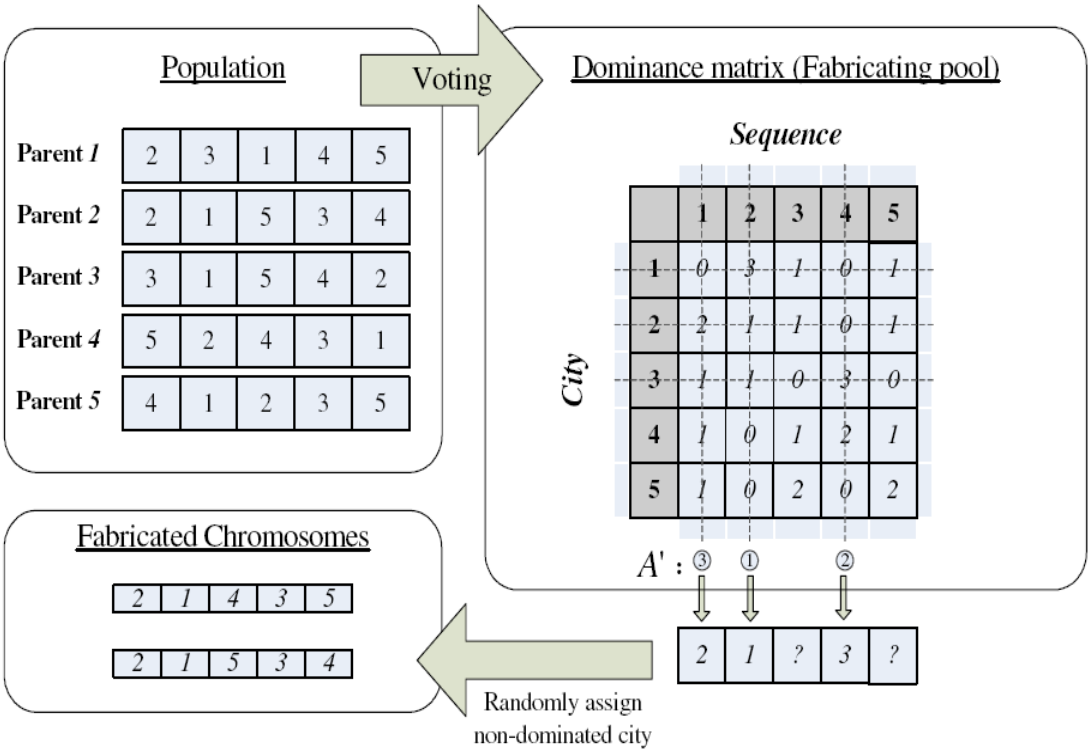


Figura 31: Diagrama do Operador de Fabricação (CHANG; WANG; LIU, 2005)

O princípio de funcionamento do OF é calcular a frequência que os valores aparecem em cada posição dos cromossomos de um grupo elite para fabricar filhos com as melhores informações, nesse caso informações mais repetidas. Esse procedimento recebe o nome de mineração da estrutura genética (*mining gene structure*).

Testes realizados com instância do problema do Caixeiro Viajante e funções contínuas demonstraram que ambos os operadores são boas opções para melhorar o desempenho dos AGs, com destaque para o OF (CHANG; WANG; LIU, 2005).

4.1.3 *Mining gene structures on sub-population genetic algorithm* (MGSPGA)

O MGSPGA é uma proposta de AG feita por Chang, Chen e Liu (2007) para integrar o procedimento MG (*mining gene structure*) com o algoritmo SPGA (*sub-population genetic algorithm*).

O SPGA foi proposto por Chang, Chen e Lin (2005) para melhorar o desempenho dos AGs para resolver problemas de escalonamento multiobjetivo. Testes realizados mostram que o algoritmo tem bom desempenho, principalmente na diversificação da população. No entanto, carece de mecanismos para melhorar a convergência. Por isso, acrescentou-se no SPGA o MG, formando o MGSPGA.

Para melhor adaptar o MG ao SPGA, propõem-se (CHANG; CHEN; LIU, 2007) dois novos métodos: o *weighting mining gene structures* (WMGS) e o *threshold mining gene structures* (TMGS).

Baseado nos testes, o método TMGS foi o melhor entre os métodos e, por isso, foi escolhido para as comparações com os algoritmos NSGA-II e SPEA-II, dois algoritmos muito usados em problemas multiobjetivos. Testes com o problema *Flow Shop*, instâncias de 20, 40, 60 e 80 tarefas em 20 máquinas, mostram que o MGSPGA (com o método TMGS) obtém desempenho superior aos outros algoritmos e que o incremento do MG ao SPGA traz benefícios, principalmente quando o problema é mais complexo.

4.1.4 *Artificial Chromosome Generating Mechanism* (AC)

Seguindo a ideia da mineração da estrutura genética (MG), o trabalho Chang, Chen e Fan (2008) propõe o AC para o problema de escalonamento de máquina única.

Os AGs, após várias iterações da evolução genética, convergem os cromossomos vagarosamente em certas distribuições dos genes. Analisando a distribuição de cada gene em cada posição percebe-se que a maioria dos genes convergem para um certo local, o que significa que o gene pode ser alocado em uma posição baseando-se em uma matriz de probabilidade, que guiará o processo de alocação (CHANG; CHEN; FAN, 2008).

Assim, o AC é apresentado como um mecanismo que, baseado na matriz de probabilidade e dominância, melhora o desempenho dos AGs. A matriz de probabilidade é gerada a partir da transformação da matriz de dominância e guia a atribuição do gene na posição.

O operador AC pode ser descrito basicamente em três passos:

1. Calcular a aptidão média da população e converter as informações genéticas na matriz de dominância. Ao invés de trabalhar com toda a população para gerar a matriz, seleciona-se os cromossomos que têm a aptidão acima da média da população, formando assim o grupo elite. Posteriormente, percorre-se todos os cromossomos do grupo elite formando a matriz de dominância M_{ij}, ver equação (4.1). Sendo $X_{ij} = 1$

quando existir o gene i na posição j, ou $X_{ij} = 0$ quando não existir.

$$M_{ij}(t) = \sum_{k=1}^{n} X_{ij}^{k} \tag{4.1}$$

2. Gerar o cromossomo artificial. Após gerar a M_{ij}, aloca-se os genes nas posições de cada cromossomo artificial. Para tal, calcula-se a matriz de probabilidade pela equação (4.2).

$$P_{ij}(t) = \frac{\sum_{k=1}^{n} X_{ij}^{k}}{N} \tag{4.2}$$

   onde $P_{ij}(t)$ é a probabilidade do gene i ser atribuído à posição j.

   A ordem de atribuição das posições é definida randomicamente, para dar uma maior diversidade aos cromossomos artificiais.

3. Definir os sobreviventes para a próxima geração. Usa-se a estratégia $\mu + \lambda$, onde μ são os cromossomos pais e λ os cromossomos artificiais gerados pelo passo anterior. Após a união, seleciona-se os μ melhores do conjunto $\mu + \lambda$. Formando assim a nova população μ.

A integração do AC ao AG (ACAG) se dá pela substituição dos operadores tradicionais pelo AC a cada k iterações, diminuindo o risco de convergência prematura e o custo computacional. Como o AC depende de boas informações oriundas do grupo elite e auxilia o processo de convergência final, é necessário algumas iterações para que o grupo possa ser formado. Assim, o AC inicia a execução após *startingGen* iterações. O fluxograma, ver Figura 32, ilustra o ACAG.

Testes realizados com diferentes instâncias do problema de escalonamento de máquina única demonstram o bom desempenho da junção do AC no AG (ACAG) quando comparado ao AG. Destaca-se como uma vantagem do AC o fato de não trabalhar com população inicial melhorada por alguma heurística (CHANG; CHEN; FAN, 2008).

Instâncias do problema de *flowshop* foram aplicadas ao ACAG no trabalho Chang et al. (2008). Os resultados confirmaram o bom desempenho do ACAG comparado ao AG, melhorando significativamente a velocidade de convergência e a qualidade do resultado. Esse mesmo trabalho apresenta resultados dos testes feitos com ACNSGA-II (junção do AC ao NSGA-II) aplicado ao problema de *flowshop* multiobjetivo. Várias instâncias foram analisadas e o ACNSGA-II teve desempenho um pouco inferior ao NSGA-II em instâncias pequenas e desempenho melhor em instâncias mais complexas. Nessas, o ACNSGA-II

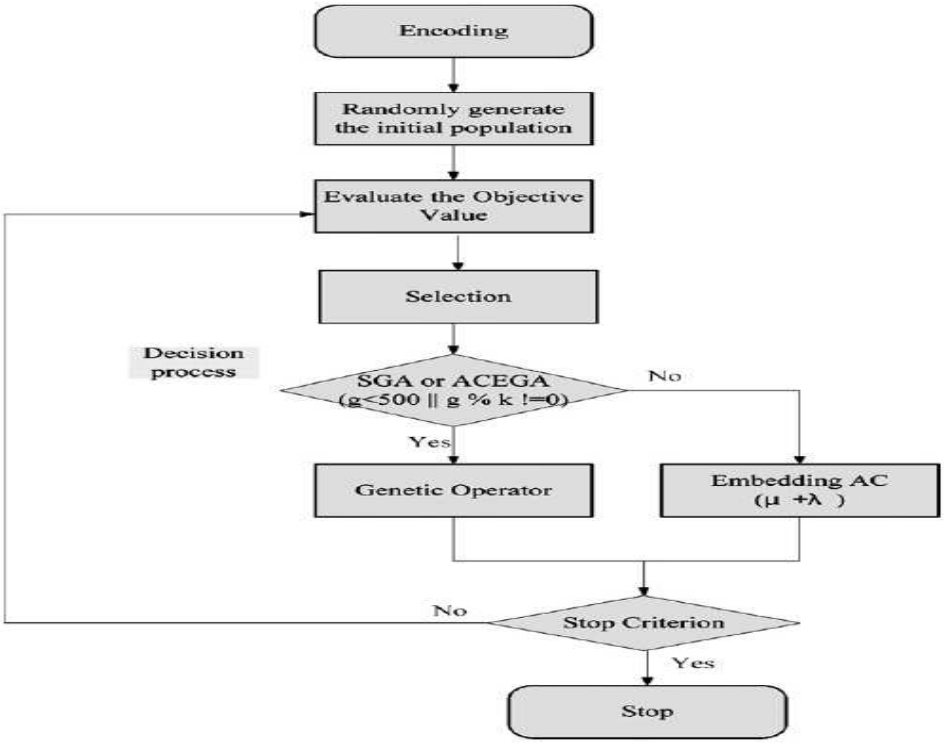


Figura 32: Fluxograma do ACAG (CHANG; CHEN; FAN, 2008)

melhorou a velocidade de convergência do NSGA-II e obteve um pequena melhoria da qualidade da solução.

4.2 Técnicas de Nicho

Define-se como nicho natural uma parte de um *habitat* onde são estabelecidas algumas condições especiais de um ecossistema e as espécies que nele vivem disputam seus recursos. A forma com que cada espécie se adequa e desenvolve, de acordo com o nicho em que vive, forma as subpopulações. O número de indivíduos em um nicho é determinado pela quantidade de recurso disponível no nicho e pela eficácia do indivíduo aproveitar esses recursos (FAN; ZENG; LI, 2009; JELASITY; DOMBI, 1998). A Figura 33 ilustra o nicho de alguns carnívoros.

Análogo ao sistema natural, podem-se definir nichos e subpopulações para os AGs, onde parcelas da população são evoluídas em condições diferentes, dando maior amplitude na busca e maior diversificação. A conservação da diversidade genética e a capacidade de manter representantes de boas soluções (ótimos locais) durante a busca são as principais características dos AGs com técnica de nicho.

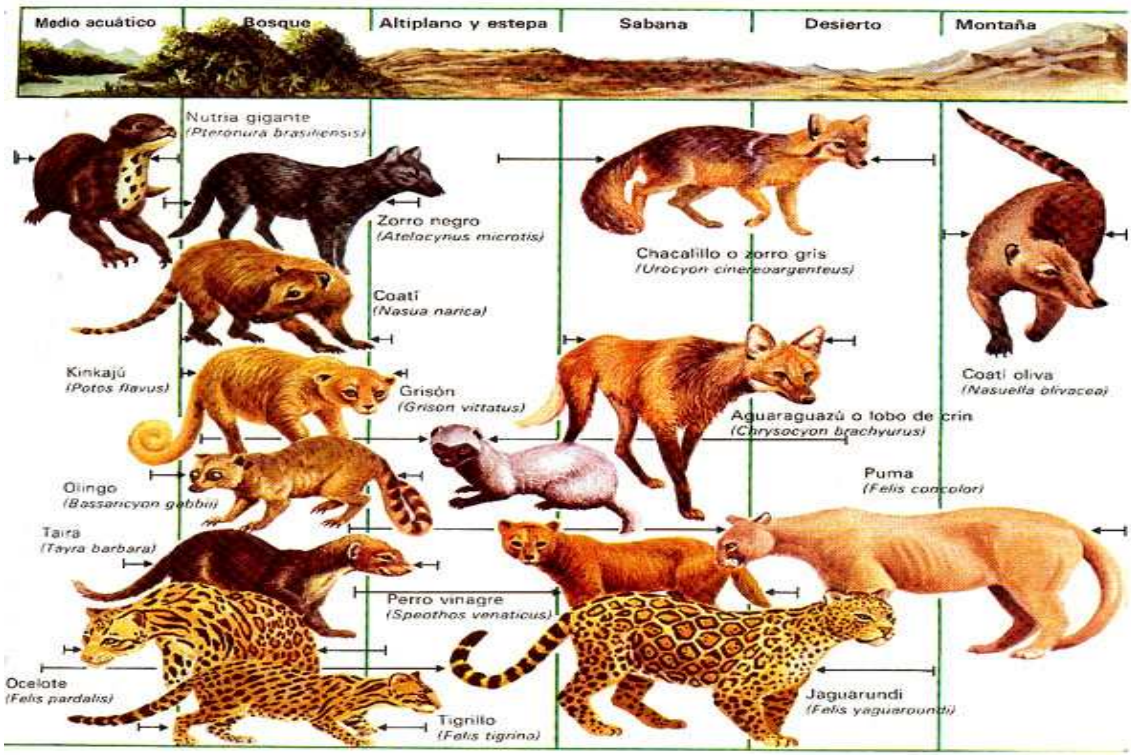


Figura 33: Nicho de alguns carnívoros (FELIX, 2010)

Os AGs sem nicho evolui de forma a convergir a busca para uma única área promissora, eliminando durante o processo bons indivíduos de outras regiões dos espaço de busca. Em problemas, como ilustrado na Figura 34, pode-se querer encontrar mais de uma solução e, por isso, é necessário que o algoritmo não descarte boas soluções das outras regiões de picos. Para tal, faz-se necessário a introdução da técnica de nicho, reduzindo a tendência genética para uma única região e permitindo a exploração e manutenção em outras regiões.

Dessa forma, as técnicas de nicho tentam manter a sobrevivência dos indivíduos que estão fora do grupo elite. Para tal, cada região de ótimo local, como exemplo os picos da Figura 35, é considerada um nicho com recursos que deverão ser repartidos com os indivíduos que nele se encontram. Assim, quanto mais indivíduos menor são os recursos, obrigando a migração de alguns para outras regiões ou a eliminação desses.

O nicho e as subpopulações podem ser implementadas de várias formas. Basicamente, calcula-se a distância entre os indivíduos, para identificar os vizinhos e os competidores, por exemplo com a função de partilha, e posteriormente, estabelece-se as ponderações, por exemplo no operador de seleção com a aptidão aparente, para determinar os melhores de cada região promissora do espaço de busca e penalizar os vizinhos próximos dos melhores (MING; NAN; XU, 2009).

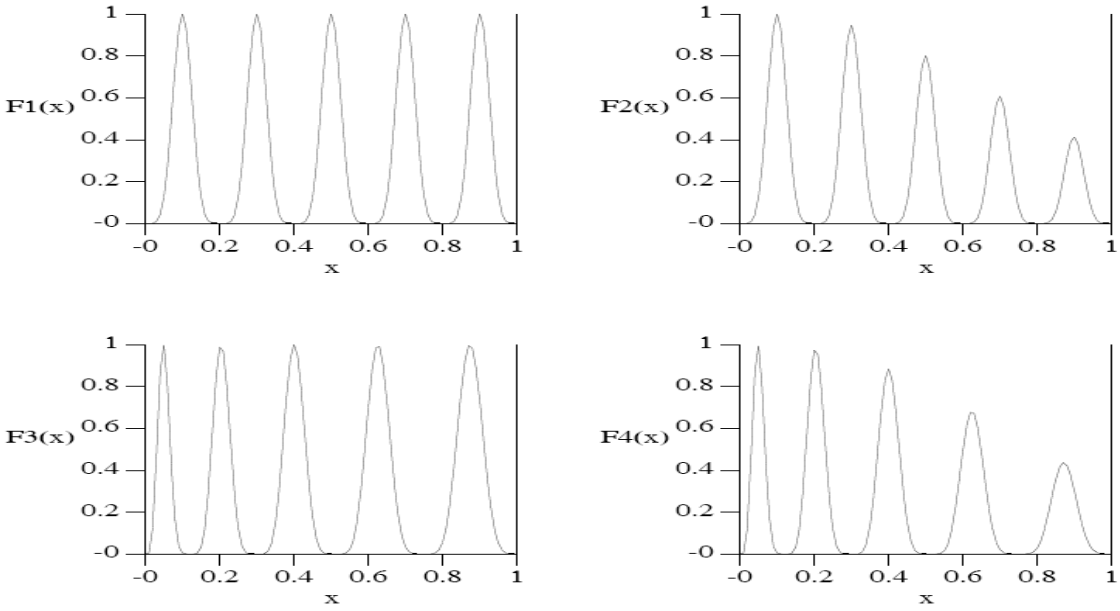


Figura 34: Funções-teste de Spears (1994)

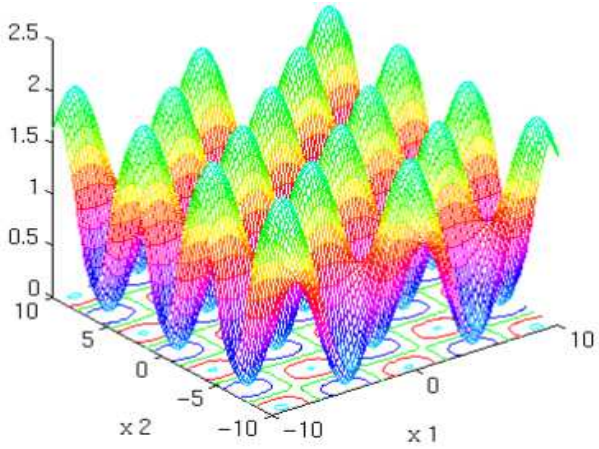


Figura 35: Função multimodal *Griewangk* (TRACER, 2009b)

Como exemplo de cálculo da aptidão aparente, a equação (4.3) define a função aparente usada na técnica de nicho de Goldberg (GOLDBERG; RICHARDSON, 1987; GOLDBERG, 1989). A aptidão aparente é obtida pela divisão da aptidão do indivíduo f_i pelo grau de vizinhança total do indivíduo i em relação a toda a população, com n_{pop} membros (HORN, 1997).

$$f_{s,i} = \frac{f_i}{\sum_{j=1}^{n_{pop}} s(d_{ij})} \tag{4.3}$$

onde s é a função de partilha e d_{ij} é a distância entre os indivíduos i e j.

Restrições nos cruzamentos também são feitas, como por exemplo o cruzamento so-

mente entre indivíduos da mesma subpopulação ou de subpopulações vizinhas. A técnica de nicho de Spears (SPEARS, 1994) usa etiquetas para identificar os indivíduos de cada subpopulação, permitindo uma separação da população e restrições no cruzamento (SS1 e SS2). Essa classificação por etiquetas apresenta-se como uma solução de baixo custo computacional.

Considerando os diferentes tipos de nicho, o trabalho Mahfoud (1995) apresenta uma classificação das técnicas, ilustrada na Tabela 5. Nesse, o autor classifica os algoritmos segundo as dimensões tempo versus espaço e ambiente único versus múltiplos ambientes.

Tabela 5: Classificação das técnicas de nicho

	Ambiente único	Múltiplos ambientes
Temporal	Sequential Location	Overspecification Ecological GAs
Espacial	Heterozygote Advantage Crowding Restricted Competition Fitness Sharing	Ecological GAs Immune Systems

Já o trabalho Li, Lin e Kou (2009) classifica os algoritmos de nicho como de estratégias implícitas ou explícitas. Como exemplo de técnicas explícitas, citam-se: *individual clustering based sharing schemes*, *the species conservation GAs*, *species conservation particle swarm optimizer*, *the restricted competition and mating*, *the sequential niching* e *the island model GAs and the parallel GAs*. Já como exemplo de técnicas implícitas, citam-se: *the crowding and preselection*, *the fitness sharing* e *crowding with nearest neighbors replacement*.

As técnicas de nicho podem ser aplicadas a vários tipos de problemas, entre eles: classificação e aprendizado de máquina, otimização de função multimodal, otimização de função multiobjetivo e simulação de sistemas complexos e adaptativos (MAHFOUD, 1995).

Dados os resultados apresentados na literatura (HORN, 1997; SARENI; KRAHENBUHL, 1998; YUAN; LI; LI, 2008b; HUANG; TIAN; FANG, 2009; MING; NAN; XU, 2009), podem-se afirmar que as técnicas de nicho são opções eficazes para a manutenção da diversidade genética e para busca e permanência de soluções de boa qualidade. Destaque para o trabalho de Li, Lin e Kou (2009) que apresenta uma comparação de diferentes tipos de técnicas de nicho, baseadas na estratégia de substituição da população, para as funções multimodais. Entre os algoritmos testados: *Nearest Neighbors Replacement Crowding* (NNRC), *Species Conservation Technique* (SCT), *Implicit Hierarchical Fair Competition* (HFC-I) e o *Clearing Procedure with Elitist* (CPE), o NNRC demonstra a melhor perfor-

mance.

Como exemplo da aplicação do NNRC no *steady-state* AG (SSAG), a Figura 36 ilustra o efeito do NNRC na representação do espaço de busca da função Schaffer, que é multimodal, tem vários ótimos locais e tem os picos distribuídos em forma de anel.

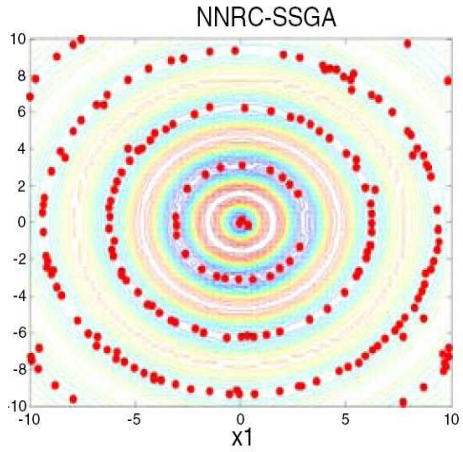


Figura 36: Efeito da aplicação do NNRC na função Schaffer Li, Lin e Kou (2009)

4.3 Técnicas Híbridas

Os AGs são algoritmos muito versáteis, pois têm uma estrutura modular que permite acoplamento de outras técnicas, como métodos determinísticos, outras heurísticas, algoritmos nebulosos, redes neurais, entre outros. Esse acoplamento, que forma a estrutura híbrida, objetiva aproveitar o melhor de cada técnica para melhorar o desempenho do algoritmo híbrido e, assim, obter melhores resultados.

Das várias formas de construir um algoritmo híbrido (ver seção 2.11), essa seção apresenta o GA-EDA (ver seção 4.3.1), que incorpora modelos probabilísticos nos AGs, e o AG-BT (ver seção 4.3.2), que une a diversificação do AG e a intensificação do BT.

4.3.1 GA-EDA

O uso de modelos probabilísticos nos algoritmos genéticos gerou um novo paradigma na Computação Evolucionária chamado EDA. Essa é uma área crescente e com bons resultados na literatura (LARRANAGA; LOZANO, 2001). Baseado nisso, o algoritmo GA-EDA é uma proposta do trabalho Pena et al. (2004), que une os AGs com os Algoritmos de Estimativa de Distribuição (*estimation of distribution algorithms* - EDA), ambas heurísticas de base populacional, com o intuito de aproveitar o melhor de cada técnica.

