ALGORITMO EVOLUCIONÁRIO PBIL MULTI_OBJETIVO
APLICADO AO PROBLEMA DA RECARGA DE REATORES NUCLEARES

Marcelo Dornellas Machado

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA NUCLEAR.

Aprovada por:

_____
Prof. Roberto Schirru, D.Sc.

_____
Prof. Fernando Carvalho da Silva, D.Sc.

_____
Prof. Cláudio Márcio do Nascimento Abreu Pereira, D.Sc.

_____
Dr. Celso Marcelo Franklin Lapa, D.Sc.

_____
Dr. Hermes Alves Filho, D.Sc.

_____
Dr. Jorge Luiz Cachoeira Chapot, D.Sc.

RIO DE JANEIRO, RJ - BRASIL
JUNHO DE 2005

MACHADO, MARCELO DORNELLAS

Algoritmo Evolucionário PBIL Multi_Objetivo Aplicado ao Problema da Recarga de Reatores Nucleares [Rio de Janeiro] 2005

XI, 127 p.29,7 cm (COPPE/UFRJ, D.Sc., Engenharia Nuclear, 2005)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1 - Algoritmo Evolucionário PBIL

2 - Recarga de Reatores Nucleares

3 - Problemas Multi_Objetivo

4 - Otimização

I. COPPE/UFRJ II. Título (série)

DEDICATÓRIA

Ao meu pai, à minha mãe sem os quais nada seria possível.

Aos meus irmãos, Maurício, Márcio e Mauro, companheiros de todas as horas.

A Samantha e Gabriel que deixaram nossa família mais completa.

“Quem fala menos ouve melhor,
e quem ouve melhor aprende mais.”

André Luiz – Sinal Verde, 39

Agradecimentos:

Ao Prof. Dr. Roberto Schirru, orientador e amigo, pela sua dedicação e apoio indispensáveis durante a realização deste trabalho de pesquisa;

Aos amigos Carlos Canedo, José Carlos e Mauro, companheiros desde os tempos de mestrado, exemplos de estudantes e de profissionais.

A Kelling, Roberto, Airton, Jorge Wagner, Wagner Sacco, Alan e Pius, companheiros dos bons momentos e das famosas reuniões de alunos;

Aos demais alunos da área de inteligência artificial Gustavo, Berg, Rafael e Anderson;

Ao Dr. Jorge L. C. Chapot por disponibilizar o código RECNOD para a realização dos casos de estudo;

Aos amigos Teresinha, Sidinei, Vanderlei, Mandarano, Gunter, Amory, João Calixto e demais funcionários da Eletronuclear, pela paciência e colaboração com informações sobre o problema da recarga.

Aos funcionários do Laboratório de Monitoração de Processos (LMP) e do programa de Engenharia Nuclear, pela cordialidade com que sempre me trataram e pela grande amizade.

Ao CNPq e à FAPERJ pelo apoio financeiro sem o qual não seria possível a realização desta tese.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

ALGORITMO EVOLUCIONÁRIO PBIL MULTI_OBJETIVO APLICADO AO PROBLEMA DA RECARGA DE REATORES NUCLEARES

Marcelo Dornellas Machado

Junho/2005

Orientador: Roberto Schirru

Programa: Engenharia Nuclear

O problema da recarga de reatores nucleares consiste em determinar o arranjo entre os combustíveis parcialmente queimados e novos no núcleo do reator de forma a se otimizar o próximo ciclo de operação da usina. Este processo de otimização vem sendo resolvido ao longo do tempo empregando-se o conhecimento de um especialista humano, sendo que recentemente, técnicas de inteligência artificial vêm sendo empregadas, com sucesso, neste processo de otimização. Geralmente, as técnicas de inteligência artificial empregam um único objetivo. Entretanto, a maioria dos problemas no campo da engenharia, incluindo o da recarga, apresenta mais de um objetivo (problemas Multi_Objetivo), que normalmente são conflitantes entre si. Este trabalho tem por objetivo desenvolver uma ferramenta para a solução de problemas Multi_Objetivo baseadas no algoritmo PBIL (Population_Based Incremental Learning). Esta nova ferramenta será aplicada na solução do problema da recarga para a usina nuclear Angra 1, visando à formação de uma superfície de Pareto, que permita que um determinado arranjo seja escolhido para o próximo ciclo de operação da usina.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

