Rubens Barbosa Filho

Multi-PBil: Um Algoritmo de Estimação de Distribuição Aplicado a Problemas de Otimização Multimodais

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Informática, Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná

Professora Dra. Aurora Trinidad Ramirez Pozo

Universidade Federal do Paraná

Curitiba

2005

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

PE Programação Evolucionária

EE Estratégias Evolutivas

AG Algoritmos Genéticos

PG Programação Genética

CE Computação Evolucionária

AE Algoritmos Evolucionários

EMAS Evolutionary MultiAgent System

AED Algoritmos de Estimação de Distribuição

GNU General Public License

PBIL Population-Based Incremental Learning

AIBP Aprendizagem Incremental Baseada em Populações

Lista de Símbolos

x Vetor de parâmetros

f(x) Função objetivo

g(x) Vetor de restrições do tipo desigualdade

h(x) Vetor de restrições do tipo igualdade

i Variáveis reais

n Restrições do tipo desigualdade

p Restrições do tipo igualdade

x^* Valor ideal ou ponto ótimo

F(x) Função de avaliação de desempenho

Resumo

Os Algoritmos de Estimação de Distribuição (AED) compõem uma meta-heurística evolutiva cuja principal característica é a construção de soluções de forma completamente aleatória, com o emprego de uma distribuição de probabilidades que evolua durante a execução.

O algoritmo *Population-Based Incremental Learning* (PBIL) é um tipo de AED onde as variáveis são independentes, isto é, não possuem interações significativas entre si. O PBIL considera que as soluções podem ser representadas como vetores de variáveis discretas, o que o torna mais adequado a problemas de otimização combinatória.

Este trabalho apresenta um método chamado Multi-PBil, que é uma extensão do PBIL com aplicações em problemas multimodais. O Multi-PBil foi desenvolvido a partir da necessidade de ter um algoritmo de busca eficiente e pouco custoso em espaços multimodais.

A partir do PBIL foi implementado uma rotina que permite ao Multi-PBil criar mais de um modelo de probabilidade para atuar no espaço de busca. Aplicou-se no processo de inicialização dos modelos de probabilidade uma fórmula que permite inicializar os modelos em regiões do espaço de busca mais próximas de pontos globais procurados. E por fim, foi criado e implementado um método de medida do espaço de busca, o qual permite a eliminação de modelos não-satisfatórios e a melhora na otimização da busca.

O método Multi-PBil foi testado e analisado, e apresenta alguns resultados experimentais que destacam sua viabilidade e características. São mostrados também resultados de uma comparação do desempenho do método Multi-PBIL com um Algoritmo Genético tradicional que utiliza a técnica de *sharing*.

Abstract

The Estimation Distribution Algorithms (EDA's) compose an evolutive metaheuristic whose main characteristic is the construction of solutions in randomly form, using a distribution of probabilities that evolves during the execution.

The Population-Based Incremental Learning (PBIL) is a type of EDA where the variables are independent, that is, they do not possess significant interactions between thenselves. The PBIL considers that the solutions can be represented as vectors of discrete variables, what makes it more adequate to optimization combinatorial problems.

This paper presents a method called Multi-PBil, that is an extension of PBIL with applications in multimodal problems. The Multi-PBil was developed from the necessity to have an efficient and non-expensive algorithm of search in multimodal spaces.

From PBIL, it was implemented a routine that allows the Multi-PBil to create a probability model to act in the search space. It was applied in the process of the probability model initialization a formula that allows to initiate the probability models in regions of the search space next to the searched global points. Was developed and implemented a method of measure of the search space, which allows the elimination of not-satisfactory models and the improvement in the optimization of the search.

The Multi-PBil method was tested and analyzed, presenting some experimental results that highlight its viability and characteristics. It is also shown a comparison of the performance between the Multi-PBil method and a traditional Genetic Algorithm using the sharing method.

1 Introdução

A Computação Evolucionária (CE) se refere a uma grande classe de algoritmos, os quais são inspirados nos sistemas naturais e evolucionários [Dar59]. Geralmente, cada algoritmo começa com uma população de indivíduos que irão se reproduzir e competir pela sobrevivência [NFM99]. Cada indivíduo possui um valor de aptidão, o qual indica o seu potencial dentro da população. Entre as técnicas que fazem parte da computação evolucionária estão [WBPN98]: Programação Evolucionária(PE), Estratégias Evolucionárias(EE), Algoritmos Genéticos(AG), Programação Genética(PG) e os Algoritmos de Estimação de Distribuição (AED'S).

Destacando-se os Algoritmos Genéticos, estes são técnicas de otimização estocásticas que imitam a Evolução Darwiniana através do processo de modelagem de seleção natural e dos modificadores genéticos [Mic96]. Baseam-se em princípios da biologia genética e operam de forma análoga à teoria da evolução [Coo91]. Os Algoritmos Genéticos têm-se mostrado viáveis em uma variedade de domínios unimodais [RS94]. Em um ambiente de evolução natural é comum ter-se uma variedade entre as espécies, cada uma ocupando um nicho ecológico em separado. Porém, quando se trabalha com Algoritmos Genéticos, pode ocorrer de rapidamente uma população artificial sofrer uma convergência, e todos os indivíduos da população tornarem-se quase idênticos [Lan89].

Entretanto, este fato atrativo pode não ser interessante em muitos problemas, como por exemplo, no caso das funções multimodais, onde o algoritmo corre o risco de convergir para um ponto de convergência prematura [MAB99]. A convergência prematura é devida à perda de diversidade populacional. Para superar este problema, a diversidade deve ser mantida durante o processo de geração de indivíduos na população, prevenindo que os indivíduos sejam cópias do indivíduo de melhor *fitness* [JDS89].

Os Algoritmos de Estimação de Distribuição (AED's), [PL01] são muito semelhantes aos Algoritmos Genéticos. Os AED's são algoritmos heurísticos de otimização que baseiam sua busca no caráter estocástico. Da mesma forma que os Algoritmos Genéticos, os

AED's estão fundamentados em populações que evoluem. Porém, ao invés de evoluir a população, os AED's evoluem os parâmetros de uma distribuição probabilistica da qual são amostradas as populações de indivíduos. Um dos exemplos mais simples de AED é a Aprendizagem Incremental Baseada em População (AIBP) [Bal94].

A Aprendizagem Incremental Baseada em População (AIBP) é um dos algoritmos evolucionários baseados em modelos. O PBIL é fruto de uma combinação de duas técnicas, o Algoritmo Genético e a Aprendizagem Competitiva [Bal94]. Originalmente, o PBIL foi configurado para busca em espaços binários. O PBIL faz uso das variáveis aleatórias de Bernoulli como modelo para cada *bit*, o qual é colocado em um vetor de valores reais. Diferentemente dos algoritmos genéticos que evoluem suas populações de indivíduos, o PBIL evolui seu modelo de probabilidade.

Baluja mostrou que o PBIL é mais eficiênte que os Algoritmos Genéticos e o Algoritmo *Hillclimbing* em um conjunto de problemas [Bal94], e desde então o PBIL tem sido utilizado em uma variedade de aplicações. Dentre as possíveis aplicações do PBIL, estão, por exemplo, pesquisas envolvendo o uso de modelos de probabilidade complexos [IS97, MS98, PL01, MP99].

O desafio de resolver problemas de otimização, como por exemplo, combinatórios multimodais é uma função não apenas da complexidade combinatorial, mas também da dificuldade em alcançar todas as soluções eficientes que crescem com o número de objetivos do problema. Neste contexto, para resolver problemas de otimização multimodal, são bem vindos os métodos mais flexíveis e de fácil implementação. Sendo assim, os algoritmos classificados dentro da computação evolucionária, mais especificamente os AED's, apresentam um conjunto de características aplicáveis à solução deste tipo de problema.

Neste tipo de classe de problemas, os métodos baseados no conceito evolucionário privilegiam os indivíduos mais adaptados, onde o procedimento de busca é baseado na evolução dos vetores de probabilidade.

Assim sendo, procurando fazer uso dos benefícios particulares do método citado AED's e, em função da diversidade de problemas aplicáveis, os quais podem ser resolvidos com muita flexibilidade, este trabalho propõe estudar e implementar um método de computação evolucionária, mais especificamente um algoritmo de estimação de distribuição chamado Multi-PBil, com a inserção de características que permitam a busca em espaços multimodais.

São três as características importantes que fazem do Multi-PBil um método robusto

e eficiênte. A primeira é a possibilidade de se criar quantos modelos de probabilidade forem necessários para uma busca eficiênte em um espaço multimodal. A segunda característica permite inicializar as regras de inicialização dos modelos de probabilidade com valores próximos de zero ou um; E isto significa que os modelos podem ser inicializados próximos dos objetivos procurados pelo problema. E como terceira característica esta a possibilidade de se eliminar modelos de probabilidade com desempenho insatísfatório, aumentando desta forma a performance e desempenho do método Multi-PBil.

1.1 Motivação e Justificativa

Este trabalho tem a motivação de desenvolver uma técnica que seja eficiente e não custosa para trabalhar em problemas de otimização em espaços multimodais chamada Multi-PBil. Esta técnica é baseada em modelos de probabilidade e possui a vantagem de poder gerar mais de um modelo, e consequentemente poder realizar uma busca por mais de um objetivo simultaneamente. Com este trabalho, os pesquisadores da área de Algoritmos Evolucionários, que utilizam técnicas de busca multimodal, irão dispor de um método que irá lhes auxiliar e servir de comparativo na solução de problemas tradicionais.

1.2 Organização

Este trabalho encontra-se dividido em 5 capítulos. No Capitulo 2 é abordada a teoria de computação evolucionária, dando ênfase aos Algoritmos Genéticos e seus componentes. No Capítulo 3, tem-se uma exposição sobre os algoritmos de distribuição de probabilidade (AED's), onde destaca-se a teoria, e o processo de execução de um de seus algoritmos, no caso o algoritmo PBIL. O Capítulo 4 trata do algoritmo Multi-PBil e todos os componentes necessários para a composição da técnica proposta. O Capítulo 5 traz a descrição dos experimentos e resultados dos testes realizados para avaliar o desempenho da técnica proposta. O Capítulo 6 apresenta a conclusão do trabalho e como parte final têm-se os Apêndices, onde é destacada a teoria sobre Problemas de Otimização e o Código do algoritmo implementado para testar o método Multi-PBil.

2 Computação Evolucionária

Neste capítulo serão explicados os conceitos sobre Computação Evolucionária, dando-se ênfase ao estudo dos Algoritmos Genéticos. É importante destacar o papel dos Algoritmos Genéticos porque os AED's, que é o foco principal deste trabalho, é fruto dos Algoritmos Genéticos.

Para todos os problemas existentes no campo da Inteligência Artificial deseja-se desenvolver um algoritmo eficiente. Muitos destes, são problemas de otimização (númerica, combinatorial), outros são de síntese de um objeto (programa de computador, circuito eletrônico, etc.)e, em outros, busca-se um modelo que reproduza o comportamento de determinado fenômeno (aprendizado de máquina).

Para vários desses problemas é frequentemente possível encontrar um algoritmo que ofereça uma solução ótima ou aproximadamente ótima. Alguns desses algoritmos requerem informação auxiliar, informação esta muitas vezes não disponível ou difícil de se obter. A Computação Evolucionária oferece algoritmos gerais (Algoritmos Genéticos, Programação Genética e etc.) que são aplicados em problemas complexos, com grandes espaços de busca, de difícil modelagem, ou para os quais não há um algoritmo eficiente disponível.

A Computação Evolucionária compreende diversos algoritmos inspirados no princípio Darwiniano da evolução das espécies e na genética. São algoritmos probabilísticos, que fornecem um mecanismo de busca paralela e adaptativa baseado no princípio de sobrevivência dos mais aptos e na reprodução.

Usa-se o termo Computação Evolucionária (CE) em geral, quando se quer referir a uma classe de técnicas computacionais inspiradas nos princípios da evolução natural. Os primeiros Algoritmos Evolucionários (AE) surgiram no final da década de 60 do século XX. Na Alemanha com Rechemberg [Rec73] surgiram as Estratégias Evolutivas (EE) e nos Estados Unidos com Holland surgiram os Algoritmos Genéticos (AG) [Hol75].

As estratégias evolutivas propostas por Rechemberg tinham como objetivo resolver

problemas que tratavam de codificações reais, enquanto os algoritmos genéticos tinham como objetivo fornecer um bom desempenho nos problemas de otimização utilizando uma codificação binária [Gol89b].

Um tempo mais adiante, surgiu também a Programação Evolucionária (PE) desenvolvida por Fogel [LJF66], com aplicação na Inteligência Artificial(IA), mais especificamente em espaço de busca discretos. Há também a Programação Genética (PG) proposta por Koza [Koz92], com o objetivo de utilizar os AGs para produzir programas mais eficientes na resolução de uma determinada tarefa.

Os Algoritmos de Estimação de Distribuição (AED's) são um conjunto de Algoritmos Evolucionários caracterizados pelo uso explícito de modelos de probabilidade usados com o intuito de recuperar informações de indivíduos selecionados e prover novas soluções, bem como a possibilidade natural de se poder incorporar o conhecimento prévio sobre o problema de otimização a ser resolvido [PLN05].

A comunidade de IA têm apresentado um interesse crescente em estudar os AED's, que constituem agora uma disciplina estabelecida no campo da Computação Evolucionária (CE) [PLN05].

2.1 Algoritmos Genéticos

Os Algoritmos Genéticos (AG) são um modelo computacional de pesquisa probabilística, inspirados no processo de seleção natural e na genética. Foram desenvolvidos por [Hol75] na Universidade de Michigan, onde foram aplicados com sucesso em diversas áreas, principalmente em problemas de otimização numérica [Bet81], ou problemas relacionados com processos adaptativos [Jon80].

Os AGs trabalham sobre um população de potenciais soluções, ou indivíduos, às quais é aplicado o princípio da sobrevivência dos melhores indivíduos, isto é, os indivíduos competem entre si pela sobrevivência. Após serem avaliados, os melhores indivíduos têm uma maior probabilidade de serem escolhidos(pais) para gerarem novos indivíduos(descendentes) [Gol89a]. A geração de novos indivíduos é feita através de operadores baseados na genética. Desta forma, os descendentes são gerados a partir da recombinação dos pais, dos quais herdam algumas de suas características. Há também o operador de mutação, que permite o aparecimento de uma característica nova. Os descendentes gerados são re-inseridos na população de indivíduos segundo uma regra de inserção relativa ao problema. Esse processo é repetido durante um determinado número

de gerações.

Ao longo das gerações, uma vez que os melhores indivíduos têm maior probabilidade de serem selecionados para gerar descendentes, e possivelmente bons descendentes, a população tende a ter cada vez melhores indivíduos. Este processo de procura genética conduz à evolução da população e os melhores indivíduos tendem a sobreviver, tal como sucede na evolução natural [Gol89a].

Os componentes básicos de um AG são a população de indivíduos, em que cada um deles representa uma potencial solução para o problema considerado, o mecanismo de avaliação dos indivíduos da população e os operadores genéticos que pesquisam novas soluções. A Figura 1 mostra o funcionamento de um AG simples.

Um pseudocódigo do Algoritmo Genético é mostrado a seguir:

1. Geração da população inicial

   A população inicial de indivíduos é criada aleatoriamente.

2. Avaliação ou Cálculo da aptidão

   Cada indivíduo da população é avaliado de acordo com uma medida de desempenho.

3. Seleção

   Um conjunto de indivíduos é selecionado para gerar descendentes de tal forma que os que tiverem melhor desempenho têm maior probabilidade de serem selecionados.

4. Reprodução

   Ao conjunto de indivíduos selecionados é permitido reproduzir-se entre si, gerando novos indivíduos.

5. Operadores Genéticos

   Ao conjunto de indivíduos selecionados são aplicados operadores genéticos como a recombinação e a mutação.

   SE não se satisfizer o critério de parada ENTÃO volta-se ao passo 2, SENÃO terminar.

2.1.1 População de Indivíduos

Nos AGs, os indivíduos são representados por sequências de dígitos binários. As sequências de dígitos binários são análogas às sequências de amino-ácidos (A, C, T e G)

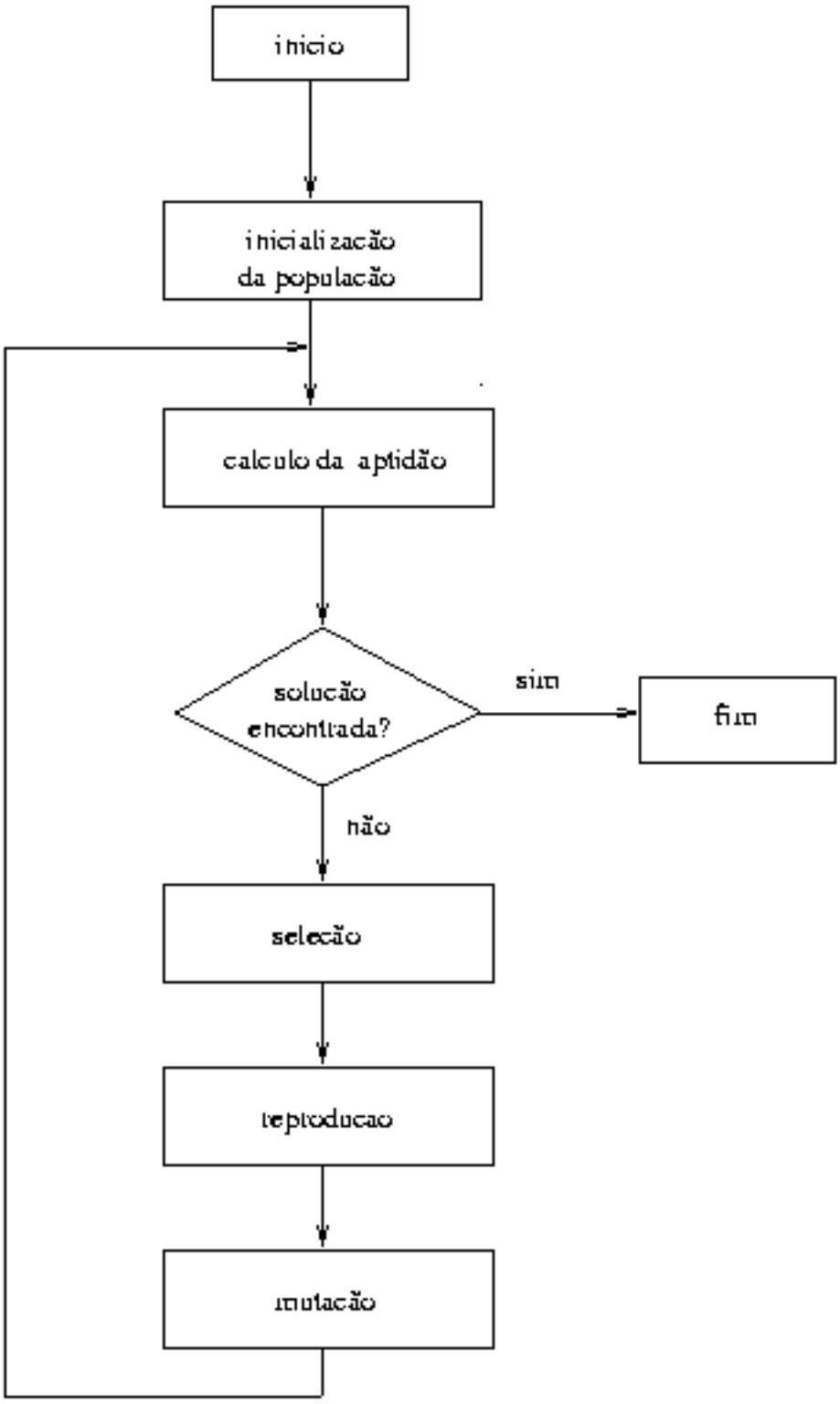

Figura 1: Um algoritmo genético simples

nos cromossomas dos sistemas biológicos. Em sistemas biológicos, os cromossomos são constituídos por genes que podem assumir um determinado número de valores chamados de alelos. Define-se também o locus do gene como sendo a sua posição no cromossoma. Já em AGs, as sequências binárias são constituídas por dígitos binários (0 ou 1), que podem estar localizados em diferentes posições [Gol89a].