O GA-EDA tem, como principal diferencial, a forma de gerar os filhos e a definição da população sobrevivente. Nessa técnica dois grupos de filhos são gerados, sendo um grupo gerado pelo AG e o outro grupo gerado pelo EDA. O filhos gerados pelo AG são frutos dos operadores de cruzamento e mutação. Já o EDA gera modelos probabilísticos baseados nos melhores indivíduos e extrai do modelo alguns filhos artificiais.

A população produzida ($Population_{p+1}$) é a união dos filhos do AG com os filhos artificiais do EDA e a população corrente ($Population_p$). Por fim, aplica-se um filtro na população ($Population_{p+1}$) de forma que os melhores indivíduos são selecionados como sobreviventes para a próxima geração. A Figura 37 ilustra o GA-EDA.

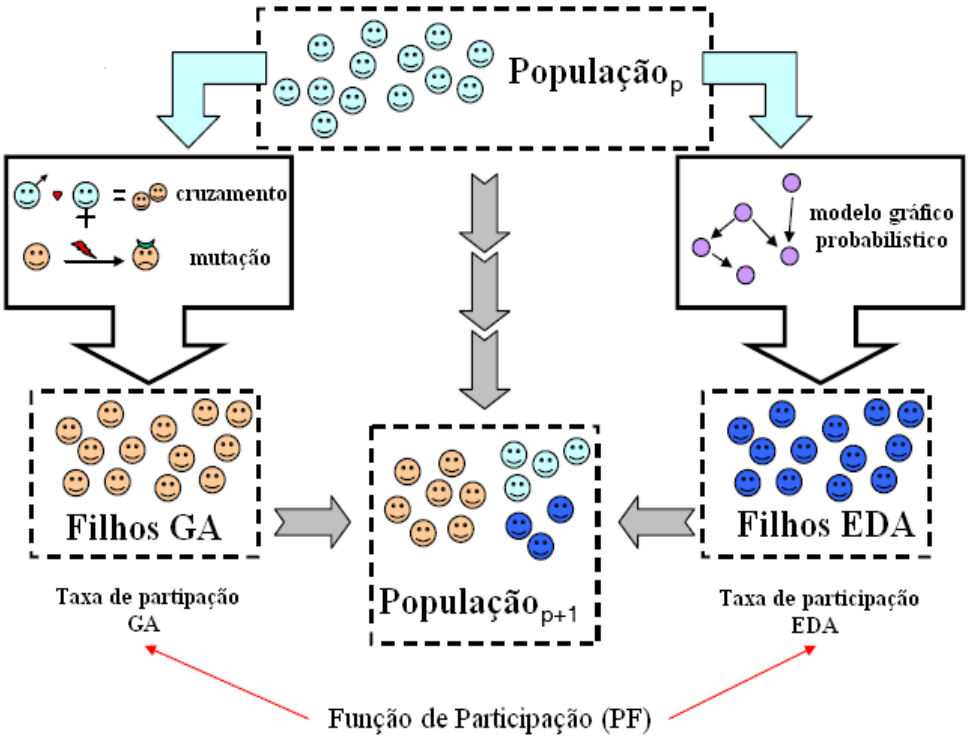


Figura 37: Esquema do GA-EDA (PENA et al., 2004)

O GA-EDA tem um parâmetro adicional chamado PF (função de participação). A PF é uma taxa que define a quantidade de indivíduos gerados pelas técnicas e, consequentemente, como as técnicas irão interferir na evolução. Apesar da PF definir o tamanho do conjunto de filhos de cada técnica, ela não determina a quantidade desses filhos que irão sobreviver para a próxima geração. A definição de sobreviventes, como já citado, é baseado na escolha determinística dos melhores.

Com o PF é possível controlar a participação de cada técnica e, assim, favorecer o mecanismo que esteja gerando melhores descendentes. Para definir os valores de PF existem alternativas como: Taxa Constante, Taxa Alternada, Taxa Incremental, Taxa

Dinâmica e a Taxa baseada em Faixa. Essa última é a melhor entre elas, segundo o trabalho Robles et al. (2005).

Alguns testes foram realizados com o AG canônico e o *Univariate Marginal Distribution Algorithm* (UMDA), como EDA. Nesses testes foram gerados o dobro de indivíduos da população, sendo que os melhores foram escolhidos como sobreviventes para a próxima geração. Foram aplicados problemas de otimização de função e combinatoriais (TSP e MaxBit). Os resultados demonstram que o híbrido é promissor e eficaz (PENA et al., 2004; ROBLES et al., 2005).

4.3.2 AG-BT

A Busca Tabu (BT) é um método heurístico, originalmente proposto por Glover (1986), que vem sendo aplicado a diversos problemas de busca e otimização, principalmente em problemas combinatórios. Em muitos casos, tem apresentado-se como uma ferramenta eficiente e uma das mais eficazes, tornando-a uma das mais populares na literatura.

O BT é um método de busca local, por isso, busca soluções melhores a partir de uma única solução factível inicial s_0. A busca é feita com movimentos m em um subconjunto V da vizinhança $N(s)$ da solução s. Define-se como movimentos as alterações na solução base s que levam a uma outra solução vizinha s'. O movimento, ou um conjunto M desses, deve possibilitar alcançar qualquer ponto do espaço de busca. Assim, pode-se representar a operação de movimento pela expressão $s' \leftarrow s \oplus m$, sendo $m \in M$ (SOUZA, 2009). O tamanho da vizinhança pode ser definido por um raio, caracterizado pela restrição no movimento.

Para superar os ótimos locais, o BT usa uma pequena memória definida como Lista Tabu (LT), a lista de movimentos proibidos. A LT clássica tem os movimentos reversos aos últimos $|T|$ movimentos realizados, onde $|T|$ define seu tamanho. Assim, na exploração do subconjunto de vizinhos V, ficam excluídos da busca os vizinhos s' que são obtidos por movimentos $m \in T$ a partir de s.

Dado o bom desempenho do BT como busca local (onde o foco é a *exploitation*) e dos AGs como busca populacional, o híbrido AG-BT é teoricamente uma união de elementos que proporcionam uma melhoria na busca. Na prática, vários trabalhos usam o híbrido como uma técnica mais eficiente para resolver os mais variados tipos de problema (SENTINELLA; CASALINO, 2009; ZHANG; SHI; GAO, 2008; YANG; ZHANG; BAI, 2008).

Os AGs podem ser acrescidos do BT de várias formas (MAK; SUN, 2009; YANG; ZHANG; BAI, 2008; ZDÁNSKÝ; POZIVIL, 2002), entre elas citam-se o uso de operadores baseados em BT; a execução do BT para cada indivíduo inicial do AG, com o intuito de iniciar os AGs com uma boa população; a execução do BT após n iterações do AG sem melhorias; e a execução do BT sobre n melhores indivíduos a cada iteração do AG, para intensificar a busca local em torno dos melhores.

Entre as opções, este trabalho aborda a execução do BT sobre o melhor indivíduo a cada n iterações do AG, ver Figura 38. Espera-se com essa forma de hibridismo intensificar a busca local com o BT em torno do melhor indivíduo do AG a cada n iterações. O uso de intervalo de n iterações possibilita a minimização do custo computacional, já que o acréscimo do BT acarreta no aumento de procedimentos e avaliações de pontos do espaço de busca.

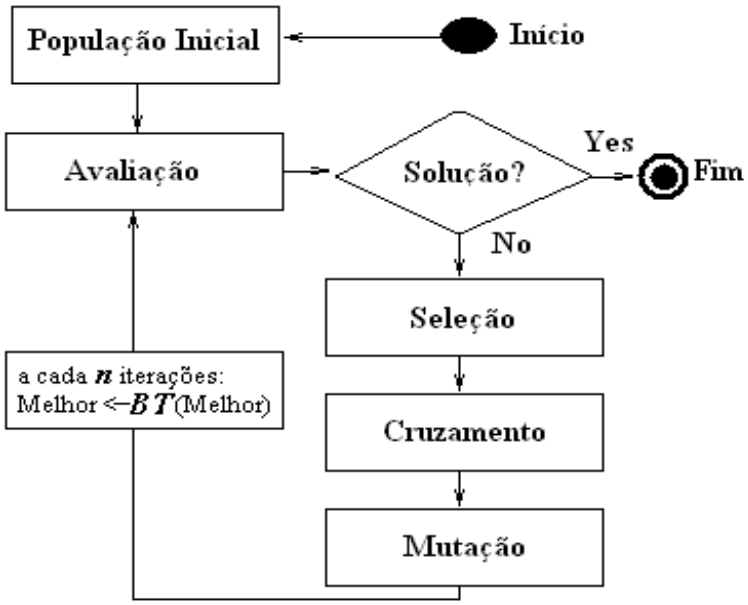
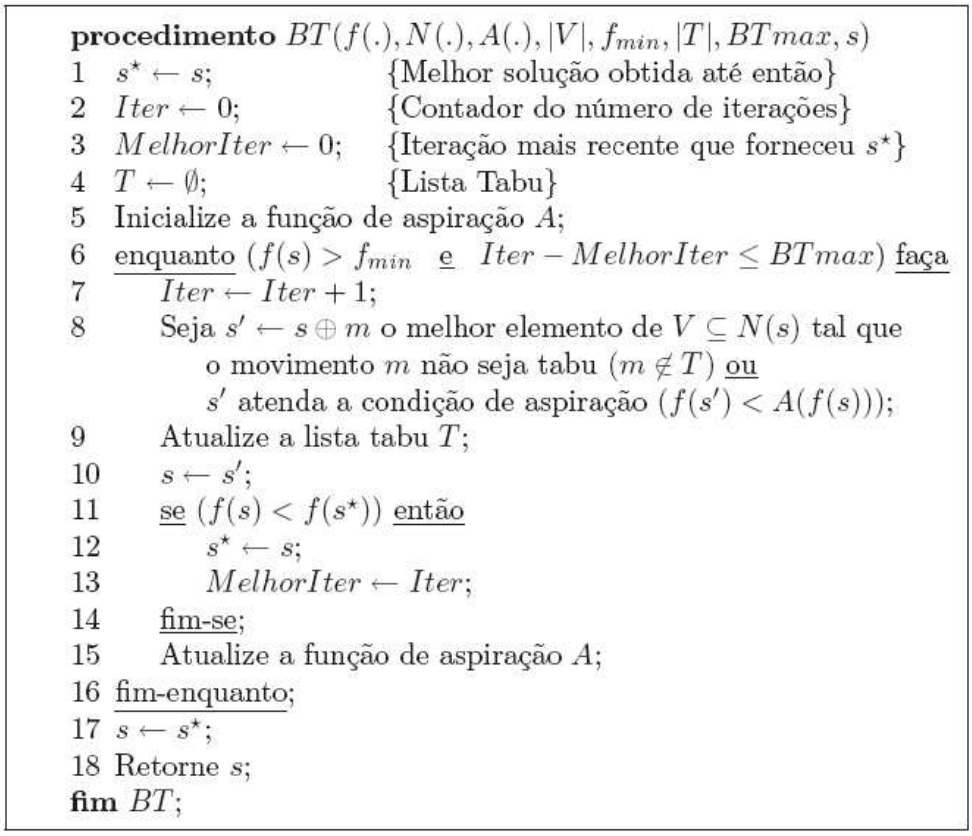


Figura 38: Fluxo do AG-BT

O BT abordado neste trabalho (AG-BT), e utilizado nos experimentos do capítulo 5, segue o procedimento ilustrado na Figura 39, com LT fixa e tamanho máximo $|T|$. A função de aspiração é a por objetivo global, onde o movimento tabu que gera uma solução melhor que a melhor corrente é aspirado da LT. O critério de parada é pelo número de iterações sem melhora no valor da melhor solução.



procedimento $BT(f(.), N(.), A(.), |V|, f_{min}, |T|, BTmax, s)$
1 $s^{\star} \leftarrow s$; {Melhor solução obtida até então}
2 $Iter \leftarrow 0$; {Contador do número de iterações}
3 $MelhorIter \leftarrow 0$; {Iteração mais recente que forneceu $s^{\star}$}
4 $T \leftarrow \emptyset$; {Lista Tabu}
5 Inicialize a função de aspiração A;
6 enquanto $(f(s) > f_{min}$ e $Iter - MelhorIter \leq BTmax)$ faça
7 $Iter \leftarrow Iter + 1$;
8 Seja $s' \leftarrow s \oplus m$ o melhor elemento de $V \subseteq N(s)$ tal que
o movimento m não seja tabu $(m \notin T)$ ou
s' atenda a condição de aspiração $(f(s') < A(f(s)))$;
9 Atualize a lista tabu T;
10 $s \leftarrow s'$;
11 se $(f(s) < f(s^{\star}))$ então
12 $s^{\star} \leftarrow s$;
13 $MelhorIter \leftarrow Iter$;
14 fim-se;
15 Atualize a função de aspiração A;
16 fim-enquanto;
17 $s \leftarrow s^{\star}$;
18 Retorne s;
fim BT;

Figura 39: O algoritmo Busca Tabu para minimização (SOUZA, 2009)

4.4 Algoritmo Auxiliar Paralelo

Analisando os problemas apresentados na seção 3.3, da perda e do baixo aproveitamento das informações presentes nas populações dos AGs, este trabalho propõe o Algoritmo Auxiliar Paralelo (AAP) que, baseado na população corrente dos AG e em novos indivíduos gerados, recombina os cromossomos, para melhor aproveitar as informações, e seleciona os melhores indivíduos de forma análoga à Fertilização in Vitro.

A Reprodução Assistida (RA) (LEITE, 2009) é um conjunto de técnicas de reprodução onde a concepção de uma vida é manipulada e assistida por especialistas a fim de garantir o sucesso do processo. Dentre as técnicas da RA destacam-se duas para este trabalho: à Fertilização in Vitro (FIV) e a *Preimplantation Genetic Diagnosis* (PGD). A FIV é uma técnica que retira óvulos do ovário da mãe e os fertiliza em laboratório com espermas preparados do pai, formando os pré-embriões. Após manipulação, de dois a quatro pré-embriões são selecionados e transferidos para a mãe. As fases da FIV tradicional são ilustradas na Figura 40.

O primeiro bebê humano nascido pela técnica de FIV foi promovido por Steptoe e

Edwards em 1978 (ver Figura 41), após anos de estudos em reprodução assistida em mamíferos. De 1978 até os dias atuais, muitas pessoas foram geradas pela técnica de FIV, que é indicada para casos de infertilidade, seja por obstrução tubária bilateral ou outra causa de infertilidade não solucionada com técnicas mais simples.

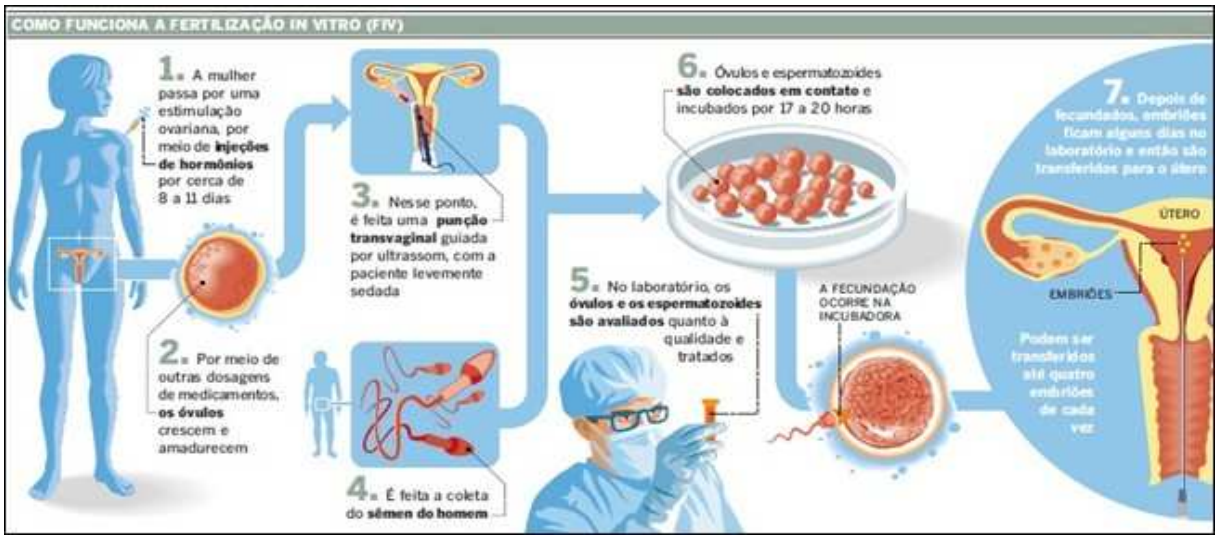


Figura 40: O processo da FIV tradicional (LACERDA, 2009)


Evening News

Meet Louise, the world's first test-tube arrival

SUPERBABE

Wide-eyed Louise Brown pictured in hospital 18 hours after she was born. Today she's doing well. See Page Three

Figura 41: Louise Brown, o primeiro ser humano concebido pela FIV (PIRES, 2009)

A PGD (KAKOUROU et al., 2007; HARPER; RODECK, 2002) é uma técnica desenvolvida como alternativa ao diagnóstico pré-natal para casais com risco de transmissão de doenças genéticas para os filhos. A PGD segue a rotina da FIV, pois os embriões são produzidos fora do corpo da mãe; no entanto, antes da transferência, os embriões passam por uma análise genética (diagnóstico genético) para descobrir possíveis variações genéticas causadoras de doenças. Assim, somente os bons embriões, ausentes de variações prejudiciais, são implantados na mãe.

São usados dois métodos para fazer a análise genética nos embriões: FISH - *Flourescent In Vitro Hybridization* (Figura 42) e a PCR - *Polymerase Chain Reaction* (Figura 43). Com esses métodos, já é possível detectar 130 doenças, tais como: trissomia do cromossomo 13, Síndrome de Edwards, Síndrome de Down, Síndrome de Turner, Distrofia Muscular Progressiva, Fibrose Cística, entre outras (IPGO, 2009). Como exemplo de aplicação da PGD, recentemente nasceu na Inglaterra o primeiro bebê livre da variação causadora do câncer de mama ou de ovário, selecionado pela técnica PGD (CNNHEALTH.COM, 2009).

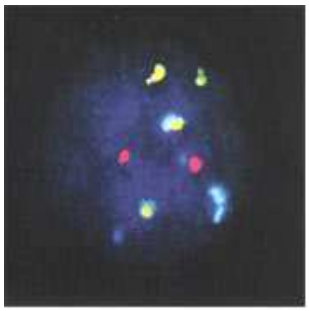


Figura 42: Exemplo de resultado obtido do FISH (PRó-EMBRYO, 2009)

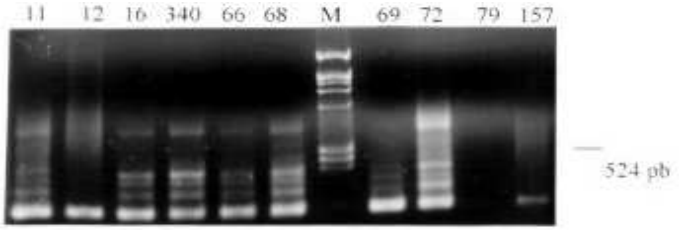


Figura 43: Exemplo de resultado obtido do PCR (CALDAS et al., 2000)

Análogo à Reprodução Assistida, especificamente a Fertilização in Vitro (FIV) e a *Preimplantation Genetic Diagnosis* (PGD), o AAP avalia várias possibilidades (combinações) e seleciona os genes do Pai e da Mãe que irão compor o novo indivíduo. A manipulação genética na FIV é feita com os gametas dos Pais ou doadores; já no AAP são os genes do melhor indivíduo (o Pai) e de outros indivíduos (as Mães). Após seleção do material e a fecundação, o novo indivíduo gerado é avaliado para verificar sua aptidão; no caso da FIV/PGD há uma seleção dos melhores embriões a serem implantados. Caso haja bons indivíduos, ou embriões no caso da FIV/PGD, o material produzido, de forma assistida, é introduzido na população.

Para tal, o AAP é executado em um fluxo paralelo ao dos AGs, recebendo como entrada uma parcela da população corrente e emitindo como saída um indivíduo ou mais, melhor que o melhor corrente (Figura 44). Desta forma, o AAP recebe e analisa uma boa quantidade de informações, que poderia ser perdida, recombina-as e abastece o AG

de bons indivíduos, agilizando assim o processo evolutivo dos AGs.

Como a interferência do AAP na população é feita com poucos indivíduos, mantém-se as características de diversidade e aproveitamento produzidas pelo AG clássico e evita-se que a otimização da perda da informação prejudique o aproveitamento da informação.

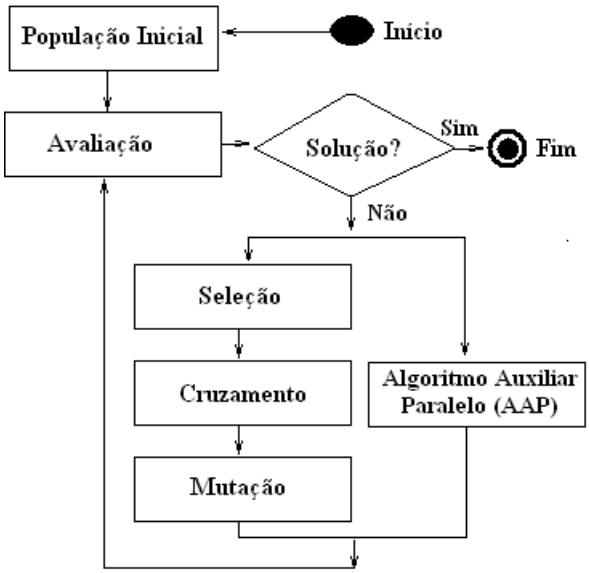


Figura 44: O acoplamento do AAP ao AG

Como ilustrado na Figura 45, o AAP pode ser dividido em três fases sequenciais:

Coleta A forma ideal de aproveitar todas as informações presentes na população seria recombinar os indivíduos gene a gene até encontrar a melhor combinação, o que seria inviável computacionalmente, principalmente para cromossomos maiores. Por isso, optou-se por trabalhar com uma parcela da população (N indivíduos), captada por alguma estratégia de coleta de material genético. A estratégia usada neste trabalho é a coleta dos N melhores indivíduos da população (porcentagem dos melhores), sendo que o melhor é considerado o Pai e os demais as Mães. Outras estratégias como a coleta de sub-populações, porcentagem da população, porcentagem da população com o melhor indivíduo, entre outras, podem ser usadas para geração de matéria-prima utilizada na próxima fase, manipulação genética.

Manipulação Genética Após a seleção do material genético de entrada, aplica-se a Manipulação Genética para avaliar, alterar e recombinar as estruturas selecionadas. Para tal, essa fase pode ser dividida em duas operações, sendo a primeira de Alteração Genética, onde parte de alguns cromossomos podem ser alterados, e a segunda de Recombinação Genética, onde as combinações das informações são feitas e avaliadas (ver seção 4.4.4).

Transferência Caso o processo gere melhores indivíduos do que o melhor indivíduo corrente, esses são transferidos para a população do AG. Uma das estratégias de transferência, e a usada neste trabalho, é inserí-los em substituição aos melhores correntes (estratégia elitista). Na ausência do elitismo, substitui-se qualquer indivíduo da população. Caso o operador não produza o melhor indivíduo, não há interferência na população. Outras estratégias de transferência, como substituir o pior, substituir o mais distante, substituir o mais próximo, entre outras, podem ser usadas. Assim, essa fase é a responsável por controlar e definir a forma de interferência do AAP nos AGs.