A MULTI-OBJECTIVE PBIL EVOLUTIONARY ALGORITHM APPLIED TO A NUCLEAR REACTOR CORE RELOAD OPTIMIZATION PROBLEM

Marcelo Dornellas Machado

June/2005

Advisor: Roberto Schirru

Department: Nuclear Engineering

The nuclear reactor core reload optimization problem consists in finding a pattern of partially burned-up and fresh fuels that optimizes the plant's next operation cycle. This optimization problem has been traditionally solved using an expert's knowledge, but recently artificial intelligence techniques have also been applied successfully. The artificial intelligence optimization techniques generally have a single objective. However, most real-world engineering problems, including nuclear core reload optimization, have more than one objective (multi-objective) and these objectives are usually conflicting. The aim of this work is to develop a tool to solve multi-objective problems based on the Population-Based Incremental Learning (PBIL) algorithm. The new tool is applied to solve the Angra 1 core reload optimization problem with the purpose of creating a Pareto surface, so that a pattern selected from this surface can be applied for the plant's next operation cycle.

ÍNDICE

ÍNDICE DE FIGURAS

Capítulo 1

Introdução

A energia, particularmente a energia elétrica, é essencial para o desenvolvimento econômico e social e para uma melhor qualidade de vida da população. A demanda de energia elétrica continua crescendo no mundo, apesar dos grandes esforços direcionados na busca pela eficiência energética e à economia de energia. A demanda de eletricidade nos países em desenvolvimento deve continuar crescendo muito mais rapidamente do que nos atuais países de renda alta.

No mundo atual, cada vez mais se busca formas de geração de eletricidade que reduzam os impactos ambientais, principalmente visando à redução das emissões de gases poluentes, particularmente dos gases que causam o efeito estufa. No setor energético, há um número limitado de opções (das quais a energia nuclear é uma) que sejam tecnicamente maduras e economicamente competitivas para esta geração de eletricidade com baixo impacto ambiental. No ano de 2005, segundo dados da Agência Internacional de Energia Atômica (AIEA), havia em operação, em 33 países, 440 usinas nucleares perfazendo uma capacidade instalada líquida de 366 Gigawatts elétricos (IAEA, 2005).

A energia nuclear já é a segunda maior fonte de produção de eletricidade nos países de renda alta e a terceira maior no mundo inteiro, depois do carvão mineral (40%) e da hidreletricidade (19%), contribuindo com a geração de 16% da produção global de

eletricidade. A Figura 1.1 apresenta a contribuição das principais fontes de geração de eletricidade no mundo (WNA, 2005).

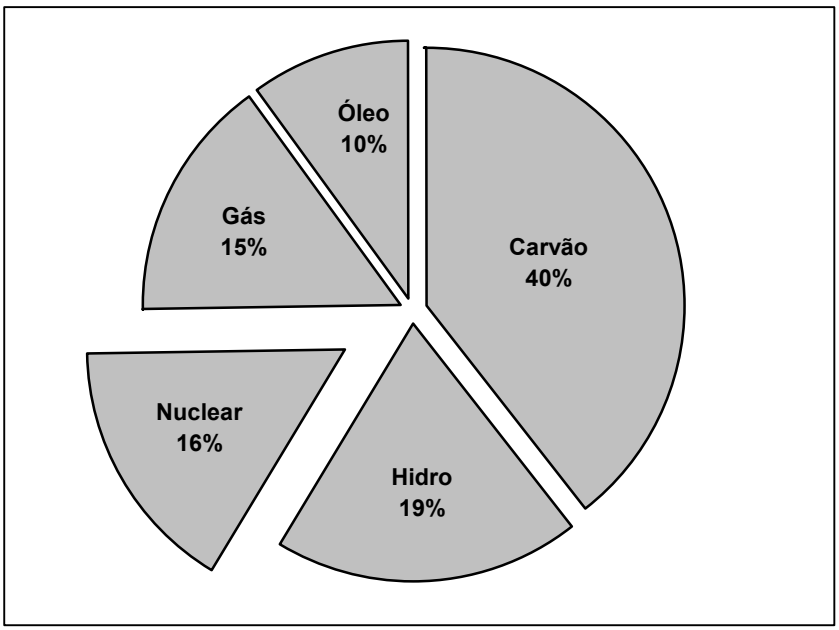


Figura 1.1 – Contribuição das principais fontes de geração de eletricidade no mundo

O Brasil possui no total 1.416 empreendimentos em operação , gerando 91.313.342 kW de potência, sendo 41 destes localizados no estado do Rio de Janeiro, gerando 6.838.420 kW de potência. Entre todos estes empreendimentos em operação, dois são usinas termonucleares (Angra 1 e Angra 2), ambas situadas no estado do Rio de Janeiro e que contribuem com 29,4% da geração no estado (ANEEL, 2005). A Tabela 1.1 apresenta a contribuição de cada fonte geradora no estado do Rio de Janeiro.