A posição de cada dígito binário na sequência binária define o seu significado. O fato de se trabalhar com uma codificação em binário faz com que se tome um cuidado redobrado, pois neste tipo de código, dependendo da sua posição, a mudança de um único dígito binário pode determinar uma pequena ou grande mudança dos valores das variáveis de decisão.

Em termos biológicos, a totalidade do pacote gênico (combinação de um ou mais cromossomas que descrevem a forma de construção e operação do organismo) constitui o genótipo que corresponde, em analogia aos AGs, às chamadas estruturas. Por outro lado, a interação do genótipo com o ambiente é definida como o fenótipo, e, analogamente, as estruturas são decodificadas para formar soluções [Gol89a].

Na Tabela 1 está uma correspondência entre os Sistemas Biológicos e os Algoritmos Genéticos.

Tabela 1: Correspondência entre os sistemas biológicos e os algoritmos genéticos

Sistemas Biológicos	Algoritmos Genéticos
Cromossoma	Sequência de Dígitos Binários
Gene	Dígito Binário
Alelo	0 ou 1
Locus	Posição na Sequência de Dígitos Binários
Genótipo	Estrutura
Fenótipo	Solução

Um exemplo de cromossomo de genes binários pode ser visto na Figura 2.

Vale ressaltar que a utilização de uma representação binária, faz com que os indivíduos possam ser vistos em dois níveis distintos: nível do genótipo e nível do fenótipo. O fenótipo de um indivíduo é o seu valor no domínio em que a função objetivo está definida. Ou seja, o fenótipo é a expressão do genótipo, isto é, a estrutura consistindo na codificação das variáveis de decisão através de uma sequência de dígitos binários. Assim sendo,

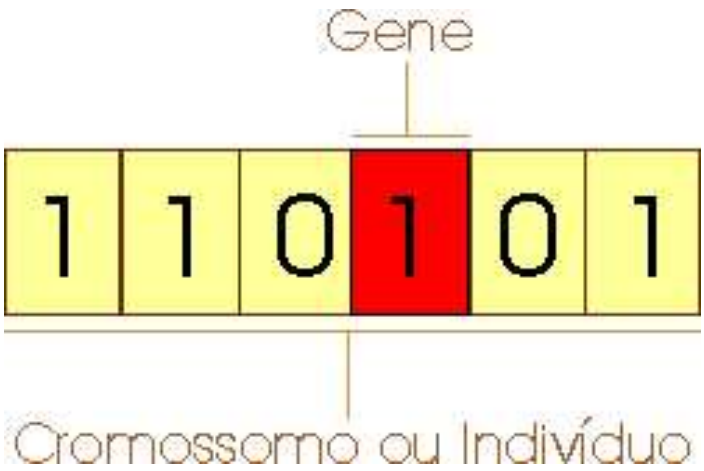

Figura 2: Exemplo de representação de um cromossomo de genes binários

a codificação especifica um mapeamento que transforma uma potencial solução para o problema numa estrutura que pode ser manipulada pelo AG [Gol89a].

2.1.2 Tipos de Seleção

A seleção em algoritmos genéticos determina quais os indivíduos pertencentes à população serão escolhidos para serem progenitores. Nestes indivíduos progenitores aplicam-se os operadores genéticos gerando novos indivíduos. A função de seleção é probabilística, e é feita de acordo com as medidas de avaliação de desempenho dos indivíduos, de forma que os melhores tenham maior probabilidade de serem selecionados. Assim, o número de descendentes de cada indivíduo selecionado está relacionado com o mecanismo de seleção considerado [Gol89a]. Dentre os principais algoritmos de seleção, destacam-se:

- **Seleção por Roleta** - A seleção por roleta é um mecanismo de escolha estocástica. A cada indivíduo associa-se uma fatia da roleta, proporcional, em tamanho, à medida de avaliação do desempenho do indivíduo. A seguir, gera-se um número aleatório e, o indivíduo cuja fatia inclua esse número é escolhido. O processo é repetido até se selecionar o número de indivíduos desejados [Gol89a].

- **Seleção por Torneio** - Neste tipo de seleção, um número t de indivíduos da população é escolhido de forma aleatória, e o melhor indivíduo deste grupo, de acordo com as medidas de avaliação desempenho, é selecionado para progenitor. Este processo repete-se tantas vezes quanto o número de indivíduos a escolher. Faz-se necessário definir o valor de t, sendo que o valor deste parâmetro está relacionado

com a pressão de seleção. Quanto maior é o valor do parâmetro t, maior é a pressão de seleção [Gol89a].

- **Seleção por Truncagem** - Neste tipo, os indivíduos são ordenados de acordo com sua medida de avaliação de desempenho. Após a ordenação, somente os melhores indivíduos são selecionados para serem progenitores. Faz-se necessário definir a proporção de indivíduos da população a serem selecionados para progenitores [Gol89a].

2.1.2.1 Avaliação ou Cálculo do Desempenho

A avaliação ou cálculo do desempenho consiste na atribuição de valores aos indivíduos de uma população de acordo com a sua utilidade para a resolução de um problema. A avaliação do desempenho permite condicionar a eficiência do processo de procura. Entre os possíveis esquemas de avaliação de desempenho, destacam-se os seguintes:

- **Proporcional Simples** [Gol89b] - onde os valores associados aos indivíduos de uma população dependem dos valores da função objetivo **f(x)**: $F(x) = f(x)$ onde $F(x)$ significa a função de avaliação de desempenho. Como a função objetivo é sempre positiva, a função de avaliação de desempenho tende em certos casos, a ser o fator causa de uma convergência prematura, principalmente quando as médias das medidas de desempenho dos indivíduos da população se aproximar das medidas de desempenho dos melhores indivíduos.

- **Escala Linear** [Gol89b] - A avaliação de desempenho com escala linear faz uso da seguinte função linear entre a função de medida do desempenho $F(x)$ e a função objetivo $f(x)$:

$$F(x) = mf(x) + b,$$

onde os coeficientes m e b podem ser calculados de diversas formas. É preciso garantir que a média dos valores em escala $\overline{F}$ é igual à média dos valores da função objetivo $\overline{f}$. O valor máximo da função de avaliação de desempenho F_{max} é obtido a partir da relação:

$$F_{max} = c\overline{f}$$

onde tipicamente, o coeficiente c é escolhido de tal forma que $c > 1$. E os valores de m e b são calculados da seguinte forma:

$$m = \frac{F_{max} - \overline{f}}{f_{max} - \overline{f}} = \frac{c\overline{f} - \overline{f}}{f_{max} - \overline{f}} = \frac{\overline{f}(c-1)}{f_{max} - \overline{f}}$$

e

$$b = \overline{f} - m\overline{f}.$$

Quando o valor de f_{min} estiver bastante afastado dos valores de $\overline{f}$ e de f_{max}, os valores da função de avaliação de desempenho $F(x)$ podem tornar-se negativos. Para que isto não aconteça, deve-se impor que a medida dos valores em escala $\overline{F}$ seja igual à medida dos valores da função objetivo $\overline{f}$ e $F_{min} = 0$. Para tal, o valor de m deve ser calculado da seguinte forma:

$$m = \frac{\overline{F} - F_{min}}{\overline{f} - f_{min}} = \frac{\overline{F}}{\overline{f} - f_{min}}$$

- **Janela de Escala** [Gre84] - Este tipo de avaliação requer uma relação entre a função de avaliação de desempenho $F(x)$ e a função objetivo $f(x)$:

$$F(x) = f_{max} - f(x),$$

onde f_{max} é o valor máximo que a função objetivo $f(x)$ pode tomar num determinado espaço de busca. Desta forma, a função de avaliação de desempenho $F(x)$ é sempre positiva quaisquer que sejam as características da função objetivo $f(x)$.

- **Lei de Potência** [Gil85] - Esta avaliação de desempenho usa uma Lei de Potência para alterar a escala. Neste tipo de medição, os valores da função de avaliação de desempenho $F(x)$ são obtidos através de uma determinada potência da função objetivo $f(x)$:

$$F(x) = f(x)^k.$$

Porém, a determinação do valor para a constante k não é simples, e depende do problema considerado.

2.1.3 Recombinação

A recombinação ou cruzamento produz novos indivíduos (descendentes) combinando a informação de seus progenitores. Considerando que os indivíduos são representados por sequências de dígitos binários, diferentes algoritmos de recombinação podem ser aplicados [Gol89a]. Os mais utilizados são mostrados a seguir:

- **Cruzamento Simples** - No cruzamento simples, determina-se aleatoriamente um ponto de cruzamento. A troca de informação a partir desse ponto entre os indivíduos

progenitores produz dois descendentes. A Figura 3 exemplifica a aplicação deste tipo de operador genético considerando o ponto de cruzamento na posição 4.

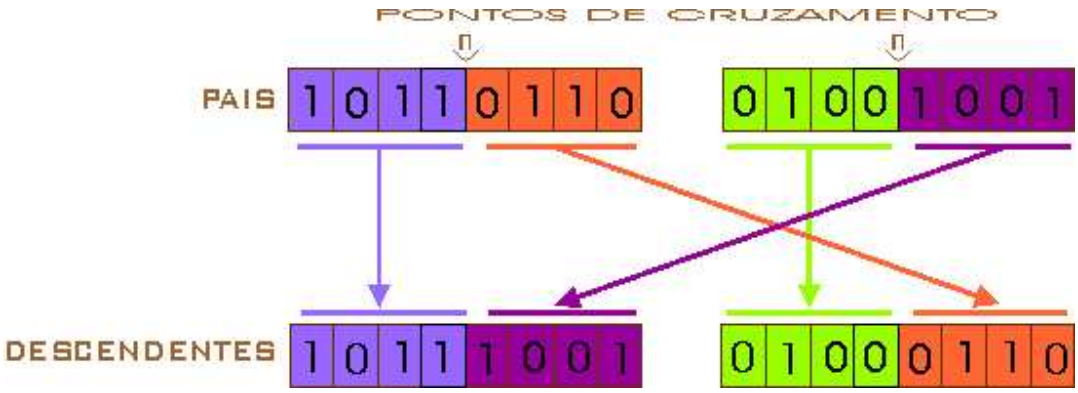

Figura 3: Esquema gráfico do cruzamento de um ponto

- **Cruzamento Múltiplo** [Boo87] - Neste, determina-se aleatoriamente i pontos de cruzamento. Depois de ordenados por ordem crescente os pontos de cruzamento, entre cada dois pontos de cruzamento consecutivos, troca-se informação entre os indivíduos progenitores produzindo dois descendentes. A informação até o primeiro ponto de cruzamento não é trocada entre os dois indivíduos progenitores. Na Figura 4, o operador genético é aplicado considerando dois pontos de cruzamento nas posições 2 e 6 (cruzamento duplo).

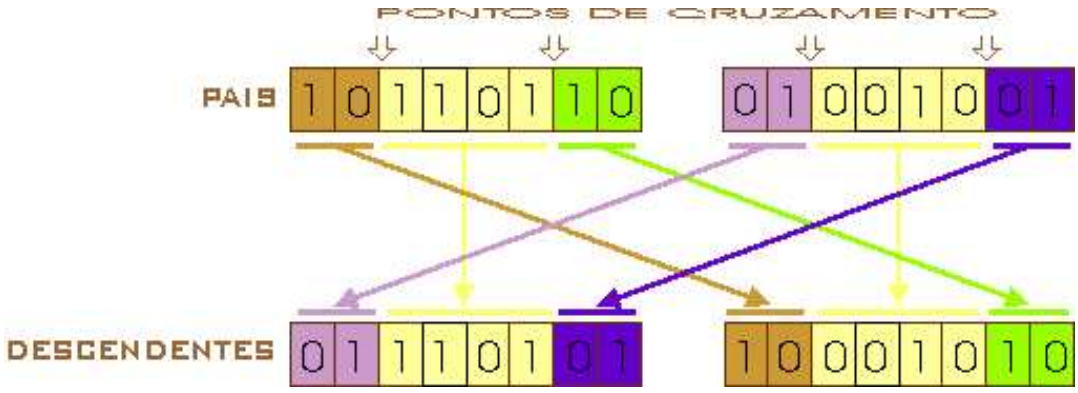

Figura 4: Esquema gráfico do cruzamento em dois pontos

O cruzamento múltiplo possibilita a diminuição dos fenômenos de ruptura do cruzamento simples. A ruptura é um fenômeno que consiste em, durante o processo de recombinação genética, certa informação dos indivíduos progenitores não ser passada aos indivíduos descendentes. Na prática, pode-se dizer que a redução deste efeito de ruptura conduz a um aumento da eficiência do mecanismo de cruzamento.

- **Cruzamento Uniforme** No cruzamento uniforme, todos os pontos são potencialmente pontos de cruzamento. Uma máscara é gerada aleatoriamente, e a paridade dos dígitos binários da máscara indica qual progenitor e que dígitos binários fornecerá aos descendentes [Sys89]. A Figura 5 mostra a aplicação do cruzamento uniforme.

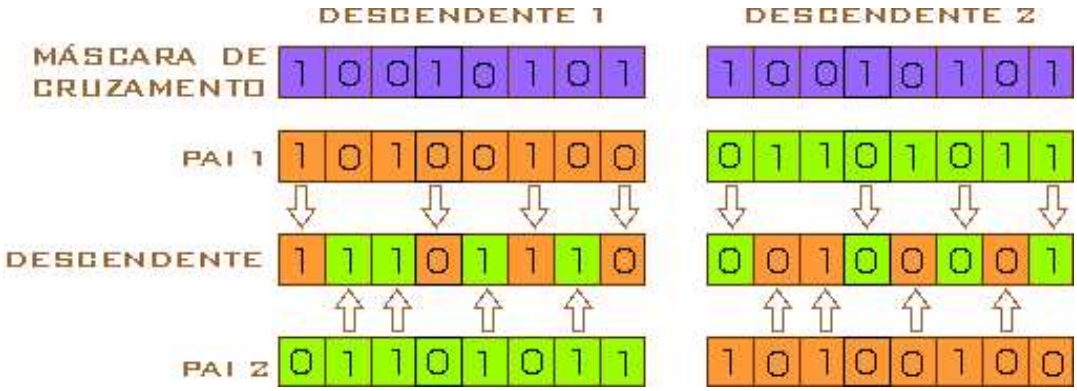

Figura 5: Esquema gráfico da aplicação de um cruzamento uniforme

2.1.4 Mutação

A mutação é um operador unário que consiste em perturbar ligeiramente (com uma probabilidade pequena) os indivíduos descendentes gerados pela recombinação. Uma vez que os indivíduos são representados por sequências de dígitos binários, cada dígito binário é trocado, aleatoriamente, de 0 para 1 ou vice-versa, de acordo com uma determinada probabilidade [Gol89a].

É importante destacar que o operador genético mutação não garante que um determinado caracter genético não tende a desaparecer na população. A Figura 6 mostra a aplicação do operador genético unário mutação onde o dígito binário na posição 4 é mutado.

2.1.5 Re-Inserção

Os novos indivíduos gerados devem ser inseridos na população. Porém, é necessário considerar o número de indivíduos descendentes e qual a política de escolha dos indivíduos da população que serão substituídos. Algumas políticas de substituição são mostradas a seguir [Gol89a]:

- **Pura** - Na re-inserção pura, o número de indivíduos descendentes gerados é igual ao

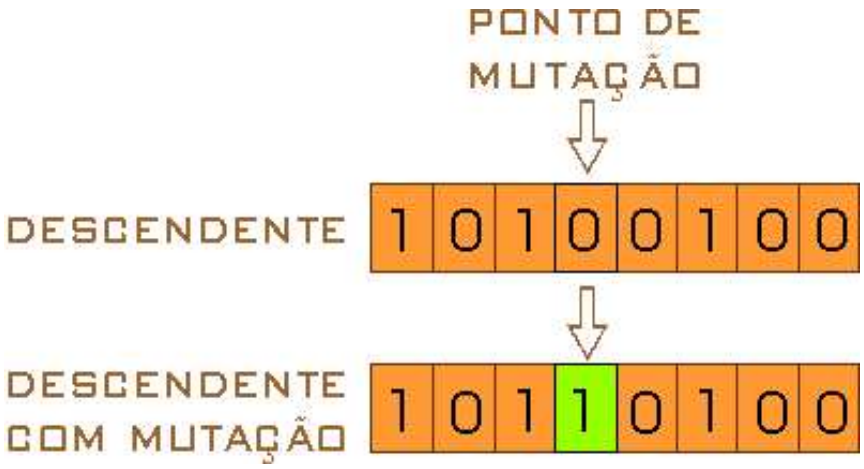

Figura 6: Esquema gráfico da aplicação do operador genético unário mutação

tamanho da população. No final de uma geração, a nova população será constituída exclusivamente pelos indivíduos descendentes. Este é o esquema de re-inserção utilizado pelo AG na sua forma mais simples.

- **Uniforme** - Na re-inserção uniforme, o número de indivíduos descendentes gerados é inferior ao tamanho da população. Os indivíduos descendentes vão substituir indivíduos da população escolhidos de forma aleatória.

- **Elitista** - Este esquema garante que os melhores indivíduos encontrados ao longo da procura, se encontrem presentes na população da última geração. Na re-inserção elitista, define-se uma elite, como sendo um conjunto composto pelos melhores indivíduos encontrados até o momento. Assim, o número de indivíduos descendentes gerados é inferior ao tamanho da população. Os indivíduos descendentes irão substituir os indivíduos da população que não fazem parte da elite.

2.1.6 Métodos de Niching

Quando se lida com problemas de otimização multimodal onde, muitas vezes, se quer encontrar diferentes soluções, é necessário implementar mecanismos de controle da diversidade populacional [Lar02]. Os principais métodos de *niching* que permitem tratar a questão da diversidade populacional são:

- *Sharing*[DEG87]. Este método de *sharing* é utilizado para distribuir as soluções da população ao longo da região das soluções ótimas. Ele promove a formação e manutenção de sub-populações de indivíduos, os nichos, com o objetivo de se encontrar

as diversas soluções pretendidas, baseando-se na idéia de que os indivíduos de um determinado nicho devem partilhar os recursos, isto é, a medida de desempenho. Assim, a função de *fitness sharing* decrementa o *fitness* de cada elemento proporcionalmente ao número de indivíduos similares na população, isto é, no mesmo nicho. Essa medida de similaridade é baseada tanto no genótipo (por exemplo a distância de Hamming) quanto no fenótipo (por exemplo a distância Euclidiana). Este método possui uma implementação simples e um custo computacional reduzido, porém implica em uma seleção antes da seleção real de indivíduos. Entre os trabalhos que utilizam este método estão [Pet, Pet97, DB93, BLM95].

- *Crowding* [Jon75]. A técnica de *crowding* insere novos elementos na população através da recolocação de elementos similares. O método de *crowding* implicitamente define uma vizinhança através das regras de torneio. Assim, quando um descendente é criado, um indivíduo é escolhido aleatoriamente para ser eliminado de um subconjunto populacional, ou seja, seleciona-se uma certa quantidade de indivíduos e aquele que for mais semelhante ao novo indivíduo é eliminado, e o novo indivíduo é incluído na população. Entre os trabalhos que utilizam este método estão [Pet, Pet97, MKD01, Mah92, BLM95, MB00].

- *Clustering.* O objetivo da técnica de *clustering* é agrupar unidades de dados em um *cluster* de tal forma que essas unidades de dados em conjunto com os *clusteres*, sejam os mais similares possível, enquanto que os *clusteres* sejam relativamente distintos uns dos outros. Através do método de *clustering* é possível identificar múltiplos globais e ótimos locais em um espaço de busca multimodal, onde a idéia básica é transferir o conceito biológico de não-produção de espécies, que evoluem em nichos ecológicos separados, para Algoritmos Evolucionários com o intuito de preservar a diversidade da população. Um trabalho que faz uso desta técnica é [FS94].