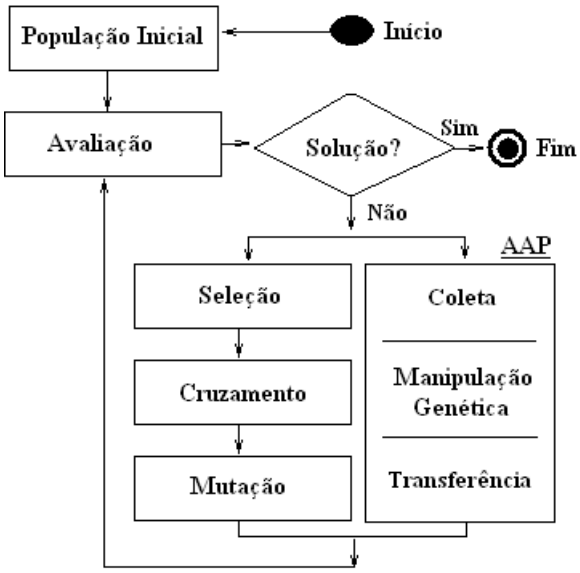


Figura 45: Fases do AAP

Este trabalho apresenta dois grupos de operadores para o AAP, sendo o primeiro com operadores que utilizam como material genético somente a população gerada pelo AG, sem a operação de Alteração Genética na fase de Manipulação Genética, e, o segundo, com operadores que alteram alguns cromossomos da população antes de serem recombinados, com Alteração Genética. A estratégia do segundo grupo é enriquecer a população do AAP com informações que podem ser benéficas para o processo de recombinação. Todos os operadores aplicam como estratégia de Coleta a seleção dos $X\%$ melhores indivíduos da população corrente do AG, e como estratégia de Transferência a substituição dos melhores indivíduos corrente do AG (elitismo) pelos melhores indivíduos encontrados pelo AAP, caso sejam melhores do que o melhor indivíduo do AG. Faz parte do primeiro grupo o operador AR (ver seção 4.4.3.1) e do segundo, os operadores EAR-T, EAR-P, EAR-PA e EAR-N (ver seção 4.4.3.2).

As seções seguintes apresentam o AAP, em maior detalhe, quanto ao seu mecanismo de funcionamento. A seção 4.4.1 mostra o fluxo de execução do algoritmo, a seção 4.4.2 discorre sobre a divisão do material genético, a seção 4.4.3 apresenta os operadores e a seção 4.4.4 o processo de recombinação.

4.4.1 Fluxo de execução do AAP

No início da execução do AG, define-se a divisão do material genético (seção 4.4.2) e as estratégias de Coleta ($qtdIndiv$), Manipulação Genética e Transferência. Essa configuração é feita uma única vez, no início da execução do algoritmo.

Todos os outros procedimentos do algoritmo são executados a cada geração do AG, após receber a população corrente. O pseudo-código do AAP é apresentado no algoritmo 4.4.1 e segue os passos:

1. Aplique a estratégia de Coleta, ver algoritmo 4.4.1 linha 1;
2. Encontre o indivíduo mais apto entre os N selecionados e rotule-o como Pai, ver linha 2 do algoritmo 4.4.1;
3. Aplique a estratégia de Manipulação Genética, ver algoritmo 4.4.2;

   Recombine o indivíduo mais apto (o Pai) e os indivíduos P'. Como o Pai é o mesmo no processo de recombinação, os filhos gerados são irmãos (material genético similar), ver linha 1 do algoritmo 4.4.2;
4. Aplique a estratégia de Transferência, ver linha 4 do algoritmo 4.4.1.

Algoritmo 4.4.1: AAP($Populacao$)

$IndSel \leftarrow null$;
$Pai \leftarrow null$;
$SuperIndiv \leftarrow null$;
Divisão Material Genético();
$IndSel \leftarrow$ Coleta($Populacao$); (1)
$Pai \leftarrow$ Melhor($IndSel$); (2)
$SuperIndiv \leftarrow$ Manipulação Genética($Pai, IndSel$); (3)
Transferência($SuperIndiv$); (4)

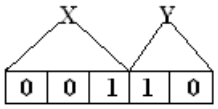


Figura 46: Dois grupos selecionados como material genético para a troca genética

Algoritmo 4.4.2: MANIPULAÇÃO GENÉTICA($Pai, IndSel$)

if $OperadorEAR$
  then $P' \leftarrow$ ALTERAÇÃO GENÉTICA($IndSel$);
  else $P' \leftarrow IndSel$;
$SuperIndiv \leftarrow$ RECOMBINAÇÃO(Pai, P'); (1)
return ($SuperIndiv$);

4.4.2 Divisão do material genético

Antes da recombinação, o cromossomo é dividido em grupos de genes ou gene a gene. Isto é intitulado de Divisão do Material Genético a ser trocado. Esta é uma etapa importante do algoritmo, pois acredita-se que o conhecimento do problema auxilia o projetista a escolher a melhor forma de dividir o cromossomo, e assim ajuda o algoritmo a resolver o problema.

Uma das opções é que o cromossomo seja dividido segundo as variáveis codificadas. Por exemplo, se os 3 primeiros genes representam a variável x e os 2 últimos a y, então o cromossomo é dividido em dois grupos, com um contendo 3 genes e o outro, 2. A Figura 46 ilustra um exemplo de divisão com codificação binária. Dividindo assim, no momento da recombinação a troca é feita entre valores das variáveis (valores decodificados) e não entre valores codificados. Portanto, este processo provê para o projetista uma ferramenta de transferência de conhecimento do problema.

4.4.3 Os operadores do AAP

Criou-se para o AAP quatro operadores como estratégias para auxiliar os AGs a explorarem o espaço de busca. Todos os EAR (seção 4.4.3.2) promovem alterações genéticas na base genética com o intuito de inserir informações relevantes. Já o AR (seção 4.4.3.1) tenta aproveitar as informações existentes sem alterações na base genética de entrada. Os operadores são independentes, por isso não se testou, até o momento, o uso concomitante desses.

Outros operadores são possíveis a partir da mudança das estratégias de Coleta, Manipulação Genética e Transferência, o que caracteriza o AAP como um algoritmo modular e versátil. Por isso, considera-se a proposta promissora e, assim como os AGs, adaptável a diferentes problemas.

4.4.3.1 Operador *Assisted Recombination*

Dada o baixo aproveitamento do material genético produzido pelos AGs, foi desenvolvido o *Assisted Recombination* (AR) - um operador do AAP para melhor explorar as informações genéticas presentes na população dos AGs e contribuir para um melhoramento genético mais ágil.

O operador é constituído de uma única operação na fase de Manipulação Genética, a recombinação (ver detalhes na seção 4.4.4). Nesta, o operador recebe a população da fase de Coleta e recombina, sem fazer alterações, os indivíduos recebidos com o melhor, gerando filhos que são analisados. Caso o processo encontre um melhor indivíduo e opte-se por elitismo, insere-se este na população em substituição ao melhor corrente. Na ausência do elitismo, o indivíduo gerado substitui qualquer indivíduo da população. Caso o operador não produza um melhor indivíduo, não há interferência na população.

Uma das principais características deste operador é a capacidade de assistir as recombinações feitas, identificando indivíduos com aptidão superior ao melhor indivíduo corrente.

4.4.3.2 Operadores *Exploratory Assisted Recombination*

Assim como o *Assisted Recombination*, os operadores *Exploratory Assisted Recombination* (EAR) têm o objetivo de melhor aproveitar as informações produzidas pelos AGs e agilizar o processo de melhoramento genético da população.

A maior contribuição deste novo operador, em relação ao *Assisted Recombination*, é sua maior capacidade de fazer busca global, através das características exploratórias acrescentadas.

Estes operadores passam pelas duas operações da fase de Manipulação Genética, sendo a primeira de alteração de alguns cromossomos recebidos do AG e a segunda de recombinação.

Durante a primeira fase, os N indivíduos da população selecionados pela fase de Coleta

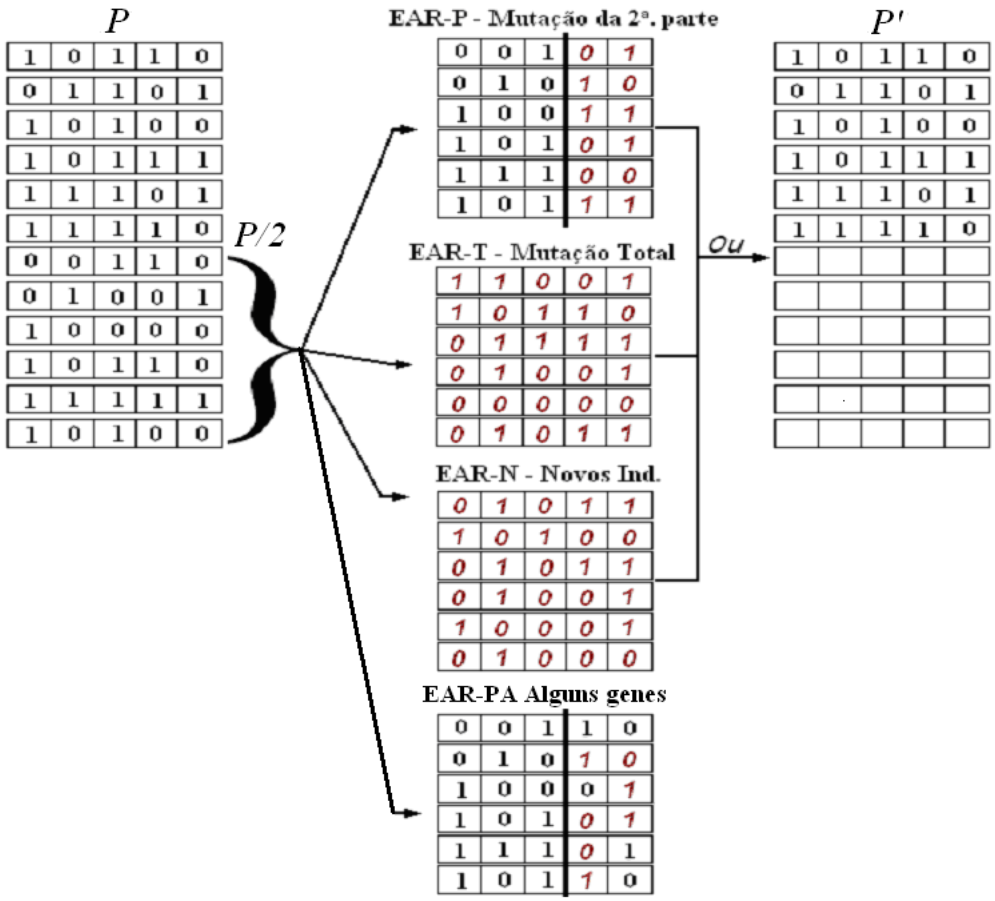


Figura 47: O processo de *exploration* dos EAR

são ordenados e divididos pela metade ($P/2$) e a última porção sofre alteração, de parte (EAR-P ou EAR-PA) ou de todo (EAR-T) o cromossomo.

O operador EAR-P muta parte do cromossomo; para tanto, é sorteada uma das partes definidas pela divisão de material genético a cada iteração. O operador EAR-PA muta alguns genes de uma das partes do cromossomo, tanto a parte quanto os genes são definidos aleatoriamente a cada iteração. Para a escolha dos genes é feito o sorteio de uma taxa que define a probabilidade de cada gene ser alterado. Posteriormente, é aplicado a taxa para definir os genes a serem mutados. O operador EAR-T muta todo o cromossomo, já o operador EAR-N gera aleatoriamente novos indivíduos. Após a alteração, os indivíduos substituem a metade da população escolhida para ser alterada.

A Figura 47 descreve o processo em uma codificação binária, desde a escolha dos $P/2$ indivíduos até a nova população alterada (P'), que será utilizada na fase de recombinação (seção 4.4.4). Neste exemplo, é estabelecido a Divisão do Material Genético em duas partes, sendo a primeira de 3 genes e a segunda com 2 genes. O número de indivíduos provenientes do AG é 12 (qtdIndv), assim os últimos 6 ($P/2$) indivíduos sofrem alterações.

4.4.4 A Recombinação

Estabelecidos os grupos de genes (divisão do material genético) e a população dos P' indivíduos, o passo seguinte é a recombinação. Neste, o cromossomo é representado por um vetor, sendo que, cada elemento deste é um grupo de genes, estabelecido pela divisão do material genético. Assim, estabelecidas as duas partes na divisão, por exemplo, o vetor terá dois elementos.

O melhor indivíduo é reservado como pai e as mães são os indivíduos restantes, sendo que a quantidade de mães é limitada pela fase da Coleta. Para gerar um filho, a mãe doa um elemento do seu vetor e o pai, com os outros elementos do seu vetor, completa o cromossomo do filho. Este processo é repetido para todas as mães, gerando um grupo de filhos. Outros grupos de filhos são gerados pela troca do elemento doado pela mãe e do complemento do cromossomo pelo pai; assim, a quantidade de grupos de filhos é igual à quantidade de elementos do vetor, que por sua vez, é igual à quantidade de grupos de genes.

Após a geração de todos os grupos de filhos, o melhor filho é considerado o melhor Super Indivíduo e é comparado ao pai. Caso seja melhor, o filho sobrepõe ao pai na população do AAP e recomeça-se todo o processo de recombinação, com o melhor filho sendo o pai da iteração. Caso contrário, o algoritmo interrompe o laço de repetição e insere o pai corrente na população final.

Como exemplo de uma iteração da recombinação em uma codificação binária, a Figura 48 mostra:

- O cromossomo é dividido (divisão do material genético) em duas partes. Grupo 1 com 3 genes e o grupo 2 com 2 genes;
- Três mães, estabelecidas pelo parâmetro qtdIndiv = 4, sendo as mães os 3 melhores indivíduos após o pai;
- O primeiro grupo de filhos, gerados pela doação dos primeiros elementos dos vetores das mães e o complemento do pai. Por exemplo, o segundo filho do "Filhos do Grupo 1"(01110) é o resultado da união do primeiro elemento da segunda mãe (011) e do complemento, segundo elemento do vetor, do pai (10);
- No segundo grupo de filhos, a mãe doa o segundo elemento do vetor e o pai completa com o seu primeiro elemento do vetor. Por exemplo, o terceiro filho do "Filhos do

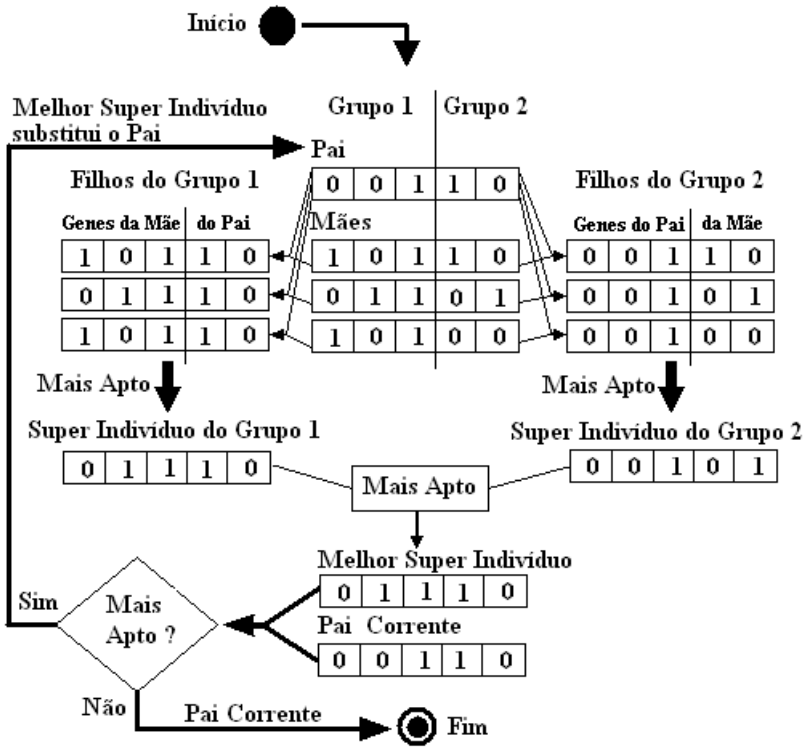


Figura 48: Exemplo do processo de recombinação

Grupo 2"(00100) é o resultado da união do segundo elemento da terceira mãe (00) e do complemento, primeiro elemento do vetor, do pai (001). Neste caso, ao contrário do "Filhos do Grupo 1", o filho é formado pela primeira parte do pai e a segunda parte da mãe. Caso existam mais partes, as mães doam a parte correspondente ao grupo e o pai completa o cromossomo do filho;

- O melhor indivíduo do grupo de filhos é considerado o Super Indivíduo do grupo. O melhor entre os Super Indivíduos é comparado com o pai. Caso seja melhor que o pai, o melhor Super Indivíduo substitui o pai na próxima iteração, caso não, o algoritmo é interrompido e retorna o pai corrente.

5 Experimentos

Para testar o desempenho do algoritmo proposto (AAP) e seus operadores quando acoplados aos AGs, opta-se pelo uso de dois problemas *benchmark*. O primeiro de minimização da função *Rastrigin*, reconhecidamente um problema muito difícil por ter um grande espaço de busca, ser não linear e com alto grau de multimodalidade (DIGALAKIS; MARGARITIS, 2000, 2002), e, o segundo, o da Mochila Multidimensional (*Multidimensional Knapsack Problem*), por ser um problema altamente combinatório e multidimensional. Com isso, é possível medir o desempenho da proposta em dois tipos diferentes de problema: otimização de função multimodal não restritiva e otimização combinatória restritiva.

Considerando que existem excelentes algoritmos específicos para os problemas citados, o AAP não objetiva ter o melhor desempenho, Teorema *No Free Lunch* (WOLPERT; MACREADY, 1997). Todavia, se apresenta como uma proposta de acoplamento benéfica aos Algoritmos Genéticos. Em vista disso, usa-se também como base comparativa um algoritmo híbrido AG-BT, apresentado na seção 4.3.2, que tem objetivo similar ao AAP - alimentar os AGs de bons indivíduos - e estratégia de solução parecida para o problema, um balanço entre busca local e global.

O híbrido tem a base dos AGs e o acréscimo da Busca Tabu, com o papel de busca local. A intervenção do BT no AG se dá a cada 10 iterações do AG canônico. O AG envia o melhor indivíduo corrente e recebe o resultado da busca em vizinhança feita pelo BT.

Os experimentos foram executados em um PC com o processador Intel Pentium 4 com 1.6 GHz e memória de 1 GB. Considera-se como ponto de parada quando o algoritmo atingi um determinado número de avaliações dos indivíduos, ou seja, um determinado número de cálculos de função desempenho. Esse critério é usado devido ao caráter comparativo em alguns cenários com o algoritmo híbrido, como sugere a literatura (SOARES, 1997).

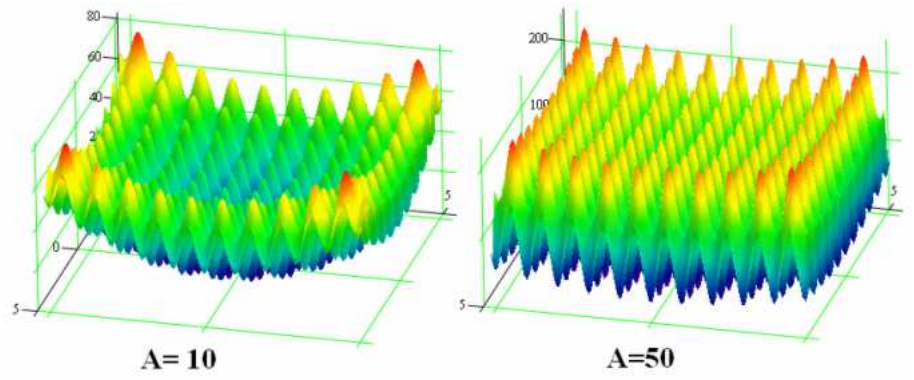


Figura 49: Função Bidimensional Rastrigin com A=10 e A=50

5.1 Função *Rastrigin*

A função de Rastrigin (equação 5.1) é muito utilizada por ser multimodal e ter um comportamento com vários picos e vales. Caracteriza, portanto, vários ótimos locais. A Figura 49 ilustra a função.

$$f(x) = nA + \sum_{i=1}^{n}(x_i^2 - A\cos(2\pi x_i)); x_i \in [-5,12; 5,12], \forall i \in [1..n] \tag{5.1}$$

Dada a função, o objetivo é minimizar $f(x)$, respeitando as restrições: $-5.12 \leq Xi \leq 5.12$ e valores com quatro casas decimais. Considerando as restrições, o ponto mínimo da função é atingido quando $Xi = 0$. Nesse ponto ótimo $f(Xi) = 0; \forall i$.

Para esse problema, apresentam-se 6 experimentos. No primeiro, analisa-se o desempenho do AG com o do AAP, dado o mesmo número de cruzamentos por geração. No segundo, analisa-se a eficácia do AAP na minimização (cenário 2.1) e o comportamento dele quando alterado o parâmetro A da função (cenário 2.2). No terceiro, analisa-se o comportamento do AAP quando alterado o número de indivíduos (diversidade genética) na população. Para isso, define-se o cenário 3.1 com 50 indivíduos, o cenário 3.2 com 25 indivíduos e o cenário 3.3 com 10 indivíduos. No quarto, analisa-se o impacto do aumento de dimensões (variáveis) no AAP. O cenário 4.1 é executado com duas variáveis e o cenário 4.2 com 10 variáveis. No quinto experimento, analisa-se o impacto da diminuição da população de entrada do AAP no desempenho. Por último, no sexto experimento, faz-se uma comparação entre o melhor operador do AAP para o problema, identificado pelos cenários anteriores, e um algoritmo híbrido (AG-BT).

Considerando o AAP como um algoritmo que faz *exploration* e *exploitation*, o objetivo do sexto experimento é comparar o AAP com outro algoritmo que tenha o mesmo objetivo,

Tabela 6: Configuração do experimento 1

Experimento 1								
Parâmetros								
Cenário	**AG**			**AAP**			**Função**	
1.1	**Pop**	**QtdG**	**CP**	**Pop**	**DMG**	**QtdIndiv**	**Variáveis**	**A**
	50	34	60k	25	4	25%	2	10

Tabela 7: Configuração do experimento 2

Experimento 2							
Parâmetros							
Cenário	**AG**			**AAP**		**Função**	
2.1	**Pop**	**QtdG**	**CP**	**DMG**	**QtdIndiv**	**Variáveis**	**A**
	50	34	60k	4	20%	2	10
2.2	**Pop**	**QtdG**	**CP**	**DMG**	**QtdIndiv**	**Variáveis**	**A**
	50	34	60k	4	20%	2	50

alimentar os AGs de bons indivíduos, e estratégia de solução similar para o problema, um balanço entre busca local e global. Por isso, um híbrido com base nos AGs e acrescido de Busca Tabu como método de busca local é usado.

Para cada cenário, 10 execuções de cada algoritmo são feitas, sendo que a condição de parada é atingir o ótimo global ou 60000 avaliações (60k). Para uma comparação justa, a cada execução uma nova população inicial é criada aleatoriamente e compartilhada entre os algoritmos. Desta forma, em cada execução os algoritmos iniciam com a mesma população inicial. As execuções são independentes e sequenciais. O operador de seleção do AG para os experimentos é o Torneio, por ser um dos mais usados na literatura.

As Tabelas 6 a 11 mostram as configurações dos seis experimentos. Para melhor visualização, algumas siglas foram usadas, QtdG para a quantidade de genes no cromossomo, DMG para a divisão do material genético, CP para a condição de parada em avaliações e QtdIndiv para a porcentagem da população que será manipulada pelo AAP. Para todos os experimentos a Probabilidade de Cruzamento (PC) =0,65 e Probabilidade de Mutação (PM) =0,05 para os AAPs, já no AG PC=1 e PM=0,01, por apresentar melhores resultados. No experimento 6 o BT recebe, a cada 10 gerações do AG, o melhor indivíduo corrente e executa até 100 iterações sem melhora (BTMax=100). Além disso, a Lista Tabu tem 10 elementos ($|T|$=10) e a vizinhança analisada é 30% dos vizinhos possíveis. Os vizinhos são escolhidos aleatoriamente.

