UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

ALGORITMOS CULTURAIS COM ABORDAGEM MEMÉTICA E MULTIPOPULACIONAL APLICADOS A PROBLEMAS DE OTIMIZAÇÃO

DEAM JAMES AZEVEDO DA SILVA

TD: 07/2012

UFPA / ITEC / PPGEE
CAMPUS UNIVERSITÁRIO DO GUAMÁ
BELÉM – PARÁ – BRASIL
2012

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

DEAM JAMES AZEVEDO DA SILVA

ALGORITMOS CULTURAIS COM ABORDAGEM MEMÉTICA E MULTIPOPULACIONAL APLICADOS A PROBLEMAS DE OTIMIZAÇÃO

Tese submetida à Banca Examinadora do Programa de Pós-graduação em Engenharia Elétrica da UFPA como parte dos requisitos necessários para a obtenção do Título de Doutor em Engenharia Elétrica.

Área de Concentração: Computação Aplicada.

Orientador: Prof. Dr. Roberto Célio Limão de Oliveira.

UFPA / ITEC / PPGEE
CAMPUS UNIVERSITÁRIO DO GUAMÁ
BELÉM – PARÁ – BRASIL
2012

S586a Silva, Deam James Azevedo da

Algoritmos culturais com abordagem memética e multipopulacional aplicados a problemas de otimização / Deam James Azevedo da silva; orientador, Roberto Célio Limão de Oliveira . - 2012.

Tese (doutorado) – Universidade Federal do Pará, Instituto de Tecnologia, Programa de Pós-Graduação em Engenharia Elétrica, Belém, 2012.

1. Otimização combinatória. 2. Algoritmos genéticos. 3. Computação evolucionaria. I. Orientador. II. Título.

CDD 22. ed. 519.64

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

ALGORITMOS CULTURAIS COM ABORDAGEM MEMÉTICA E MULTIPOPULACIONAL APLICADOS A PROBLEMAS DE OTIMIZAÇÃO

AUTOR: DEAM JAMES AZEVEDO DA SILVA

TESE DE DOUTORADO SUBMETIDA À AVALIAÇÃO DA BANCA EXAMINADORA E APROVADA PELO COLEGIADO DO PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA DA UNIVERSIDADE FEDERAL DO PARÁ, JULGADA ADEQUADA PARA OBTENÇÃO DO TÍTULO DE DOUTOR EM ENGENHARIA ELÉTRICA NA ÁREA DE COMPUTAÇÃO APLICADA.

BANCA EXAMINADORA:

_____

Prof. Dr. Roberto Célio Limão de Oliveira
(Orientador –PPGEE/UFPA)

_____

Prof. Dr. Ronaldo de Freitas Zampolo
(Membro- FCT/UFPA)

_____

Prof. Dr. Dionne Cavalcante Monteiro
(Membro – PPGCC/UFPA)

_____

Prof. Dr. Adamo Lima de Santana
(Membro- PPGEE/UFPA)

_____

Prof. Dr João Viana Fonseca Neto
(Membro Externo- UFMA)

_____

Prof. Dr. Adrião Duarte Dória Neto
(Membro Externo- UFRN)

AGRADECIMENTOS

Em primeiro lugar, agradeço a Deus pela sabedoria, saúde e perseverança;

Ao meu orientador professor Dr. Roberto Célio Limão de Oliveira pela dedicação, paciência e boa disposição na orientação desse trabalho;

À minha família pelo amor, paciência e compreensão no decorrer deste trabalho, especialmente para Jacy Mary Silva;

A todos os professores do PPGEE pelo conhecimento e inspiração repassados ao longo dos anos;

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) e a Fundação Amazônia Paraense (FAPESPA), pela bolsa de doutorado.

SUMÁRIO

LISTA DE FIGURAS ..... **X**

LISTA DE TABELAS ..... **XII**

LISTA DE ABREVIATURAS ..... **XIII**

RESUMO ..... **XIV**

ABSTRACT ..... **XV**

1 INTRODUÇÃO **1**

1.1 MÉTODOS DE OTIMIZAÇÃO MODERNOS 2

1.2 COMPUTAÇÃO EVOLUTIVA 3

1.2.1 Algoritmos Genéticos *5*

1.2.2 Algoritmos Culturais (ACs) *5*

1.3 PROBLEMAS DE OTIMIZAÇÃO ESCOLHIDOS 6

1.3.1 Problema de Funções Multimodais (PFM) *6*

1.3.2 Problema de Funções Restritas (PFR) *6*

1.3.3 Problema da Mochila Multidimensional (PMM/MKP) *7*

1.3.4 O Problema do Caixeiro Viajante (PCV/TSP) *8*

1.4 DIFICULDADES ENCONTRADAS NO USO DE METAHEURÍSTICAS 8

1.4.1 Dificuldades Encontradas na Otimização de Funções Multimodais (PFM) *9*

1.4.1.1 Trabalhos Relacionados na Otimização de Funções Multimodais (PFM) 10

1.4.2 Dificuldades Encontradas na Otimização de Funções Restritas (PFR) *10*

1.4.2.1 Trabalhos Relacionados na Otimização de Funções Restritas (PFR) 11

1.4.3 Dificuldades Encontradas na Otimização de Problemas da Mochila Multidimensional (PMM/MKP) *12*

1.4.3.1 Trabalhos Relacionados na Otimização de Problemas da Mochila Multidimensional (MKP) 13

1.4.4 Dificuldades Encontradas na Otimização do Problema do Caixeiro Viajante (PCV/TSP) *13*

1.4.4.1 Trabalhos Relacionados na Otimização do Problema do Caixeiro Viajante (PCV/TSP) 14

1.5 CONTRIBUIÇÕES 14

1.5.1 Abordagem Memética *14*

1.5.2 Abordagem Multipopulacional *15*

1.5.3 Outras contribuições (Framework Evolutivo Java) *16*

1.6 ESTRUTURA DA TESE 16

2 ALGORITMO GENÉTICO (AG) 17

2.1 INTRODUÇÃO 17

2.2 CARACTERÍSTICAS DOS AGS 19

2.3 CLASSIFICAÇÕES DE ALGORITMOS GENÉTICOS 20

2.3.1 Algoritmo Genético Simples (AGS) 21

2.3.1.1 Indivíduos 22

2.3.1.2 Genes 22

2.3.1.3 População 23

2.3.1.4 Codificação 23

2.3.1.5 Seleção 23

2.3.1.6 Mutação, Recombinação e Substituição da População. 24

2.3.1.6.1 Operador de Recombinação 25

2.3.1.6.2 Operador de Mutação 26

2.3.1.6.3 Substituição da População 26

2.3.2 Algoritmos Genéticos Paralelos e Distribuídos (AGP e AGD) 26

2.3.2.1 AGP com Paralelização Global (*Master-Slav*e ou Mestre-Escravo) 29

2.3.2.2 AGP com Granulação Grossa (AGs Distribuidos) 31

2.3.2.3 AGP com Granulação Fina 33

2.4 CONSIDERAÇÕES SOBRE OS AGS 34

3 ALGORITMOS CULTURAIS 35

3.1 INTRODUÇÃO 35

3.2 CARACTERÍSTICAS CULTURAIS 35

3.2.1 Meme 36

3.2.2 Evolução Cultural 36

3.2.3 Propagação de Memes 37

3.2.4 Evolução dos Memes 37

3.3 PRINCÍPIOS DO ALGORITMO CULTURAL 37

3.4 INSPIRAÇÃO NATURAL 38

3.5 FUNCIONAMENTO BÁSICO DE UM ALGORITMO CULTURAL 39

3.6 PRINCIPAIS CARACTERÍSTICAS 41

3.7 MICROEVOLUÇÃO E MACROEVOLUÇÃO 42

3.7.1 Espaço Populacional 42

3.7.2 Espaço de Crenças 43

3.7.2.1 Conhecimento Situacional 43

3.7.2.2 Conhecimento Normativo 44

3.7.2.3 Conhecimento do Domínio 45

3.7.2.4 Conhecimento Topográfico 45

3.7.2.5 Conhecimento Histórico 46

3.7.3 Protocolos de Comunicação *47*
3.7.3.1 Função de Aceitação *47*
3.7.3.2 Função de Influência *47*
3.8 Trabalhos Relacionados 48
4 HIBRIDIZAÇÃO DE ALGORITMOS EVOLUTIVOS COM BUSCA LOCAL: "ALGORITMOS MEMÉTICOS". 50
4.1 Introdução 50
4.2 Benefícios da Busca Local 52
4.3 Combinação de Metaheurísticas com Busca Local na Melhoria de Soluções 54
4.3.1 Operadores de Busca Especializados *54*
4.3.2 Busca Local Real *55*
4.4 Implementação de Algoritmos Meméticos 55
4.4.1 Frequência da Aplicação da Busca Local *56*
4.4.2 Escolha dos Indivíduos para Aplicar a Busca Local *56*
4.4.3 Tempo de Execução da Busca Local *57*
4.4.4 Quantidade de Eficiência que a Busca Local Precisa Ter *58*
4.5 Busca Local em um Algoritmo Memético 59
4.5.1 Posicionamento da Busca Local *61*
4.5.2 Busca Local Dentro do Ciclo Evolutivo *62*
4.6 Estratégias de Busca Local 63
4.6.1 Hill Climbing *63*
4.6.2 Simulated Annealing (SA) *64*
4.6.3 Tabu Search *66*
4.7 Algoritmos Meméticos em Problemas Contínuos de Otimização 66
4.8 Estrutura de Vizinhança Utilizada na Busca Local 66
4.8.1 Combinação de Estruturas de Vizinhanças (Movimentos Distintos) *67*
4.9 Problemas Discretos e Problemas Contínuos 68
5 ALGORITMOS CULTURAIS: ABORDAGEM MEMÉTICA COM BUSCA LOCAL E ABORDAGEM MULTIPOPULACIONAL 70
5.1 Introdução 70
5.2 Implementação do Algoritmo Cultural 70
5.3 Escolha da Busca Local 70
5.4 *Simulated Annealing* em Problemas Discretos e Problemas Contínuos 71
5.5 Estrutura de Vizinhanças Proposta em Problemas Contínuos: Problema das Funções Multimodais (PFM) 72
5.5.1 Perturbações Pré-Definidas (N_1) *73*
5.5.2 Perturbações com Pontos-Chave do Espaço de Busca (N_2) *75*
5.5.3 Perturbações de Baixa Intensidade (N_3) *77*

5.5.4 Perturbações de Alta Intensidade (N_4) ... *78*
5.6 Problemas Restritos ... 79
5.6.1 Função de Penalidade em Problemas Restritos Proposta ... *79*
5.7 Considerações sobre Estrutura de Vizinhança para Funções Irrestritas e Restritas ... 80
5.8 Estrutura de Vizinhança para Problemas Discretos: Problema do Caixeiro Viajante (PCV) ... 80
5.8.1 Busca local 2-OPT e 3-OPT ... *81*
5.9 ACs com Abordagem Memética ... 84
5.10 ACs com Abordagem Multipopulacional ... 86
5.10.1 Algoritmo Cultural Baseado Modelo de Ilhas (AC-MI) ... *87*

6 RESULTADOS ... 89

6.1 Resultados para Otimização Funções Multimodais ... 89
6.1.1 Gráficos para as Funções Multimodais ... *92*
6.2 Resultados para Otimização de Funções Restritas (Problemas de Engenharia) ... 94
6.3 Resultados para o Problema da Mochila Multidimensional (MKP) ... 97
6.3.1 DGA e DGA-SRM ... *98*
6.3.2 AC-MI_1 ... *100*
6.3.3 AC-MI_2 ... *102*
6.3.4 Algoritmo Cultural Padrão (AC) ... *103*
6.4 Resultados para o Problema do Caixeiro Viajante (PCV) ... 104

7 CONCLUSÃO ... 106

8 REFERÊNCIAS BIBLIOGRÁFICAS ... 108

ANEXOS ... 124

A1- Otimização de Funções Multimodais ... 124
A2-Problemas Restritos ... 124
A3-Problema da Mochila Multidimensional (*Multiple Knapsack Problem-MKP*) ... 124
A4-O Problema do Caixeiro Viajante (PCV) ... 125
A6 - Benchmark Utilizado nos Problemas Restritos ... 127
A6.1 Problema-1 ... *127*
A6.2 Problema-2 ... *127*
A7 - Benchmark Utilizado nos Problemas da Mochila ... 128
A8- Visão geral do Framework Implementado (*Java Evolutionary Framework* - JEF) ... 129
A8.1 Estrutura principal do Framework ... *130*
A8.2 Exemplo de uma Estrutura de Representação para o Cromossomo ... *131*
A8.3 Estrutura de Representação dos Algoritmos ... *132*
A9 Arquivo de Parâmetros do framework em XML ... 133

A9.1 Exemplo: *133*
A9.2 Parâmetros de Perturbação independente..... *134*

LISTA DE FIGURAS

FIGURA 1: REPRESENTAÇÃO DO FENÓTIPO E GENÓTIPO (SIVANANDAM, 2007). 22
FIGURA 2: REPRESENTAÇÃO DE UM CROMOSSOMO. 23
FIGURA 3: REPRESENTAÇÃO DE GENES 23
FIGURA 4: UM ESQUEMA DE UM AG PARALELO MESTRE-ESCRAVO (CANTU-PAZ, 2000). 30
FIGURA 5: UM ESQUEMA DE AG PARALELO COM MÚLTIPLAS POPULAÇÃO (MÚLTIPLOS *DEMES*) (CANTU-PAZ, 2000). 31
FIGURA 6: UM ESQUEMA DE UM AG PARALELO COM GRANULAÇÃO FINA (CANTU-PAZ, 2000). 33
FIGURA 7: FUNCIONAMENTO BÁSICO DE UM ALGORITMO CULTURAL (REYNOLDS ET AL., 2010). 39
FIGURA 8: REPRESENTAÇÃO DO CONHECIMENTO SITUACIONAL (IACOBAN;REYNOLDS;BREWSTER, 2003B). 44
FIGURA 9: REPRESENTAÇÃO DE CONHECIMENTO NORMATIVO (IACOBAN; REYNOLDS; BREWSTER, 2003B). 45
FIGURA 10: REPRESENTAÇÃO DE CONHECIMENTO TOPOGRÁFICO (BECERRA; COELLO, 2005). 46
FIGURA 11: REPRESENTAÇÃO DE CONHECIMENTO HISTÓRICO (BECERRA; COELLO, 2005). 47
FIGURA 12: SITUAÇÃO QUE UM AG ESTÁ CONVERGENDO PARA UM ÓTIMO GLOBAL (YOUNGSU ,2006). 53
FIGURA 13: SITUAÇÃO QUE UM AG NÃO ESTÁ CONVERGENDO PARA UM ÓTIMO GLOBAL (YOUNGSU ,2006). 53
FIGURA 14: POSICIONAMENTO DA BUSCA LOCAL 62
FIGURA 15: PROCESSO DE BUSCA LOCAL PRESO NO MÍNIMO LOCAL (ADRA, 2003) 63
FIGURA 16: SA ESCAPANDO DE UM ÓTIMO LOCAL (ADRA, 2003). 65
FIGURA 17: PERTURBAÇÃO PRÉ-DEFINIDA DE UM PONTO P(X) 74
FIGURA 18: SA ESCAPANDO DE UM ÓTIMO LOCAL COM VIZINHANÇA N_1 75
FIGURA 19: PONTOS-CHAVE DO ESPAÇO DE BUSCA N_2 76
FIGURA 20: SA ESCAPANDO DE UM ÓTIMO LOCAL COM N_2 77
FIGURA 21: SA ESCAPANDO DE UM ÓTIMO LOCAL COM N_3 77
FIGURA 22: SA ESCAPANDO DE UM ÓTIMO LOCAL COM N_4 78
FIGURA 23: (A) OPERAÇÃO 2-OPT E (B) OPERAÇÃO 3-OPT. 82
FIGURA 24: REPRESENTAÇÃO DA MUTAÇÃO UTILIZANDO CONHECIMENTO SITUACIONAL 84
FIGURA 25:REPRESENTAÇÃO DA MUTAÇÃO UTILIZANDO CONHECIMENTO NORMATIVO 84
FIGURA 26: FRAMEWORK DO AC COM BUSCA LOCAL PROPOSTO. 85
FIGURA 27 TOPOLOGIA DE COMUNICAÇÃO +1+2. 87
FIGURA 28: :FAMEWORK DO AC MULTIPOPULACIONAL (SILVA; TEIXEIRA; OLIVEIRA, 2012) 88
FIGURA 29: DESEMPENHO DO AG XAC PARA A FUNÇÃO F2. 92
FIGURA 30:DESEMPENHO DO AC XAC-BL PARA A FUNÇÃO F2. 93

FIGURA 31: DESEMPENHO DO AG X AC PARA A FUNÇÃO F5 (N=20). 93
FIGURA 32: DESEMPENHO DO AC X AC-BL PARA A FUNÇÃO F5 (N=20). 94
FIGURA 33: DESEMPENHO DO AC X AC-BL PARA O PROBLEMA-1. 97
FIGURA 34: DESEMPENHO DO AC X AC-BL PARA O PROBLEMA-2. 97

LISTA DE TABELAS

TABELA 1: PARÂMETROS BÁSICOS PARA AS FUNÇÕES F1 A F9. 90

TABELA 2: PARÂMETROS DA BUSCA LOCAL SA COM VIZINHANÇA *GERARVIZINHOPFM (..)* COM K=4 (VALOR FIXO). 90

TABELA 3: COMPARAÇÃO DOS RESULTADOS PARA AS FUNÇÕES F1 A F9 91

TABELA 4:INTERVALO DE PERTURBAÇÃO PARA O PROBLEMA-1. 94

TABELA 5: INTERVALO DE PERTURBAÇÃO PARA O PROBLEMA-2. 95

TABELA 6: COMPARAÇÃO DO AC-BL COM O ALGORITMO *SOCIAL FABRIC* COM 20 EXECUÇÕES INDEPENDENTES PARA O PROBLEMA-1. 95

TABELA 7: COMPARAÇÃO DO AC-BL COM OUTROS ALGORITMOS COM 30 EXECUÇÕES INDEPENDENTES PARA O PROBLEMA-1. 95

TABELA 8: COMPARAÇÃO DO AC-BL COM OUTORS ALGORITMOS (MELHOR RESULTADO ENCONTRADO PARA O PROBLEMA-1)..... 96

TABELA 9: COMPARAÇÃO DO AC-BL COM OUTROS ALGORITMOS COM 30 EXECUÇÕES INDEPENDENTES PARA O PROBLEMA-2. 96

TABELA 10: COMPARAÇÃO DO AC-BL COM OUTORS ALGORITMOS (MELHOR RESULTADO ENCONTRADO PARA O PROBLEMA-2). 96

TABELA 11: OS MELHORES RESULTADOS PARA WEING7 (105,2) PELO DGA E DGA-SRM(λ_{total}=800; T=8X10^5). 99

TABELA 12: OS MELHORES RESULTADOS PARA OUTROS PROBLEMAS UTILIZANDO DGA AND DGA-SRM (Λ_{TOTAL}= 800, T=4X10^5).. 100

TABELA 13: OS MELHORES RESULTADOS PARA WEING7 (105, 2) BY AC-MI_1 (Λ_{TOTAL}=800 AND T=8X10^5)..... 101

TABELA 14: OS MELHORES RESULTADOS PARA OUTROS PROBLEMAS UTILIZANDO AC-MI_1 (Λ_{TOTAL} = 800, T=4X10^5)..... 102

TABELA 15: OS MELHORES RESULTADOS PARA O PROBLEMA WEING7 (105,2) PELO AC-MI_2 (Λ_{TOTAL} =800, T=8X105). 102

TABELA 16: OS MELHORES RESULTADOS PARA OUTROS PROBLEMASC OM O AC-MI_2 (Λ_{TOTAL} = 800, T=4X105). 103

TABELA 17: OS MELHORES RESULTADOS PARA TODOS OS *BENCHMARKS* DA MOCHILA PELO AC (T=4X10^5). 103

TABELA 18:RESULTADOS PARA INSTÂNCIAS COM CIDADES MENORES QUE 100 (AC-BL PARA O PCV). 104

TABELA 19 : RESULTADOS PARA INSTÂNCIAS COM CIDADES MENORES QUE 100 (AC-BL PARA O PCV) 104

TABELA 20:ALGORITMO DE BUSCA PARALELA TABU 105

TABELA 21 COMPARAÇÃO COM OUTROS ALGORITMOS. 105

LISTA DE ABREVIATURAS

AC – Algoritmo Cultural

AC-MI/CA-IM – Algoritmo Cultural Modelo de Ilhas/*Cultural Algorithm Island Model*

AC-BL – Algoritmo Cultural com Busca Local

AE – Algoritmo Evolutivo

AG – Algoritmo Genético

AG-BL – Algoritmo Genético padrão com Busca Local

AGS – Algoritmo Genético Simples (AG padrão com uma população)

AGP – Algoritmo Genético Paralelo (AG padrão com múltiplas populações)

AGD – Algoritmo Genético Distribuído

AM – Algoritmo Memético

ASPARAGOS – *Asynchronous Parallel Genetic Optimization Strategy*

CA – Cultural Algorithm

CA-LS – *Cultural Algorithm with Local Search*

CE – Computação Evolutiva

FEJ – Framework Evolutivo Java (FEJ)

GA-EAX – *Genetic Algorithm Edge Assembly Crossover*

GA-LK – *Genetic Algorithm with LK Local Search*

GLS – *Genetic Local Search*

GRASP – *Greedy Randomized Adaptive Search Procedure*

JCLEC – *Java Class Library for Evolutionary Computation*

LK – *Lin-Kernighan*

LKH– *Lin Kernighan-Helsgaun*

NGA – *Natural Crossover Genetic Algorithm*

PFM – Problema de Funções Multimodais

PFR – Problema de Funções Restritas

PMM /MKP– Problema da Mochila Multidimensional/*Multiknapsack Problem*

PCV/TSP – Problema do Caixeiro Viajante/*Travel Salesman Problem*

SA – *Simulated Annealing*

RESUMO

Em muitos problemas de otimização há dificuldades em alcançar um resultado ótimo ou mesmo um resultado próximo ao valor ótimo em um tempo viável, principalmente quando se trabalha em grande escala. Por isso muitos desses problemas são abordados por heurísticas ou metaheurísticas que executam buscas por melhores soluções dentro do espaço de busca definido. Dentro da computação natural estão os Algoritmos Culturais e os Algoritmos Genéticos, que são considerados metaheurísticas evolutivas que se complementam devido ao mecanismo dual de herança cultura/genética. A proposta do presente trabalho é estudar e utilizar tais mecanismos acrescentando tanto heurísticas de busca local como multipopulações aplicados em problemas de otimização combinatória (caixeiro viajante e mochila), funções multimodais e em problemas restritos. Serão executados alguns experimentos para efetuar uma avaliação em relação ao desempenho desses mecanismos híbridos e multipopulacionais com outros mecanismos dispostos na literatura de acordo com cada problema de otimização aqui abordado.

Palavras-chave: Algoritmos Genéticos, Algoritmos Culturais, Algoritmos Meméticos, Multipopulação, Problemas Multimodais, Problemas com Restrições, Problema do Caixeiro Viajante, Problema da Mochila Multidimensional.

ABSTRACT

In many optimization problems is hard to reach a good result or a result close to the optimum value in a feasible time, especially when working on large scale. So, many of these problems are addressed by heuristics or metaheuristics running search for better solutions within the defined search space. Within the natural computing algorithms there are the cultural and genetic algorithms. These are evolutionary metaheuristics complement each other due to the dual mechanism of cultural heritage/genetic. The purpose of this paper is to study and use such mechanisms adding local search heuristics and multipopulation applied to combinatorial optimization problems (knapsack and travel salesman problems), constrained problems and multimodal functions. Some experiments have been conducted to assess the performance of the proposed combination of meta-heuristisc and heuristics mechanisms against approaches found in literature as applied to problem addressed here.

Keywords: Genetic Algorithms, Cultural Algorithms, Memetic Algorithms, Multipopulation, Multimodal Problems, Problems with Constraints, Traveling Salesman Problem, Multidimensional Knapsack Problem**.**

1 INTRODUÇÃO

Otimização é uma ciência que está sempre em demanda, uma vez que se encontra direta ou indiretamente relacionada com capital e é empregada em todos os campos de aplicações tais como: engenharia civil, mecânica, automobilística, aérea, econômica, eletrônica, química, etc (LINDEN, 2006). A otimização está em toda parte, envolvendo desde projetos de engenharia até aplicações para os mercados financeiros. Uma organização sempre deseja maximizar seus lucros, minimizar custos e maximizar o desempenho. Mesmo no cotidiano a otimização está presente quando, por exemplo, uma pessoa pretende maximizar sua satisfação com um menor custo no planejamento de férias. Portanto, a busca por soluções ideais é um processo realizado constantemente para cada problema encontrado, embora ocorra incapacidade de se encontrar essas possíveis soluções. A otimização pode ser descrita como o ato de obter o melhor resultado em dadas circunstâncias. Otimização é o ato de atingir o melhor resultado possível em determinadas circunstâncias. A palavra “otimização” tem origem latina, e significa “ótimo final”. Similarmente, “optimus” significa o melhor. Portanto, otimização refere-se à tentativa de trazer o melhor resultado de um problema em questão (ANDRÉASSON, EVGRAFOV, PATRIKSSON, 2005).

No projeto, construção e manutenção de qualquer sistema de engenharia, os engenheiros têm de tomar muitas decisões tecnológicas e gerenciais em diversas etapas. O objetivo final de todas estas decisões ou é minimizar o esforço requerido ou maximizar o benefício desejado. Levando-se em consideração que o esforço necessário ou o benefício desejado em qualquer situação prática pode ser expresso através de uma função de variáveis de decisão, a otimização pode ser definida como o processo de encontrar as condições que darão o valor máximo ou mínimo de uma função (RAO, 2009).

Em termos mais gerais, a teoria de otimização é um conjunto de resultados matemáticos e métodos numéricos para encontrar e identificar o melhor candidato de uma coleção de alternativas, sem ter que explicitamente enumerar e avaliar todas as alternativas possíveis. O processo de otimização está na raiz da engenharia, onde a função clássica do engenheiro é projetar novos sistemas que possam ser cada vez melhores, mais eficientes e menos dispendiosos, bem como elaborar planos e procedimentos para melhorar o funcionamento dos sistemas existentes. O poder dos métodos de otimização em determinar o melhor caso, sem realmente testar todos os possíveis casos, vem por meio do uso de um nível modesto de matemática e da realização de cálculos numéricos iterativos usando

procedimentos lógicos claramente definidos ou algoritmos implementados em máquinas de computação (RAVINDRAN, RAGSDELL, REKLAITIS, 2007).

O estudo de problemas de otimização é também tão antigo quanto a própria ciência. Sabe-se que os matemáticos gregos antigos resolveram muitos problemas de otimização. Por exemplo, cerca de 300 a.C., Euclides provou que um quadrado encerra a maior área entre todos os retângulos possíveis com o mesmo comprimento total de quatro lados. Mais tarde, por volta de 100 a.C., Heron sugeriu que a distância entre dois pontos ao longo do caminho, refletida por um espelho é o mais curto. Ele demonstrou que a igualdade dos ângulos de incidência e reflexão em um espelho segue o princípio de sua fonte ao olho do observador pelo caminho mais curto (problema conhecido como Catoptrica Heron) (YANG, 2010).

Foram as contribuições de Newton e Leibniz que permitiram o desenvolvimento de métodos de cálculo diferencial de otimização. Os fundamentos de cálculo de variações, que trata de minimização funcional, foram estabelecidos por Bernoulli, Euler, Lagrange e Weirstrass (ASTOLFI, 2004). O método de otimização para problemas restritos, que envolve a adição de multiplicadores desconhecidos, tornou-se conhecido pelo nome de seu inventor, “Lagrange”. Cauchy fez a primeira aplicação do método de descida mais íngreme para resolver problemas de minimização irrestrita. Apesar destas primeiras contribuições, pouco progresso foi feito até meados do século XX, quando computadores digitais de alta velocidade tornaram possível a implementação dos procedimentos de otimização e estimulou novas pesquisas sobre novos métodos. Surgiram avanços espetaculares, produzindo uma literatura enorme sobre técnicas de otimização. Este avanço também resultou no surgimento de diversas áreas bem definidas e novas na teoria de otimização.

1.1 Métodos de Otimização Modernos

Os métodos de otimização modernos, também, por vezes, chamados métodos de otimização não tradicionais, surgiram como métodos populares para resolverem problemas de otimização complexos de engenharia nos últimos anos (RAO, 2009). Seu surgimento tem relação com a estratégia de solução baseada em heurísticas. A heurística é uma estratégia de solução por tentativa e erro para produzir soluções aceitáveis para um problema complexo em um tempo razoavelmente prático. A complexidade do problema de interesse torna impossível a busca a cada solução possível ou combinação. O objetivo é encontrar boas soluções viáveis em uma escala de tempo aceitável mesmo que não exista garantia de que as melhores soluções possam ser encontradas (RAO, 2009). Alan Turing foi provavelmente o primeiro a usar

algoritmos heurísticos durante o período da Segunda Guerra Mundial, quando estava tentando decifrar a criptografia das mensagens alemãs. Foi nesse período que Turing, juntamente com o matemático britânico Gordon Welchman, projetou uma máquina eletromecânica "cryptanalytic" denominada *Bombe*, que auxiliava na quebra de códigos criptográficos (YANG, 2010).

Um dos avanços na computação inspirada na natureza foi o aparecimento da computação evolutiva e o passo mais importante foi o desenvolvimento de algoritmos evolutivos na década de 1960 e 1970, durante a qual John Holland e seus colaboradores da Universidade de Michigan, desenvolveram o Algoritmo Genético (AG) (HOLLAND, 1975). Já em 1962, na Holanda, os sistemas adaptativos eram estudados e foi onde primeiro se usou manipulações do tipo recombinação, frequentemente referenciado como *crossover* para a modelagem de tais sistemas (YANG, 2010). O resumo do desenvolvimento de algoritmos genéticos foi publicado em 1975. No mesmo ano, Kenneth De Jong terminou sua importante dissertação mostrando o potencial e o poder dos algoritmos genéticos para uma ampla gama de funções objetivo, com características multimodais ou mesmo descontínuos (YANG, 2011).

O Algoritmo Genético (AG) é um método de pesquisa baseado na abstração da evolução de Darwin e da seleção natural dos sistemas biológicos. Desde então, os AGs tornam-se bem sucedidos na solução de uma ampla gama de problemas de otimização com vários artigos de pesquisa e com uma grande quantidade de livros escritos.

1.2 Computação Evolutiva

Na computação evolutiva, há quatro paradigmas históricos que serviram como base para grande parte da atividade do campo: os algoritmos genéticos, a programação genética, as estratégias evolutivas e a programação evolutiva. As diferenças básicas entre esses paradigmas estão na natureza dos esquemas de representação, os operadores de reprodução e métodos de seleção (ZHANG, KIM, 1998).

A Computação Evolutiva (CE) pertence a um dos ramos da Computação Natural e consiste na otimização baseado no aprendizado de máquina e em paradigmas de classificação. Pertence à classe de sistemas solucionadores de problemas inspirados na evolução natural. A computação evolutiva baseia-se nos processos evolutivos, que são influenciados por mecanismos biológicos, ou seja, os elementos essenciais da evolução biológica que explora o espaço de soluções por herança genética, mutação e seleção dos candidatos cujas soluções são mais fortes. Dentro da computação evolutiva encontram-se os algoritmos evolutivos que são algoritmos de otimização com base populacional baseados em metaheurísticas. Os algoritmos

evolutivos mais utilizados são: os algoritmos genéticos, estratégias evolutivas, programação genética, otimização por enxame de partículas, otimização por colônias de formigas, sistema imunológico artificial, evolução diferencial, e algoritmos meméticos. Estes métodos evolutivos têm provado o seu sucesso em vários problemas de otimização complexos. A computação evolutiva oferece vantagens práticas para os pesquisadores que enfrentam problemas de otimização. Estas vantagens são várias, incluindo a simplicidade da abordagem, sua resposta robusta às diferentes circunstâncias, sua flexibilidade e muitas outras (UMA; GURUMURTHY; SINGH, 2011). A abordagem evolutiva pode ser aplicada a problemas onde as soluções heurísticas não estão disponíveis ou geralmente levam a resultados insatisfatórios. Como resultado, a computação evolutiva tem recebido um crescente interesse, particularmente no que diz respeito à maneira pela qual eles podem ser aplicados para a resolução de problemas práticos (ABRAHAM A., NEDJAH N., MOURELLE, 2006**)**.

Algumas vantagens sobre o uso dos algoritmos evolutivos podem ser citadas como (UMA; GURUMURTHY; SINGH, 2011):

- O desempenho do algoritmo evolutivo está relacionado com diferentes representações de cromossomo e de operadores de recombinação e mutação, em contraste com outras técnicas numéricas, que são aplicáveis apenas para valores contínuos ou outros conjuntos limitados.
- Os algoritmos evolutivos oferecem um *framework* de tal forma que é comparativamente fácil de incorporar o conhecimento prévio sobre o problema. Dessa forma, o uso de tais informações concentradas na busca evolutiva, produz uma exploração mais eficiente do espaço de estados de possíveis soluções.
- Os algoritmos evolutivos podem também ser combinados com técnicas de otimização mais tradicionais. Isto pode ser tão simples como a utilização de um gradiente de minimização aplicado após a pesquisa primária do algoritmo evolutivo (por exemplo, a sintonia fina de pesos de uma rede neural evolutiva), ou pode envolver a aplicação simultânea de outros algoritmos (por exemplo, hibridação com *simulated annealing* ou *tabu search* para melhorar a eficiência da pesquisa evolutiva).
- A avaliação de cada solução pode ser tratada em paralelo e apenas a seleção (que requer pelo menos uma competição em pares) requer algum processamento serial.

- Uma das principais vantagens dos algoritmos evolutivos é a capacidade de resolver problemas para os quais não existem especialistas humanos. Embora o conhecimento humano possa ser usado quando ele estiver disponível, eles são muitas vezes menos do que o mínimo adequado para automatizar rotinas de resolução de problemas.

1.2.1 Algoritmos Genéticos

Na computação evolutiva o algoritmo mais utilizado é o Algoritmo Genético (AG) que foi inventado por John Holland nos anos de 1960 (HOLLAND, 1975). John Holland apresentou os Algoritmos Genéticos como uma abstração da evolução biológica, inspirada na teoria de Darwin pela qual os AGs são capazes de desenvolver soluções para problemas do mundo real, tais como problemas de busca e otimização (BEASLEY et al., 1993; HAUPT R., HAUPT S. 2004). A abordagem genética fornece uma eficiente capacidade de busca e movimentação no espaço de solução. Essa abordagem identifica as soluções que estão mais próximas da solução ideal (solução ótima), conforme determinado por algumas medidas de distância. Algoritmos genéticos são utilizados para resolverem problemas que são difíceis de serem resolvidos através da aplicação de métodos convencionais. Em termos gerais, os algoritmos genéticos representam a técnica mais popular da computação evolutiva (MOHAMED et al., 2010).

1.2.2 Algoritmos Culturais (ACs)

Asssim como os AGs os ACs são Algoritmos Evolutivos (AEs) e tornaram-se métodos eficazes para a solução de problemas tradicionais de otimização complexos, devido às suas características, como o paralelismo implícito e a busca aleatória. No entanto, os AEs frequentemente caem na convergência prematura produzindo baixa eficiência evolutiva porque as informações implícitas incorporadas no processo de evolução e na área de conhecimento correspondente aos problemas de otimização, não são plenamente aproveitadas (GUO et al, 2011). Com a finalidade de aproveitar eficazmente as informações na evolução implícita, Reynolds (1994) propôs os Algoritmos Culturais (ACs) que são derivados do processo de evolução cultural humano. A evolução cultural permite que as sociedades evoluam ou adaptem-se a seus ambientes em taxas que excedem a da evolução biológica baseada apenas na herança genética. Os ACs possuem três componentes principais: um espaço populacional, um espaço de crença e um protocolo que descreve como o conhecimento se move entre os dois primeiros componentes (ZHANG, 2011). Nos algoritmos culturais,

(JIN; REYNOLDS, 1999), o espaço populacional pode suportar qualquer modelo computacional de base populacional tais como: Algoritmos Genéticos e Programação Evolucionária. Ao incorporar essas populações baseadas em computação evolutiva com seus operadores evolutivos dentro do *framework* do Algoritmo Cultural, o espaço de crença pode servir como um repositório de conhecimento que permanece preservado além das fronteiras geracionais. No espaço de crenças, o conhecimento implícito é extraído dos melhores indivíduos provenientes da população e armazenado em diferentes formas. Em seguida, eles são utilizados para direcionar o processo de evolução do espaço populacional, de modo a induzir a população a fugir de soluções de ótimos locais. Os ACs podem efetivamente melhorar o desempenho do processo evolutivo, além de fornecerem um modelo universal para a extração e utilização das informações evolutivas (GUO et al., 2011; ZHANG, 2011).

1.3 Problemas de Otimização Escolhidos

Nesta seção serão apresentados de forma sucinta os problemas de otimização escolhidos para testar as contribuições científicas deste trabalho e sua importância na utilização de problemas do mundo real. Esses problemas serão utilizados para avaliar os Algoritmos Culturais com abordagem memética e multipopulacional. O principal objetivo é verificar a capacidade de contribuição de tais algoritmos no sentido de extrair melhorias sobre o valor ótimo resultante em cada problema.

1.3.1 Problema de Funções Multimodais (PFM)

A otimização de Funções Multimodais é um problema que apresenta mais de um ótimo local ou apresenta apenas um ótimo global com vários ótimos locais no espaço de soluções viáveis. A maioria dos problemas do mundo real carrega características multimodais e o desenvolvimento de algoritmos eficientes para problemas de otimização multimodais ainda é uma área de pesquisa bem abrangente. O problema das funções multimodais é de grande importância para testes de robustez de algoritmos e das metaheurísticas de buscas locais. Além disso, algoritmos capazes de encontrar várias soluções ótimas (ou até mesmo um ótimo global em funções desconhecidas do tipo multimodal) auxiliam o especialista em testes e simulações.

1.3.2 Problema de Funções Restritas (PFR)

Na otimização de funções restritas o problema é formulado por um grupo de funções que satisfazem um conjunto de restrições. Existem inúmeras aplicações tais como:

otimização estrutural, projeto de engenharia, economia, problemas de alocação e de localização. Dentre os problemas restritos existentes os que mais se destacam são os problemas de engenharia. Problemas de otimização em projeto de engenharia são normalmente adotados na literatura especializada para mostrar a eficácia de otimização restrita e novos algoritmos de problemas de engenharia não-lineares têm sido investigados por muitos pesquisadores que utilizaram diferentes método para resolvê-los. A presença de restrições dá origem a um número de problemas técnicos que não são encontrados em problemas sem restrições. As restrições são muito importantes em problemas de projeto de engenharia, uma vez que normalmente são impostas na declaração do problema e que às vezes são muito difíceis de satisfazer, o que pode tornar a busca difícil e ineficiente.

1.3.3 Problema da Mochila Multidimensional (PMM/MKP)

O problema da mochila tem numerosas aplicações em teoria, bem como na prática. Uma variação do problema clássico da mochila é chamada de problema da mochila compartimentada ou multidimensional, pois a mochila pode ser dividida em compartimentos (cada compartimento da mochila, pode ser visto como um Problema da Mochila clássico) que, por sua vez, agrupam itens com características semelhantes tais como alimentos, remédios, roupas, etc. Esse tipo de problema tem grande importância na economia é exige um grau de dificuldade alto para sua solução. Tal problema surge nas linhas de produção de algumas indústrias como na produção de placas de circuitos impressos (caso bidimensional) e, também, nas indústrias de tubos de aço, onde as bobinas do estoque são cortadas em bobinas intermediárias para então serem recortadas em fitas que darão origem aos tubos, seguindo uma série de restrições físicas e técnicas. Como ilustração da importância deste problema, as sobras de material numa indústria de tubos de aço chegam a 10%, acarretando um prejuízo acima dos padrões desejados (MARQUES, 2000).

Outras aplicações práticas para esse problema são: problemas de alocação de carga, problema de corte em estoque, embalagem, controle de orçamento e gestão financeira. Existe um problema que ocorre rotineiramente que serve como um bom exemplo. Esse problema é o de cortar ou empacotar objetos em diversas situações práticas como: carregamento de contêineres ou caminhões, corte de espuma, cortes de vidro, cortes de madeira, etc. O objetivo consiste em minimizar as perdas no corte e/ou maximizar a ocupação do espaço utilizado no empacotamento (QUEIROZ et. al, 2009).

1.3.4 O Problema do Caixeiro Viajante (PCV/TSP)

O Problema do Caixeiro Viajante ou *Travel Salesman Problem* (PCV/TSP) é um dos mais tradicionais e conhecidos problemas de otimização combinatória (SOUZA, 2006). O problema de roteamento lida em sua maior parte com circuitos (ou *tours*) sobre pontos de demanda ou oferta. Dentre os tipos de circuitos, um dos mais importantes é o denominado *hamiltoniano*. Seu nome é devido a William Rowan Hamilton que em 1857 propôs um jogo chamado *Around the World*. O desafio do jogo consistia em encontrar uma rota através dos vértices do dodecaedro que iniciasse e terminasse em uma mesma cidade passando uma única vez em cada cidade. Em homenagem a Hamilton, uma solução do seu jogo passou a ser chamada de *ciclo hamiltoniano.* O PCV é importante devido a pelo menos três de suas características: grande aplicação prática, grande relação com outros modelos e grande dificuldade de solução exata.

As aplicações envolvem perfuração de placas de circuito impresso, revisão de turbinas a gás, seleção de pedidos em armazéns, roteamento de veículos e muito mais. A aplicação direta do PCV no problema de perfuração de placas de circuito impresso (PCB) tem relação com a ligação de um condutor em uma camada com um condutor em outra camada, ou para posicionar os pinos de circuitos integrados, dado que os furos têm que ser perfurados através da placa. Os orifícios podem ser de diferentes tamanhos. Para dois furos de diâmetros diferentes, consecutivamente, a cabeça da máquina tem de se mover para uma caixa de ferramenta e trocar o equipamento de perfuração, o que pode consumir muito tempo. Em turbinas a gás para que ocorra um fluxo de gás uniforme através das turbinas reduzindo a vibração, aumentando a uniformidade do fluxo e reduzindo o consumo de combustível se faz necessário escolher as melhores posições das palhetas da turbina. O problema de colocar as palhetas da melhor maneira possível pode ser modelado como um PCV com uma função objetivo especial. Outro processo que pode ser modelado pelo PCV é a seleção de pedidos em armazéns que tem relação com a seleção do material solicitado e sua coleta entre os diversos armazéns. Nesse caso, alguns veículos têm de recolher todos os itens para enviá-los aos clientes (MATAI; SINGH; MITTAL, 2010).

1.4 Dificuldades Encontradas no uso de Metaheurísticas

O grau de dificuldade em resolver um determinado problema com um algoritmo dedicado está estreitamente relacionado com a sua complexidade computacional, ou seja, a quantidade de recursos como tempo e memória necessária para fazê-lo. A complexidade computacional depende do número de elementos de entrada necessários para a aplicação do

algoritmo. Uma das classes de complexidade mais difíceis é chamada de NP (problema polinomial não-deterministico), o conjunto de todos os problemas de decisão que são resolvidos em tempo polinomial por máquinas de Turing não-determinísticas. A importância desta classe de problemas de decisão está na presença de muitos problemas de busca e de otimização para onde se deseja saber se existe uma solução.

Embora muitas tentativas tenham sido feitas, nenhum algoritmo que seja capaz de resolver um problema NP-completo com um tempo polinomial em um computador determinístico foi encontrado. Uma abordagem para a obtenção de soluções quase ótimas para os problemas em NP em tempo razoável é a aplicação de metaheurística, guiados por funções objetivo. Uma função é difícil a partir de uma perspectiva matemática se não é contínua, não diferenciável ou que tenha valores máximos e mínimos múltiplos. Em muitas aplicações do mundo real de otimização por metaheurística, as características das funções objetivo não são conhecidas antecipadamente. Nesses casos os problemas são geralmente NP ou têm complexidade desconhecida (WEISE, 2009).

Dentre os problemas encontrados por metaheurísticas, o problema da convergência prematura é o que mais se destaca. Um dos problemas na otimização global, é que não é possível determinar se a melhor solução atualmente conhecida está situada em um ótimo local ou global. Na convergência prematura, um processo de otimização converge para um mínimo local, e não é capaz de explorar outras partes do espaço de busca, mesmo que exista outra região que contenha uma solução superior. Em outras palavras, geralmente não é claro se o processo de otimização pode ser interrompido, se deve concentrar em aperfeiçoar a solução ótima atual, ou se deve examinar outras partes do espaço de busca. Isto pode, naturalmente, se tornar difícil se houver múltiplos ótimos locais, ou seja, se o problema é multimodal.