- *Paralelo.* Este método objetiva produzir múltiplas soluções, quando aplicado a problemas de otimização multimodal. Isto é feito através da divisão da população em subpopulações de maneira que cada subpopulação possa evoluir de forma paralela. Caso não haja comunicação entre essas subpopulações, o método paralelo se torna equivalente a evoluir cada subpopulação singularmente, tendo cada pequena população passando pelo processo de evolução muitas vezes. Em contrapartida, métodos paralelos aplicados a subpopulações que fazem uso da comunicação entre as subpopulações permitem compartilhar boas características dos indivíduos que serão propagados. Um trabalho que faz uso desta técnica esta descrito em [JPL02].

Um dos mecanismos mais utilizados, e que merece um destaque adicional, pois foi explorado na ferramenta de teste deste trabalho, é o da partilha da medida de avaliação de desempenho (*sharing*) [DEG87]. Este mecanismo promove a formação e manutenção de subpopulações de indivíduos, os nichos, com o objetivo de se encontrar as diversas soluções desejadas.

Este mecanismo baseia-se na idéia de que os indivíduos de um determinado nicho devem partilhar os recursos, isto é, a medida de avaliação de desempenho.

O mecanismo de *sharing* altera apenas o procedimento de atribuição do valor da função de avaliação de um indivíduo. Ele trabalha alterando a função de avaliação de cada elemento da população de acordo com o número de indivíduos semelhantes a ele na população [Mat00].

O compartilhamento da função de avaliação *fitness sharing* de um indivíduo, denominado **F**, é igual a sua função de avaliação **F** dividida por seu contador de nichos *(niching count)*. O contador de nichos é a soma dos valores das funções de compartilhamento *sh* entre ele e os demais indivíduos da população (incluindo ele mesmo).

A função de compartilhamento é gerada em função de uma distância **d** entre dois elementos da população, e retorna 1 se os elementos são iguais, 0 se a diferença entre eles é maior que um limiar de dissimilaridade, e um valor intermediário entre 0 e 1 de acordo com seus níveis de dissimilaridade.

Um limiar de dissimilaridade é especificado por uma constante denominada σ_{share}. Se a distância entre dois elementos da população for maior ou igual a σ_{share} eles não afetam o compartilhamento da função de avaliação um do outro.

Desta forma, a medida de avaliação de desempenho de um dado indivíduo é degradada na proporção do número de indivíduos que estejam localizados na sua vizinhança. Esta noção de vizinhança de um indivíduo implica na especificação de um parâmetro, o σ_{share}, que define o raio dos nichos. Para a determinação dos indivíduos contidos numa dada vizinhança é necessária a utilização de uma medida de distância, $\boldsymbol{d(x_i, x_j)}$, de dois indivíduos $\boldsymbol{x_i}$ e $\boldsymbol{x_j}$.

Matematicamente, a função de medida de avaliação de desempenho de *sharing*,

$\boldsymbol{F_{sharing}(x_i)}$ para um indivíduo $\boldsymbol{x_i}$ com $0 \leqslant i \leqslant (P-1)$ é dada por

$$\boldsymbol{F_{sharing}(x_i)} = \frac{\boldsymbol{F(x_i)}}{\sum_{j=0}^{P-1} Sh(d(x_i, x_j))},$$

onde $\mathrm{Sh}(d(x_i, x_j))$ é uma função de *sharing*. De modo geral, a distância entre dois indivíduos, $\boldsymbol{x_i}$ e $\boldsymbol{x_j}$ é medida através das variáveis de decisão, isto é, $\mathrm{d}(x_i, x_j) = \| x_i - x_j \|_2$.

No decorrer deste capítulo foi possível perceber a extensão da aplicabilidade dos Algoritmos Genéticos por meio de uma revisão teórica e exemplos. O uso dos AG's nos processos de otimização é algo há muito estudado, e não há dúvidas de que seja um método viável a ser usado neste contexto. Como alternativa aos AG's nos processos de busca em problemas de otimização utilizando funções multimodais, tem-se os algoritimos de estimação de distribuição, que é alvo de estudo do Capítulo 3.

3 Algoritmos de Estimação de Distribuição

Os Algoritmos de Estimação de Distribuição (AEDs) são algoritmos heurísticos de otimização que baseiam sua busca no carácter estocástico da mesma. Como nos algoritmos genéticos, também os AEDs estão baseados em populações que evoluem. Entretanto, a diferença entre os algoritmos genéticos e os AEDs é que nestes a evolução das populações não se dá por meio de operadores de cruzamento e mutação. No lugar destes a nova população de indivíduos provém de uma distribuição de probabilidade, a qual é estimada na base de dados contendo o conjunto de indivíduos selecionados que constituiam a geração anterior.

Enquanto que, em outras heurísticas utilizadas em Computação Evolucionária, as inter-relações entre as variáveis que representam os indivíduos são mantidas implicitamente *(building block hypothesis)*, nos AEDs as inter-relacões são expressas explicitamente por meio de associações entre a distribuição de probabiidade e os indivíduos selecionados em cada interação [Ben02].

De fato, a tarefa de estimar a distribuição de probabilidade associada com a base de dados contendo os indivíduos selecionados a partir de gerações anteriores, constitui o trabalho mais difícil de se executar.

Entre as vantagens desse novo tipo de algoritmo, está a idéia de tornar mais fácil a predição de movimentos da população no espaço de busca. Este tipo de algoritmo é também baseado na evolução da população como sendo um progresso na busca pela solução e, da mesma forma que os AGs, os AEDs possuem uma fundamentação teórica em teoria da probabilidade. Assim, AEDs são algoritmos de busca de base populacional baseados em um modelo probabilístico de soluções promissoras, que utiliza a simulação de modelos introduzidos para guiar o processo de busca [Ben02].

Um esquema genérico sobre os AEDs pode ser entendido a partir dos seguintes passos:

1. Primeiro, a população inicial D_0 dos R indivíduos é gerada. A geração dos R indivíduos usualmente é realizada assumindo-se uma distribuição uniforme sobre cada variável, e seguidamente cada indivíduo é evoluído.

2. Segundo, com o intuito de se gerar a $(l - 1)^a$ população, D_{l-1} evolui em direção ao D_l, e um número N ($N < R$) de indivíduos são selecionados a partir de D_{l-1} segundo um critério previamente estipulado. Assim, denota-se por D_{l-1}^N o conjunto dos N indivíduos selecionados a partir da geração $l - 1$.

3. Terceiro, o modelo probabilístico n-dimensional que melhor representa a interdependência entre as N variáveis é induzido. Este passo é conhecido como procedimento de aprendizagem, sendo o mais crucial, pois a representação apropriada das dependências entre as variáveis é essencial para a evolução apropriada dos indivíduos mais aptos.

4. Finalmente, a nova população D_l constituída pelos R novos indivíduos é obtida realizando-se a simulação da distribuição de probabilidade aprendida no item 3. Geralmente, uma aproximação usando elitismo é aplicada, e consequentemente o melhor indivíduo da população D_{l-1}^N é mantido em D_l.

   Os passos 2, 3 e 4 são repetidos até que uma condição de parada seja satisfeita. Entre as possíveis condições de parada estão: encontrar o número desejado de populações, encontrar o número desejado de diferentes indivíduos evoluídos, uniformidade na geração da população e a não obtenção de nenhum indivíduo com valor de avaliação de desempenho melhor após um certo número de gerações.

Os AEDs podem ser divididos em três diferentes abordagens [Ben02]:

1. **Variáveis Independentes**. Os Algoritmos de Distribuição Marginal Unidimensional (ADMU), Aprendizado Incremental Baseado em População (AIBP) e o Algoritmo Genético Compacto (AGc) estão nesse grupo. Esses algoritmos calculam a estimação da distribuição de probabilidade considerando as variáveis do problema como unidimensionais. Os algoritmos mencionados apresentam melhor desempenho em problemas onde as variáveis não possuem interações significativas entre si [Ben02].

2. **Dependências Bidimensionais**. Os Algoritmos Agrupamento de Entrada por Maximização de Informações Mutua *(Mutual Information Maximization for Input*

Clustering - MIMIC), Otimizadores Combinantes com Árvores de Informação Mutua (OCAIM) e Algoritmo de Distribuição Marginal Bidimensional (ADMB) pertencem a esse grupo. Esses algoritmos exploram as dependências de ordem dois entre as variáveis para prover melhores resultados em problemas cujas variáveis estão relacionadas [Ben02].

3. **Dependências Multidimensionais**. Os Algoritmos de Distribuição Fatorizada (ADF), Algoritmo Genético Compacto Extendido (AGCe), Algoritmo de Otimização Bayesiano (AOB) e o Algoritmo de Estimação de Rede Bayesiana (AERB) estão nesse grupo. Esses algoritmos são capazes de capturar dependências multidimensionais entre variáveis [Ben02].

3.1 Algoritmo PBIL

O Algoritmo Aprendizagem Incremental Baseado em População (AIBP) pertence à abordagem referente aos AED's que trata de variáveis tidas como **variáveis independentes**, isto é, variáveis que não possuem interações significativas entre si. O PBIL é fruto de uma combinação de Otimização Evolucionária e *Hillclimbing* [Bal94]. O algoritmo PBIL possui como objetivo criar um vetor de probabilidades de valores binários, o qual, pode revelar através de uma amostra, um alto valor de *fitness* dos vetores solução com alta probabilidade.

As fases de execução do PBIL são similares aos passos de execução do AG. As diferenças estão no fato de o PBIL usar vetores de probabilidade para descrever as populações, não usar *crossover* e, usar Aprendizagem Competitiva [SB95].

Os Algoritmos Genéticos possuem uma população implícita, a qual é criada partindo-se de uma geração para a seguinte. Já no algoritmo PBIL, por outro lado, somente o vetor de probabilidade é mantido durante as gerações, sendo que amostras de indivíduos são geradas de acordo com as probabilidades de cada interação. Em outras palavras, mais do que manter uma população de potenciais soluções, o PBIL mantém o modelo de probabilidade das regiões promissoras do espaço de busca[SB95].

Inicialmente, os valores do vetor de probabilidade são inicializados com 0.5. Conforme a execução prossegue, os valores do vetor de probabilidade vão se alterando gradualmente com a função de representar a melhor solução avaliada pelos vetores.

O algoritmo PBIL é caracterizado por três parâmetros. O primeiro é o número de in-

divíduos a serem gerados com base no vetor de probabilidades. O segundo parâmetro está relacionado com o valor utilizado pela taxa de aprendizagem. E como terceiro parâmetro está o número de indivíduos utilizados para atualizar o vetor.

Em relação ao primeiro parâmetro, que diz respeito ao número de indivíduos que são gerados com base no vetor de probabilidade, este valor está diretamente relacionado com o problema em questão. Poucos indivíduos na população podem fazer com que o algoritmo sofra uma convergência prematura para algum ponto local do espaço do busca. E se o número de indivíduos for muito grande, corre-se o risco do algoritmo não convergir para nenhum ponto do espaço de busca.

Para o parâmetro de aprendizagem, recomenda-se um valor pequeno [Sha03]. O parâmetro de aprendizagem controla a relação entre confiabilidade e velocidade de convergência. Enquanto nos algoritmos evolucionários tradicionais, os indivíduos velhos são substituídos por indivídos novos instantaneamente, no PBIL há o uso da estratégia de aprendizagem incremental, onde se modifica cautelosamente o modelo, mantendo-se dessa forma uma memória de longo prazo [SB95]. A explicação para isso, é que o vetor de probabilidade não representa um indivíduo isolado no espaço de busca. O vetor representa todo o espaço de busca com certa polarização em determinadas áreas.

Se o parâmetro de aprendizagem for muito grande, o vetor de probabilidade convergirá rapidamente para 0 ou 1 em cada elemento, reduzindo drasticamente todo o potencial do PBIL.

E quanto ao terceiro parâmetro, que diz respeito ao número de indivíduos usados para atualizar o vetor, algumas versões do PBIL usam o melhor indivíduo para a atualização, enquanto outras selecionam mais de um dentre um conjunto previamente escolhido.

No Algoritmo 1 está uma estrutura referente ao algoritmo PBIL [Bal94]:

Onde:

São Constantes Definidas pelo Usuario:

GERAÇÕES: Numero de iterações permitidas.
Amostras: Tamanho da população, numero de individuos produzidos por geração.
Tamanho: Tamanho da solução codificada.
Mut_Probabilidade: Probabilidade da mutação ocorrer em cada posição.
Mut_Shift: Valor da mutação que afeta o vetor de probabilidade.
TA: Taxa de aprendizagem.

Algoritmo 1 Algoritmo PBIL básico

$P \leftarrow$ Inicializa Vetor de Probabilidade. (Cada posição = 0.5)
{Laço até fim das gerações} {Gerar Amostras}
$i \leftarrow$ laço {Amostras}
$Amostra_i \leftarrow$ Gera vetor de acordo com as probabilidades de P
$Avaliacao_i \leftarrow$ avalia($Amostra_i$) {Encontrar melhor amostra}
$Max \leftarrow$ Encontrar vetor correspondente a maxima avaliação {Atualizar o vetor de probabilidade}
$I \leftarrow$ laço {Tamanho}
Se random(0,1] < Mut_Probabilidade {Mutação do Vetor de Probabilidade}
$I \leftarrow$ laço {Tamanho}
if random(0,1] < Mut_Probabilidade **then**
  $P_I \leftarrow P_I * (1.0 - Mut\_Shift) + random(0.0ou1.0) * (Mut\_Shift)$
end if

Variaveis:

P: Vetor de probabilidade.

$Amostra_i$: Vetor solução.

$Avaliacao_i$: Avaliações dos vetores solução.

Max: Vetor solução correspondente a avaliação maxima.

No decorrer deste capítulo pode-se entender as características que tornam os AED's métodos robustos e flexíveis. Dentre as abordagens verificadas nos AED's, destaca-se a que trata de variáveis independentes, mais precisamente o Algoritmo PBIL.

Por meio da análise do algoritmo PBIL e seus parâmetros de configuração, pode-se entender o processo de criação do vetor de probabilidade e sua fórmula de inicialização, e partir destes processos compreender as modificações realizadas ao PBIL que levaram à concretização do Método Multi-PBil.

O capítulo 4 traz um estudo minucioso e preciso do Método Multi-PBil. Inicialmente faz-se uma recordação dos pontos principais do Algoritmo PBIL. Posteriormente destaca-se o processo de construção dos modelos de probabilidade, a fórmula utilizada para inicializar tais modelos e uma explicação precisa sobre as fases de execução do método, incluindo os parâmetros utilizados e operadores genéticos trabalhados.

Finalmente, encerra-se o capítulo com uma seção abordando uma heurística trabalhada no processo de eliminação em tempo real dos modelos de probabilidade com desempenho insatisfatório.

4 O Método Multi-PBil

Neste capítulo são apresentados os principais fundamentos do método Multi-PBIl, incluindo suas definições, procedimentos, vantagens, parâmetros de configuração e demais tópicos importantes relacionados ao método.

4.1 Extensões do PBIL para o Multi-PBil

O PBIL é um dos mais simples Algoritmos Evolucionários baseados em modelos, os quais não possuem dependências entre variáveis. O modelo probabilistico usado é um vetor de valores reais onde cada elemento representa independentemente a probabilidade de se gerar o valor 1 em cada bit correspondente.

O algoritmo PBIL permite representar toda a base populacional genética por meio de um vetor de probabilidades. Um vetor de probabilidade é simplesmente uma sequência de probabilidades. Cada valor probabilistico na sequência representa a probabilidade de se gerar 1 ou 0 na posição gênica.

Por meio de um cenário, a representação básica da solução em um algoritmo PBIL tradicional pode ser a mesma de um algoritmo genético, porém ao invés de guardar cada possibilidade explicitamente, a população é substituída por uma distribuição de probabilidade.

O que o PBIL atualiza é muito mais a distribuição de probabilidade do que uma população fixa. A forma utilizada para realizar a atualização é encontrar um bom candidato à solução, e então incrementar a probabilidade de cada um dos valores de seus alelos na distribuição.

A regra para a atualização dos valores de probabilidade podem ser bem simples, e são usualmente amarradas a uma taxa de aprendizagem. A taxa de aprendizagem determina o quanto um valor de probabilidade sobre um dado alelo deve ser incrementado, e assim, por quanto as probabilidades restantes do valor devem ser reduzidas.

Este trabalho apresenta uma extensão do PBIL. Esta versão está voltada para problemas multimodais e combina algumas características adicionais: 1) Uma fórmula usada para criar muitos modelos de probabilidade, os quais permitem ao método encontrar diversos objetivos simultaneamente; 2) Uma nova regra para inicialização dos modelos, que permite iniciar os modelos com valores próximos de 0 (zero) ou 1 (um), fazendo com que os modelos sejam iniciados próximos dos objetivos no espaço de busca; e 3) um esquema de eliminação dos modelos não satisfatórios durante a execução do algoritmo.

4.2 Construindo Modelos de Probabilidade a Partir da População Inicial

4.2.1 Inicialização da População

Inicialmente, quando se trabalha com computação evolucionária, mais especificamente com Algoritmos de Estimação de Distribuição, o primeiro passo é constituir uma população de indivíduos.

Os indivíduos da população representam uma possível solução para o problema, ou seja, um ponto no espaço de soluções, e são gerados por meio de uma função matemática obedecendo a um formato específico.

Cada indivíduo possui um valor de aptidão o qual determina o seu potencial.

Uma vez que se tenha o conjunto de indivíduos, deve-se calcular o valor de fitness de todos os indivíduos. O cálculo do valor de fitness é feito por meio de um critério específico relacionado ao problema.

Os indivíduos são avaliados e ordenados decrescentemente, isto é, o indivíduo com o maior valor de fitness ocupa a primeira posição; este é seguido pelo indivíduo com o segundo maior valor de fitness, e desta forma se segue até o último indivíduo.

4.2.2 Criação dos Modelos de Probabilidade

O método Multi-PBil usa o conjunto de indivíduos para criar os modelos de probabilidade. Obrigatoriamente, o primeiro indivíduo da população é usado para criar o primeiro modelo de probabilidade.

Os modelos de probabilidade são criados por meio da equação:

$$\boldsymbol{M_i} = (1 - \alpha) * \boldsymbol{X_i} + \alpha * (\text{media da metade dos melhores individuos})$$

onde:

- $\boldsymbol{M_i}$ é o valor binário da posição corrente **i** do modelo criado;
- α é o parâmetro de aprendizagem;
- $\boldsymbol{X_i}$ é o valor binário da posição corrente do indivíduo.

O critério usado para criar os modelos subsequentes segue uma estrutura condicional.

A partir do segundo indivíduo até o último, realiza-se um teste com o objetivo de verificar se os indivíduos podem também criar os modelos de probabilidade seguintes.

A estrutura condicional utiliza um parâmetro chamado **Fator de Criação ($\boldsymbol{F_c}$)**, para verificar a probabilidade de cada indivíduo ser usado para criar um novo modelo de probabilidade.

O $\boldsymbol{F_c}$ representa um termômetro na criação de poucos ou muitos modelos de probabilidade. Isto é, quanto maior for o valor do fator de criação, maior será o número de modelos de probabilidade criados. E, inversamente, quanto menor for o valor do fator de criação, menor será o número de modelos criados.

Para calcular as possibilidades do segundo indivíduo criar o modelo de probabilidade 1, utiliza-se um cálculo que mede a quantidade de posições equivalentes que possuem valores iguais entre o segundo indivíduo e o primeiro indivíduo. Este cálculo compara todas as posições ocupadas por bits dos indivíduos 1 e 2, e calcula o número de posições onde os bits são iguais.

A Tabela 2 mostra um exemplo de um cálculo que mede a proximidade do indivíduo 2 com o indivíduo 1.

Tabela 2: Comparação entre dois indivíduos

	0	1	2	3	4	5	6
Ind 1	1	0	1	1	0	1	0
Ind 2	0	1	1	1	0	0	1

Na Tabela 2, o indivíduo 1 e o indivíduo 2 apresentam 3 posições onde os bits são iguais. São as posições 2, 3 e 4. Nas posições 2 e 3 ambos os bits dos indivíduos 1 e 2 possuem valor 1, e na posição 4, os bits dos indivíduos 1 e 2 possuem valor 0.