Tabela 8: Configuração do experimento 3

Experimento 3							
Parâmetros							
Cenário	**AG**			**AAP**		**Função**	
3.1	**Pop**	**QtdG**	**CP**	**DMG**	**QtdIndiv**	**Variáveis**	**A**
	50	34	60k	4	20%	2	10
3.2	**Pop**	**QtdG**	**CP**	**DMG**	**QtdIndiv**	**Variáveis**	**A**
	25	34	60k	4	20%	2	10
3.3	**Pop**	**QtdG**	**CP**	**DMG**	**QtdIndiv**	**Variáveis**	**A**
	10	34	60k	4	20%	2	10

Tabela 9: Configuração do experimento 4

Experimento 4							
Parâmetros							
Cenário	**AG**			**AAP**		**Função**	
4.1	**Pop**	**QtdG**	**CP**	**DMG**	**QtdIndiv**	**Variáveis**	**A**
	50	34	60k	4	20%	2	10
4.2	**Pop**	**QtdG**	**CP**	**DMG**	**QtdIndiv**	**Variáveis**	**A**
	50	170	60k	10	20%	10	10

Tabela 10: Configuração do experimento 5

Experimento 5							
Parâmetros							
Cenário	**AG**			**AAP**		**Função**	
5.1	**Pop**	**QtdG**	**CP**	**DMG**	**QtdIndiv**	**Variáveis**	**A**
	25	34	60k	4	100%	2	10
5.2	**Pop**	**QtdG**	**CP**	**DMG**	**QtdIndiv**	**Variáveis**	**A**
	25	34	60k	4	50%	2	10
5.3	**Pop**	**QtdG**	**CP**	**DMG**	**QtdIndiv**	**Variáveis**	**A**
	25	34	60k	4	25%	2	10

Tabela 11: Configuração do experimento 6

Experimento 6							
Parâmetros							
Cenário	**AG-BT**			**AAP**		**Função**	
6.1	**Pop**	**QtdG**	**CP**	**DMG**	**QtdIndiv**	**Variáveis**	**A**
	50	34	60k	4	20%	2	10
6.2	**Pop**	**QtdG**	**CP**	**DMG**	**QtdIndiv**	**Variáveis**	**A**
	50	34	60k	4	20%	2	50
6.3	**Pop**	**QtdG**	**CP**	**DMG**	**QtdIndiv**	**Variáveis**	**A**
	50	34	60k	4	20%	2	10
6.4	**Pop**	**QtdG**	**CP**	**DMG**	**QtdIndiv**	**Variáveis**	**A**
	25	34	60k	4	20%	2	10
6.5	**Pop**	**QtdG**	**CP**	**DMG**	**QtdIndiv**	**Variáveis**	**A**
	10	34	60k	4	20%	2	10
6.6	**Pop**	**QtdG**	**CP**	**DMG**	**QtdIndiv**	**Variáveis**	**A**
	50	34	60k	4	20%	2	10
6.7	**Pop**	**QtdG**	**CP**	**DMG**	**QtdIndiv**	**Variáveis**	**A**
	50	170	60k	10	20%	10	10

5.1.1 Resultados

São coletados nos experimentos: o menor valor de função encontrado (MFO), a quantidade de gerações (QG) para atingir o ponto de parada, o tempo gasto (Ts) em segundos, a quantidade de avaliações até o critério de parada (QA) e o número de vezes em que o ótimo global é atingido (Taxa de Sucesso - TxS). Além desses elementos, alguns valores estatísticos são apresentados, como o valor máximo (Max), o valor mínimo (Min) e a média (Md).

As Tabelas 12 a 24 mostram os resultados dos seus experimentos, nos diferentes cenários, para os algoritmos AG, AAP/AR (AR), AAP/EAR-P (EAR-P), AAP/EAR-T (EAR-T), AAP/EAR-N (EAR-N) e AG-BT.

5.1.2 Análise dos Resultados

O objetivo dos experimentos é mensurar a eficiência e a eficácia do algoritmo proposto. Neste trabalho, mede-se a eficiência pela média (Md) do Ts e pela Md de QA, quanto menor a média maior é a eficiência, e a eficácia pela TxS, quanto maior a taxa de sucesso melhor é a eficácia. A Md de MFO também pode ser considerada quando a TxS for igual ou quando não se obtiver sucesso.

Tabela 12: Resultados do experimento 1

Experimento 1					
Cenário 1.1					
Algoritmo		**Média**	**Max**	**Min**	**TxS**
AG	MFO	0,0995	0,99495	0	9/10
	QG	214,20	1200,00	34,00	
	Ts	15,81	91,72	1,53	
	QA	10710,00	60000,00	1700,00	
AR	MFO	0	0	0	10/10
	QG	50,70	119,00	22,00	
	Ts	6,52	18,49	2,03	
	QA	2709,10	6639,00	1142,00	
EAR-T	MFO	0	0	0	10/10
	QG	161,90	361,00	31,00	
	Ts	19,38	61,75	3,50	
	QA	7036,30	15169,00	1479,00	
EAR-P	MFO	0	0	0	10/10
	QG	61,00	103,00	31,00	
	Ts	7,22	13,90	4,64	
	QA	3358,60	5839,00	1767,00	
EAR-PA	MFO	0	0	0	10/10
	QG	60,60	97,00	24,00	
	Ts	10,09	17,61	1,75	
	QA	3220,60	4949,00	1224,00	
EAR-N	MFO	0	0	0	10/10
	QG	149,00	284,00	46,00	
	Ts	21,67	65,22	3,53	
	QA	6582,60	12097,00	2302,00	

Tabela 13: Resultados do experimento 2, cenário 2.1

Experimento 2					
Cenário 2.1					
Algoritmo		**Média**	**Max**	**Min**	**TxS**
AG	MFO	0,0995	0,99495	0	9/10
	QG	214,20	1200,00	34,00	
	Ts	15,81	91,72	1,53	
	QA	10710,00	60000,00	1700,00	
AR	MFO	0	0	0	10/10
	QG	36,00	72,00	10,00	
	Ts	2,97	8,33	0,72	
	QA	4298,40	8388,00	1256,00	
EAR-T	MFO	0	0	0	10/10
	QG	36,00	84,00	15,00	
	Ts	2,96	9,86	0,88	
	QA	4273,20	10212,00	1722,00	
EAR-P	MFO	0	0	0	10/10
	QG	36,70	68,00	16,00	
	Ts	2,94	8,52	1,15	
	QA	4326,20	7828,00	1808,00	
EAR-PA	MFO	0	0	0	10/10
	QG	26,40	44,00	9,00	
	Ts	2,49	6,42	0,71	
	QA	3174,00	5260,00	990,00	
EAR-N	MFO	0	0	0	10/10
	QG	39,60	90,00	9,00	
	Ts	3,64	13,18	0,72	
	QA	4773,60	10548,00	1098,00	

Tabela 14: Resultados do experimento 2, cenário 2.2

Experimento 2					
Cenário 2.2					
Algoritmo		**Média**	**Max**	**Min**	**TxS**
AG	MFO	0,79942	1,0001	0	2/10
	QG	968,00	1200,00	39,00	
	Ts	88,98	131,67	3,75	
	QA	48400,00	60000,00	1950,00	
AR	MFO	0	0	0	10/10
	QG	61,90	133,00	31,00	
	Ts	8,24	17,54	4,08	
	QA	7152,20	15074,00	3530,00	
EAR-T	MFO	0	0	0	10/10
	QG	35,10	86,00	17,00	
	Ts	4,21	9,83	1,71	
	QA	4073,40	10600,00	1894,00	
EAR-P	MFO	0	0	0	10/10
	QG	34,00	72,00	19,00	
	Ts	4,28	8,89	2,36	
	QA	3867,20	8352,00	2526,00	
EAR-PA	MFO	0	0	0	10/10
	QG	42,70	105,00	17,00	
	Ts	6,95	16,55	2,58	
	QA	5011,40	12702,00	1894,00	
EAR-N	MFO	0	0	0	10/10
	QG	49,60	87,00	12,00	
	Ts	6,71	12,03	1,74	
	QA	5507,60	9462,00	1572,00	

Tabela 15: Resultados do experimento 3, cenário 3.1

Experimento 3					
Cenário 3.1					
Algoritmo		**Média**	**Max**	**Min**	**TxS**
AG	MFO	0,0995	0,99495	0	9/10
	QG	214,20	1200,00	34,00	
	Ts	15,81	91,72	1,53	
	QA	10710,00	60000,00	1700,00	
AR	MFO	0	0	0	10/10
	QG	36,00	72,00	10,00	
	Ts	2,97	8,33	0,72	
	QA	4298,40	8388,00	1256,00	
EAR-T	MFO	0	0	0	10/10
	QG	36,00	84,00	15,00	
	Ts	2,96	9,86	0,88	
	QA	4273,20	10212,00	1722,00	
EAR-P	MFO	0	0	0	10/10
	QG	36,70	68,00	16,00	
	Ts	2,94	8,52	1,15	
	QA	4326,20	7828,00	1808,00	
EAR-PA	MFO	0	0	0	10/10
	QG	26,40	44,00	9,00	
	Ts	2,49	6,42	0,71	
	QA	3174,00	5260,00	990,00	
EAR-N	MFO	0	0	0	10/10
	QG	39,60	90,00	9,00	
	Ts	3,64	13,18	0,72	
	QA	4773,60	10548,00	1098,00	

Tabela 16: Resultados do experimento 3, cenário 3.2

Experimento 3					
Cenário 3.2					
Algoritmo		**Média**	**Max**	**Min**	**TxS**
AG	MFO	0,34673	2,47234	0	8/10
	QG	731,90	2400,00	52,00	
	Ts	84,14	321,80	2,53	
	QA	18297,50	60000,00	1300,00	
AR	MFO	0	0	0	10/10
	QG	50,70	119,00	22,00	
	Ts	6,52	18,49	2,03	
	QA	2709,10	6639,00	1142,00	
EAR-T	MFO	0	0	0	10/10
	QG	161,90	361,00	31,00	
	Ts	19,38	61,75	3,50	
	QA	7036,30	15169,00	1479,00	
EAR-P	MFO	0	0	0	10/10
	QG	61,00	103,00	31,00	
	Ts	7,22	13,90	4,64	
	QA	3358,60	5839,00	1767,00	
EAR-PA	MFO	0	0	0	10/10
	QG	60,60	97,00	24,00	
	Ts	10,09	17,61	1,75	
	QA	3220,60	4949,00	1224,00	
EAR-N	MFO	0	0	0	10/10
	QG	149,00	284,00	46,00	
	Ts	21,67	65,22	3,53	
	QA	6582,60	12097,00	2302,00	

Tabela 17: Resultados do experimento 3, cenário 3.3

Experimento 3					
Cenário 3.3					
Algoritmo		**Média**	**Max**	**Min**	**TxS**
AG	MFO	0,86532	2,47234	0	5/10
	QG	3409,20	6000,00	308,00	
	Ts	412,03	777,43	30,94	
	QA	34092,00	60000,00	3080,00	
AR	MFO	0	0	0	10/10
	QG	145,90	250,00	72,00	
	Ts	35,48	76,49	12,43	
	QA	2370,20	4160,00	1218,00	
EAR-T	MFO	0	0	0	10/10
	QG	226,00	347,00	106,00	
	Ts	50,79	123,01	9,25	
	QA	3442,40	5254,00	1644,00	
EAR-P	MFO	0	0	0	10/10
	QG	207,20	578,00	95,00	
	Ts	45,74	113,67	12,03	
	QA	3188,80	8996,00	1454,00	
EAR-PA	MFO	0	0	0	10/10
	QG	183,60	327,00	91,00	
	Ts	52,74	109,23	17,21	
	QA	2783,60	4926,00	1362,00	
EAR-N	MFO	0	0	0	10/10
	QG	298,70	533,00	84,00	
	Ts	89,36	189,83	8,58	
	QA	4181,00	7462,00	1172,00	

Tabela 18: Resultados do experimento 4, cenário 4.1

Experimento 4					
Cenário 4.1					
Algoritmo		**Média**	**Max**	**Min**	**TxS**
AG	MFO	0,0995	0,99495	0	9/10
	QG	214,20	1200,00	34,00	
	Ts	15,81	91,72	1,53	
	QA	10710,00	60000,00	1700,00	
AR	MFO	0	0	0	10/10
	QG	36,00	72,00	10,00	
	Ts	2,97	8,33	0,72	
	QA	4298,40	8388,00	1256,00	
EAR-T	MFO	0	0	0	10/10
	QG	36,00	84,00	15,00	
	Ts	2,96	9,86	0,88	
	QA	4273,20	10212,00	1722,00	
EAR-P	MFO	0	0	0	10/10
	QG	36,70	68,00	16,00	
	Ts	2,94	8,52	1,15	
	QA	4326,20	7828,00	1808,00	
EAR-PA	MFO	0	0	0	10/10
	QG	26,40	44,00	9,00	
	Ts	2,49	6,42	0,71	
	QA	3174,00	5260,00	990,00	
EAR-N	MFO	0	0	0	10/10
	QG	39,60	90,00	9,00	
	Ts	3,64	13,18	0,72	
	QA	4773,60	10548,00	1098,00	

Tabela 19: Resultados do experimento 4, cenário 4.2

Experimento 4					
Cenário 4.2					
Algoritmo		**Média**	**Max**	**Min**	**TxS**
AG	MFO	1,35851	2,35295	0,02378	0/10
	QG	1200,00	1200,00	1200,00	
	Ts	134,14	146,49	114,89	
	QA	60000	60000	60000	
AR	MFO	0	0	0	10/10
	QG	114,90	169,00	72,00	
	Ts	25,56	37,48	16,01	
	QA	26391	38060	16650	
EAR-T	MFO	0	0	0	10/10
	QG	111,50	169,00	81,00	
	Ts	23,14	32,67	16,88	
	QA	25762	33470	18900	
EAR-P	MFO	0	0	0	10/10
	QG	139,60	206,00	88,00	
	Ts	29,48	44,34	20,38	
	QA	31388	49630	21770	
EAR-PA	MFO	0	0	0	10/10
	QG	119,30	198,00	81,00	
	Ts	30,91	50,11	20,90	
	QA	28195	48240	20160	
EAR-N	MFO	0	0	0	10/10
	QG	166,40	224,00	87,00	
	Ts	38,56	51,99	20,42	
	QA	35959	52510	21360	

Tabela 20: Resultados do experimento 5, cenário 5.1

Experimento 5					
Cenário 5.1					
Algoritmo		**Média**	**Max**	**Min**	**TxS**
AR	MFO	0	0	0	10/10
	QG	31,1	75	10	
	Ts	5,295	16,22	1,03	
	QA	6307,1	13203	2458	
EAR-T	MFO	0	0	0	10/10
	QG	27,5	51	7	
	Ts	3,944	9,45	0,64	
	QA	5919,5	10683	1519	
EAR-P	MFO	0	0	0	10/10
	QG	18,8	30	10	
	Ts	2,834	4,76	1,2	
	QA	4223,6	6510	2362	
EAR-PA	MFO	0	0	0	10/10
	QG	25,2	38	11	
	Ts	5,021	9,82	1,48	
	QA	5190	8822	2291	
EAR-N	MFO	0	0	0	10/10
	QG	20,9	29	17	
	Ts	3,803	6,87	1,7	
	QA	4612,1	6869	3209	

Tabela 21: Resultados do experimento 5, cenário 5.2

Experimento 5					
Cenário 5.2					
Algoritmo		**Média**	**Max**	**Min**	**TxS**
AR	MFO	0	0	0	10/10
	QG	47,4	70	20	
	Ts	8,637	16,32	2,91	
	QA	4894,2	7690	2392	
EAR-T	MFO	0	0	0	10/10
	QG	31,6	60	11	
	Ts	4,946	7,99	1,34	
	QA	3447,6	6340	1419	
EAR-P	MFO	0	0	0	10/10
	QG	38,3	85	18	
	Ts	6,514	16,83	2,03	
	QA	4055,1	9561	1946	
EAR-PA	MFO	0	0	0	10/10
	QG	38	65	10	
	Ts	7,838	18,84	1,22	
	QA	4113,6	7037	1086	
EAR-N	MFO	0	0	0	10/10
	QG	42,7	114	19	
	Ts	8,371	26,31	2,33	
	QA	4600,7	12794	1927	

Tabela 22: Resultados do experimento 5, cenário 5.3

Experimento 5					
Cenário 5.3					
Algoritmo		**Média**	**Max**	**Min**	**TxS**
AR	MFO	0	0	0	10/10
	QG	50,70	119,00	22,00	
	Ts	6,52	18,49	2,03	
	QA	2709,10	6639,00	1142,00	
EAR-T	MFO	0	0	0	10/10
	QG	161,90	361,00	31,00	
	Ts	19,38	61,75	3,50	
	QA	7036,30	15169,00	1479,00	
EAR-P	MFO	0	0	0	10/10
	QG	61,00	103,00	31,00	
	Ts	7,22	13,90	4,64	
	QA	3358,60	5839,00	1767,00	
EAR-PA	MFO	0	0	0	10/10
	QG	60,60	97,00	24,00	
	Ts	10,09	17,61	1,75	
	QA	3220,60	4949,00	1224,00	
EAR-N	MFO	0	0	0	10/10
	QG	149,00	284,00	46,00	
	Ts	21,67	65,22	3,53	
	QA	6582,60	12097,00	2302,00	

Tabela 23: Resultados do experimento 6

Experimento 6					
Cenário 6.1					
Algoritmo		**Média**	**Max**	**Min**	**TxS**
AG-BT	MFO	1,6E-06	7,9E-06	0	8/10
	QG	61,00	120,00	20,00	
	Ts	6,11	12,98	2,11	
	QA	32052,90	62799,00	13140,00	
EAR-PA	MFO	0	0	0	10/10
	QG	26,40	44,00	9,00	
	Ts	2,49	6,42	0,71	
	QA	3174,00	5260,00	990,00	
Cenário 6.2					
Algoritmo		**Média**	**Max**	**Min**	**TxS**
AG-BT	MFO	0,1998	0,999	3E-06	7/10
	QG	82,60	130,00	40,00	
	Ts	14,29	24,83	5,65	
	QA	44143,20	63972,00	21078,00	
EAR-P	MFO	0	0	0	10/10
	QG	34,00	72,00	19,00	
	Ts	4,28	8,89	2,36	
	QA	3867,20	8352,00	2526,00	
Cenário 6.3					
Algoritmo		**Média**	**Max**	**Min**	**TxS**
AG-BT	MFO	1,6E-06	7,9E-06	0	8/10
	QG	61,00	120,00	20,00	
	Ts	6,11	12,98	2,11	
	QA	32052,90	62799,00	13140,00	
EAR-PA	MFO	0	0	0	10/10
	QG	26,40	44,00	9,00	
	Ts	2,49	6,42	0,71	
	QA	3174,00	5260,00	990,00	
Cenário 6.4					
Algoritmo		**Média**	**Max**	**Min**	**TxS**
AG-BT	MFO	1,6E-06	7,9E-06	6,1E-07	8/10
	QG	69,80	130,00	20,00	
	Ts	12,59	23,00	4,32	
	QA	38254,00	62945,00	15454,00	
AR	MFO	0	0	0	10/10
	QG	50,70	119,00	22,00	
	Ts	6,52	18,49	2,03	
	QA	2709,10	6639,00	1142,00	

Tabela 24: Resultados do experimento 6 - Continuação

Experimento 6					
Cenário 6.5					
Algoritmo		**Média**	**Max**	**Min**	**TxS**
AG-BT	MFO	8,5E-07	3E-06	0	9/10
	QG	92,90	140,00	30,00	
	Ts	28,55	56,92	12,61	
	QA	46327,60	62062,00	24453,00	
AR	MFO	0	0	0	10/10
	QG	145,90	250,00	72,00	
	Ts	35,48	76,49	12,43	
	QA	2370,20	4160,00	1218,00	
Cenário 6.6					
Algoritmo		**Média**	**Max**	**Min**	**TxS**
AG-BT	MFO	1,6E-06	7,9E-06	0	8/10
	QG	61,00	120,00	20,00	
	Ts	6,11	12,98	2,11	
	QA	32052,90	62799,00	13140,00	
EAR-PA	MFO	0	0	0	10/10
	QG	26,40	44,00	9,00	
	Ts	2,49	6,42	0,71	
	QA	3174,00	5260,00	990,00	
Cenário 6.7					
Algoritmo		**Média**	**Max**	**Min**	**TxS**
AG-BT	MFO	14,9084	20,7413	9,15002	0/10
	QG	27,00	30,00	20,00	
	Ts	22,10	23,91	18,52	
	QA	73020,1	76693	61772	
EAR-T	MFO	0	0	0	10/10
	QG	111,50	169,00	81,00	
	Ts	23,14	32,67	16,88	
	QA	25762	33470	18900	

Com o cenário 1.1 objetivou-se comparar o AG com o AAP, configurado para fazer o mesmo número de cruzamentos por geração que o AG. Com as configurações estabelecidas no cenário, o AG fez 25 cruzamentos por geração e os AAPs 28. Optou-se por não igualar o número de mutações por geração, pois testes preliminares demonstraram que o aumento da taxa de mutação, além do estabelecido no cenário, era prejudicial para o AG. Nesse contexto, observa-se que os AAPs tiveram desempenho superior ao AG, apesar do EAR-T e EAR-N terem a média de Ts um pouco maior do que a obtida pelo AG.

Entre os AAPs, o AR foi o melhor no cenário 1.1. Atribui-se essa superioridade nesse cenário, apesar de pequena, ao fato de que com poucos indivíduos na população inicial a população de entrada do AAP é também pequena, o que prejudica o tratamento das informações dos indivíduos elites pelo processo de recombinação. Na estratégia EAR, alguns dos poucos indivíduos elitizados de entrada são alterados antes da recombinação e, por isso, informações importantes de indivíduos bem adaptados são descartadas, prejudicando o resultado final. Esse comportamento não é percebido quando o número de indivíduos de entrada no AAP é maior, pois os melhores indivíduos com as melhores informações não sofrem alterações.

Destaca-se também nesse cenário o baixo número de avaliações feitas pelos AAPs, especialmente o AR, para alcançar o ótimo. A vantagem de ter menos QA é que quanto menor o número de avaliações, mais rápido e melhor é o algoritmo. Em problemas com alta complexidade no cálculo da função objetivo a redução do número de avaliações para alcançar o ótimo contribui muito para o desempenho do algoritmo.

Analisando o comportamento dos algoritmos no experimento 2, quando o parâmetro "A" da função aumenta de 10 para 50, observa-se que o AG sofreu uma queda no desempenho, já que a TxS de 90% caiu para 20% e a média da MFO de 0,09995 passou para 0,79942. Já os AAPs, mantiveram a eficácia entre os cenários e uma pequena perda de eficiência, pois aumentaram a média de Ts, demonstrando que com o aumento da complexidade do problema o AAP sofre um impacto negativo.

No cenário 2.1 o melhor desempenho foi do EAR-PA; já no cenário 2.2, o EAR-P e o EAR-T tiveram os melhores resultados. No cenário 2.2, percebe-se uma vantagem da estratégia EAR em relação à AR. Essa diferença não foi observada no cenário 2.1. A Figura 50 ilustra o desempenho dos algoritmos nesses cenários, avaliados pela média de QA.

Analisando o impacto da diminuição de diversidade, experimento 3, percebe-se que todos os algoritmos sofrem um impacto negativo no desempenho quando se diminui o

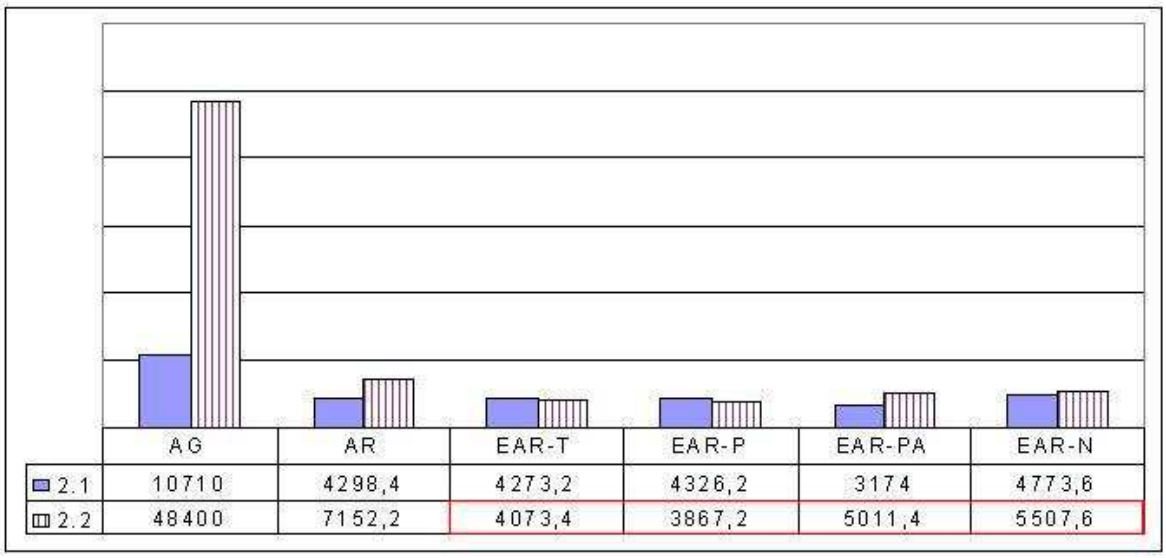


Figura 50: Média de QA nos cenários 2.1 e 2.2

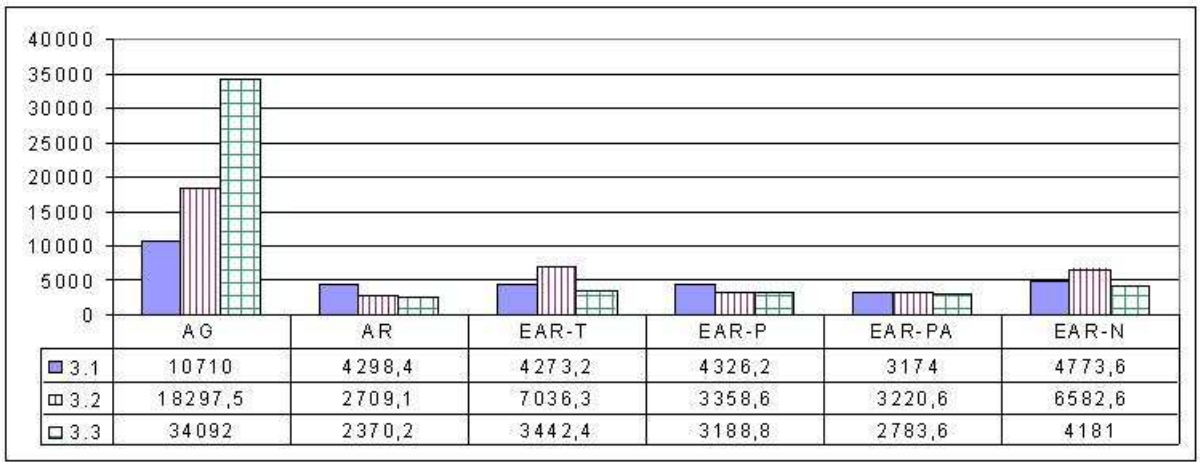
número de indivíduos na população. O AG foi o mais impactado, com perda de eficácia de 40% e um aumento significativo na média MFO, entre os cenários 3.1 e o 3.3.