1.4.1 Dificuldades Encontradas na Otimização de Funções Multimodais (PFM)

Em problemas de otimização práticos, as funções objetivo são geralmente multimodais. Ao usar o algoritmo genético tradicional para lidar com esses problemas, o algoritmo é apenas capaz de procurar uma solução global ou local ótima devido à perda da diversidade da população. Para que a busca por soluções não fique presa em apenas uma região com um ótimo local se faz necessário manter a diversidade da população (ZHANG Y.; ZHANG H., 2011).

Os Algoritmos Evolutivos (AEs) aplicados para problemas multimodais foram investigados desde os primórdios da computação evolucionária e numerosos algoritmos e

extensões foram sugeridas para melhorar seu desempenho para tais problemas. Outro objetivo é o de desenvolver algoritmos capazes de localizar várias soluções boas. Este é uma vantagem quando as soluções precisam ser avaliadas por um especialista humano antes que uma delas possa ser utilizada na realidade. (URSEM, 2003)

1.4.1.1 Trabalhos Relacionados na Otimização de Funções Multimodais (PFM)

Parthasarathy, Goldberg e Burns (2001) apresentam um Algoritmo Memético (AM) para resolver problemas multimodais. O conceito de partilha de *fitness* é estendido para os algoritmos de busca local, definindo assim a partilha baldwiniana. Na prática, o algoritmo emprega uma técnica de partilha (isto é, uma normalização dos valores de aptidão com base nas distâncias euclidianas de ordenação/seleção, dando preferência para uma população mais espalhada), a fim de garantir a preservação da diversidade.

Em Lozano et al. (2004) um AM é proposto com dois mecanismos para preservar a diversidade. O primeiro mecanismo é o acasalamento negativo que consiste na seleção dos pais genotipicamente distantes, a fim de obter uma prole que não seja semelhante para qualquer um dos progenitores geradores. O segundo mecanismo, *Breeder Genetic Algorithm* (BGA) é um operador de mutação que promove a geração de genes distantes dentro das soluções através do emprego de uma função *ad-hoc* de distribuição de probabilidade.

Nguyen e Yao (2008) propõem um estudo experimental sobre ACs com busca local aplicado em funções multimodais (*Cultural Algorithms Iterated Local Search* CA-ILS). Os resultados dos testes mostram que o CA-ILS pode ser um método competitivo, pelo menos nos problemas avaliados.

Swapna, Devidas e Shyam (2011) propõem um algoritmo com busca local (Algoritmo Memético) baseado em enxame de partículas para funções unimodais e multimodais. O algoritmo memético proposto por eles segue outros trabalhos como os de Akbari e Ziarati (2008). Todos afirmam que alguns dos algoritmos de buscas baseados em população como Otimização por Enxame de Partículas (PSO) têm uma tendência de convergência prematura. Para evitar esse tipo de problema eles combinam a otimização por enxame de partículas e de busca local estocástica, a qual leva a uma solução que escapa o ótimo local.

1.4.2 Dificuldades Encontradas na Otimização de Funções Restritas (PFR)

O ótimo global de um problema de otimização restrito pode ocupar tanto uma área de atração viável, assemelhando-se a um problema de otimização irrestrita, ou o ocupar o

limite entre os espaços vizinhos viáveis e inviáveis, o que sugere que algumas restrições ativas, simplesmente, evitam melhorias da solução impedindo a exploração de espaços viáveis. Para este problema, é bastante intuitiva focar o esforço das buscas sobre as regiões vizinhas da fronteira. Isso parece promissor como muitos problemas de otimização restrita que tem ótimo global deste tipo (HANDOKO; KWOH; ONG, 2008).

1.4.2.1 Trabalhos Relacionados na Otimização de Funções Restritas (PFR)

Muitos pesquisadores têm desenvolvido Algoritmos Evolutivos (AEs) híbridos para problemas restritos (LE, 1995; CHUNG, REYNOLDS, 1996; WAH, CHEN, 2001; TAKAHAMA, SAKAI, IWANE, 2005; WANG et al.,2007; SUN, GARIBALDI, 2010; WEI, ZHANG, 2011).

O desenvolvimento de AEs híbridos procede em duas direções principais:

- Por um lado, duas metaheurísticas são combinadas para tirar vantagens de seus respectivos pontos fortes de busca como, por exemplo, em Le (1995) que utiliza uma combinação de lógica fuzzy e Programação Evolutiva (PE) para problemas com restrições. Em Chung e Reynolds (1996), a Programação Evolucionária (PE) também é hibridizada com GENOCOP (*GEnetic algorithm for Numerical Optimization for COnstrained Problems*) para a otimização restrita. Em Wah e Chen (2001) e em Takahama, Sakai e Iwane (2005), os AGs são combinados com *Simulated Annealing* e PSO, respectivamente.
- Por outro lado, abordagens clássicas de otimização numéricas para problemas restritos também são hibridizadas em AEs, por exemplo, Wang et al. (2007) utilizam um algoritmo evolutivo híbrido, tendo como busca local uma heurística baseada no particionamento da população em subpopulações, utilizando múltiplas recombinações com vários conjuntos de pares (pais). Sun e Garibald (2010) propõem uma nova busca local com *estimation of distribution algorithm* (EDA) para problemas restritos com uma nova estratégia para reduzir o custo computacional em termos de número de avaliações de aptidão e aumentar a possibilidade de encontrar as soluções viáveis. Wei e Zhang (2011) utilizam a busca global por enxame de partículas com um operador de busca local baseado em atração.

Uma das principais vantagens destes métodos clássicos é que eles são geralmente muito eficientes na localização de um mínimo local viável, mas a eficiência dependente

altamente da qualidade das soluções iniciais. A partir de uma solução "ruim" inicial, as abordagens clássicas poderiam apenas encontrar uma solução inviável, ou necessitam de um alto custo computacional para chegar a um ótimo local viável.

1.4.3 Dificuldades Encontradas na Otimização de Problemas da Mochila Multidimensional (PMM/MKP)

O problema clássico da mochila é um problema de otimização combinatorial que envolve a seleção de um subconjunto de itens disponíveis com a finalidade de obter o lucro máximo de modo que o peso total do subconjunto escolhido não exceda a capacidade da mochila. O problema da mochila multidimensional (*Multidimensional Kanapsack Problem* MKP) envolve muitas aplicações do mundo real tais como: a seleção de projetos, carregamento de cargas, produção industrial, orçamento de capital, planejamento de cardápio e alocação de processador em ambientes distribuídos.

Para problemas da mochila em maior escala, a eficiência dos algoritmos de aproximação se torna limitada, tanto na qualidade da solução e custo computacional. MKP é considerado um problema NP-difícil, portanto, qualquer solução de programação dinâmica irá produzir resultados em tempo exponencial. Os AGs têm sido considerados os procedimentos de busca fortes para problemas da mochila em larga escala (HILL; HIREMATH, 2005; AHMED, YOUNAS, 2011). Ao mesmo tempo em que os AGs têm uma capacidade de busca global forte, com a evolução contínua, a velocidade de convergência se torna gradualmente lenta e leva um tempo alto para se aproximar da solução global ótima ou o conjunto de soluções ótimas. Ao melhorar estas deficiências, é possível fazer um algoritmo mais eficiente e melhorar a diversidade de soluções melhores.

Alguns AEs para MKP utilizam métodos pré-otimizados para a geração da população, mas o método mais comum continua sendo o método clássico de geração aleatória da população. O uso do método clássico para a geração da população fornece carga computacional menor e diversidade suficiente no espaço da solução, mas este método não pode garantir a geração de até mesmo uma única solução viável. Uma população dominada por soluções inviáveis pode levar a retardar a inclusão de solução viável no espaço de busca durante as operações genéticas, bem como convergência lenta para a solução ótima. Além disso, uma estratégia para lidar com as soluções inviáveis deve ser definida como estratégia de rejeitar, reparar ou penalizar. Assim, o tempo de execução para heurística global tenderá a aumentar enquanto for utilizado o método convencional (AHMED; YOUNAS, 2011).

1.4.3.1 Trabalhos Relacionados na Otimização de Problemas da Mochila Multidimensional (MKP)

Cotta e Troya (1998) propuseram um AG híbrido com busca local e um AG paralelo para o MKP. O algoritmo híbrido resultante teve um desempenho melhor do que outras abordagem que utilizam AG padrão. O AG paralelo é executado com 2 e 4 AGs e os resultados em relação ao número de ótimos encontrados são muito bons.

Puchinger e Raidl (2005) consideram a combinação paralela de um Algoritmo Memético (AM) e um algoritmo *branch-and -cut* para o MKP. Hong et al. (2007) discutem taxas de migração em um AG multipopulacional para MKP. As taxas de migrações são baseadas em valores fixo ou valores dinâmicos. Os resultados mostram melhores soluções quando se utiliza valores dinâmicos nas taxas de migração. Os trabalhos de Jagtap, Pani e Shinde (2011) demonstram o uso de AGs paralelos com hibridização dos melhores atributos de ilhas em dois modelos *master slave* para MKP.

1.4.4 Dificuldades Encontradas na Otimização do Problema do Caixeiro Viajante (PCV/TSP)

O Problema do caixeiro viajante (PCV) é um problema NP-difícil de otimização combinatória. Na literatura existe uma série de algoritmos baseados em metaheurística que foram aplicados a este problema para a obtenção de melhores resultados em tempos computacionais aceitáveis.

Os AGs podem encontrar boas soluções para o problema do caixeiro viajante, no entanto, depende muito da forma como o problema é codificado, quais os métodos de recombinação e mutação são utilizados e a quantidade de elementos que representam o problema combinatorial (por exemplo, o número de cidades). Alguns algoritmos determinísticos clássicos como programação dinâmica, *branch-and-bound* e outros só são aplicáveis para o PCV de pequeno porte. Para o PCV com tamanho médio e com grandes entradas, eles são praticamente incapazes de resolver o PCV, porque o tempo de computação necessário é muito longo. Até agora, uma maneira eficaz de resolver grandes PCV em escala é a aplicação de algoritmos bio-inspirados como algoritmos genéticos (AGs) e Otimização por Colônia de Formigas (ACO) (MICHALEWICZ, 1996; WANG, CAO, 2002; DUAN, WANG, YU, 2006). O AG cai facilmente em um ótimo local e causa a estagnação do processo de otimização, que normalmente necessita ser melhorado quando utilizado no problema de combinatória. Os métodos que utilizam informação heurística melhoram o desempenho do algoritmo.

1.4.4.1 Trabalhos Relacionados na Otimização do Problema do Caixeiro Viajante (PCV/TSP)

O AG com melhorias como, por exemplo, a busca tabu (*Tabu search*) e *Simulated Annealing* (SA) dá mais ênfase na busca global e pode melhorar a eficácia da busca (JIAO, WANG, 2000; DELIN, LIXIAO, ZHIHUI, 2011). Karaboga e Gorkemli (2011) propõem uma colônia de abelhas artificial para o PCV. Eles utilizam dois exemplos para teste e conseguem bons resultados.

Delin, Lixiao e Zhihui (2011) apresentam um AG com *Simulated Annealing* (SA) para o PCV, onde heurísticas de inversão são utilizadas para gerar uma nova solução. Uma solução viável no novo vizinho é gerada através das duas etapas seguintes: primeiro são selecionados aleatoriamente dois pontos em uma solução. Em seguida se inverte a *substring* entre os dois pontos, enquanto os outros elementos da solução permanecem inalterados.

1.5 Contribuições

O presente trabalho está focado no estudo comportamental sobre os resultados provenientes da aplicação dos Algoritmos Evolutivos Culturais através do uso do espaço populacional e dos operadores genéticos provenientes dos Algoritmos Genéticos. Os problemas de otimização abordados nesse trabalho são de grande importância e interesse prático, uma vez que os resultados satisfatórios encontrados nesses tipos de problemas podem ser adaptados e aplicados a outros problemas similares. Dessa forma, as soluções encontradas para os problemas de otimização abordados nesse trabalho, no sentido de garantir resultados tão bons e/ou melhores que outros algoritmos de otimização, estão relacionados com a hibridização de heurísticas de buscas locais (abordagem memética) e na introdução de multipopulações no espaço populacional. As contribuições para cada tipo de abordagem são descritas nos tópicos seguintes.

1.5.1 Abordagem Memética

Na abordagem memética escolheu-se a heurística *simulated annealing*. Nesse contexto, as principais contribuições desse trabalho são:

- Proposição de estratégias de buscas representadas por um conjunto de estruturas de vizinhanças (perturbações de soluções) chamadas pelo algoritmo Simulated Annealing, atuando como um algoritmo de busca local, tendo como base algumas características presentes nas funções multimodais;

- Proposição dos algoritmos geradores de vizinhanças baseado nas estratégias de vizinhanças proposta (GerarVizinhoPFM( ) e GerarVizinhoPCV( )).
- Proposição da busca local inserida em dois pontos do algoritmo cultural baseada nos melhores indivíduos encontrados e em outros escolhidos aleatoriamente;
- Proposição de uma função de penalidade para a otimização dos problemas restritos (PFR);
- Composição de um conjunto de heurísticas de melhorias baseadas em Simulated Annealing, 2-opt e 3-opt para o problema do caixeiro viajante.
- Proposição de novos parâmetros para o processo de busca local, com sugestões de sintonia dos mesmos.

1.5.2 Abordagem Multipopulacional

O presente trabalho também inclui a abordagem multipopulacional que é uma adaptação do modelo de ilhas sobre a estrutura padrão do algoritmo cultural aqui identificado como AC Modelo de Ilhas (AC-MI) (ou CA *Island Model* CA-IM), brevemente introduzida em Silva e Oliveira (2009b) e que posteriormente foi detalhada por Silva, Teixeira e Oliveira (2012). As implementações tornaram-se simples porque as estruturas dos ACs foram adaptadas tanto para comportar multipopulações na estrutura do espaço populacional, criando-se o espaço multipopulacional, quanto para atualizar as subpopulações na estrutura do espaço de crenças, criando-se o espaço de crenças multipopulacional. A principal característica presente no AC-MI é a ligação entre o espaço de crenças da população principal e espaço de crença das subpopulações. Eles armazenam informações sobre a evolução independente tanto da população principal quanto das subpopulações, ou seja, as transformações culturais que ocorrem em paralelo entre a população principal e as subpopulações das ilhas.

Assim, as contribuições com a abordagem multipopulacional são:

- Proposição de um *framework* mutipopulacional para ACs do tipo *master-slave* baseado em AGs paralelos para o problema da mochila multidimensional;

- Proposição do protocolo de comunicação do framework mutipopulacional responsável pela migração dos indivíduos entre os espaços de crenças.
- Aplicação do *framework* mutipopulacional no problema da mochila multidimensional.
- Aplicação de operadores (mutação/recombinação) estáticos e dinâmicos nas diferentes subpopulações.

1.5.3 Outras contribuições (Framework Evolutivo Java)

Outra contribuição é a construção de uma ferramenta baseada em um *framework* Evolutivo Java (FEJ) que facilita a combinação de metaheurísticas com outras heurísticas bem como o mapeamento dos problemas contínuos e discretos. Toda a descrição do problema como intervalo de variavéis, tipo de recombinação e mutação, algoritmo utilizado para otimização, buscas locais são descritos em um padrão XML. Cada ensaio é descrito como um arquivo XML que é carregado pelo *framework* desenvolvido. O *framework* Java (FEJ) é baseado no JCLEC (http://jclec.sourceforge.net/) que é um sistema de software para pesquisa em Computação Evolutiva (CE), desenvolvido na linguagem de programação Java. Ele fornece uma plataforma de software de alto nível para fazer qualquer tipo de Algoritmo Evolutivo (AE), fornecendo suporte para algoritmos genéticos (codificação binário, inteiro e real), a programação genética e programação evolutiva.

1.6 Estrutura da Tese

No Capitulo 2 será apresentada uma revisão sobre o Algoritmo Genético padrão (serial) e uma visão geral sobre o Algoritmo Genético paralelo. No Capítulo 3 será apresentada uma revisão sobre Algoritmos Culturais e em seguida no Capítulo 4, a abordagem memética envolvendo os algoritmos evolutivos e as buscas locais. No Capitulo 5 serão apresentados as contribuições desenvolvidas nesse trabalho bem como os detalhes da implementação dos Algoritmos Culturais com Abordagem Memética e Multipopulacional. Os resultados alcançados através de diversos experimentos utilizando *benchmarks* serão apresentados no Capítulo 6 e em seguida as conclusões no Capítulo 7.

2 ALGORITMO GENÉTICO (AG)

2.1 Introdução

Na natureza, cada indivíduo de uma população concorre uns com os outros na busca de recursos como alimento, abrigo e assim por diante. Também na mesma espécie, os indivíduos competem para atrair parceiros para reprodução. Devido a esta seleção, um indivíduo com desempenho insatisfatório tem menos chance de sobreviver, e os indivíduos mais adaptados produzem um número relativamente grande de descendentes. Nota-se também que durante a reprodução, uma recombinação das boas características de cada ancestral pode produzir descendentes melhores, cuja aptidão é maior do que a de seus pais. Depois de algumas gerações, as espécies evoluem espontaneamente para tornar-se mais e mais adaptadas ao seu ambiente. Em 1975, Holland (HOLLAND, 1975) desenvolveu essa ideia em seu livro "A adaptação em sistemas naturais e artificiais". Basicamente ele descreveu como aplicar os princípios da evolução natural para problemas de otimização. Holland estudou formalmente o fenômeno da adaptação natural e desenvolveu formas para que seu mecanismo pudesse ser importado para sistemas de computadores, construindo assim os Algoritmos Genéticos (AGs) (LINDEN, 2006; SONKAR et al., 2012).

Um algoritmo genético (AG) é uma técnica de otimização e de busca, utilizada na computação com base nos princípios da genética e seleção natural para encontrar soluções exatas ou aproximadas para problemas de otimização. Algoritmos Genéticos (AGs) são classificados como heurísticas de busca global e pertencem a uma classe particular de Algoritmos Evolutivos (AEs), que usam técnicas inspiradas pela biologia evolutiva como hereditariedade, mutação, recombinação e seleção (KUMAR et al., 2010).

Um AG permite que uma população composta por vários indivíduos evolua sob as regras de seleção específicas para um estado que maximiza ou minimiza sua função de custo ou de aptidão (*fitness*). O método desenvolvido por John Holland (1975) ao longo dos anos 1960 e 1970 foi finalmente popularizado por um de seus alunos, David Goldberg (GOLDBERG, 1989). Goldberg foi capaz de resolver um problema difícil, envolvendo o controle de transmissão em gasoduto em sua dissertação e resumiu o trabalho de Holland em seu livro. Ele foi o primeiro a tentar desenvolver uma base teórica para AGs através do teorema dos esquemas (HAUPT R.; HAUPT S., 2004). O trabalho de De Jong (1975) demonstrou a utilidade do AG para a otimização de funções e fez o primeiro esforço concentrado para encontrar parâmetros otimizados para AGs (SONKAR et al., 2012).

O método de Holland é especialmente eficaz porque não só analisou o papel da mutação, mas ele também utilizou a recombinação genética (cruzamento - *crossover*). O cruzamento das soluções parciais permite melhorar significativamente a capacidade do algoritmo em abordar e, eventualmente, encontrar a melhor solução, sendo considerado um operador-chave para o processo de evolução natural. Muitas pessoas acreditam que essa mistura de material genético através da recombinação genética (*crossover*) é uma das características mais fortes dos AGs. A mutação é a outra forma de obter novos genótipos. A mutação consiste na mudança do valor dos genes. Na evolução natural, mutação gera mais genomas não viáveis. De fato, a mutação não é um operador muito frequente na evolução natural. No entanto, algumas mudanças aleatórias podem ser uma boa forma de explorar o espaço de busca rapidamente, contribuindo assim para o processo de otimização (LINDEN, 2006).

O Algoritmo Genético levanta algumas características importantes. Primeiro, é um algoritmo estocástico; sendo a aleatoriedade um papel essencial nos algoritmos genéticos. Tanto a seleção quanto a reprodução necessita de procedimentos aleatórios. Um segundo ponto muito importante é que os algoritmos genéticos sempre consideram uma população de soluções. Dessa forma, os AGs mantêm em memória mais do que uma única solução em cada iteração oferecendo assim uma série de vantagens. O algoritmo pode recombinar soluções diferentes para obter as melhores e assim, ele pode usar de forma eficaz os benefícios da diversidade. Um algoritmo de base populacional também é muito favorável para a paralelização (SIVANADAM, 2007). Os AGs fornecem a capacidade de pesquisa por soluções oferecendo a capacidade de se movimentar no espaço de soluções. Esta abordagem identifica soluções que são mais próximas da solução ótima, que pode ser determinada por algumas medidas de distância (MOHAMED; ZIDAN, 2010). A robustez do AG também deve ser mencionada como algo essencial para o sucesso do algoritmo. Ao falar em robustez é importante destacar que existe uma discussão muito grande a respeito de sua definição. De acordo com Beyer e Sendhoff (2006) o termo “robustez“ pode significar:

- a capacidade de um algoritmo de otimização para lidar com diferentes cenários de otimização (por exemplo, diferentes classes de problemas de otimização);
- que o desempenho do algoritmo seja insensível em relação às configurações específicas dos parâmetros do algoritmo (por exemplo, taxa de mutação ou parâmetro aprendizagem em algoritmos evolutivos);

- robustez com relação a detalhes de implementação;
- robustez da solução produzida pelo algoritmo, também referido como "robustez da solução".

Beyer e Sendhoff (2006) adotam, por exemplo, a quarta definição levando-se em consideração os seguintes aspectos:

- Insensibilidade da solução final para diferentes inicializações;
- Insensibilidade da solução final em relação a fontes internas ou externas de perturbações e ruídos que são de natureza aleatória, resultando assim em funções objetivas não determinísticas.

Apesar das diversas discussões é comum referirem-se a robustez do algoritmo como sendo a capacidade do algoritmo em apresentar uma boa consistência e um bom comportamento sobre uma ampla gama de classes de problemas.

Não existe qualquer requisito particular sobre o problema antes de usar AGs, de modo que podem ser aplicado para resolver uma variedade de problemas que possam ser modelados por uma função de fitness. Todas essas características fazem do AG uma ferramenta de otimização realmente eficaz no processo de otimização (SIVANADAM, 2007).

Dessa forma, os AGs foram apresentados como uma abstração da evolução biológica e derivou seu comportamento a partir de uma metáfora genética evolutiva. Algoritmos Genéticos (AGs) pertencem ao ramo dos algoritmos Evolutivos e como tal podem ser definidos como uma técnica de busca baseada numa metáfora do processo biológico de evolução natural. Os AGs são o grupo com uma maior variedade de métodos e ferramentas de aplicação tornando-se uma boa opção para resolver problemas de busca e otimização (LINDEN, 2006).

2.2 Características dos AGs

Os AGs são considerados técnicas probabilísticas. Dessa forma, os AGs podem encontrar diferentes soluções a cada vez que é executado, mesmo que sejam fixados os mesmos parâmetros e a mesma população. Os AGs são programas de simples implementação e requerem apenas informações sobre o ponto avaliado, bem como sua adequabilidade à solução do problema. Nesse caso não há necessidade de derivadas ou qualquer outro tipo de informação adicional, o que facilita sua aplicação em situações onde os dados são discretos e não possuem derivadas (LINDEN, 2006).

Os AGs são caracterizados como heurísticas de busca no espaço de soluções e diferentemente dos esquemas enumerativos, não realizam buscas em todos os pontos de soluções possíveis, mas apenas em subconjunto desses pontos. O problema do caixeiro viajante (PCV), por exemplo, é um problema muito conhecido e o número de possíveis soluções é proporcional ao fatorial do número de cidades. Nesse caso, o uso dos AGs torna-se uma ótima opção para solução do problema, pois realizam uma busca direcionada, através do mecanismo de seleção com informações pertinentes ao problema. Isso significa que, apesar dos pontos de busca a serem percorridos sejam aleatórios, a busca passa a ser direcionada, pois utilizam informações históricas na tentativa de encontrar novos pontos de busca com bom desempenho. Dessa forma, é correto afirmar que os AGs não podem ser chamados de buscas aleatórias não direcionadas (LINDEN, 2006).

Os AGs contam com o uso de seleção, operadores de cruzamento/mutação e substituição por gerações de novos indivíduos. Intuitivamente um AG cria sucessivas gerações de indivíduos cada vez melhores através da aplicação de operações muito simples. A busca por soluções são guiadas apenas pelo valor da aptidão associada a cada indivíduo pertencente a população. Esse valor é usado para classificar os indivíduos em função da sua aptidão, que tem relação com o problema a ser resolvido. A avaliação do indivíduo influencia na aplicação dos operadores genéticos, tendo-se em vista a sobrevivência do indivíduo mais apto (SIVANANDAM, 2007).

2.3 Classificações de Algoritmos Genéticos

Os algoritmos genéticos fornecem uma busca robusta em problemas complexos comprovados teoricamente e empiricamente. Os AGs vêm melhorando o processo de busca tornando-o eficiente e eficaz em diversos domínios de problema e, portanto, ampliando sua aplicabilidade para a área dos negócios, áreas científicas e de engenharia. Estes algoritmos são computacionalmente menos complexos e apresentam boa eficácia no processo de busca por melhorias. Estas características vêm permitindo diferentes abordagens. A primeira é a abordagem tradicional conhecida com AG Simples (AGS - tradicional com uma população). A segunda é a abordagem paralela conhecida como AGs paralelos, ou seja, paralelismo com uma ou várias populações (SIVANANDAM, 2007). A definição de Algoritmos Genéticos Paralelos (AGP) está relacionada desde a execução em modo paralelo de várias operações dos AG Simples (AGS) até o compartilhamento de indivíduos em modelos paralelos multipopulacionais. O principal objetivo do AGP é reduzir o tempo de execução de grandes dimensões que estão associadas com algoritmos genéticos simples para que encontrem as

melhores soluções em espaços de busca grandes.

2.3.1 Algoritmo Genético Simples (AGS)

Muitas técnicas de busca necessitam de informações auxiliares para trabalhar corretamente. Por exemplo, técnicas de gradiente necessitam de derivadas e outros procedimentos como a técnica GRASP (*Greedy Randomized Adaptive Search Procedure* - Procedimento de busca adaptativa gulosa e randômica) requer acesso em muitos parâmetros tabulares. O método GRASP, por exemplo, é um método iterativo, proposto por Feo e Resende (1995), sendo constituído por duas fases: uma fase chamada de construção, onde uma solução é gerada elemento a elemento, e outra chamada de fase de busca local, onde um ótimo local na vizinhança da solução construída é pesquisado. A melhor solução encontrada proveniente de todas as iterações GRASP realizadas é retornada como resultado (SOUZA, 2011). Em termos de comparação, os AGs não necessitam de todas estas informações auxiliares. Sendo assim, o mecanismo de um Algoritmo Genético Simples (AGS) não envolve nada mais complexo do que copiar seqüências de símbolos e trocar parcialmente essas sequências. A forma de correspondência entre todas as soluções possíveis para uma questão específica é conhecida. O conjunto de todas as soluções que atendem a essa forma constituem o espaço de busca. O problema consiste em descobrir a solução que se encaixa melhor, ou seja, uma com mais ganhos possíveis dentre todas as soluções possíveis. Se for possível enumerar rapidamente todas as soluções, o problema não apresenta muita dificuldade. Mas, quando o espaço de busca se torna maior, a enumeração dessas soluções deixa de ser simples porque levaria muito tempo. Neste caso é necessário usar uma técnica específica para encontrar a melhor solução (SIVANANDAM, 2007). Resumindo, um AGS é útil e eficaz, quando:

- O espaço de busca é grande, complexo ou mal compreendido.
- O conhecimento sobre o domínio é escasso ou o conhecimento especialista é difícil de codificar para um espaço de busca consideravelmente menor.
- Não existe possibilidade de análise matemática do problema.
- Os métodos de busca tradicionais falham.

Os cinco passos básicos usados em algoritmo genético simples para resolver um problema são (CHAMBERS, 1999):

- A codificação do cromossomo de modo a representar o problema.

- Uma população inicial de soluções.
- O cálculo da função de aptidão (*fitness*).
- Método de seleção de soluções para produzir novas soluções.
- Operadores de recombinação e mutação para produzir novas soluções através das soluções já existentes.

2.3.1.1 Indivíduos

Uma solução para um problema em consideração é representada por um indivíduo. O conjunto de indivíduos considerados é chamado de população. Cada indivíduo tem um cromossomo que codifica as características de seus dados. Então, um cromossomo é uma seqüência de "alelos" que representam um *quantum* de informações, tais como bits, dígitos, letras, etc. A representação alternativa de dados requer codificação e decodificação, a fim de trocar soluções com o nominal espaço de objeto (KUMAR et al., 2010).

Duas formas de soluções podem ser agrupadas no cromossomo de acordo com a necessidade (SIVANANDAM, 2007). Uma delas é o **genótipo** onde o cromossomo contém a informação "genética" pura e outra é o **fenótipo** que é a expressão do cromossomo em função de um modelo (Figura 1).

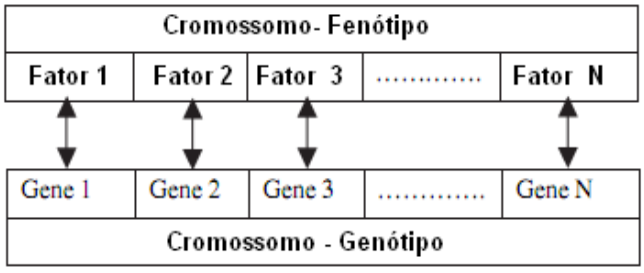


Figura 1: Representação do Fenótipo e Genótipo (SIVANANDAM, 2007).

2.3.1.2 Genes

Um cromossomo está subdividido em genes. Um gene é a "instrução" básica de um cromossomo. Um gene é a representação simples de um fator para um fator de controle. Cada fator no conjunto de solução corresponde a um gene no cromossomo.

Um cromossomo deve conter, de alguma forma, informação sobre a solução que representa (Figura 2).

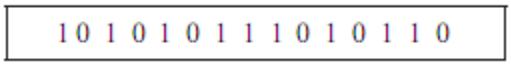


Figura 2: Representação de um cromossomo.

Um gene pode ser representado por uma seqüência de bits de comprimento arbitrário dentro de intervalo delimitado pelos seus limites inferiores e superiores (Figura 3).

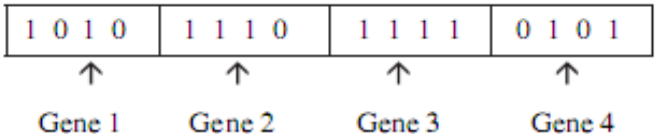


Figura 3: Representação de genes

Esta função tem uma solução única como um parâmetro e retorna um número que indica quão boa é a solução (aptidão). O número pode ser de inteiros ou reais em um dado intervalo. Para o cálculo da aptidão o cromossomo tem de ser primeiramente decodificado e a função objetivo tem que ser avaliada. A aptidão não só indica o quão boa é a solução, mas indica também o quanto o cromossomo está próximo da solução ótima.

2.3.1.3 População

Uma população é um conjunto de indivíduos. A população é constituída por um número de indivíduos que serão avaliados, por parâmetros que definem o fenótipo dos indivíduos e algumas informações sobre o espaço de busca. Os dois aspectos importantes da população utilizada nos AGs são: a geração da população inicial e o tamanho da população.

Tradicionalmente, a população é gerada aleatoriamente, cobrindo toda a gama de soluções possíveis (o espaço de busca). Ocasionalmente, as soluções podem ser "semeada" em áreas onde as melhores soluções possam ser encontradas. Ou seja, pode haver casos em que a inicialização da população seja realizada por algum método conhecido (KUMAR, 2010). O tamanho da população dependerá da complexidade do problema. Idealmente, a primeira população deve ter um conjunto de genes tão grande quanto possível, a fim de ser capaz de explorar todo o espaço de busca.

2.3.1.4 Codificação

A codificação é um processo de representar os genes individuais. O processo pode ser executado usando bits, números, árvores, matrizes, listas ou quaisquer outros objetos.

2.3.1.5 Seleção

Durante cada geração, uma proporção da população existente é selecionada para

produzir uma nova geração. Soluções individuais são selecionadas através de um processo baseado na aptidão de cada indivíduo (fitness), onde o indivíduo com solução mais apta é tipicamente mais provável de ser selecionado. Certos métodos de seleção classificam a aptidão de cada individuo e, preferencialmente, selecionam as melhores soluções. A maioria das funções é estocástica, concebidos de forma que uma pequena proporção de soluções dos indivíduos menos aptos também seja selecionada. Isso ajuda a manter a diversidade da população, evitando assim, a convergência prematura sobre soluções inferiores. Métodos de seleção mais populares e bem estudados incluem seleção por roleta e seleção do torneio (KUMAR et al., 2010).

Alguns exemplos para tipos de seleção são (CHAMBERS, 1999):

- Roleta - cada indivíduo da população tem uma fatia na roleta, proporcional a solução que representa, através da sua aptidão (*fitness*). As Melhores soluções têm proporcionalmente seções maiores da roleta, portanto, há uma chance maior de que eles sejam selecionados.
- Seleção Torneio – escolhem-se duas ou *n* soluções de forma aleatória e utiliza uma estratégia de torneio, onde um número de rodadas é executado. Em cada rodada um número de soluções são reunidas para competir, e apenas a melhor solução vencedora é selecionada.

2.3.1.6 Mutação, Recombinação e Substituição da População.

Depois de selecionar as soluções é necessária uma técnica ou técnicas de combinação dessas soluções de uma maneira que irá produzir novas e boas soluções. Mutação e recombinação são processos que geram variações, de modo similar aos sistemas biológicos, e podem ser incluídas no ambiente artificial de modo a permitir que os indivíduos evoluam com ligeiras modificações. Estes dois processos de mutação e de cruzamento têm diferentes papéis em satisfazer as variações. Enquanto a mutação altera uma propriedade específica de um modo randômico fornecendo novas alternativas para o ambiente de simulação, o cruzamento funde propriedades das duas soluções individuais, a fim de formar uma nova solução, que tem o potencial de utilizar as boas propriedades dos indivíduos pais (DUGAN; ERKOÇ, 2009).

No processo de mutação novas soluções são alteradas de uma forma totalmente aleatória, isso ajuda a manter a diversidade da população, que de outra forma não poderiam ser produzidas sozinhas por recombinação.

2.3.1.6.1 Operador de Recombinação

Recombinação ou reprodução sexual é um operador-chave para a evolução natural. Tecnicamente, a partir de dois genótipos produz-se um novo genótipo pela mistura dos genes originais. Em biologia, a forma mais comum de recombinação é o crossover (ou *crossing-over*), onde novos cromossomos surgem pela duplicação e combinação de dois cromossomos. O efeito de recombinação é muito importante porque permite que as características pertencentes a dois pais diferentes sejam variadas. Se o pai e a mãe possuem diferentes qualidades, se espera então que todas essas boas qualidades sejam repassadas para os descendentes (SIVANANDAM, 2007).

As formas mais comuns de recombinação são: um ponto, dois pontos, n-ponto e recombinação uniforme. No cruzamento de um ponto, um ponto aleatório é escolhido ao longo do cromossomo dos pais, cada metade dos cromossomos é então trocado. Por exemplo, considere dois pais (CHAMBERS, 1999):

pai_l	**1234 | 5678**
pai_2	**8765 | 4321**

Se o cruzamento for realizado escolhendo-se um ponto na metade de cada um, dois descendentes serão gerados:

descendente_l	**1234 | 4321**
descendente_2	**8765 | 5678**

O cruzamento de dois pontos é semelhante ao de um ponto, mas há dois pontos de cruzamento.

pai_l	**123|456|78**
pai_2	**876|543|21**

Isso resulta em um pai que mantém a cabeça e a cauda de seu cromossomo, ganhando uma nova seção média. O outro progenitor irá reter sua secção média e ganhar uma nova cabeça e cauda.

descendente_l	**123|543|78**
descendente_2	**876|456|21**

O cruzamento de *n* pontos é a mesma técnica, mas *n* pontos são selecionados, onde há troca de partes dos cromossomos para cada ponto de cruzamento. O cruzamento uniforme seleciona número de pontos *x* aleatórios menor do que o comprimento dos

cromossomos. Cada ponto selecionado é então trocado com os demais.

2.3.1.6.2 Operador de Mutação

Na natureza, a duplicação do DNA às vezes pode resultar em erros, o DNA também é vulnerável a danos no dia-a-dia o que também resulta em erros. Esses erros ou mutações às vezes podem resultar em boas características, que podem ser unidas através de reprodução sexual e, eventualmente, levar a novas espécies. Em computação evolutiva podemos imitar a mutação, gerando erros na prole. Um ponto no cromossomo da prole pode ser definido com um valor aleatório.

2.3.1.6.3 Substituição da População

Durante a última etapa, os indivíduos da população anterior são descartados e substituídos por novos indivíduos. Normalmente dois pais são extraídos de uma população de tamanho fixo, onde dois filhos são criados, mas nem todos os quatro podem voltar para a população. Assim dois indivíduos devem ser substituídos. Nessa situação pode haver um método para determinar quais dos indivíduos da população atual devem ser substituídas por novas soluções. Basicamente, existem dois tipos de métodos para manter a população:

- **Atualização da geração** – em uma população de N indivíduos uma prole de tamanho N é produzida para substituir a população inteira;
- **Estado Estacionário**- apenas uma parte da população é substituída, escolhendo-se geralmente os piores indivíduos ou os indivíduos mais antigos da população.

2.3.2 Algoritmos Genéticos Paralelos e Distribuídos (AGP e AGD)

Apesar dos AGS serem bem sucedidos em muitas aplicações e em domínios muito bem diferentes, existe alguns problemas que podem ser tratados de alguma forma com AGs Paralelos (AGP) tais como (NOWOSTAWSKI, 1999):

- Se em um determinado problema, a população precisa ser muito grande e a memória necessária para armazenar cada indivíduo tende a ter um tamanho considerável, tornando impossível executar uma aplicação de forma eficiente em uma única máquina.

- Quando a avaliação da função de aptidão (fitness) normalmente demanda muito tempo.
- Um AG seqüencial pode ficar preso em uma sub-região ótima do espaço de busca, assim, tornando-se incapaz de encontrar soluções de melhor qualidade. A busca por subespaços diferentes no espaço de busca em paralelo torna-se mais viável.

A vantagem mais importante do AGP é que em muitos casos, eles fornecem um desempenho melhor do que algoritmos baseados em população única, mesmo quando o paralelismo é simulado em máquinas convencionais. A razão é que múltiplas populações permitem a especificação, que é um processo pelo qual diferentes populações evoluem em direções diferentes (SIVANANDAM, 2007). Portanto, as técnicas relacionadas ao AGP operam sobre uma população de indivíduos em paralelo fazendo a busca em diferentes áreas do espaço de solução, alocaando um número de membros apropriado para a busca em várias regiões (ZUBEN, 2000; PESSINI, 2003).

A idéia básica por trás da maioria dos programas paralelos é dividir um problema grande em tarefas menores para resolver as tarefas simultaneamente usando vários processadores. Esta abordagem de dividir para conquistar pode ser aplicada pelo AGs em muitas maneiras diferentes, e a literatura contém numerosos exemplos de bem sucedidas implementações paralelas. Alguns métodos de paralelização utiliza uma única população, enquanto outros dividem a população em várias subpopulações relativamente isoladas. Alguns métodos exploram arquiteturas de computadores maciçamente paralelos, enquanto outros são mais adequadas para computadores com elementos de processamento menor e superior conectados por uma rede mais lenta. (CANTU-PAZ, 2000).

A execução paralela de vários AGs Simples (AGS) é chamada de AGP (Algoritmos Genéticos Paralelos). Eles são usados para resolve diversos problemas (tais como *Job shop scheduling)* que fazem uso de diversas restrições de precedência para alcançar uma boa otimização. Algoritmos Genéticos Paralelos têm sido desenvolvidos para reduzir o tempo de execução de grandes dimensões que estão associadas à AGS para encontrar soluções quase ótimas em espaços de busca maiores. Os AGPs possuem ganhos consideráveis em termos de desempenho e escalabilidade. Os AGPs podem ser facilmente implementados em redes de computadores heterogêneos ou em mainframes paralelos. A maneira como os AGs podem ser paralelizados dependem dos seguintes elementos:

- Forma de avaliação da função de aptidão e a aplicação da mutação;
- Se a seleção é aplicada localmente ou globalmente;
- Se forem utilizadas uma simples subpopulação ou múltiplas subpopulações;
- Se múltiplas populações são utilizadas como é feita a migração dos indivíduos?

A maneira mais simples de paralelização de um AG é executar várias cópias do mesmo AGS, uma em cada núcleo de processamento. Cada um dos núcleos de processamento começa com uma subpopulação diferente. Cada subpopulação evolui e finaliza de forma independente uma das outras. O AGP termina quando todos os núcleos de processamento param. Não há comunicações entre as unidades de processamento. As diferentes modalidades de AGP são:

- AGP independentes – Sua vantagem é que cada núcleo começa com uma subpopulação independente. Essa diversidade das subpopulações reduz a chance de que todos os núcleos de processamento possam convergir prematuramente para uma mesma solução de qualidade ruim. Esta abordagem é equivalente a simplesmente tomar a melhor solução, após várias execuções do AGS em diferentes populações.
- AGP com Migração – essa abordagem melhora a modalidade de AGP independente, com as migrações periódicas dos cromossomos entre os núcleos para evitar convergência prematura e compartilhar soluções de alta qualidade. As migrações dos cromossomos ocorrem após algumas iterações, onde cada núcleo de processamento envia uma cópia de seu melhor cromossomo local para outro núcleo. O cromossomo recebido substitui o pior cromossomo local a menos que um cromossomo idêntico já exista na população local.
- AGP com Partição - significa particionar o espaço de busca em subespaços disjuntos e forçar os núcleos de processamento a efetuarem buscas em subespaços diferentes.
- AGP com Segmentação - A segmentação começa por segmentar um caminho em sub-caminhos. Então, depois de melhorias efetuadas nos sub-caminhos, eles são recombinados em caminhos mais longos.

- AGP com Segmentação-Migração- É a combinação da segmentação e da migração. A recombinação ocorre no final de cada fase, subcaminhos são controlados por um grupo de núcleos de processadores numerados em ordem crescente.

Os AGPs podem ser implementados utilizando-se a abordagem paralela padrão e a abordagem de decomposição. Na primeira abordagem, o modelo AG sequencial (AGS) é implementado em um computador paralelo, dividindo a tarefa de execução entre os processadores. Na abordagem de decomposição, a população inteira passa a existir na forma distribuída. As múltiplas subpopulações existentes podem estar na forma independente ou na forma de interação (Granulação Grossa ou AG distribuídos) ou só existe uma população com cada membro da população interagindo apenas com um número limitado de membros (Granulação Fina). As interações entre as populações ou os membros da população, tem relação com uma estrutura espacial de um problema. Estes modelos mantêm subpopulações mais diversificadas para evitar o problema de convergência prematura. Eles também se encaixam no modelo de evolução, com um grande grau de independência da subpopulação.

Existem mais de dez diferentes métodos de paralelização de AGs. As classes que mais se destacam são (NOWOSTAWSKI, 1999; CANTU-PAZ, 1998; KNYSH, KUREICHIK, 2010):

- AGs Paralelos com Paralelização Global (Mestre-Escravo);
- AGs Paralelos com Granulação Grossa;
- AGs Paralelos com Granulação Fina.

2.3.2.1 AGP com Paralelização Global (*Master-Slav*e ou Mestre-Escravo)

Este método, também conhecido como avaliação da aptidão distribuída, é uma das primeiras aplicações bem sucedidas de AGs Paralelos. É conhecida como paralelização global, modelo mestre-escravo com população simples, ou avaliação da aptidão distribuída (CANTU-PAZ, 2000). O algoritmo usa uma única população e a avaliação dos indivíduos e/ou a aplicação de operadores genéticos é realizada em paralelo. A seleção e a comparação são feitas globalmente, portanto, cada indivíduo pode competir com qualquer outro.

A operação que é mais comumente paralelizado é a avaliação da função de aptidão, porque normalmente exige apenas o conhecimento do indivíduo que está sendo avaliado (e não da população inteira), e assim não há necessidade de se comunicar durante

essa fase. Isso geralmente é implementado através de programas mestre-escravo (*master-slave*), onde o mestre armazena a população e os escravos avaliam a aptidão. A Figura 4 representa esse modelo.

A paralelização de avaliação da aptidão é feita mediante a atribuição uma fração da população para cada um dos processadores disponíveis (no caso ideal um indivíduo por núcleo de processamento).

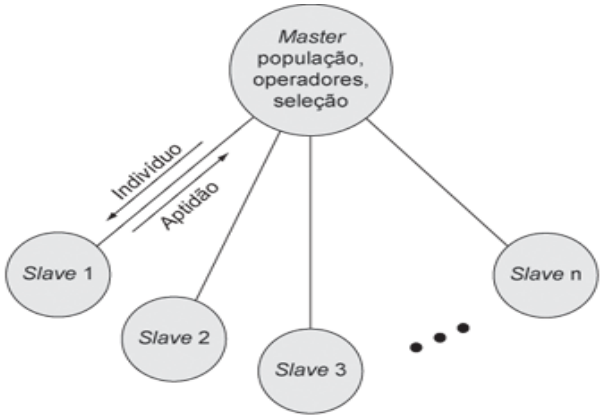


Figura 4: Um esquema de um AG paralelo mestre-escravo (CANTU-PAZ, 2000).