Então, através da equação :

$$\boldsymbol{P_2} = (\mathbf{p} * 100)/(\mathbf{t}).$$

onde:

- $\boldsymbol{P_2}$ é o valor de proximidade do indivíduo 2;
- $\mathbf{p}$ é a quantidade de posições com bits iguais existentes entre dois indivíduos;
- $\mathbf{t}$ é o tamanho total do indivíduo em número de bits.

Substituindo os valores referentes à Tabela 2, tem-se:

$$\boldsymbol{P_2} = (3 * 100)/7; \boldsymbol{P_2} = 43.$$

Assim, o indivíduo 2 apresenta uma proximidade com o indivíduo 1 com valor 43.

Se o valor de $\boldsymbol{P_2}$ for menor que $\boldsymbol{F_c}$ então, o indivíduo 2 dará origem a um novo modelo de probabilidade 2. Caso contrário, se o valor de $\boldsymbol{P_2}$ for maior ou igual a $\boldsymbol{F_c}$, então o indivíduo analisado também apresenta grande probabilidade de criar o modelo de probabilidade 1. Assim sendo, o inivíduo 2 é descartado e passa-se a analisar o indivíduo 3.

A leitura dos indivíduos segue até que o último tenha sido analisado.

Generalizando, para cada indivíduo $\boldsymbol{X_i}$ da população, com $i > 1$, realiza-se um teste para saber se esse indivíduo em questão também pode gerar o último modelo de probabilidade criado até então.

Desta forma, a cada execução do algoritmo com o objetivo de criar os modelos de probabilidade, pode-se ter um número quase sempre diferente de modelos criados, isto é, a quantidade de modelos de probabilidade criados está diretamente relacionada com a diversidade da população inicial.

No Algoritmo 2 está a subrotina responsável pela criação dos modelos.

Algoritmo 2 Subrotina criacao dos modelos

for individuo $X_i = 2$ ate N **do**
  if $max\ (P_r(X_i|M_i(k)), k = 1...n) < F_c$ **then**
    Cria-se um novo modelo de probabilidade
  else
    X_i++
  end if
end for

4.3 Fases do Algoritmo Multi-PBil

Cada um dos modelos de probabilidade criados dá origem a uma população de indivíduos. Cada população de indivíduos tem como função encontrar o ponto ótimo procurado através do espaço de busca.

Cada modelo de probabilidade possui o seguinte formato inicial:

Tabela 3: Exemplo de um modelo de probabilidade inicial

0	1	2	3	...	N
X_i	X_{i+1}	X_{i+2}	X_{i+3}	...	X_{i+n}

Onde o termo X_i pertence ao intervalo [0,1], para $0 < i < n - 1$.

Cada modelo de probabilidade gera a sua respectiva população de indivíduos.

Inicialmente, a primeira posição do modelo de probabilidade é responsável por gerar o valor da primeira posição do vetor de cada indivíduo. A segunda posição do modelo de probabilidade é responsável por gerar o valor da segunda posição do vetor de cada indivíduo. E assim, sucessivamente, até que a última posição do modelo de probabilidade gere o último valor da última posição do último indivíduo. E esse processo é repetido para cada um dos modelos de probabilidade criados.

4.4 Evolução e Função de Aptidão

Cada modelo de probabilidade trabalha com uma população própria de indivíduos. Desta forma, cada modelo faz sua atualização independente dos demais modelos, isto é,

o modelo **M1** seleciona um indivíduo de sua população utilizando um método de seleção, para ser usado em sua atualização; o modelo **M2** seleciona um outro indivíduo de sua população para usar em sua atualização; e esse processo de atualização segue até que todos os modelos tenham sido atualizados.

A função de aptidão permite avaliar os indivíduos da população. Esta avaliação considera o valor de cada indivíduo, segundo os critérios da função estabelecida.

Depois que todos os indivíduos da população foram avaliados e ordenados segundo seus valores de aptidão, um destes é escolhido para atualizar o modelo de probabilidade.

A escolha do indivíduo para a atualização do modelo de probabilidade é feita utilizando um método de seleção.

A atualização é feita utilizando a seguinte equação 4.4:

$$P_{i+1} = P_i * (1.0 - \alpha) + X_i * (\alpha)$$

onde:

- P_{i+1} é o valor da posição i do vetor de probabilidade após a atualização;
- P_i é o valor da posição atual i do vetor de probabilidade;
- $\boldsymbol{\alpha}$ é o parâmetro de aprendizagem fornecido durante a inicialização do processo;
- X_i é o valor binário na posição i do indivíduo.

A Equação 4.4 é aplicada para todos os modelos de probabilidade.

O ciclo do método Multi-PBil é composto dos seguintes passos:

1. Geração dos indivíduos;
2. A avaliação do fitness de cada indivíduo baseada numa função de fitness de um problema em particular;
3. A criação dos modelos de probabilidades;

4. Para cada modelo de probabilidade criado:

   (a) Inicializar os modelos de probabilidade por meio da fórmula 4.2.2;

   (b) Gerar uma população de N indivíduos para cada modelo de probabilidade, e aleatoriamente determinar o gene de cada posição de cada indivíduo (1 ou 0);

   (c) Avaliar o fitness de cada indivíduo;

   (d) Ordenar os indivíduos pelo valor de fitness;

   (e) Atualizar os valores probabilísticos de cada posição do modelo, baseado em um indivíduo selecionado por uma função utilizando uma taxa de aprendizagem;

   (f) Verificar se o modelo de probabilidade convergiu. Se não, repetir os passos (b) até (e).

O ciclo do algoritmo Multi-PBil continua até que um critério de parada seja satisfeito. Os critérios de parada para cada modelo de probabilidade podem ser:

1. O algoritmo executa até que o número total de gerações termine;

2. O algoritmo executa até que um objetivo seja encontrado;

3. O algoritmo executa até que o modelo seja eliminado;

4.5 O Operador Mutação

A principal função de um operador genético é introduzir variedade em uma população, através de sucessivas gerações, estendendo a busca até chegar a um resultado satisfatório. O operador muda o valor contido no genoma de acordo com uma distribuição de probabilidade dada.

Com o operador mutação, o algoritmo pode ser capaz de encontrar uma solução melhor que uma previamente possível. A mutação é uma parte importante da busca porque ajuda a prevenir a população de se estagnar em um ótimo local. A mutação ocorre durante a evolução, de acordo com uma probabilidade de mutação definida pelo usuário.

O operador mutação pode ser aplicado em um conjunto de indivíduos da população e/ou em um modelo de probabilidade. O impacto causado em cada um deles é relativo ao deslocamento conseguido no espaço de busca.

Assim, se a mutação for aplicada em um indivíduo da população, o deslocamento causado por essa operação fará com que esse indivíduo escape de mínimos locais. Porém, se a aplicação do operador mutação for no modelo de probabilidade, esta causará um movimento de toda a população pelo espaço de busca, criando desta forma condições para que mais de um indivíduo possa encontrar o ponto ótimo global procurado [Bal94].

A aplicação do operador mutação segue a seguinte regra. Para todos os elementos do modelo de probabilidade, se um valor no intervalo (0, 1], gerado aleatoriamente for menor que Mut_prob, então a posição onde ocorreu essa verdade sofre um movimento segundo a Equação:

$$P_i = P_i * (1.0 - Mut\_Sh) + rand(0.0\_ou\_1.0) * (Mut\_Sh)$$

onde:

- **rand()** é uma função matemática de geração de números aleatórios
- **Mut_Sh** é uma constante definida pelo usuário, que indica a taxa de deslocamento do operador mutação quando aplicado ao modelo de probabilidade.

A seguir está o Algoritmo 3 responsável pela subrotina de aplicação do operador mutação:

Algoritmo 3 Subrotina Mutação

```
for cada posicao i do modelo do
  if (rand(0,1] < Mut_Prob) then
    P_i = P_i * (1.0 - Mut_Sh) + rand(0.0_ou_1.0) * (Mut_Sh)
  end if
end for
```

onde:

- **Mut_Prob** é uma constante definida pelo usuário, que indica a probabilidade do operador mutação ocorrer em cada posição.

4.6 Eliminação dos Modelos de Probabilidade

A execução do algoritmo Multi-PBil inicia-se com uma quantidade de modelos de probabilidade superior ao número de objetivos procurados. A idéia é que durante a

execução do algoritmo os modelos de probabilidades em excesso sejam eliminados, com o intuito de otimizar a execução.

Para realizar essa tarefa, implementou-se uma medida de proximidade entre os modelos de probabilidades e os objetivos. Essa medida de proximidade, que é utilizada para a eliminação dos modelos de probabilidade em excesso, é aplicada aos modelos cujos valores estejam próximos de um objetivo $\boldsymbol{O_i}$ já encontrado.

Quando um objetivo é encontrado, todos os modelos de probabilidade que possuem um valor próximo ao valor do objetivo encontrado são eliminados. Essa medida usa uma Distância Euclidiana entre os valores do objetivo encontrado $\boldsymbol{O_i}$ e os valores dos modelos de probabilidade. Quando o valor do modelo de probabilidade possui uma grande porcentagem de proximidade com o valor do objetivo, então o modelo é eliminado.

Uma vez que o modelo já encontrou o objetivo $\boldsymbol{O_i}$, não é de interesse do método que outro modelo também encontre este objetivo $\boldsymbol{O_i}$. Desta forma, todos aqueles modelos que estiverem próximos de encontrarem o objetivo $\boldsymbol{O_i}$ devem ser eliminados.

Os modelos de probabilidade podem se situar em três estados distintos durante a execução do algoritmo. São eles, o estado de **estável**, o estado **eliminado**, e o estado de **sucesso**. Durante o estado **estável**, o modelo permanece em um processo constante de busca pelo objetivo, e situa-se dentro de um espaço de busca aceitável.

O estado **eliminado** ocorre quando o modelo se encontra em uma situação onde um determinado objetivo $\boldsymbol{O_i}$ foi encontrado, e o valor decimal de um modelo $\boldsymbol{M_i}$ qualquer esta mais próximo de $\boldsymbol{O_i}$ do que qualquer outro objetivo.

Por fim, o estado de **sucesso** ocorre quando um modelo tem em sua população algum indivíduo que possui a forma idêntica a um dos objetivos procurados. Quando esta situação ocorre, significa que o modelo encontrou um objetivo do conjunto.

5 *Experimentos*

O método Multi-PBil foi implementado e aplicado a um conjunto de problemas com diferentes configurações. Os resultados são comparados com um Algoritmo Genético usando o método de *sharing*.

5.1 Simulador

O simulador de objetivos foi elaborado com a função de fornecer um conjunto de problemas, os quais foram usados para validar o método.

O simulador gera um conjunto de N vetores. Os vetores possuem um tamanho M. As variáveis N e M são parâmetros definidos durante a inicialização do processo.

Os vetores são gerados por meio de uma função matemática que gera números aleatórios uniformemente distribuídos. Os campos dos vetores são codificados em uma sequência de 0's (zeros) ou 1's (um(s)), isto é, são representados em um vetor de números binários.

Os objetivos são gerados nas quantidades de 10 a 100 em intervalos de 10 em 10; E são configurados para apresentarem distâncias entre si. Essas distâncias apresentam valores mínimos no intervalo de 5, 10, 20, 30, 40 e 50 entre os objetivos. E valores máximos limitados somente pelo tamanho do objetivo.

Para preservar a distância mínima requerida ao conjunto de objetivos gerados, aplicou-se um método de verificação, que compara cada posição equivalente entre um objetivo e os demais objetivos do conjunto. Cada comparação deve retornar um número que deve ser igual ou maior à distância mínima requerida. Isto é, o número de bits diferentes apresentados pelos dois objetivos quando comparadas suas posições equivalentes deve igual ou maior ao valor de distância mínima estipulado.

Quando a distância mínima entre dois objetivos for menor que o estipulado, realiza-se uma operação de correção de valor. Entre os dois objetivos que apresentaram a distância

menor que o mínimo estipulado, um deles é escolhido aleatoriamente e tem os valores de seus campos modificados também aleatoriamente.

A cada modificação feita no valor dos campos do objetivo, uma nova comparação é realizada com o intuito de verificar se a distância mínima se encontra dentro do limite estipulado. Este processo de correção de valores se repete até que todos os objetivos do conjunto apresentem distâncias mínimas entre si dentro do estipulado pelo problema.

5.2 Codificação, Funcão Fitness e Método de seleção

Os indivíduos são gerados no formato binário, e são configurados para diferentes problemas com tamanhos de 100 bits, 500 bits e 1000 bits. O total de indivíduos gerados para os testes foram de 500 indivíduos.

A função de fitness usada para calcular a aptidão de cada indivíduo da população utilizou o conceito de Distância de Hamming (DH) [WBPN98].

A função de fitness mede a quantidade de bits iguais em posições equivalentes entre os indivíduos da população e os objetivos do conjunto.

Cada indivíduo da população é comparado com todos os objetivos gerados pelo simulador. A comparação deve retornar um valor que represente o maior número de posições equivalentes com bits iguais. Esse valor representa o valor de aptidão do indivíduo.

O método de seleção utilizado pelo algoritmo Multi-PBil foi um método de seleção por torneio.

O número total de indivíduos presentes em cada população foi de 500 indivíduos. A taxa de porcentagem utilizada pelo método de seleção por torneio foi de 10%. Assim, a cada ciclo do algoritmo Multi-PBil, sorteava-se sempre 50 indivíduos aleatoriamente, para que, aquele dentre os 50 que apresentasse o maior valor de aptidão fosse escolhido para atualizar o modelo de probabilidade.

5.3 Descrição do Problema

O método foi testado, medido e observado após 50 execuções. A intenção foi de observar se o método poderia desenvolver um comportamento de sucesso em diferentes configurações de problemas.

A geração dos objetivos seguiu um critério não-uniforme na execução do teste. A intenção desse critério é buscar uma medida de sucesso comparada com diferentes instâncias de problemas, gerando objetivos muito próximos e objetivos mais distantes, e observando o quanto isto afeta a convergência do método.

Para que os objetivos fossem criados com a configuração desejada, implementou-se uma rotina que interfere na geração, de forma que essa geração seja não-uniforme. Caso a geração dos objetivos seguisse uma característica uniforme dos bits, isto é, 0's (zeros) e 1's (um(s)) gerados com a mesma probabilidade, como consequência não se teria um conjunto de objetivos com as características desejadas.

A maneira utilizada para que os objetivos fossem gerados, de forma a criá-los com as características desejadas, implicou na utilização de um algoritmo gerador de números aleatórios *MersenneTwister* da biblioteca para computação científica do projeto GNU (General Public License) [Mis05].

A estratégia utilizada aplicou uma manipulação na quantidade de zeros e uns gerados para cada objetivo. Quando dois objetivos apresentam uma diferença menor que o valor estipulado, aplica-se uma manipulação binária em um dos objetivos com o intuito de tornar esse valor igual ao mínimo desejado.

Os objetivos foram separados em grupos de 10 em 10, iniciando primeiramente a busca com um conjunto de 10 objetivos, depois 20, e chegando ao limite de 100 objetivos.

A Taxa de Eliminação usada pelo método foi de 95%, isto é, a soma dos campos do modelo de probabilidade deve estar a menos de 95% do valor da soma dos campos de cada objetivo. Se a soma exceder 95%, então o modelo é eliminado.

Utilizou-se também um teste estatístico com o objetivo de medir o nível de significância dos resultados obtidos pela implementação do método Multi-PBil [Cos88, Spi72]. Este teste foi aplicado para os conjuntos de objetivos configurados com distâncias mínimas no valor de 5 e 10 bits, pois estes conjuntos de valores representaram o maior desafio ao método Multi-PBil no intuito de identificar objetivos muito próximos.

Para realizar essa avaliação, escolheu-se um **nível de significância (α)** que define a probabilidade de, através do teste, aceitar ou não a hipótese formulada para o problema tratado.

Os testes aplicados aos dados estatísticos foram testes unilaterais com direcionamento para a esquerda, isto é, verificar se os valores obtidos pela implementação do método demonstram sua realidade, ou apresentam uma realidade menos precisa que a demonstrada,

isto é, um valor menor que o obtido.

Os procedimentos para a aplicação do **Teste de Hipótese** seguiram os seguintes passos:

1. Explicitar as hipóteses H_o e H_1;
2. Fixar α e identificar a variável do teste (distribuição de probabilidade a ser adotada);
3. Determinar a região crítica em função da variável tabelada;
4. Calcular o valor do teste, a partir da(s) amostra(s);
5. Decidir sobre as hipóteses.

Para obter o valor do teste calculado $(\boldsymbol{Z_c})$ em cada instância utilizou-se a seguinte fórmula:

$$(\boldsymbol{Z_c}) = \frac{(\overline{x} - \mathbf{MED})}{(\mathbf{DP}/\sqrt{n})}$$

onde:

- $\boldsymbol{Z_c}$ representa o valor do teste calculado;
- $\overline{x}$ representa a média obtida nas amostras;
- **MED** representa a média do conjunto de objetivos;
- **DP** representa o desvio-padrão do conjunto de objetivos;
- **n** representa o número de objetivos trabalhados;

A Tabela 4 mostra os parâmetros de configuração usados pelo algoritmo Multi-PBil.

Um algoritmo genético usando *"sharing"* foi usado com o intuito de se comparar com os resultados obtidos pelo método Multi-PBil. Foi utilizada a ferramenta *"EO Evolutionary Computation Framework"* [jeg05]. A ferramenta EO é baseada em *"templates"*, que utiliza bibliotecas de computação evolucionária.

Foram realizadas 50 execuções na ferramenta para cada configuração, utilizando a mesma base de dados, isto é, o conjunto de objetivos com as mesmas características de distâncias utilizados no método Multi-PBil.

Tabela 4: Configuração dos parâmetros usados pelo Multi-PBil

Parâmetros	**Valor**
Quantidade de modelos	10, 20, 30, 40, 50, 60, 70, 80, 90 e 100 modelos
Tamanho dos Indivíduos	100 bits, 500 bits e 1000 bits
Quantidade de Indivíduos	500
Taxa de Aprendizagem	0.005
Taxa de Eliminação	95%
Seleção por torneio (k)	10%
Fator de criação (Fc)	80
Evolução	200 gerações
Mut_Prob	0.02
Mut_Sh	0.05

O objetivo dos testes foi de observar o comportamento de sucesso utilizando indivíduos com tamanho 100 bits, 500 bits e 1000 bits.

A Tabela 5 mostra os parâmetros de configuração usados na ferramenta EO pelo algoritmo genético.

Tabela 5: Configuração dos parâmetros usados pela ferramenta EO

Parâmetros	**Valor**
Tamanho da população	500 indivíduos
Número de descendentes	100%
Tamanho do Cromossomo	100 bits, 500 bits e 1000 bits
Gerações	200
Prob. Crossover	0.6
Prob. Mutação	0.1
Taxa de crossover 1 ponto	1
Taxa de crossover 2 pontos	1
Crossover uniforme	2
Prob. de mudança de 1 bit na mutação	0.01

5.4 Resultados Experimentais

As Tabelas 6, 7, 8, 9, 10, 11, 12, 13 e 14 apresentam as médias de sucesso para cada instância do problema.

Os testes mostram uma relação proporcional entre a distância e o número de objetivos procurados. Para objetivos com distâncias pequenas entre si, o método apresenta bons resultados quando o número de objetivos é pequeno. À medida que o número de objetivos vai aumentando, o método começa a apresentar uma diminuição no desempenho.

Os objetivos com uma distância mediana, a partir do valor 20, apresentam um melhor desempenho em relação aos objetivos com distâncias menores, isto é, quando comparados com objetivos com distâncias 5 e 10.

Objetivos com uma distância grande entre si, por volta de 30 a 50, apresentaram um comportamento de sucesso, sendo limitados somente pela quantidade de objetivos procurados no espaço de busca, e também pelo tamanho dos objetivos.

O valor do fator de criação **(Fc)** foi utilizado em sua capacidade máxima, isto é, com valor 80. Com este valor, houve uma alteração na quantidade de modelos gerados para cada conjunto de objetivos.

Com indivíduos de tamanho 100 bits, a implicação de um **Fc** no valor 80 fez com que a média de modelos criados pelo algoritmo fosse de 327 modelos. Para indivíduos de tamanho 500 bits, a média de modelos criados pelo algoritmo foi de 211 modelos. E para indivíduos de tamanho 1000 bits, a média de modelos criados foi de 170 modelos.