Entre os AAPs, o AR obteve o melhor resultado. Conjectura-se que esse desempenho é devido à manutenção dos indivíduos elite para a recombinação do AAP. Nas estratégias EARs os indivíduos são alterados, perdendo assim algumas informações elitisadas. Essa perda é menor quando o número de indivíduos de entrada no AAP é maior (cenário 3.1), já que assim preserva-se um maior número de elitisados e altera-se outros indivíduos menos aptos.

Entre os EARs, o EAR-P e EAR-PA foram os melhores, com destaque para o desempenho do EAR-PA, quanto à média QA. Esse operador obteve o menor número de avaliações para alcançar o ótimo. A Figura 51 ilustra o desempenho dos algoritmos no experimento 3, avaliados pela média de QA.

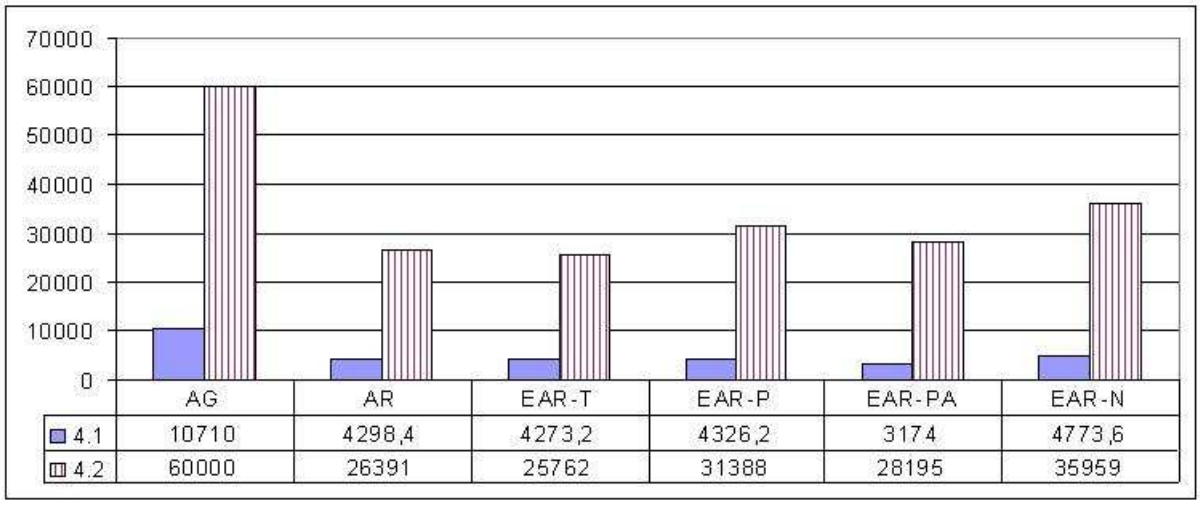
Com o aumento das dimensões do problema, experimento 4, o AG obteve uma grande perda de eficácia. Entre os AAPs, todos obtiveram bons resultados, apesar de impactados pelo aumento das dimensões. Desses, o EAR-T foi o que obteve melhor resultado no cenário com 10 dimensões, seguido do AR e o EAR-P. A Figura 52 ilustra o desempenho dos algoritmos nesse experimento, avaliados pela média de QA.

A população de entrada determina os indivíduos (informações) que serão trabalhados pelo AAP. Quanto mais indivíduos maior é o número de informações e melhor é para o processo de recombinação e aproveitamento. No entanto, o crescimento da população de entrada tende a aumentar o número de avaliações por geração (custo computacional).



	AG	AR	EAR-T	EAR-P	EAR-PA	EAR-N
3.1	10710	4298,4	4273,2	4326,2	3174	4773,6
3.2	18297,5	2709,1	7036,3	3358,6	3220,6	6582,6
3.3	34092	2370,2	3442,4	3188,8	2783,6	4181

Figura 51: Média de QA nos cenários 3.1, 3.2 e 3.3



	AG	AR	EAR-T	EAR-P	EAR-PA	EAR-N
4.1	10710	4298,4	4273,2	4326,2	3174	4773,6
4.2	60000	26391	25762	31388	28195	35959

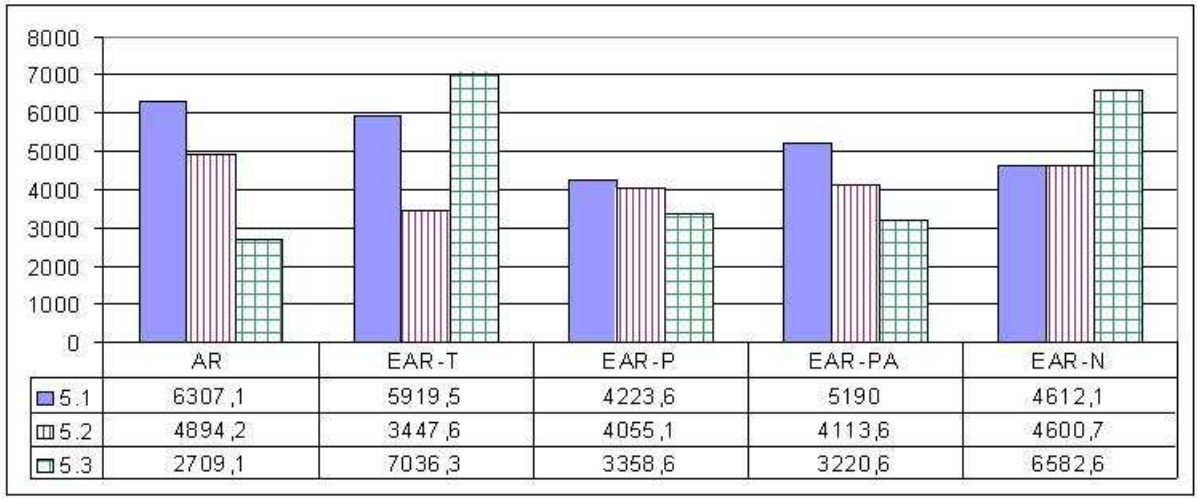
Figura 52: Média de QA nos cenários 4.1 e 4.2

Por isso, deve-se buscar o equilíbrio entre o custo computacional e o benefício de uma população de entrada maior.

Analisando os resultados do experimento 5, percebe-se que para este problema o aumento da população de entrada não acarretou em prejuízo computacional, pelo contrário, com o aproveitamento de mais informações e um pequeno custo na avaliação, agilizou-se a busca. Demonstrou-se, assim, que o benefício do AAP compensou o pequeno aumento dos custos.

Entre os AAPs, o AR, o EAR-P e o EAR-PA foram os que obtiveram os menores tempos no cenário 5.3 e sofreram menos com a diminuição da população de entrada. No melhor caso, cenário 5.1, o EAR-P e o EAR-N foram os melhores, conseguindo atingir o ótimo em 2,8s e 3,8s em média, respectivamente.

Destaca-se a alta dependência do EAR-N à população de entrada. No cenário 5.3, com 25% de população de entrada, o algoritmo teve o pior desempenho entre os AAPs e mostrou-se bem pior do que no cenário 5.2, onde a população de entrada era de 50%. Nos demais algoritmos a diferença entre os cenários não foi tão acentuada. A Figura 53 ilustra o desempenho dos algoritmos nos cenários 5.1, 5.2 e 5.3, avaliados pela média de QA.



	AR	EAR-T	EAR-P	EAR-PA	EAR-N
5.1	6307,1	5919,5	4223,6	5190	4612,1
5.2	4894,2	3447,6	4055,1	4113,6	4600,7
5.3	2709,1	7036,3	3358,6	3220,6	6582,6

Figura 53: Média de QA nos cenários 5.1, 5.2 e 5.3

Comparando os cenários 5.2 e o 3.1, que têm quase o mesmo número de indivíduos na população de entrada do AAP (12 e 10, respectivamente), mas tem população diferente no AG (25 indivíduos e 50 indivíduos, respectivamente), percebe-se que o aumento de indivíduos na população do AG traz benefícios, pois com 50 indivíduos (cenário 3.1) os AAPs, em geral, obtiveram melhores resultados. Isso mostra que a evolução do AG também contribui para o bom desempenho do AAP. Ou seja, é uma relação de mutualismo, onde o AAP auxilia o AG mas depende de um bom desempenho desse para ter bom desempenho.

No experimento 6 os melhores AAPs para os cenários anteriores são comparados ao AG-BT, para verificar o desempenho dos AAPs em relação à outra técnica com busca local.

Analisando os resultados apresentados nas Tabelas 23 e 24, referentes aos cenários 6.1 e 6.2, o AG-BT teve eficácia e eficiência inferior ao EAR-PA no cenário 6.1 e inferior ao EAR-P no cenário 6.2. Destaca-se o alto impacto negativo no AG-BT com o aumento do parâmetro "A"da função, característica que não foi apresentada nos AAPs.

Analisando o impacto da redução de diversidade, percebe-se que o AG-BT tem de-

pendência do número de indivíduos na população. Apesar de manter uma boa eficácia, o AG-BT aumentou o número de avaliações para atingir o ótimo.

Já os AAPs mantiveram a eficácia e oscilaram a eficiência. Entre os cenários 6.3 e 6.4, com a diminuição de 50% da população, houve um acréscimo de 261,84% na média Ts, que representa uma redução significativa da eficiência do melhor AAP e uma alta sensibilidade ao número de indivíduos na população.

No cenário 6.5, com 10 indivíduos, o AG-BT teve um acréscimo de 4,67 vezes na média de tempo em relação ao cenário 6.3, com 50 indivíduos. Percebe-se, também, que o AG-BT, apesar de ter obtido menor média Ts, teve eficácia menor do que o AAP e uma grande quantidade de avaliações para alcançar o ótimo. Constata-se, nesse cenário, que o AAP gasta mais tempo para fazer as avaliações do que o AG-BT, que apresentou uma relação de 1654 avaliações por segundo, contra 67 avaliações por segundo do AAP.

Quando aumenta-se as dimensões no problema, aumenta-se também a complexidade e, consequentemente, fica mais difícil a resolução do mesmo. No cenário 6.7, com 10 dimensões, percebe-se que o AG-BT teve dificuldades, pois não obteve o ótimo em nenhum dos testes e teve uma média de MFO alta. Já o melhor AAP teve uma boa eficácia e eficiência, apesar de ter feito várias avaliações, fato não apresentado em outros cenários. Atribui-se esse acréscimo de avaliações à dificuldade estabelecida com o aumento de dimensões no problema. A Figura 54 ilustra o desempenho dos algoritmos no experimento 6, avaliados pela média de QA.

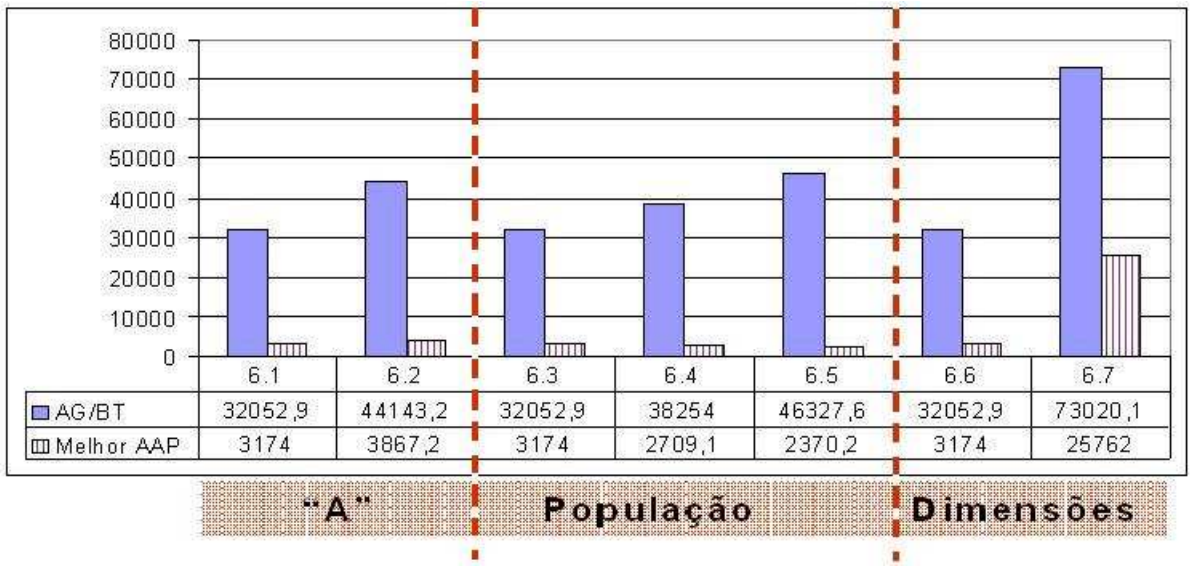


Figura 54: Média de QA no experimento 6

5.2 Mochila Multidimensional

O Problema da Mochila Multidimensional (PMM) é um *benchmark* largamente estudado na literatura, por ser um problema de otimização combinatório NP-Difícil e por ocorrer em vários tipos de aplicações reais (PUCHINGER; RAIDL; PFERSCHY, 2006). O PMM é a variante de maior complexidade do Problema da Mochila (GAREY; JOHNSON, 1979) e, por isso, é usado neste trabalho conforme as equações (5.2) a (5.4).

$$Maximizar \quad \sum_{j=1}^{n} c_j * x_j \tag{5.2}$$

sujeito a:

$$\sum_{j=1}^{n} a_{i,j} * x_j \leq b_i, \forall i = 1..m \tag{5.3}$$

$$x_j \in \{0,1\}, 1 \leq j \leq n \tag{5.4}$$

em que n é o número de objetos, m é o número de dimensões (compartimentos) da mochila, c_j representa o benefício do objeto j na mochila, x_j é uma variável binária que indica se o objeto j está guardado na mochila ($x_j = 1$) ou se está fora ($x_j = 0$), b_i representa a capacidade i-ésima dimensão da mochila, e $a_{i,j}$ representa as entradas da matriz de pesos dos objetos para cada dimensão.

A matriz dos coeficientes guarda os valores de peso de cada objeto para cada dimensão. Assim, a matriz tem o tamanho de $m \times n$, sendo n colunas e m linhas.

O objetivo é selecionar um subconjunto de objetos que maximize o benefício total, (ver equação 5.2); no entanto, os itens escolhidos não podem exceder a capacidade de cada compartimento (ver equação 5.3).

Esse problema, diferentemente da função Rastrigin, é um problema restritivo e, por isso, foi escolhido para verificar o comportamento da proposta nesse tipo de aplicação.

Para os testes, foram coletadas na literatura 21 instâncias do PMM, disponíveis na OR-Library (PETERSEN, 1967). Esses problemas consistem de m= 5 a 10 e n=6 a 100.

Para cada instância, montou-se um cenário e foram feitas 10 execuções, em que a condição de parada é atingir o ótimo global ou 50000 avaliações (50k) da função objetivo.

As instâncias do experimento 2 têm um fator de correlação alfa (α) que determina o nível de correlação entre a capacidade de cada dimensão da mochila e o somatório dos pesos dos objetos para essa dimensão.

A codificação é binária. Cada gene representa um objeto e a função a ser maximizada é da equação (5.2), que não penaliza as soluções infactíveis. O uso exclusivo de soluções factíveis melhora o desempenho do algoritmo para o problema (HOFF; LØKKETANGEN; MITTET, 1996). Em vista disso, foi criado um operador de reparação construtivo para que as soluções que não atendam as restrições tornem-se factíveis. O operador é baseado no trabalho de (CHU; BEASLEY, 1998), mas menos guloso e menos determinístico. O operador implementado também tem uma fase de retirada e outra de inserção de objetos na solução; no entanto, ao invés de sempre inserir o mais apto e retirar o menos apto, faz um torneio entre os indivíduos para verificar qual será inserido ou retirado. A pressão de seleção do torneio para esse problema é de 15%, tanto para a retirada quanto para a inserção, por apresentar melhor resultado.

A população inicial é, normalmente, fator preponderante no sucesso dos AGs, por isso, foi usada a mesma heurística construtiva de inserir objetos na solução, utilizada no operador de reparação, para criar os indivíduos iniciais. A pressão de seleção do torneio nesse caso é de 20%, por apresentar melhor resultado.

Para uma comparação justa, a cada execução uma nova população inicial é criada aleatoriamente e compartilhada entre os algoritmos. Dessa forma, em cada execução os algoritmos começam com a mesma população inicial. As execuções são independentes e sequenciais.

O operador de seleção do AG para os experimentos é o Torneio, por se tratar de um método largamente utilizado na literatura. Para todos os experimentos definiu-se a PC=0.8, PM=1/N e Pop= 5*N, sendo N o número de objetos (HOFF; LØKKETANGEN; MITTET, 1996). O BT é executado a cada 10 iterações do AG. As configurações dos experimentos estão nas Tabelas 25 e 26. Os vizinhos são escolhidos aleatoriamente.

5.2.1 Resultados

O maior valor de função encontrado (MFO), o número de vezes que atinge o ótimo global (QO), o tempo gasto em segundos (Ts) e a quantidade de avaliações (Aval) até atingir a condição de parada são dados coletados nos experimentos. No experimento 2, é calculado o Gap%, que é a diferença entre o maior valor encontrado na literatura e o alcançado pelos algoritmos.

A Tabela 27 apresenta os resultados dos algoritmos AG, AAP/AR (AR), AAP/EAR-P (EAR-P), AAP/EAR-T (EAR-T) para o experimento 1. A Tabela 28 apresenta os

Tabela 25: Configuração dos algoritmos para o experimento 1

Experimento 1									
Cenários	Parâmetros do AG		Parâmetros do Busca Tabu		Parâmetros AAP		Parâmetros do Problema		
	Pop	QtdG	BTMax	Tabu	DMG	QtdIndiv	N	M	Melhor
1	5*N	N	100	10	2	20%Pop	6	10	3800
2	5*N	N	100	10	2	20%Pop	10	10	8706,1
3	5*N	N	100	10	2	20%Pop	10	15	4015
4	5*N	N	100	10	2	20%Pop	10	20	6120
5	5*N	N	100	10	2	20%Pop	10	28	12400
6	5*N	N	100	10	2	20%Pop	5	39	10618
7	5*N	N	100	10	2	20%Pop	5	50	16537

Tabela 26: Configuração dos algoritmos para o experimento 2

Experimento 2									
Cenários	Parâmetros do Problema			Parâmetros do AG		Parâmetros do Busca Tabu		Parâmetros AAP	
	N	M	α	Pop	QtdG	BTMax	Tabu	DMG	QtdIndiv
2.1 a 2.10	5	100	0,25	50	N	50	2	2	20%Pop
2.11 e 2.12	5	100	0,50	50	N	50	2	2	20%Pop
2.13 e 2.14	5	100	0,75	50	N	50	2	2	20%Pop

resultados dos algoritmos AAP/EAR-N (EAR-N) e AG/BT para o experimento 1. A Tabela 29 apresenta os resultados dos algoritmos para o experimento 2.

Tabela 27: Resultado do experimento 1 para os algoritmos AG, EAR-T, EAR-P e AR

		Experimento 1											
		AG			**EAR-T**			**EAR-P**			**AR**		
Cenário		**MFO**	**Ts**	**Aval**	**MFO**	**Ts**	**Aval**	**MFO**	**Ts**	**Aval**	**MFO**	**Ts**	**Aval**
1	**Md**	3800	0,05	42	3800	0,05	47	**3800**	**0,05**	**47**	3800	0,05	51
	QO	10/10			10/10			**10/10**			10/10		
2	**Md**	8706,1	0,32	325	8706,1	0,26	326,8	**8706,1**	**0,15**	**199**	8706,1	0,20	248,4
	QO	10/10			10/10			**10/10**			10/10		
3	**Md**	4015	0,84	735	4015	0,57	614,3	**4015**	**0,46**	**511,3**	4015	0,44	442
	QO	10/10			10/10			**10/10**			10/10		
4	**Md**	6120	0,50	400	6120	1,69	1570,4	**6120**	**0,97**	**894,2**	6120	0,13	131,4
	QO	10/10			10/10			**10/10**			10/10		
5	**Md**	12400	14,86	10402	12400	7,05	5524,8	**12400**	**3,05**	**2583,6**	12400	3,81	3169,8
	QO	10/10			10/10			**10/10**			10/10		
6	**Md**	10589,6	59,05	50115	10592,6	49,98	50111,8	**10599**	**51,77**	**50128,3**	10596	54,83	50133,6
	QO	0/10			0/10			**0/10**			0/10		
7	**Md**	16473,4	66,17	50000	16494	61,41	50159,8	**16516,4**	**58,39**	**48520,2**	16500,6	60,83	49205,4
	QO	0/10			0/10			**2/10**			1/10		

Tabela 28: Resultado do experimento 1 para os algoritmos EAR-N e AG/BT

		Experimento 1 continuação					
		EAR-N			AG/BT		
Cenário		MFO	Ts	Aval	MFO	Ts	Aval
1	Md	3800	0,05	51	3800	0,05	42
	QO	10/10			10/10		
2	Md	8706,1	0,17	214,4	8706,1	0,29	325,1
	QO	10/10			10/10		
3	Md	4015	0,40	392,4	4015	0,44	452,6
	QO	10/10			10/10		
4	Md	6120	0,18	159	6120	1,01	1170,5
	QO	10/10			10/10		
5	Md	12400	8,46	6173,6	12399	10,92	13031,5
	QO	10/10			9/10		
6	Md	10595,7	68,38	50165	10585,6	38,56	50125,9
	QO	0/10			0/10		
7	Md	16510,9	77,07	47055,2	16492,8	42,57	50985,2
	QO	2/10			1/10		

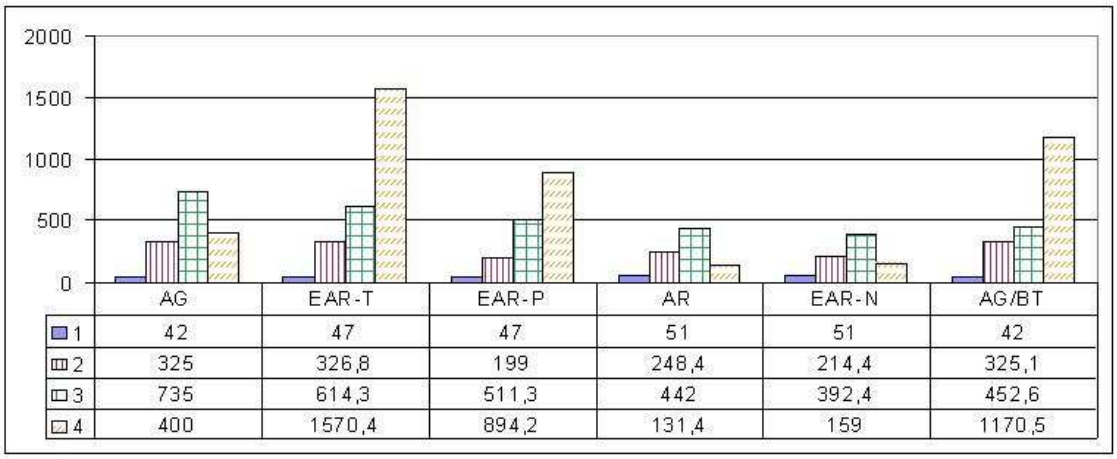
Tabela 29: Resultado do experimento 2 para os algoritmos

	Experimento 2											
	AG		EAR-T		EAR-P		AR		EAR-N		AG/BT	
Cen	*Gap%*	Ts	*Gap%*	Ts	*Gap%*	Ts	*Gap%*	Ts	*Gap%*	Ts	*Gap%*	Ts
2.1	0,558	189,969	0,656	241,36	0,213	216,657	**0,455**	**193,015**	0,394	233,625	1,046	44,625
2.2	0,101	158,437	0,202	228,674	0,091	207,51	**0,202**	**180,86**	0,101	230,32	0,610	67,83
2.3	0,234	190,94	0,242	240,82	0,301	215,22	**0,119**	**194,66**	0,119	234,54	0,119	69,17
2.4	0,954	192,54	0,988	245,26	1,126	219,795	**0,699**	**194,965**	1,037	231,665	1,557	67,28
2.5	0,277	191,395	0,288	241,735	0,217	216,505	**0,217**	**194,475**	0,369	236,385	0,329	67,56
2.6	0,154	192,16	0,049	246,46	0,154	220,05	**0,049**	**195,69**	0,049	237,17	0,154	69,21
2.7	0,584	194,305	0,334	243,36	0,477	220,02	**0,557**	**198,515**	0,539	242,17	0,807	69,065
2.8	0,000	193,375	0,171	244,31	0,000	219,055	**0,000**	**195,79**	0,000	238,49	0,092	68,945
2.9	0,516	193,5	0,570	243,39	0,743	219,62	**0,516**	**195,33**	0,273	238,06	0,834	68,56
2.10	0,283	192,78	0,553	240,84	0,283	217,84	**0,000**	**193,43**	0,283	234,64	0,291	69,53
Md	**0,366**		**0,405**		**0,361**		**0,281**		**0,316**		**0,584**	
2.11	0,122	193,835	0,122	191,225	0,122	197,91	**0,122**	**205,605**	0,122	280,085	0,122	71,545
2.12	0,235	187,35	0,221	180,13	0,209	183,65	**0,235**	**186,14**	0,235	256,85	0,613	68,22
Md	**0,178**		**0,171**		**0,165**		**0,178**		**0,178**		**0,368**	
2.13	0,000	171,6	0,000	163,65	0,000	178,81	**0,000**	**200,35**	0,000	325,34	0,301	72,05
2.14	0,269	185,22	0,272	162,44	0,351	183,12	**0,179**	**197,97**	0,280	330,38	0,288	72,35
Md	**0,135**		**0,136**		**0,176**		**0,089**		**0,140**		**0,295**	
MdT	**0,306**		**0,333**		**0,306**		**0,239**		**0,271**		**0,512**	

As Figuras 55 e 56 ilustram o desempenho dos algoritmos no experimento 1, avaliados pela média de QA. Já as Figuras 57 e 58 ilustram o desempenho dos algoritmos no experimento 2, avaliados pelo Gap%.