A comunicação ocorre apenas quando cada escravo recebe o indivíduo (ou um subconjunto de indivíduos) para avaliar e, quando devolvem os valores de aptidão. O algoritmo é dito ser ***síncrono***, se o mestre pára e aguarda para receber os valores de aptidão para toda a população antes de prosseguir com a próxima geração. Os AGs síncronos mestre-escravo têm exatamente as mesmas propriedades que um AG simples, exceto pela sua velocidade, ou seja, esta forma de AG paralelo executa exatamente a mesma busca que um AG simples. Uma versão ***assíncrona*** do AG mestre-escravo também é possível. Neste caso, o algoritmo não para, para esperar qualquer resultado de um processador que esteja lento. Por esta razão, o AG mestre-escravo assíncrono não funciona exatamente como um AG simples, pois a seleção é realizada com uma fração da população que já foi processada, ou seja, opera sobre a população existente.

Um AG paralelo mestre-escravo síncrono é relativamente fácil de implementar, além de se esperar uma aceleração significativa se o custo das comunicações não dominar o custo computacional. No entanto, há um gargalo clássico nesse modelo. O processo inteiro tem que esperar o processador mais lento para terminar as suas avaliações de aptidão. Só depois disso é que o operador de seleção pode ser aplicado. O AG mestre-escravo assíncrono supera isso, mas como dito antes, o algoritmo altera significativamente a dinâmica do AG.

Talvez a forma mais fácil de implementar um AG paralelo mestre-escravo e aplicar a seleção de torneio considerando apenas a fração de indivíduos da população cuja aptidão já foram avaliados.

No entanto, é importante notar que os ganhos de desempenho não podem crescer indefinidamente quando se utiliza mais escravos. Portanto, há um limite que se for ultrapassado pela adição de novos escravos torna o algoritmo mais lento do que um AG Simples com processamento serial. A razão é que quanto mais escravos são utilizados, maior será o tempo de comunicação de informações gasto entre os processos. Se o tempo de comunicação sobrepõe o tempo de computação então não deverá ocorrer ganhos nesse modelo (CANTÚ-PAZ, 1999).

2.3.2.2 AGP com Granulação Grossa (AGs Distribuidos)

São também definidos como AGs de Subpopulações Estática com Migração ou AGs paralelos com múltiplos *demes* (múltiplos *demes* significa múltiplas populações) ou AGs Distribuídos. O termo AGs com Granulação Grossa se refere a um modelo com um número relativamente pequeno de *demes* com muitos indivíduos em cada *deme*. Estes modelos são caracterizados pelo tempo relativamente longo que levam para processar (de forma sequencial) cada geração de cada *deme*, e pelas eventuais altas taxas de comunicações realizadas para a migração de indivíduos. Algumas vezes os AGs com Granulação Grossa (c*oarse grained*) são conhecidos como AGs distribuídos porque são geralmente implementados em computadores com memória distribuída MIMD (*Multiple Input Multiple Data*). Essa abordagem também é adequada para redes heterogêneas.

São os algoritmos mais sofisticados e consistem em várias subpopulações de indivíduos que são trocados ocasionalmente (Figura 5).

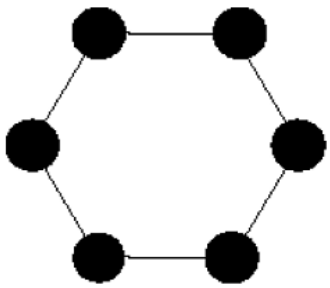


Figura 5: Um esquema de AG paralelo com múltiplas população (múltiplos *deme*s) (CANTU-PAZ, 2000).

Esta troca de indivíduos é chamada de migração, que é controlada por vários parâmetros. AGs com múltiplos *demes* são muito populares, mas também pertencem a uma classe de AGs paralelos com maior dificuldade de compreensão, porque os efeitos da migração não são totalmente compreendidos. Esse modelo introduz mudanças fundamentais no funcionamento dos AGs e possuem um comportamento diferente daqueles descritos nos AGs simples.

As características importantes da classe de AGs com múltiplos *demes* são o uso de múltiplas subpopulações (*demes*), as quais evoluem de maneira independente, e a presença de um operador de migração. Os *demes* são separados um do outro (isolamento geográfico), e os indivíduos competem apenas dentro de um *deme*. De vez em quando, alguns indivíduos são copiados de um *deme* para outro através de um processo chamado de migração.

A migração de indivíduos de um *deme* para o outro é controlada por diversos parâmetros, como:

- A topologia que define as conexões entre os subpopulações. As topologias comumente usadas incluem: malha hipercubo bidimensional/tridimensional, torus, etc;
- A taxa de migração que controla quantidade de indivíduos que migram;
- Um esquema de migração, que controlam quais são os indivíduos que devem migrar de um *deme* escolhido (melhor, pior, aleatório) para outro *deme*, e quais os indivíduos que serão substituídos (pior, aleatório, etc);
- Um intervalo de migração que determina a freqüência de migrações.

Se os indivíduos puderem migrar para qualquer outro *deme*, o modelo é chamado de modelo de ilhas (*island model*). Além desses, existem outros possíveis modelos de migração. O modelo de ilhas apresenta subpopulações geograficamente separados com tamanho relativamente grande. As subpopulações podem efetuar intercâmbio de informações ao longo do tempo, permitindo que alguns indivíduos migrem de uma subpopulação para outra de acordo com diversos padrões. A principal razão para esta abordagem é re-injetar periodicamente diversidade em outras subpopulações. Considera-se, nesse caso, que as diferentes subpopulações tendem a explorar diferentes partes do espaço de busca. Dentro de cada subpopulação um algoritmo genético padrão sequencial é executado entre as fases de migração (KNYSH; KUREICHIK, 2010).

A migração é um dos aspectos mais importantes nesse modelo que é controlada

por três parâmetros: (1) a taxa de migração, o número de indivíduos que vão migrar, (2) uma agenda de migração, que determina quando as migrações irão ocorrer, e (3) a topologia de comunicação entre as subpopulações. (CANTÚ-PAZ, 1999). Várias topologias de migração têm sido utilizadas tais como: a estrutura do anel, malhas 2-D e 3-D, hipercubos, sendo mais comum o uso de gráficos aleatórios (SIVANANDAM, 2007).

2.3.2.3 AGP com Granulação Fina

Esse modelo é referido muitas vezes como *modelo de vizinhança* ou *modelo maciçamente paralelo*, que mantém a sobreposição de áreas vizinhas.

Nos AGs que implementam esse modelo, há apenas uma única população com uma estrutura espacial que limita a interação entre os indivíduos da população. Os indivíduos podem competir e casar somente com seus vizinhos (Figura 6).

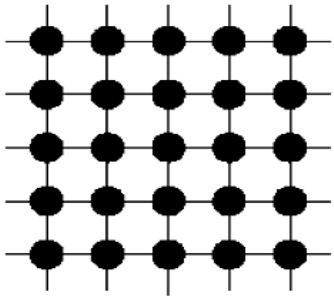


Figura 6: Um esquema de um AG paralelo com granulação fina (CANTU-PAZ, 2000).

Este modelo é adequado para execução em computadores maciçamente paralelos, podendo também ser simulado em um cluster de estações, com a desvantagem de envolver um alto custo de comunicação.

A vizinhança onde a seleção é aplicada (para casamento, reprodução e substituição) se restringe a uma região local para cada indivíduo. O que se considera como vizinho de cada indivíduo depende da topologia utilizada. Por exemplo, se a população está organizada em cima de algum tipo de estrutura esférica, os indivíduos podem ter permissão para recombinar com (e forçados a competir com) seus vizinhos dentro de um determinado raio (BACK, 1997).

Nesse modelo é necessário um grande número de processadores, porque a população está dividida em um grande número de pequenos *demes*. A comunicação entre os *demes* é realizada usando um operador de migração, ou utilizando *demes* de sobreposição. A avaliação da aptidão é feita simultaneamente para todos os indivíduos. A seleção e a

reprodução ocorrem localmente dentro de uma pequena vizinhança (SIVANANDAM, 2007).

2.4 Considerações sobre os AGs

É importante compreender que os AGs não dão garantia de sucesso. Um problema representado em um sistema estocástico ou genético pode ficar longe demais da solução, ou uma convergência muito rápida pode interromper o processo da evolução. Estes algoritmos, no entanto, são extremamente eficientes, e são utilizados em domínios tão diversos como bolsa de valores, programação de produção ou de programação de Robôs de montagem na indústria automobilística.

Os AGs podem até mesmo ser mais rápido em encontrar máximos globais do que métodos convencionais, principalmente quando os usos de derivadas fornecem informações enganosas. É importante ressaltar que na maioria dos casos em que métodos convencionais podem ser aplicados, os AGs tornam-se mais lentos porque eles não têm informações auxiliares sobre o problema. Nestes casos, não há necessidade de aplicar AGs, pois teríamos soluções menos precisas depois de muito tempo de computação.

3 ALGORITMOS CULTURAIS

3.1 Introdução

A computação evolutiva utiliza conceitos metafóricos, princípios e mecanismos de extração da compreensão de como sistemas naturais são capazes de ajudar a resolver problemas computacionais considerados complexos. Atualmente muitos trabalhos são focados no processo de seleção natural e genética como os algoritmos genéticos (HOLLAND, 1975) que sãoferramentas que mapeiam a evolução biológica. No entanto, os AGs não são suficientes para modelar a evolução cultural que é um traço fundamental das sociedades humanas. Embora existam pontos de coincidência, a evolução cultural tem suas próprias características diferenciais. Entre eles o fato de que a evolução cultural se torna mais rápida do que a evolução genética e da existência de um espaço de crença compartilhado que permite que os indivíduos incorporarem diretamente o conhecimento sem ter de reaprender tudo do zero. Reynolds (1994) deixa claro essa diferença quando afirma que a evolução cultural permite que as sociedades envolvam ou adaptem seu meio ambiente a taxas que excedem a evolução biológica, que se baseia apenas na herança genética.

A pesquisa sobre algoritmos culturais foi conduzida por observações sobre a cultura e até que ponto sua influência sobre os indivíduos pode ser levada em consideração para se criar um sistema que utilize tal hereditariedade. A tomada de decisão tanto humana quanto animal é fortemente influenciada pelo conhecimento adquirido através da observação do comportamento dos outros, e quando os padrões de comportamento entre os indivíduos estão espalhados ao longo de gerações passa a ser considerada como uma forma de evolução cultural (DANCHIN et al., 2004).

Reynolds (1994) desenvolveu alguns modelos para investigar as propriedades de algoritmos culturais. Nestes modelos, um espaço de crença é usado para restringir a combinação de traços que os indivíduos possam assumir aceitando ou não um comportamento em um determinado nível. Na próxima seção serão apresentados alguns conceitos importantes que estão ligados ao processo de evolução cultural.

3.2 Características Culturais

Para compreender o processo de evolução cultural, é importante conhecer o conceito de *meme*. O *meme* é uma adaptação do grego *mimema* (que significa algo imitado). Dawkins introduziu a noção de *memes* definindo-os como homólogos de genes no mundo

cultural (DAWKINS, 1976). *Memes* são, portanto, a unidade de transmissão cultural ou imitação tal como um pedaço de pensamento, um fragmento de música e assim por diante.

3.2.1 *Meme*

Um *meme* passou a ser então um termo cunhado em 1976 por Richard Dawkins em "O Gene Egoísta" (DAWKINS, 1976):

> "O *meme* é para a memória o análogo do gene para genética, ou seja, a sua unidade mínima. O *meme* também é considerado como uma unidade de informação que se multiplica de cérebro em cérebro, ou entre locais onde a informação é armazenada (como livros) e outros locais de armazenamento ou cérebros. Podem ser idéias ou partes das idéias, línguas, sons, desenhos, capacidades, valores estéticos e morais, ou qualquer outra coisa que possa ser aprendida facilmente e transmitida enquanto unidade autônoma".

O estudo dos modelos evolutivos da transferência de informação é conhecido como memética (BLACKMORE, 2007). A cultura, na linguagem sociológica, é tudo o que resulta da criação humana. O homem cria, transforma e é afetado por essas transformações. O homem, ao produzir cultura, produz-se a si mesmo. Não há cultura sem o homem, como não há homem sem que haja cultura (DAWKINS, 2001). As modificações trazidas por uma geração passam à geração seguinte, de modo que a cultura transforma-se perdendo e incorporando aspectos mais adequados à sobrevivência, reduzindo o esforço das novas gerações. A principal vantagem da cultura é o chamado mecanismo adaptativo: a capacidade de responder ao meio de acordo com mudança de hábitos, mais rápida do que uma possível evolução biológica.

3.2.2 Evolução Cultural

De acordo com Dawkins (2001), a cultura pode evoluir de modo muito similar ao das populações de um organismo, pois entre as gerações muitas idéias podem ser passadas, o que podem aumentar ou diminuir a sobrevivência dos indivíduos que as obtêm e as usam. Esse processo posui um mecanismo de seleção das idéias que continuarão a serem passadas às gerações futuras. Por exemplo, cada cultura pode possuir métodos e projetos únicos para a construção de determinadas ferramentas, mas a que possuir métodos mais eficazes - assumindo que todas as outras variáveis se conservam inalteradas - irá provavelmente prosperar sobre as outras culturas. Isso leva a adoção desses métodos, que serão usados cada

vez mais por uma fração maior da população com o passar do tempo. Cada projeto de ferramenta funciona então da mesma forma que um gene biológico (que pode existir em algumas populações, mas não em outras). A presença desse mesmo projeto nas gerações futuras é diretamente afetada pela sua eficácia enquanto *meme.*

3.2.3 Propagação de *Memes*

O *meme* é propagado por imitação, conceito proposto pelo sociólogo francês Gabriel Tarde onde o elemento ambiental pode ser inanimado (como é o caso dos livros). Os *memes* (ou comportamentos adquiridos e propagados por imitação) apenas podem ser observados num reduzido número de espécies terrestres, caso dos hominídeos, dos golfinhos, e de aves que aprendem a cantar por imitação dos seus progenitores. Assim com os genes, os *memes* também sobrevivem para além dos indivíduos que os transportam. Um gene bem sucedido (como é o caso dos genes para dentes fortes numa população de leões) pode conservar-se sem mutações no conjunto de genes de uma população por centenas de milhares de anos.

3.2.4 Evolução dos *Memes*

Para que ocorra a evolução, não basta a existência de mecanismos de hereditariedade e de seleção natural. É igualmente necessária uma possibilidade de mutação, propriedade que também é atribuída aos *memes*. É o caso das ideias que são transmitidas de cérebro em cérebro que podem sofrer modificações que se acumulam ao longo do tempo.

Se um cientista ouve ou lê uma ideia boa ele a transmite aos seus colegas e alunos (artigos e conferências). Se a ideia “pegar”, pode-se dizer que ela se propaga, espalhando-se de cérebro em cérebro.

Essas modificações no "fenótipo" (a informação nos cérebros ou sistemas retentores) são então transmitidas sob uma nova forma. Por exemplo, os contos populares e mitos são frequentemente recontados com o objetivo de serem mais bem recordados, aumentando, dessa forma, a probabilidade de serem recontados. Contrariamente à história dos genes, que sofrem mutações aleatórias, as mutações nos *memes* geralmente são intencionais. São "pessoas" que alteram os contos na intenção de melhorá-los ou de que eles sejam recontados, apresentando ou não sucesso.

3.3 Princípios do Algoritmo Cultural

Os Algoritmos Culturais (ACs) foram propostos por Robert Reynolds

(REYNOLDS, 1994; REYNOLDS, 1999; REYNOLDS, 2001a; REYNOLDS, 2001b; REYNOLDS, 2003; REYNOLDS et al., 2010) como um complemento à metáfora evolutiva utilizada na Computação Evolutiva, metáfora essa que se concentra nos aspectos genético da evolução e na teoria da seleção natural proposta por Darwin. Dessa forma, os Algoritmos Culturais são Algoritmos Evolutivos baseados tanto na metáfora genética quanto no processo de evolução cultural da humanidade (REYNOLDS, 1994).

O princípio do algoritmo cultural pode ser abordado através da definição de cultura. Segundo Reynolds e Zhu (2001), o termo cultura aqui pode ser entendido como um veículo de armazenamento de informações acessíveis globalmente por todos os membros da sociedade e que pode ser útil em direcionar suas atividades para solução de problemas.

A cultura também pode ser vista como um conjunto de fenômenos ideológicos compartilhados por uma população, mas por meio do qual um indivíduo pode interpretar as suas experiências e decidir sobre o seu comportamento. Nestes modelos fica claro que o conhecimento é a parte do sistema que é compartilhado pela população e compilado por membros da sociedade, com uma codificação que permita o acesso a todos. Ao mesmo tempo distingue a parte do sistema que é individual, que significa também incluir as experiências individuais, e como essas experiências podem trazer algo para o conhecimento compartilhado (BECERRA, 2002).

3.4 Inspiração Natural

Sugere-se que ao longo dos tempos o ser humano evoluiu um conjunto único de capacidades que permitiu a formação, codificação e transmissão de informações culturais. O conhecimento deve ser codificado de forma a ser acessível a todos os indivíduos de uma sociedade. Uma vez codificado, esse conhecimento é assimilado por cada indivíduo da sociedade sob o prisma das suas experiências anteriores, podendo este indivíduo incorporar ou não um novo conhecimento àqueles presentes na sua sociedade.

Algoritmos culturais baseiam-se nas teorias de alguns sociólogos e arqueólogos, que tentaram modelar a evolução cultural sugerindo que a evolução cultural pode ser vista como um processo de herança, em dois níveis: o nível microevolutivo, que é o material genético herdado dos pais para sua prole e o nível macroevolutivo, que é o conhecimento adquirido por pessoas físicas através das gerações, e que uma vez codificados e armazenados, utilizados para orientar o comportamento de indivíduos pertencentes a uma população (BECERRA, 2002).

Reynolds et al. (2010) confirma esse sistema duplo de herança nos ACs focando

também o processo de herança em dois níveis (microevolutivo e macroevolutivo). No nível microevolutivo tem-se a modelagem da população em si composta por um conjunto de indivíduos, levando-se em consideração qualquer modelo evolutivo baseado em populações. Já no nível macroevolutivo é modelado o conhecimento adquirido pelos indivíduos ao longo das gerações e que codificado e armazenado no espaço de crenças, ajuda a guiar o comportamento dos indivíduos em suas populações. É importante ressaltar que as informações culturais podem ser transmitidas tanto entre indivíduos de uma população quanto de uma população para outra.

Os algoritmos culturais aceleram a taxa de convergências dos Algoritmos Evolutivos ou melhoram as populações geradas através de um mecanismo dual de herança. Dessa forma, a evolução cultural é mais rápida do que a evolução genética. Isso permite uma melhor adaptação ao ambiente do que a possível adaptação pela genética (REYNOLDS; ZANONI, 1992): *"A **evolução cultural** habilita as sociedades a evoluir ou a se adaptar ao seu ambiente em taxas que excedem as da evolução biológica baseada somente na herança genética"*. Na próxima seção será apresentado o funcionamento geral de um Algoritmo Cultural.

3.5 Funcionamento Básico de um Algoritmo Cultural

Os Algoritmos Culturais (ACs) proposto por Reynolds (REYNOLDS, 1994; REYNOLDS et al., 2010; ALI M.; REYNOLDS, ALI R., 2010), são descritos por dois componentes principais: Espaço Populacional e Espaço de Crenças (Figura 7).

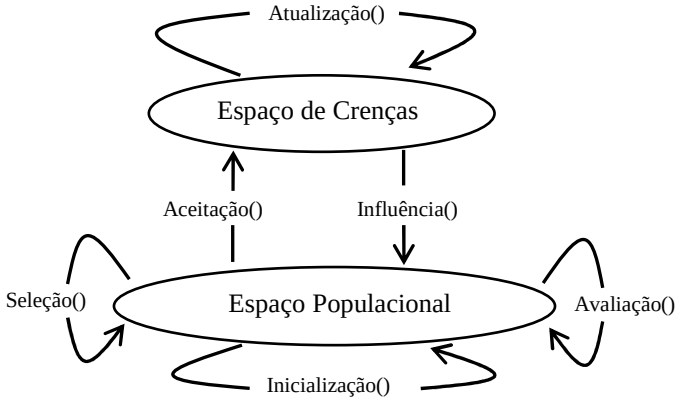


Figura 7: Funcionamento Básico de um Algoritmo Cultural (REYNOLDS et al., 2010).

É importante ressaltar que não é obrigatório a implementação de todas as propriedades descritas. Mesmo assim elas serão apresentadas para que se mantenha a completude. Os dois componentes principais de um Algoritmo Cultural são:

- Espaço Populacional: conjunto de soluções que pode ser modelado utilizando qualquer técnica de Inteligência Computacional que faça uso de uma população de indivíduos;
- Espaço de Crença (Mapa do grupo): local onde ocorre o armazenamento e representação do conhecimento (experiência ou mapas individuais) adquirido ao longo do processo evolutivo. É a partir desse conhecimento armazenado que os indivíduos são guiados na direção das melhores regiões do espaço de busca.

Os indivíduos são descritos por um conjunto de características e comportamentos e por um mapa que generaliza os conhecimentos e experiências adquiridas por esse indivíduo. Essas características e comportamentos são modificados por operadores genéticos que podem ser influenciados socialmente.

Da mesma maneira, o conhecimento e experiências são unidos e modificados para formar o espaço de crenças. Para fins de comparação entre a evolução genética e cultural é apresentado o pseudocódigo do **algoritmo genético** e o pseudocódigo do **algoritmo cultural** (REYNOLDS, 2003).

Algoritmo Genético

```
01:INÍCIO
02:     t=0                          ;primeira geração
03:     inicializar população P(t)        ;população inicial aleatória
04:     avaliar população P(t)            ;calcula f(i) para cada indivíduo
05:     ENQUANTO (não condição_fim) FAÇA
06:          t=t+1                   ;próxima geração
07:          selecionar P(t) de P(t-1)
08:          altera P(t)                  ;crossover e mutação
09:          avaliar população P(t)     ;calcula f(i) para cada indivíduo
10:     FIMENQUANTO
11:FIM
```

A cada nova geração de indivíduos é feita uma avaliação e o conhecimento dos melhores indivíduos pode ou não fazer parte do Espaço de Crença. A nova população a ser gerada é influenciada com o conhecimento anteriormente armazenado através de operadores.

Algoritmo Cultural

```
01: INÍCIO
02: t=0                                  ;primeira geração
03: inicializar população P(t)           ;população inicial aleatória
04: Inicializar Espaço de Crença EP(t)
05: avaliar população P(t)               ;calcula f(i) para cada indivíduo
06: ENQUANTO (não condição_fim) FAÇA
07:     Comunicação (P(t), EP(t));       ;votação(Aceitação)
08:     Atualização EP(t);               ;uso de operadores culturais
09:     Comunicação (EP(t), P(t));       ;promoção (função de influência)
10:     t←t+1                            ;próxima geração
11:     selecionar P(t) de P(t-1)
12:     altera P(t)                      ;crossover e mutação
13:     avaliar P(t)                     ;calcula f(i) para cada indivíduo
14: FIMENQUANTO
15:FIM
```

Tanto o espaço de crença quanto os indivíduos de uma população podem ser influenciados pelo que se chama de Protocolos de Intercomunicação. A comunicação entre os indivíduos de uma geração e o Espaço de Crenças é dada por um protocolo dito Função de Aceitação. Já a interação do Espaço de Crenças com a população de indivíduos é chamada de Função de Influência.

3.6 Principais Características

De acordo com Reynolds (2003), as principais características demonstradas por um Algoritmo Cultural são:

- **Mecanismo Dual de Herança**: são herdadas características tanto no nível da população quanto no nível do espaço de crenças;
- **Evolução Guiada por Conhecimento**: a população é guiada na direção que, segundo o conhecimento armazenado no espaço de crenças, seja a melhor;
- **Suporta Hierarquização**: tanto a população quanto o espaço de conhecimento podem ser organizados de forma hierárquica, permitindo a criação de nichos e, ao mesmo tempo, uma distribuição do conhecimento adquirido;
- **Conhecimento Sobre o Domínio Separado dos Indivíduos**: todo o conhecimento adquirido é armazenado no espaço de crenças e

compartilhado entre os indivíduos; assim, quando um indivíduo é eliminado da população, o conhecimento adquirido pelo mesmo permanece.

- **Suporte a Auto-Adaptação em Vários Níveis**: permite tanto a auto-adaptação da população quanto do conhecimento e a forma como o conhecimento é adquirido. Ou seja, os parâmetros de controle, a representação, os operadores (tanto genéticos quanto sociais), a avaliação dos indivíduos, e o protocolo de intercomunicação podem ser alterados a qualquer momento da evolução;
- **Diferentes Taxas de Evolução**: a evolução das populações e do conhecimento não precisa ocorrer na mesma taxa. Segundo Reynolds e Zanoni (1992), o conhecimento é evoluído a uma taxa dez vezes maior que a população;
- **Funcionamento**: é um modelo computacional que permite a modelagem de diversas formas de evolução cultural.

De acordo com Alotto e Coelho (2009) os ACs consistem de três componentes básicos: componente população (espaço populacional), componente conhecimento (espaço de crença) e o componente protocolo de comunicação. O desenvolvimento de um Algoritmo Cultural é baseado no desenvolvimento desses três componentes. Esses componentes serão apresentados nas próximas subseções.

3.7 Microevolução e Macroevolução

Como visto anteriormente os Algoritmos Culturais (ACs) utilizam um mecanismo dual de herança. Esse mecanismo permite que os ACs explorem tanto a microevolução quanto a macroevolução. A microevolução diz respeito à evolução que acontece no nível populacional e a macroevolução é a que ocorre sobre a cultura em si, ou seja, a evolução do espaço de crenças.

3.7.1 Espaço Populacional

O espaço populacional contém a população a ser evoluída e os mecanismos para sua avaliação, modificação, reprodução (ALOTTO; COELHO 2009). No Espaço Populacional são representadas as características e comportamentos dos indivíduos. Essa representação pode ser feita através de qualquer técnica que faça uso de uma população de indivíduos, como é o caso dos Algoritmos Genéticos, Programação Evolutiva, Programação

Genética, Evolução Diferencial, Sistemas Imunes, entre outros (JIN; REYNOLDS, 1999b). O conhecimento produzido no espaço populacional na perspectiva do nível microevolutivo é seletivamente aceito ou passado para o espaço de crença e usado para adaptar suas estruturas simbólicas (IACOBAN; REYNOLDS; BREWSTER, 2003b).

3.7.2 Espaço de Crenças

O Espaço de Crenças é o repositório de símbolos que representam o conhecimento adquirido pelo Espaço Populacional ao longo do processo evolutivo. O Espaço de Crenças permite que os indivíduos sejam removidos da população sem que o conhecimento por eles adquiridos seja perdido, ou seja, se durante o processo de evolução um indivíduo bom é perdido, o seu conhecimento armazenado é propagado para outras gerações. Este conhecimento pode ser usado para influenciar as alterações feitas pela população na próxima geração. Em um algoritmo cultural a informação adquirida por um indivíduo da população pode ser compartilhado com a população inteira (ALOTTO; COELHO, 2009). O Espaço de Crenças foi criado para guiar os indivíduos em busca de melhores regiões. Nos trabalhos realizados por Reynolds e Saleem (2003) foram identificadas cinco categorias básicas de conhecimento, que são úteis na tomada de decisões (REYNOLDS, SALEEM, 2003; REYNOLDS, ALI, 2008):

- *conhecimento situacional*: exemplos de soluções com sucesso e sem sucesso, etc.;
- *conhecimento normativo:* intervalos de comportamentos aceitáveis;
- *conhecimento de domínio*: conhecimento de objetos do domínio, suas relações e interações;
- *conhecimento topográfico*: padrões espaciais de comportamento;
- *conhecimento histórico*: padrões temporais de comportamento.

Este conjunto de categorias é visto como completo para um dado domínio, no sentido de que todo conhecimento disponível pode ser expresso em termos de uma dessas classificações.

3.7.2.1 Conhecimento Situacional

O Conhecimento Situacional fornece um conjunto de casos tidos como exemplo que são úteis para a interpretação de experiências específicas individuais. O conhecimento

situacional leva os indivíduos a avançar em direção dos indivíduos tidos como exemplares. Esta foi a primeira fonte de conhecimento utilizado com algoritmos culturais e foi inspirado em abordagens elitistas nos algoritmos genéticos. Esta fonte de conhecimento colabora com o conhecimento de domínio para explorar regiões acima da média (REYNOLDS; ALI, 2008).

Representa a classificação dos melhores indivíduos encontrados até um determinado momento da evolução. Segundo Iacoban, Reynolds e Brewster (2003b) o conhecimento situacional contém um conjunto de indivíduos da população que servem como exemplo para o resto da população. A quantidade de exemplos pode variar de implementação para implementação, mas costuma ser pequena. A Figura 8 contém um exemplo da estrutura utilizada para representar esse tipo de conhecimento. Cada indivíduo aqui considerado como um exemplar (E_1, E_2,..,E_n) é armazenado junto com suas características ou variáveis (X_1, X_2,.., X_n) e com sua aptidão f(x).

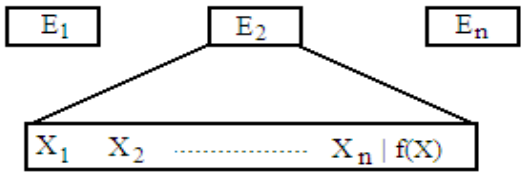


Figura 8: Representação do Conhecimento Situacional (IACOBAN;REYNOLDS;BREWSTER, 2003b).

O Conhecimento Situacional é atualizado sempre que é encontrado um indivíduo cuja aptidão supere a aptidão do pior indivíduo armazenado.

3.7.2.2 Conhecimento Normativo

Representa um conjunto de intervalos que caracterizam os intervalos de valores assumidos pelas características que compõem as melhores soluções. Esses intervalos servem para guiar os ajustes (mutações) que ocorrem nos indivíduos.

O conhecimento normativo é um conjunto de intervalos de variáveis promissores que fornecem padrões para os comportamentos individuais e diretrizes permitindo a realização de ajustes individuais. O conhecimento normativo entrou em cena durante o aprendizado de regras para aplicações de sistemas especialistas (REYNOLDS; ALI, 2008).

A estrutura de dados utilizada por Reynolds está representada na Figura 9, onde são armazenada cada variável do cromossomo correspondente a *n* variáveis V_1, V_2,...,V_n com seus respectivos intervalos. Esses intervalos correspondem a mínimos e máximos das variáveis (*l* e *u*) e mínimos e máximos das suas respectivas aptidões (*L* e *U)* (IACOBAN; REYNOLDS; BREWSTER, 2003b).

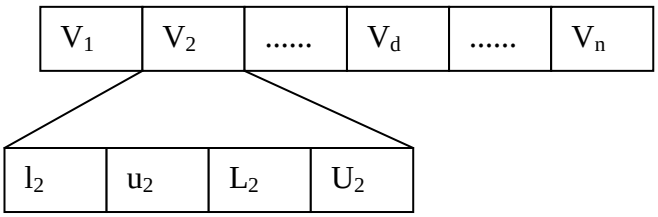


Figura 9: Representação de Conhecimento Normativo (IACOBAN; REYNOLDS; BREWSTER, 2003b).

O ajuste do intervalo do Conhecimento Normativo varia de acordo com o melhor indivíduo. Ou seja, se o indivíduo passou pela função de aceitação e seu intervalo é menor que o intervalo armazenado no espaço de crença, o intervalo é reajustado e vice-versa.

3.7.2.3 Conhecimento do Domínio

Como o próprio nome pressupõe, é específico de cada aplicação. Ele representa o conhecimento sobre o domínio do problema para guiar a busca. Esse é o tipo de conhecimento menos utilizado, pois é o mais difícil de ser extraído e representado.

3.7.2.4 Conhecimento Topográfico

O Conhecimento topográfico foi originalmente proposto em razão de padrões de paisagem funcionais baseado em regiões. Sua função principal é extrair padrões de comportamento do espaço de busca. Esse tipo de conhecimento pode espalhar indivíduos sobre todo o espaço de busca identificando regiões promissoras, fazendo com que novos indivíduos as explorem.

Esse tipo de conhecimento foi motivado, em conjunto com problemas de mineração de dados, onde o espaço do problema era tão grande que foi necessário criar uma forma sistemática de particionamento do espaço durante o processo de pesquisa. Um estado é associado a cada região do espaço que pode variar dinamicamente à medida que novas sub-regiões são descobertas e adicionadas (REYNOLDS; ALI, 2008).

De acordo com Coello (2002) esse conhecimento pode ser representado através de uma árvore *kd* (Figura 10). Em uma árvores *kd*, cada nó pode ter apenas dois filhos (ou nenhum, se ele é um nó folha), e representa uma divisão no meio de qualquer uma das k dimensões. Para inicializar o conhecimento topográfico, é só criar o nó raiz, que representa todo o espaço de busca, e contém a melhor solução encontrada na população inicial.

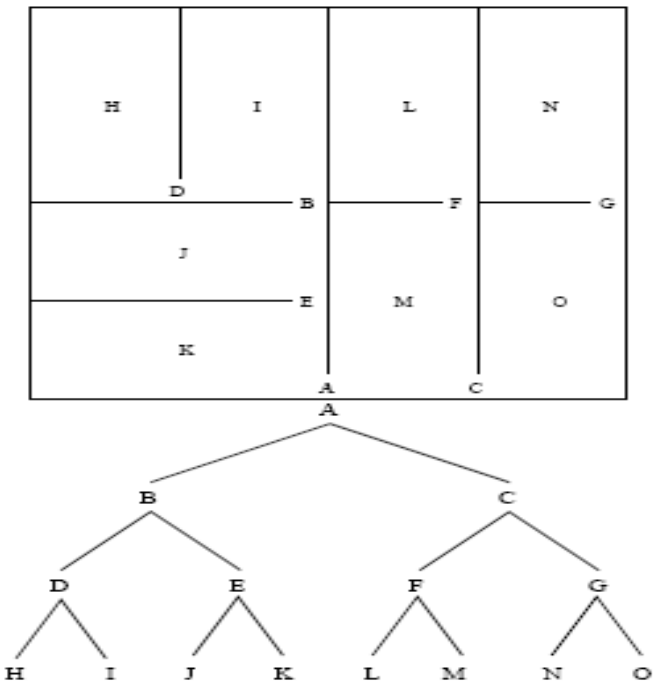


Figura 10: Representação de Conhecimento Topográfico (BECERRA; COELLO, 2005).

A função de influência tenta mover os filhos para qualquer uma das B células da lista:

$$x'_{i,j} = \begin{cases} x_{i,r3} + F^* \mid x_{i,r1} - x_{i,r2} \mid if \;\; x_{i,r3} < l_{i,c} \\ x_{i,r3} - F^* \mid x_{i,r1} - x_{i,r2} \mid if \;\; x_{i,r3} > u_{i,c} \\ x_{i,r3} + F^* (x_{i,r1} - x_{i,r2}) \;\; otherwise \end{cases}$$

onde $l_{i,c}$ e $u_{i,c}$ representam respectivamente os limites inferiores e superiores da célula c , escolhida aleatoriamente através da lista das melhores células b.

A atualização do Conhecimento Topográfico é feito através de uma função de atualização quando se encontra um novo indivíduo melhor do que existe na célula. A função divide um nó da árvore se: uma solução melhor é encontrada na célula e se a árvore não tiver atingido a sua profundidade máxima.

A dimensão em que a divisão é feita, é a única que tem a maior diferença entre a solução armazenada e a nova solução de referência (ou seja, a nova solução considerada como a "melhor" encontrada até agora). Em relação à memória utilizada para a representação do conhecimento, essa representação é mais eficiente (BECERRA; COELLO, 2005).

3.7.2.5 Conhecimento Histórico

Tem relação com o processo de busca durante a evolução. Através de um processo

de Monitoramento guarda-se importantes eventos que ocorrem durante a busca. Esse conhecimento foi motivado pela necessidade de desenvolver aprendizado em ambientes dinâmicos. Indivíduos guiados pelo conhecimento histórico podem consultar aqueles eventos armazenados para guiar suas decisões quanto a qual direção seguir. A estrutura utilizada para representar o Conhecimento Histórico é demonstrada na Figura 11 onde e_i representa o melhor indivíduo (exemplar) encontrado antes da *i-ésima* alteração do ambiente. A distância média das mudanças para a característica *i* é representada por ds_i. Se existem mudanças para a característica *i*, dr_i representa a direção média dessas mudanças.

e_1	.....	e_i	.....	e_w

ds_1	ds_2	.....	ds_n
dr_1	dr_2	.....	dr_n

Figura 11: Representação de Conhecimento Histórico (BECERRA; COELLO, 2005).

3.7.3 Protocolos de Comunicação

Os Protocolos de Comunicação ditam as regras sobre os indivíduos que podem contribuir com conhecimentos para o Espaço de Crenças (Função de Aceitação) e como o Espaço de Crenças vai influenciar novos indivíduos (Função de Influência).

3.7.3.1 Função de Aceitação

Na Função de Aceitação são selecionados os indivíduos que irão influenciar o Espaço de Crenças atual. A Função de Aceitação pode ser de dois tipos: estática ou dinâmica. Na estática pode-se utilizar o ranking absoluto (uma porcentagem da população é selecionada) ou do ranking relativo (os indivíduos com aptidão acima da média são selecionados). Já na dinâmica o percentual dos indivíduos selecionados varia ao longo do processo evolutivo. Inicialmente o processo é menos seletivo, e se torna mais restrito ao longo da evolução. Para Reynolds e Zhu (2001) a Função de Aceitação é a que mais influencia o desempenho de um Algoritmo Cultural.

3.7.3.2 Função de Influência

Na Função de Influência é que se estabelece como o conhecimento armazenado no Espaço de Crenças vai interferir nos operadores do Espaço Populacional. Geralmente é utilizada uma *Função de Influência* para cada tipo de conhecimento armazenado.

A *Função de Influência* pode ser vista como um mecanismo de auto-adaptação do processo evolutivo, já que ela adapta os operadores de acordo com o conhecimento adquirido. Geralmente é a Função de Influência que determina a direção e o tamanho das modificações impostas aos novos indivíduos.

3.8 Trabalhos Relacionados

Nos trabalhos de Reynolds e Chung (1997) diferentes formas de conhecimento foram examinadas para que pudessem ser aplicadas no processo de auto-adaptação no nível populacional para funções de otimização. Uma função de aceitação utilizando sistema de inferência nebuloso foi empregada para selecionar os indivíduos aceitos para a atualização dos conhecimentos no Espaço de Crenças. Para implementar o Espaço Populacional foi utilizada a técnica de Programação Evolutiva. De acordo com os resultados obtidos o framework Cultural produziu melhorias substanciais de desempenho em termos da qualidade das soluções e do tempo computacional despendido na obtenção das mesmas em problemas de minimização sem restrições.

No trabalho de Jin e Reynolds (1999a) foi definida uma região n-dimensional, chamada célula de crença, que pode fornecer um mecanismo explícito capaz de suportar aquisição, armazenamento e integração de conhecimento sobre as restrições. No Algoritmo Cultural, o espaço de crença pode conter um conjunto de esquemas, cada um deles pode ser usado para guiar a busca da evolução da população. Esse tipo de região baseada em esquemas pode ser usada para guiar a busca otimizada para punir as regiões infactíveis e promover as regiões promissoras. Esse modelo foi comparado com quatro configurações de Algoritmos Culturais que manipulam os mesmos esquemas de problemas.

Algoritmos culturais também têm sido utilizados em engenharia e gestão (DASGUPTA, MICHAALEWICZ, 1997; OSTROWXKI, SCHLEIS, REYNOLDS, 2003) e em áreas tão diversas como a sistemas de decisão baseados em regras (STERNBERG; REYNOLDS, 1997). De acordo com Sternberg e Reynolds (1997) um Algoritmo Cultural foi inserido em um Sistema Especialista de detecção de fraudes, dado que a re-engenharia desse tipo de sistema, que é dinâmico, torna-se muito complexa. Por essa razão foi que se aplicou um Algoritmo Cultural, pois ele provê a capacidade de auto-adaptação em sistemas dinâmicos. Para representar um ambiente de desempenho dinâmico, foram usados quatro objetivos de aplicação diferentes. Os objetivos foram caracterização de reivindicações fraudulentas, reivindicações não fraudulentas, reivindicações não fraudulentas ditas anteriormente como fraudulentas e reivindicações fraudulentas ditas anteriormente como não

fraudulentas. Os resultados indicam que o Sistema Especialista Aculturado pode produzir informações necessárias para responder ao ambiente de desenvolvimento dinâmico. É possível também implementar uma comunicação direta entre o Algoritmo Cultural e o Sistema Especialista e fornecer uma resposta automatizada para mudanças ambientais.

Um dos exemplos de uso em ambientes comerciais tem relação com estratégias de modelagem em evolução de preços para as vendas de veículos (OSTROWSKI et al., 2002). Quatro tipos de agentes estavam presentes: de consumo, veículos, fabricantes e distribuidores. O modelo mostrou que estratégias ótimas foram gerados mais rapidamente usando Algoritmos Culturais do que Algoritmos Genéticos.

Os ACs também têm sido utilizados em reengenharia. Na reengenharia, quando feita manualmente, se torna muito trabalhosa. O uso de ACs provou que esta tarefa pode ser automatizada (OSTROWSKI et al., 2003). Este trabalho foi feito usando o Sistema de Gestão Direta do Trabalho (*Direct Labour Management System* - DLMS), usado pela Ford para gerenciar a montagem de veículos.

Outro exemplo da utilização de algoritmos culturais é para a aprendizagem para usuários. Um caso concreto foi a sua utilização para aprendizagem o uso correto dos cintos de segurança para crianças (KOBTI et al., 2006). Neste caso um conhecimento correto, obtido a partir de estudos, é mantido juntamente com o conhecimento no espaço de crença e uma distância entre ambos é calculada. Um experimento interessante foi feito, neste caso, com a adição de um componente de rede social baseada no parentesco e vizinhança local. A pesquisa mostrou que a aprendizagem de um especialista é melhor porque a rede social fez todo o sistema mais resistente a mudanças.

Uma variedade de aplicações têm aproveitado algoritmos culturais além das principais categorias que acabamos de discutir. Entre eles, estão os jogos, desde jogos de tabuleiro (OCHOA et al, 2007; OCHOA et al., 2008;) até jogos de corridas (KINNAIRD-HEETHER; REYNOLDS, 2009). No entanto, a otimização em todas as suas formas, tem sido certamente uma área frutífera, principalmente problemas que envolvem restrições, bastante utilizados em computação evolutiva e também explorados por Algoritmos Culturais (BECERRA; COELLO, 2004; REYNOLDS et. Al., 1995), otimização multiobjetivo (COELLO; BECERRA, 2003) ou mesmo de lógica *fuzzy* (ALAMI et al., 2007). Finalmente, a aprendizagem tem sido obviamente também uma área de trabalho para os algoritmos culturais (CURRAN, 2006).

4 Hibridização de Algoritmos Evolutivos com Busca Local: "Algoritmos Meméticos".

4.1 Introdução

Métodos de optimização iniciais como método de Newton, método simplex, o algoritmo de gradiente conjugado, e outros semelhantes, têm sido desenvolvidos em problemas com certas características matemáticas. No entanto, como muitos dos problemas do mundo real de otimização são problemas de caixa-preta no qual um conhecimento do problema não está disponível previamente, o uso de metaheurísticas passou a prevalecer (NGUYEN, 2006).

Metaheurísticas são geralmente algoritmos baseados em população que exploram o espaço de busca estocasticamente de acordo com algumas heurísticas. Como eles não são algoritmos para problemas específicos, eles têm uma boa chance na otimização desses problemas do tipo caixa-preta. Algoritmos Evolutivos, otimizações por enxame de partículas, algoritmos de colônia de formigas, e afins, são alguns dos renomados tipos de metaheurísticas que têm sido amplamente adotados (LIN; CHEN, 2011).

De acordo com Nguyen (2006) pode-se classificar os métodos de metaheurística em dois grupos: o grupo de base populacional e o grupo de métodos simples baseados em único ponto. Os métodos baseados em população avaliam e desenvolvem um conjunto de soluções (também chamado de população) após cada iteração do processo evolutivo de busca, enquanto os de um único ponto, apenas uma solução a cada iteração é avaliada.

Alguns exemplos de metaheurísticas com base populacional são: Algoritmos Genéticos, Estratégias Evolutivas, Programação Evolutiva, Optimização por Enxame de Partículas, entre outros. Alguns exemplos baseados em um único ponto são Busca Tabu (GLOVER, 1989), Pesquisa de Vizinhança Variável (HANSEN; MLADENOVIC, 1998) e Basins Hopping também chamado de *Iterated Local Search* (WALES; DOYE, 1997).