A tática de se manter o valor de **Fc** em sua capacidade máxima foi utilizada com o intuito de se ter sempre um número de modelos acima do número de objetivos procurados, pois quanto mais modelos no espaço de busca, maiores as chances de se encontrar todos os objetivos. Porém, ainda permanece de forma intuitiva e empírica a descoberta da quantidade ideal de modelos de probabilidade necessários para uma boa execução.

Para indivíduos com tamanhos 500 bits e 1000 bits, e objetivos com grandes distâncias, o algoritmo apresentou um comportamento semelhante ao demonstrado quando se trabalhou com indivíduos de tamanho 100 bits.

Notou-se um desempenho melhor quando a quantidade de objetivos e modelos era pequena. À medida que se aumenta a quantidade de objetivos e, permanece pequena a distância entre os objetivos, o desempenho diminui.

Tabela 6: Média de sucesso para indivíduos de tamanho 100 bits parte I

Tamanho 100 bits										
	Distância entre os Objetivos									
Obj	D=5	DMi	DMa	Med	DP	D=10	DMi	DMa	Med	DP
10	100	5	98	66	8.63	100	10	72	38	5.54
20	100	5	93	52	5.19	100	10	90	47	5.61
30	100	5	99	48	6.14	100	10	98	52	4.57
40	100	5	94	48	3.40	100	10	98	59	4.14
50	100	5	95	57	3.75	100	10	94	50	3.54
60	98	5	98	50	3.65	100	10	98	53	3.08
70	96	5	99	54	3.20	99	10	98	51	3.28
80	92	5	98	51	2.96	96	10	99	59	2.88
90	91	5	98	51	2.79	94	10	97	51	2.62
100	89	5	98	54	2.86	92	10	99	51	2.65

As variáveis das Tabelas 6, 7, 8, 9, 10, 11, 12, 13 e 14 indicam **DMi** como Distância Mínima, **DMa** como Distância Máxima, **Méd** como Média dos Objetivos, **(D=)** como a distância trabalhada no conjunto de objetivos e **DP** como Desvio Padrão dos Objetivos.

Tabela 7: Média de sucesso para indivíduos de tamanho 100 bits parte II

Tamanho 100 bits										
	Distância entre os Objetivos									
Obj	D=20	DMi	DMa	Med	DP	D=30	DMi	DMa	Med	DP
10	100	20	97	60	6.48	100	30	93	62	5.42
20	100	20	97	67	5.15	100	30	92	59	4.33
30	100	20	96	54	4.41	100	30	97	62	3.67
40	100	20	99	58	3.94	100	30	99	70	3.43
50	100	20	98	54	3.67	100	30	99	66	3.45
60	100	20	95	57	2.42	100	30	99	63	2.59
70	100	20	99	60	2.67	100	30	99	63	2.47
80	99	20	99	60	2.56	100	30	99	61	2.13
90	98	20	98	58	2.36	99	30	99	67	2.15
100	87	20	99	61	2.40	99	30	97	67	1.84

Tabela 8: Média de sucesso para indivíduos de tamanho 100 bits parte III

Tamanho 100 bits										
	Distância entre os Objetivos									
Obj	D=40	DMi	DMa	Med	DP	D=50	DMi	DMa	Med	DP
10	100	40	94	64	5.71	100	50	99	68	4.93
20	100	40	97	69	3.40	100	50	94	68	3.45
30	100	40	99	75	3.01	100	50	98	73	2.53
40	100	40	99	69	2.87	100	50	99	74	2.38
50	100	40	98	73	2.29	100	50	99	73	2.09
60	100	40	98	68	2.42	100	50	97	74	1.83
70	100	40	99	70	2.14	100	50	98	75	1.69
80	100	40	99	69	1.69	100	50	99	71	1.54
90	100	40	99	70	1.68	100	50	99	74	1.54
100	100	40	99	70	1.76	100	50	99	73	1.37

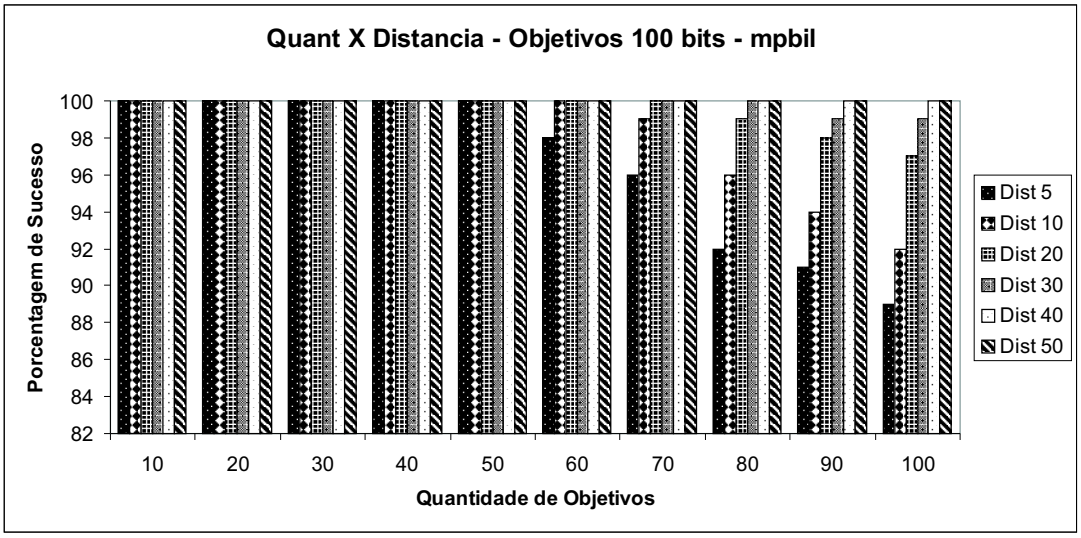

Figura 7: Gráfico com a média de sucesso do Alg. MPBil para 100 bits

Tabela 9: Média de sucesso para indivíduos de tamanho 500 bits parte I

Tamanho 500 bits										
	Distância entre os Objetivos									
Obj	D=5	DMi	DMa	Med	DP	D=10	DMi	DMa	Med	DP
10	100	5	407	203	40.67	100	10	420	198	44.72
20	100	5	492	222	32.93	100	10	499	258	36.94
30	100	5	499	285	26.09	100	10	492	236	23.59
40	100	5	477	254	20.93	100	10	498	240	25.15
50	99	5	474	219	17.50	100	10	498	272	20.76
60	94	5	475	213	17.54	99	10	494	244	19.75
70	92	5	497	242	15.94	98	10	499	231	16.47
80	90	5	498	249	15.80	94	10	468	231	16.07
90	87	5	489	244	14.39	92	10	496	259	15.27
100	86	5	499	242	13.86	90	10	498	223	13.91

Tabela 10: Média de sucesso para indivíduos de tamanho 500 bits parte II

Tamanho 500 bits										
	Distância entre os Objetivos									
Obj	D=20	DMi	DMa	Med	DP	D=30	DMi	DMa	Med	DP
10	100	20	487	223	38.69	100	30	480	301	44.61
20	100	20	448	227	30.65	100	30	497	301	32.25
30	100	20	496	249	26.36	100	30	497	273	25.92
40	100	20	493	269	23.17	100	30	498	267	22.91
50	100	20	469	244	18.86	100	30	474	277	18.31
60	100	20	498	255	19.11	100	30	499	254	19.02
70	100	20	497	240	16.05	100	30	451	247	15.02
80	99	20	498	236	14.71	100	30	494	286	14.31
90	97	20	493	266	14.65	99	30	499	270	15.51
100	96	20	494	250	13.10	98	30	498	153	13.99

Tabela 11: Média de sucesso para indivíduos de tamanho 500 bits parte III

Tamanho 500 bits										
	Distância entre os Objetivos									
Obj	D=40	DMi	DMa	Med	DP	D=50	DMi	DMa	Med	DP
10	100	40	497	292	55.68	100	50	457	236	46.76
20	100	40	472	281	26.91	100	50	459	237	30.61
30	100	40	469	259	24.43	100	50	456	263	21.69
40	100	40	493	247	19.87	100	50	498	257	18.47
50	100	40	496	274	18.95	100	50	494	251	19.10
60	100	40	491	246	17.95	100	50	497	265	18.05
70	100	40	496	261	16.04	100	50	487	262	14.33
80	100	40	498	269	15.98	100	50	496	256	15.22
90	100	40	493	254	14.97	100	50	478	269	13.56
100	99	40	499	268	13.83	99	50	492	292	13.09

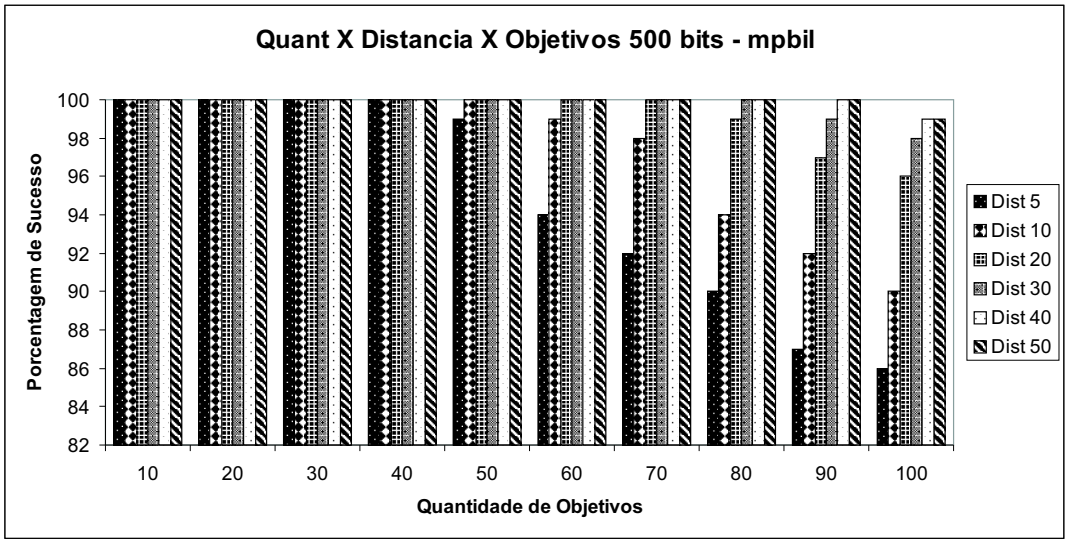

Figura 8: Gráfico com a média de sucesso do Alg. MPBil para 500 bits

Tabela 12: Média de sucesso para indivíduos de tamanho 1000 bits parte I

Tamanho 1000 bits										
	Distância entre os Objetivos									
Obj	D=5	DMi	DMa	Med	DP	D=10	DMi	DMa	Med	DP
10	100	5	901	387	85.36	100	10	861	375	95.76
20	100	5	974	530	65.65	100	10	980	396	73.80
30	100	5	929	539	54.84	100	10	995	445	55.31
40	98	5	922	389	41.75	100	10	994	552	43.02
50	96	5	986	522	45.16	98	10	980	506	38.75
60	93	5	985	495	36.53	94	10	998	464	37.34
70	89	5	996	528	36.55	92	10	987	510	35.99
80	87	5	990	512	32.82	90	10	996	522	32.77
90	85	5	998	517	30.71	88	10	977	489	28.80
100	83	5	994	514	29.39	86	10	997	529	29.85

Tabela 13: Média de sucesso para indivíduos de tamanho 1000 bits parte II

Tamanho 1000 bits										
	Distância entre os Objetivos									
Obj	D=20	DMi	DMa	Med	DP	D=30	DMi	DMa	Med	DP
10	100	20	950	412	103.02	100	30	696	319	68.92
20	100	20	968	493	69.81	100	30	918	489	63.43
30	100	20	951	453	56.75	100	30	999	511	56.42
40	100	20	971	539	47.37	100	30	887	509	34.19
50	100	20	997	546	37.73	100	30	985	494	39.13
60	100	20	996	440	34.66	100	30	974	496	32.66
70	100	20	991	539	32.62	100	30	961	454	31.76
80	99	20	981	458	32.82	99	30	976	510	29.91
90	96	20	996	480	29.32	96	30	989	472	28.85
100	96	20	999	502	29.10	94	30	985	537	27.20

Tabela 14: Média de sucesso para indivíduos de tamanho 1000 bits parte III

Tamanho 1000 bits										
	Distância entre os Objetivos									
Obj	D=40	DMi	DMa	Med	DP	D=50	DMi	DMa	Med	DP
10	100	40	959	452	81.24	100	50	780	356	73.77
20	100	40	966	494	61.43	100	50	995	544	68.19
30	100	40	977	440	44.54	100	50	980	473	53.66
40	100	40	978	534	43.37	100	50	996	399	42.76
50	100	40	988	492	35.53	100	50	988	569	36.07
60	100	40	974	484	36.25	100	50	996	485	32.64
70	100	40	993	575	29.95	100	50	985	473	32.00
80	100	40	983	506	32.84	100	50	984	522	31.50
90	99	40	977	551	27.73	100	50	994	542	29.19
100	97	40	998	522	27.80	97	50	995	524	24.92

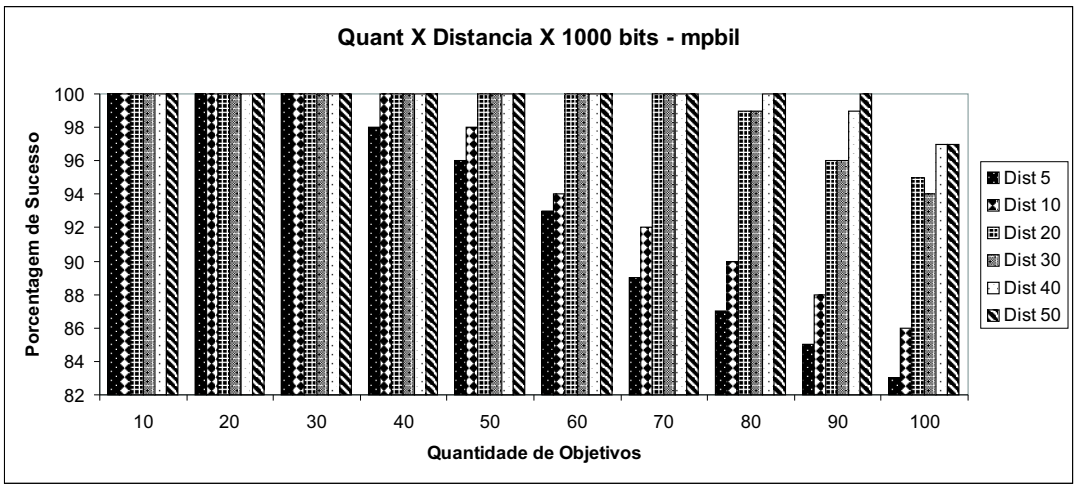

Figura 9: Gráfico com a média de sucesso do Alg. MPBil para 1000 bits

O tempo médio de execução dos modelos pode ser visto na Tabela 15.

Pode-se perceber uma relação de aumento linear gradual do tempo de execução do algoritmo à medida que se aumenta o número de objetivos procurados, como também o tamanho dos indivíduos. Para a configuração usada, com população de 500 inidivíduos, 200 gerações de execução, o tempo médio de execução de um modelo para um conjunto

de 10 objetivos, e indivíduos de tamanho 100 bits é de 6 segundos.

Já o tempo médio de execução de um modelo para um conjunto de 50 objetivos, e indivíduos com tamanho 500 bits é de 2 minutos e 26 segundos. E com um conjunto de 100 objetivos e indivíduos com tamanho 1000 bits, o tempo médio de execução de um modelo é de 6 minutos e 47 segundos.

De acordo com a tabela 15 é possível perceber que o fator responsável pelo expressivo aumento de tempo gasto na execução de cada modelo é o tamanho dos indivíduos trabalhados no problema, e em menor escala esta a quantidade de objetivos procurados.

A tabela 15 apresenta uma relação do tempo gasto por cada modelo durante a execução do algoritmo Multi-PBil.

Tabela 15: Tempo de execução gasto por cada modelo

	Tamanho dos Indivíduos		
Objetivos	**100 bits**	**500 bits**	**1000 bits**
10	6 segundos	2 minutos e 19 segundos	6 minutos e 28 segundos
20	6 segundos	2 minutos e 20 segundos	6 minutos e 29 segundos
30	7 segundos	2 minutos e 21 segundos	6 minutos e 32 segundos
40	7 segundos	2 minutos e 25 segundos	6 minutos e 34 segundos
50	8 segundos	2 minutos e 26 segundos	6 minutos e 37 segundos
60	8 segundos	2 minutos e 28 segundos	6 minutos e 39 segundos
70	8 segundos	2 minutos e 31 segundos	6 minutos e 41 segundos
80	10 segundos	2 minutos e 36 segundos	6 minutos e 44 segundos
90	10 segundos	2 minutos e 39 segundos	6 minutos e 46 segundos
100	11 segundos	2 minutos e 42 segundos	6 minutos e 47 segundos

As Tabelas 16, 17 e 18 mostram respectivamente, para indivíduos de tamanho 100 bits, 500 bits e 1000 bits, algumas médias de modelos de probabilidade eliminados durante as gerações do algoritmo.

Pode-se perceber pelos valores fornecidos pelas tabelas, que no decorrer das gerações a tendência é de que se diminua a porcentagem de modelos eliminados pelo algoritmo durante as gerações. E que à medida que se aumenta o tamanho dos indivíduos, as diferenças entre os valores percentuais de modelos eliminados tendem a ficar mais próximos.

Outro ponto de destaque é que para poucos objetivos, há uma eliminação grande de modelos no início das gerações e menos no final. E com muitos objetivos, a eliminação dos modelos tende a ficar constante no decorrer das gerações, porém, mesmo assim verifica-se a permanência da ordem de se eliminar mais modelos no início das gerações e menos no final.

Quanto à distância, nas execuções onde as distâncias entre os modelos são pequenas, há um número maior de eliminação no início das gerações, e uma eliminação menor nas gerações finais. Com modelos que apresentam grandes distâncias entre si, as diferenças de eliminação entre as gerações tendem a permanecer quase constante, porém com valores mais baixos.

E quando as distâncias e objetivos se encontram próximos da média, trabalhando com indivíduos de tamanho 100 bits, os valores percentuais de modelos eliminados tendem a ficar próximos. E à medida que se aumenta o tamanho dos indivíduos, essa relação de proximidade permanece, porém com valores menores.

Tabela 16: Média de modelos eliminados durante as gerações com indivíduos de tamanho 100 bits

Indivíduos de tamanho 100 bits				
	Gerações			
Distância - Objetivos	**1 - 50**	**51 - 100**	**101 - 150**	**151 - 200**
5 - 10	37%	21%	12%	7%
5 - 50	31%	19%	11%	6%
5 - 100	12%	14%	9%	11%
30 - 10	35%	20%	12%	10%
30 - 50	28%	19%	12%	11%
30 - 100	16%	14%	13%	12%
50 - 10	29%	22%	17%	11%
50 - 50	28%	26%	13%	22%
50 - 100	17%	11%	11%	9%

Tabela 17: Média de modelos eliminados durante as gerações com indivíduos de tamanho 500 bits

Indivíduos de tamanho 500 bits				
	Gerações			
Distância - Objetivos	**1 - 50**	**51 - 100**	**101 - 150**	**151 - 200**
5 - 10	35%	20%	12%	9%
5 - 50	28%	19%	8%	6%
5 - 100	12%	12%	8%	8%
30 - 10	32%	20%	12%	11%
30 - 50	28%	17%	12%	9%
30 - 100	17%	14%	14%	11%
50 - 10	21%	18%	13%	11%
50 - 50	20%	20%	19%	18%
50 - 100	15%	10%	9%	7%

Tabela 18: Média de modelos eliminados durante as gerações com indivíduos de tamanho 1000 bits

Indivíduos de tamanho 1000 bits				
	Gerações			
Distância - Objetivos	**1 - 50**	**51 - 100**	**101 - 150**	**151 - 200**
5 - 10	29%	17%	8%	7%
5 - 50	27%	17%	8%	6%
5 - 100	12%	11%	8%	7%
30 - 10	24%	20%	12%	11%
30 - 50	17%	13%	13%	9%
30 - 100	11%	12%	14%	8%
50 - 10	20%	18%	11%	6%
50 - 50	18%	16%	13%	10%
50 - 100	9%	9%	8%	6%

As Tabelas 19, 20 e 21 mostram os resultados obtidos durante os testes feitos pela ferramenta *EO Evolutionary Computation Framework*.