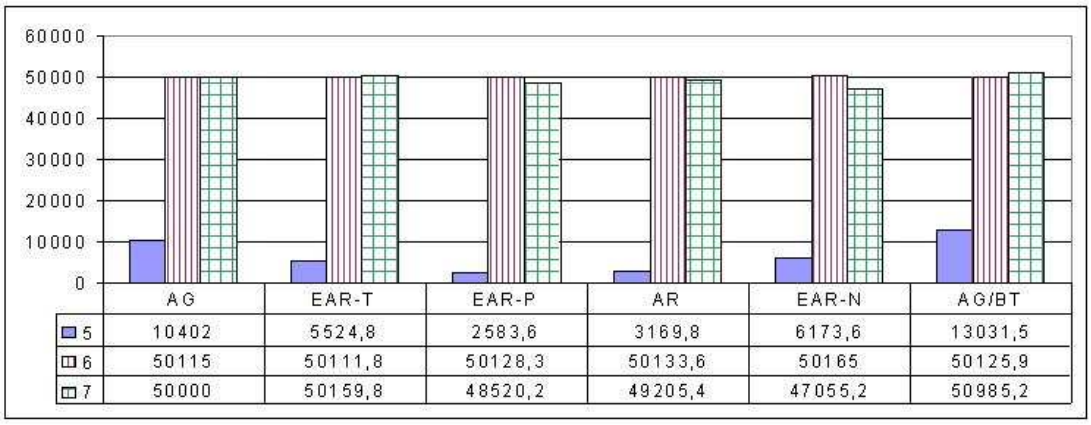
5.2.2 Análise dos resultados

Analisando o experimento 1, nos cenários 1 e 2, identifica-se um comportamento similar entre os algoritmos, pois todos obtiveram 100% de eficácia e excelente eficiência,



	AG	EAR-T	EAR-P	AR	EAR-N	AG/BT
1	42	47	47	51	51	42
2	325	326,8	199	248,4	214,4	325,1
3	735	614,3	511,3	442	392,4	452,6
4	400	1570,4	894,2	131,4	159	1170,5

Figura 55: Média de QA no experimento 1, cenários de 1 a 4



	AG	EAR-T	EAR-P	AR	EAR-N	AG/BT
5	10402	5524,8	2583,6	3169,8	6173,6	13031,5
6	50115	50111,8	50128,3	50133,6	50165	50125,9
7	50000	50159,8	48520,2	49205,4	47055,2	50985,2

Figura 56: Média de QA no experimento 1, cenários de 5 a 7

apesar do pequeno acréscimo de Ts nos algoritmos AG e AG/BT. No cenário 3, todos os algoritmos mantiveram os 100% de eficácia e boa eficiência, apesar de apresentarem um acréscimo de Ts em relação aos cenários 1 e 2. Destaca-se a diferença do Ts do AG (0,84s) em relação aos demais algoritmos (0,46s em média). No cenário 4, manteve-se a eficácia de 100% de todos os algoritmos, mas a eficiência teve variações. Destaca-se positivamente o AR e o EAR-N, que gastaram, respectivamente, 0,13s e 0,18s de Ts, além de 131,4 e 159 de Aval. No cenário 5, o AG/BT teve 90% de eficácia e os demais algoritmos 100%. Quanto à eficiência, destaca-se positivamente o EAR-P, que obteve os Ts e Aval mais baixos.

No cenário 6, experimento 1, nenhum dos algoritmos alcançou o ótimo global nas 10 execuções, provavelmente, por ser um problema mais complexo e pela condição de parada de 50000 Aval. Como o objetivo é fazer uma análise de impacto do AAP no AG, o cenário 6 demonstra, pela média da MFO, que os AAPs tiveram os melhores resultados, com destaque para o EAR-P com média de 10599 Aval ($Gap\% = 0{,}17\%$). Destaca-se

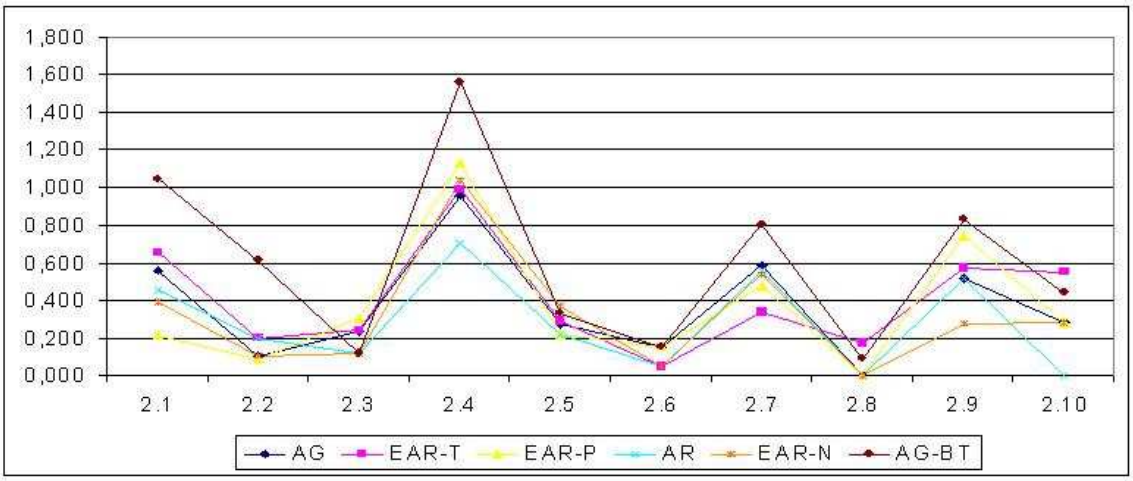


Figura 57: Gap% no experimento 2, cenários 2.1 a 2.10

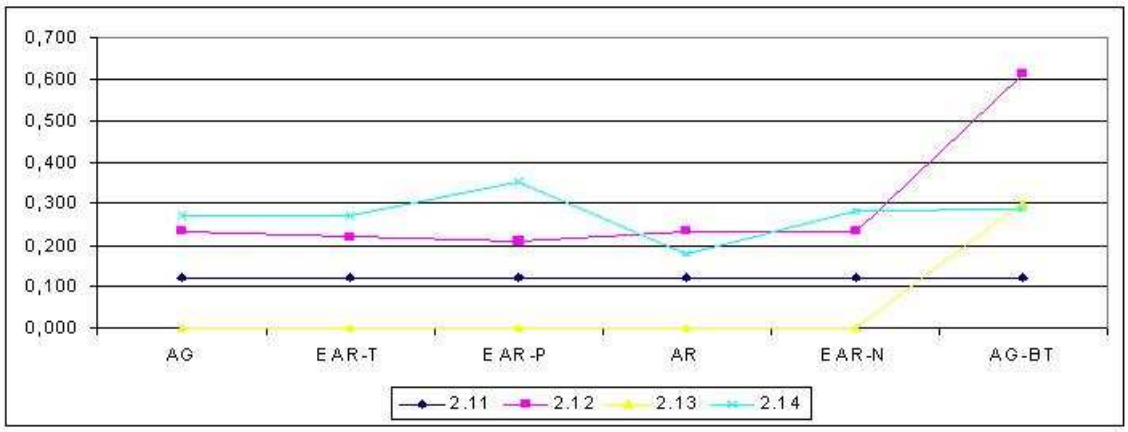


Figura 58: Gap% no experimento 2, cenários 2.11 a 2.14

negativamente o desempenho do AG/BT, que teve, nesse cenário, resultado pior que o AG, provavelmente causado pela convergência prematura para mínimos locais oriunda do acréscimo da busca local. No cenário 7, quatro algoritmos alcançaram o ótimo global em uma das dez execuções, o EAR-P com 20% de eficácia, o AR com 10%, o EAR-N com 20% e o AG/BT com 10%. Analisando a média da MFO, os AAPs tiveram os melhores resultados: o EAR-P foi o melhor (Gap = 0,12%), seguido de EAR-N (Gap = 0,16%), AR (Gap = 0,22%) e EAR-T (Gap = 0,26%). O AG/BT foi melhor que o AG, mas foi pior do que os AAPs.

No experimento 2, o EAR-P foi melhor nas instâncias com α = 0,50; no entanto, o AR foi o melhor na maioria das outras e, por isso, obteve a melhor média do experimento. Nas instâncias com coeficiente mais alto (α = 0,75), o AR teve o melhor desempenho entre os cenários.

Os resultados do AG, apresentados neste trabalho, foram melhores do que os apre-

sentados no trabalho de Khuri, Back e Heitkotter (1994). Acredita-se que a configuração atribuída para o AG, neste trabalho, apesar de similar, favoreceu o desempenho. Destaca-se que as configurações usadas neste presente trabalho foram baseadas em Hoff, Løkketangen e Mittet (1996).

Os resultados apresentados pelo AAP não são os melhores da literatura, já que existem propostas mais específicas e que têm resultados melhores (FLESZAR; HINDI, 2009); no entanto, atenderam ao objetivo de mostrar que o acoplamento do AAP no AG traz benefícios, que são tão bons, ou melhores, do que o algoritmo de estratégia similar AG/BT.

6 Considerações Finais e Trabalhos Futuros

Após a apresentação do trabalho e análise dos resultados, a seção 6.1 apresenta algumas conclusões sobre a proposta e a seção 6.2 direciona os trabalhos futuros com sugestões de aprimoramento, expansão do AAP e aplicações em novas áreas.

6.1 Conclusão

O AG é uma das metaheurísticas mais usadas em problemas de otimização complexos, dado a simplicidade de formulação e a adaptabilidade às diversas classes de problemas. No entanto, os AGs apresentam dificuldades de convergência em alguns casos, tanto na velocidade quanto na qualidade da solução final encontrada (GEN; CHENG, 1997; MICHALEWICZ, 1996).

Por isso, um grande número de trabalhos são desenvolvidos para melhorar os AGs (CHAINATE; THAPATSUWAN; PONGCHAROEN, 2007; WU et al., 2004; RONG-LONG; KOZO, 2005; PARK et al., 2000; RAJAN; MOHAN; MANIVANNAN, 2002). Todavia, poucas foram as melhorias de aplicabilidade mais ampla apresentadas na literatura recentemente (CHANG; CHEN; FAN, 2008; MATHIAS; WHITLEY, 1992). A maioria das propostas são dependentes do problema ou, no máximo, a uma classe de problemas, limitando assim a área de impacto da melhoria.

Durante as iterações dos AGs algumas informações podem ser perdidas ou ter um baixo aproveitamento, pois fatores como a pressão de seleção e os operadores de cruzamento e mutação podem retirar informações, provocar degeneração nas estruturas do indivíduo ou ignorar arranjos genéticos.

Com o intuito de aproveitar melhor as informações e sugerir um AG com boa aplicabilidade, este trabalho propõe o Algoritmo Auxiliar Paralelo (AAP), que é um algoritmo auxiliar executado em um fluxo paralelo ao do AG e que recombina cromossomos para maximizar o aproveitamento das informações presentes nos indivíduos, oriundos das po-

pulações criadas pelo AG ou gerados pelo operador do AAP. Como resultado, o módulo pode gerar indivíduos artificiais mais aptos, que são inseridos na nova geração e manipulados pelo AG na iteração seguinte.

Para testar o desempenho do algoritmo proposto (AAP), e seus operadores, quando acoplados ao AG, optou-se por dois problemas *benchmark*. O primeiro de minimização da função *Rastrigin*, reconhecidamente um problema muito difícil por ter um grande espaço de busca, ser não linear e com alto grau de multimodalidade (DIGALAKIS; MARGARITIS, 2000, 2002), e o segundo o da Mochila Multidimensional (*Multidimensional Knapsack Problem*), por ser um problema altamente combinatório e multidimensional. Com isso, foi possível medir o desempenho da proposta em dois tipos diferentes de problema: otimização de função multimodal não restritiva e otimização combinatória restritiva.

Considerando que existem excelentes algoritmos para os problemas citados, o AAP não objetivou ter o melhor desempenho, mas se apresentou como uma proposta de acoplamento benéfica aos Algoritmos Genéticos. Em vista disso, usou-se também como base comparativa um algoritmo híbrido AG-BT, que tem objetivo similar ao AAP - alimentar os AGs de bons indivíduos - e estratégia de solução parecida para o problema, um balanço entre busca local e global.

Foram construídos ao todo 39 cenários, sendo 18 para a função *Rastrigin* e 21 no problema da Mochila Muldimensional. Os resultados apresentados mostram o AAP com uma boa ferramenta para auxiliar os AGs a melhorarem o desempenho de convergência. Percebe-se também nos experimentos feitos que houve uma considerável melhoria na velocidade de convergência sem prejudicar a qualidade da solução final.

Entre os operadores do AAP não houve o melhor para todos os casos, pois cada cenário favoreceu determinada característica. No entanto, os operadores EAR-PA e AR obtiveram os melhores resultados. Destaca-se, ainda, que o EAR-N teve um desempenho bem melhor no PMM do que na função Rastrigin e que o EAR-T foi o pior entre os operadores, devido a estratégia radical e mudanças bruscas no indivíduo.

Na comparação entre o AAP e o algoritmo híbrido (AG-BT), o AAP auxiliou melhor o AG durante o processo de evolução e, por isso, apresentou melhores resultados.

Por fim, constata-se que o melhor aproveitamento das informações genéticas é uma boa estratégia de melhoria dos AGs e, por isso, uma área promissora de pesquisa. Por fim, destaca-se a estrutura modular do AAP, que permite várias combinações e aprimoramentos.

6.2 Trabalhos Futuros

Como descrito na seção 6.1, este trabalho pode servir de base para outros que queiram investigar formas de melhor aproveitar as informações genéticas. Assim, sugerem-se como trabalhos futuros pesquisas sobre estratégias de aproveitamento da informação genética e a construção de novos operadores para o AAP, com diferentes estratégias de Divisão do Material Genético, Coleta, Manipulação Genética e Transferência. Como indicação para diminuir o custo computacional, sugere-se a aplicação do AAP em intervalos de algumas gerações e/ou a substituição dos operadores dos AGs pelo AAP.

Dados os resultados, sugere-se também a aplicação do AAP a outras classes de problemas, incluindo problemas multiobjetivo, e o acoplamento do AAP a outras heurísticas populacionais, como os Algoritmos Evolucionários e o PSO (*Particle Swarm Optimization*).

Referências

ABBASS, H. Self-adaptive pareto differential evolution. In: *Proceedings of the IEEE Congress on Evolutionary Computation*. Honolulu, USA: IEEE Press, 2002. p. 831–836.

ACHICHE, S.; BARON, L.; BALAZINSKI, M. Real/binary-like coded genetic algorithm to automatically generate fuzzy knowledge bases. In: *Proceedings of 4th International Conference on Control and Automation*. [S.l.: s.n.], 2003.

ACKLEY, D. H. *A connectionist machine for genetic hillclimbing*. Boston: Kluwer, 1987.

ANGELINE, P. J. Adaptive and self-adaptive evolutionary computations. In: *Computational Intelligence: A Dynamic Systems Perspective*. [S.l.]: IEEE Press, 1995. p. 152–163.

ASOH, H.; MÜHLENBEIN, H. On the mean convergence time of evolutionary algorithms without selection and mutation. In: DAVIDOR, Y.; SCHWEFEL, H.-P.; MÄNNER, R. (Ed.). *Parallel problem solving from nature: PPSN III*. Berlin: Springer–Verlag, 1994. p. 88–97.

ATMAR, W. The philosophical errors that plague both evolutionary theory and simulated evolutionary programming. In: *Proceedings of the First Annual Conference on Evolutionary Programming*. San Diego, CA, United States: Evolutionary Programming Society, 1992. p. 27–34.

BäCK, T. Optimal mutation rates in genetic search. In: *Proceedings of the 5th International Conference on Genetic Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993. p. 2–8.

BACK, T. *Evolutionary algorithms in theory and practice*. [S.l.]: Oxford University Press, 1996.

BAMBHA, N. K. et al. Systematic integration of parameterized local search into evolutionary algorithms. *IEEE Trans. Evolutionary Computation*, v. 8, n. 2, p. 137–155, 2004.

BANZHAF, W. The "molecular" traveling salesman. *Biol. Cybern.*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, v. 64, n. 1, p. 7–14, 1990.

BARCELLOS, J. C. H. d. *Algoritmos genéticos adaptativos: um estudo comparativo*. Dissertação (Dissertação de Mestrado) — Escola Politécnica da Univesidade de São Paulo, São Paulo, Brasil, 2000.

BARRA, T. V. *Um Ambiente Evolutivo para Apoio ao Projeto de Antenas de Microfita*. Dissertação (Mestrado) — Universidade Estadual de Campinas - Faculdade de Engenharia Elétrica e de Computação, Campinas, 2007.

BAUM, E. B.; BONEH, D.; GARRETT, C. On genetic algorithms. In: *COLT '95: Proceedings of the eighth annual conference on Computational learning theory*. New York, NY, USA: ACM, 1995. p. 230–239. ISBN 0-89791-723-5.

BäCK, T.; HOFFMEISTER, F.; SCHWEFEL, H.-P. A survey of evolution strategies. In: *Proceedings of the Fourth International Conference on Genetic Algorithms*. [S.l.]: Morgan Kaufmann, 1991. p. 2–9.

BELEW, R. K.; MCINERNEY, J.; SCHRAUDOLPH, N. N. Evolving networks: Using the genetic algorithm with connectionist learning. In: *In Artificial Life II*. [S.l.]: Addison-Wesley, 1991. p. 511–547.

BEYER, H.-G. On the ëxplorative poweröf es/ep-like algorithms. In: *EP '98: Proceedings of the 7th International Conference on Evolutionary Programming VII*. London, UK: Springer-Verlag, 1998. p. 323–334. ISBN 3-540-64891-7.

BLICKLE, T.; THIELE, L. *A Comparison of Selection Schemes Used in Genetic Algorithms*. Gloriastrasse 35, 8092 Zurich, Switzerland, 1995.

BLICKLE, T.; THIELE, L. A comparison of selection schemes used in evolutionary algorithms. *Evol. Comput.*, MIT Press, Cambridge, MA, USA, v. 4, n. 4, p. 361–394, 1996.

BOOKER, L. B. et al. The handbook of evolutionary computation. In: _____. Philadelphia, PA: IOP Publishing Ltd. and Oxford University Press, 1997. cap. Recombination, p. C3.3:1–C3.3:27.

BURKE, E. K.; GUSTAFSON, S.; KENDALL, G. Diversity in genetic programming: an analysis of measures and correlation with fitness. *IEEE Trans. Evolutionary Computation*, v. 8, n. 1, p. 47–62, 2004.

CALDAS, A. P. F. et al. Detecção do provírus da imunodeficiência felina em gatos domésticos pela técnica de reação em cadeia da polimerase. *Pesquisa Veterinária Brasileira*, v. 20, n. 1, 2000. ISSN 0100-736X.

CAMARGO, G. d. M. *Controle da pressão seletiva em algoritmo genético aplicado a otimização de demanda em infra-estrutura aeronáutica*. Dissertação (Dissertação de Mestrado) — Escola Politécnica (EP), USP, 2006.

CAMILO, C. G.; YAMANAKA, K. Assisted recombination: Accelerating genetic improvements of populations in a genetic algorithm. In: IASTED. *Proceedings of the 10th IASTED International Conference on Intelligent Systems and Control*. [S.l.], 2007.

CAMILO, C. G.; YAMANAKA, K. Parallel auxiliary algorithm to improve the binary-coded genetic algorithms. *Learning and Nonlinear Models*, v. 6, n. 2, 2009.

CARVALHO, D. R.; FREITAS, A. A. A hybrid decision tree/genetic algorithm method for data mining. *Inf. Sci.*, Elsevier Science Inc., New York, NY, USA, v. 163, n. 1-3, p. 13–35, 2004. ISSN 0020-0255.

CERAVOLO, F.; FELICE, M.; PIZZUTI, S. Combining back-propagation and genetic algorithms to train neural networks for ambient temperature modeling in italy. In: *EvoWorkshops '09: Proceedings of the EvoWorkshops 2009 on Applications of Evolutionary Computing*. Berlin, Heidelberg: Springer-Verlag, 2009. p. 123–131. ISBN 978-3-642-01128-3.

CHAINATE, W.; THAPATSUWAN, P.; PONGCHAROEN, P. A new heuristic for improving the performance of genetic algorithm. *Academy of Science, Engineering and Technology*, v. 21, n. 1, p. 217–220, 2007.

CHAKRABORTY, U. K.; JANIKOW, C. Z. An analysis of gray versus binary encoding in genetic search. *Information Sciences*, v. 156, p. 253–269, 2003.

CHANG, P.; CHEN, S.; FAN, C. Mining gene structures to inject artificial chromosomes for genetic algorithm in single machine scheduling problems. *Applied Soft Computing*, v. 8, n. 1, p. 767–777, 2008.

CHANG, P. et al. Genetic algorithm integrated with artificial chromosomes for multi-objective flowshop scheduling problems. *Applied Mathematics and Computation*, v. 205, n. 2, p. 550–561, 2008.