Durante as duas primeiras décadas de pesquisa em metaheurísticas, diferentes comunidades de pesquisa trabalharam em diferentes técnicas de metaheurísticas sem muita interação entre as técnicas e sem interação entre as comunidades de pesquisa. Ao longo dos últimos anos foi relatado um grande número de algoritmos que não seguem o paradigma de uma única metaheurística tradicional. Pelo contrário, eles combinam vários componentes de algoritmos, muitas vezes proveniente de outras áreas de pesquisa sobre otimização. Estas abordagens são comumente referidas como metaheurísticas híbridas. A falta de uma definição

precisa deste termo por algumas vezes tem sido alvo de críticas. A principal motivação por trás da hibridação de diferentes algoritmos é exploração do caráter complementar de diferentes estratégias de otimização. Portanto, a escolha de uma combinação adequada de conceitos algorítmicos complementares pode ser a chave para alcançar o melhor desempenho na resolução de muitos problemas de otimização difíceis. Infelizmente, o desenvolvimento de uma abordagem híbrida eficaz é em geral uma tarefa difícil, que exige conhecimentos de diferentes áreas de otimização. Além disso, a literatura mostra que não é trivial generalizar, ou seja, um algoritmo híbrido pode funcionar bem para problemas específicos, mas pode executar mal para os outros. No entanto, existem vários tipos de hibridação que têm demonstrado ser bem sucedidos para muitas aplicações e podem servir como orientação para novos desenvolvimentos (BLUM et al., 2011).

Quando problemas complexos são tratados, sem levar vantagens de informações especificas do problema (informações sobre o problema previamnete informado) ou de informações recuperadas durante o processo de otimização, as metaheurísticas podem simplesmente oferecer um desempenho medíocre. Já as heurísticas levam vantagens por se aproveitarem de informações específicas de um problema em questão. As técnicas que utilizam um modo híbrido entre metaheurísticas e heurísticas de problemas específicos têm sido criadas para fornecer técnicas de otimização mais eficientes para os problemas mais complexos. Técnicas que empregam metaheurísticas como busca global e heurísticas específicas do problema como a busca local, são comumente referidos como Algoritmos Meméticos (AMs) (GALLARDO et al, 2007; BLUM et al., 2011). Com o desenvolvimento de um projeto adequado, os AMs não só podem apresentar uma boa capacidade exploratória, similar ao que faz um algoritmo com base populacional de busca global, mas também proporciona um bom desempenho de exploração durante a busca, semelhante ao que faz um algoritmo de busca local. Dependendo do tipo de problema abordado, os algoritmos meméticos podem apresentar um desempenho melhor do que algoritmos sem hibridização, (LIN; CHEN, 2011).

Os métodos de metaheurísticas são propostos para busca em um nível global e hibridizá-los com busca local passa a ser uma boa ideia, dado que são métodos especializados em buscas em um nível local para encontrar ótimos locais (NGUYEN, 2006).

Este tipo de hibridização tem muitas vantagens. Primeiro, a busca local ajuda a melhorar a qualidade das soluções encontradas por metaheurísticas (IBARAKI 1997). Em segundo lugar, quando combinado com abordagem baseada em população, a busca local pode ajudar a manter uma população mais diversificada e, portanto, oferecer mais chances de

encontrar o ótimo global (HART, 1994).

A busca local é também a maneira intrínseca de muitos métodos baseados em trajetória (BLUM; ROLI, 2003). A busca local ajuda a reduzir o espaço de busca de todos os estados disponíveis a um subconjunto contendo apenas valores ótimos locais em que a transição de um ótimo para os outros pode ser feita por metaheurísticas de alto nível. Esta vantagem abre os caminhos para combinar diferentes metaheurísticas com outras. Finalmente, é relatado que hibridar busca local com metaheurística de algoritmos é a melhor opção para resolver muitos problemas do mundo real de otimização contínua, envolvendo cristais, agrupamentos e biomoléculas (WALES, SCHERAGA, 1999; NGUYEN, 2006).

Existe uma grande quantidade de fatores que motivam a hibridização e Algoritmos Evolutivos (AE) com outras técnicas. Entretanto, aqui a discussão é sobre as que mais se destacam.

4.2 Benefícios da Busca Local

Os benefícios de desempenho que podem ser alcançados por hibridação de Algoritmos Evolutivos (AEs) com Busca Local (BL), assim também chamados de Algoritmos Meméticos (AMs), já estão bem documentados através de uma vasta gama de domínios de problemas. Com o aumento do interesse em pesquisar algoritmos meméticos, a computação memética tem evoluído bastante, partindo sempre do princípio da hibridação de busca global com busca local (MEUTH et al., 2009). O sucesso de algoritmos meméticos em vários domínios de aplicação, variando de problemas combinatórios NP-difíceis a problemas de programação não-linear, tem sido relatado (ONG; LIM; CHEN, 2010). Outras aplicações recentes incluindo algoritmo de controle cartesiano de robô (NERI; MININNO, 2010), sistemas de ensino eletrônico (ACAMPORA; GAETA; LOIA, 2010) segmentação de imagens, (JIAO et al., 2010), a seleção de recursos (ZHU, JIA, JI, 2010), gestão de missão (MEUTH et al., 2010), e seleção de *portifólios* (RUIZ-TORRUBIANO; SUAREZ, 2010), também demonstram a eficácia de algoritmos meméticos aplicados em diferente domínios. Outros trabalhos também incluem otimização combinatória, não estacionários e problemas multiobjetivos (KRASNOGOR; SMITH, 2005; SMITH, 2007; LIN, CHEN, 2011).

A capacidade da busca global da parte evolutiva de um algoritmo memético cuida da exploração, tentando identificar as regiões mais promissoras do espaço de busca, a parte de busca local examina os arredores de uma solução inicial, explorando-a. Esta ideia não é só atraente como também tem sido muito bem sucedida na prática. De fato, para a maioria dos problemas de otimização combinatória e, como também em muitos problemas de otimização

contínua, esta combinação produz bons resultados em algoritmos de otimização.O papel da busca local é fundamental e a seleção de suas regras de busca juntamente com sua harmonização com o esquema da busca global produz o sucesso dos algoritmos meméticos (NERI; COTTA; MOSCATO, 2011).

Youngsu (2006) descreve a necessidade da busca local, levando em consideração o exemplo de um AG e sua relação com o processo de convergência para uma solução ótima global ou local. Estas duas situações podem ser resumidas nas Figuras 12 e 13.

Quando o AG está convergindo para uma solução ótima global como a Figura 12, a sua solução é continuamente melhorada. Cada quadrado da figura representa uma solução ou a aptidão conforme as suas respectivas características presente em cada cromossomo. Nesse exemplo (Figura 12) diz-se que as soluções conseguiram escapar de ótimos locais.

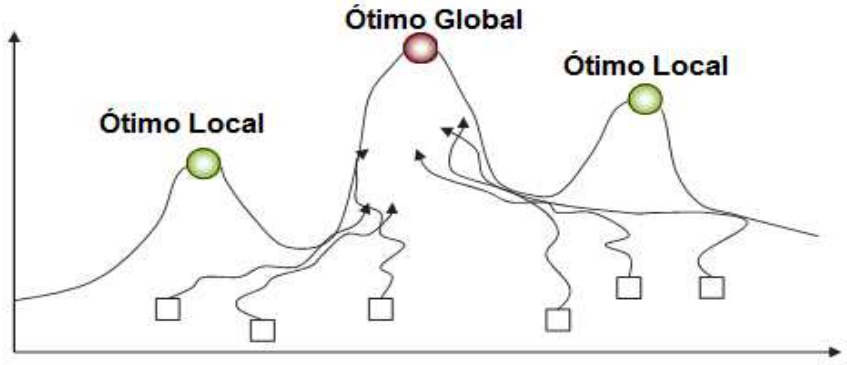


Figura 12: Situação que um AG está convergendo para um ótimo global (YOUNGSU ,2006).

No entanto, quando o AG não está convergindo para uma solução ótima global, o desempenho do AG é deteriorado, conforme a representação da Figura 13.

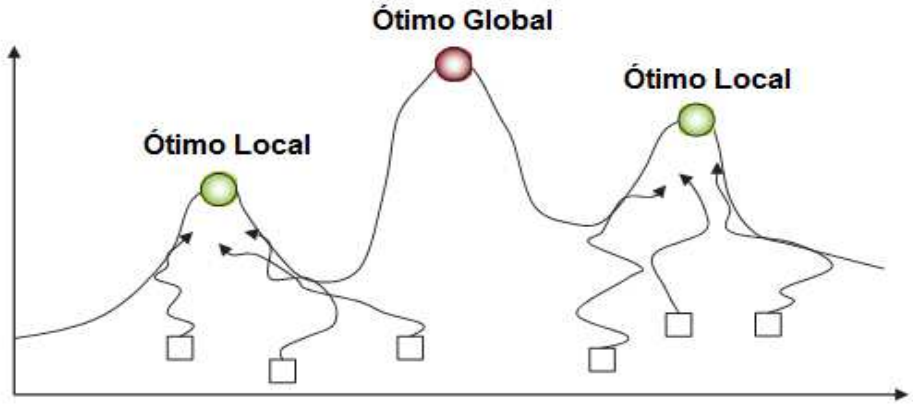


Figura 13: Situação que um AG não está convergendo para um ótimo global (YOUNGSU ,2006).

Se esta situação prossegue continuamente, pode ser difícil para o AG evitar a

convergência prematura para uma solução ótima local. A técnica que ajuda a melhorar esta situação é a inserção de novos indivíduos com bons valores de *fitness* no atual ciclo (*loop*) do AG.

O uso de busca local tem sido cada vez mais reconhecido como um operador importante e de grande impacto sobre a eficácia da hibridação. A escolha do operador de movimento, o qual define a função de vizinhança é de grande importância e, assim, regula a maneira em que as novas soluções são geradas e testadas. Os pontos que são localmente ótimos em relação a uma estrutura de vizinhança podem não ser em relação à outra (a menos, que eles sejam globalmente melhor). Portanto, mesmo que ocorra uma convergência prematura em uma população, passando a ter apenas ótimos locais, as mudanças feitas pela busca local através dos movimentos realizados nos pontos vizinhos, podem fornecer um importante progresso além da operação de recombinação e mutação. Esta observação levou uma série de autores a investigar e propor mecanismos para a escolha de uma busca local entre um conjunto de buscas locais predefinidos que pode ser utilizado durante uma particular execução de uma metaheurísticas tais como os algoritmos evolutivos (SMITH, 2007).

4.3 Combinação de Metaheurísticas com Busca Local na Melhoria de Soluções

Acredita-se que muitos métodos de metaheurísticas têm uma convergência lenta para o valor ótimo na otimização de problemas contínuos, devido ao fato de que eles confiam muito em movimentos estocásticos ao invés de informação local (OSMAN; KELLY, 1996). Como resultado, muitas tentativas têm sido feitas para fornecer uma convergência mais rápida com soluções mais precisas. Geralmente, existem duas técnicas que podem ser usados em combinação com metaheurística para melhorar a precisão de soluções: usando os operadores de busca especializados e usando buscas locais reais.

4.3.1 Operadores de Busca Especializados

A primeira técnica, que normalmente é utilizada em algoritmos evolutivos (AEs), refere-se a projetar operadores especiais evolutivos que são muito bons em exploração local. Com esta técnica, o processo de busca local pode ser alcançado pela ligação simultânea dos diferentes pontos de amostragem dentro de uma região considerada promissora e, em seguida, utilizando as interações evolutivas (por exemplo, recombinação) para dirigir a busca para o ótimo local. No entanto, os algoritmos que utilizam esta técnica não devem ser considerados como algoritmos híbridos baseados em busca local. Isso porque eles são hibridizados com operadores especializados, em vez de buscas locais reais independentes (NGUYEN, 2006).

4.3.2 Busca Local Real

A segunda técnica consiste em aplicar amplamente o método de busca local para refinar as soluções encontradas por uma metaheurística. Nos métodos que seguem esta técnica, o algoritmo de metaheurística ainda desempenha o papel principal de busca, enquanto a busca local é utilizada simplesmente para melhorar as soluções encontradas.

4.4 Implementação de Algoritmos Meméticos

Na implementação de um Algoritmo Memético (AM) não é exigido apenas um mecanismo de busca global juntamente com os operadores de pesquisa locais, mas também se deve estabelecer uma coordenação sutil para expor o ponto de vista de ambos.

Ao utilizar a heurística de busca local, é assumido que a busca evolutiva fornece uma exploração ampla do espaço de busca, enquanto a busca local pode desenvolver um tipo de exploração com base em soluções promissoras. Estes métodos permitem a pesquisa local para refinar uma solução de acordo com a vizinhança que esta possui. Em outras palavras, dada uma solução particular *s*, um método de busca local explora a vizinhança de *s*, a fim de encontrar uma solução *s'* melhor do que uma solução *s*. Então, deve-se definir de que forma *s'* substitui *s*. Esta substituição pode ser realizada seguindo a abordagem de Lamarck ou Baldwin (EIBEN; SMITH, 2003).

Em geral, os Algoritmos Meméticos (AMs) são considerados Lamarkianos se o resultado produzido pela busca local substitui o indivíduo selecionado na população (genótipo e aptidão). Essa abordagem pressupõe que os indivíduos podem passar geneticamente características de aprendizagem. Os AMs são considerados Baldwinianos se o genótipo original do indíviduo selecionado for mantido. Ou seja, as melhorias ou aprendizagem realizadas na busca local não podem alterar diretamente o genótipo do indivíduo, exceto sua aptidão (fitness). Nesse caso o cromossomo selecionado será associado à aptidão da busca local. Essa abordagem incrementa a diversidade, levando-se em consideração que no processo de seleção o indivíduo que tiver aptidão inferior terá chances de ser selecionado, desde que sua aptidão proveniente da busca local seja melhor (KU; MAK, 1998).

Os Algoritmos Meméticos levantam uma série de questões importantes que devem ser abordadas pelo desenvolvedor (KRASNOGOR; SMITH, 2005). Talvez a mais importante destas questões possa ser enunciada como: "Qual é o melhor equilíbrio entre a busca local e busca global no processo evolutivo?" Algumas questões foram levantadas por Hart (1994) que, em seu trabalho sobre problemas de otimização contínua, fez alguns questionamentos sobre a concepção de algoritmos meméticos eficientes:

- Qual a frequência da aplicação da busca local?
- Quais indivíduos devem ser utilizados na busca local?
- Em quanto tempo deve ser executada a busca local?
- Qual a quantidade de eficiência que a busca local precisa ter?

Todos esses detalhes serão discutidos logo a seguir.

4.4.1 Frequência da Aplicação da Busca Local

Para efeito de avaliação Hart (1994) utiliza um AG padrão híbrido com Busca Local (AG-BL), onde aplica busca local a cada indivíduo da população do AG. Essa configuração faz pleno uso das informações potenciais fornecidas pela busca local e aumenta o custo computacional produzindo algumas restrições sobre o uso da hibridização como o caso do AG-BL. Um exemplo de restrição é a utilização de populações com tamanho pequeno. Além disso, a aplicação de busca local para cada indivíduo não melhora necessariamente a eficiência da busca do AG-BL. Isso por que buscas locais podem ser realizadas em soluções que são claramente distante do ótimo global. Hart (1994) propõe métodos para que a freqüência de busca local seja adaptada automaticamente, e descreve como a freqüência da busca local está relacionada com o tipo de problema a ser otimizado.

Hart (1994) investigou o impacto da frequência de busca local para a otimização de funções comuns de teste em espaços contínuos como a função Rastrigin, a função Griewank, e suas variantes. Seus resultados experimentais sugerem que os algoritmos genéticos (AGs) com grandes populações são mais eficazes quando a busca local é utilizada com pouca frequência. Ele também afirma que uma grande frequência de busca local é necessária se o algoritmo não é capaz de identificar as regiões que contêm ótimos globais.

A maioria dos AMs na literatura aplica busca local para cada indivíduo em cada geração do algoritmo evolutivo (KRASNOGOR; SMITH, 2005). Alguns autores (HART, 1994; LAND, 1998) exploram estas questões e sugerem vários mecanismos pelos quais os indivíduos são escolhidos para serem otimizados pela busca local, a intensidade dessa busca local e a probabilidade de realizá-la. Isto é conseguido através de sofisticados escalonadores que medem estatisticamente a população para determinar o momento da aplicação de busca local.

4.4.2 Escolha dos Indivíduos para Aplicar a Busca Local

Se a busca local não é aplicada a cada indivíduo em uma população, então é

preciso decidir a forma como os indivíduos são selecionados para a busca local. De acordo com Hart (1994) o método mais simples para selecionar indivíduos é o aleatório uniforme. Entretanto, é possível utilizar métodos mais sofisticados que usam informações da população para selecionar melhor os indivíduos. Estes métodos reduzem a probabilidade de um indivíduo ser utilizado para a busca local se sua solução é semelhante a outros indivíduos na população. Hart (1994) utiliza uma métrica de similaridade entre indivíduos que mede a distância entre os genótipos para evitar redundância na busca local das mesmas soluções ou soluções similares.

4.4.3 Tempo de Execução da Busca Local

Um AG híbrido com uma busca local de longa duração irá executar menos gerações do AG do que um AG híbrido com uma busca local de menor duração, se ambos terminarem com o mesmo número de avaliações de função. Em problemas combinatórios, uma busca local pode ser realizada até que a convergência da solução alcance um ótimo local. No entanto, em domínios contínuos, a busca local é normalmente truncada antes de atingir um ótimo local, quando o tamanho do passo torna-se muito pequeno. Realizar uma busca local até que ocorra a convergência da solução para um ótimo local, que é referida como uma busca local completa, pode levar à perda da diversidade da população, dependendo da estratégia de aprendizagem utilizada. Algoritmos genéticos híbridos que adotam a abordagem pura lamarckiana são mais propensos à perda de diversidade do que outros que utilizam outras técnicas de aprendizagem. A aplicação de uma busca local completa aumenta o custo computacional pela quantidade de avaliações realizada na função de aptidão. No entanto, há classes de problemas, que permitem a decomposição da aptidão, onde o cálculo da aptidão de uma solução computacionalmente se torna menor do que computar a sua aptidão a partir do zero (EL-MIHOUB et al., 2006).

A área de atração de um mínimo local é o conjunto de soluções a partir da qual a busca local irá convergir para aquele mínimo local. Ao usar a busca local na área de atração do ótimo global, fica claro que será realizada uma busca local completa para o ótimo global. No entanto, ao realizar busca local em outras áreas de atração, a busca local completa pode não ser necessária. Se a hibridização utilizada não requer soluções refinadas para discriminar entre duas regiões do espaço de busca, então a minimização completa pode não ser necessária (HART, 1994).

O conceito de áreas de atração de um ótimo local é interessante, pois expressa o conjunto de pontos no espaço de busca de tal forma que um processo de busca local começa a

partir de qualquer membro dentro de uma área e acabará por encontrar o melhor local nesta mesma área. Nesta linha de pesquisa, o espaço de busca é uma união das áreas de atração. Nos trabalhos de Sinha, Chen, Goldberg (2004) e Nguyen, Ong, Lim (2009) esse conceito é adotado para estimar o tamanho ideal da busca local. Nesses artigos, áreas de atração no espaço de busca são classificadas em dois tipos, em que as soluções de destino podem ser alcançadas e as que não podem ser alcançadas. Nesse contexto, o tamanho ideal de busca local é estimado através da probabilidade de acertos das áreas de atração anteriores (LIN; CHEN, 2011).

Existem poucos estudos na investigação da duração ideal da busca local. Entretanto Hart (1994) concluiu que o uso de um curto período de busca local produziu melhores resultados para as funções de Griewank, enquanto uma longa duração produziu melhores resultados para as funções de Rastrigin. Rosin et al. (1997) experimentou várias buscas locais com curta e longa duração em um sistema híbrido para otimizar a configuração de um complexo de drogas.

Em ambas as durações encontradas foram obtidas um desempenho similar. Em Hart et al. (2000) concluiu-se que a duração da busca local é um fator importante e algoritmos genéticos híbridos com longas pesquisas locais serão mais eficazes para problemas não triviais. Muitas vezes o tempo de execução de uma busca local é limitado a um valor chamado profundidade de busca local.

4.4.4 Quantidade de Eficiência que a Busca Local Precisa Ter

Muitas vezes, é possível ter vários métodos de busca local disponíveis para um domínio de pesquisa específico. Por exemplo, para otimização de funções suaves em R^n, numerosos métodos locais têm sido propostos utilizando a informação da derivada para realizar a busca.

Um deles é frequentemente confrontado com uma escolha entre dois ou mais métodos de busca local com eficiência diferente, de modo que os algoritmos de buscas locais mais eficientes possuem um maior custo para executar. Quando se seleciona um método de busca local para um modelo híbrido, o custo da busca local e sua eficiência são os dois fatores que podem afetar a eficiência híbrida global.

No trabalho de Lin e Chen (2011) a eficiência do *framework* sobre algoritmos meméticos implementados é alcançada, aplicando-se elitismo em populações de tamanho grande e busca local pouco frequente. Também são propostas duas estratégias, a seleção baseada em aptidão e seleção baseada em diversidade, produzindo soluções candidatas para

aplicar a busca local. O trabalho concluiu que estas duas estratégias ajudam muito.

4.5 Busca Local em um Algoritmo Memético

Inicialmente é importante descrever o algoritmo de busca local na sua forma padrão. Krasnogor e Smith (2005) descrevem um algoritmo padrão, reforçando que este algoritmo está implícito em muitos Algoritmos Meméticos (AMs), conforme o algoritmo *BuscaLocalPadrão*(...) apresentado abaixo:

```
Algoritmo BuscaLocalPadrao (x):
01:INÍCIO
02      Produzir uma solução inicial s para uma instância do problema x;
03:     REPITA ATÉ (ótimo local) FAÇA
04:             usando s e x gerar a vizinhança Nx,s;
05:             SE (Nx,s é melhor que s) ENTÃO
06:                 s:= Nx,s;  /*pode-se atualizar toda a solução s ou não*/
07:     FIMREPITA   /*se  s não melhora então condição de saída*/
08:FIM
```

O algoritmo *BuscaLocalPadrao* (...) capta a noção intuitiva de busca em uma vizinhança como um meio de identificar a melhor solução. Ele não especifica políticas de desempate, estrutura de vizinhança, etc. Este algoritmo utiliza um método "guloso" em vez de uma política "íngreme" isto é, aceita o primeiro vizinho melhor que encontrar.

Em geral, uma determinada solução pode ter vários vizinhos melhores, e a regra que atribui um dos melhores vizinhos para uma solução é chamada de regra pivô. A seleção da regra pivô (ou regras) no algoritmo de busca padrão local tem enorme impacto sobre a complexidade da busca e potencialmente na qualidade das soluções exploradas.

Note também que o algoritmo acima implica que a pesquisa local continua até que um ótimo local seja encontrado. Isto pode levar muito tempo, e em domínios contínuos a otimalidade pode ser não trivial.

Levando-se em consideração essa característica, na literatura muitos algoritmos de buscas locais embutidos nos AMs não são padrões, isto é, eles costumam fazer uma busca local mais curta (truncada) (KRASNOGOR; SMITH, 2005).

Eiben e Smith (2003) propõem um modelo de algoritmo para busca local, conforme o algoritmo *BuscaLocal*(...) mostrado a seguir.

***Algoritmo* BuscaLocal (*i*):**

```
01:INÍCIO
02      Dada uma solução inicial i e uma função que busca n vizinhos
03:     melhor := i;
04:     interações: = 0;
05:     REPITA ATÉ <profundidade da busca local)
06:           contador: = 0;
07:           REPITA ATÉ (regra pivô for satisfeita)
08:             Geração do vizinho mais próximo j ∈ n(i);
09:             contador: = contador + 1 ;
10:             SE (f(j) é melhor do que f(melhor)) ENTÃO
11:                 melhor: = j;
12:           FIMREPITA
13:           i: = melhor;
14:          interacões = interações + 1;
15:     FIMREPITA
16:FIM
```

O algoritmo *BuscaLocal(..)* possui três componentes principais (EIBEN, SMITH, 2003; SMITH, 2007):

- O primeiro é a escolha da regra pivô, que pode ser a "subida acentuada" ou a "subida gulosa". No primeiro caso, a condição de encerramento é que toda a vizinhança *n(i)* seja pesquisada (*contador*=|*n(i)*|), enquanto que o último deve parar assim que uma melhoria for encontrada, isto é, a condição de terminação é: (*contador*=|*n(i)*|) ∨ (*melhor*≠*i*). É importante ressaltar que alguns autores acabam considerando apenas uma amostra aleatória de busca de tamanho $N<|n(i)|$ se a busca na vizinhança é muito grande.
- O segundo componente é a "profundidade" da busca local, ou seja, a condição sobre o número de iterações a serem executadas. Essa condição situa-se entre apenas um passo, melhorando em um passo (*iterações= 1*) e a busca pela otimalidade local: (*contador*=|*n(i)*|) ∧ (*melhor*=*i*). Uma considerável atenção tem sido dada no efeito de mudar estes parâmetros dentro AMs, como por exemplo, em Hart (1994). A escolha de regras de pivô também tem demonstrado um efeito no desempenho do algoritmo de busca local, tanto em termos de tempo necessário como na qualidade de solução encontrada.

- O terceiro fator primário e que afeta o comportamento da busca local é a escolha da função geradora de busca de soluções vizinhas. Isto pode ser pensado como a definição de um conjunto de pontos *n(i)* que podem ser alcançado através da aplicação de alguns operadores de movimentos para o ponto *i*. Uma representação como um grafo *G = (v, a)*, onde o conjunto de vértices *v* são os pontos no espaço de busca, e as arestas *a* estão relacionadas com as o operador de movimento, ou seja, $a_{i,j} \in G \Leftrightarrow j \in n(i)$. A previsão de um valor escalar *f* de aptidão, que é definida sobre o espaço de busca, significa que pode considerar os grafos definidos pelos diferentes operadores de movimentos como "cenário de aptidão". Merz e Freisleben (1999) apresentam medidas estatísticas que podem ser usadas para caracterizar cenários de aptidão propostas como potenciais medidas para dificuldade de problemas. Eles mostram que a escolha do operador de movimento pode ter um efeito substancial sobre a eficiência e eficácia da busca local e, portanto, no resultado do AM.

Em alguns casos, a informação sobre o domínio específico pode ser utilizada para guiar a escolha da estrutura de vizinhança dentro dos algoritmos de busca local. No entanto, recentemente, tem sido mostrado que a escolha ótima de operadores não pode ser apenas baseada em uma classe de problemas específicos, mas também dependente de todo o estado da busca evolutiva. Este resultado não é surpreendente quando se considera que os pontos que são localmente ótimos em relação a uma estrutura de vizinhança não podem ser em relação à outra, a não ser é claro que eles sejam globalmente melhor. Assim, se um conjunto de pontos convergiu para o estado em que todos são localmente ótimos em relação ao operador busca de vizinhança corrente, a alteração do operador de busca de vizinhança pode fornecer um meio de progressão, além das fornecidas pelos operadores de recombinação e mutação. Esta observação tem sido aplicada também em outros campos da otimização e forma o coração de métodos como os algoritmos de busca de vizinhança variável (EIBEN, SMITH, 2003).

4.5.1 Posicionamento da Busca Local

Existem diversas maneiras em que um Algoritmo Evolutivo (AE) pode ser usado em conjunto com outros operadores de busca local tal como ilustrado pela Figura 14, que mostra a estrutura genérica de um AM na estrutura básica de um AG. Nesta estrutura, foram identificados e marcados a busca local. Cada um dos lugares marcados oferece uma

oportunidade para incorporar métodos de otimização. Por exemplo, a população inicial pode conter soluções heurísticas projetadas especificamente para o problema a ser resolvido.

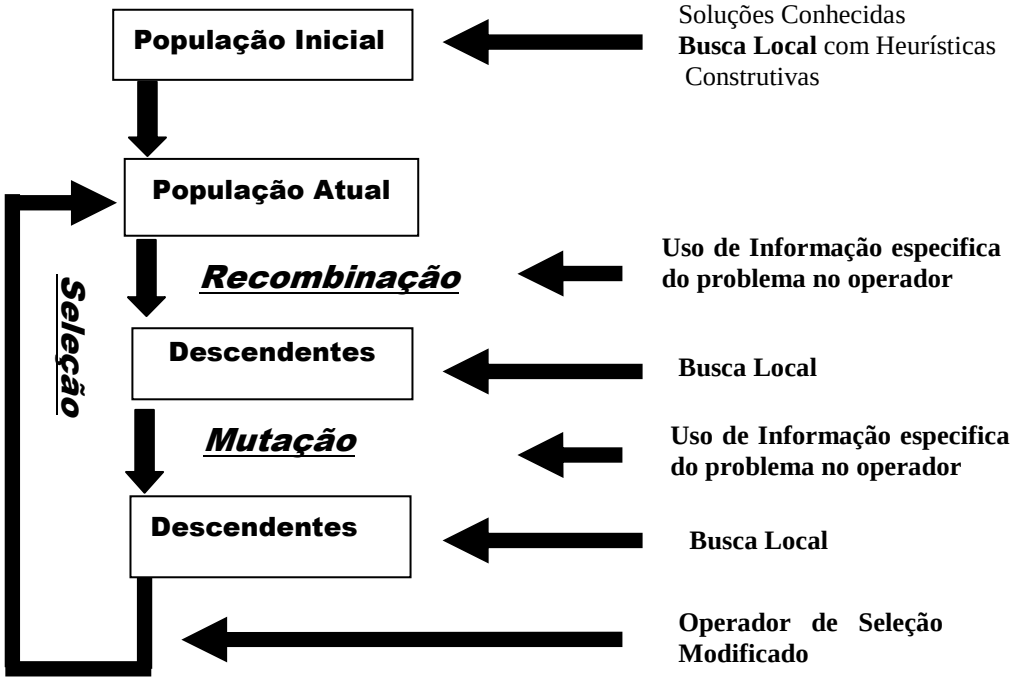


Figura 14: Posicionamento da Busca Local

Os métodos de buscas locais podem ser aplicados a um ou a todos os conjuntos de soluções intermediárias. Por exemplo, poderia ser aplicado ao conjunto de soluções obtidas após recombinação, o conjunto de soluções obtidas após a mutação, ou em ambos os conjuntos. No entanto, a forma mais aceitável de integração é aplicar uma ou mais fases de optimização local, com base em um parâmetro de probabilidade, os membros individuais da população em cada geração (KRASNOGOR; ARAGON; PACHECO, 2006).

4.5.2 Busca Local Dentro do Ciclo Evolutivo

Alguns pesquisadores (IBARAKI, 1997) consideram que quando o operador de Busca Local (BL) é aplicado antes do cruzamento e mutação, o AM é *lamarckiano*, e quando o operador de BL é aplicado após o cruzamento e mutação o AM é *darwiniano* puro. Esta é uma interpretação errônea da versão *Lamarckiana* contra *Baldwiniana*. Em ambos os casos a busca local é usada para melhorar (se possível) a aptidão da solução candidata, alterando assim as probabilidades de seleção.

A diferença é, simplesmente, que no caso de lamarckiana a aprendizagem e as modificações são também assimiladas pelo indivíduo - por outras palavras, o indivíduo

vizinho substitui o candidato da solução original. A aprendizagem Lamarckiana em AMs pode ocorrer antes ou após a aplicação dos outros operadores genéticos. No entanto, não há muito sentido em aplicar busca Baldwiniana após a seleção dos pais, mas sim antes da recombinação e mutação, uma vez que a prole resultante terá de ser reavaliada de qualquer maneira. Na prática, muitos trabalhos tendem a usar uma abordagem lamarckiana (KRASNOGOR; SMITH, 2005).

4.6 Estratégias de Busca Local

As estratégias mais utilizadas de busca local são: *Hill-Climbing*, *Simulated Annealing* e *Tabu Search*.

4.6.1 Hill Climbing

Esta estratégia começa em um ponto gerado aleatoriamente. Em seguida, é realizada uma pesquisa em uma direção apropriada, de acordo com uma distância definida para cada passo de pesquisa, até que finalmente se chegar a um ponto ótimo. A direção e a distância do passo podem ser ajustadas de forma adaptativa. O uso de *Hill-Climbing* em domínios continuos não garante uma solução ótima, porque tende a encontrar um mínimo local e fazer várias visitas a outros pontos já descobertos.

Em situações com vários mínimos locais, o método *Hill Climbing* enfrenta o problema de ficar preso no ótimo local, como na Figura 15 onde o método *Hill Climbing* fica preso em um mínimo local e não é capaz de encontrar um vizinho com menor aptidão em um problema de minimização.

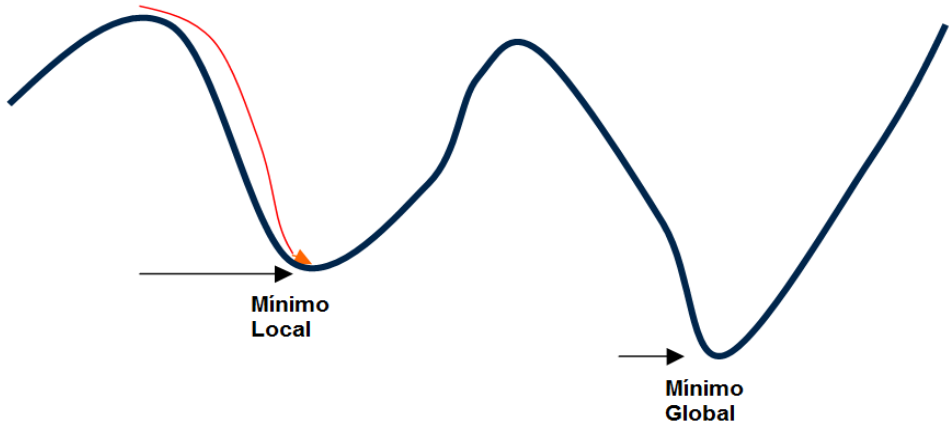


Figura 15: Processo de busca local preso no mínimo local (ADRA, 2003)

O algoritmo padrão *Hill-Climbing* é mostrado a seguir:

***Algoritmo* Hill-Climbing**

```
01:INICIO
02: n ← número de extensões a serem geradas
03: S ← solução inicial qualquer
04: REPITA ATÉ (que S seja a solução ideal ou o tempo tenha se esgotado) FAÇA
05:     S'← NovaSolução( S );
06:     REPITA ATÉ n - 1 vezes FAÇA
07:             W ← NovaSolução( S );
08:             SE (W  melhor que S') Então
09:                 S' ← W;
10:     FIMREPITA
11:     SE (S' melhor S) ENTÃO
12:         S ← s';
13: FIMREPITA
14:RETORNE S;
15:FIM
```

4.6.2 Simulated Annealing (SA)

Simulated annealing (SA) é assim chamado pela analogia com o processo de recozimento físico com sólidos, em que um sólido cristalino é aquecido e, em seguida, é resfriado lentamente, permitindo pequenos aquecimentos durante o resfriamento, até atingir a sua possível e mais regular configuração permitindo que as moléculas alcancem uma configuração de baixa temperatura e formem uma estrutura cristalina, livre de defeitos. O *Simulated Annealing* estabelece uma ligação entre esse tipo de comportamento termodinâmico e a busca de mínimos globais para um problema de otimização discreta. Dessa forma, o *Simulated Annealing* tornou-se uma estratégia de busca local probabilística, com base na termodinâmica (HENDERSON, JACOBSON, JOHNSON ,2003).

O SA pode ser visto como um algoritmo que "resfria" a solução lentamente para garantir que ela tenha a melhor função objetivo, ao mesmo tempo em que permite configurações que vão de encontro ao melhor valor da função objetivo encontrado (situação correspondente a pequenos aquecimentos). Tal abordagem pode resolver os problemas com a função de elevado grau de complexidade, mas não há maneira de saber quando se atinge uma solução ótima. Ao término do processo não há garantias de que o ótimo global foi encontrado. Quanto mais se permite o processamento do algoritmo, melhor será a solução. Uma característica importante da SA é a aceitação de configurações que têm temperatura mais elevada, que pode parecer pior, ou seja, permite a aceitação de uma configuração que proporciona um "pior" valor para a função objetivo, evitando assim a convergência para um local mínimo. Esta aceitação é determinada por um número aleatório sendo controlado por: P

= $e^{-\Delta/T}$. Assim, dada qualquer solução inicial, o procedimento gera aleatoriamente um vizinho único em cada iteração para a solução corrente. O algoritmo *SimulatedAnnealing*(...) é descrito a seguir:

```
Algoritmo SimulatedAnnealing( )
01:INÍCIO
02:    Pegar a solução S, a temperature inicial T e o fator de temperatura k
03:   melhor←s;
04:   REPITA ATÉ (que melhor solução ideal, ou o tempo tenha esgotado, ou T < 0)
05:     S'← GerarVizinho(S)  /*aqui  deve-se escolher um método de vizinhança*/
06:     Δcusto ← custo(S') - custo(S);
07:     SE (Δcusto < =0 ) ENTÃO
08:              S ← S';
09:     SENÃO
10:              S ← S' com probabilidade e^(-Δcusto/T);
11:     T ←  K*T;    //redução de T com uma nova temperatura
12:     SE (S< melhor) ENTÃO
13:              melhor←S;
14:   FIMREPITA
15: RETORNE S (melhor solução)
16:FIM
```

A Figura 16 demostra um comportamento esperado do SA levando em consideração a tentativa de escapar de um ótimo local considerando configuração inviáveis.

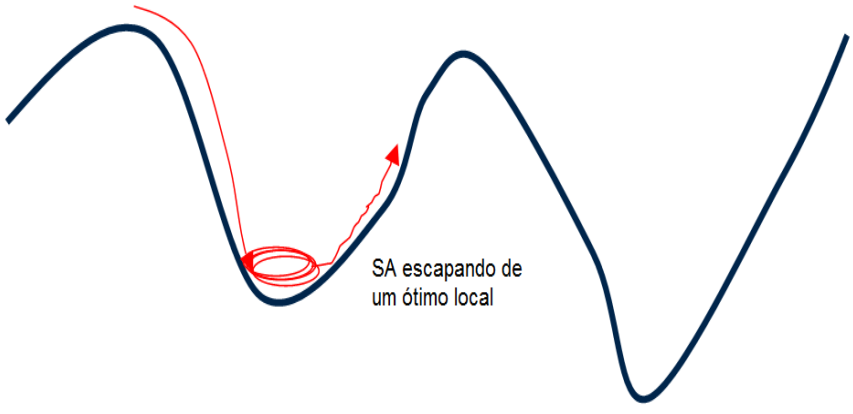


Figura 16: SA escapando de um ótimo local (ADRA, 2003).

Embora o *Simulated Annealing* seja relativamente simples de implementar, a atenção cuidadosa deve ser dada a dois importantes parâmetros que são a configuração do tamanho do passo para a perturbação e o valor da temperatura.

O fator de minimização da temperatura é representado por *K* e a temperatura inicial é representada por *T*. Muitos pesquisadores sugerem configurações de tamanho de passo e temperaturas *T* que permitem que cerca de 80% dos movimentos sejam aceitos, embora estes dois parâmetros sejam totalmente subjetivos para a natureza do domínio de aplicação (ADRA, 2003).

4.6.3 *Tabu Search*

A principal idéia por trás da *Tabu Search* é muito simples. Uma "memória" força a busca a explorar novas áreas do espaço de busca. Pode-se dessa forma memorizar algumas soluções que foram examinados recentemente e estas se tornam pontos Tabus (proibidos) a serem evitados na tomada de decisões sobre a seleção para a próxima solução. Note que busca tabu é basicamente determinista (ao contrário do SA), mas é possível adicionar alguns elementos probabilísticos para ele.

4.7 Algoritmos Meméticos em Problemas Contínuos de Otimização

A primeira categoria de metaheurísticas descrita neste trabalho é baseada em busca de soluções vizinhas ou movimento ponto-a-ponto. Uma característica destas metaheurísticas é que as soluções candidatas são criadas através da movimentação de uma solução para outra solução relacionada. A relação entre uma solução corrente e uma solução candidata é variada. Além disso, o caminho para aceitar ou rejeitar soluções candidatas são diferentes. As metaheurísticas baseadas em vizinhança são bem conhecidas na opimização combinatória. O mais simples é o método de descida *Hill Climber* e os mais interessantes são o *Simulated Annealing* (SA) e *Tabu Search* (TS), que podem escapar mínimos locais.

4.8 Estrutura de Vizinhança Utilizada na Busca Local

Seja (S, f) o problema de busca, onde S e f são, respectivamente, o espaço de busca e a função objetivo. Uma vizinhança N sobre S é qualquer função que associa a cada solução $s \in S$ algumas outras soluções $N(s) \subset S$. Qualquer solução de $s' \in N(s)$ é chamada de uma solução vizinha ou simplesmente vizinha de s. Para uma determinada vizinhança N, uma solução de s é um mínimo local em relação a N se *s* é o melhor em termos de f entre as soluções em N(s).

A noção de vizinhança pode ser explicada em termos do operador de movimento. Normalmente a aplicação de um movimento *mv* a uma solução *s* muda-a levemente e leva a uma solução vizinha *s'*. Esta transição de uma solução para outra vizinha é denotada por s' = s

$\oplus$ *mv*. Considere β(s) o conjunto de todos os movimentos possíveis que podem ser aplicados a s, então, a vizinhança N(s) de S pode ser definida por: N(s)={s $\oplus$ mv|mv $\in$ β (s)}.

Um algoritmo de busca local típico começa com uma configuração inicial s em S e continua iterativamente para visitar uma série de configurações seguindo a vizinhança.

Em cada iteração, um vizinho particular s' $\in$ N (s) substitui a configuração atual e a escolha de s' é determinada pela metaheurística que busca como referência a qualidade da solução vizinha. Por exemplo, um algoritmo de descida estrita sempre substitui a corrente solução s por um melhor vizinho s', enquanto Tabu Search substitui a solução atual por uma solução vizinha melhor s' mesmo que s' seja de qualidade inferior. Já no Simulated Annealing, a transição de s para alguns vizinhos escolhidos aleatoriamente é condicionada por uma probabilidade de mutação (NERI; COTTA; MOSCATO; 2011).

Na concepção de um algoritmo memético deve-se considerar cuidadosamente a definição da estrutura da vizinhança de uma solução, isto é, especificando que são as soluções viáveis que são acessíveis a partir de um dado ponto no espaço de solução. Se a vizinhança é muito pequena ou restrita, é improvável que bons ótimos locais sejam encontrados pela busca local. Por outro lado, se a vizinhança está completa, logo, a partir de cada ponto no espaço de busca é possível construir um caminho para qualquer outro ponto, mas neste caso, a busca passa a ser muito longa (talvez mesmo exponencial). Além disso, o tamanho das vizinhanças, embora importante, não é a única característica a ser considerada. Merz e Freisleben (1999), Kallel, Naudts e Reeves (2001) e muitos outros têm demonstrado através de uma análise estatística de panorama da aptidão que a escolha do operador de movimentação pelas soluções vizinhas pode ter um impacto significativo sobre a eficiência e a eficácia da busca local, e, portanto, do desempenho global do algoritmo memético. Além disso, a fim de reduzir os piores resultados durante a execução do algoritmo memético, é necessário escolher um método de busca local, cujo operador de movimento das soluções, seja diferente dos operadores de recombinação e mutação (KRASNOGOR, 2002). A recombinação, e particularmente a mutação, têm papéis valiosos na geração de pontos que se encontram em diferentes áreas de atração em relação ao operador de busca local (KRASNOGOR; ARAGON; PACHECO, 2006).

4.8.1 Combinação de Estruturas de Vizinhanças (Movimentos Distintos)

A vantagem da combinação de movimentos distintos ou estruturas de vizinhanças já foi demonstrada há um tempo em Lin e Kernighan (1973) para resolver o Problema do Caixeiro Viajante. Mais genericamente, a questão da transição entre os vizinhos alternativos

foi discutida com o framework Tabu Search e projeto de oscilação estratégica em Glover (1997). Exemplos mais recentes de métodos de busca local com foco em múltiplas estruturas de vizinhanças incluem Busca de Vizinhança Variável (HANSEN; MLADENOVIC, 1999), e Busca Progressiva de Vizinhança (GOEFFON, RICHER, HAO, 2008). Mais exemplos sobre estudos relacionados com combinações de vizinhanças podem ser encontrados em Hamiez, Robet e Hao (2009) e Lu, Hao, Glover (2010). Em Lu, Hao, Glover (2010), por exemplo, uma combinação de vizinhanças é utilizada onde quatro movimentos distintos são descritos denotados por: *SimpleMove*, *SimpleSwap*, *KempeMove* e *KempeSwap*, indicados por vizinhanças N1, N2, N3 e N4, onde apenas os movimentos que produzem os vizinhos que não impliquem qualquer violação das restrições são aplicados.

4.9 Problemas Discretos e Problemas Contínuos

Problemas de otimização discreta tem como principal interesse a busca por uma "melhor" configuração (solução ótima) entre um conjunto de configurações candidatas finito de acordo com um critério específico. Existem várias maneiras de descrever um problema de otimização discreta. Na sua forma mais geral, pode ser definida como uma coleção de instâncias do problema, onde cada instância é especifcada por um par (S, f), onde S é o conjunto de configurações candidatas finito, definindas no espaço de busca; f é o custo ou a função objetivo, dada por um mapeamento f: S $\rightarrow$ R+.