Uma forma de se melhorar o desempenho das usinas nucleares é através da otimização de seus ciclos de operação. Após um determinado período de tempo (ciclo de operação) não é mais possível se manter a usina produzindo energia a plena potência. Faz-se necessário o desligamento da usina para substituição de parte de seus elementos

combustíveis (normalmente 1/3) por elementos combustíveis novos. Neste ponto surge um problema - Problema da Recarga de Reatores Nucleares – que consiste em determinar em qual posição do núcleo será colocado cada elemento combustível de forma a se otimizar o próximo ciclo de operação da usina, garantindo que restrições operacionais e de segurança sejam respeitadas.

Tabela 1.1 – Capacidade de Geração do Estado do Rio de Janeiro

Tipo	Quantidade	Potência (kW)	%
CGH	3	984	0,01
PCH	7	28.797	0,42
UHE	10	1.230.520	17,99
UTE	19	3.571.119	52,22
UTN	2	2.007.000	29,35
Total	**41**	**6.838.420**	100

onde:

CGH -> Central Geradora Hidrelétrica
PCH -> Pequena Central Hidrelétrica
UHE -> Usina Hidrelétrica de Energia
UTE -> Usina Termelétrica de Energia
UTN -> Usina Termonuclear

Apesar de possuir uma formulação simples, o problema da recarga de reatores nucleares é um fabuloso problema combinatório onde, dependendo do número de elementos combustíveis no reator, existem aproximadamente 10^{267} possíveis formas de carregamento do núcleo (CARTER, 1997). Além disso, reatores nucleares possuem características altamente não lineares (POON e PARKS, 1992). Isto significa que toda função objetivo e restrições utilizadas para definir e quantificar os padrões de carregamento gerados possuirão, inevitavelmente, características não lineares. Uma conseqüência deste fato é a criação de numerosos pontos de mínimo (ou máximo) locais.

Outra característica do problema da recarga é o tempo necessário para a avaliação de cada uma das possíveis formas de carregamento que venham a ser gerados durante o processo de solução do problema. Este tempo é muito elevado, pois na avaliação da forma de carregamento, é necessário a execução de um código de física de reatores, que apresenta um elevado custo computacional.

A combinação destes atributos:

- alto número de combinações;
- objetivos e restrições não lineares;
- multi-modalidade;
- elevado custo computacional;

descrevem o formidável problema da recarga, que vem sendo solucionado ao longo do tempo empregando-se o conhecimento de especialistas humanos. Recentemente, pesquisadores vêm desenvolvendo/aplicando métodos de otimização "inteligentes" na solução deste problema, tais como o FORMOSA (KROPACKEK e TURINSKY, 1991), o FORMOGA (POON e PARKS, 1992) e o CIGARO (DECHAINE e FELTUS, 1995). Mais recentemente, Chapot (2000) e Machado (2001) utilizaram com sucesso, o Algoritmo Genético e o Sistema de Colônia de Formigas, respectivamente, na otimização do problema da recarga para a usina nuclear Angra 1.

Na maioria dos casos, para a solução do problema da recarga utiliza-se um único objetivo, entretanto os problemas reais no campo da engenharia, entre eles o problema da recarga, apresentam mais de um objetivo (Problemas Multi_Objetivo) que normalmente são conflitantes entre si, ou seja, a melhoria de um objetivo resulta na

deterioração dos demais. Nestes casos de otimização Multi_Objetivo, a determinação do "valor ótimo" para o problema se torna extremamente difícil e a utilização do conceito de ótimo de Pareto (PARETO, 1896) se apresenta como uma forma de solução destes conflitos.