Os parâmetros de *sharing* estudados compõem o intervalo fechado [0, 1]. Os resultados apresentados nas tabelas 19, 20 e 21 mostram somente o valor de *sharing* que apresentou os melhores resultados durante os testes realizados com a ferramenta.

Para indivíduos de tamanho 100 bits, a ferramenta EO apresentou um desempenho satisfatório na resolução do problema. Os testes mostraram que para os conjuntos de 30, 40 e 50 objetivos, os resultados estão próximos dos apresentados pelo método Multi-PBil, tendo pequenas diferenças entre seus valores.

Entretanto, o teste com os objetivos configurados com distâncias 5 e 10 apresentaram resultados inferiores aos conseguidos pelo método Multi-PBil. Embora a comparação apresente diferenças, estas não são muito distantes, como por exemplo, para um conjunto de 100 objetivos com distância mínima 5, o método Multi-PBil apresentou um valor de 89% de sucesso, enquanto o algoritmo genético utilizando *sharing* apresentou 82% de sucesso.

Comparando-se a relação de sucesso entre a quantidade de objetivos pelas distâncias apresentadas entre eles, para indivíduos de tamanho 500 bits, percebe-se que os resultados obtidos são equivalentes aos apresentados pelos indivíduos de tamanho 100 bits, quando ambos são testados pela ferramenta EO. Os resultados mostram que para indivíduos de tamanho 500 bits, o desempenho foi inferior ao apresentado pelos indivíduos de tamanho 100 bits.

Já os resultados obtidos com os indivíduos de tamanho 500 bits pela ferramenta EO, quando comparados com os resultados apresentados pelo método Multi-PBil, percebe-se que o método Multi-PBil apresenta um desempenho melhor.

A maior diferença aparece na configuração máxima dos métodos, isto é, para um conjunto de 100 objetivos configurados a uma distância mínima 5, tem-se os resultados de 86% de sucesso para o método Multi-PBil e 82% de sucesso para o algoritmo genético usando *"sharing"*.

Para indivíduos de tamanho 1000 bits, os resultados apresentados pela ferramenta EO demonstram valores inferiores aos apresentados pelas configurações de indivíduos cujos tamanhos são 100 bits e 500 bits. Esse desempenho inferior se deve ao aumento do espaço de busca ao qual o teste é submetido. Quando comparado com os resultados obtidos pelo método Multi-PBil, pode-se perceber um desempenho bem inferior ao apresentado pelo método proposto neste trabalho.

Tabela 19: Média de sucesso com indivíduos de tamanho 100 bits utilizando a ferramenta EO

Tamanho 100 bits							
	Distância entre os Objetivos						
Objetivos	**5**	**10**	**20**	**30**	**40**	**50**	**Parâmetro de Sharing**
10	100	100	100	100	100	100	0.73
20	100	100	100	100	100	100	0.69
30	98	99	100	100	100	100	0.62
40	94	98	100	100	100	100	0.57
50	93	96	100	100	100	100	0.51
60	90	93	100	100	100	100	0.48
70	89	91	100	100	100	100	0.43
80	88	89	99	99	99	100	0.36
90	83	87	96	98	95	99	0.27
100	82	85	94	96	91	97	0.22

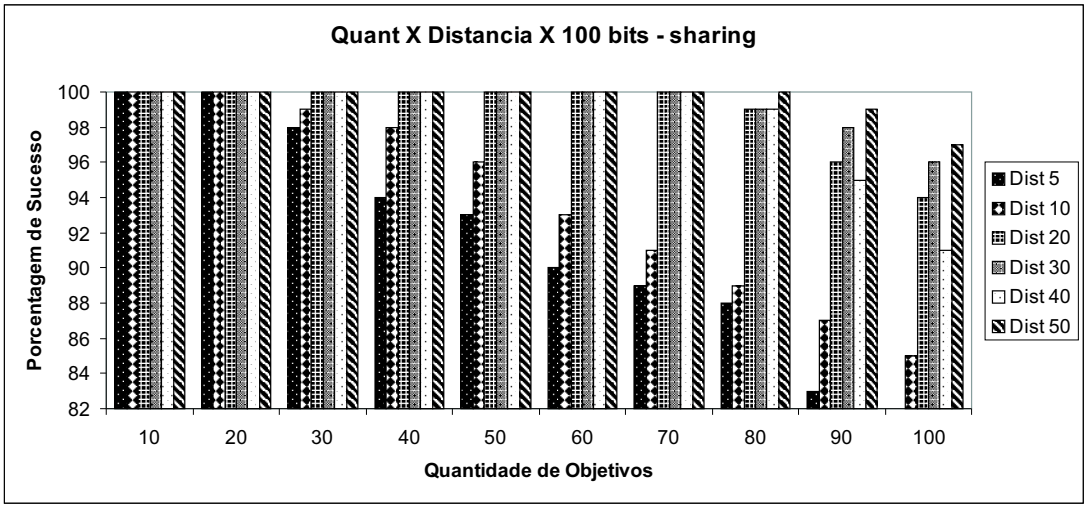

Figura 10: Gráfico com a média de sucesso da ferramenta EO para 100 bits usando *sharing*

Tabela 20: Média de sucesso com indivíduos de tamanho 500 bits utilizando a ferramenta EO

Tamanho 500 bits							
	Distância entre os Objetivos						
Objetivos	**5**	**10**	**20**	**30**	**40**	**50**	**Parâmetro de Sharing**
10	100	100	100	100	100	100	0.73
20	99	100	100	100	100	100	0.66
30	98	99	100	100	100	100	0.59
40	96	97	100	100	100	100	0.50
50	94	95	100	100	100	100	0.46
60	91	93	100	100	100	100	0.41
70	90	92	99	100	100	100	0.35
80	89	90	97	99	98	99	0.29
90	87	89	95	98	97	98	0.24
100	82	87	93	97	96	95	0.20

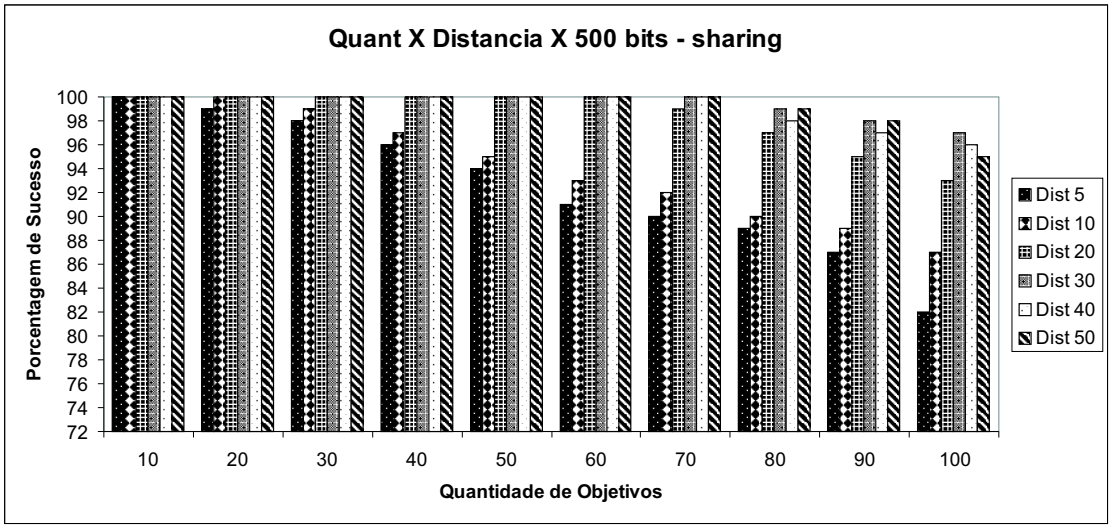

Figura 11: Gráfico com a média de sucesso da ferramenta EO para 500 bits usando *sharing*

Tabela 21: Média de sucesso com indivíduos de tamanho 1000 bits utilizando a ferramenta EO

Tamanho 1000 bits							
	Distância entre os Objetivos						
Objetivos	**5**	**10**	**20**	**30**	**40**	**50**	**Parâmetro de Sharing**
10	97	100	100	100	100	100	0.71
20	95	98	100	100	100	100	0.65
30	92	97	100	100	100	100	0.57
40	90	96	100	100	100	100	0.48
50	87	96	100	100	100	100	0.41
60	85	93	99	100	100	100	0.35
70	84	92	98	99	98	99	0.33
80	81	91	96	96	97	98	0.29
90	78	90	95	94	96	97	0.21
100	77	88	93	93	95	96	0.19

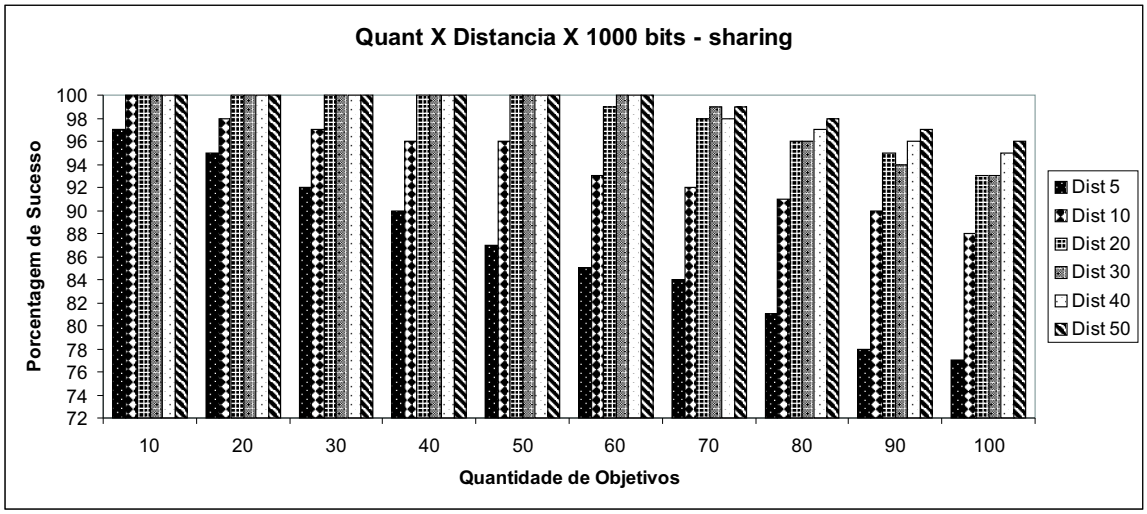

Figura 12: Gráfico com a média de sucesso da ferramenta EO para 1000 bits usando *sharing*

Para a aplicação do **Teste de Hipótese** utilizou-se as seguintes hipóteses:

- H_o Os resultados apresentados nas tabelas de sucesso representam a realidade do método quando aplicado;

- H_1 Os resultados nas tabelas de sucesso da aplicação do método Multi-PBil representam valores menores que os demonstrados nas tabelas de sucesso.

O valor de α foi estipulado em 5%. O valor das amostras utilizado para calcular o valor do teste utilizou um número referente a cada grupo de objetivos, no valor de 70%, isto é, de cada grupo de objetivos escolheu-se aleatoriamente 70% dos objetivos para compor a média de cada amostra.

O valor do limiar à esquerda $(-\boldsymbol{Z})$ foi extraído de uma tabela que representa a área de uma distribuição normal padrão [Hoe74].

As tabelas 22, 23, 24, 25, 26 e 27 mostram os resultados obtidos com o aplicação do Teste de Hipótese (Significância).

As variáveis das tabelas indicam **Obj** como número de objetivos trabalhados, a variável $\overline{x}$ identifica a média do conjunto de amostra escolhido, a variável **Med** como a média obtida do conjunto total de objetivos, a variável **DP** como o desvio-padrão do conjunto de objetivos trabalhados, a variável $\boldsymbol{Z_c}$ como o valor de teste calculado, a variável $(-\boldsymbol{Z})$ representa o limiar à esquerda entre a área de aceitação e a área de rejeição da hipótese e, a variável **Conclusão** emite um paracer sobre a hipótese referente ao conjunto de objetivos trabalhados.

Por meio das tabelas 22, 23, 24, 25, 26 e 27 é possível perceber em quase sua totalidade, que os valores obtidos nos testes exaustivos com o método Multi-PBil demonstram a veracidade dos resultados, e mostram que tais resultados estão dentro de uma região de aceitação da hipótese proposta.

Para objetivos com tamanho 100 bits, os resultados das tabelas 22, 23 mostram que para objetivos com distância 5 bits, somente o conjunto com 40 objetivos apresentou resultados fora da área de aceitação da hipótese, sendo que o valor obtido para $\boldsymbol{Z_c}$ situa-se muito próximo do limiar da área de aceitação $-\boldsymbol{Z}$. Para objetivos com distância 10 bits, os grupos com 40 e 80 objetivos apresentaram valores de $\boldsymbol{Z_c}$ fora da área de aceitação da hipótese.

Para objetivos com tamanho 500 bits, somente a tabela 25 apresentou um resultados para um conjunto de 40 objetivos fora da área de aceitação da hipótese, sendo que o valor de $\boldsymbol{Z_c}$ se encontra próximo do valor do limiar da área de aceitação. A tabela 24 apresentou todos os resultados situados dentro da área de aceitação da hipótese.

Para objetivos com tamanho 1000 bits, somente a tabela 27 apresentou valores fora da área de aceitação da hipótese para um conjunto de 80 objetivos, sendo este valor também próximo do valor do limiar da área de aceitação da hipótese. A tabela 26 apresentou todos os valores dentro da área de aceitação da hipótese.

Tabela 22: Resultados dos Testes de Hipóteses para indivíduos de tamanho 100 bits e distância entre objetivos no valor 5

Tamanho 100 bits e distância no valor 5						
Obj	**Med**	$\overline{x}$	**DP**	$-Z$	Z_c	**Conclusão**
10	66	64	8.63	-1.65	-0.73	Aceita-se H_0
20	52	51	5.19	-1.65	-0.86	Aceita-se H_0
30	48	47	6.14	-1.65	-0.89	Aceita-se H_0
40	48	47	3.40	-1.65	-1.88	Rejeita-se H_0, Aceita-se H_1
50	57	58	3.75	-1.65	+1.88	Aceita-se H_0
60	50	50	3.65	-1.65	0.00	Aceita-se H_0
70	54	55	3.20	-1.65	+2.63	Aceita-se H_0
80	51	52	2.96	-1.65	+3.03	Aceita-se H_0
90	51	52	2.79	-1.65	+3.44	Aceita-se H_0
100	54	55	2.86	-1.65	+3.57	Aceita-se H_0

Tabela 23: Resultados dos Testes de Hipóteses para indivíduos de tamanho 100 bits e distância entre objetivos no valor 10

Tamanho 100 bits e distância no valor 10						
Obj	**Med**	$\overline{x}$	**DP**	$-Z$	Z_c	**Conclusão**
10	38	36	5.54	-1.65	-1.14	Aceita-se H_0
20	47	45	5.61	-1.65	-1.60	Aceita-se H_0
30	52	51	4.57	-1.65	-1.20	Aceita-se H_0
40	59	57	4.14	-1.65	-3.07	Rejeita-se H_0, Aceita-se H_1
50	50	51	3.54	-1.65	+2.00	Aceita-se H_0
60	53	55	3.08	-1.65	+5.12	Aceita-se H_0
70	51	52	3.28	-1.65	+2.56	Aceita-se H_0
80	59	57	2.88	-1.65	-6.25	Rejeita-se H_0, Aceita-se H_1
90	51	51	2.62	-1.65	0.00	Aceita-se H_0
100	51	52	2.65	-1.65	+3.84	Aceita-se H_0

Tabela 24: Resultados dos Testes de Hipóteses para indivíduos de tamanho 500 bits e distância entre objetivos no valor 5

Tamanho 500 bits e distância no valor 5						
Obj	**Med**	$\overline{x}$	**DP**	$-Z$	Z_c	**Conclusão**
10	203	191	40.67	-1.65	-0.93	Aceita-se H_0
20	222	211	32.93	-1.65	-1.49	Aceita-se H_0
30	285	293	26.09	-1.65	+1.68	Aceita-se H_0
40	254	259	20.93	-1.65	+1.51	Aceita-se H_0
50	219	215	17.50	-1.65	-1.61	Aceita-se H_0
60	213	217	17.54	-1.65	+1.76	Aceita-se H_0
70	242	239	15.94	-1.65	-1.57	Aceita-se H_0
80	249	253	15.80	-1.65	+2.27	Aceita-se H_0
90	244	247	14.39	-1.65	+1.98	Aceita-se H_0
100	242	241	13.86	-1.65	-0.72	Aceita-se H_0

Tabela 25: Resultados dos Testes de Hipóteses para indivíduos de tamanho 500 bits e distância entre objetivos no valor 10

Tamanho 500 bits e distância no valor 10						
Obj	**Med**	$\overline{x}$	**DP**	$-Z$	Z_c	**Conclusão**
10	198	176	44.72	-1.65	-1.55	Aceita-se H_0
20	258	252	36.94	-1.65	-0.72	Aceita-se H_0
30	236	239	23.59	-1.65	+0.69	Aceita-se H_0
40	240	233	25.15	-1.65	-1.76	Rejeita-se H_0, Aceita-se H_1
50	272	281	20.76	-1.65	+3.07	Aceita-se H_0
60	244	242	19.75	-1.65	-0.78	Aceita-se H_0
70	231	229	16.47	-1.65	-1.01	Aceita-se H_0
80	231	234	16.07	-1.65	+1.67	Aceita-se H_0
90	259	265	15.27	-1.65	+3.72	Aceita-se H_0
100	223	227	13.91	-1.65	+2.87	Aceita-se H_0

Tabela 26: Resultados dos Testes de Hipóteses para indivíduos de tamanho 1000 bits e distância entre objetivos no valor 5

Tamanho 1000 bits e distância no valor 5						
Obj	**Med**	$\overline{x}$	**DP**	$-Z$	Z_c	**Conclusão**
10	387	364	85.36	-1.65	-0.85	Aceita-se H_0
20	530	517	65.65	-1.65	-0.88	Aceita-se H_0
30	539	548	54.84	-1.65	+0.89	Aceita-se H_0
40	389	401	41.75	-1.65	+1.81	Aceita-se H_0
50	522	531	45.16	-1.65	+1.41	Aceita-se H_0
60	495	491	36.53	-1.65	-0.84	Aceita-se H_0
70	528	534	36.55	-1.65	+1.37	Aceita-se H_0
80	512	507	32.82	-1.65	-1.36	Aceita-se H_0
90	517	513	30.71	-1.65	-1.23	Aceita-se H_0
100	514	521	29.39	-1.65	+2.38	Aceita-se H_0

Tabela 27: Resultados dos Testes de Hipóteses para indivíduos de tamanho 1000 bits e distância entre objetivos no valor 10

Tamanho 1000 bits e distância no valor 10						
Obj	**Med**	$\overline{x}$	**DP**	$-Z$	Z_c	**Conclusão**
10	375	358	95.76	-1.65	-0.56	Aceita-se H_0
20	396	403	73.80	-1.65	+0.42	Aceita-se H_0
30	445	433	55.31	-1.65	-1.18	Aceita-se H_0
40	552	546	43.02	-1.65	-0.88	Aceita-se H_0
50	506	501	38.75	-1.65	-0.91	Aceita-se H_0
60	464	452	37.34	-1.65	-1.45	Aceita-se H_0
70	510	518	35.99	-1.65	+1.86	Aceita-se H_0
80	522	515	32.77	-1.65	-1.91	Rejeita-se H_0, Aceita-se H_1
90	489	493	28.80	-1.65	+1.32	Aceita-se H_0
100	529	526	29.85	-1.65	-1.01	Aceita-se H_0

As Figuras 13, 14 e 15 mostram os gráficos comparativos entre os resultados obtidos pelo método proposto Multi-PBil e, os resultados obtidos pela ferramenta *EO Evolutionary Computation Framework* utilizando um Algoritmo Genético com *"sharing"*.