Dada a sua generalidade, a otimização discreta permite que muitos problemas de importância prática e teórica possam ser formulados convenientemente. Exemplos são os problemas clássicos de programação inteira geral, problemas de permutação (por exemplo, problema do caixeiro viajante, minimização de largura de banda em sistemas de comunicação, o arranjo linear de antenas), e satisfação das restrições e problemas de otimização (problemas de satisfação da lógica proposicional, particionamento de um grafo, k-coloração). Otimização discreta, naturalmente, abrange problemas práticos do meio ambiente, energia renovável, distribuição de energia elétrica, projeto de infraestrutura, comunicações e produtividade nos setores de manufatura e serviços.

Para problemas de otimização contínua, o espaço de decisão é, em princípio, um conjunto denso e é assim composto por uma quantidade infinita de pontos. Isto faz com que a enumeração fique impossível para a busca do ótimo. Deve-se observar que na representação dentro de uma máquina (finita), os conjuntos densos não podem ser representados como um conjunto infinito, e os conjuntos são tecnicamente finito e a distância entre cada um dos pares de pontos é pelo menos igual a precisão da máquina. Pode-se definir formalmente um ótimo

local em um espaço contínuo como um ponto de $so \in S$, tal que $f(so) \leq f(s)$, para todos os pontos $s \in S$ que satisfazem a relação $0 \leq \| so - s \| \leq \varepsilon$. O conjunto de pontos rodeados na região limitada pela magnitude de ε é a vizinhança do ótimo local *so*. Note que qualquer ótimo global também é um ótimo local, no entanto, o oposto não é necessariamente verdadeiro. De acordo com uma representação alternativa, na optimização contínua, um ótimo global é uma solução $s^* \in S$, de tal modo que $s^* = \min = f(s), s \in S$, onde $S \subseteq Rn$ é o espaço de busca viável, e $f: S \rightarrow R$ é a função de custo para minimizar. Quando $S = Rn$ o problema de encontrar o ótimo de *f* é chamado sem restrições, caso contrário, o problema é restrito (NERI, COTTA, MOSCATO, 2011).

5 Algoritmos Culturais: Abordagem Memética com Busca Local e Abordagem MultiPopulacional

5.1 Introdução

Nos Capítulos 2, 3 e 4 foram introduzidos algumas características sobre os Algoritmos Genéticos Seriais e Paralelos, Algoritmos Culturais e a abordagem memética.

Este capítulo tem como objetivo demonstrar algumas características importantes sobre a implementação dos Algoritmos Culturais (ACs) na perspectiva de uma abordagem memética e de uma abordagem multipopulacional. O objetivo principal dessas abordagens é auxiliar os ACs a superarem o desafio de escapar de ótimos locais em problemas de otimização. A abordagem memética aqui descrita foi construída tendo como base diferentes ensaios abordando os seguintes problemas: Problema das Funções Multimodais (PFM), Problema das Funções Restritas (PFR) e o Problema do Caixeiro Viajante (PCV). Já a abordagem multipopulacional leva em consideração o problema da mochila multidimensional (MKP).

5.2 Implementação do Algoritmo Cultural

Os dois componentes principais de um Algoritmo Cultural são o Espaço Populacional e o Espaço de Crenças. Neste trabalho, o espaço populacional é representado pela estrutura de um algoritmo genético e o espaço de crença utiliza dois tipos de conhecimento, por serem os mais utilizados na literatura e mais simples de implementar: (1) conhecimento situacional, que fornece o ponto exato onde o melhor indivíduo de cada geração foi encontrado, e (2) conhecimento normativo, que armazena os intervalos para as variáveis de decisão do problema, nas regiões onde foram encontrados os bons resultados.

5.3 Escolha da Busca Local

A busca local escolhida para esse trabalho foi o *Simulated Annealing.* A escolha foi motivada pela sua simplicidade de implementação, pela capacidade de aceitar parâmetros que permitam uma configuração ideal em relação à temperatura incial T e ao decremento dessa temperatura e o mais importante, a capacidade de escapar de ótimos locais devido a aceitação de configurações pouco favoráveis em relação à aptidão da função objetivo. O algoritmo *Simulated Annealing* padrão (SA) com duas opções para as estruturas de vizinhança é mostrado logo abaixo:

Algoritmo Simulated Annealing()

01:**INICIO**
// Pegar a solução inicial *S*, a temperature inicial *T e o fator de temperatura k*
02: *melhor←S;*
03: **REPITA ATÉ** (que *melhor* = solução ideal, ou o tempo tenha esgotado, ou T < 0)
04: S'← GerarVizinhoPFM (S); ou S'← GerarVizinhoPCV (S);
05: ∆custo ← custo(S') - custo(S);
06: **SE** (∆custo < =0) **ENTÃO**
07: S ← S';
08: **SENÃO**
09: S ← S' com probabilidade $e^{-\Delta custo/\mathrm{T}}$
10: *T* ← *K*T* /*redução de T com uma nova temperatura*/
11: **SE** (S< *melhor)* **ENTÃO**
12: *melhor←S;*
13: **FIMREPITA**
14: **RETORNE** S; //melhor solução
15:**FIM**

5.4 *Simulated Annealing* em Problemas Discretos e Problemas Contínuos

A maioria do desenvolvimento teórico e trabalhos que utilizam *Simulated Annealing* (*SA*) tem sido em problemas de otimização discretos. No entanto, o *SA* também tem sido utilizado como uma ferramenta para tratar de problemas no domínio contínuo. Há um grande interesse no uso de *SA* para otimização global sobre regiões que contêm vários mínimos locais e globais (devido a inerente não-linearidade das funções objetivo) (HENDERSON; JACOBSON; JOHNSON, 2003).

Alrefaei e Andradottir (1999) apresentam uma versão modificada do algoritmo de SA com uma temperatura constante para resolver os problemas otimização discreta e usam duas abordagens para estimar uma solução ótima para o problema. Tian, Ma e Zhang (1999) investigam a aplicação de SA para problemas de otimização discreta com a propriedade da permutação, como o Problema do Caixeiro Viajante (PCV), o problema de programação de fluxo de loja e os problemas de atribuição quadrática.

Fabian (1997) estuda o desempenho de SA para encontrar um mínimo global de uma função e Bohachevsky, Johnson e Stein (1986) apresentam um algoritmo SA para a otimização da função para o uso em aplicações estatísticas. A otimização de funções contínuas envolve a busca de um candidato local, escolhendo uma direção a partir da solução atual (histórico) e um tamanho de passo a dar nesse sentido, e avaliar a função no novo local (candidato). Se o valor da função desta localização candidata é melhor em relação ao valor da

função atual, então, o candidato se torna a solução atual. Esta migração através de mínimos locais em busca de um mínimo global continua até que o mínimo global seja encontrado ou alguns critérios de finalizaçao sejam atingidos. Belisle (1992) apresenta um algoritmo de SA especial para a otimização global, que usa um escalonador de resfriamento heuristicamente motivado. Este algoritmo é fácil de implementar e fornece uma alternativa razoável para os métodos existentes. Locatelli (1996) apresenta propriedades de convergência para uma classe de SA para otimização global contínua. Siarry et al. (1997) estuda SA para a minimização de funções de muitas variáveis contínuas.

Yang (2000) e Locatelli (2000) também fornecem resultados de convergência e os critérios para que o SA seja aplicado a problemas contínuos de otimização. Zhou e Chen (2010) também propõem um algoritmo com uma população baseada em SA. Idoumghar et al. (2011) propõe um sistema híbrido baseado em enxame de partículas e SA em otimização de funções multimodais.

A versão do *Simulated Annealing* utilizada neste trabalho é o algoritmo padrão *BuscaLocal_SA(....)* bastante utilizado (conforme mostrado anteriormente). Essa escolha foi motivada pela simplicidade do algoritmo, ou seja, de fácil implementação e também pelo fato que o algoritmo permite a aceitação de configurações (soluções) diferentes das melhores configurações alcançadas. A aceitação de diferentes configurações permite ao algoritmo escapar de um ótimo local em algumas situações. Em contrapartida, preferiu-se dar ênfase as estruturas de vizinhança utilizada pelo SA em função dos execelentes resultados mostrados durante os testes realizados. Dessa forma, trabalhou-se seguindo o SA padrão fazendo-se apenas alterações no processo de perturbações das soluções.

5.5 Estrutura de Vizinhanças Proposta em Problemas Contínuos: Problema das Funções Multimodais (PFM)

Mladenovic & Hansen (1997, 1999) propuseram o Método de Pesquisa em Vizinhança Variável (Variable Neighborhood Search, VNS) para a solução de problemas de otimização combinatorial e de otimização global, considerado um método de busca local que explora o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança.

De forma similar este trabalho aborda uma estrutura variável de vizinhança, onde a estrutura da busca local é mantida, ou seja, o algoritmo padrão de busca local *Simulated Annealling (SA)* padrão é utilizado, combinando diferentes estruturas de vizinhanças dentro da função *GerarVizinho(S)* que é chamada pelo *SA*.

Relembrando que o a descrição do problema de busca considera:

- Seja (S, f) onde S e f são, respectivamente, o espaço de busca e a função objetivo, uma vizinhança N sobre S caracterizada como qualquer função que associa a cada solução s ∈ S algumas outras soluções N(s) ⊂ S.
- Qualquer solução de s' ∈ N (s) é chamada de uma solução vizinha ou simplesmente vizinha de s.
- Para uma determinada vizinhança N, uma solução de s é um mínimo local em relação a N se s é o melhor em termos de f entre as soluções em N(s).

Em Silva e Oliveira (2009a) foi detectado que na maioria dos casos é apenas necessário aplicar perturbações de intensidade alta ou baixa em torno de pontos locais ótimos para obter melhorias em otimização de funções unimodais e multimodais, uma vez que estes pontos são próximo uns dos outros. Caso contrário, para escapar de um ótimo local é necessário aplicar uma movimentação ou perturbação com uma itensidade adequada, a fim de mover de um ponto para outros pontos em outras regiões do espaço de busca.

Uma das soluções encontradas nesse trabalho foi a definição de uma constante que é multiplicada pelo conjunto de perturbações pré-definidos. Outra exceção requer uma diversificação de pontos no espaço de busca (exploração). Isto normalmente ocorre quando uma função multimodal apresenta um grande número de locais ótimos. A fim de aumentar a exploração, foram criados pontos-chaves dentro do espaço de busca permitindo a perturbação para outras regiões (ver exploração de pontos-chaves logo adiante na Figura 19). Dessa forma, a busca local utiliza a combinação de quatro estruturas de vizinhanças: perturbações pré-definidas (N1), perturbações com pontos-chaves do Espaço de Busca (N2), perturbações de baixa intensidade (N3) e perturbações de alta intensidade (N4).

5.5.1 Perturbações Pré-Definidas (N_1)

As perturbações pré-definidas são perturbações estabelecidas pelo usuário e passadas por parâmetro. Esse tipo de perturbação é considerado como sendo uma "perturbação padrão", de intensidade média e que servirá de base para as outras perturbações.

Para as perturbações realizadas próxima de uma solução, considera-se uma variável chamada "Perturbation Distance" (P_D) em um "Intervalo da vizinhança" N_1 sobre S definido como N_{1S} de acordo com o parâmetro passado pelo usuário para cada variavel da função envolvida $x(i)$, definina como "Disturbance Limit" ($DL(i)$): $N_{1S(i)}$= [$-DL(i)$, $+DL(i)$]. Por exemplo, $DL(i)$=1.0, $N_{1S(i)}$ = [-1.0, 1.0] e $P_{D(i)} \in N_{1S(i)}$. Como comentado anteriormente, esse intervalo de perturbação é pré-definido como um parâmetro antes do início do algoritmo.

Outro parâmetro é chamado de fatia do intervalo “Slice Number” (*SN*) que irá determinar o incremento da perturbação, ou valor mínimo da perturbação “Increment of Disturbance” (*IncD(i)*) de acordo com a equação (1).

$$IncD_{(i)} = \frac{(DL(i)-(-DL(i)))}{SN(i)} \quad (1)$$

Ignorando o valor zero no intervalo N1S, o novo intervalo N1S(i) será: [-DL(i),..,-IncD(i), +IncD(i),..,+DL(i)]. Para a maioria dos testes realizados em Silva e Oliveira (2009a) para funções multimodais, foram utilizados os seguintes parâmetros: DL=1.0, SN=200. Dessa forma, utilizando o cálculo da equação 1 para essa configuração, o resultado para a variável IncD será 0.01. Esse intervalo deverá ser proporcional aos valores limites inferior e superior de cada variável x(i) utilizada pela função a ser otimizada. Por exemplo, o intervalo da variável x(i) = [-30,+30], DL=1.0 e SN=200. Então o intervalo de perturbação será: N1S(i) = [-1.0,..,-0.01, +0.01,..,+ 1.0] com IncD=0.01.

Já se o intervalo da variável x(i) =[-2048,2048], DL(i)=0.01 e SN(i)=200, o intervalo da perturbação será: N1S(i)= [-0.01,..,-0.0001,+0.0001,..,+0.01] e IncD=0.0001. Cada elemento de IncD(i) de cada variável x(i) é multiplicado pela direção da variável D onde D= +1 ou -1 que irá compor a “Perturbation Distance” (PD(i)= +/-D*IncD(i)). A Figura 17 mostra um exemplo. Considere um ponto do cromossomo Px(i) correspondente a variável x(i). O valor de PD(i) irá determinar a intensidade da perturbação na direção da esquerda (PLx(i)) ou direita (PRx(i)).

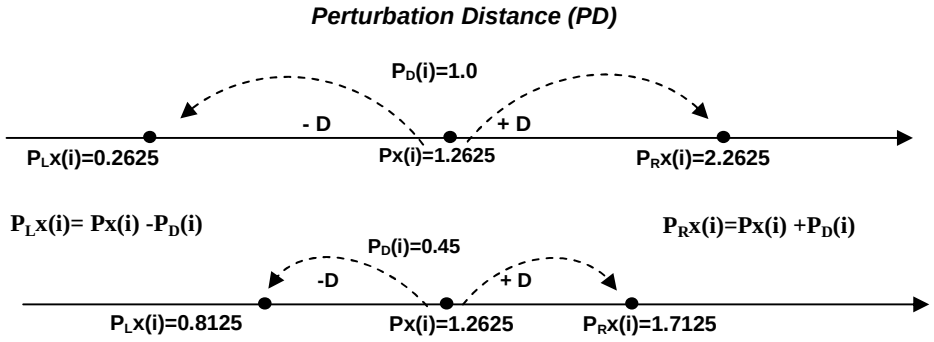


Figura 17: Perturbação pré-definida de um ponto P(x)

A Figura 18 descreve o comportamento da busca local utilizando perturbações pré-fixadas sobre o espaço de busca.

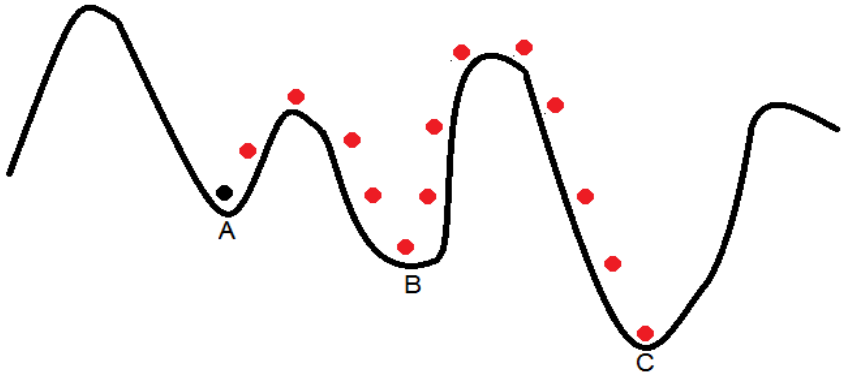


Figura 18: SA escapando de um ótimo local com vizinhança N_1

As perturbações são incrementadas gradativamente e tendem a ter um número maior de passos para escapar de um ótimo local, como no exemplo da Figura 25 onde parte-se de um ponto A para alcançar um ótimo global C.

5.5.2 Perturbações com Pontos-Chave do Espaço de Busca (N_2)

Para as perturbações realizadas próxima de uma solução em um "Intervalo da vizinhança" *N2* sobre *S* definido como *N2S* fixado pelo parâmetro "Disturbance Limit" (DL) que é proporcional aos limites inferiores (L(i)) e superiores (U(i)) de cada variável x(i) tal que N2S(i)= [L(i), U(i)]. Cada *i* corresponde ao índice da variável do cromossomo. Cada índice *p* corresponde a um ponto-chave de perturbação para cada *i*. Então maxP(i) corresponde ao número máxmo de pontos-chave para cada variável *i*.

Esse intervalo de perturbação é estabelecido através da divisão do intervalo da variável x(i) por um valor fixo maxP(i). Ao invés de apenas incrementar cada distância da perturbação PD(i)=+/-D*IncD(i), um dos ponto *p* para a posição do cromossomo *i* em *KP*(i,p) é escolhido aleatoriamente. Por exemplo, se o número de pontos escolhidos (maxP(i)) no intervalo do espaço de busca da variável *x*(i), for igual a 10, (*maxP*(i)=10), então seleciona-se um valor inteiro *p* escolhido aleatóriamente igual a *random*(1, 10) que será o valor fixo correspondente a *i* nos pontos-chave *KP[i][p]* (*Key Points*), ou seja o valor correspondente a *KP*(i,p). Por exemplo, se p=5 então a perturbação realizada sobre a posição *i* do cromossomo será *KP*[i][5]+D*P_D[i] (individuo[u,i]← *KP*[j][v]+D*P_D[j]).

Pode haver situações em que a aplicação de perturbações pré-definidas tais como as apresentadas anteriormente não são suficientes ou demanda um número maior de passos para alcançar uma solução ótima. Isso ocorre porque o ótimo global está longe da melhor solução atual e, portanto, distante do ótimo local. Nesse caso o algoritmo tenta escapar de um

ótimo local e encontra outro levando um tempo maior no número de passos de busca local. Para esta situação, a exploração segue uma definição de um número fixo de pontos de perturbação (maxP) espalhados no intervalo do espaço de busca de cada variável x(i) de acordo com a Figura 19, chamados de pontos-chave ou *Key Points* (KP).

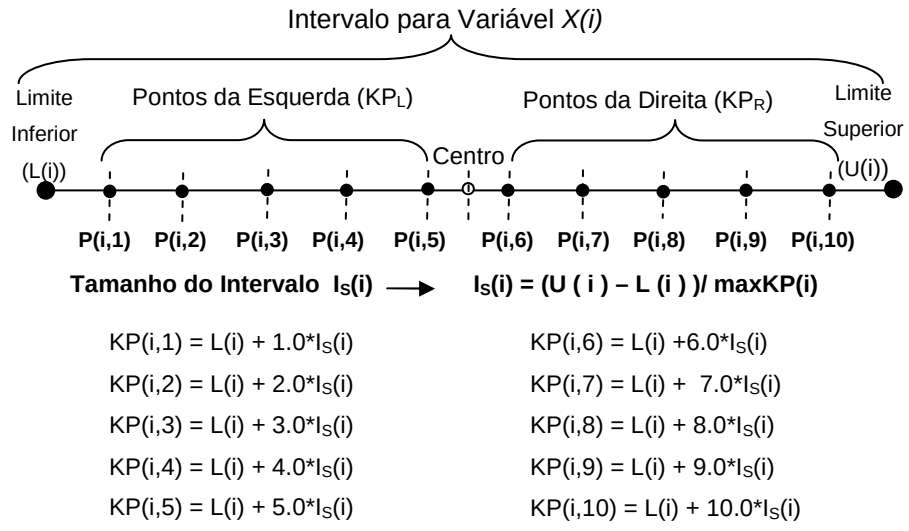


Figura 19: Pontos-chave do espaço de busca N_2.

Em testes de desempenho, as adições destes novos pontos ajudam o algoritmo de busca local no espaço de decisão. As variações aleatórias desses pontos usando a busca local garantem a diversidade, pois os pontos são modificados com grande intensidade de forma aleatória. Em cada geração de uma solução vizinha um ponto aleatório é escolhido de cada vez, através de uma sequência de números aleatórios com uma distribuição normal random (1, maxKP).

Além do deslocamento aleatório para um dos pontos KP(i,j), cada ponto passa a ter também influência das perturbações N1,ou seja, se v=5 então individuo[i,j]← *KP*[j][5] + D*P_D[j] (Linha 16 do algoritmo proposto *GerarVizinhoPFM(...)*).Ou seja, não se trata apenas em deslocar para ponto aleatórios, mas também trabalhar nas soluções vizinhas desses pontos após o deslocamento.

A Figura 20 representa possíveis deslocamentos do ponto *A* diretamente para outros pontos-chave *B*, *C*, *D* e *E*.

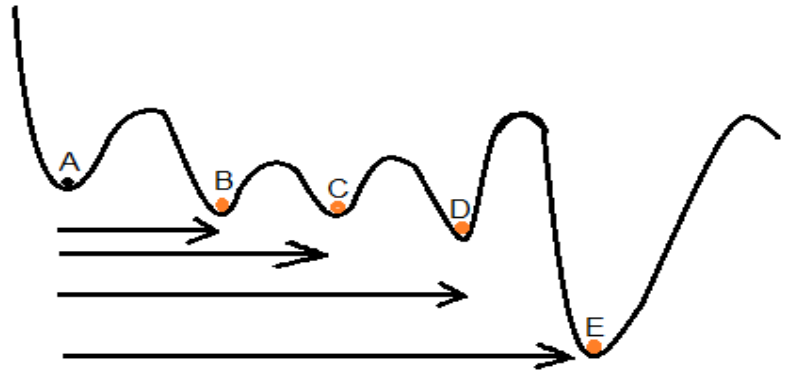


Figura 20: SA escapando de um ótimo local com N_2

5.5.3 Perturbações de Baixa Intensidade (N_3)

Foi detectado que o uso de pequenas perturbações ajuda a obter o ponto de mínimo ou máximo global na maioria das execuções. Esse tipo de perturbação é essencial, pois na maioria dos casos pequenas alterações no cromossomo pode representar uma grande diferença na função objetivo (fitness). Assim, para perturbações de baixa intensidade LD (low disturbance), o valor inicial para a variável x(i) (LD(i)) é o menor incremento da perturbação pré-definida: LD(i) = IncD(i). Em seguida, o valor LD é decrementado, multiplicando o seu valor por 0,5, ou seja, LD(i)= LD(i)*0,5. O valor mínimo para LD(i) é μ que pode ser fixado na faixa de valores baixos (por exemplo, μ = [1.0E-30,..,1.0E-17].

Para as perturbações realizadas próximas de uma solução em um "Intervalo da vizinhança" N3 sobre S definido como N3S fixado pelo parâmetro "Disturbance Limit" (DL) é proporcional aos limites inferiores e superiores tal que N3S(i)= [1.0E-17, IncD(i)]. A Figura 21 representa pequenos deslocamentos das soluções, dado que pequenos deslocamentos proporcionam pequenas melhorias nas soluções.

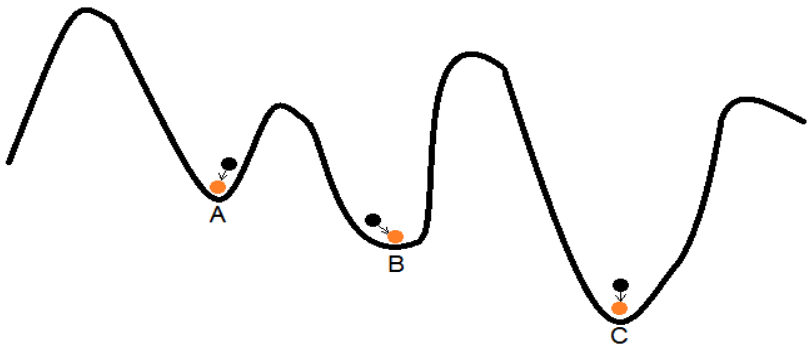


Figura 21: SA escapando de um ótimo local com N_3

5.5.4 Perturbações de Alta Intensidade (N_4)

A perturbação de alta itensidade cobre situações em que a aplicação de perturbações de baixa ou média intensidade não é suficiente ou requerem um número maior de passos por parte da busca local. Assim, para Distúrbios de alta intensidade (High Disturbance -HD) a Distância de Perturbação (PD) é aumentada. Neste caso, a Distância da Perturbação PD(i) é multiplicada por uma constante c: HD (i) = PD (i) * c. Por exemplo, a Figura 22 apresenta saltos de alta intensidade para outras regiões do espaço de busca, onde se encontram valores ótimos locais e até mesmo um ótimo global.

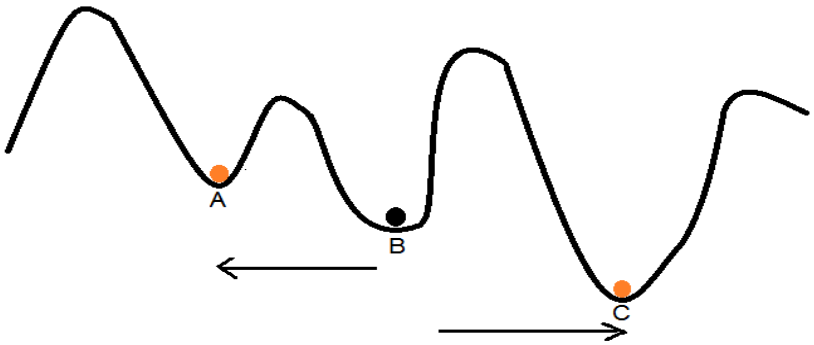


Figura 22: SA escapando de um ótimo local com N_4.

O algoritmo proposto (*GerarVizinhoPFM(...)*) para combinar a estrutura de vizinhanças N_1 a N_4 para o problema das funções multimodais é mostrado logo abaixo:

```
Algoritmo GerarVizinhoPFM(S)
01:INICIO
02:     D=-1; Min_D [ ]; Max_D [ ]; P_D[ ]= Max_D[ ];incD = - Min_D[ ];
03:     Inicializa vetor   perturbacaoN_1 (P_D[ ])
04:     Inicializa matriz perturbacaoN_2 (KP[ ][ ])
04:     Inicializa vetor   perturbacaoN_3 (L_D[ ])
05:     Inicializa vetor   perturbacaoN_4 (H_D[ ])
06:     L_D[ ]← Min_D [ ]
07:     KP[ ][ ] ←criarPontosVizinhançaN_2(); /* Figura 19*/
08:     Individuo: Real [ K ] [ tamCromossomo ]; /*K=4, N1..N4*
09:     PARA i = 1 to K  FAÇA  // k individuos
10:             p=Random (1 .. 10);
11:             PARA j =0 to tamCromossomo FAÇA
12:                 Hd[j]= P_D[j]*10.0;
13:                 Case i =1 Individuo [i,j]←C[j]+ D*P_D[j];
14:                 Case i =2 Individuo [i,j]←C[j]+D*L_D[j];
15:                 Case i =3 Individuo [i,j]←C[j]+D*Hd[j];
16:                 Case i =4 Individuo [i,j]←KP[j][p]+D*P_D[j];
17:                 Prob←ramdom(0,1);
18:                 SE (Prob <=0.5) ENTÃO
19:                     L_D[j]= L_D[j]*0.5;
```

```
20:                SE (L_D [j]< 1.0e-17) ENTÃO
21:                   L_D [j]= I_Low[j]
22:                SE (Prob(<=0.5) ENTÃO
23:                   P_D [j]= P_D [j]+ incD[j];
24:                SE (P_D[j] >= I_High[j]) ENTÃO
25:                   incD[j] ← - sd[j];
26:                SE (P_D [j]<= I_Low[j]) ENTÃO
27:                  incD[j] ← + sd[j];
28:                     FIMPARA j
29:             D← - D;
30:      FIMPARA i
31:      SE (Prob<=0.5) ENTÃO
32:      INICIO
33:         N←random(1,K)
34:        RETORNE Individuo (N);
35:      FIM
36:      ELSE
37:      INÍCIO
38:        avaliar_individuos(Individuo[ ])
39:        melhorIndividuo=selecionaMelhor(Individuo[ ])
40         RETORNE melhorIndividuo;
41:      FIM
42:FIM
```

5.6 Problemas Restritos

A abordagem memética para os problemas restritos utilizam os mesmos algoritmos descritos para o problema das funções multimodais. O único problema que deve ser levado em consideração são justamente as restrições impostas para o conjunto de funções elaboradas para cada tipo de problema.

Uma forma muito simples de lidar com restrições é o uso de penalização de funções. Nesse sentido, o valor de aptidão (fitness) é proporcional a presença ou ausência de restrições e até mesmo no número de restrições violadas durante o processo de otimização.

Portanto, uma das contribuições aqui é a elaboração de uma regra de penalização para as funções restritas envolvidas.

5.6.1 Função de Penalidade em Problemas Restritos Proposta

O processo de aceitação da função de aptidão depende da satisfação de todas as restrições dadas. Se alguma vez uma única restrição é violada, então, as soluções são chamadas de inviáveis.

O método da função penalidade proposta para este trabalho é dado logo a seguir:

Algoritmo Penalidade das Funções Restritas( )

```
01:INICIO
02:     FuncObj =CalculaFitness (individuo)
03:     SE (todas as restriçõs forem satisfeitas)  ENTÃO
04:     fitness = FuncObj;
05:     SENÂO
06:     INICIO
07:       valor_penalidade1= FuncObj +soma_fitness_Violados;
08:       valor_penalidade2=melhor-fitness-com-restr-satisfeita+soma_fitness_Violados;
09:       SE (FuncObj >= melhor-fitness-com-restr-satisfeita) ENTÃO
10:          fitness = valor_penalidade1;
11:     SENÃO
12:          fitness = valor_penalidade2;
13:     FIM
13: FIM
```

5.7 Considerações sobre Estrutura de Vizinhança para Funções Irrestritas e Restritas

De acordo com os algoritmos apresentados nesse capítulo é possivel verificar que a estrutura de vizinhança é proporcional ao número de váriáveis envolvidas na função escolhida para otimização. Por exemplo, o algoritmo GerarVizinhoPFM(S) apresenta na linha 11 um laço proporcional ao tamanho do cromossomo. Nesse sentido o presente trabalho leva em consideração uma codificação real em que o tamanho do cromossomo é proporcional ao número de váriáveis envolvidas, contemplando assim o espaço multivariável.

5.8 Estrutura de Vizinhança para Problemas Discretos: Problema do Caixeiro Viajante (PCV)

A criação de estuturas de vizinhança ou mecanismos de geração de soluções de vizinhança é um elemento crítico na concepção do *simulated annealing* eficiente para problemas de otimização discreta (HENDERSON; JACOBSON; JOHNSON, 2003). Tian, Ma e Zhang (1999) investigam a aplicação de *simulated annealing* para problemas de otimização discreta com uma propriedade de permutação. Eles introduzem seis tipos de esquemas de permutação para a geração de soluções aleatórias e provam que cada sistema satisfaz os requisitos de convergência.

Os resultados das avaliações experimentais sobre o PCV, o problema de programação de fluxo de loja, e o problema de atribuição quadrática sugerem que os ganhos de eficiência dos esquemas de perturbação são diferentes para cada tipo de problema e espaço de solução. Eles concluem que, com o esquema de perturbação adequada, *simulated*

annealing produz soluções eficientes para problemas diferentes de otimização discreta que possuem uma propriedade de permutação.

Koulamas, Antony e Jaen (1994) centram-se especificamente em *simulated annealing* aplicado na produção de gestão de operações e pesquisa de operações. Eles discutem os problemas tradicionais como o PCV, bem como problemas não tradicionais para incluir coloração de grafos e concluem que o *simulated annealing* é uma ferramenta eficaz para a solução de muitos problemas em operações de investigação e que o grau de precisão que o algoritmo consegue pode ser controlado pelo especialista, em termos de número de iterações e funções de estruturas de vizinhança. Ou seja, um aumento do número de iterações (loops externos) combinado com aumento do número de buscas a cada iteração (loops internos) pode resultar em soluções com maior probabilidade de convergência para a solução ótima.

Um Algoritmo Híbrido Cultural com Busca Local (HCALS) é apresentado por Kim e Cho (2009) aplicado ao PCV utilizando a busca local 2-opt. Majazi (2011) seleciona a melhor configuração para parâmetros primitivos de *simulated annealing* aplicado ao PCV através de planejamento de experimentos, a metodologia de superfície de resposta e programação de metas.

5.8.1 Busca local 2-OPT e 3-OPT

Johnson e McGeoch (1995) fornecem um excelente levantamento sobre algoritmos aproximados para o Problema do Caixeiro Viajante (PCV). Os algoritmos de busca local 2-OPT e 3-OPT são os mais conhecidos. O algoritmo 2-OPT remove duas arestas do circuito que são substituídas por outras duas arestas, caso essas sejam melhores em relação ao comprimento total do circuito.

O algoritmo 3-OPT é uma generalização do 2-OPT, substituindo até três arestas conforme a Figura 23 (as linhas pontilhadas indicam o circuito antes da transformação). Johnson e McGeoch (1995) também relatam melhorias de 6,9% e 4,6% acima do melhor valor encontrado (ótimo) para estes dois algoritmos se eles forem aplicados a circuitos gerados por uma versão aleatória do algoritmo guloso (*greedy).*

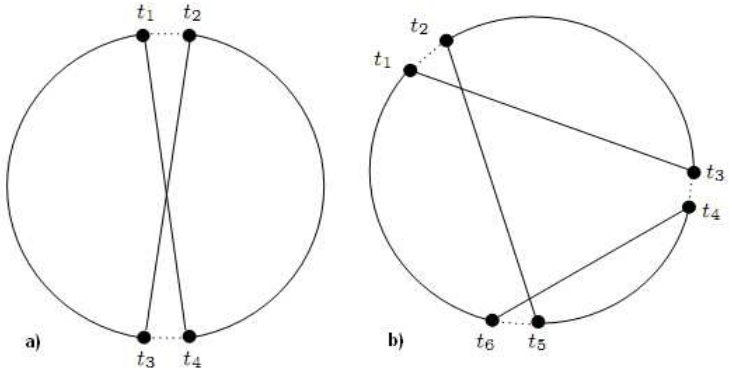


Figura 23: (a) Operação 2-OPT e (b) operação 3-OPT.

Heurísticas de melhoria de circuito, tais como 2-Opt, 3-Opt, e Lin-Kernighan (LK) (LIN; KERNINGHAN, 1973) produzem melhores circuitos, mas precisam de mais tempo de execução. Entretanto nada se compara quando se utiliza AGs puros nesse tipo de problema. Os AGs gastam muito mais tempo e encontram piores circuitos do que o método LK (Lin & Kerninghan, 1973). Os AGs, no entanto, podem ser utilizados em combinação com heurísticas de busca local para produzir soluções de alta qualidade. Entre os melhores métodos de AGs híbridos para o PCV estão o AG compacto com busca local LK (*GA with LK local search* GA-LK) e estratégia de otimização genética paralela assíncrona (*asynchronous parallel genetic optimization strategy* -Asparagos), AG com cruzamento natural (NGA), AG com cruzamento de arestas *edge assembly* (GA-EAX) , Busca Local Genética (Genetic Local Search GLS) e o método LK-Helsgaun (LKH). Estes métodos são bastante utilizados com êxito para resolver casos de milhares de cidades. Apenas GLS e LKH foram testados para instâncias de larga escala com até 100 000 cidades (NGUYEN; YOSHIHARA; YASUNAGA, 2007).

Provavelmente o mais famoso de todos os algoritmos para PCV é o algoritmo de Lin-Kernighan. Este também pertence à classe de algoritmos de otimização local, mas é significantemente mais complexo do que os algoritmos 2-OPT e 3-OPT. A idéia principal do método de Lin-Kernighan tem relação com o sucesso dos algoritmos k-OPT, que tinha fornecido anteriormente resultados promissores para k = 2 e K = 3. Para valores crescentes de ***k*** as soluções melhoram, mas a um grande custo computacional. A ideia básica do algoritmo de Lin-kernighan foi aplicar diferentes configurações de ***k*** durante o processo de busca (BOOMSMA, 2003).

O AC com busca local *Simulated Annealing* aplicada ao PCV proposto nesse trabalho utiliza a estrutura 2-OPT e 3-OPT, conforme o algoritmo *GerarVizinhoPCV(...)* mostrado logo abaixo.

```
Algoritmo GerarVizinhoPCV(S)
01:INICIO
02:     D=-1; Min_D [ ]; Max_D [ ]; P_D[ ]= Max_D[ ];incD = - Min_D[ ];
03:     Inicializa vetor perturbacao2-OPT (P_2opt[ ])
04:     Inicializa vetor perturbacao3-OPT (P_3opt[ ])
05:     Individuo: Inteiro [2] [ tamCromossomo ];
06:     PARA i = 1 to 2  FAÇA
07:        Case i =1 Individuo [i]← 2-OPT(S);
08:        Case i =2 Individuo [i]← 3-OPT(S);
09:     FIMPARA i
10:     Prob←ramdom(0,1);
11:     SE (Prob<=0.5) ENTÃO
12:     INICIO
13:         N←random(1,2)
14:         RETORNE Individuo (N);
15:     FIM
16:     ELSE
17:     INÍCIO
18:          avaliar_individuos(Individuo[ ])
19:          melhorIndividuo=selecionaMelhor(Individuo[ ])
20          FIM
21:    RETORNE melhorIndividuo;
22:FIM
```

Além disso, optou-se pela utilização de cruzamento de mapeamento parcial *partially mapped crossover* (PMX), recombinação de arestas (*edge recombination- ERX)* (LINDEN, 2006).

A mutação é realizada através de operadores de inversão do circuito em um determinado trecho, onde se inverte o cromossomo de acordo com valores de inicio e fim que podem ser escolhidos aleatoriamente. Outra operação é de troca de posições, onde duas posições são trocadas aleatoriamente. Os espaços de crenças utilizados nessa proposta para o problema são o espaço de crença situacional e o normativo baseado no trabalho de CARVALHO (2011). O espaço situacional guarda os melhores indivíduos encontrados com uma percentagem *p* sobre a população (geralmente 20%).

Para o problema abordado cada cromossomo representa um circuito, constituído por valores inteiros *i* que indicam a ordem do percurso. Para cada indivíduo *i* da população que será aplicada a mutação é feito o seguinte processo: primeiro busca-se uma posição no vetor do circuito (cidades), que tenha a mesma cidade do melhor cromossomo. Logo em seguida é realizado um sorteio para identificar o número de trocas que irá ocorrer (máximo 0.5). A mutação é feita, aplicando-se a cidade vizinha do melhor cromossomo conforme a Figura 24.

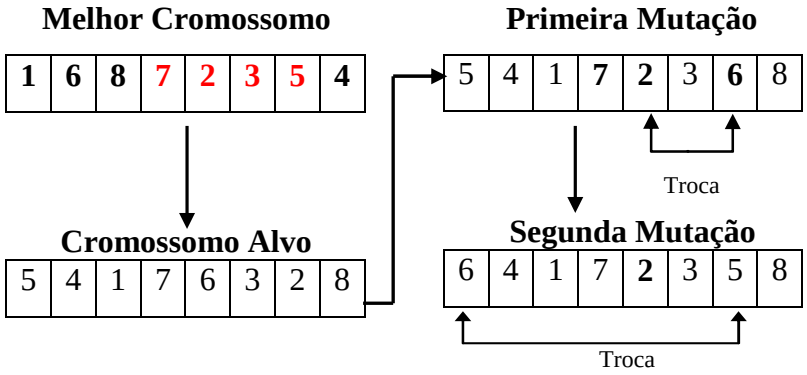


Figura 24: Representação da Mutação utilizando conhecimento situacional

Como o espaço de crença normativo trabalha com intervalos, a codificação foi baseada nas menores distâncias encontradas durante o processo evolutivo. A estrutura do conhecimento normativo é constituída por vários nós proporcionais ao número de cidades. Para cada cidade um nó contém a melhor ligação entre a cidade origem e a cidade destino, bem como suas distâncias encontradas durante a evolução (Figura 25).

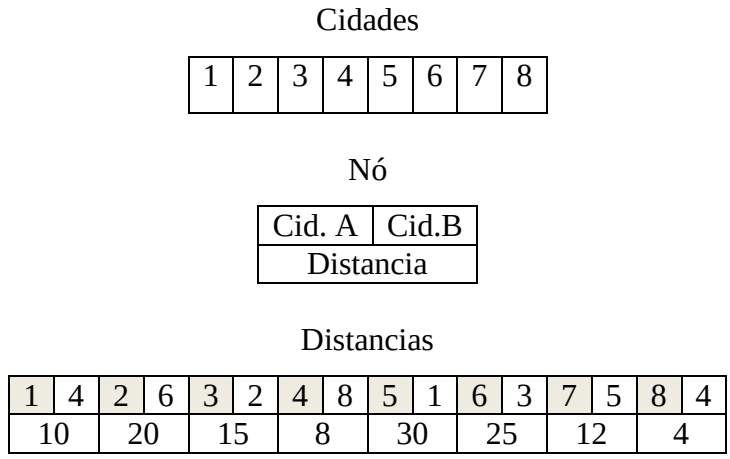


Figura 25:Representação da Mutação utilizando conhecimento Normativo

5.9 ACs com Abordagem Memética

Nesta seção são apresentados os detalhes do algoritmo híbrido proposto AC-BL (Algoritmo Cultural com Busca Local) ou CA-LS (*Cultural Algorithms with Local Search*). Esta abordagem oferece ao ACs meios para melhorar a velocidade de convergência e encontrar soluções ótimas. No algoritmo são realizadas quatro chamadas ao protocolo de comunicação, ou seja, duas a mais do que o AC tradicional. As quatro chamadas são: Comunicação (P(t), EP(t)), Comunicação (EP(t), P(t)) e duas chamadas Comunicação (Melhor_Encontrado(t),EP(t)).

O framework do algoritmo cultural com busca local proposto é mostrado na Figura 26.

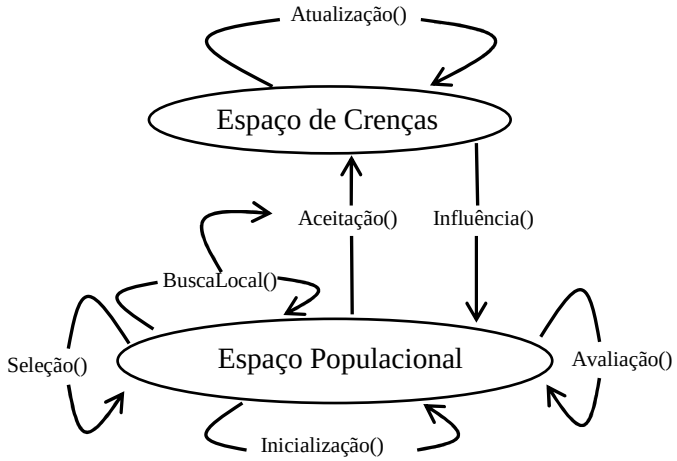


Figura 26: Framework do AC com busca local proposto.

Toda a operação do AC-BL leva em consideração a busca local em dois pontos seguindo a estrutura do AC como mostrado abaixo (*Algoritmo CulturalBuscaLocal( )*):

Algoritmo CulturalBusca Local( )

```
01: INÍCIO
02:     variáveis: t=0 (geração atual); tu=0 (geração do último valor ótimo),
03:     z  (diferença  mínima para chamar a BuscaLocal-SA)
04:     inicializar população P(t)              //população inicial aleatória
05:     Inicializar Espaço de Crença EP(t)
06:     avaliar população P(t)                  // calcula f(i) para cada indivíduo
07:     ENQUANTO (não condição_fim) FAÇA
08:         diferença=t- tu;                     // o valor de tu dependerá da evolução
09:         Comunicação (P(t), EP(t));          // votação(Aceitação)
10:         Atualização EP(t);                  // uso de operadores culturais
11:         SE (diferenca>=z) ENTÃO
12:             Selecionar Melhor_Individuo da população P(t);
13:             Melhor_Encontrado ←BuscaLocal-SA(Melhor_Individuo);
14:             Comunicação (Melhor_Encontrado(t) ,EP(t));
15:             Atualização EP(t);              // uso de operadores culturais
16:         FIMSE
17:         Comunicação (EP(t), P(t));      // promoção (função de influência)
18:         t←t+1;                          // próxima geração
19:         selecionar P(t) de P(t-1);
20:         altera P(t);                     // crossover e mutação
21:         avaliar P(t);                    //  calcula f(i) para cada indivíduo
22:         SE (diferenca>=z) ENTÃO
23:             Selecionar Individuo_Aleatório da população P(t);
24:             Melhor_Encontrado ← BuscaLocal-SA(Individuo_Aleatório);
25:             Comunicação (Melhor_Encontrado(t) ,EP(t));
26:             Atualização EP(t);  // uso de operadores culturais
27:         FIMSE
28:     FIMENQUANTO
29:FIM
```

5.10 ACs com Abordagem Multipopulacional

Algoritmos Genéticos Multipopulação ou Modelo de ilhas é uma extensão do tradicional algoritmo genético de população simples (AGS), dividindo a população em várias subpopulações em que o produto da evolução e os indivíduos estão autorizados a migrar de uma subpopulação para outra. Valores diferentes para os parâmetros tais como taxa de recombinação, seleção e mutação podem ser escolhidas para cada subpopulação.