CHANG, P.-C.; CHEN, S.-H.; LIN, K.-L. Two-phase sub population genetic algorithm for parallel machine-scheduling problem. *Expert Syst. Appl.*, v. 29, n. 3, p. 705–712, 2005.

CHANG, P.-C.; CHEN, S.-H.; LIU, C.-H. Sub-population genetic algorithm with mining gene structures for multiobjective flowshop scheduling problems. *Expert Syst. Appl.*, Pergamon Press, Inc., Tarrytown, NY, USA, v. 33, n. 3, p. 762–771, 2007. ISSN 0957-4174.

CHANG, P.-C.; LAI, C.-Y.; LAI, K. R. A hybrid system by evolving case-based reasoning with genetic algorithm in wholesaler's returning book forecasting. *Decision Support Systems*, v. 42, n. 3, p. 1715–1729, 2006.

CHANG, P.-C.; WANG, Y.-W.; LIU, C.-H. New operators for faster convergence and better solution quality in modified genetic algorithm. In: *Advances in Natural Computation, First International Conference*. Changsha, China: Springer, 2005. (Lecture Notes in Computer Science, v. 3611), p. 983–991. ISBN 3-540-28325-0.

CHEN, Y. et al. Solving deceptive problems using a genetic algorithm with reserve selection. In: *IEEE Congress on Evolutionary Computation*. [S.l.: s.n.], 2008. p. 884–889.

CHIU, H.-P. et al. A tabu genetic algorithm with search area adaptation for the job-shop scheduling problem. In: *AIKED'07: Proceedings of the 6th Conference on 6th WSEAS Int. Conf. on Artificial Intelligence, Knowledge Engineering and Data Bases*. Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society (WSEAS), 2007. p. 76–80. ISBN 978-960-8457-59-1.

CHU, P. C.; BEASLEY, J. E. A genetic algorithm for the multidimensional knapsack problem. *Journal of Heuristics*, v. 4, n. 1, p. 63–86, June 1998.

CNNHEALTH.COM. *"Cancer-free" baby born in London*. Jan 2009. Disponível em: <http://www.cnn.com/2009/HEALTH/01/09/uk.cancerfree.baby/index.html>.

CORDóN, O.; ANEGóN, F. de M.; ZARCO, C. A new evolutionary algorithm combining simulated annealing and genetic programming for relevance feedback in fuzzy information retrieval systems. *Soft Comput.*, v. 6, n. 5, p. 308–319, 2002.

DAI, C.; ZHU, Y.; CHEN, W. Adaptive probabilities of crossover and mutation in genetic algorithms based on cloud model. In: *Information Theory Workshop, 2006. ITW '06 Chengdu. IEEE*. [S.l.: s.n.], 2006. p. 710–713.

DAVIDOR, Y. Epistasis variance: A viewpoint on ga-hardness. In: *Proceedings of the First Workshop on Foundations of Genetic Algorithms*. Bloomington Campus, Indiana, USA: Morgan Kaufmann, 1990. p. 23–35.

DAVIS, L. Applying adaptive algorithms to epistatic domains. In: *IJCAI'85: Proceedings of the 9th international joint conference on Artificial intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1985. p. 162–164.

DAVIS, L. (Ed.). *Handbook of Genetic Algorithms*. [S.l.]: Van Nostrand Reinhold, 1991.

DENGIZ, B.; ALTIPARMAK, F. Local search genetic algorithm for optimization of highly reliable communications networks. *IEEE Transactions on Evolutionary Computation*, v. 1, p. 179–188, 1997.

DERIS, S. et al. Incorporating constraint propagation in genetic algorithm for university timetable planning. *Engineering Applications of Artificial Intelligence*, v. 13, n. 2, p. 241–253, 1999.

DIGALAKIS, J. G.; MARGARITIS, K. G. On benchmarking functions for genetic algorithms. *Inter. J. Computer Math*, v. 00, p. 1–27, 2000.

DIGALAKIS, J. G.; MARGARITIS, K. G. An experimental study of benchmarking functions for genetic algorithms. *Int J Comput Math*, v. 79, n. 4, p. 403–416, 2002.

DIJK, S. van. *Genetic Algorithms for Map Labeling*. Tese (Ph.D. tese) — Utrecht University, 2001.

DUNKER, T.; RADONS, G.; WESTKAMPER, E. Combining evolutionary computation and dynamic programming for solving a dynamic facility layout problem. *European Journal of Operational Research*, v. 165, n. 1, p. 55–69, August 2005.

EIBEN, A. E.; SCHIPPERS, C. A. On evolutionary exploration and exploitation. *Fundam. Inf.*, IOS Press, Amsterdam, The Netherlands, The Netherlands, v. 35, n. 1-4, p. 35–50, 1998.

EIBEN Ágoston E. et al. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, v. 3, p. 124–141, 2000.

ELMIHOUB, T. et al. Performance of hybrid genetic algorithms incorporating local search. In: HORTON, G. (Ed.). *18th European Simulation Multiconference (ESM2004)*. Magdeburg, Germany: [s.n.], 2004. p. 154–160.

ELMIHOUB, T. et al. Hybrid genetic algorithms - a review. *Engineering Letters*, v. 13, n. 2, p. 124–137, 2006. ISSN: 1816-093X.

ESHELMAN, L. J.; CARUANA, R. A.; SCHAFFER, J. D. Biases in the crossover landscape. In: *Proceedings of the third international conference on Genetic algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1989. p. 10–19. ISBN 1-55860-006-3.

ESPINOZA, F. P.; MINSKER, B. S.; GOLDBERG, D. E. Performance evaluation and population reduction for a self adaptive hybrid genetic algorithm (sahga). In: *GECCO*. [S.l.: s.n.], 2003. p. 922–933.

FAN, H.; ZENG, G.; LI, X. Crawling strategy of focused crawler based on niche genetic algorithm. In: *Dependable, Autonomic and Secure Computing, 2009. DASC '09. Eighth IEEE International Conference on*. [S.l.: s.n.], 2009. p. 591–594.

FELIX, J. M. G. *Conceitos Ecológicos*. Jan 2010. Disponível em: <http://www.bioeducacao.com/3%BA-FORM-GERAL.php>.

FERREIRA, C. Gene expression programming: A new adaptive algorithm for solving problems. *Complex Systems*, v. 13, n. 2, p. 87–129, 2001.

FISHER, R. A. *The Genetical Theory of Natural Selection*. [S.l.]: Clarendon, 1930.

FLESZAR, K.; HINDI, K. S. Fast, effective heuristics for the 0-1 multi-dimensional knapsack problem. *Comput. Oper. Res.*, Elsevier Science Ltd., Oxford, Uk, v. 36, n. 5, p. 1602–1607, 2009.

FLEURENT, C.; FERLAND, J. A. Genetic hybrids for the quadratic assignment problem. In: *DIMACS Series in Mathematics and Theoretical Computer Science*. [S.l.]: American Mathematical Society, 1993. p. 173–187.

FOGEL, D. B.; ATMAR, J. Comparing genetic operators with gaussian mutations in simulated evolutionary processes using linear systems. *Biological Cybernetics*, v. 63, p. 111–114, 1990.

FOGEL, D. B.; BÄCK, T.; MICHALEWICZ, Z. *Evolutionary Computation 2 - Advanced Algorithms and Operators*. Bristol - Philadelphia: Institute of Physics Publishing, 2000. 6–7 p. ISBN 0-7503-0665-3.

FOGEL, L. J.; OWENS, A. J.; WALSH, M. J. *Artificial Intelligence through Simulated Evolution*. New York, USA: John Wiley, 1966.

FREDERICK, W. G.; SEDLMEYER, R. L.; WHITE, C. M. The hamming metric in genetic algorithms and its application to two network problems. In: *SAC '93: Proceedings of the 1993 ACM/SIGAPP symposium on Applied computing*. New York, NY, USA: ACM, 1993. p. 126–130. ISBN 0-89791-567-4.

FREITAG, K.; HILDEBRAND, L.; MORAGA, C. Quaternary coded genetic algorithms. In: *Proceedings of Twenty Ninth IEEE International Symposium on Multiple-Valued Logic*. [S.l.: s.n.], 1999.

FREITAS, C. C. et al. Uma ferramenta baseada em algoritmos genéticos para a geração de tabela de horário escolar. In: *Sétima Escola Regional de Computação Bahia-Sergipe*. Vitória da Conquista: [s.n.], 2007.

FREITAS, F. et al. Aplicação de metaheurísticas em problemas da engenharia de software: Revisão de literatura. In: *Anais do II Congresso Tecnológico Infobrasil*. [S.l.: s.n.], 2009.

FRENCH, A. P.; ROBINSON, A. C.; WILSON, J. M. Using a hybrid genetic-algorithm/branch and bound approach to solve feasibility and optimization integer programming problems. *Journal of Heuristics*, Kluwer Academic Publishers, Hingham, MA, USA, v. 7, n. 6, p. 551–564, 2001. ISSN 1381-1231.

FU, Z. Using genetic algorithms-based approach for better decision trees: A computational study. In: *DS '02: Proceedings of the 5th International Conference on Discovery Science*. London, UK: Springer-Verlag, 2002. p. 390–397. ISBN 3-540-00188-3.

GAREY, M. R.; JOHNSON, D. S. *Computers and Intractability : A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. [S.l.]: W. H. Freeman, 1979. Paperback.

GARNIER, J.; KALLEL, L.; SCHOENAUER, M. Rigorous hitting times for binary mutations. *Evol. Comput.*, MIT Press, Cambridge, MA, USA, v. 7, n. 2, p. 173–203, 1999. ISSN 1063-6560.

GEN, M.; CHENG, R. *Genetic Algorithms and Engineering Design.* [S.l.]: Wiley-Interscience, 1997.

GLICKMAN, M.; SYCARA, K. Reasons for premature convergence of self-adapting mutation rates. In: *Proceedings of the 2000 Congress on Evolutionary Computation*. La Jolla, CA, USA: [s.n.], 2000. p. 62–69.

GLOVER, F. Future paths for integer programming and links to artificial intelligence. *Comput. Oper. Res.*, Elsevier Science Ltd., Oxford, UK, UK, v. 13, n. 5, p. 533–549, 1986. ISSN 0305-0548.

GOLDBERG, D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. [S.l.]: Addison-Wesley Professional, 1989. Hardcover. ISBN 0201157675.

GOLDBERG, D. E. Construction of high-order deceptive functions using low-order walsh coefficients. *Ann. Math. Artif. Intell.*, v. 5, n. 1, p. 35–47, 1992.

GOLDBERG, D. E.; DEB, K.; HORN, J. *Massive Multimodality, Deception, and Genetic Algorithms*. [S.l.], 1992.

GOLDBERG, D. E.; LINGLE, R. Alleles, loci, and the tsp. In: *1st Int. Conf. on Genetic Algorithms*. [S.l.: s.n.], 1985. p. 154–159.

GOLDBERG, D. E.; RICHARDSON, J. Genetic algorithms with sharing for multimodal function optimization. In: *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*. Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 1987. p. 41–49. ISBN 0-8058-0158-8.

GOLDBERG, D. E.; SEGREST, P. Finite markov chain analysis of genetic algorithms. In: *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*. Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 1987. p. 1–8.

GOLDBERG, D. E.; VOESSNER, S. Optimizing global-local search hybrids. In: *PROCEEDINGS OF THE GENETIC AND EVOLUTIONARY COMPUTATION CONFERENCE*. [S.l.]: Morgan Kaufmann, 1999. p. 220–228.

GREFENSTETTE, J. Optimization of control parameters for genetic algorithms. *IEEE Trans. Syst. Man Cybern.*, IEEE Press, Piscataway, NJ, USA, v. 16, n. 1, p. 122–128, 1986. ISSN 0018-9472.

GREFENSTETTE, J. J. *Genetic Algorithms and Simulated Annealing*. London: Pitman, 1987.

GüLER, I.; POLAT, H.; ERGüN, U. Combining neural network and genetic algorithm for prediction of lung sounds. *J. Med. Syst.*, Plenum Press, New York, NY, USA, v. 29, n. 3, p. 217–231, 2005. ISSN 0148-5598.

HAJELA, P.; LIN, C.-Y. Real versus binary coding in genetic algorithms: a comparative study. Civil-Comp press, Edinburgh, UK, UK, p. 77–83, 2000.

HAMZAçEBI, C. Improving genetic algorithms' performance by local search for continuous function optimization. *Applied Mathematics and Computation*, v. 196, n. 1, p. 309–317, 2008.

HANDA, H. et al. Coevolutionary genetic algorithm with effective exploration and exploitation of useful schemata. In: *Proceedings of ICONIP'97*. [S.l.: s.n.], 1997. p. 424–427.

HARADA, K.; IKEDA, K.; KOBAYASHI, S. Hybridization of genetic algorithm and local search in multiobjective function optimization: recommendation of ga then ls. In: *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*. New York, NY, USA: ACM, 2006. p. 667–674. ISBN 1-59593-186-4.

HARPER, J.; RODECK, C. Preimplantation genetic diagnosis. *Progress in Obstetrics and Gynaecology*, v. 15, p. 33–43, 2002. Disponível em: <http://www.conception-acu.com/>.

HAUPT, R. L.; HAUPT, S. E. *Practical Genetic Algorithms*. 2st. ed. [S.l.]: Wiley-Interscience, 2004. ISBN 0-471-45565-2.

HAUPT, R. L.; HAUPT, S. E. *Practical Genetic Algorithms*. 2st. ed. [S.l.]: Wiley-Interscience, 2004. ISBN 0-471-45565-2.

HE, R.-S.; HWANG, S.-F. Damage detection by an adaptive real-parameter simulated annealing genetic algorithm. *Comput. Struct.*, Pergamon Press, Inc., Elmsford, NY, USA, v. 84, n. 31-32, p. 2231–2243, 2006. ISSN 0045-7949.

HERRERA, F.; LOZANO, M. Adaptation of genetic algorithm parameters based on fuzzy logic controllers. In: *Genetic Algorithms and Soft Computing*. [S.l.]: Physica-Verlag, 1996. p. 95–125.

HILL, R. R. A monte-carlo study of genetic algorithm initial population generation methods. In: *WSC '99: Proceedings of the 31st conference on Winter simulation*. New York, NY, USA: ACM, 1999. p. 543–547.

HINTERDING, R.; MICHALEWICZ, Z.; EIBEN, A. E. Adaptation in evolutionary computation: A survey. In: *In Proceedings of the Fourth International Conference on Evolutionary Computation (ICEC 97*. [S.l.]: IEEE Press, 1997. p. 65–69.

HOFF, A.; LøKKETANGEN, A.; MITTET, I. Genetic algorithms for 0/1 multidimensional knapsack problems. In: *Proceedings Norsk Informatikk Konferanse, NIK '96*. [S.l.: s.n.], 1996.

HOLLAND, J. H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Cambridge, MA, USA: MIT Press, 1992. ISBN 0262082136.

HONG, T.-P. et al. Evolution of appropriate crossover and mutation operators in a genetic process. *Applied Intelligence*, Kluwer Academic Publishers, Hingham, MA, USA, v. 16, n. 1, p. 7–17, 2002. ISSN 0924-669X.

HORN, J. *The Nature of Niching: Genetic Algorithms and the Evolution of Optimal, Cooperative Populations*. Tese (Phd Tesis) — Illinois Genetic Algorithms Laboratory, Champaign, IL, USA, 1997.

HUANG, M. L.; HUNG, Y. H. Combining radial basis function neural network and genetic algorithm to improve hdd driver ic chip scale package assembly yield. *Expert Syst. Appl.*, Pergamon Press, Inc., Tarrytown, NY, USA, v. 34, n. 1, p. 588–595, 2008. ISSN 0957-4174.

HUANG, Y.; TIAN, C.; FANG, W. Research on real estate portfolio optimization model based on improved niche genetic algorithm. In: *International Symposium on Information Engineering and Electronic Commerce*. [S.l.: s.n.], 2009. p. 375–378.

HUNG, J.-C. Combining a genetic algorithm and simulated annealing to design a fixed-order mixed h2/h deconvolution filter with missing observations. *J. Control Sci. Eng.*, Hindawi Publishing Corp., New York, NY, United States, v. 2008, p. 1–10, 2008. ISSN 1687-5249.

HWANG, S.-F.; HE, R.-S. Improving real-parameter genetic algorithm with simulated annealing for engineering problems. *Adv. Eng. Softw.*, Elsevier Science Ltd., Oxford, UK, UK, v. 37, n. 6, p. 406–418, 2006. ISSN 0965-9978.

ICA. *f6 Function*. Nov 2009. Disponível em: <http://www.ica.ele.puc-rio.br>.

ICLANZAN, D. Crossover: the divine afflatus in search. In: *GECCO '07: Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation*. New York, NY, USA: ACM, 2007. p. 2497–2502.

IPGO. *PGD (Diagnóstico Genético Pré-Implantacional)*. Dez 2009. Site. Disponível em: <http://www.ipgo.com.br/pgd.html>.

JANSEN, T. *On Classifications of Fitness Functions*. Dortmund, Germany, Novembro 1999.

JANSEN, T.; WEGENER, I. On the analysis of evolutionary algorithms - a proof that crossover really can help. In: *ESA '99: Proceedings of the 7th Annual European Symposium on Algorithms*. London, UK: Springer-Verlag, 1999. p. 184–193.

JANSEN, T.; WEGENER, I. Real royal road functions: where crossover provably is essential. *Discrete Appl. Math.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 149, n. 1-3, p. 111–125, 2005. ISSN 0166-218X.

JELASITY, M.; DOMBI, J. Gas, a concept on modeling species in genetic algorithms. *Artif. Intell.*, Elsevier Science Publishers Ltd., Essex, UK, v. 99, n. 1, p. 1–19, 1998. ISSN 0004-3702.

JIAO, L.; WANG, L. A novel genetic algorithm based on immunity. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, v. 30, n. 5, p. 552–561, 2000.

JIN, C.; LI, F. Fuzzy optimization method based on dynamic uncertainty restriction. In: *IITA '08: Proceedings of the 2008 Second International Symposium on Intelligent Information Technology Application*. Washington, DC, USA: IEEE Computer Society, 2008. p. 736–740. ISBN 978-0-7695-3497-8.

JIN, Y. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Comput.*, Springer-Verlag, Berlin, Heidelberg, v. 9, n. 1, p. 3–12, 2005. ISSN 1432-7643.

JONG, K. A. D. *An analysis of the behavior of a class of genetic adaptive systems.* Tese (Doutorado), Ann Arbor, MI, USA, 1975.

JONG, K. D. Perspectives on adaptation in natural and artificial systems. In: _____. [S.l.]: Oxford University Press,, 2005. cap. Genetic algorithms: a 30 year perspective.

KAKOUROU, G. et al. Preimplantation genetic diagnosis for myotonic dystrophy type 1: detection of cross-over between the gene and the linked marker apoc2. *Prenatal Diagnosis*, v. 27, n. 2, p. 111–116, 2007. Disponível em: <http://www.conception-acu.com/>.

KAUFFMAN, S. A. Adaptation on rugged fitness landscapes. *Lectures in the Sciences of Complexity*, Addison-Wesley, p. 527–618, 1989.

KENNETH; SPEARS, W. M. A formal analysis of the role of multi-point crossover in genetic algorithms. *Annals of Mathematics and Artificial Intelligence*, v. 5, p. 1–26, 1992.

KHURI, S.; BACK, T.; HEITKOTTER, J. The zero/one multiple knapsack problem and genetic algorithms. In: *Proceedings of the 1994 ACM Symposium on Applied Computing*. [S.l.]: ACM Press, 1994. p. 188–193.

KIM, K. M. et al. Binary decision tree using k-means and genetic algorithm for recognizing defect patterns of cold mill strip. In: *IEA/AIE'2004: Proceedings of the 17th international conference on Innovations in applied artificial intelligence*. [S.l.]: Springer Springer Verlag Inc, 2004. p. 341–350. ISBN 3-540-22007-0.

KIM, Y.-H.; MOON, B.-R. Distance measures in genetic algorithms. In: *Proceedings of the Genetic and Evolutionary Computation*. [S.l.]: Springer Berlin,Heidelberg, 2004. p. 400–401.

KRASNOGOR, N.; SMITH, J. A tutorial for competent memetic algorithms: model, taxonomy and design issues. *IEEE Transactions on Evolutionary Computation*, v. 9, n. 5, p. 474–488, 2005.

LACERDA, S. I. e A. P. *Infertilidade*. Dez 2009. Site. Disponível em: <http://integras.blogspot.com/2009/02/brasil-ainda-nao-tem-legislacao-para-o.html>.

LARRAñA, P. et al. Genetic algorithms for the travelling salesman problem: A review of representations and operators. *Artificial Intelligence Review*, Kluwer Academic Publishers, Norwell, MA, USA, v. 13, n. 2, p. 129–170, 1999.

LARRANAGA, P.; LOZANO, J. A. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. [S.l.]: Springer, 2001. Hardcover. ISBN 0792374665.

LAU, H. C. W. et al. A fuzzy genetic algorithm for the discovery of process parameter settings using knowledge representation. *Expert Syst. Appl.*, Pergamon Press, Inc., Tarrytown, NY, USA, v. 36, n. 4, p. 7964–7974, 2009. ISSN 0957-4174.

LAW, N. L.; SZETO, K. Y. Adaptive genetic algorithm with mutation and crossover matrices. In: *IJCAI'07: Proceedings of the 20th international joint conference on Artifical intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007. p. 2330–2333.

LEITE, L. *Fertilização "in Vitro"*. Jan 2009. Disponível em: <http://www.ghente.org/temas/reproducao/art_fiv.htm>.

LI, H.-t.; SHI, A.-s.; ZHANG, B.-s. A dyeing color matching method combining rbf neural networks with genetic algorithms. In: *SNPD '07: Proceedings of the Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*. Washington, DC, USA: IEEE Computer Society, 2007. p. 701–706. ISBN 0-7695-2909-7.

LI, J.; LI, M. An improved genetic algorithm for solving deceptive problems. In: *Proceedings of the IEEE International Conference on Granular Computing*. Beijing, China: [s.n.], 2005. p. 502–505.

LI, M.; LIN, D.; KOU, J. An investigation on niching multiple species based on population replacement strategies for multimodal functions optimization. *Soft Comput.*, Springer-Verlag, Berlin, Heidelberg, v. 14, n. 1, p. 49–69, 2009. ISSN 1432-7643.

LIEPINS, G. E.; VOSE, M. D. Deceptiveness and genetic algorithm dynamics. In: *FOGA*. [S.l.: s.n.], 1990. p. 36–50.

LINDEN, R. *Algoritmos Genéticos - Uma importante ferramenta da Inteligência Computacional*. 1st. ed. [S.l.]: Brasport, 2006.

LO, C. H. et al. Fuzzy-genetic algorithm for automatic fault detection in hvac systems. *Appl. Soft Comput.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 7, n. 2, p. 554–560, 2007. ISSN 1568-4946.

LOUIS, S. J.; LI, G. Case injected genetic algorithms for traveling salesman problems. *Inf. Sci.*, Elsevier Science Inc., New York, NY, USA, v. 122, n. 2-4, p. 201–225, 2000.

MAHFOUD, S. Crowding and preselection revisited. In: MANNER, R.; MANDERICK, B. (Ed.). *Parallel Problem Solving from Nature 2*. Amsterdan: [s.n.], 1992. p. 27–36.

MAHFOUD, S. *Niching Methods for Genetic Algorithms*. [S.l.], 1995.

MAHFOUD, S. W. *Niching Methods for Genetic Algorithms*. Tese (Phd Tesis) — Illinois Genetic Algorithms Laboratory, Champaign, IL, USA, 1995.

MAK, K.; SUN, D. A new hybrid genetic algorithm and tabu search method for yard cranes scheduling with inter-crane interference. In: IAENG. *Proceedings of the World Congress on Engineering 2009*. London, UK, 2009.

MANELA, M.; CAMPBELL, J. A. Harmonic analysis, epistasis and genetic algorithms. In: *Parallel Problem Solving from Nature 2*. Brussels, Belgium: Elsevier, 1992. p. 59–66.

MARIANO, A. S. *Geração automática de Grade Horária para a Faculdade de Engenharia Elétrica da UFU usando Algoritmos Genéticos*. Dissertação (Monografia) — Universidade Federal de Uberlândia - Faculdade de Engenharia Elétrica, Uberlândia - MG, 2007.