As Figuras 13, 14 e 15 mostram, respectivamente, as comparações com indivíduos de tamanho 100 bits, 500 bits e 1000 bits. Nestas comparações destaca-se o melhor comportamento obtido por ambos os métodos.

Para indivíduos de tamanho 100 bits pode-se identificar um desempenho melhor e relevante por parte do algoritmo Multi-PBil para as distâncias 5 e 10. As demais distâncias apresentam comportamentos próximos quando testadas pelos dois métodos.

Para indivíduos de tamanho 500 bits, as diferenças são próximas, porém, o método Multi-PBil consegue desempenhar uma busca mais satisfatória que a ferramenta EO.

E por fim, para indivíduos de tamanho 1000 bits, as diferenças maiores aparecem quando se trabalha com distâncias de valor 5, onde o método Multi-PBil consegue desempenhar uma busca mais satisfatória que a ferramenta EO. Nas demais distâncias estudadas, os resultados comparativos estão próximos, tendo se destacado uma pequena margem de sucesso por parte do método Multi-PBil.

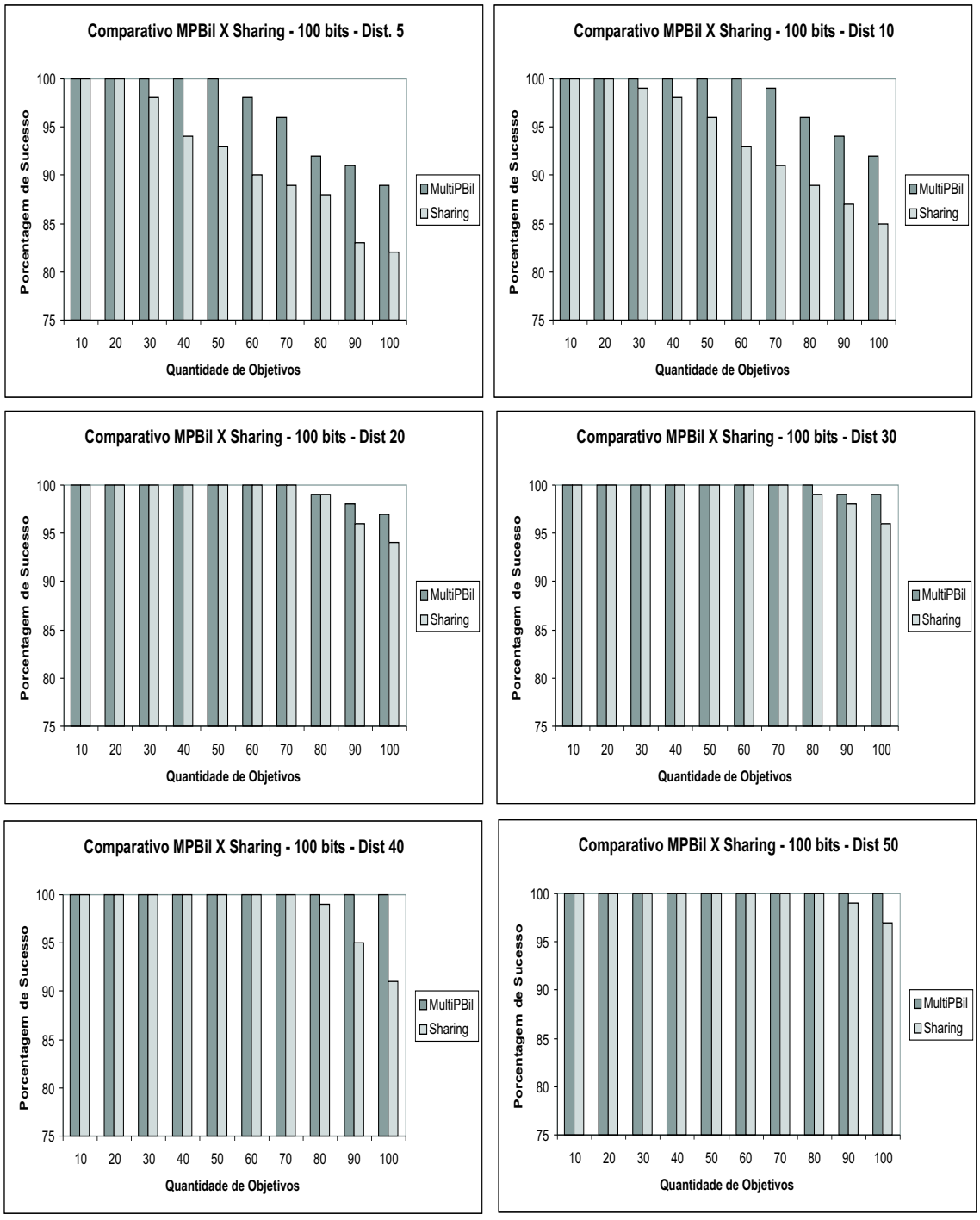

Figura 13: Gráfico comparativo entre o método MPBil e o Algoritmo Genético usando sharing para indivíduos de tamanho 100 bits.

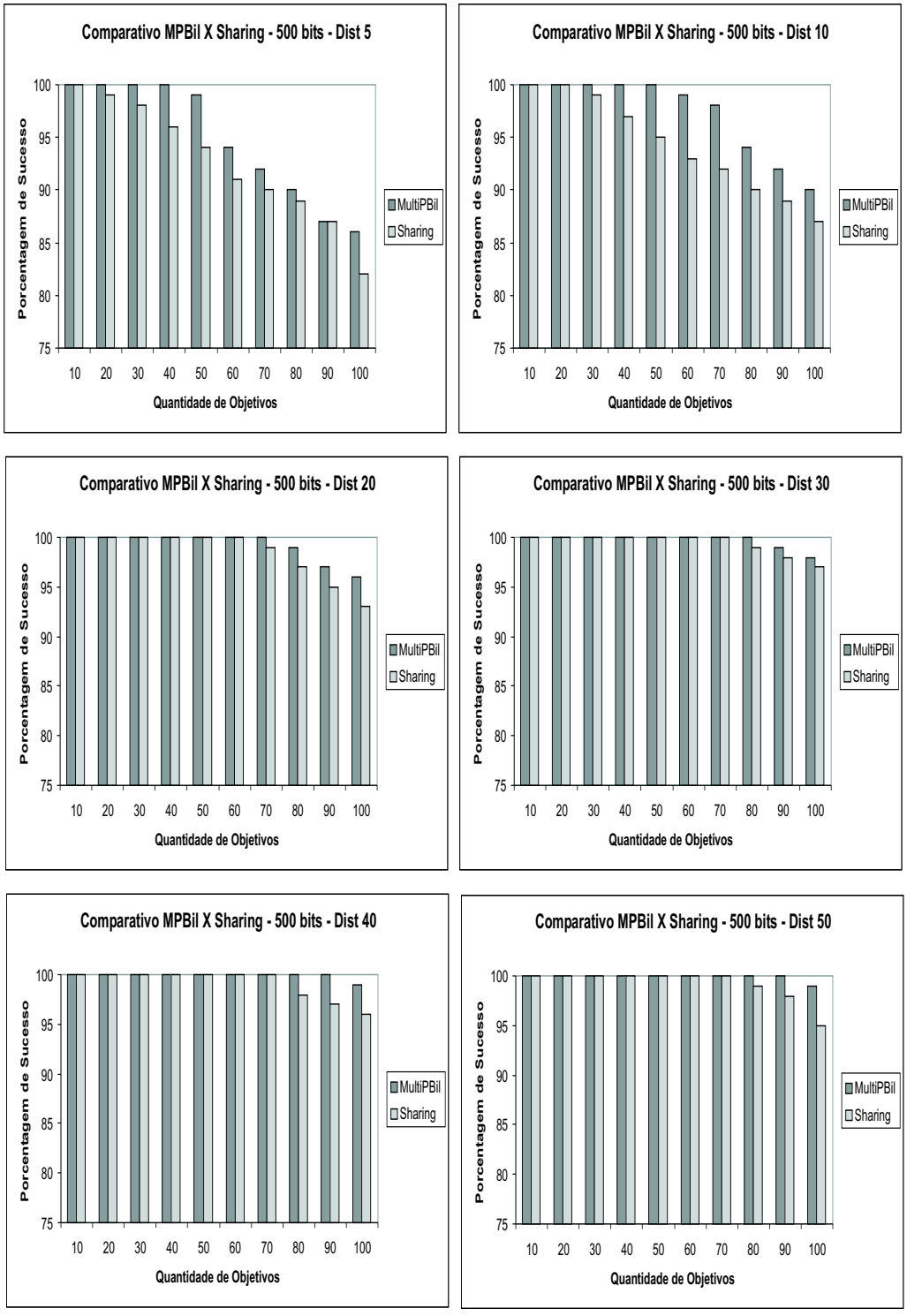

Figura 14: Gráfico comparativo entre o método MPBil e o Algoritmo Genético usando sharing para indivíduos de tamanho 500 bits.

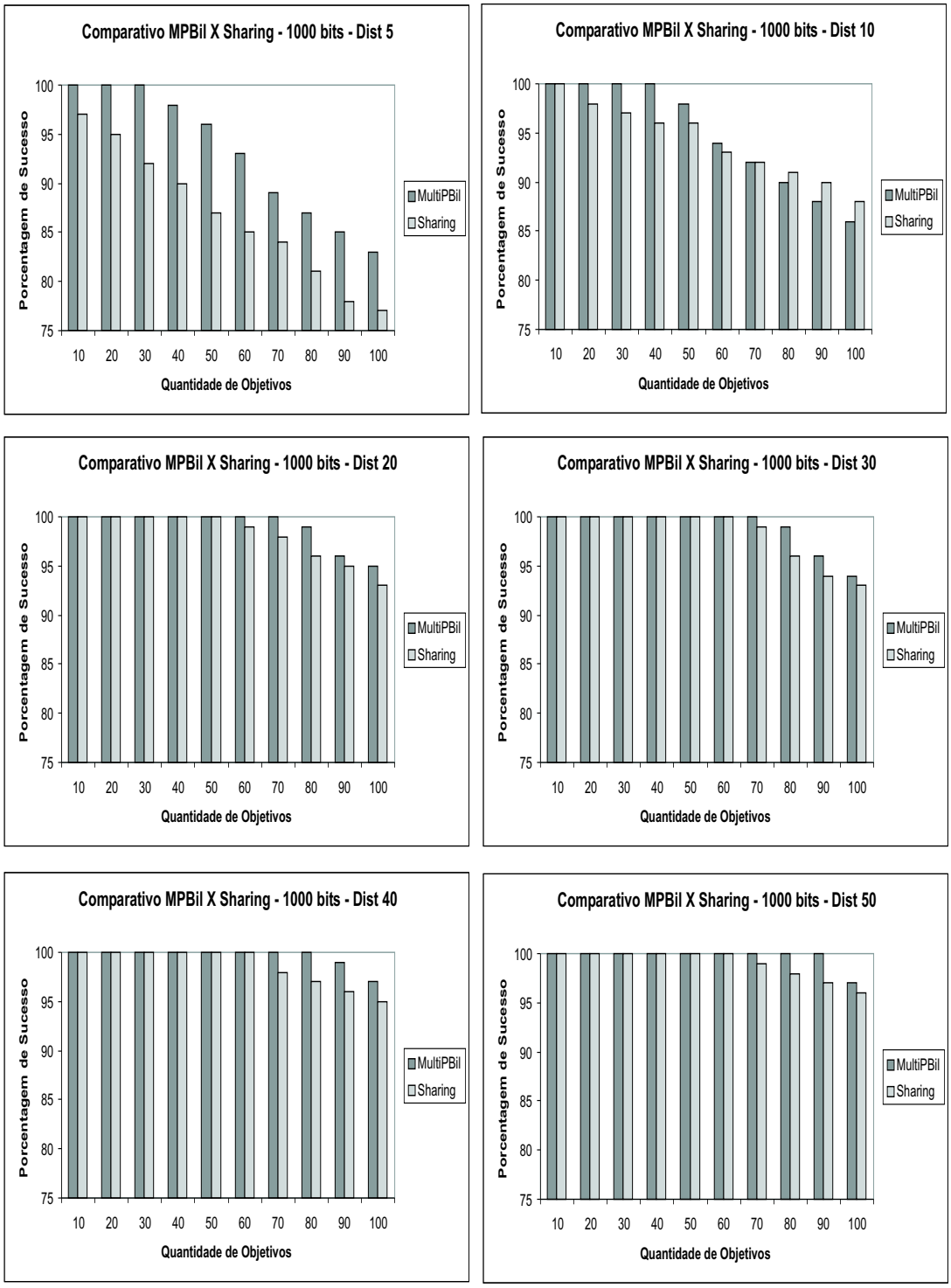

Figura 15: Gráfico comparativo entre o método MPBil e o Algoritmo Genético usando sharing para indivíduos de tamanho 1000 bits.

6 Conclusão e Trabalhos Futuros

Os Algoritmos de Estimação de Distribuição (AED's) são algoritmos evolucionários que usam modelos de probabilidade para representar propriedades estatísticas das populações. Possui como característica o uso de modelos de probabilidade com o objetivo de guiar a população de indivíduos na busca por soluções de problemas.

Este trabalho apresentou um método chamado Multi-PBil, cujo objetivo é o de fornecer um conjunto de modelos de probabilidade capazes de encontrar soluções em problemas multimodais. Diferentemente do algoritmo PBil, onde este possui somente um modelo de probabilidade, sendo este modelo de probabilidade inicializado com o valor 0.5 em todas as posições. O método Multi-PBil tem a capacidade de criar quantos modelos forem necessários, e não obrigatoriamente esses modelos começam com o valor 0.5.

O método Multi-PBil apresenta como vantagem o fato de inicializar os modelos de probabilidade com valores próximos de 0 (zero) ou 1 (um), o que permite uma migração mais rápida da população para o ponto ótimo global. Apresenta também a criação de vários modelos de probabilidade e um sistema métrico que permite identificar modelos com desempenho insatisfatórios, os quais podem ser eliminados da busca em tempo real.

O método permite também a busca simultânea da solução de mais de um objetivo. Essa possibilidade de se trabalhar com mais de um modelo na busca de vários objetivos permite uma paralelização dos modelos de probabilidade. Vários modelos buscando vários objetivos possibilitam identificar situações onde se têm um mesmo modelo encontrando mais de um objetivo, ou um mesmo objetivo sendo encontrado por mais de um modelo, como ainda o conjunto de modelos como um todo alcançar a totalidade de objetivos propostos na solução dos problemas trabalhados.

O método, porém, apresentou algumas limitações quanto ao seu desempenho. A primeira delas diz respeito ao tempo de execução gasto pelos modelos de probabilidade. À medida que se aumenta o tamanho dos indivíduos, gasta-se mais tempo na busca dos objetivos.

Outra limitação do método está relacionada com a quantidade de modelos de probabilidade necessária e suficiente para resolver os problemas propostos. A identificação da quantidade suficiente de modelos para resolver os problemas foi tratada de forma empírica. Na maior parte das execuções, a quantidade de modelos criados foi muito superior à quantidade de objetivos procurados.

E uma terceira limitação relacionada ao método Multi-PBil diz respeito à técnica adotada para eliminação de modelos em excesso. A técnica usada neste trabalho faz uso de Distâncias Euclidianas para resolver este problema. Embora tal técnica tenha conseguido desempenhar o papel a que se propôs, a média de modelos eliminados durante as primeiras gerações do algoritmo variou muito à medida que se aumentou o tamanho dos indivíduos. Sugere-se a elaboração de uma heurística que permita descobrir uma relação mais precisa do valor ideal do número de modelos a serem eliminados durante as gerações do algoritmo.

No entanto, o algoritmo Multi-PBil se mostrou capaz de resolver os problemas elaborados neste trabalho. Nos problemas propostos, o método apresentou um bom desempenho para um conjunto de indivíduos com tamanhos pequenos, e em relação às distâncias configuradas pelos objetivos, os conjuntos cujas as distâncias eram superiores a 10 apresentaram melhores resultados que as distâncias menores ou iguais a 10.

Quando comparado o desempenho obtido pelo método Multi-PBil com o obtido pela ferramenta de teste *EO Evolutionary Computation Framework*, que utilizou um algoritmo genético com *sharing*, ambas utilizando o mesmo conjunto de dados de entrada, o método Multi-PBil apresentou resultados melhores que a ferramenta de teste. Os resultados mais expressivos foram observados para conjuntos de objetivos com distâncias pequenas, no caso 5 e 10.

O Teste de Hipótese (significância) permitiu comprovar que os valores obtidos nos testes exaustivos com a implementação do Método Multi-PBil, para diferentes instâncias, encontram-se dentro da área de aceitação da hipótese trabalhada.

Esses dados demonstram que o método Multi-PBil possui uma eficiência e desempenho melhor que o Algoritmo Genético com *sharing* no processo de busca por objetivos situados muito próximos em um espaço de busca multimodal. Isto é, o método Multi-PBil é capaz de identificar e diferenciar objetivos próximos com pequenas variações binárias, como os que foram tratados no capítulo Experimentos.

Os bons resultados apresentados servem de estímulo para a aplicação do método

em novos domínios de problemas. Novos avanços devem ser alcançados em trabalhos futuros. Dentre as melhorias possíveis de serem realizadas, estão o estudo de uma nova métrica que permita analisar com maior precisão as distâncias ocupadas pelos modelos de probabilidade e como consequente impacto a eliminação dos modelos com desempenho insatisfatório.

Um estudo mais aprofundado sobre a quantidade de modelos necessários para uma busca de um conjunto de objetivos. Objetiva-se ter uma quantidade de modelos de probabilidade que esteja próxima da quantidade ideal, evitando dessa maneira um gasto excessivo de tempo pelo espaço de busca. E um terceiro estudo pode ser feito analisando-se o impacto do método Multi-PBil em uma arquitetura paralela, a qual pode desempenhar a busca em um tempo menor ao tempo destacado neste trabalho.

Referências

[Bal94] S Baluja. Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Technical report, Carnegie Mellon University, 1994.

[Ben02] E. Bengoetxea. *Inexact Graph Matching Using Estimation of Distribuition Algorithms*. PhD thesis, Ecole Nationale Superieure des Telecommunications, Paris, France, Dec 2002.

[Bet81] A. D. Bethke. Genetic algorithms as function optimizers. Ph. D. thesis, Departament of Computer and Comunication Sciences, University of Michigan, 1981.

[BLM95] M. J. Shaw B. L. Miller. Genetic algorithms with dynamic niche sharing for multimodal function optimization. Urbana: University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, December 1995. Illigal Report No. 95010.

[Boo87] L. Booker. Improving search in genetic algorithms. Morgan Kaufmann Publishers, 1987. Genetic Algorithms and Simulated Annealing (USA).

[Coo91] M. L. Cook. Genetic and ecological diversity: The sport of nature. London: Chapman & Hall, 1991.

[Cos88] Sérgio Francisco Costa. *Introdução Ilustrada à Estatística*. Harbra, 1988. São Paulo.

[Dar59] C. Darwin. On the origin of species by means of natural selection of the preservation of favored races in the struggles for life., 1859. Murray, London, UK.

[DB93] R. R. Martin D. Beasley, D. R. Bull. A sequential niche technique for multimodal function optimization. Evolutionary Computation 1(2), 1993. pp 101-125.

[DEG87] J. Richardson D. E. Goldberg. Genetic algorithms with sharing for multimodal function optimization. Proceedings of the Second Interntional Conference on Genetic Algorithms, Lawrence Erlbaum, 1987.

[FS94] H. Ulmer A. Zell F. Streichert, G. Stein. A clustering based niching ea for multimodal search space. Technical report, University of Tubingen, 1994.

[Gil85] A. M. Gilles. Machine learning procedures for generating image domain feature detectors. Tech Report, Ann Harbor, University of Michigan, 1985.

[Gol89a] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learnig*. Addison Wesley, 1989.

[Gol89b] D. E. Goldberg. Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Reading, Massachussetts, 1989.