Normalmente, o modelo básico de ilhas utiliza os mesmos valores para esses parâmetros em todas as subpopulações. Com a finalidade de controlar a migração dos indivíduos, vários parâmetros foram definidos, tais como: (i) a topologia de comunicação que define as conexões entre as subpopulações, (ii) uma taxa de migração que controla a quantidade de indivíduos durante a migração, e (iii) um intervalo de migração que afeta a frequência de migração. Além disso, a migração deve incluir estratégias para a seleção de migrantes e sua inclusão na sua nova subpopulação (AGUIRRE et al., 2000).

O tamanho das subpopulações, topologia de comunicação (o seu grau de conectividade), taxa de migração e frequência de migração são fatores importantes relacionados com o desempenho de AGs distribuídos. Em geral, tem sido mostrado que AGs distribuídos pode produzir soluções com qualidade semelhante ou melhor do que o AGS (AGUIRRE et al., 2000). No modelo AG baseado em ilhas, as subpopulações são isoladas durante a reprodução, seleção e avaliação. As ilhas normalmente incidem sobre o processo evolutivo dentro das subpopulações de indivíduos antes de migrar para outras ilhas, ou processadores conceituais, que também realizam um processo evolutivo. Em horários pré-determinados, durante o processo de busca, ilhas enviam e recebem migrantes de outras ilhas.

Um exemplo da topologia de comunicação pode ser definido como um gráfico em que as subpopulações de Pi (i = 0, 1, ..., K-1) são os vértices e cada borda definida Li,j especifica uma comunicação entre o incidente vértices Pi e Pj (subpopulações vizinhas) (AGUIRRE et al., 2000). Em geral, assumindo um gráfico dirigido para cada ligação definida por Li,j pode-se indicar o número de indivíduos Ri,j que irá migrar de P para Pj (taxa de migração) e o número de gerações M entre os eventos de migração (intervalo de migração). A topologia de comunicação e as taxas de migração podem ser estáticas ou dinâmicas enquanto que a migração pode ser assíncrona ou síncrona.

Várias estratégias de migração podem ser escolhidas para serem aplicadas nesse modelo. Duas estratégias frequentemente utilizadas para selecionar os migrantes são a seleção dos melhores e seleção aleatória. Por exemplo, a migração pode implementar uma estratégia

de transmissão síncrona elitista ocorrendo a cada geração *M*. Cada subpopulação transmite uma cópia de seus *R* melhores indivíduos a todas subpopulações vizinhas. Assim, cada subpopulação em todos os eventos de migração recebem migrantes. A Figura 27 ilustra um modelo de ilhas com uma topologia de comunicação +l+2 em que cada subpopulação está ligada a dois vizinhos (L = 2).

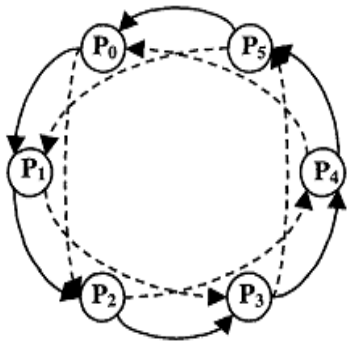


Figura 27 Topologia de comunicação +1+2.

Neste exemplo, a subpopulação P_0 pode enviar os indivíduos apenas para P_1 e P_2 e receber migrantes apenas a partir de P_4 e P_5.

5.10.1 Algoritmo Cultural Baseado Modelo de Ilhas (AC-MI)

Nesta seção é apresentada uma abordagem sobre a topologia de comunicação para o processo de migração implementada em um Algoritmo Cultural baseado no modelo de ilhas. Como observado anteriormente na implementação clássica do modelo de ilhas, há subpopulações conectadas em uma estrutura de anel. Indivíduos em modelo de ilhas clássico migram após cada intervalo de migração (M) entre as gerações e a política do pior/melhor indivíduo na migração é utilizada.

Como já descrito anteriormente a abordagem utilizada neste trabalho é uma adaptação do modelo de ilhas sobre a estrutura do algoritmo cultural aqui identificado como AC Modelo de Ilhas (AC-MI) ou *CA Island Model* (CA-IM), brevemente introduzida em Silva e Oliveira (2009b) e que posteriormente foi detalhada por Silva, Teixeira e Oliveira (2012). A principal característica presente no AC-MI é a ligação entre o espaço de crenças da população principal e espaço de crença das subpopulações. As transformações culturais ocorrem em paralelo tanto em relação à população principal quanto às subpopulações das ilhas. O elo de comunicação entre os dois espaços de crenças, permite a migração entre os melhores indivíduos armazenados na estrutura do conhecimento cultural implementada. A Figura 28 mostra o correspondente quadro para a estrutura proposta.

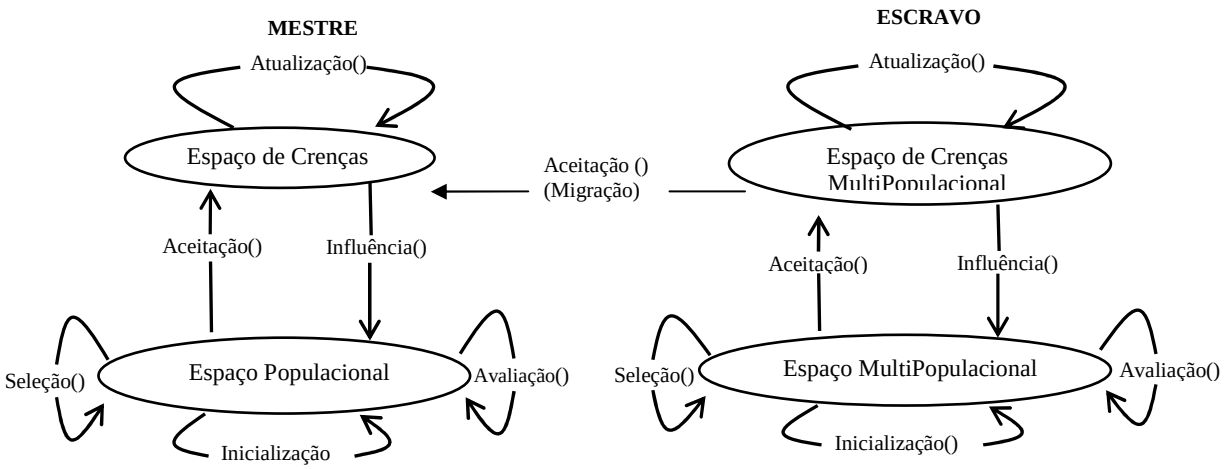


Figura 28: :Famework do AC multipopulacional (SILVA; TEIXEIRA; OLIVEIRA, 2012).

Na estrutura proposta pode-se notar a falta de um link do **mestre** parao **escravo**. Isso foi feito propositalmente, para evitar que algumas informações presentes na população mestre possam influenciar a população presente na população escravo, uma vez que o conhecimento adquirido na população mestre permanecerá até o fim. Já na população escravo as informações podem ser recriadas, já que a população pode ser eleiminada e recriada a qualquer momento. Essa estratégia trouxe resultados positivos. As migrações entre as ilhas ocorrem através da estrutura do espaço de crença **Espaço MultiPopulacional** (escravo) que realiza o processo de comunicação entre as sub-populações e envia os melhores indivíduos do **Espaço MultiPopulacional** para o **Espaço Populacional** (mestre) através da função de Aceitação( ), inserida no modelo somente para a migração dos indivíduos. Ela ocorre em um intervalo predefinido cujo parâmetro é *M* (a cada geração *M*), onde os indivíduos são avaliados pela função de aceitação e atualizados no espaço de crenças principal. A migração do **Espaço MultiPopulacional** para o **Espaço Populacional** é realizada utilizando-se um número de indivíduos que são considerados como um conjunto de exemplos para o resto da população (Conhecimento Situacional).

É importante notar que o AC-MI fornece uma verificação contínua entre a última solução (valor ótimo) encontrada e a solução atual. Em seguida, calcula o número de gerações onde não ocorrem melhorias. Assim, se a distância entre a última geração, onde a solução corrente foi encontrada, e a geração atual for um valor alto, então as subpopulações são eliminadas e recriadas aleatoriamente. Há uma diferença para esta ocorrência na faixa de 60 a 100 gerações. Se uma nova solução não for encontrada nesse intervalo, então, as subpopulações das ilhas (Espaço Multipopulacional), bem como a informação cultural sobre todas as subpopulações (Espaço de Crenças Multipopulacional) são recriadas de forma aleatória pelo algoritmo.

6 Resultados

Neste capítulo serão mostrados os resultados da aplicação dos algoritmos descritos no capítulo 5, para os problemas de otimização delimitados nesse trabalho. Para verificar o desempenho dos algoritmos propostos, foram avaliados os resultados obtidos para cada problema abordado bem como a comparação desses resultados com outros algoritmos já publicados em trabalhos que têm relação com a computação evolucionária e com as técnicas mais utilizadas de otimização. Ou seja, são realizadas comparações com os algoritmos na literatura que utilizam os mesmos *bechmarks* para cada problema apresentado nesse trabalho. Os parâmetros utilizados foram escolhidos manualmente, logicamente sintonizados um pouco para mais ou menos, levando-se em consideração os parâmetros utilizados em outros trabalhos. Já os novos parâmetros aqui propostos tem como base um conjunto de ensaios baseados em cada tipo de problema. Foram os ensaios realizados previamente que deram subsídios para os novos parâmetros propostos nesse trabalho.

6.1 Resultados para Otimização Funções Multimodais

Com o objetivo de comprovar o desempenho do Algoritmo Cultural com Busca Local (AC-BL), 9 funções sem restrições foram escolhidas (funções definidas no anexo A5). Inicialmente (o primeiro grupo), quatro funções de *benchmarks* (F1 a F4) são utilizadas. Essas quatro funções são apresentadas por Xue e Guo (2007) na descrição dos algoritmos: um algoritmo Cultural baseada em Algoritmo Genético (GACA) e um Algoritmo Cultural com base em Algoritmo Genético Multi-Janelas (MWGACA).

O GACA é um AC padrão, com uma população baseada em AG. No algoritmo MWGACA mais de uma pesquisa populacional são introduzidas, como uma população para a realização de busca local e uma população para busca global. Como observado anteriormente, todas as funções de *bechmark* deste trabalho podem ser vistas nos anexo A5. A primeira função, F1, é a função de "peaks". As funções F2 e F3 são as funções de teste propostas por De Jong e a função F4 é a função de “Shuber". No segundo grupo são utilizadas cinco funções de *benchmarks* (F5 a F9) que são utilizados em Ling et al.(2008), Nguyen et al. (2009) e Idoumghar et al. (2011) para mostrar as diferenças de desempenho entre o AG, AC e o AC-BL. Algumas comparações com outros algoritmos serão realizadas. Os algoritmos para comparação são baseados em otimização por enxame de partículas (Particle Swarm Optimization - PSO) como: *Hybrid Particle Swarm Optimization* (HPSOM), *Hybrid Particle Swarm Optimization with Wavelet Mutation* (HPSOWM) (LING et al., 2008) and *Hybrid*

Particle Swarm Optimization with Simulated Annealing (HPSO-SA) (IDOUMGHAR et al., 2011). Os algoritmos meméticos celulares são apresentados de acordo com Nguyen et al. (2009). As funções F5 e F6 são as de Rosenbrock e de Rastrigin respectivamente, já F7, F8 e F9 são: função de Hartman, função de Ackley e a função penalizada generalizada. Para efeito de comparação, apenas o tamanho da população e o número de gerações utilizadas pelos *benchmarks* são levados em consideração, dada a importância em estabelecer um parâmetro em relação ao numero de avaliações realizadas com base na função objetivo (*fitness*).

Os parâmetros básicos do AC-BL para as funções F1 a F9 são mostrados pela Tabela 1. O anexo A9 mostra como são passados esses parâmetros para o *framework.*

Tabela 1: Parâmetros básicos para as funções F1 a F9.

Funções	**Tam.da População**	**Seleção Torneio**	**Tx. de Cruzamento**	**Tx. de Mutação**	**Núm. de Gerações**	**Núm. Max de Aval. com/sem Busca Local**
F1-F4	200	7	0,75	0,025	50	10.000
F5-F6 (20D)	30	7	0,85	0,025	2000	60.000
F5- F6(30D)	100	7	0,85	0,025	3000	300.000
F7	20	7	0,65	0,025	100	2.000
F8	20	7	0,65	0,025	1500	30.000
F9	20	7	0,65	0,025	1000	20.000

Os resultados foram obtidos através de 50 execuções independentes por problema, onde “média" indica os valores médios das aptidões provenientes dos melhores valores encontrados para cada execução independente. A Tabela 2 mostra os parâmetros utilizados no AC-BL para F1 a F9. Os valores relativos aos vetores MinD[ ] e MaxD[ ] na Tabela 2 são fixados para todas as variáveis de cada função, isto é, no processo de perturbação, o valor do tamanho de passo das perturbações é o mesmo para cada variável da função.

Tabela 2: Parâmetros da busca Local SA com vizinhança *GerarVizinhoPFM (..)* com K=4 (valor fixo).

Funcão	**MinD[ ]**	**MaxD[ ]**	**Temperatura T**	**Z**
F1	0.001	1.0	15	*10*
F2	0.0001	0.01	15	*10*
F3	0.001	1.0	10	*10*
F4	0.001	1.0	15	*30*
F5	0.1	1.0	10	*30*
F6	0.25	1.0	10	*30*
F7	0.01	1.0	10	*40*
F8	0.01	1.0	10	*40*
F9	0.1	1.0	10	*40*

A Tabela 3 mostra os resultados das simulações onde se observa que o AC melhora os resultados em relação ao AG. No entanto, de forma semelhante ao AG, o ótimo global não foi encontrado em alguns ensaios independentes. Os resultados mostram que o AC-BL superaram os resultados do AC, na maioria dos aspectos. A Tabela 3 é composta pelo melhor valor encontrado, média dos melhores valores, Desvio Padrão e Taxa de Sucesso

(ocorrências do valor ótimo) para 50 ensaios. O número de variáveis é representado por *n*.

Tabela 3: Comparação dos resultados para as funções F1 a F9

Função	Algoritmo	Referência do Algoritmo	Melhor	Média	Desvio Padrão	Taxa de Sucesso
F1	AG	-	8.10484841	7.46189623	1.37570881	5/50
n=2	AC	-	8.10621358	8.10618741	1.248686E- 4	15/50
	AC-BL	**-**	**8.10621358**	**8.10621413**	**7.167712E-13**	**50/50**
F2	AG	-	4.614E-12	0.008790	0.017099	00/50
n=2	AC	-	7.88E-13	3.142845E-5	9.975652E-5	00/50
	AC-BL	**-**	**0.0**	**0.0**	**0.0**	**50/50**
F3	AG	-	10.0	9.98	0.141422	49/50
n=2	AC	-	10.0	10.0	0.0	50/50
	AC-BL	**-**	**10.0**	**10.0**	**0.0**	**50/50**
	AG	MWCA	-186.730908	-186.716875	0.044273	44/50
F4	AC	MWGACA	-186.730908	-186.721363	0.058464	47/50
n=2	MWCA	(XUE ; GUO,	-186.730908	-186.64	-	3/50
	MWGACA	2007)	-186.730908	-186.69	-	13/50
	AC-BL		**-186.730908**	**-186.730908**	**2.8710E-14**	**50/50**
F5	AG	HPSO-SA	433,7775	5111.344900823	4421.85385224	00/50
n=20	AC	(IDOUMGHAR	0.202879132	6.243332	8.3768305	00/50
(20D)	HPSO-SA	et al. , 2011)	0.227048188	0.58359742	0.2155	-
	AC-BL	**-**	**0.000000**	**0.0**	**0.0**	**50/50**
F5	AG	CMA e ACMA	60.489894	119.873665	60.153712	00/50
n=30	AC	(NGUYEN	0.008079	11.166426	8.614923	00/50
(30D)	CMA	et al. ,	0.000089	1.436206	2.725633	00/50
	CMAR5	. , 2009)	0.007964	7.762101	10.577379	00/50
	ACMA5	-	0.000072	2.288657	3.677642	00/50
	ACMA10	**-**	0.000466	1.230378	2.738209	00/50
	AC-BL	-	**0.000000**	**0.000000**	**0.000000**	**50/50**
F6	AG	HPSO-SA	0.001288	0.026577458	0.026253578	00/50
n=20	AC	(IDOUMGHAR	3.8341312E-08	0.001608234	0.003196314	00/50
(20D)	HPSO-SA	et al. , 2011)	4.3599434E-09	0.000000000	0.000000000	-
	AC-BL	**-**	**0.000000**	**0.000000**	**0.000000**	**50/50**
F6	AG	CMA- e ACMA	0.00655224	0.108048	0.070683	0/50
n=30	AC	(NGUYEN	1.04345E-06	0.001882	0.005063	0/50
(30D)	CMA	et al. ,	0.000000	0.000000	0.000000	50/50
	CMAR5	. , 2009)	0.000000	0.000000	0.000000	50/50
	ACMA5	-	0.000000	0.000000	0.000000	50/50
	ACMA10	-	0.000000	0.000000	0.000000	50/50
	AC-BL	**-**	**0.000000**	**0.000000**	**0.000000**	**50/50**
F7	AG	HPSOM e -	-3.3120487	-3.2403534	0.088258	00/50
n=6	AC	HPSOWM -	-3.3215512	-3.2710123	0.058047	20/50
	HPSOM	Ling et al.	-3.3219952	-3.2577928	0.059857	-
	HPSOWM	(2008)	-3.3219952	-3.2934608	0.051292	-
	AC-BL	-	**-3.3219121**	**-3.3217256**	**2.24858E-4**	**50/50**
F8	AG	HPSOM e -	7.84973E-06	3.2031931E-4	9.9585E-4	0/50
n=30	AC	HPSOWM -	4.0241E-8	1.5278E-4	3.3269E-4	0/50
(30D)	HPSOM	Ling et al.	6.6542E-5	13.8681E-5	3.8260E-5	-
	HPSOWM	(2008)	0.4919E-5	1.0607E-5	0.3120E-5	-
	AC-BL	-	**0.000000**	**0.000000**	**0.000000**	**50/50**
	GA	HPSOM e -	1.64E-05	0.001432	0.002015	00/50
F9	CA	HPSOWM -	1.21E-12	1.498E-7	8.6895E-7	33/50
n=30	HPSOM	Ling et al.	0.0033E-5	0.0097E-5	0.0089E-05	-
(30D)	HPSOWM	(2008)	0.000000	0.0001E-05	0.0001E-05	-
	AC-BL		**0.000000**	**0.000000**	**0.000000**	**50/50**

As funções F1 e F3 são solucionadas mais facilmente. Além disso, a função F2 precisa escapar de um mínimo local. Neste caso, o AC padrão necessita de muitas gerações para um pequeno aumento no desempenho. Os resultados mostram que a fuga de mínimos locais pelo AC-BL para a função F2, melhoraram os resultados para a função de F1 e F3.

De acordo com os resultados em Xue e Guo (2007) o algoritmo MWGACA alcança apenas 13 melhores ótimos conhecidos de um total de 50 execuções (13/50) que correspondem a 26%, com uma média de -186,69.

O algoritmo proposto AC-BL atingiu 50 melhores ótimos conhecidos de um total de 50 execuções (50/50) para as funções F5 e F6, o que corresponde a 100% no desempenho. É evidente que a partir dos resultados de testes, o AC-BL supera os algoritmos propostos em Ling et al.(2008), Nguyen et al. (2009) e Idoumghar et al. (2011) para as funções F7 a F9. É importante notar que o AC-BL tem uma forte capacidade de sair do ótimo local, desde que os parâmetros sejam sintonizados corretamente, prevenindo assim a convergência prematura.

6.1.1 Gráficos para as Funções Multimodais

Com objetivo de visualizar o desempenho entre os algoritmos implementados algumas funções foram selecionadas para plotar os gráficos conforme a seguir.

A Figura 29 e a Figura 30 mostram a média para cada uma das 50 gerações distribuidas nos 50 ensaios realizados para F2 em relação ao AG x AC e AC x AC-BL respectivamente.

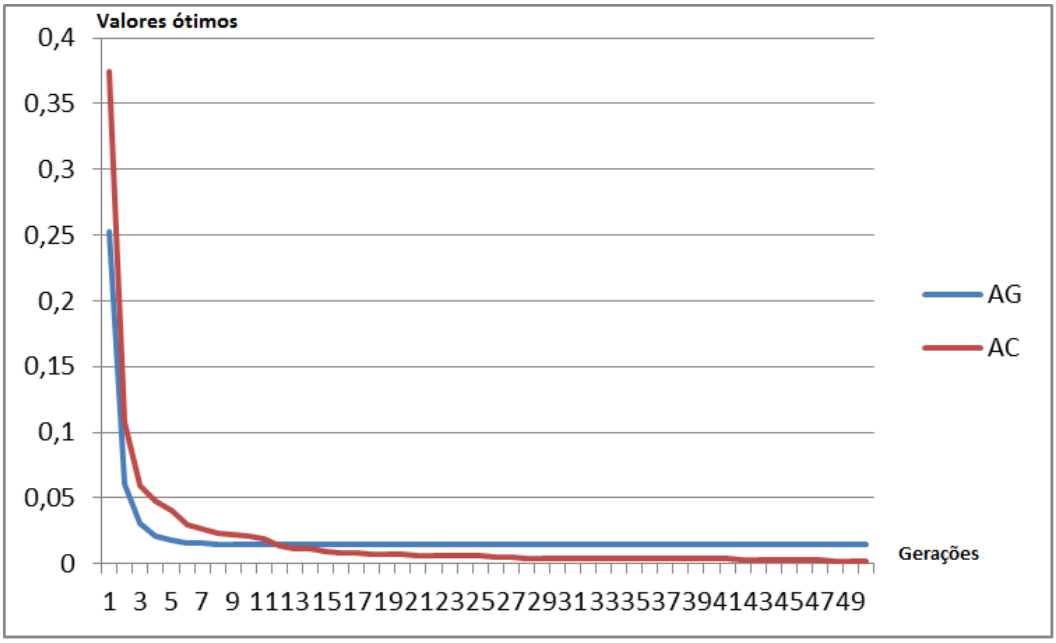


Figura 29: Desempenho do AG xAC para a função F2.

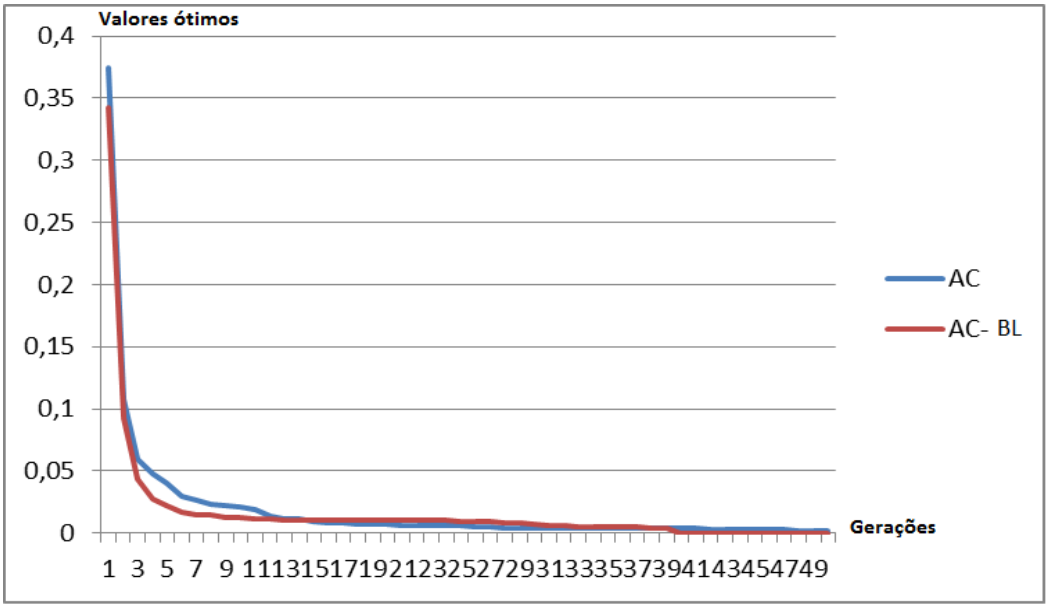


Figura 30:Desempenho do AC xAC-BL para a função F2.

A Figura 31 (escala logaritma de base 10) e a Figura 32 mostram a média para cada uma das 3000 gerações distribuidas nos 50 ensaios realizados para F5 (n=20) em relação ao AG x AC e AC x AC-BL respectivamente.

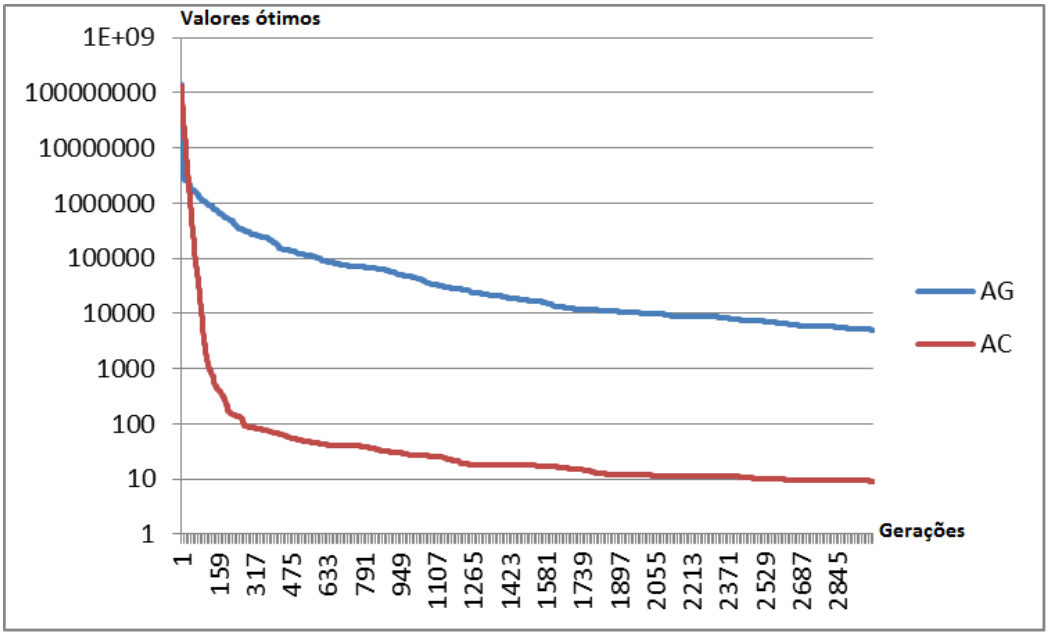


Figura 31: Desempenho do AG x AC para a função F5 (n=20).

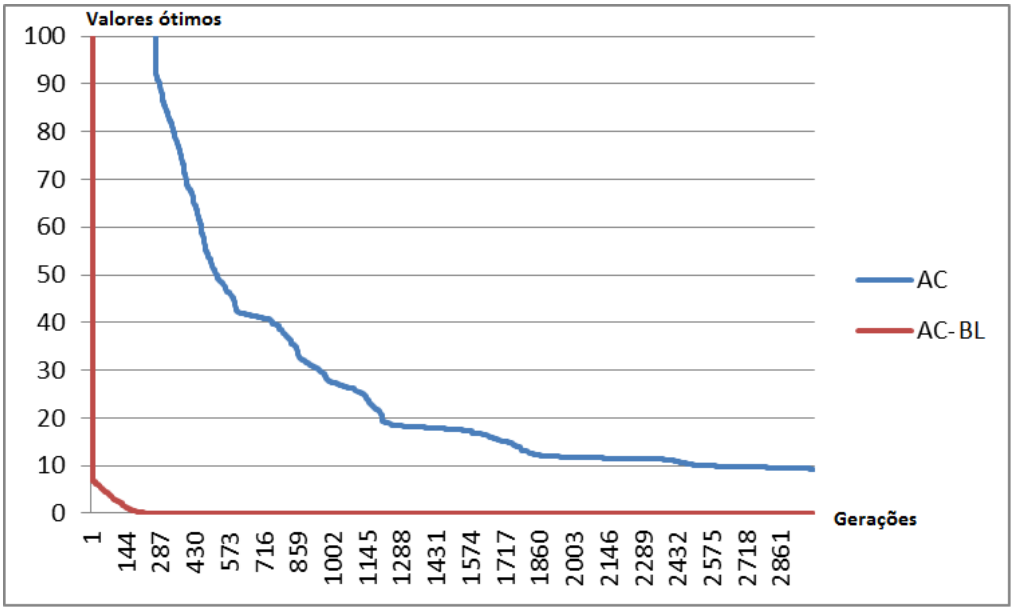


Figura 32: Desempenho do AC x AC-BL para a função F5 (n=20).

6.2 Resultados para Otimização de Funções Restritas (Problemas de Engenharia)

Nesta seção, serão mostrados alguns resultados baseados nos ensaios do algoritmo proposto AC-BL em alguns problemas com restrições bem conhecidos de engenharia para investigar o desempenho do algoritmo AC-BL. Os problemas selecionados foram: minimização do peso da tensão/compressão sobre mola (Problema-1) e projeto de vaso de pressão (problema-2). Os parâmetros do AC-BL para o problema-1 são: tamanho da população = 100, número total de gerações = 500, com 20 e 30 execuções independentes. Temperatura T = 100 e Z = 10. O total de avaliação do número de avaliações do fitness (Functions Evaluations Number- *FEN*), incluindo a busca local, é de 100.000. A Tabela 4 mostra o intervalo de perturbações utilizado no Problema-1.

Tabela 4:Intervalo de perturbação para o Problema-1.

Variável	Min_D	Max_D
X_1(d)	0.001	0.05
X_2(D)	0.001	0.05
X_3(P)	0.05	1.0

Os parâmetros do AC-BL para o Problema-2 são: tamanho da população = 200, número total de gerações = 500 e 50 execuções independentes. Temperatura T = 100 e Z = 10. O FEN total, incluindo a busca local, é de 200.000. A Tabela 5 apresenta o intervalo de perturbações para o Problema-2:

Tabela 5: Intervalo de perturbação para o Problema-2.

Variável	Min_D	Max_D
X_1	0.0625	1.25
X_2	0.0625	1.25
X_3	0.025	10.0
X_4	0.025	10.0

A partir da Tabela 6, Tabela 7, Tabela 8, Tabela 9 e Tabela 10, é possível visualizar que as melhores soluções encontradas pelo AC-BL para o problema de tensão/compressão da corda e pressão do vaso são melhores do que os melhores resultados obtidos pelo AC, pelo algoritmo *Social Fabric* (Topologia: Ibest e Topologia: square) (Reynolds; Mostafa, 2008), Coello e Montes (2002), G-QPSO (2) (COELHO, 2010) e CPSO (HE; WANG, 2007).

A partir da Tabela 6, Tabela 7 e Tabela 9 é possível visualizar os resultados sobre a média, o pior resultado e o desvio padrão que são produzidos pelo AC-BL para o Problema1 e Problema2.

Já a Tabela 8 e a Tabela 10 mostram os resultados na comparação com outros algoritmos para o problema1 e Problema2 respectivamente.

Tabela 6: Comparação do AC-BL com o algoritmo *Social Fabric* com 20 execuções independentes para o Problema-1.

Problema-1	AC-BL	AC	Social Fabric (Topology: *Ibest*) (Reynolds; Mostafa, 2008)	Social Fabric (Topology: *Square*) (Reynolds; Mostafa, 2008)
Melhor	**0,012665236**	0,0126683	0.012665969	0.012667
Média	0.012665374	0.014887945	0.012767165	0.012759
Pior	0,012665496	0,017773	0.012962345	0.012965
Desv. Pad	7.561996420E-8	0,001681	0.000102533	0.000102

Tabela 7: Comparação do AC-BL com outros algoritmos com 30 execuções independentes para o Problema-1.

Problema-1	AC-BL	G-QPSO (2) (COELHO, 2010)	COELLO; MONTES, 2002)	CPSO (HE, WANG, 2007)	SC-ABC (BRAJEVIC; TUBA; SUBOTIC, 2011)
Melhor	0,012665246	0.012665	0.0126810	0.0126747	0.012667
Média	0.012666739	0,013524	0.0127420	0.012730	-
Pior	0,012678977	0,017759	0.012973	0.012924	-
Desv. Pad	4.302528111E-6	0,001268	5.900000e-5	5.198500e-5	-

Tabela 8: Comparação do AC-BL com outors algoritmos (melhor resultado encontrado para o Problema-1).

Variáveis	AC-BL	G-QPSO(1) (COELHO, 2010)	(COELLO; MONTES, 2002)	CPSO (HE, WANG, 2007)	SC-ABC (BRAJEVIC; TUBA; SUBOTIC, 2011)
$X_1(d)$	0,0517	0.051515	0.051989	0.051728	0.051871
$X_2(D)$	0,3564	0.352529	0.363965	0.357644	0.361108
$X_3(P)$	11,307993	11.538862	10.890522	11.244543	11.036860
g1(x)	-4,08701E-10	-4.8341E-5	-0.000013	-0.000845	-1.634E-7
g2(x)	-2,23663E-10	-3.5774E-5	-0.000021	-1.2600e-05	-4.383E-5
g3(x)	-4,053145012	-4.0455	-4.061338	-4.051300	-4.062131
g4(x)	-0,727953864	-0.73064	-0.722698	-0.727090	-0.724680
f(x)	**0,012665236**	0.012665	0.0126810	0.0126747	0.012667

Tabela 9: Comparação do AC-BL com outros algoritmos com 30 execuções independentes para o Problema-2.

Problema-2	AC-BL	AC	G-QPSO(1) (COELHO, 2010)	COELLO, MONTES, 2002)	CPSO (HE, WANG, 2007)	SC-ABC (Brajevic; Tuba; Subotic, 2011)
Melhor	6059.7177	6059.7181	6059.7208	6059.9463	6061.0777	6059.768058
Média	6059.7864	6467.9786	6440.3786	6177.2533	6147.1332	-
Pior	6059.9781	7332.8415	7544.4925	6469.3220	6363.8041	-
Desv. Pad	0.0922	345.1618	448.4711	130.9297	86.4545	-

Tabela 10: Comparação do AC-BL com outors algoritmos (melhor resultado encontrado para o Problema-2).

Design variáveis	AC-BL	AC	G-QPSO(1) (COELHO, 2010)	COELLO, MONTES, 2002)	CPSO (HE, WANG, 2007)	SC-ABC (BRAJEVIC; TUBA; SUBOTIC, 2011)
$X_1(T_S)$	0.81250	0.8125000	0.8125	0.812500	0.812500	0.81250
$X_2(T_H)$	0.43750	0.4375000	0.4375	0.437500	0.437500	0.43750
$X_3(R)$	42.098423	42.098414	42.0984	42.097398	42.091266	42.098187
$X_4(L)$	176.636897	176.636986	176.6372	176.654050	176.746500	176.640750
g1(x)	-4.361E-07	-6,098E-07	-8.7999E-7	-0.000020	-0.000139	-4.988451
g2(x)	-0.0358810	-0,03588113	-3.5881E-2	- 0.035891	-0.035949	-0.035883
g3(x)	-0,11781066	0,007605214	-0.2179	- 27.886075	-116.382700	-5.297613
g4(x)	-63.36310	-63,36301	-63.3628	- 63.345953	-63.253500	-63.359250
f(x)	6059.7177	6059.7181	6059.7208	6059.9463	6061.0777	6059.768058

Com base nos resultados de simulação e comparações acima, pode-se concluir que o AC-BL foi capaz de produzir qualidade superior, relativo aos algoritmos que utilizam os problemas de engenharia com restrição aqui apresentados. O AC-BL foi mais eficaz do que: *Gaussian quantum-behaved particle swarm optimization* G-QPSO (1) (COELHO, 2010), *co-*

evolutionary particle swarm optimization CPSO (HE; WANG, 2007) e melhor do queos resultados obtidos por Coello and Montes (2002) usando um esquema de seleção de dominação baseada na integração das restrições na função de aptidão de um AG. Estes resultados foram positivos tanto para o problema-1 como para o problema-2.

A Figura 33 e a Figura 34 mostram o desempenho do AC-BL em relação ao AC para o Problema-1 e Problema-2 respectivamente.

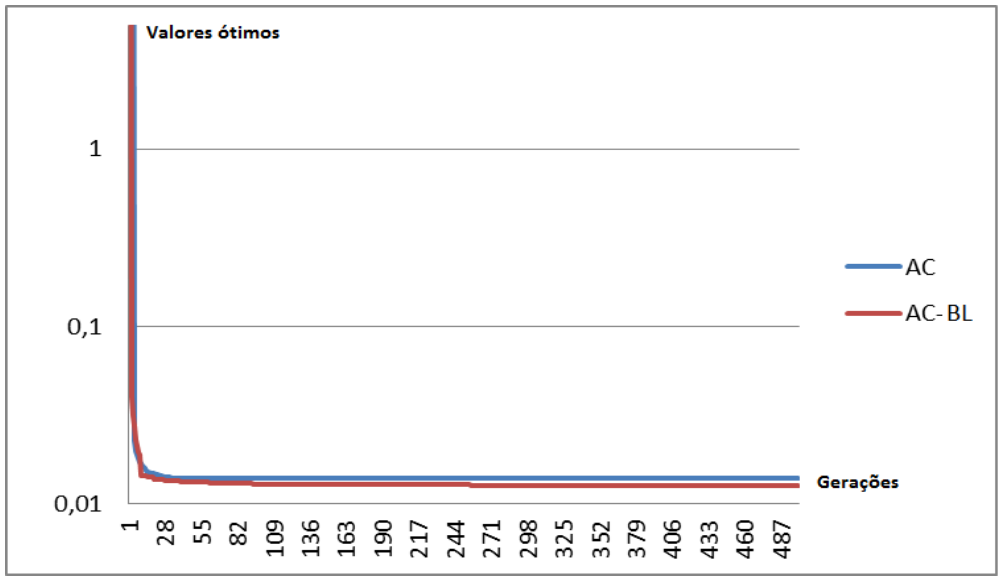


Figura 33: Desempenho do AC x AC-BL para o Problema-1.

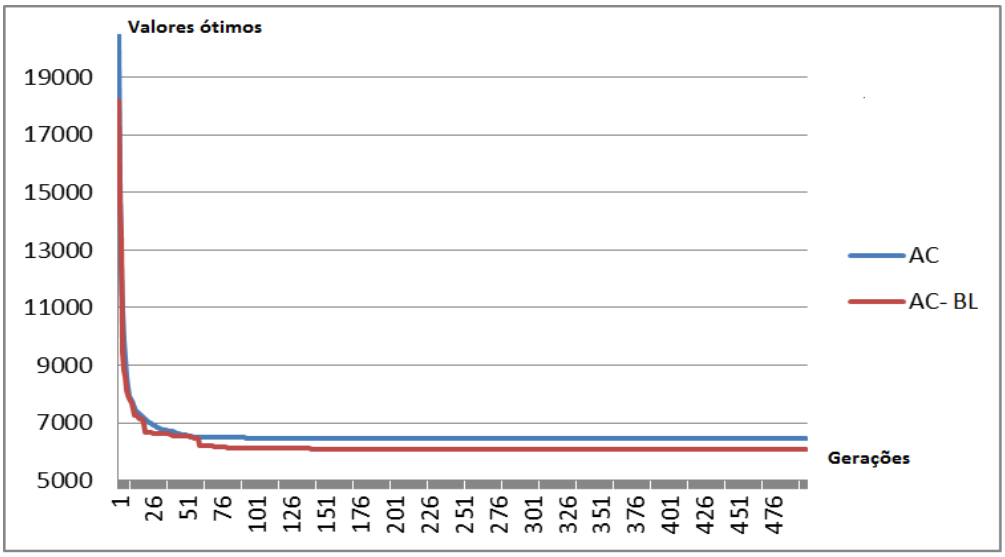


Figura 34: Desempenho do AC x AC-BL para o Problema-2.

6.3 Resultados para o Problema da Mochila Multidimensional (MKP)

Para avaliar o desempenho do algoritmo proposto AC-MI (modelo de ilhas), foi

realizada uma comparação de vários testes com algoritmos genéticos distribuídos, utilizando os mesmos problemas da mochila. Para fazer as comparações foram utilizados duas versões de algoritmos baseados em AGs distribuídos (AGUIRRE et al., 2000.): (i) AG Distribuído (denotado como *Distributed canonical GA*, DGA), e (ii) AG-SRM Distribuído (denotado como *Distributed* GA-SRM. O termo SRM significa *Self-Reproduction with Mutation* ou "auto-reprodução com mutação", e introduz a diversidade por meio da mutação que induz o aparecimento de mutações benéficas.

Para o algoritmo AC-MI existem duas versões: AC-MI_1 e AC-MI_2. A única diferença é que o AC-MI_1 tem uma taxa fixa de mutação e recombinação, enquanto AC-MI_2 tem uma taxa de mutação e recombinação aleatória. O AC padrão é o Algoritmo Cultural com uma única população.

6.3.1 DGA e DGA-SRM

O DGA trabalha com vários problemas da mochila multidimensional (problema combinatorial NP difícil), que a partir de esforços relatados anteriormente parecem ser bastante difícil para o AGS (AGUIRRE et al., 2000). Esses algoritmos foram avaliados em problemas de *benchmarks* que são retirados da literatura. O tamanho dos problemas varia de 15 a 105 objetos de 2 a 30 mochilas e podem ser encontrados em OR-Library (Beasley, 1990). Os problemas da mochila são definidos por: *problema (n, m)* onde *n* representa o número de objetos e *m* representa o número de mochilas. Cada mochila tem uma capacidade específica, assim cada objeto tem um peso específico. Por exemplo, Weing7 (105, 2) representa um MKP com 105 objetos e 2 mochilas.

Todo experimento aqui apresentado tem uma capacidade semelhante ao trabalho descrito no DGA e DGA-SRM (AGUIRRE et al., 2000), como: tamanho da população, número de avaliações da função em cada execução e um total de 100 execuções independentes. Cada execução utiliza uma semente diferente para a população aleatória inicial. Para melhorar a compreensão dos algoritmos DGA e DGA-SRM, alguns parâmetros e símbolos são apresentados:

- O tamanho máximo da população é representado por λ_{total} (fixado em 800);
- Os tamanhos populacionais de pais e filhos são representados por μ e λ, respectivamente;
- O parâmetro K representa o número de sub-populações (partições). Assim, $\lambda * \boldsymbol{K} = \lambda_{total}$ (máximo = 800);

- O parâmetro *M* representa o número de gerações entre os eventos de migração (intervalo de migração);
- O símbolo *N* representa o número de vezes que o ótimo global foi encontrado em 100 execuções;
- O símbolo τ representa um *threshold* (utilizado para o controle da taxa de sobrevivência de uma mutação normalizada).
- O símbolo *T* representa o número de avaliações da função objetivo em cada execução;
- *média* é a média das melhores soluções e DesvPd é o desvio padrão em torno da média;

Em DGA e DGA-SRM, cada subpopulação transmite uma cópia de seus melhores indivíduos *R* a todas subpopulações vizinhas. Assim, cada subpopulação em todos os eventos de migração recebe λ_m = *L* x *R* migrantes, onde L é o número de links. Quando não há migração e as subpopulações evoluem no isolamento total, os valores correspondentes a essa característica são denotados por *X* na tabela. Os resultados para problema da mochila Weing7 com DGA e DGA-SRM são mostrados na Tabela 11 (AGUIRRE et al., 2000).

Tabela 11: Os melhores resultados para weing7 (105,2) pelo DGA e DGA-SRM(λ_{total}=800; T=8x10^5).

K	λ_m/λ	**DGA**						**DGA-SRM**					
		L R	λ	**M**	**N**	**Média**	**DesvPd**	μ	λ	**M**	**N**	**Média**	**DesvPd**
8	0.10	5 2	100	5	0	**1094423.4**	433.38	50	100	80	63	1095421.44	30.84
8	0.05	5 1	100	5	0	1093284.95	733.24	50	100	100	66	1095423.58	29.84
8	0.01	1 1	100	5	0	1089452.96	1082.41	50	100	80	77	**1095430.51**	26.51
8	X	X	100	X	0	1087385.56	1729.4	50	100	X	60	1095419.80	30.86

De acordo com a Tabela 11, o melhor valor encontrado na média é igual à 1094423,4, para DGA e 1095.430,51 para DGA-SRM. A Tabela 11 também indica que DGA-SRM melhora os resultados em relação à DGA. A Tabela 12 mostra os resultados encontrados para outros problemas da mochila por DGA e DGA-SRM. A fim de simplificar os resultados mostrados na Tabela 12, os seguintes parâmetros de configuração devem ser considerados: K = 16 subpopulações e $\mu = 25$ (Aguirre et al., 2000.)

Tabela 12: Os melhores resultados para outros problemas utilizando DGA and DGA-SRM (λ_{total}= 800, T=4x10^5).