MATHIAS, K.; WHITLEY, D. Genetic operators, the fitness landscape and the traveling salesman problem. In: *Parallel Problem Solving from Nature*. [S.l.]: Elsevier Science Publishers, 1992. p. 219–228.

MATHIAS, K. E. et al. Staged hybrid genetic search for seismic data imaging. In: *In Proceedings of the First IEEE Conference on Evolutionary Computation*. [S.l.]: IEEE Service Center, 1994. p. 356–361.

MICHALEWICZ, Z. *Genetic algorithms + data structures = evolution programs*. Berlin: Springer-Verlag, 1996.

MING, H.; NAN, L.; XU, L. An improved niche genetic algorithm. In: *IEEE International Conference on Intelligent Computing and Intelligent Systems, 2009.* [S.l.: s.n.], 2009. v. 2, p. 291–293.

MISHRA, K. K. et al. Combining neural networks and genetic algorithms to predict and to maximize lemon grass oil production. In: *CSO '09: Proceedings of the 2009 International Joint Conference on Computational Sciences and Optimization*. Washington, DC, USA: IEEE Computer Society, 2009. p. 297–299. ISBN 978-0-7695-3605-7.

MITCHELL, M. *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: MIT Press, 1998. ISBN 0262631857.

MITCHELL, M.; FORREST, S. Genetic algorithms and artificial life. *Artificial Life*, v. 1, p. 267–289, 1994.

MITCHELL, M.; FORREST, S.; HOLLAND, J. H. The royal road for genetic algorithms: Fitness landscapes and ga performance. In: *Proceedings of the First European Conference on Artificial Life*. [S.l.]: MIT Press, 1991. p. 245–254.

MITCHELL, M.; FORREST, S.; HOLLAND, J. H. The royal road for genetic algorithms: Fitness landscapes and ga performance. In: VARELA, F. J.; BOURGINE, P. (Ed.). *Towards a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*. Cambridge, MA: MIT Press, 1992. p. 245–254.

MOGNON, V. R. *Algoritmos Genéticos Aplicados na Otimização de Antenas*. Dissertação (Mestrado) — Universidade Federal do Paraná - Faculdade de Engenharia Elétrica, Curitiba, 2004.

MOLE, V. L. D. *Algoritmos Genéticos - Uma Abordagem Paralela Baseada em Populações Cooperantes*. Dissertação (Mestrado) — Universidade Federal de Santa Catarina - Departamento de Informática e Estatística, Florianópolis - SC, 2002.

MONTANA, D. J.; BERANEK, B.; INC, N. Neural network weight selection using genetic algorithms. In: *Intelligent Hybrid Systems*. [S.l.]: Wiley, 1995. p. 85–104.

MONTIEL, O. et al. Human evolutionary model: A new approach to optimization. *Inf. Sci.*, v. 177, n. 10, p. 2075–2098, 2007.

MOSCATO, P. *On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms*. Pasadena, California, 1989.

MüHLENBEIN, H.; SCHLIERKAMP-VOOSEN, D. Predictive models for the breeder genetic algorithm – i. continuous parameter optimization. *Evol. Comput.*, MIT Press, Cambridge, MA, USA, v. 1, n. 1, p. 25–49, 1993.

MUHLENBEIN, H.; SCHOMISCH, D.; BORN, J. The parallel genetic algorithm as function optimizer. *Parallel Computing*, v. 17, n. 6-7, p. 619–632, 1991.

MUNETOMO, M.; GOLDBERG, D. E. Identifying linkage groups by nonlinearity/non-monotonicity detection. In: *Proc. of the Genetic and Evolutionary Computation Conference*. [S.l.]: Morgan Kaufmann Publishers, Inc, 1999. p. 433–440.

MUSIL, M.; WILMUT, M. J.; CHAPMAN, R. A hybrid simplex genetic algorithm for estimating geoacoustic parameters using matched-field inversion. *IEEE Journal of Oceanic Engineering*, v. 24, n. 3, p. 358–369, 1999.

NAGAR, A.; HERAGU, S. S.; HADDOCK, J. A combined branch-and-bound and genetic algorithm based approach for a flowshop scheduling problem. In: *Annals of Operations Research*. [S.l.: s.n.], 1996. v. 63, n. 3, publisher = Springer Netherlands,.

NAUDTS, B.; KALLEL, L. A comparison of predictive measures of problem difficulty in evolutionary algorithms. *Evolutionary Computation, IEEE Transactions on*, v. 4, n. 1, p. 1–15, Apr 2000. ISSN 1089-778X.

NOVKOVIC, S.; SVERKO, D. The minimal deceptive problem revisited: the role of "genetic waste". *Comput. Oper. Res.*, Elsevier Science Ltd., Oxford, UK, UK, v. 25, n. 11, p. 895–911, 1998. ISSN 0305-0548.

OH, S.-K.; PEDRYCZ, W.; ROH, S.-B. Hybrid fuzzy set-based polynomial neural networks and their development with the aid of genetic optimization and information granulation. *Appl. Soft Comput.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 9, n. 3, p. 1068–1089, 2009. ISSN 1568-4946.

OLIVER, I. M.; SMITH, D. J.; HOLLAND, J. R. C. A study of permutation crossover operators on the traveling salesman problem. In: *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*. Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 1987. p. 224–230.

ORVOSH, D.; DAVIS, L. Shall we repair? genetic algorithmscombinatorial optimizationand feasibility constraints. In: *Proceedings of the 5th International Conference on Genetic Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993. p. 650. ISBN 1-55860-299-2.

PAPPA, G. L. *Seleção de atributos utilizando Algoritmos Genéticos multiobjetivos*. Dissertação (MSc. tese) — PUC-PR, Curitiba, 2002.

PARK, J. et al. An improved genetic algorithm for generation expansion planning. *Power Systems*, v. 15, n. 1, p. 916–922, 2000.

PENA, J. M. et al. Ga-eda: hybrid evolutionary algorithm using genetic and estimation of distribution algorithms. In: *IEA/AIE'2004: Proceedings of the 17th international conference on Innovations in applied artificial intelligence*. [S.l.]: Springer Springer Verlag Inc, 2004. p. 361–371. ISBN 3-540-22007-0.

PETERSEN, C. C. Computational experience with variants of the balas algorithm applied to the slection of r&d projects. *Management Science*, v. 13, p. 736–750, 1967.

PIRES, A. *Infertilidade*. Dez 2009. Site. Disponível em: <http://www.apfertilidade.org/blog/2008/07/26/30-anos-de-fiv/>.

PRó-EMBRYO. *Pró-Embryo : Técnicas de Reprodução Assistida*. Dez 2009. Site. Disponível em: <http://www.proembryo.com.br/proembryo/portugues/medico/tecnicas_rla.aspx>.

PREUX, P.; TALBI, E.-G. Towards hybrid evolutionary algorithms. *International Transactions in Operational Research*, IFORS, Laboratoire d'Informatique du Littoral, France, v. 6, n. 6, p. 557–570, 1999. ISSN 0924-669X.

PUCHINGER, J.; RAIDL, G. R.; PFERSCHY, U. The core concept for the multidimensional knapsack problem. In: *IN EVOLUTIONARY COMPUTATION IN COMBINATORIAL OPTIMIZATION - EVOCOP 2006*. [S.l.]: Springer, 2006. p. 195–208.

QU, B.; XU, B. On epistasis for measure of genetic algorithm hardnesss. In: *Proceedings of the Annual meeting of the North American*. [S.l.: s.n.], 2006. p. 73–76.

RABINI, Y.; RABINOVICH, Y.; SINCLAIR, A. A computational view of population genetics. *Random Struct. Algorithms*, John Wiley & Sons, Inc., New York, NY, USA, v. 12, n. 4, p. 313–334, 1998. ISSN 1042-9832.

RAJAN, C.; MOHAN, M.; MANIVANNAN, K. Improved genetic algorithm solution to unit commitment problem. In: *Transmission and Distribution Conference and Exhibition*. Asia Pacific: IEEE/PES, 2002. p. 255–260.

RASTRIGIN, L. A. *Rastrigin function*. Nov 2009. Disponível em: <http://www.cs.rtu.lv/dssg/en/staff/rastrigin/rastr-function.html>.

RECHENBERG, I. *Evolutionsstrategie: optimierung technischer systeme nach prinzipien der biologischen evolution*. [S.l.]: Frommann-Holzboog, 1973.

REYNOLDS, R. G.; SVERDLIK, W. Problem solving using cultural algorithms. In: *Proceedings of the International Conference on Evolutionary Computation*. [S.l.: s.n.], 1994. p. 645–650.

RICHTER, J. N.; PEAK, D. Fuzzy evolutionary cellular automata. In: *International Conference on Artificial Neural Networks in Engineering*. Saint Louis, USA: [s.n.], 2002. v. 12, p. 185–191.

ROBLES, V. et al. Extending the ga-eda hybrid algorithm to study diversification and intensification in gas and edas. In: *Proceedings of the 6th International Symposium on Intelligent Data Analysis*. Madrid, Spain: [s.n.], 2005. p. 339–350.

ROCHET, S. Epistasis in genetic algorithms revisited. *Inf. Sci.*, Elsevier Science Inc., New York, NY, USA, v. 102, n. 1-4, p. 133–155, 1997. ISSN 0020-0255.

RODRIGUES, E. Genetic programming in the optimization of digital circuits (in portuguese). In: *IV Encontro Nacional de Inteligência Artificial*. Campinas, Brazil: [s.n.], 2003.

ROGERS, D. G/splines: A hybrid of friedman's multivariate adaptive regression splines (mars) algorithm with holland's genetic algorithm. In: *In Proceedings of the Fourth International Conference on Genetic Algorithms*. [S.l.]: Morgan Kaufmann, 1991. p. 384–391.

RONG-LONG, W.; KOZO, O. Solving facility layout problem using an improved genetic algorithm. *Fundamentals of Electronics, Communications and Computer Sciences*, E88-A, n. 2, p. 606–610, 2005.

ROTHLAUF, F. Binary representations of integers and the performance of selectorecombinative genetic algorithms. Springer Berlin / Heidelberg, p. 99–108, 2002.

ROWE, J.; EAST, I. Deception and ga-hardness. In: *Proceedings of the 1NWGA*. Vaasa, Finlândia: [s.n.], 1992. p. 165–172.

ROWE, J. et al. Properties of gray and binary representations. *Evolutionary Computation*, v. 12, n. 1, p. 47–76, 2004.

RUTTKAY, Z.; EIBEN, A.; RAUE, P. Improving the performances of gas on a ga-hard csp. In: *Workshop on Studying and Solving Really Hard Problems*. Cassis, França: [s.n.], 1995. p. 157–171.

Sá Ângela A. R. et al. Exploration vs. exploitation in differential evolution. In: *Proceedings of the AISB 2008 Symposium on Swarm Intelligence Algorithms and Applications*. Aberdeen, Scotland: The Society for the Study of Artificial Intelligence and Simulation of Behaviour, 2008. p. 57–63.

SANCHEZ-VELAZCO, J.; BULLINARIA, J. A. Gendered selection strategies in genetic algorithms for optimization. In: *Proceedings of the UK Workshop on Computational Intelligence*. Bristol, UK: [s.n.], 2003.

SARENI, B.; KRAHENBUHL, L. Fitness sharing and niching methods revisited. In: *IEEE Transactions on Evolutionary Computation*. [S.l.]: IEEE, 1998. v. 2, n. 3, p. 97–106.

SASTRY, K.; GOLDBERG, D.; KENDALL, G. Search methodologies. In: _____. [S.l.]: Springer US, 2005. cap. Genetic Algorithms, p. 97–125.

SASTRY, K.; GOLDBERG, D. E. Let's get ready to rumble redux: crossover versus mutation head to head on exponentially scaled problems. In: *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*. New York, NY, USA: ACM, 2007. p. 1380–1387. ISBN 978-1-59593-697-4.

SCHAFFER, J. D. et al. A study of control parameters affecting online performance of genetic algorithms for function optimization. In: *Proceedings of the third international conference on Genetic algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1989. p. 51–60.

SCHAFFER, J. D.; ESHELMAN, L. J. On crossover as an evolutionarily viable strategy. In: *Proceedings of the Fourth International Conference on Genetic Algorithms - ICGA*. [S.l.: s.n.], 1991. p. 61–68.

SCHWEFEL, H.-P. *Numerical Optimization of Computer Models*. New York, NY, USA: John Wiley & Sons, Inc., 1981.

SCHWEFEL, H.-P. Handbook of evolutionary computation. In: _____. [S.l.]: IOP Publishing and Oxford University Press, 1997. cap. Advantages (and disadvantages) of evolutionary computation over other approaches, p. A1.3.

SENTINELLA, M. R.; CASALINO, L. Enhanced continuos tabu search in a hybrid evolutionary algorithm for the optimization of interplanetary trajectories. In: CENTRE NATIONAL DÉTUDES SPATIALES - CNES. *International Symposium on Space Flight Dynamics*. Toulouse, França, 2009.

SHIMODAIRA, H. An empirical performance comparison of niching methods for genetic algorithms. *IEICE Trans Inf Syst*, E85-D, n. 11, p. 1872–1880, 2002.

SHIZEN; YANG, C. T. Z. Comparison of steady state and elitist selection genetic algorithms. In: IEEE. *Proceedings of the 2004 International Conference on Intelligent Mechatronics and Automation*. Chengdu, China, 2004.

SILVA, F. da et al. Alineaga: a genetic algorithm with local search optimization for multiple sequence alignment. *Applied Intelligence*, 2009.

SINGH, G.; DEB, K. Comparison of multi-modal optimization algorithms based on evolutionary algorithms. In: *8th annual conference on Genetic and evolutionary computation*. Seattle, USA: [s.n.], 2006. p. 1305–1312.

SINHA, A.; GOLDBERG, D. E. *A Survey of Hybrid Genetic and Evolutionary Algorithms*. [S.l.], 2003.

SKOLNICK, D. P. M.; TONG, S. Handbook of genetic algorithms. In: _____. New York: Van Nostrand Reinhold, 1991. cap. Interdigitation: A Hybrid Technique for Engineering Design Optimization Employing Genetic Algorithms, Expert Systems, and Numerical Optimization, p. 312–331.

SMITH, J.; FOGARTY, T. C. Operator and parameter adaptation in genetic algorithms. *Soft Comput.*, v. 1, n. 2, p. 81–87, 1997.

SOARES, G. L. *Algoritmos Genéticos: Estudo, Novas Técnicas e Aplicações*. Tese (Tese de Doutorado) — Escola de Engenharia da Universidade Federal de Minas Gerais, 1997.

SOUZA, M. J. F. *Inteligência Computacional para Otimização*. 2009. Notas de aula. Disponível em: <http://www.iceb.ufop.br/decom/prof/marcone>.

SPEARS, W. Simple subpopulation schemes. In: *Proceedings of Evolucionary Programming Conference*. [S.l.]: World Scientific, 1994. p. 296–307.

SPEARS, W. The handbook of evolutionary computation. In: _____. Philadelphia, PA: IOP Publishing Ltd. and Oxford University Press, 1997. cap. E1.3, p. E1.3:1–E1.3:13.

SPEARS, W. M. Crossover or mutation? In: *Foundations of Genetic Algorithms 2*. [S.l.]: Morgan Kaufmann, 1992. p. 221–237.

SPEARS, W. M.; ANAND, V. A study of crossover operators in genetic programming. In: *ISMIS '91: Proceedings of the 6th International Symposium on Methodologies for Intelligent Systems*. London, UK: Springer-Verlag, 1991. p. 409–418.

SPEARS, W. M.; JONG, K. A. D. An analysis of multi-point crossover. In: *Proceedings of the First Foundations of Genetic Algorithms Conference*. San Francisco, CA, USA: Morgan-Kaufman, 1991. p. 301–315.

SRINIVAS, M.; PATNAIK, L. M. Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Trans. Syst. Man Cybern.*, IEEE Press, v. 24, n. 4, p. 656–667, 1994.

SULTAN, A. B. M.; MAHMUD, R.; S., S. M. Reducing premature convergence problem through numbers structuring in genetic algorithm. *International Journal of Computer Science and Network Security*, v. 7, n. 4, p. 215–217, 2007.

SYBERFELDT, A.; PERSSON, L. Using heuristic search for initiating the genetic population in simulation-based optimization of vehicle routing problems. In: *Proceedings of Industrial Simulation Conference*. [S.l.]: EUROSIS-ETI, 2009.

SYSWERDA, G. Handbook of genetic algorithms. In: _____. [S.l.]: Van Nostrand Reinhold Company, 1991. cap. Scheduling optimization using algorithm genetics, p. 332–349.

SYWERDA, G. Uniform crossover in genetic algorithms. In: *Proceedings of the third international conference on Genetic algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1989. p. 2–9.

TACKETT, W. A.; CARAMI, A. The unique implications of brood selection for genetic programming. In: *International Conference on Evolutionary Computation*. [S.l.: s.n.], 1994. p. 160–165.

TACKETT, W. A.; CARMI, A. The unique implications of brood selection for genetic programming. In: *IEEE World Congress on Computational Intelligence*. Orlando, FL, USA: [s.n.], 1994. p. 160–165.

TALBI, E.-G. A taxonomy of hybrid metaheuristics. *Journal of Heuristics*, Kluwer Academic Publishers, Hingham, MA, USA, v. 8, n. 5, p. 541–564, 2002. ISSN 1381-1231.

TAN, K. et al. Balancing exploration and exploitation with adaptive variation for evolutionary multi-objective optimization. *European Journal of Operational Research*, v. 197, n. 2, p. 701–713, September 2009.

TAN, K. et al. Evolving better population distribution and exploration in evolutionary multi-objective optimization. *European Journal of Operational Research*, v. 171, n. 2, p. 463–495, June 2006.

TAN, K. C.; LEE, T. H.; KHOR, E. F. Evolutionary algorithms with dynamic population size and local exploration for multiobjective optimization. *IEEE Trans. Evolutionary Computation*, v. 5, n. 6, p. 565–588, 2001.

THIERENS, D.; GOLDBERG, D. E. Mixing in genetic algorithms. In: *Proceedings of the 5th International Conference on Genetic Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993. p. 38–47.

TOGAN, V.; DALOGLU, A. T. An improved genetic algorithm with initial population strategy and self-adaptive member grouping. *Comput. Struct.*, Pergamon Press, Inc., Elmsford, NY, USA, v. 86, n. 11-12, p. 1204–1218, 2008.

TORN, A.; ZILINSKAS, A. Global optimization. *Lecture Notes in Computer Science*, v. 350, 1989.

TRACER. *The Ackley Problem*. Nov 2009. Disponível em: <http://tracer.lcc.uma.es/problems/ackley/ackley.html>.

TRACER. *The Griewangk Problem*. Nov 2009. Disponível em: <http://tracer.lcc.uma.es/problems/Griewangk/griewangk.html>.

TRAUTMANN, H. et al. A convergence criterion for multiobjective evolutionary algorithms based on systematic statistical testing. In: *Proceedings of the 10th international conference on Parallel Problem Solving from Nature*. Berlin, Heidelberg: Springer-Verlag, 2008. p. 825–836. ISBN 978-3-540-87699-1.

VASCONCELOS, J. A.; SALDANHA, R. R. Genetic algorithm coupled with a deterministic method for optimization in electromagnetics. *IEEE TRANSACTIONS ON MAGNETICS*, IEEE Press, v. 33, n. 2, p. 1860–1863, 1997.

VENTURINI, G. Ga consistently deceptive functions are not challenging problems. In: *First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*. [S.l.: s.n.], 1995. p. 357–364.

VOSE, M. D.; LIEPINS, G. E. Schema disruption. In: *Proceedings of the Fourth International Conference on Genetic Algorithms - ICGA*. [S.l.: s.n.], 1991. p. 237–243.

WANG, C.-X. et al. A novel genetic algorithm based on gene therapy theory. *Transactions of the Institute of Measurement and Control*, v. 28, n. 3, p. 253–262, 2006.

WANG, C.-X. et al. A novel genetic algorithm based on cure mechanism of traditional chinese medicine. In: *Proceedings of the First International Conference on Advances in Natural Computation*. Changsha, China: [s.n.], 2005. p. 86–92.

WANG, C.-X. et al. A novel genetic algorithm based on dynastic changes mechanism of nation. In: *WGEC '08: Proceedings of the 2008 Second International Conference on Genetic and Evolutionary Computing*. Washington, DC, USA: IEEE Computer Society, 2008. p. 109–112. ISBN 978-0-7695-3334-6.

WHITLEY, D. Deception, dominance and implicit parallelism in genetic search. *Ann. Math. Artif. Intell.*, v. 5, n. 1, p. 49–78, 1992.

WHITLEY, L. D. Fundamental principles of deception in genetic search. In: *Foundations of Genetic Algorithms*. [S.l.]: Morgan Kaufmann, 1991. p. 221–241.

WIKIPEDIA.ORG. *Genetic diversity*. Nov 2009. Disponível em: <http://en.wikipedia.org/wiki/Genetic_diversity>.

WOLPERT, D. H.; MACREADY, W. G. No free lunch theorems for optimization. *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, v. 1, n. 1, p. 67–82, 1997.

WU, A. S. et al. An incremental genetic algorithm approach to multiprocessor scheduling. *Parallel and Distributed Systems*, v. 15, n. 9, p. 824–834, 2004.

YAGIURA, M.; IBARAKI, T. The use of dynamic programming in genetic algorithms for permutation problems. *European Journal of Operational Research*, v. 92, n. 2, p. 387–401, July 1996.

YANG, F.; ZHANG, C.; BAI, G. A novel genetic algorithm based on tabu search for hmm optimization. In: *ICNC '08: Proceedings of the 2008 Fourth International Conference on Natural Computation*. Washington, DC, USA: IEEE Computer Society, 2008. p. 57–61. ISBN 978-0-7695-3304-9.

YANG, R.; DOUGLAS, I. Simple genetic algorithm with local tuning: Eficient global optimizing technique. *J. Optim. Theory Appl.*, v. 98, n. 2, p. 449–465, 1998.

YEN, J. et al. A hybrid approach to modeling metabolic systems using a genetic algorithm and simplex method. *IEEE Trans. on Syst., Man, and Cybern.*, v. 28, n. 2, p. 173–191, 1998.

YUAN, L.; LI, M.; LI, J. Research on diversity measure of niche genetic algorithm. In: *Proceedings of the 2008 Second International Conference on Genetic and Evolutionary Computing*. Washington, DC, USA: IEEE Computer Society, 2008. p. 47–50. ISBN 978-0-7695-3334-6.

YUAN, L.; LI, M.; LI, J. Research on diversity measure of niche genetic algorithm. In: *Second International Conference on Genetic and Evolutionary Computing*. [S.l.: s.n.], 2008. p. 47–50.

YUN, Y. Hybrid genetic algorithm with adaptive local search scheme. *Comput. Ind. Eng.*, Pergamon Press, Inc., Tarrytown, NY, USA, v. 51, n. 1, p. 128–141, 2006. ISSN 0360-8352.

ZDÁNSKÝ, M.; POZIVIL, J. Combination genetic/tabu search algorithm for hybrid flowshops optimization. In: SLOVAK UNIVERSITY OF TECHNOLOGY. *Proceedings of ALGORITMY, Conference on Scientific Computing*. Slovakia, 2002. p. 230–236.

ZHANG, G.; SHI, Y.; GAO, L. A genetic algorithm and tabu search for solving flexible job shop schedules. In: *ISCID '08: Proceedings of the 2008 International Symposium on Computational Intelligence and Design*. Washington, DC, USA: IEEE Computer Society, 2008. p. 369–372. ISBN 978-0-7695-3311-7.

ZHANG, J.; CHUNG, H. S. H.; ZHONG, J. Adaptive crossover and mutation in genetic algorithms based on clustering technique. In: *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*. New York, NY, USA: ACM, 2005. p. 1577–1578.

ZHENG, J. et al. Hybrid genetic algorithms for parameter identification of a hysteresis model of magnetostrictive actuators. *Neurocomputing*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 70, n. 4-6, p. 749–761, 2007. ISSN 0925-2312.

ZHU, K. Q.; LIU, Z. Population diversity in permutation-based genetic algorithm. In: *Proceedings of the Machine Learning: ECML 2004, 15th European Conference on Machine Learning*. Pisa, Italy: Springer, 2004. (Lecture Notes in Computer Science, v. 3201), p. 537–547. ISBN 3-540-23105-6.

ZUBEN, F. J. V. *Computação Evolutiva: Uma Abordagem Pragmática*. [S.l.], 2000.