[Gre84] J. J. Grefenstette. Genesis: A system for using genetic search procedures. Proceedings of the 1984 Conference on Intelligent Systems and Machines, 1984.

[Hoe74] Paul G. Hoel. *Estatística Elementar*. Fundo de Cultura, 1974. Rio de Janeiro.

[Hol75] J. H. Holland. Adaptation in natural and artificial systems. Ann Arbor, The University of Michigan Press, Michigan, 1975.

[IS97] D. Stern I. Servet, L. Trave Massuyes. Telephone network traffic overloading diagnosis and evolutionary computation techniques. In Artificial Evolution 07, J. K. Hao et al. Eds., 1997. 137-144.

[JDS89] L. J. Eshelman J. D. Schaffer, R. A. Caruana. A study of control parameters affecting online performance of genetic algorithms for function optimization. Proceedings of the 3rd International Conference on Genetic Algorithms, Morgan kaufmann Publishers, 1989.

[jeg05] jeggermo. Evolutionary computation framework. http://www.liacs.nl/ jeggermo/eo/snapshot/, Maio 2005.

[JN99] S. J. Wright J. Nocedal. Numerical optimization. Springer-Verlag, New York, 1999.

[Jon75] K. A. De Jong. *An Analysis of the Behavior of a Class of Genetic Adaptative Systems*. PhD thesis, University of Michigan, 1975.

[Jon80] K. A. De Jong. Adaptive system design: A genetic approach. IEEE Trans. Syst., Man and Cyber. 10, 1980.

[JPL02] G. T. Parks P. J. Clarkson J. P. Li, M. E. Balazs. A species conserving genetic algorithm for multimodal function optimization. Mit Press Cambridge, MA, USA, 2002. pages 207 - 234.

[Koz92] J. R. Koza. Genetic programming: On the programming of the computers by means of natural selection. Mit Press, Cambridge, Massachussetts, 1992.

[Lan89] C. G. Langton. Artificial life. Addison-Wesley, 1989. Redwood City: CA.

[Lar02] O. Larsen. Construção de atributos x-of-n utilizando algoritmos genéticos. Master's thesis, Pontifícia Universidade Católica do Paraná, 2002.

[LJF66] M. J. Walsh L. J. Fogel, A. J. Owens. Artificial intelligence through simulated evolution. John Wiley, New York, 1966.

[MAB99] I. C. Parmee M. A. Beck. Extending the bounds of the search space: A multipopulation approach. GECCO - 99, Proceedings of the Genetic and Evolutionary Computation Conference, Morgan Kaufmann Publishers, 1999.

[Mah92] S. W. Mahfoud. Crowding and preselection revisited. Parallel Problem Solving from Nature, 2, R. Manner and B. Manderick. Elsevier Science Publishers: Amsterdam, 1992. pp 27 - 36.

[Mat00] S. W. Mathfound. Niching methods, in:back, t.; fogel, d.b.; michalewicz(eds), evolutionary computation 2. Institute of Physics, 2000. pp 87-92.

[MB00] P. Siarry M. Bessaou, A. Petrowski. Island model cooperating with speciation for multimodal optimization. PPSN VI, 6th International Conference, Paris - France, Setember 2000. Proceedings. Lecture Notes in Computer Science 1917 Springer 2000, ISBN 3-540-41056-2.

[Mic96] Z. Michalewics. Genetic algorithms + data structures = evolution programs. 3rd ed. Springer-Verlag, Berlim Heidelberg, New York, 1996.

[Mis05] Trevor Misfeldt. Algoritmo mersenne twister. http://www.c-sharpcorner.com/Code/2003/April/MersenneTwisterAlgo.asp, Janeiro 2005.

[MKD01] K. Socha M. K. Dorohinicki. Crowding factor in evolutionary multi-agent system for multiobjective optimization. IC-AI'01 Conference, Las Vegas (USA), June 2001.

[MP99] F. Lobo M. Pelikan, D. E. Goldberg. A survey of optimization by building and using probabilistic models. Technical report, University of Illinois at Urbana Champaign, Illinois Genetic Algorithms Laboratory, 1999.

[MS98] A. Ducoulombier M. Sebag. Extending population-based incremental learning to continuous search spaces. In Parallel Problem Solving from Nature - PPSN V, A.E. Eiben, et al. Eds., 1998. 418 - 427.

[NFM99] N. J. Hopper N. F. McPhee. Analisys of genetic diversity through population history. In: GECCO99: Proceedings of the Genetic and Evolutionary Computation Conference, July 1999.

[Pet] A. Petrowski. A clearing procedure as a niching method for genetic algorithms. url = "citeseer.ist.psu.edu/153950.html".

[Pet97] A. Petrowski. A new selection operator dedicated to speciation. Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA 97) Morgan Kaufmann, San Francisco, CA, 1997.

[PL01] J. A. Lozano P. Larranãga. Estimation of distribution algorithms: A new tool for evolutionary computation. Kluwer Academic Publisher, 2001.

[PLN05] J. A. Lozano P. Laura Naga. Special issue on estimation of distribution algorithms. The MIT Press, 2005. Vol. 13, Issue 1.

[Rao] S. S. Rao.

[Rec73] I. Rechenberg. Evolutionsstrategie - optimierung technischer systeme nach prinzipien der biologischen evolution. Frommann-Holzboog, Stuttgart, 1973.

[RS94] M. J. Shaw R. Sikora. A double-layered learning approach to acquiring rules for classification: Integrating genetic algorithms with similarity-based learning. O RSA Journal on Computing, 1994.

[SB95] R. Caruana S. Baluja. Removing the genetics from the standard genetic algorithm. Proceedings of the Twelfth International Conference on Machine Learning, July 1995. CMU-CS-95-141.

[SGN96] A. Sofer S. G. Nash. Linear e nonlinear programming. McGraw-Hill International Editions, New York, 1996.

[Sha03] J. L. Shapiro. Scaling of probability-based optimization algorithms. Advances in Neural Information Processing Systems 15, 2003. NIPS2002.

[Spi72] Murray R. Spiegel. *Estatística*. McGraw-Hill do Brasil, 1972. São Paulo.

[Sys89] G. Sysverda. Uniform crossover in genetic algorithms. Proceedings of the Third Internationl Conference on Genetic Algorithms, San Mateo, Califórnia, Morgan Kaufmann Publishers, 1989.

[WBPN98] R. E. Keller F. D. Francone W. Banzhaf P. Nordin. *Genetic Programming: An Introduction*. Morgan Kaufmann, 1998.

[YS85] T. Tanino Y. Sawaragi, H. Nakayama. Theory of multiobjective optimization. Academic Press, 1985.

APÊNDICE A – Problemas de Otimização

Este apêndice aborda conceitos relativos aos processos de otimização unimodal e multimodal, mostrando seus conceitos e explicações. Tem-se também uma breve explicação sobre otimização contínua e otimização discreta. Embora tanto as definições de otimização unimodal quanto multimodal trazem consigo um conceito mais amplo, que inclui vetores de restrições, os experimentos tratados neste trabalho se limitam a uma otimização mais simples, isto é, sem as restrições.

A.1 Otimização Unimodal

Matematicamente, um problema que aborda técnicas de otimização com uma única solução pode ser enquadrado no processo de minimização ou maximização de uma função que pode estar sujeita a restrições [JN99]. Em outras palavras, uma *função unimodal* é uma função que tem somente um pico (máximo) ou um vale (mínimo) em um dado intervalo [Rao]. Desta forma, tem-se a seguinte notação:

- $\mathbf{x}$ é o vetor de parâmetros;
- $\mathbf{f(x)}$ é uma função objetivo;
- $\mathbf{g(x)}$ é um vetor das restrições, isto é, pode conter restrições do tipo desigualdade que devem ser satisfeitas;
- $\mathbf{h(x)}$ é o vetor das restrições do tipo igualdade que devem ser satisfeitas.

Com esta notação, um problema, por exemplo, de minimização pode ser definido como sendo:

min $\mathbf{f(x)}$ onde $\mathbf{x} \in \Omega$.

com as seguintes restrições,

$g_i(\mathbf{x}) \geq 0$ com i = 1,...,n

e $h_k(\mathbf{x}) = 0$ com k = n+1,...,n+p.

Na definição apresentada acima, existem **i** variáveis reais, **n** restrições do tipo desigualdade e **p** restrições do tipo igualdade (sendo o número total de restrições n+p). O espaço de busca da variável é dado por $\Omega \subseteq \Re^n$, correspondente ao conjunto de todos os valores possíveis para a variável **x**.

Na procura da solução de um problema de otimização como o definido acima, o valor ideal ou ponto ótimo $\mathbf{x}^*$, é feita no espaço de busca. De um modo geral, ao se trabalhar com problemas de otimização, supõe-se que o ponto ótimo $\mathbf{x}^*$ existe, é único e, pode ser localizado utilizando um algoritmo de otimização.

Um exemplo de uma função unimodal por ser vista na Figura 16, onde o ponto ótimo pode ser localizado acima do eixo dos X, ou primeiro quadrante.

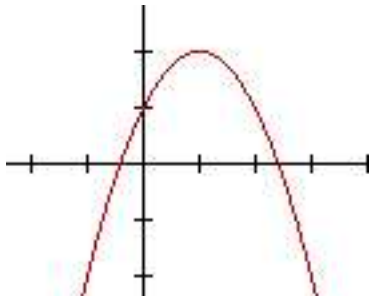

Figura 16: Uma Função unimodal

É importante ressaltar que muitas vezes existem situações na qual não se verifica $\mathbf{x}^*$ ser único. Isto aconteçe quando uma função $\mathbf{f(x)}$ não é limitada inferiormente, ou então $\mathbf{x}^*$ não existe, ou ainda para determinados tipos de funções, $\mathbf{x}^*$ pode não ser único.

A.1.1 Otimização Contínua e Otimização Discreta

Quando os problemas em estudo envolvem a procura de um ponto ótimo através de um conjunto infinito de pontos, isto é, o espaço de busca é infinito, trata-se de Otimização Contínua. Nestes problemas, as variáveis envolvidas são reais. Já problemas que envolvem a procura de um ponto ótimo através de um conjunto finito de pontos são ditos de Otimização Discreta, e envolvem números inteiros ou naturais [JN99].

De acordo com a natureza do trabalho, existe um grande número de problemas cujas variáveis só fazem sentido se tiverem valores inteiros. De outra forma, a resolução de

problemas desse tipo ignorando a natureza inteira das variáveis, isto é, considerando-as reais (e arredondando-as para o inteiro mais próximo) não garante a obtenção de uma solução próxima do ótimo do problema original. Problemas que envolvem variáveis inteiras são ditos problemas de Programação Inteira [SGN96].

A.1.2 Ponto Local e Ponto Global

Geralmente, o que se pretende ao se trabalhar com problemas de minimização é encontrar o ponto para o qual a função atinja o seu mínimo. E com problemas de maximização é encontrar o ponto para o qual a função atinja o seu máximo. Em ambos os casos, o que se procura é o **ponto global ótimo**.

Definição 2.1.1(Mínimo Global)

Um ponto $\mathbf{x}^*$ é mínimo global se $\mathbf{f(x^*)} \leq \mathbf{f(x)}$ para todo $\mathbf{x} \in \Omega$.

Com alguns algoritmos é possível apenas encontrar o **mínimo local** que é o ponto que na sua vizinhança apresenta o menor valor da função. Por vizinhança de $\mathbf{x}$, entende-se o conjunto aberto que contém $\mathbf{x}$ e que está contido no domínio de $\mathbf{f(x)}$.

Definição 2.1.2(Mínimo Local Fraco) Um ponto $\mathbf{x}^*$ é um mínimo local fraco se existe uma vizinhança $\mathbf{Z}$ de $\mathbf{x}^*$ tal que $\mathbf{f(x^*)} \leq \mathbf{f(x)}$ para todo o $\mathbf{x} \in \mathbf{Z}$;

Definição 2.1.3(Mínimo Local Forte) Um ponto $\mathbf{x}^*$ é um mínimo local forte se existe uma vizinhança $\mathbf{Z}$ de $\mathbf{x}^*$ tal que $\mathbf{f(x^*)} < \mathbf{f(x)}$ para todo o $\mathbf{x} \in \mathbf{Z}$ e $\mathbf{x} \neq \mathbf{x^*}$.

A.2 Otimização Multimodal

Se uma função é tida como sendo *multimodal*, isto é, possuidora de vários vales e picos, então a formalização desta função segue o mesmo preceito da formalização da função *unimodal*. Porém, a função multimodal é subdividida em várias partes, e assim, a função é tratada como uma função unimodal em cada parte.

Matematicamente, um problema de otimização multimodal é a minimização ou maximização de um conjunto de funções sujeitas a restrições nas variáveis [YS85]. Desta forma, tem-se a seguinte notação:

- $\mathbf{x}$ é o vetor das variáveis de decisão;
- $\mathbf{f(x)}$ é o vetor das funções objetivo, onde cada componente é uma função que se

pretende maximizar ou minimizar;

- $\mathbf{g(x)}$ é um vetor das restrições, isto é, pode conter restrições do tipo desigualdade que devem ser satisfeitas;
- $\mathbf{h(x)}$ é o vetor das restrições do tipo igualdade que devem ser satisfeitas.

Com esta notação, um problema, por exemplo, de minimização pode ser definido como sendo:

min $\mathbf{f(x)} = (f_1(x), ..., f_s(x))$ onde $\mathbf{x} \in \Omega$.

com as seguintes restrições,

$g_i(\mathbf{x}) \geq 0$ com i = 1,...,n

e $h_k(\mathbf{x}) = 0$ com k = n+1,...,n+p.

Da mesma forma que nas funções de otimização unimodal, as restrições do tipo desigualdade e igualdade definem a região do espaço de busca.

Um exemplo que ilustra bem o conceito de uma função multimodal pode ser visto na Figura 17, onde entre os limites [-6, +6] encontram-se vários picos e vales da função.

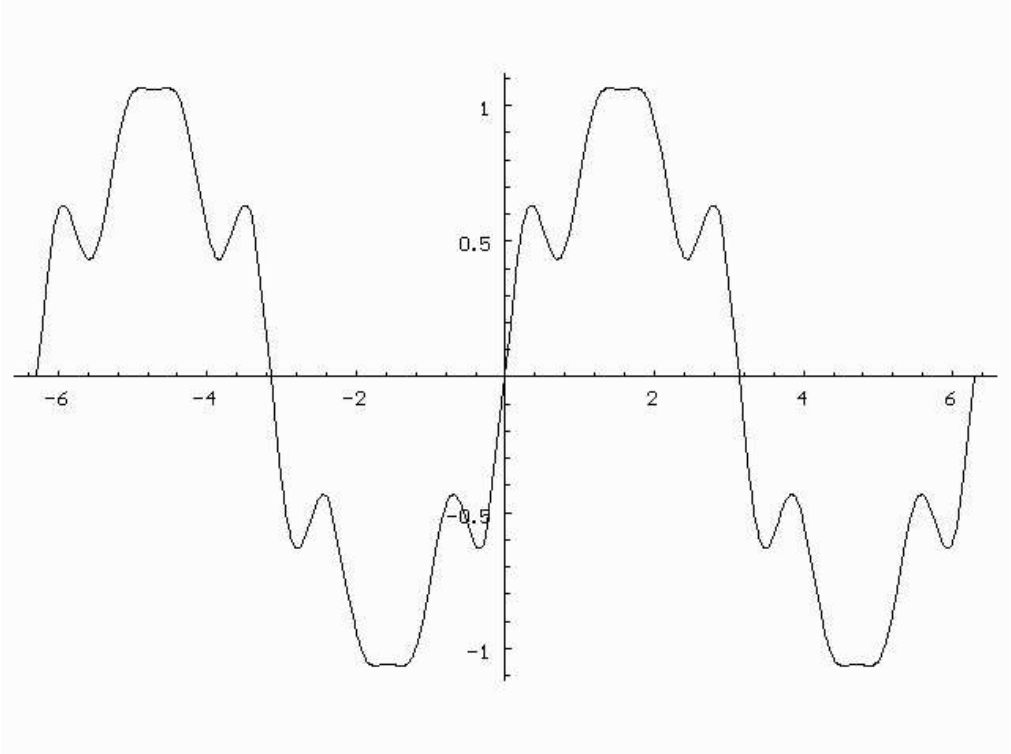

Figura 17: Uma Função multimodal

APÊNDICE B – Algoritmo Multi-PBil

Este apêndice apresenta os algoritmos utilizados no processo de implementação do método Multi-PBil. Estão relacionados o algoritmo principal, e as rotinas pertencentes ao algoritmo.

As rotinas relacionadas são a rotina responsável pela criação dos modelos utilizados no processo de busca; a rotina responsável pela aplicação do fator de criação que determina o valor percentual da quantidade de modelos que serão criados; e a rotina responsável pela eliminação dos modelos com comportamento insatisfatório.

A rotina responsável pela criação dos modelos de probabilidade utiliza a equação

$M_i \leftarrow (1.0 - \mathbf{T.A.})$ * Val_i + $(\mathbf{T.A.})$ * (media da metade dos melhores individuos);

onde destaca-se a variável T.A. que representa a taxa de aprendizagem. Por meio desta rotina, é possível criar quantos modelos de probabilidade forem necessários na realização da busca pelo espaço.

A rotina responsável pela aplicação do fator de criação $(\boldsymbol{F_c})$ é determinante no grau de exigência da diversidade populacional trabalhada. Isto é, $(\boldsymbol{F_c})$ com valor alto, implica em muitos modelos, e $(\boldsymbol{F_c})$ com valor baixo implica em na criação de poucos modelos.

E a rotina responsável pela eliminação dos modelos de probabilidade insatisfatórios faz uso do conceito de Distância Euclidiana. Por meio dela, mede o percentual de proximidade entre os modelos atuantes no espaço de busca e os objetivos já encontrados pelo método.

Algoritmo 4 Algoritmo Multi-PBil

```
Gerar O objetivos;
Gerar N individuos;
Calcular fitness dos N individuos;
Ordenar decrescentemente individuos X^1, ..., X^n;
Criar modelos M_i a partir da população de individuos;
loop
  N ← 1;
  M_i(N) ← cria_modelos(X^i);
  Fator_criacao();
end loop
Execução modelos;
for M_i de 1 ate M_n passo 1 do
  for j de 1 ate fim_geração passo 1 do
    loop
      Gerar N indivíduos usando M_i;
      Calcular fitness dos individuos;
      Ordenar decrescentemente os individuos;
      Procurar objetivos;
      loop
        for modelo M_i de 1 ate N passo 1 do
          if modelo convergiu then
            "Objetivo O_k encontrado"
            Retorne Modelo M_i e Objetivo O_k;
            Saída;
          end if
        end for
      end loop
      Selecionar individuo;
      Atualizar modelo;
      for i=1 ate Tam_modelo passo 1 do
        P_i ← P_i * (1.0 − T.A.) + Val_i * (T.A.);
      end for
      Mutação modelo;
      for j de 1 ate L passo 1 do
        if (rand(0,1] < Mut_Prob) then
          P_i ← P_i * (1.0 − Mut_Sh) + Rand(0,1) * (Mut_Sh);
        end if
      end for
    end loop
    Eliminar_modelos(M_i);
  end for
end for
```

Algoritmo 5 Rotina Cria_Modelos(X^i)

```
M_i ← (1.0 − T.A.) * Val_i + (T.A.) * (media da metade dos melhores individuos);
Retorne(M_i);
```

Algoritmo 6 Rotina Fator Criação Fc

for indivíduo X_i, $i > 1$ ate **n do**
  if Max Pr($X_i \mid M_{i-1}$) < Fc **then**
    M_i(**N**) $\leftarrow$ Cria_Modelos(X_i);
    Retorne (M_i);
  else
    X_i++;
  end if
end for

Algoritmo 7 Rotina Eliminar_Modelos(M_i)

Avaliar e Ordenar os O_n objetivos;
for objetivo O_n encontrado ate último **do**
  Soma (**S**(O_i)) dos valores dos campos de cada objetivo;
  Soma (**S**(M_i)) dos valores dos campos do modelo (M_i);
  if (**S**(M_i)) * 100 / (**S**(O_i)) < ER **then**
    Matar(M_i);
    Saida;
  end if
end for