Problema (n, m)	λ_m/λ	**DGA**						**DGA-SRM (τ =0.35)**				
		LR	λ	**M**	**N**	**Média**	**DesvPd**	λ	**M**	**N**	**Média**	**DesvPd**
Petersen6 (39,5)	0.01	5 1	50	5	0	**10552.96**	19.46	50	140	77	10614.82	5.82
Petersen7 (50,5)	0.10	5 1	50	5	0	16442.39	21.52	50	40	89	16535	5.94
Sento1 (60, 30)	0.10	5 1	50	5	0	7694.22	9.97	50	40	98	7771.78	1.54
Sento2 (60, 30)	0.10	5 1	50	5	0	8661.23	19.18	50	40	84	8721.32	2.11

6.3.2 AC-MI_1

Para o algoritmo proposto (AC-MI) de vários parâmetros e símbolos são também consideradas, tais como:

- O parâmetro *P* é o tamanho da população principal;
- O parâmetro *PM* é a Probabilidade de Mutação e *PR* a Probabilidade de Recombinação.
- O número de ilhas é *K* (número de subpopulações);
- O parâmetro α é a percentagem que define o tamanho da população de cada ilha em função de *P*.
- O tamanho da subpopulação de cada ilha é SI, uma vez que SI = α *P.
- A percentagem dos melhores indivíduos no Conhecimento Situacional do espaço populacional principal é representada por SK_P e a percentagem dos melhores indivíduos no Conhecimento Situacional sobre o espaço multipopulacional é representada por SK_M.
- O parâmetro *M* representa o número de gerações entre os eventos de migração (intervalo de migração). Aqui *M* determina o intervalo de influência das subpopulações através do Conhecimento Situacional.
- O símbolo *T* representa o número de avaliações da função em cada execução;
- O símbolo *N* representa o número de vezes que o ótimo global foi encontrado nas 100 execuções.
- *média* é a média das melhores soluções e *DesvPd* é o desvio padrão em torno da *média*;
- *média de gerações* é a média de gerações onde as melhores soluções foram encontradas nos ensaios (execuções).

Para os ensaios realizados para o AC-MI_1, a seleção escolhida foi a do torneio, cujo valor é 3, a taxa de mutação (*PM*) é de 0,025 e a taxa de recombinação (*PR*) é de 0,6. As configurações do conhecimento situacional são: SKP = 0,2 e SKM = 0,5. A Tabela 13 mostra os resultados encontrados pelo AC-MI_1, cujo melhor valor médio encontrado é 1095445 (o valor ótimo) com a média de gerações de 44,49. Todos os valores encontrados alcançaram um valor ótimo. No entanto, se a *média de gerações* é baixa em relação ao total de gerações, então, isto significa que o ótimo é encontrado em poucas gerações.

Tabela 13: Os melhores resultados para Weing7 (105, 2) by AC-MI_1 (λ_{total}=800 and T=8x10^5).

P	**K**	α	**SI**	**M**	**N**	**Média de Gerações**	**Média**	**DesvPd**
400	8	0.125	50	20	100	52.9	1095445	0.0
400	8	0,125	50	05	100	**44.49**	**1095445**	0.0
100	7	1.0	100	05	100	68.87	1095445	0.0

Conforme mostrado na Tabela 13, é possível observar que o AC-MI supera o DGA-SRM para qualquer configuração, como o número de subpopulações (ilhas) e o tamanho da subpopulação. Da mesma forma, AC-MI também apresenta maior confiabilidade de convergência do que o DGA-SRM com maiores valores para *N* e *média* com menor *DesvPad*. Estes resultados mostram que o AC-MI produz um maior desempenho para todos os parâmetros utilizados.

Como já descrito anteriormente, um novo resultado *média de gerações* foi introduzido, a fim de avaliar outro tipo de desempenho cujo valor representa a média de gerações em que o valor ótimo foi encontrado para 100 execuções independentes para cada problema apresentado. Particularmente, isso ocorre quando *M* é baixo e *K* é elevado (ver resultado para *média de gerações*). Isto significa que um maior número de ilhas com pequenas subpopulações pequenas produz uma convergência melhor.

De acordo com a Tabela 13, o melhor valor encontrado na média é de 1095445 (o valor ótimo), enquanto a média das gerações é de 44,49. Isso significa um valor baixo, considerando-se que 500 gerações foram utilizadas em cada execução com T = 4x10^5. Isto representa 500 gerações com um tamanho da população igual a 800 (incluindo todas as subpopulações). A Tabela 14 mostra os resultados para os outros MKPs encontrados pelo algoritmo AC-MI_1.

Tabela 14: Os melhores resultados para outros problemas utilizando AC-MI_1 (λ_{total} = 800, T=4x10^5).

Problema (n, m)	P	K	α	SI	M	N	Média de Gerações	Média	DesvPd
Petersen6 (39,5)	400	8	0.125	50	20	100	30.22	10618.0	0.0
		4	0,25	100	05	100	**26.29**	10618.0	0.0
Petersen7 (50,5)	400	8	0.125	50	20	100	78.49	16537.0	0.0
		4	0,25	100	05	100	**71.51**	16537.0	0.0
Sento1 (60,30)	400	8	0.125	50	20	100	100.21	7772.0	0.0
		4	0,25	100	05	100	**87.44**	7772.0	0.0
Sento2 (60,30)	400	8	0.125	50	20	99	185.19	8721.81	0.099
		4	0,25	100	05	100	**166.12**	8722.0	0.0

Assim, é possível observar que AC-MI_1 supera DGA-SRM. Da mesma forma, AC-MI_1 também apresenta maior confiabilidade de convergência (valores mais elevados de *N* e *média* com menor *DesvPad*) do que DGA-SRM. Estes resultados mostram que a AC-MI_1 é capaz de encontrar o valor ótimo global para MKP, levando em consideração os resultados dos testes com sucesso de 100%. O problema que apresentou maior dificuldade foi Sento2, que em alguns casos, os valores ótimos ficaram próximos de 100%, tal como *N* = 98 e *N* = 99. Mesmo apresentando resultados de *N <100* são ainda melhores do que os resultados obtidos pelos outros algoritmos para os *benchmarks* escolhidos. Entretanto, alguns ajustes permitem que AC-MI_1 alcance o valor de N = 100 para Sento2.

6.3.3 AC-MI_2

Para os ensaios realizados para AC-MI_2 a seleção escolhida foi torneio cujo valor é 3. A taxa de mutação (PM) é um valor aleatório num intervalo específico: PM = [0,01, 0,5]. A taxa de recombinação (PR) é também um valor aleatório em um intervalo de: PR = [0,1, 0,99].

As configurações do conhecimento situacionais são: SKP = 0,2 e SKM = 0,5. Os resultados para o AC-MI_2 são apresentados na Tabela 15, que mostra os resultados para Weing7 e na Tabela 16 que mostra os resultados para os outros problemas da mochila.

Tabela 15: Os melhores resultados para o problema Weing7 (105,2) pelo AC-MI_2 (λ_{total} =800, T=8x105).

P	K	α	SI	M	N	Média de Gerações	Média	DesvPd
400	8	0.125	50	20	100	**70.48**	1095445	0.0
400	8	0,125	50	05	100	72.72	1095445	0.0
100	7	1.0	100	05	100	107.11	1095445	0.0

Tabela 16: Os Melhores resultados para outros problemasc om o AC-MI_2 (λ_{total} = 800, T=4x105).

Problema (n, m)	P	K	α	SI	M	N	Média de Gerações	Média	DesvPd
Petersen6 (39,5)	400	8	0.125	50	20	100	37.89	10618.0	0.0
		4	0,25	100	05	100	**33.39**	10618.0	0.0
Petersen7(50,5)	400	8	0.125	50	20	100	81.46	16537.0	0.0
		4	0,25	100	05	100	**74.38**	16537.0	0.0
Sento1(60,30)	400	8	0,25	50	20	98	**112.55**	7771.75	1.7717
		4	0,25	100	05	100	126.46	7772.0	0.0
Sento2(60,30)	400	8	0.125	50	20	71	183.35	8720.0	3.7199
		4	0,25	100	05	88	**173.53**	8721.38	2.1732

A aplicação da taxa de mutação aleatória para e recombinação em AC-MI_2 não produz resultados satisfatórios em comparação com AC-MI_1, como é mostrado na Tabela 16. Além disso, a *média de gerações* do algoritmo AC-MI_2 é maior do que AC-MI_1 para todos os problemas da mochila. No entanto, em comparação com AC-MI_1, existem poucas diferenças nos resultados para Weing7 como é mostrado na Tabela 13 e Tabela 15.

6.3.4 Algoritmo Cultural Padrão (AC)

Para o AC, foi utlizada a mesma configuração, tais como: torneio = 3, PM = 0,025 e PR = 0,6. A configuração do conhecimento situacional é igual a 0,2 (SKP = 0,2). Os ensaios aqui apresentados consistem em 100 execuções independentes e cada execução utiliza uma semente (*seed*) diferente para a população inicial aleatória. A Tabela 17 mostra os resultados do Algoritmo Cultural Padrão (AC), que utiliza uma única população. Segundo os resultados, o AC atinge uma média de valores ótimos em 100 execuções somente para Sento1 e Weing7. No entanto, os resultados de AC para Petersen6, Pertersen7 e Sento2 superaram os resultados apresentados por DGA-SRM.

Tabela 17: Os melhores resultados para todos os *benchmarks* da mochila pelo AC ($T=4x10^5$).

Problema (, m)	P	N	Média	DesvPd.
Petersen6 (39,5)	800	97	10617.58	2.4002
Petersen7 (50,5)	800	81	16533.7	6.8703
Sento1 (60,30)	800	100	7772.0	0.0
Sento2 (60,30)	800	82	8721.14	2.4495
Weing7 (105,2)	800	100	1095445.0	0.0

6.4 Resultados para o Problema do Caixeiro Viajante (PCV)

Os testes realizados para o PCV tiveram como objetivo avaliar o comportamento do algoritmo proposto em relação a testes de *benchmarks*, levando-se em consideração o número de cidades. Muitos algoritmos têm um bom comportamento com um número de até 100 cidades. Passando desse número muitos algoritmos passam a ter dificuldades em achar o valor ótimo. Quando o número de cidades é bem alto o problema passa a ser tratado como PCV em larga escala.

Como visto anteriormente o algoritmo memético aqui aplicado utiliza o mesmo princípio do AC-BL, ou seja AC com Simulated Annealling com heurísticas 2-opt e 3 opt (AC-BL para o PCV).

Foram realizados testes com três instâncias de cidades menores ou iguais a 100. Também foram utilizadas outras duas instâncias com um número de cidades iguais a 442 (PCB442) e 532 (Att532). Essas instâncias e outras tais como: kroA10 e eil76, são provenientes da biblioteca TSPLIB (GERHARD, 2001) .

Os resultados para instâncias maiores que 100 foram comparados com os trabalhos de HE, LIU e QIU (2006), que apresenta um algoritmo de Busca Paralela Tabu (*Parallel Tabu Search* - PTS) com resultados para PCB442 e att532. Além disso, seu trabalho apresenta comparações dessas instancias para outros algoritmos.

A Tabela 18 mostra os resultados do algoritmo proposto (AC-BL para o PCV) com instâncias menores ou iguais a 100 cidades para 50 execuções do algoritmo.

Tabela 18:Resultados para instâncias com cidades menores que 100 (AC-BL para o PCV).

Problema	Ótimo	Melhor	Média	GAP % (Erro) Melhor	GAP% (Erro) Média
26 Capitais Brasileiras	2020	2020	2020	0,0	0,0
KROA100	21282	21282	21282	0,0	0.0
EIL76	538	538	538	0,0	0,0

Os resultados para cidades maiores que 100 são mostrados na Tabela 19.

Tabela 19 : Resultados para instâncias com cidades menores que 100 (AC-BL para o PCV)

Problema	Ótimo	Melhor	Média	GAP % (Erro) Melhor	GAP% (Erro) Média
KROA150	26524	26524	26524	0,0	0,0
PCB442	50778	50778	50854,1	0,0	0.14
ATT532	27686	27705	27717	0,07	0.12

Os erros calculados são referentes à média dos melhores valores ótimos encontrados e em relação ao melhor ótimo encontrado:

$$Erro(media) = \frac{media - otimo}{otimo} x100 \qquad Erro\ (melhor\ ) = \frac{melhor\ \ - otimo}{otimo} x100$$

Os resultados para a o Algoritmo *Parallel Tabu Search* são mostrados na Tabela 20.

Tabela 20:Algoritmo de Busca Paralela Tabu .

Problema	Ótimo	Melhor	Média	GAP % (Erro) Melhor	GAP% (Erro) Média
PCB442	50778	50837	51601,7	0,12	1,62
Aatt532	27686	28753	29394,3	3,85	6,17

Tabela 21 Comparação com outros algoritmos.

Problema	Ótimo	GAP % (Erro) Melhor			
		AC-BL (2-opt e 3-opt) Proposto	**Algoritmo Genético** (CHENG *ET AL.*, 2002)	**Algoritmo Paralelo 2-opt** (VERHOEVEN; AARTS; SWINKELS, 2005)	**Busca Tabu Paralela** (HE, LIU e QIU, 2006)
PCB442	50778	0,0	12,76	----	0,12
Aatt532	27686	0,07	16,18	6,5	3,85

Os resultados foram satisfatórios para cidades menores ou iguais a 100. Além do mais, o tempo de processamento para essas instâncias chegou a alguns segundos. Já os resultados para as instâncias com cidades maiores que 400 foram razoáveis. Apesar disso o algoritmo encontrou o ótimo para a instância **pcb442** e chegou a um resultado próximo para a instância **att532**. Entretanto, vale ressaltar que cada execução para essas duas últimas instâncias levaram em torno de 3 horas (computador Core 2 duo 1.8 GHz, 3 GB RAM) o que é esperado devido ao tamanho do problema.

7 Conclusão

A abordagem de diferentes problemas de otimização mostrados no decorrer desse trabalho, permitiram entender algumas nuances no processo evolucionário dos AGs e ACs para tais tipos de problemas. Também mostrou a possibilidade de hibridização desses algoritmos seguindo a abordagem memética, no sentido de melhorar o desempenho no processo de otimização. As análises provenientes de diversos ensaios contribuiram para o desenvolvimento dos algoritmos até chegar à versão aqui apresentada. Assim, foram criadas novas alternativas para tais problemas através da utilização de um mecanismo dual de herança oferecido pelo AC e sua hibridização com algoritmos de busca local para que pudesse escapar de possíveis ótimos locais encontrados pelo caminho. Os algoritmos de busca na vizinhança desempenharam um papel muito importante durante o processo de busca local, comprovados pelos resultados apresentados.

Nesse sentido, algumas considerações importantes sobre o problema de otimização global envolvendo funções multimodais, otimização de problemas restritos, problema da mochila e o problema do caixeiro viajante (PCV), puderam ser feitas, levando-se em conta os resultados obtidos durante os ensaios. No geral, observou-se que a utilização do ACs melhoram a convergência e a velocidade de busca nesses problemas. Entretanto, na maioria dos ensaios os ACs por si só não foram suficientes para encontrar valores ótimos previamente bem conhecidos em vários ensaios.

Nas funções multimodais existe o problema da quantidade dos ótimos locais e como eles estão distribuídos ao longo do espaço de busca, características intrínsecas que podem dificultar a busca do valor ótimo global. Nos problemas restritos, além do tipo da função objetivo a ser otimizada, existe o problema das restrições que devem ser satisfeitas. Por esta razão partiu-se para a escolha de uma solução alternativa que pudesse ser adicionada com base em heurísticas em um processo híbrido, no sentido de permitir que as soluções pudessem escapar de mínimos ou máximos locais.

No problema da mochila também foi possível observar o mesmo comportamento envolvendo o AC padrão. Ou seja, ocorreram situações em que não foi possivel escapar de ótimos locais em alguns ensaios realizados. Na busca de soluções para tal problema, observou-se que a inclusão de multipopulações no algoritmo melhorou a confiabilidade da convergência e a velocidade de busca.

Problema similar é encontrado no problema do caixeiro viajante (PCV). No PCV foi

inserida uma heurística de busca local no sentido de melhorar os resultados e reduzir o custo computacional, principalmente em ocasiões onde se utiliza um número superior a 100 cidades.

O uso das alternativas apresentadas permitiu um melhora significante nos resultados obtidos, possibilitando alcançar valores ótimos tão bons e/ou melhores dos que outras metaheurísticas utilizadas na literatura para (pelo menos) os mesmos problemas aqui abordados.

Os resultados positivos apóiam a idéia de que a hibridização de algoritmos e o uso de multipopulações nos Algoritmos Culturais é uma abordagem altamente desejável para direcionar o espaço de busca para encontrar soluções mais próximas do valor ótimo.

Futuramente pretende-se investigar o comportamento dos ACs com outras heurísticas de busca local, bem como a combinação de duas ou mais metaheurísticas no sentido de obter resultados cada vez melhores em um tempo de processamento e do número de avaliações de *fitnes* aceitável de acordo com cada problema a ser abordado.

8 Referências Bibliográficas

ABRAHAM A., NEDJAH N., MOURELLE L. **Evolutionary Computation: From Genetic Algorithms to Genetic Programming**, Nadia Nedjah et al. (Eds.), Studies in Computational Intelligence, Springer Verlag Germany, ISBN: 3-540-29849-5, pp. 1-20, 2006.

ACAMPORA G., GAETA M., LOIA V., **Exploring e-learning knowledge through ontological memetic agents**, IEEE Comput. Intell. Mag., vol. 5, no. 2, pp. 66–77, May 2010.

ADRA S. **Optimisation techniques for gas turbine engine control systems**. Master's thesis, Department of Computer Science, The University of Sheffield, UK, 27 August 2003.

AGUIRRE H., Tanaka K., Sugimura T., and Oshita S., **Improved Distributed Genetic Algorithm with Cooperative-Competitive Genetic Operators**, In Proceedings IEEE International Conference on Systems, Man, and Cybernetics (SMC 2000), pp. 3816-3822, Oct. 2000.

AHMED Z., YOUNAS I. **A Dynamic Programming based GA for 0-1 Modified Knapsack Problem.** International Journal of Computer Applications (0975 – 8887) Volume 16– No.7, February 2011.

AKBARI R., ZIARATI K. **Combination of Particle Swarm Optimization and Stochastic Local Search for Multimodal Function Optimization**, in Proc. IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application (PACIIA '08), pp. 388-392, 2008.

ALAMI J., BERNAMEUR L. EL IMRANI A. **Fuzzy Clustering Based Parallel Cultural Algorithm**. International Journal of Soft Computing, 2(4): 562-571, 2007.

ALI M. Z. A., REYNOLDS R. G., ALI R. **Enhancing Cultural Learning under Environmental Variability Using Layered Heterogeneous Sociometry-Based Networks**. IAT 2010: 69-74, 2010.

ALOTTO P., COELHO L. S. **Electromagnetic Optimization Using a Cultural Self-Organizing Migrating Algorithm Approach Based on Normative Knowledge**, IEEE Trans. on Magnetics,45(3): 1446 -1449, 2009.

ALREFAEI M. H., ANDRADOTTIR S. **A simulated annealing algorithm with constant temperature for discrete stochastic optimization**. Management Science, 45, 748–764, 1999.

ANDRÉASSON N., EVGRAFOV A., PATRIKSSON M. **An Introduction to Optimization: Foundations and Fundamental Algorithms**, 2005. Disponível em: http://www.math.chalmers.se/Math/Grundutb/CTH/tma947/0405/kompendium_sub.pdf. Acessado em fevereiro de 2012.

ANGELINE P. A., **Adaptive and self-adaptive evolutionary computation**, in Computation Intelligence, M. Palaniswarni et al., Eds. Piscataway,NJ: IEEE Press, pp. 152-163, 1995.

ASTOLFI A. **Optimization: An introduction,** 2004, Disponivel em http://cap.ee.imperial.ac.uk/~astolfi/Courses/outs/Optim.pdf , acessado em fevereiro de 2012.

BACK T, FOGEL D.B, AND MICHALEWICZ Z. **Handbook of Evolutionary Computation**, IOP Press, NY, (eds.), 1997.

BEASLEY, D.; BULL, D. R.; MARTIN, R. R. **An Overview of Genetic Algorithms: Part 1, Fundamentals. Inter-University Committee on Computing**. University Computing, UK, 1993.

BECERRA, R. L. **Algoritmos Culturales Aplicados a Optimizacíon con Restricciones y Optimizacíon Multiobjetivo**. Tese (Doutorado) - Instituto Politécnico Nacional do México, 2002.

BECERRA, R.L., COELLO C.A. **Culturizing Differential Evolution for Constrained Optimization**. ENC 2004: 304-311, 2004.

BECERRA, R. L.; COELLO, C. A. C. **Optimization with constraints using a cultured differential evolution approach**. In: Genetic and Evolutionary Computation Conference. [S.l.: s.n.], p. 27-34, 2005.

BELISLE C.J.P. **Convergence theorems for a class of simulated annealing algorithms on Rd**. Journal of Applied Probability, 29, 885–895, 1992.

BEST, M.L. **How Culture Can Guide Evolution: An Inquiry into Gene/Meme**. Enhancement and Opposition. International Society of Adaptive Behavior, 2007.

BEYER, H.-G., SENDHOFF B. Functions with noise-induced multimodality: a test for evolutionary robust Optimization-properties and performance analysis. IEEE Trans. Evolutionary Computation 10(5): 507-526, 2006.

BIN P. **Knowledge and population swarms in cultural algorithms for dynamic environments**, Ph.D Thesis. Wayne State University, Detroit, 2005.

BLACKMORE, S. **O Poder do Meme**. Último acesso em 04 de novenbro de 2007. Disponível em http://www.geocities.com/realidadebr/textos/meme.htm.

BLUM C.; ROLI A. **Metaheuristics in combinatorial optimization: Overview and conceptual comparison**. ACM Comput. Surv., Vol 35, N. 3, September , pp. 268-308, 2003.

BLUM C, PUCHINGER J., GÜNTHER R., ROLI A.: **Hybrid metaheuristics in combinatorial optimization: A survey**. Appl. Soft Comput. 11(6): 4135-4151, 2011.

BOHACHEVSKY, I.O., JOHNSON M.E., STEIN M.L. **Generalized simulated annealing for function optimization**. Technometrics, 28, 209–217, 1986.

BOOMSMA, W. **Adaptive Operator Scheduling in Evolutionary Algorithms**. Master's Thesis. Department of Computer Science, University of Aarhus, December 2003.

BRAJEVIC I., TUBA M., SUBOTIC M. **Performance of the improved artificial bee**

colony algorithm on standard engineering constrained problems, International Journal of Mathematics and Computers in Simulation, Vol. 5, Issue 2 , pp. 135-143, 2011.

BRAUN, H. **On solving traveling salesman problems by genetic algorithms**. In H.-P. Schwefel and R. Manner, editors, Parallel Problem Solving from Nature - Proceedings of1st Workshop, PPSN 1, volume 496 of Lecture Notes in Computer Science, pages 129-133, Dortmund, Germany: Springer-Verlag, Berlin, Germany, 1991.

CANTU-PAZ E. **A Survey of Parallel Genetic Algorithms**, Calculateurs parallèles, réseaux et systèmes répartis 10, 141-171, Hermès, Paris, 1998.

CANTU-PAZ E. **Migration policies and takeover times in parallel Genetic Algorithms**. Proceedings of the International Conference on Genetic and Evolutionary Computation, San Francisco, CA, pp.775-779, 1999.

CANTU-PAZ E. **Efficient and Accurate Parallel Genetic Algorithms**. Nor-well, MA, USA: Kluwer Academic Publishers, 2000.

CARSON, T.; IMPAGLIAZZO R. **Hill-climbing finds random planted bisections**, in: Proceedings of the ACM-SIAM Symposium on Discrete Algorithms, SODA 2001, pp. 903-909, 2001.

CARVALHO, A. B., BARRIVEIRA R. **Algoritmo Cultural para o Problema do Caixeiro Viajante** Relatório Técnico (UFPR) disponível em http://www.inf.ufpr.br/aurora/disciplinas/topicosia2/downloads/trabalhos/CulturaisTSP.pdf, acessado em outubro de 2011.

CHAMBERS, L. D. **Practical Handbook of Genetic Algorithms. Complex Coding Systems.** Volume III. CRC Press, New York. 1999.

CHENG, C.-H., LEE, W.-K., WONG, K.-F.,. A Genetic Algorithm-Based Clustering Approach for Database Partitioning. IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS — PART C: APPLICATIONS AND REVIEWS, Vol. 32, No. 3, pp.215-230, 2002.

CHUNG C.-J., REYNOLDS R., **A testbed for solving optimization problems using culture algorithms**. in Evolutionary Programming V: Proceedings of the Fifth Annual Conference on Evolutionary Programming. Cambridge, Massachusetts: MIT Press, 1996.

COELHO, L. S. **Fundamentos, Potencialidades e Aplicações de Algoritmos Evolutivos**. [S.l.]: SBMAC, 2003.

COELHO L.S., MARIANI V.C. **An efficient particle swarm optimization approach based on cultural algorithm applied to mechanical design**. In: IEEE conference of evolutionary computation, pp 1099–1104, 2006.

COELHO, L. S.: **Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems**. Expert Syst. Appl. 37(2): 1676-1683, 2010.

COELLO C., BECERRA, L. **Evolutionary multi-objective optimization using a cultural**

algorithm. IEEE Swarm Intelligence Symposium, 2003.
COELLO, C. A.; BECERRA, R. L. **Adding knowledge and efficient data structures to evolutionary programming: A cultural algorithm for constrained optimization**. In: LANGDON E.CANTÚ-PAZ, K. M. R. R. D. D. R. P. K. B. V. H. G. R. J. W. L. B. M. A. P. A. S. J. F. M. E. B. W.; N.JONOSKA (Ed.). Proceedings of the Genetic and Evolutionary Computation Conference. San Francisco, California: Morgan Kaufmann Publishers, p. 201-209, 2002.

COELLO C.A.C., MONTES E.M., **Constraint-handling in genetic algorithms through the use of dominance-based tournament selection**.Advanced Engineering Informatics 16, 193–203, 2002.

COTTA C., TROYA J.M. A Hybrid Genetic Algorithm for the 0-1 Multiple Knapsack Problem. Artificial Neural Networks and Genetic algorithms 3, Springer-Verlag, Berlin (1998), 250-254.

CRUZ A.V., PACHECO M.C, VELLASCO M et al. **Cultural operators for a quantum-inspired evolutionary algorithm applied to numerical optimization problems**. Lect Notes Comput Sci 3562:1–10, 2005.

CURRAN, D. **An Empirical Analysis of Cultural Learning: Examining Fitness, Diversity and Changing Environments in Populations of Game-Playing Neural Network Agents**. PhD Dissertation, National University of Ireland, Galway, 2006.

DANCHIN E., GIRALDEAU L., VALONE T., WAGNER R. **Public information: from nosy neighbors to cultural evolution**. Science, 305, 487-491, 2004.

DASGUPTA D., MICHALEWICZ Z. **Evolutionary Algorithms in Engineering Applications**. Springer-Verlag, 1997.

DAWKINS, Richard. **O gene egoísta**. Ed. Itatiaia Ltda, 2001.

DE JONG, K. A.. **An Analysis of the Behavior of a class of Genetic Adaptive Systems**. PhD thesis, University of Michigan, Ann Arbour. Department of Computer and Communication Sciences, 1975.

DELIN L., LIXIAO Z., ZHIHUI X. **Heuristic Simulated Annealing Genetic Algorithm for Traveling Salesman Problem**, *Computer Science & Education (ICCSE), 2011 6th International Conference on,* On page(s): 260 - 264, Volume: Issue: , 3-5 Aug. 2011

DUAN H. B., WANG D. B., YU X. F. **MAX-MIN meeting ant colony algorithm based on cloud model theory and niche ideology**, Journal of Jilin University (Engineering and Technology Edition), 36(5): 803-808, 2006

DUGAN N., ERKOÇ S. Genetic Algorithms in Application to the Geometry Optimization of Nanoparticles. Algorithms 2(1): 410-428, 2009.

EBERHART, R.; SIMPSON, P.; DOBBINS, R. **Computational Intelligence PC Tools: an indispensable resource for the latest in fuzzy logic, neural network and evolutionary**

computing. American Press Inc., 1996.

EIBEN A.E., SMITH, J. E.. **Introduction to Evolutionary Computing**. Berlin: Springer, p. 173-188, 2003.

EL-MIHOUB T. A., HOPGOOD A., NOLLE L., BATTERSBY A. **Hybrid genetic algorithms: a review**, Engineering Letters, 13:2, 2006.

ENGELBRECHT A. P. **Computational Intelligence: An Introduction**, 2nd edition, Wiley, ISBN 978-0-470-03561-0, 2007.

FABIAN, V. **Simulated annealing simulated. Computers and Mathematics with Applications**, 33, 81–94, 1997.

FEO T.A., RESENDE M.G.C. **Greedy randomized adaptive search procedures**. Journal of Global Optimization, 6:109–133, 1995.

FOGEL, D. B. **Evolutionary Computation: Toward a New Philosophy of Machine Intelligence**. Piscataway, NJ: IEEE Press, 1995.

GALLARDO J.E., COTTA C., FERNÁNDEZ A.J. **On the hybridization of memetic algorithms with branch-and-bound techniques**, IEEE Transactions on Systems Man, and Cybernetics—Part B 37 (1) 77–83, 2007.

GERHARD, R. **TSPLIB**, http://www.iwr.uniheidelberg.de/groups/c-omopt/software /TSPLIB95/, 2001.

GLOVER, F. **Tabu search and adaptive memory programming advances, applications and challenges**. In: Barr, R., Helgason, R., Kennington, J. (eds.) Interfaces in Computer Science and Operations Research, vol. 7, pp. 1–75. Springer, Heidelberg, 1997.

GLOVER, F., KOCHENBERGER, G. A. **Handbook of Metaheuristics**. Kluwer Academic Publishers, Boston, 2003.

GOEFFON A., RICHER J.M., HAO, J.K. **Progressive tree neighborhood applied to the maximum parsimony problem**. IEEE/ACM Transactions on Computational Biology and Bioinformatics 5(1), 136–145, 2008.

GOLDBERG D. E.; LINGLE R. **Alleles, loci, and the traveling salesman problem** in Proc. Int. Conf. Genetic Algorithms and Their Appl., pp. 154-159, 1985.

GOLDBERG, D. **Genetic Algorithms in Search, Optimization, and Machine Learning**. Addison-Wesley, 1989.

GREFENSTETTE J. J., GOPAL R., ROSMAITA B., AND D. VANGUCHT, **Genetic algorithms for the traveling salesman problem** in Proc. Int. Conf. GeneticAlgorithms and Their Appl., pp. 160-168, 1985.

GUO Y-N., GONG D.-W. **Extraction and utilization about knowledge in hierarchical interactive genetic algorithms**. Control Decis 22(12):1329–1335, 2007.

GUO Y.-N., CHENG J., CAO Y.-Y., LIN Y. **A novel multi-population cultural algorithm adopting knowledge migration**. Soft Comput. 15(5): 897-905, 2011.

GUSTAFSON, S. and E.K. Burke, **The speciating island model: an. alternative parallel evolutionary algorithm**, Journal of Parallel and Distributed Computing, vol. 66, no. 8,pp. 1025-1036,2006.

HAMIEZ J.-P., ROBET J., HAO J.-K. **A tabu search algorithm with direct representation for strip packing**. In: Cotta, C., Cowling, P. (eds.) EvoCOP 2009. LNCS, vol. 5482, pp. 61–72. Springer, Heidelberg, 2009.

HANDOKO S.D., KWOH C.K., ONG Y.S. **Using Classification for Constrained Memetic Algorithm: A New Paradigm**, in Proceedings of IEEE International Conference on Systems, Man and Cybernetics, pp.547–552, Oct 2008.

HANSEN P., MLADENOVIC N. **An introduction to variable neighborhood search**. In: Voss, S., et al. (eds.) Metaheuristics, Advances and Trends in Local Search Paradigms for Optimization, pp. 433–458. Kluwer Academic Publishers, Dordrecht, 1999.

HART W. E. **Adaptive global optimization with local search**, Ph.D. dissertation, Dept. Comput. Sci. Eng., Univ. California, San Diego, 1994.

HAUPT R. L; HAUPT S. E. **Practical Genetic Algorithms**. Second Edition John Wiley & Sons, Inc., 2004.

HE, Y., LIU G., QIU, Y. **A Parallel Tabu Search Algorithm Based on Partitioning Principle for TSPs**. Proceedings of IJCSNS International Journal of Computer Science and Network Security, VOL.6 No.8A, August, 2006.

HE, Y.-C., LIU, K.-Q., **A modified particle swarm optimization for solving global optimization problems**, Proceedings of the Fifth International Conference on Machine Learning and Cybernetics, (ICMLC 2006), Dalian, China, art. no. 4028423, pp. 2173-2177, IEEE 2006.

HE Q., WANG L. **An effective co-evolutionary particle swarm optimization for constrained engineering design problem**. Engineering Applications of Articial Intelligence, Elsevier, 20 (1):88-99, 2007.

HENDERSON D., JACOBSON S.H., JOHNSON A.W. **The theory and practice of simulated annealing**. In: Glover, F. and Kochenberger, G. (eds), *Handbook on Metaheuristics* , pp. 287-319. Kluwer Academic Publishers, Norwell, MA, 2003.

HERNAN E. AGUIRRE, KIYOSHI TANAKA, TATSUO SUGIMURA, **Cooperative Model for Genetic Operators to Improve GAs**, In Proceedings IEEE International Conference on Information Intelligence and Systems (ICIIS'99), pp.98-106, 1999.

HILL, R.R., HIREMATH, C. **Improving genetic algorithm convergence using problem structure and domain knowledge in multidimensional knapsack problems**, Int. J. Operational Research, Vol. 1, Nos. 1/2, pp.145–159, 2005.

HO N.B., TAY J.C. **GENACE: an effective cultural algorithm for solving the flexible job-shop problem**. In: Proceeding of 2004 congress on evolutionary computation, pp 1759–1766, 2004.

HONG T.-P., LIN W.-Y., LIU S.-M., LIN J.-H. **Dynamically Adjusting Migration Rates for Multi-Population Genetic Algorithms**. JACIII 11(4): 410-415, 2007

HOLLAND, John H.. **Adaptation in Natural and Artificial Systems**. University of Michigan Press, 1975.

HUANG H.-Y., GU X.-S., LIU M.-D. **Research on cultural algorithm for solving nonlinear constrained optimization**. Acta Automat Sin 33(10):1115–112, 2007.

IACOBAN, R.; REYNOLDS, R.; BREWSTER, J. **Cultural swarms: assessing the impact of culture on social interaction and problem solving**. In: IEEE Swarm Intelligence Symposium. [S.l.: s.n.], p. 212-219, 2003.

IACOBAN, R.; REYNOLDS, R.; BREWSTER, J. **Cultural swarms: modeling the impact of culture on social interaction and problem solving**. In: IEEE Swarm Intelligence Symposium. [S.l.: s.n.], p. 205-211, 2003.

IBARAKI T. **Combination with local search**, in **Handbook of Genetic Algorithms**, T. Back, D. Fogel, and Z. Michalewicz, Eds. Intitute of Physics Publishing and Oxford University Press, pp. D3.2–1–D3.2:5, 1997

IDOUMGHAR L., MELKEMI M., SCHOTT R., IDRISSI-AOUAD M. **Hybrid PSO-SA Type Algorithms for Multimodal Function Optimization and Reducing Energy Consumption in Embedded Systems**. Applied Comp. Int. Soft Computing 2011, 2011.

JAGTAP S. B., PANI S. K., SHINDE G. **A Hybrid Parallel Multi-Objective Genetic Algorithm for 0/1 Knapsack Problem**. JSEA 4(5): 316-319, 2011.

JIAO L. C., WANG L. A., **Novel Genetic Algorithm Based on Immunity**", IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans, Sept., 30(5): 552-561, 2000.

JIAO L., GONG M., WANG S., HOU B., ZHENG Z., WU Q. **Natural and remote sensing image segmentation using memetic computing**, IEEE Comput. Intell. Mag., vol. 5, no. 2, pp. 78–91, May 2010.

JIN, X.; REYNOLDS, R. G. **Using knowledge-based evolutionary computation to solve nonlinear constraint optimization problems: a cultural algorithm approach**. IEEE, p. 1672 - 1678, 1999.

JIN, X.; REYNOLDS, R. G. **Using knowledge-based system with hierarchical architecture to guide the search of evolutionary computation**. Tools with Artificial Intelligence - 11th IEEE International Conference, p. 29-36, 1999.

JOHNSON D.,MCGEOCH L. **The Traveling Salesman Problem: A Case Study in Local**

Optimization. Draft of November 20, 1995. To appear as a chapter in the book Local Search in Combinatorial Optimization, E. H. L. Aarts and J. K. Lenstra(eds.), John Wiley and Sons, New York., 1995.

KALLEL L., NAUDTS B., REEVES C. **Properties of fitness functions and search landscapes**, in Theoretical Aspects of Evolutionary Computing, L. Kallel, B. Naudts and A. Rogers, eds., Springer, Berlin, Heidelberg, New York, 2001.

KARABOGA D., GORKEMLI B. **A Combinatorial Artificial Bee Colony Algorithm for Traveling Salesman Problem**. INISTA 2011: International Symposium on Innovations in Intelligent Systems and Applications, 15-18, Istanbul, Turkey, June 2011.

KIM Y., CHO S.-B. **A Hybrid Cultural Algorithm with Local Search for Traveling Salesman Problem**, *Computational Intelligence in Robotics and Automation (CIRA), 2009 IEEE International Symposium on,* On page(s): 188 - 192, Volume: Issue: , 15-18 Dec. 2009

KINNAIRD-HEETHER L., REYNOLDS R.G. **Survival of the Fastest: Using Cultural Algorithms to Optimize the Design of a Controller for a 3D Racing Game**. Presentation at Wayne State University,2009.

KNYSH D., KUREICHIK V. **Parallel genetic algorithms: a survey and problem state of the art**, Journal of Computer and Systems Sciences International, vol. 49, pp. 579-589, 2010.

KOBTI Z., SNOWDON A. W., RAHAMAN S., DUNLOP T., KENT R. D. **A Cultural Algorithm to Guide Driver Learning in Applying Child Vehicle Safety Restraint**, IEEE Congress on Evolutionary Computation, pp 1111- 1118, 2006.

KOULAMAS C., ANTONY S.R., JAEN R. **A survey of simulated annealing applications to operations-research problems**. OMEGA-International Journal of Management Science, 22 , 41–56, 1994.

KRASNOGOR N., **Studies in the Theory and Design Space of Memetic Algorithms**, PhD thesis, University of the West of England, Bristol, U.K, 2002.

KRASNOGOR N., SMITH J. **A tutorial for competent memetic algorithms: Model, taxonomy and design issues**, IEEE Trans. Evol. Comput. ,vol.9, no. 5, pp. 474–488, Oct. 2005.

KRASNOGOR N., ARAGON A., PACHECO J. **Memetic Algorithms**, Book chapter, In Metaheuristics in Neural Networks Learning, (Eds. Rafael Ma rti and Enrique Alba), Kluwer, 2006.

KU K. MAK M. **Empirical analysis of the factors that affect the Baldwin Effect,** PPSN-V: Parallel Problem Solving From Nature, Proceedings 1998. Lecture Notes in Computer Science, pp. 481–490, 1998.

KUMAR M., HUSIAN M., UPRETI N., GUPTA D. Genetic algorithm: Review and Application. International Journal of Information Technology and Knowledge Management, July-December 2010, Volume 2, No. 2, pp. 451-454, 2010

LAND M. **Evolutionary algorithms with local search for combinatorial optimization**, Ph.D. Thesis, University of California, San Diego, 1998.

LE T. V., **A fuzzy evolutionary approach to constrained optimization problems**. in Proceedings of the second IEEE Conference on Evolutionary Computation. Perth: IEEE, November 1995, pp. 274–278.

LE T. V. **A fuzzy evolutionary approach to constrained optimization problems**, in Proceedings of the second IEEE Conference on Evolutionary Computation. Perth: IEEE, November 1995, pp. 274–278, 1995.

LIANG, K. H., Yao X. AND NEWTON ,C. S. **Adapting self-adaptive parameters in evolutionary algorithms**, Applied Intelligence, 15(3):171-180, November/December, 2001.

LIN S., KERNIGHAN B. **An Effective Heuristic Algorithm for the Traveling Salesman Problem**. Operations Research 21, 498–516, 1973.

LIN, W.Y., Hong, T.P. and Liu, S.M. **On adapting migration parameters for multi-population genetic algorithms**, in IEEE International Conference on Systems, Man and Cybernetics 2004, 10-13 2004, vol. 6, pp. 5731-5735, 2004.

LIN J.-Y., CHEN Y.-P. **Analysis on the Collaboration Between Global Search and Local Search in Memetic Computation**. IEEE Transactions on Evolutionary Computation, vol. 15, no. 5, october 2011.

LINDEN, R. **Algoritmos Genéticos**. 1. ed. Rio de Janeiro: Brasport,. v. 1. 372 p., 2006.

LING S.H.,.et al. **Hybrid particle swarm optimization with wavelet mutation and its industrial applications**, IEEE Trans. Sys., Man, and Cybernetics, Part B, vol. 38, no.3, pp. 743–763, Jun. 2008.

LOCATELLI, M. **Convergence properties of simulated annealing for continuous global optimization**. Journal of Applied Probability, 33, 1127–1140, 1996.

LOCATELLI, M. **Simulated annealing algorithms for continuous global optimization: convergence conditions**. Journal of Optimization Theory and Applications, 104, 121–133, 2000.

LOZANO, M., HERRERA, F., KRASNOGOR, N., MOLINA, D. **Real-coded memetic algorithms with crossover hill-climbing**. Evolutionary Computation 12(3), 273–302, 2004.

LU Z., HAO J.K., GLOVER F.**Neighborhood analysis: a case study on curriculum-based course timetabling**. Journal of Heuristics 17(2), 97–118 (2010)

MA W., ZHU X., ZHU Q. **Research on cultural continuous ant colony optimization algorithm**. Appl Res Comput 26(7):2442–2449, 2009

MARQUES, F. P. **O Problema da Mochila Compartimentada**, Dissertação de mestrado, Ciência da Computação e Matemática Computacional, USP, São Carlos-SP, 2000.

MATAI R., SINGH S., MITTAL M.L. Traveling Salesman Problem: An Overview of Applications, Formulations, and Solution Approaches in: Traveling Salesman Problem, Theory and Applications. InTech, ISBN:978-953-307-426-9, 298 pages, 2010.

MAJAZI V. **Adjustment of the primitive parameters of the simulated annealing heuristic**. Indian Journal of Science and Technology. 4, 6 (2011), pp. 627-631, 2011.

MERZ P., FREISLEBEN B. **Fitness landscapes and memetic algorithm design**, in New Ideas in Optimization, D. Corne, M. Dorigo, and F. Glover, Eds. London, U.K.: McGraw-Hill, pp. 245–260, 1999.

MEUTH R., LIM M.-H., ONG Y.-S., WUNSCH D. **A proposition on memes and meta-memes in computing for higher-order learning**, Memetic Comput., vol. 1, no. 2, pp. 85–100, 2009.

MEUTH R., SAAD E., WUNSCH D., VIAN J., **Memetic mission man-agement**, IEEE Comput. Intell. Mag., vol. 5, no. 2, pp. 32–40, May 2010.

MICHALEWICZ Z., JANIKOW C. **Handling constraints in genetic algorithms**, in Proceedings of the Fourth International Conference on Genetic Algorithms, R. Belew and L. Booker, Eds. San Mateo, California: Morgan Kaufmann Publishers, pp. 151–157, 1991.

MICHALEWICZ Z. **Genetic Algorithms + Data Structures=Evolution Programs**, Springer Press, 1996.

MICHALEWICZ Z.; FOGEL D.B., **How to solve it: Modern Heuristics**, Springer. Berlin, 2000.

MOHAMED E. R., ZIDAN I. E. **Performance evaluation of genetic algorithm for solving routing problem in communication networks**. (IJCSIS) International Journal of Computer Science and Information Security, Vol. 8, No. 3, June 2010

MOHAMED E. R. et al. **Performance Evaluation of Genetic Algorithm for solving Routing Problem in Communication Network**, International Journal of Computer Science and Information Security, pp 37-43, Vol 8 Number 3, 2010

MUNTEAN, T. TALBI E.G. **Hill-climbing, simulated annealing and genetic algorithms, a comparative study**, Proc. Int. Conf. on Task Scheduling in Parallel and Distributed Systems HICSS'93, IEEE Computer Society Press, Jan 1993.

NERI F., MININNO E., **Memetic compact differential evolution for Cartesian robot control**, IEEE Comput. Intell. Mag., vol. 5, no. 2, pp. 54–65, May 2010.

NERI F., COTTA C., MOSCATO P. (Eds.) **Handbook of Memetic Algorithms, Studies in Computational Intelligence**, Springer, Vol. 379, November 2011.

NGUYEN T. T. **Incorporating Cultural Adaptation And Local Search Into Meta-Heuristics Continuous Optimisation**, MPhil Thesis, (School of Computer Science, University of Birmingham, Birmingham, December 2006, [online], accessed September 2011, URL: https://www.cs.bham.ac.uk/~txn/theses/mphil_thesis_thanh.pdf.

NGUYEN H. D., YOSHIHARA I, YASUNAGA M. **Implementation of an Effective Hybrid GA for Large-Scale Traveling Salesman Problems**. IEEE. Ttransactions on systems, man, and cybernetics-part b: cybernetics, vol. 37, no. 1, february 2007.

NGUYEN T. T., YAO X. **An Experimental Study of Hybridizing Cultural Algorithms and Local Search**. Int. J. Neural Syst. 18 (1): 1-17, 2008.

NGUYEN Q. H., ONG Y.-S., LIM M. H., **A probabilistic memetic framework,** IEEE Trans. Evol. Comput. , vol. 13, no. 3, pp. 604–623, Jun. 2009

NGUYEN Q. H., ONG Y. S., LIM M. H., KRASNOGOR N. **Adaptive Cellular Memetic Algorithms**". Evolutionary Computation 17(2): 231-256, 2009.

NOWOSTAWSKI, M. Poli, R. **Parallel Genetic Algorithm Taxonomy**, Submitted to Publication to: KES'99, 1999.

OCHOA, A., GONZÁLEZ S., CASTRO A., PADILLA N., BALTAZAR R. **Implementing Data Mining to improve a Game Board based on Cultural Algorithms**, in proceedings of Hais'2007, 2007.

OCHOA A., PADILLA A., GONZALEZ S., CASTRO A., HAL S. **Improve a Game Board based on Cultural Algorithms**. The International Journal of Virtual Reality, 7(2):41-46, 2008.

OLIVEIRA, A. C. **Algoritmos evolutivos para problemas de Otimização numérica com variáveis reais**. Monografia de Qualificação. Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos - SP, 2001.

ONG Y.-S., LIM M. H., CHEN X. **Memetic computation: Past, present and future**, IEEE Comput. Intell. Mag. , vol. 5, no. 2, pp. 24–31, May 2010.

OSTROWSKI D.A., TASSIER T., EVERSON M., REYNOLDS R.G. **Using Cultural Algorithms to evolve strategies in agent-based models**. Proceedings of the Congress on Evolutionary Computation, 2002 vol. 1, pp 741-746, 2002.

OSTROWSKI D., SCHLEIS G., RYCHTYCKYJ N., REYNOLDS R.G. Using Cultural Algorithms in Industry. Proceedings of the Swarm Intelligence Symposium, IEEE, pp 187-192, 2003.

PARTHASARATHY, P.V., GOLDBERG, D.E., BURNS, S.A.: Tackling multimodal problems in hy-brid genetic algorithms. p. 775 (2001).

PESSINI, E. CARLOS. **Algoritmos genéticos paralelos - uma implementação distribuída baseada em javaspaces** . Dissertação de Mestrado. UFSC, Florianópolis SC, 2003.

PUCHINGER J., RAIDL G. **Cooperating memetic and branch-and-cut algorithms for solv-ing the multidimensional knapsack problem**. In: Proceedings of the 2005 Metaheuristics International Conference, Vienna, Austria, pp. 775–780, 2005.

QUEIROZ, T. A. ; MIYAZAWA, F. K. ; WAKABAYASHI, Y. ; XAVIER, E. C. . Algoritmos para os Problemas da Mochila e do Corte de Estoque Tridimensional Guilhotinado. In: XLI Simpósio Brasileiro de Pesquisa Operacional, 2009, Porto Seguro - BA. XLI SBPO. p. 1-12, 2009.

RAO S. S., Engineering Optimization: **Theory and Practice**, Wiley, 2009.

RAVINDRAN A., RAGSDELL K. M., REKLAITIS G. V. Engineering Optimization: Methods and Applications, 2nd Edition, ISBN: 978-0-471-55814-9, Wiley, 2007.

REYNOLDS, R. G. **Advances in evolutionary computation**. In: . [S.l.]: Mc-Graw Hill Press, cap. An Overview of Cultural Algorithms, 1999.

REYNOLDS, R. G. **An introduction to cultural algorithm**. In: 3rd Annual Conference on Evolutionary Programming. [S.l.: s.n.], 1994.

REYNOLDS, R. G. **Cultural and Social Evolution in Dynamic Environments**. 2001b. CASOS, 2001.

REYNOLDS, R. G. **Knowledge swarms and cultural evolution**. In: Proceedings of American Anthropological Association Annual Meeting. [S.l.: s.n.], 2001a.

REYNOLDS, R. G. **Tutorial on Cultural Algorithms**. IEEE Swarm Intelligence Symposium, 2003.

Reynolds, R. G., AND PENG, B.*, **Cultural Algorithms: Computational Modeling of How Cultures learn to Solve Problems**, Seventeenth European Meeting on Cybernetics and Systems Research, Vienna, Austria, April13-16, 2004.

REYNOLDS, R. G.; ZANONI, E. **Why cultural evolution can proceed faster than biological evolution**. In: Proceedings of International Symposium on Simulating Societies. [S.l.: s.n.], p. 81-93, 1992.

REYNOLDS R.G., MICHALEWICZ Z., CAVARETTA, M.J. **Using Cultural Algorithms for Constraint Handling** in Genocop. In Evolutionary Programming IV, J.R. MacDonnell, R.G. Reynolds and David B. Fogel, editors. Bradford Book, MIT Press, Cambridge, Massachusetts, 1995.

REYNOLDS, R. G.; CHUNG, C. **Fuzzy approaches to acquiring experimental knowledge in cultural algorithms**. IEEE, p. 260 - 267, 1997.

REYNOLDS, R.; ZHU, S. **Knowledge-based function optimization using fuzzy cultural algorithms with evolutionary programming**. IEEE Transactions on Systems, Man and Cybernetics - Part B, v. 31, n. 1, p. 1-18, 2001.

REYNOLDS R. G., SALEEM S. **The Impact of Environmental Dynamics on Cultural Emergence**. Festschrift, in Honor of John Holland, to appear, Oxford University Press, pp. 1-10, 2003.

REYNOLDS R. G., ALI M. Z. **Computing with the Social Fabric: The Evolution of Social**

Intelligence within a Cultural Framework, IEEE Computational Intelligence magazine, IEEE Press, February, vol. 3, no.1, pp. 18-30, 2008.

REYNOLDS R. G., MOSTAFA Z. A. **Cultural Algorithms: Knowledge-driven engineering optimization via weaving a social fabric as an enhanced influence function**. IEEE Congress on Evolutionary Computation 2008: 4192-4199, 2008.

REYNOLDS R.G., CHE X. D., Ali M. **Weaving the social fabric: The past, present and future of optimization problem solving with cultural algorithms**. Int. J. Intell. Comput. Cybern., 3(4): 561-592, 2010.
ROSIN C. D., HALLIDAY R. S., HART W. E., BELEW R. K. **A comparison of global and local search methods in drug docking**," in the Seventh International Conference on Genetic Algorithms, T. Bäck, Ed. Michigan, USA: Morgan Kaufmann, pp. 221-228, 1997.

RUIZ-TORRUBIANO R., SUAREZ A., **Hybrid approaches and dimension-ality reduction for portfolio selection with cardinality constraints**, IEEE Comput. Intell. Mag., vol. 5, no. 2, pp. 92–107, May 2010.

SHURONG Z., JIHAI W.;HONGWEI Z. **Multi-Population Cooperative GA and Multi-Objective Knapsack Problem**. Management and Service Science (MASS), International Conference on:1-4 24 Aug 2010

SIARRY P., BERTHIAU G., DURBIN F., HAUSSY J. **Enhanced simulated annealing for globally minimizing functions of many-continuous variables**. ACM Transactions On Mathematical Software, 23, 209–228, 1997.

SILVA, C. A.; SOUZA, S. R. **Uma Aplicação da Metaheurística Híbrida Simulated Annealing-Iterated Local Search ao Problema de Fluxo Multiproduto sob o Espaço Capacitado.** TEMA. Tendências em Matemática Aplicada e Computacional v. 9, p. 165-174. 2008.

SILVA, DEAM. J. A, OLIVEIRA R. C. LIMÃO. **A Modified Cultural Algorithm based on Genetic Algorithm for Solving Global Optimization Problems**. GEM 2009: 85-91, 2009a.

SILVA DEAM J. A, OLIVEIRA R. C. LIMÃO. **A multipopulation cultural algorithm based on genetic algorithm for the MKP**. GECCO 2009: 1815-1816, 2009b.

SILVA DEAM. J. A, TEIXEIRA, O. N. OLIVEIRA R. C. L. **Performance Study of Cultural Algorithms Based on Genetic Algorithm with Single and Multi Population for the MKP**, in Bio-Inspired Computational Algorithms and Their Applications, ISBN: 978-953-51-0214-4, 2012.

SINHA A., CHEN Y.-P., GOLDBERG D. E. **Designing efficient genetic and evolutionary algorithm hybrids,** in Recent Advances in Memetic Algorithms (Studies in Fuzziness and Soft Computing, vol. 166). Hei-delberg, Germany: Physica-Verlag, pp. 259–288, 2004.

SIVANANDAM, S. N, DEEPA S.N.**Introduction to Genetic Algorithms**. Imprint Berlin ; New York : Springer, 2007.

SMITH J. E. **Coevolving Memetic Algorithms: A Review And Progress Report**. IEEE

Transactions on Systems, Man, and Cybernetics—part b: cybernetics, vol. 37, no. 1, February, 2007.

SONKAR, S. K., MALVIYA A. K., GUPTA D. L. , GANESH C. Software Testing using Genetic Algorithm. International Journal of Computer Science & technology (IJCST) -vol 3 issue 1, ISSN: 0976-8491 (Online), 2012.

SOUZA, Givanaldo Rocha. **Uma Abordagem por Nuvem de Partículas para Problemas de Otimização** Combinatória. Dissertação de Mestrado, Sistemas de Computação, UFRN, Natal-RN, 2006.

SOUZA, M. J. F. **Notas de aula da disciplina Inteligência Computacional para Otimização**. Departamento de Computação, Instituto de Ciências Exatas e Biológicas, Universidade Federal de Ouro Preto. 2011.

STERNBERG, M.; REYNOLDS, R. G. **Using cultural algorithms to support reengineering of rule-based expert systems in dynamic performance environments**: A case study in fraud detection. IEEE Transaction on Evolutionary Computation, v. 1, n. 4, p. 225 - 243, November 1997.

SUN J., GARIBALDI J. M. **A novel memetic algorithm for constrained optimization**. IEEE Congress on Evolutionary Computation 2010: 1-8 , 2010.

SWAPNA D., DEVIDAS G. J., SHYAM S. P. **PSO Based Memetic Algorithm for Unimodal and Multimodal Function Optimization**. SEMCCO (1) 2011: 127-134, 2011.

TAKAHAMA T., SAKAI S., IWANE N., **Constrained Optimization by the ε Constrained Hybrid Algorithm of Particle Swarm Optimization and Genetic Algorithm**, Springer-Verlag Berlin Heidelberg, 2005, vol. 3809, pp. 389–400, 2005.

TIAN, P., MA, J., ZHANG, D.M. **Application of the simulated annealing algorithm to the combinatorial optimisation problem with permutation property: an investigation of generation mechanism**. European Journal of Operational Research,118, 81–94, 1999.

UMA S.V., GURUMURTHY K.S., SINGH M. K. **GA Based Optimal Design of Network Architecture for Desired Connectivity and Traffic Demand.** International Journal of Scientific & Engineering Research Volume 2, Issue 11, ISSN 2229-5518, November-2011.

URSEM, R. K.. **Models for Evolutionary Algorithms and Their Applications in System Identification and Control Optimization**. (Ph.D. thesis), EVALife, Dept. of Computer Science, University of Aarhus, Denmark (178 pp.). 2003.

VENTURA·S. et al. **JCLEC: a Java framework for evolutionary computation.** In: Soft Comput., Vol. 12, Nr. 4 (2008) , p. 381-392, 2008.

VERHOEVEN M.G.A., AARTS E.H.L., SWINKELS P.C.J. **A parallel 2-opt algorithm for the Traveling Salsman Problem, Future Generation Computer Systems**, Vol.11. No.2, pp.175-182, 1995.

WAH B., CHEN Y., **Hybrid constrained simulated annealing and genetic algorithms**

fornonlinear constrained optimization, in Proceedings of the 2001 Congress on Evolutionary Computation, vol. 2, pp. 925–932, 2001.

WANG X. P., CAO L. M., **Genetic algorithm: theory application and software implementation** , Xi'an: Press of Xi'an Jiaotong University, 2002.

WANG Y., CAI Z., GUO G., ZHOU Z. **Multiobjective Optimization and Hybrid Evolutionary Algorithm to Solve Constrained Optimization Problems**. IEEE Transactions on Systems, Man, and Cybernetics, Part B, 2007

WEI J., ZHANG M. **A memetic particle swarm optimization for constrained multi-objective optimization problems**. IEEE Congress on Evolutionary Computation 2011: 1636-1643, 2011.

WEISE T. **Global Optimization Algorithms – Theory and Application**, published by it-weise.de (self-published), Germany, 2009.

WHITLEY D., STARKWEATHER T., AND FUQUAY D., **Scheduling problems and traveling salesman: The genetic edge recombination operator**, in Proc.3rd Int. Conf. Genetic Algorithms, 1989, pp. 133-140.

WHITLEY D., STARKWEATHER T., AND SHANER D., **The traveling salesman and sequence scheduling: Quality solutions using genetic edge recombination** in The Handbook of Genetic Algorithms, L. Davis, Ed. New York:Van Nostrand, 1991, pp. 350-372.

YAMAMOTO L. **Uso de Simulated Annealing e Algoritmo Genético no Problema da Reconfiguração de uma Rede de Distribuição de Energia Elétrica**, Dissertação de Mestrado, UFPR, 2004.

YANG, R.L. **Convergence of the simulated annealing algorithm for continuous global optimization**. Journal of Optimization Theory and Applications, 104, 691–716, 2000.

YANG X. S. **Engineering Optimization: An Introduction with Metaheuristic Applications**, John Wiley and Sons, ISBN 0470582464 , 2010.

YANG, X. S., **Review of meta-heuristics and generalised evolutionary walk algorithm**, International Journal of Bio-Inspired Computation, v.3 n.2, p.77-84, April 2011.

YOUNGSU Y. **Hybrid genetic algorithm with adaptive local search scheme**. Computers & Industrial Engineering,Vol.51, pp. 128–141, 2006.

XUE Z. G., GUO Y. N. **Improved cultural algorithm based on genetic algorithm**. Proceedingsof IEEE International Conference on Integration Technology, 117-122. 2007.

ZHANG, B.T., KIM, J.J., Comparison of Selection Methods for Evolutionary Optimization, November 1998.

ZHANG Y. **Study on Cultural Algorithm**, icfcse, pp.558-560, 2011 International Conference on Future Computer Science and Education, 2011.

ZHANG Y., ZHANG H. **A Novel Niche Genetic Algorithm for Multimodal Optimization**, Journal of Convergence Information Technology, Volume 6, Number 6, June 2011.

ZHOU E., CHEN X. **A New Population-Based Simulated Annealing Algorithm**, in *Proceedings of the 2010 Winter Simulation Conference*, pp. 1211-1222, 2010.

ZHU Z., JIA S., JI Z., **Toward a memetic feature selection paradigm**,IEEE Comput. Intell. Mag., vol. 5, no. 2, pp. 41–53, May 2010.

ZUBEN, F. J. V. **Computação Evolutiva: Uma Abordagem Pragmática** - Tese de Doutorado, DCA/FEEC/Unicamp, 2000.

ANEXOS

A1- Otimização de Funções Multimodais

Um problema de otimização global pode ser formalizado como um par (S, F) onde S é um conjunto delimitado em valores reais R ($S \subseteq R$) e F é uma função n-dimensional de valores reais (f: $S \rightarrow R$). O problema de otimização do tipo mínimo global consiste em encontrar um vetor $xmin \in S$ tal que F($xmin$) seja um mínimo global em S. De forma mais específica, o problema consiste em encontrar um $xmin \in S$ tal que $\forall x = (x1, x2, .., xn) \in S : f(x\min) \leq f(x)$, onde $n \in Z$ e $xi \in [\inf n, \sup n]$, sendo *inf* e *sup* os limites inferiores e superiores respectivamente.

A2-Problemas Restritos

Um problema de otimização restrito pode ser formalizado, considerando-se um vetor x com n variáveis independentes com valores reais que minimize

$f(x)$ (A1)

sujeito a:

$g(x) \leq 0$ (A2)

$h(x)=0$ (A3)

onde $x \in \mathbb{R}^n$ é referida como a solução, $f : \mathbb{R}^n \rightarrow \mathbb{R}$ é a função objetivo, $g : \mathbb{R}^n \rightarrow \mathbb{R}$ e $h : \mathbb{R}^n \rightarrow \mathbb{R}$ são as funções restritas em desigualdade e igualdade respectivaente.

A3-Problema da Mochila Multidimensional (*Multiple Knapsack Problem-MKP*)

Dado dois conjuntos de mochila ***m*** e ***n*** objetos, onde cada mochila ***i*** tem capacidade ***Ci*** , tal que cada objeto ***j*** tem lucro ***Pi,j*** na mochila ***i***, e para cada um restrição ***i***, um valor de consumo ***Wi,j*** é atribuído. O objetivo é determinar um conjunto de objetos que maximiza o lucro total , sem que exceda a restrição da capacidade ***Ci.***

Maximize $\sum_{i=1}^{m} \sum_{j=1}^{n} p_j x_{i,j}$ (A4)

Sujeito a $\sum_{j=1}^{n} w_j x_{i,j} \leq c_{i,}$ (A5)

$x_{i,j} \in \{0,1\}$, i =1,…, m, j =1,….,n (A6)

Com $p_j > 0$ $w_{i,j} \geq 0$ $c_i \geq 0$ (A7)

A variável de decisão é o vetor binário x = (x1 ,..., xn). Cada objeto j é mapeado para um bit. Quando *xj* = 1, o objeto correspondente é considerado como parte da solução. O caso particular de *m* = 1 é geralmente conhecido como o problema da mochila ou o Problema da Mochila Unidimensional. Esse problema também é conhecido na literatura como: *M-dimensional Knapsack Problem, Multiconstraint Knapsack Problem, Multi-Knapsack Problem,* ou *Multiple Knapsack Problem*. Alguns autores também incluem na sua designação o termo zero-um, por exemplo, Problema da Mochila Multidimensional zero/um ou Problema da Mochila problema da mochila 0/1. Usando nomes alternativos para o mesmo problema é um pouco confuso, mas uma vez que, historicamente, a designação MKP tem sido o mais utilizado.

A4-O Problema do Caixeiro Viajante (PCV)

Dado G(V; A) um grafo com n vértices em arestas. A cada aresta está associado um custo cij . O PCV consiste em encontrar um circuito hamiltoniano de menor custo. PCV simétrico: Dizemos que o problema é simétrico quando cij = cji para qualquer (i; j) ϵ A.

A formulação combinatória é a seguinte: dado um conjunto *C*={c1,..,cn} de *n* cidades *ci* e uma matriz de distâncias M(i,j) onde M(i,j)=M(ci,cj) (i, j ϵ {1,..,n}), M(i,j)=M(j,i), M(i,i)=0, a tarefa tem como base encontrar a permutação π ϵ Sn={s:{1,..,n}→{1,..,n}} que faça com que a função objetivo (distância do circuito) $f : S^n \rightarrow$ □ , onde:

$$f(\pi) = \sum_{i=1}^{n-1} M\ \pi(i), \pi(i+1) + M\ \pi(n), \pi(1)$$

atinja o seu valor mínimo.

A5- Benchmark Utilizado nas Funções Multimodais

Função		Ótimo
F1	$f_1(x) = 3(1-x_1)^2 e^{-x_1^2-(x_2+1)^2} - 10(\frac{x_1}{5} - x_1^3 - x_2^5)^2 e^{-x_1^2-x_2^2} - \frac{1}{3} e^{-(x_1+1)^2-x_2^2}$ $-3 \le x_i \le 3$	**Máximo** $f_1(0.00930, 1.58135) =$ 8.106213576688
F2	$f_2(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$ $-2.048 \le x_i \le 2.048$	**Mínimo** $f_2(1,1)=0$,
F3	$f_3(x) = floor(x_1) + floor(x_2)$ $-5.12 \le x_i \le 5.12$	**Máximo** $F_3(5.12, 5.08)=10$
F4	$f_4(x) = \sum_1^5 i\cos((i+1)x_1 + i) \sum_1^5 i\cos((i+1)x_2 + i)$ $-10 \le x_i \le 10$	**Mínimo** 760 mínimos locais -186.34 e 18 mínimos globais-186.7309
F5	$f_5(x) = \sum_1^{n-1} (100*(x_{i+1} - x_i^2)^2 + (1 - x_i)^2)$ $-2.048 \le x_i \le 2.048$ n=20 e n=30	**Mínimo** $f_5(x^*) = 0; x^* = (1,..,1)$
F6	$f_6(x) = \sum_1^n (x_i^2 - 10*\cos(2\pi * x_i) + 10)$ $-50 \le x_i \le 50$ n=20 e n=30	**Mínimo** $f_6(x^*) = 0; x^* = (0,..,0)$
F7	$f_7(x) = -\sum_{i=1}^4 c_i \exp\left[-\sum_{j=1}^6 a_{i,j}(x_j - p_{i,j})^2\right]$ $0 \le x_i \le 1$ $a_{i,j} = \begin{bmatrix} 10.0 & 3.0 & 17.0 & 3.5 & 1.7 & 8.0 \\ 0.05 & 10.0 & 17.0 & 0.1 & 8.0 & 14.0 \\ 3.0 & 3.5 & 1.7 & 10.0 & 17.0 & 8.0 \\ 17.0 & 8.0 & 0.05 & 10.0 & 0.1 & 14.0 \end{bmatrix}$ $p_{i,j} = \begin{bmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5886 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1415 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{bmatrix}$ $c_i = [1.0 \quad 1.2 \quad 3.0 \quad 3.2]$	**Mínimo** $f_7(0.201, 0.15, 0.477, 0.275, 0.311, 0.627) \approx -3.32$
F8	$f_8(x) = -20\exp\left(-0.2\sqrt{\frac{1}{30}\sum_{i=1}^n x_i^2}\right) - \exp\left(\frac{1}{30}\sum_{i=1}^{30}\cos 2\pi x_i\right) + 20 + e$ $-32 \le x_i \le 32 \qquad n = 30$	**Mínimo** $f_8(x^*) = 0$ $x^* = (0,..,0)$
F9	$f_9(x) = 0.1*\left\{\begin{array}{l}\sin^2(3\pi x_1) \\ +\sum_{i=1}^{n-1}(x_i-1)^2.[1+\sin^2(3\pi x_{i+1})] \\ +(x_n-1)^2[1+\sin^2(2\pi x_n)]\end{array}\right\} + \sum_{i=1}^n u(x_i, 5, 100, 4)$ $-50 \le x_i \le 50 \qquad n = 30$	**Mínimo** $f_9(x^*) = 0$ $x^* = (1,..,1)$

A6 - Benchmark Utilizado nos Problemas Restritos

A6.1 Problema-1

Este problema tem por objetivo minimizar o peso da tensão/compressão sobre uma mol a, que está sujeita a algumas restrições, tais como: deflexão mínima, tensão do cisalhamento, a frequência de onda, limites de diâmetro externo e variáveis de projeto. As variáveis de projeto são: o diâmetro do fio (d, x1), a bobina de diâmetro médio (D, x2) e o número de bobinas ativas (P, x3), conforme a figura abaixo:

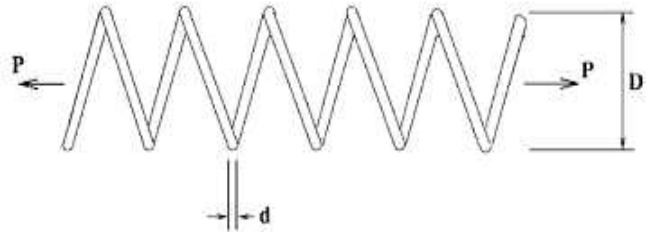


Figura A1. Minimização do peso da tensão/compressão sobre mola

O problema pode ser expresso da seguinte forma:

Minimize:

$$f(x) = (P+2)Dd^2 \tag{A8}$$

Sujeito a:

$$g_1(x) = 1 - \frac{D^3 P}{71785 d^4} \leq 0 \tag{A9}$$

$$g_2(x) = \frac{4D^2 - dD}{12566(Dd^3 - d^4)} + \frac{1}{5108d^2} - 1 \leq 0 \tag{A10}$$

$$g_3(x) = 1 - \frac{140.45d}{D^2 P} \leq 0 \tag{A11}$$

$$g_4(x) = \frac{D+d}{1.5} - 1 \leq 0 \tag{A12}$$

As variáveis estão de acordo com os seguintes intervalos:

$$0.05 \leq d \leq 2.0, 0.25 \leq D \leq 1.3, 2.0 \leq P \leq 15.0$$

A6.2 Problema-2

Este problema tem por objetivo minimizar o custo total , incluindo o custo do material, modelagem e soldagem, de um recipiente cilíndrico, que é limitado em suas extremidades por cabeças hemisféricas. Ele é composto por quatro variáveis de projeto, que são: esp e ssura da casca (Ts, x1), espessura da cabeça (Th, x2), raio interno (R, x3) e

comprimento da seção cilíndrica do recipiente (L, x4), não incluindo a cabeça, conforme pode ser visto na Figura 2. É importante observar que os valores para as variáveis Ts e Th são múltiplos de 0.0625 polegadas, pois referem-se às espessuras de chapas de aço laminadas; e,R e L são variáveis contínuas.

$$(T_s, T_H, R, L) = (x_1, x_2, x_3, x_4)$$

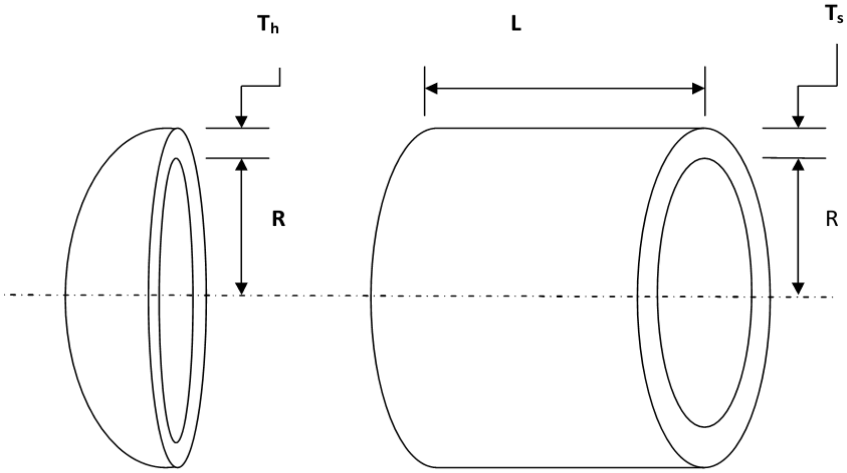


Figura. A2. Projeto de vaso de pressão.

O problema pode ser expresso da seguinte forma::

Minimize:

$$f(x) = 0.6224 x_1 x_3 x_4 + 1.7781 x_2 x_3^2 + 3.1661 x_1^2 x_4 + 19.84 x_1^2 x_3 \quad \text{(A13)}$$

Sujeito a:

$$g_1(x) = -x_1 + 0.0193 x_3 \le 0 \quad \text{(A14)}$$

$$g_2(x) = -x_2 + 0.00954 x_3 \le 0 \quad \text{(A15)}$$

$$g_3(x) = -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1,296,000 \le 0 \quad \text{(A16)}$$

$$g_4(x) = x_4 - 240 \le 0 \quad \text{(A17)}$$

As variáveis estão de acordo com os seguintes intervalos:

$$1 \le x_1 \le 99, 1 \le x_2 \le 99, 10 \le x_3 \le 200, 10 \le x_4 \le 200$$

A7 - Benchmark Utilizado nos Problemas da Mochila

O tamanho dos problemas da mochila varia de 15 a 105 objetos de 2 a 30 mochilas e podem ser encontrados em OR-Library (Beasley, 1990). Os problemas da mochila são definidos por: *problema (n, m)* onde *n* representa o número de objetos e *m* representa o número de mochilas. Cada mochila tem uma capacidade específica, assim cada objeto tem um peso específico. Por exemplo, Weing7 (105, 2) representa um MKP com 105 objetos e 2

mochilas.

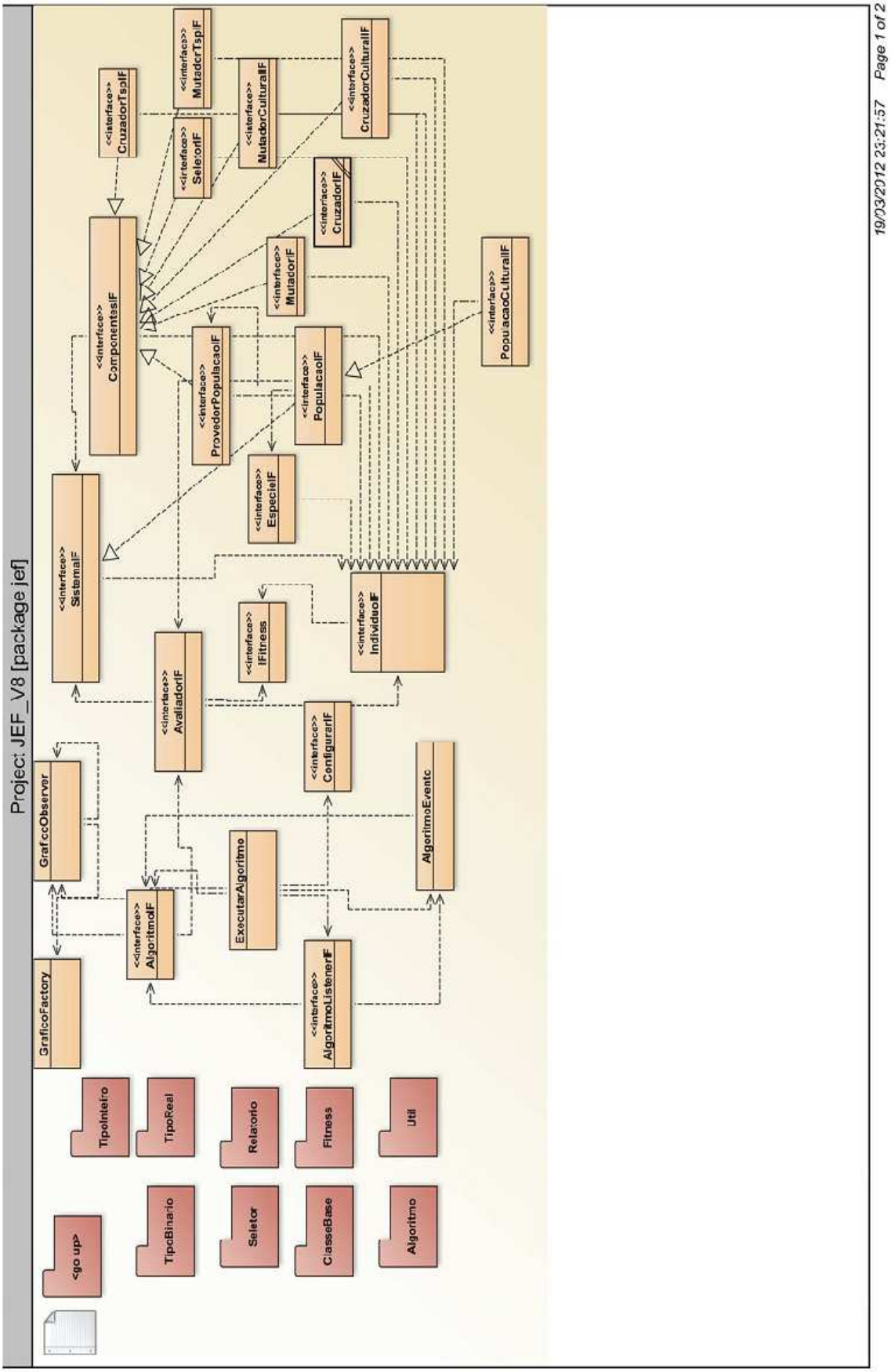
Tabela A1 Problemas da Mochila

Problema (n, m)
Weing7 (105, 2)
Petersen6 (39,5)
Petersen6 (39,5)
Petersen7(50,5)
Petersen7(50,5)
Sento1(60,30)
Sento1(60,30)
Sento2(60,30)
Sento2(60,30)

A8- Visão geral do Framework Implementado (*Java Evolutionary Framework* - JEF)

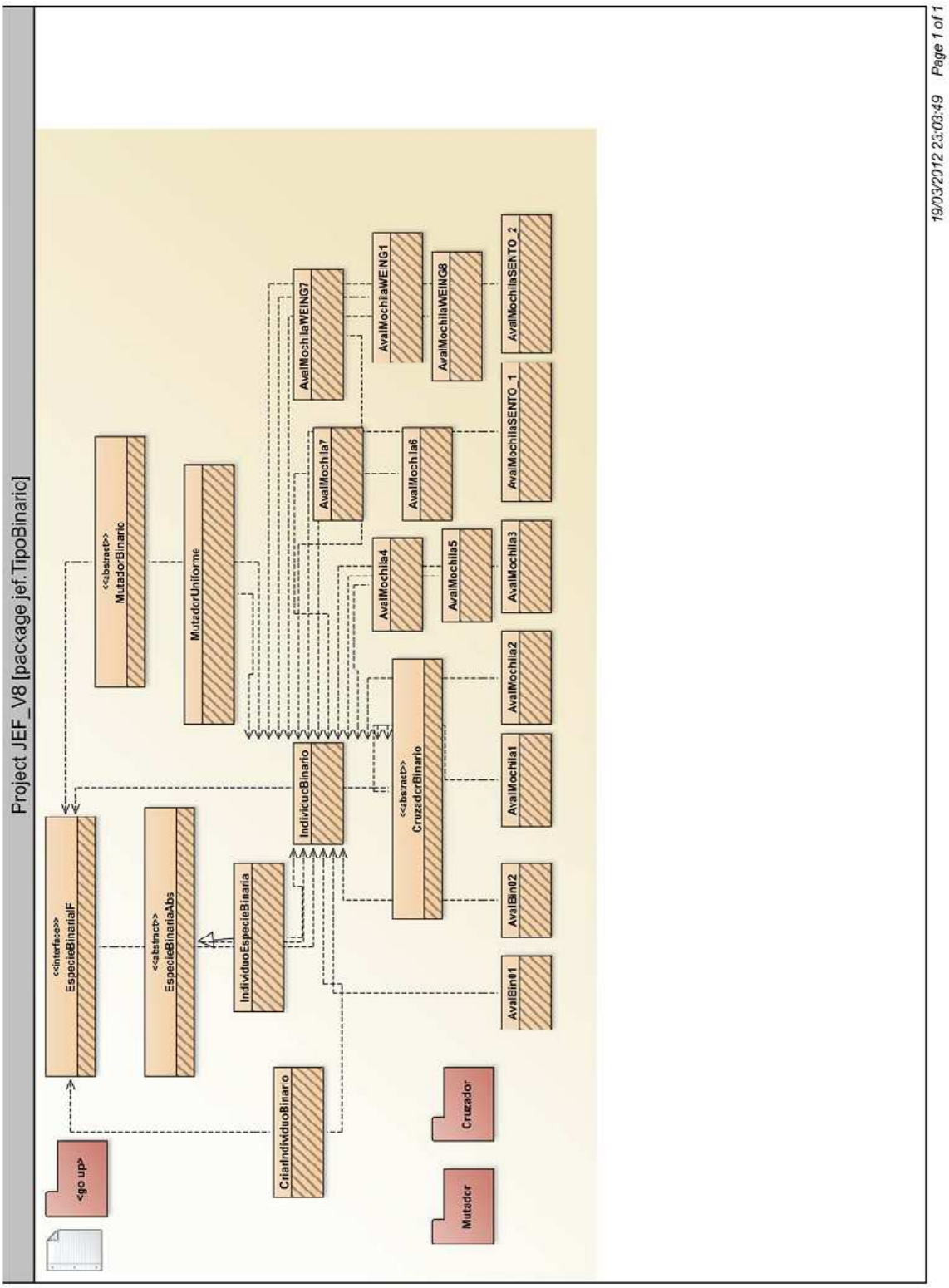
Qualquer tipo de algoritmo de Computação Evolutiva (CE) pode ser executado usando o JEF, desde que alguns requisitos mínimos sejam cumpridos. A única condição necessária é ter uma população de indivíduos a que uma sequência de operações evolutivas seja iterativamente aplicada. O sistema JEF foi codificado na linguagem de programação Java, que garante a sua portabilidade entre todas as plataformas que implementam uma JVM (*Java Virtual Machine*). Declarações de verificação e validação são embutidas no código para assegurar que as operações sejam válidas e para relatar problemas para o usuário. O uso de XML como formato de arquivo também é um aspecto central do JEF, que fornece uma base comum para o desenvolvimento de ferramentas para analisar e gerar arquivos, e para integrar o *framework* com outros sistemas (VENTURA ET AL, 2008).

A8.1 Estrutura principal do Framework



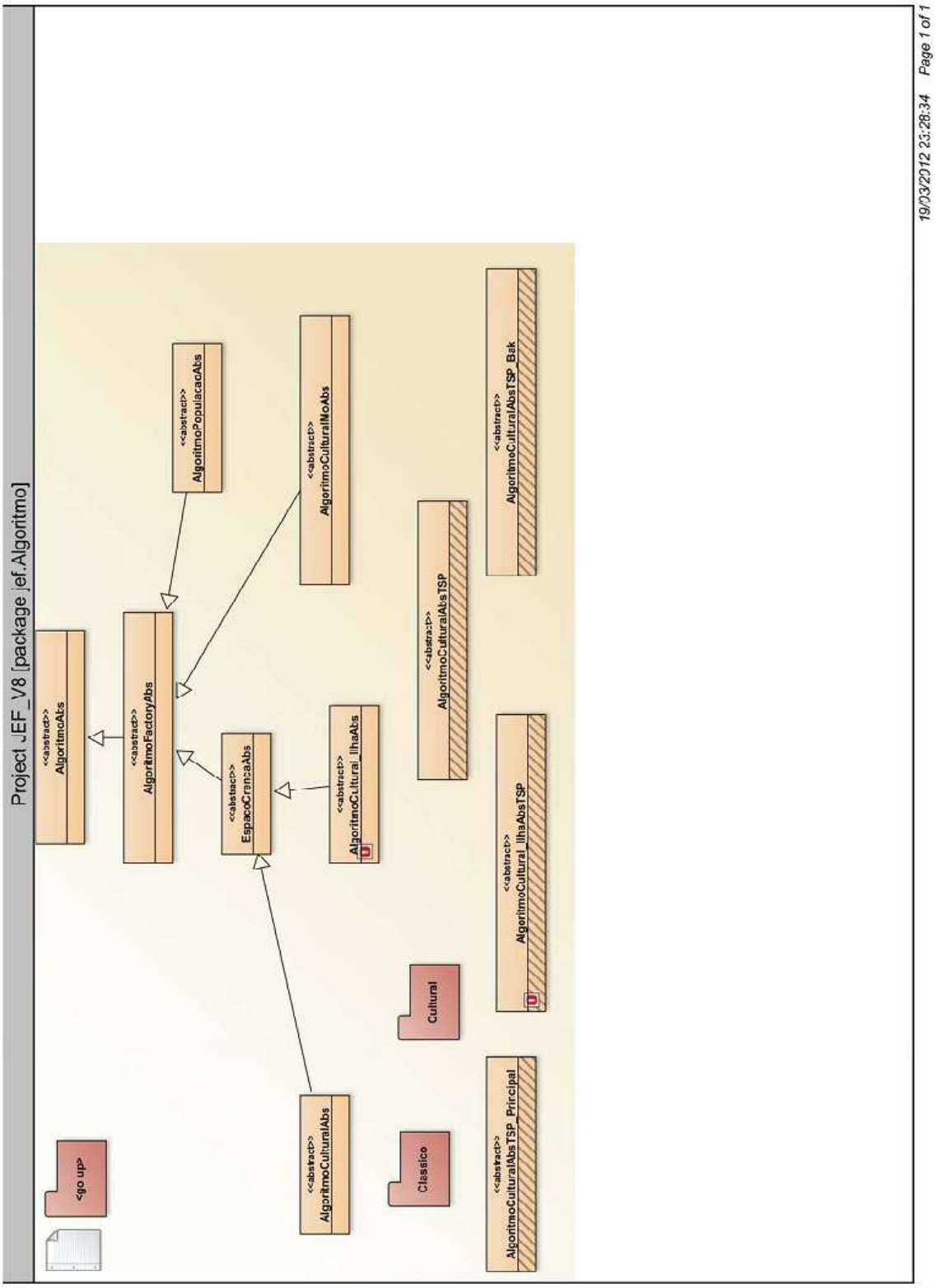
A8.2 Exemplo de uma Estrutura de Representação para o Cromossomo

A figura abaixo representa um tipo de estrutura do tipo Binário dentre as possíveis representações para o cromossomo, que também pode ser do tipo Real ou do tipo Inteiro.



A8.3 Estrutura de Representação dos Algoritmos

A figura abaixo representa a estrutura dos algoritmos evolutivos disponíveis no *framework.*



A9 Arquivo de Parâmetros do framework em XML

O aquivo de entrada segue o padrão XML. Desse modo, o framework carrega o arquivo XML e configura sua estrutura de acordo com os parâmetros definidos nesse arquivo.

A9.1 Exemplo:

```
<experiment>
    <max-repeticoes>50</max-repeticoes>
  - <process>
      - <algorithm type="jef.Algoritmo.Cultural.AlgCultural">
            <tipo-otimizacao type="Min"/>
            <population-size>200</population-size>
            <max-of-generations>50</max-of-generations>
            <conhecimento-cultural type="Situacional/Normativo"/>
            <valor-otimo> 0.0 </valor-otimo>
            <rand-gen-factory type="jef.Util.Aleatorio.RanMersenneTwistertFactory" Noseed="1234567890"/>
                <!--comentario tipo= "esquema-cromossomo=1****01****1"-->
          - <species type="jef.TipoReal.IndividuoEspecieReal">
                <precisao> 150</precisao>
              - <search-space>
                    <dimension type="jef.Util.Intervalo.IntervaloReal" right="2.048" left="-2.048" closure="closed-closed"/>
                    <dimension type="jef.Util.Intervalo.IntervaloReal" right="2.048" left="-2.048" closure="closed-closed"/>
                </search-space>
            </species>
            <provider type="jef.TipoReal.CriarIndividuoReal"/>
            <parents-selector type="jef.Seletor.SeletorTorneio" tam-torneio="7"/>
            <recombinator type="jef.TipoReal.Cruzador.FlatCrossoverCultural"    prob-cruz-ponto="0.35" prob-cruz="0.9"   />
            <mutator type="jef.TipoReal.Mutador.MutadorRandomicoCultural"   prob-mut-ponto="0.05" prob-mut="0.05
              prob-Pertuba-Max="0.5" prob-Pertuba-Min="0.5" min-Diferenca="20" max-Pertuba="0.1" min-Pertuba="0.001"
             energia-BuscaLocal="2" Busca-Local="N"    "/>
            <evaluator type="jef.TipoReal.GrupoBenchmark2.F2"/>
        </algorithm>
      + <listeners>
    </process>
</experiment>
```

São 50 execuções (max-repetições) utilizando o algoritmo Cultural (Algorithm <type>) com uma população de tamanho=200 (population-size) com 50 gerações (<max-of-generations>) utilizando o conhecimento situacional e normativo (<conhecimento-cultural>). O valor ótimo para o problema é definido em <valor-otimo>. O item <Randgen-gen-factory> indica o tipo de algoritmo utilizado para valores aleatórios (utiliza *seeds*). O problema aqui é definido do tipo real (species <type>) e contém duas variáveis no intervalo de -2.048 e 2.048 (<dimension <type>). O tipo de seleção é torneio (parentes selector <type>) com tamanho igual a 7 (<tam-torneio>). O tipo de cruzamento e mutação são descritos por <recombinator type> e <mutator type> respectivamente. As probabilidades de cruzamento e mutação são <prob-cruz> e <prob-muta> respectivamente. A função aqui é definida por <evaluator type>.

A busca local só ocorre se for =”S” (<Busca-local=”S”>) e então leva-se em consideração os seguintes parâmetros:

- A quantidade de energia do SA <energia-BuscaLocal>;
- A diferença mínima entre gerações que não houve melhoria de resultado, condição para que a busca local seja realizada <min-Diferença>;

- Valor mínimo e máximo para a perturbação das variáveis envolvidas <min-perturba> e <max-perturba> respectivamente (pode ser feito exclusivamente para cada intervalo da variável).

A9.2 Parâmetros de Perturbação independente

O exemplo abaixo, possui 3 variáveis e mostra que é possível acrescentar perturbações independentes para cada intervalo de variável. Basta acrescentar a sequência presente na marcação **<perturba-space>**.

```
- <search-space>
      <dimension type="jef.Util.Intervalo.IntervaloReal" right="2.0" left="0.05" closure="closed-closed"/>
      <dimension type="jef.Util.Intervalo.IntervaloReal" right="1.3" left="0.25" closure="closed-closed"/>
      <dimension type="jef.Util.Intervalo.IntervaloReal" right="15.0" left="2.0" closure="closed-closed"/>
  </search-space>
- <pertuba-space>
      <pertubation max-Pertuba="0.05" min-Pertuba="0.001"/>
      <pertubation max-Pertuba="0.05" min-Pertuba="0.001"/>
      <pertubation max-Pertuba="1.0" min-Pertuba="0.05"/>
  </pertuba-space>
```