O objetivo deste trabalho, cuja organização é apresentada no parágrafo seguinte, é desenvolver uma ferramenta de otimização de problemas Multi_Objetivo baseada no algoritmo PBIL (Population_Based Incremental Learning) (BALUJA, 1994) (MACHADO et al., 1999) e demostrar a viabilidade de sua aplicação na otimização do problema da recarga de reatores nucleares, para o qual será definido um conjunto de objetivos. Este processo de otimização, visa à formação de uma superfície de Pareto com a finalidade de fornecer ao especialista em recarga várias opções de carregamento do núcleo.

No capítulo 2 são apresentadas as definições e formas de soluções dos problemas Multi_Objetivos. No capítulo 3 são apresentados os métodos de otimização Algoritmo Genético (GOLDBERG, 1989) e PBIL. No capítulo 4, é apresentado o novo algoritmo desenvolvido e os resultados obtidos para a validação do novo método, com a sua aplicação a problemas *benchmarks*. No capítulo 5, é apresentado em detalhe o problema da recarga de reatores nucleares e uma pequena revisão bibliográfica das ferramentas, que empregam métodos de inteligência artificial, desenvolvidas/aplicadas para sua solução. A seguir, no capítulo 6 é apresentado o caso de estudo e resultados obtidos na solução do problema da recarga. Finalmente, no capítulo 7 são apresentadas as conclusões obtidas durante a realização do trabalho e propostas para trabalhos futuros visando dar continuidade à pesquisa.

Capítulo 2

Processos de Otimização

2.1 – Otimização

Na matemática o termo otimização se refere ao estudo de problemas que têm a seguinte forma (BOYD e VANDENBERGHE, 2004):

Dado uma função f: A → R para um conjunto A de números reais, determinar um elemento X_0 pertencente a A de forma que $f(X_0) \leq f(X)$ para todo X pertencente a A (minimização) ou $f(X_0) \geq f(X)$ para todo X pertencente a A (maximização).

Muitos problemas do mundo real e problemas teóricos podem ser modelados utilizando-se esta formulação.

Tipicamente, A é um subconjunto do espaço R^n, freqüentemente determinado por um conjunto de restrições, igualdades e desigualdades que os elementos de A devem satisfazer. Os elementos de A são chamados de soluções viáveis e a função f chamada de função objetivo ou função custo (*Fitness*), sendo esta função responsável por determinar o quão bem um determinado elemento de A soluciona o problema formulado. A solução que maximiza ou minimiza (de acordo com o problema) a função objetivo é chamada de solução ótima.

Logicamente, a definição apresentada acima pode ser bem aceita quando nos referimos a problemas simples, mas para uma grande variedade de outros tipos de problemas não é possível a aplicação desta formulação.

Embora a maioria das tentativas para se definir com precisão termos complexos e de utilização comum seja uma tarefa difícil, é necessário que partamos de um conceito inicial para que se tenha uma idéia sobre as discussões que se seguirão. Desta forma, podemos tentar definir a "otimização como um processo para se fazer alguma coisa melhor" (HAUPT e HAUPT, 1998). Esta definição é bastante ampla, uma vez que pode ser aplicada em qualquer ramo de atividade, como por exemplo:

- nos esportes: fala-se em otimizar o desempenho dos atletas;
- em computação: fala-se em otimizar o tempo de processamento de determinada tarefa.

Quando um engenheiro ou pesquisador surge com uma nova idéia é através de um processo de otimização que se busca, de alguma forma, melhorar esta idéia. Em alguns casos, a otimização consiste em experimentar variações sobre um conceito inicial e usando as informações obtidas aprimorar cada vez mais este conceito.

Quando nos é apresentada uma solução para um determinado problema duas questões surgem naturalmente:

1ª - Seria esta a única solução para o problema ? Na grande maioria das vezes não.

2ª - Esta é a melhor solução possível ? Esta é a difícil questão a ser respondida e para a qual a otimização é uma valiosa ferramenta de auxílio.

Aplicada à engenharia a otimização busca encontrar a "melhor solução" para o problema formulado. O termo "melhor solução" implica que existe mais de uma solução e que estas soluções não são iguais. A definição de melhor será sempre relativa, pois depende de vários fatores, tais como:

- a maneira como o problema é formulado;
- o método de solução a ser empregado;
- a utilização ou não de restrições (limites empregados);
- a utilização ou não do conhecimento de um especialista no problema;

entre outros.

Como a escolhas destes fatores passa a ter influência direta na "melhor solução", esta será sempre dependente da pessoa que realiza o processo de otimização e a quem cabe aceitar uma determinada solução como a "melhor solução".

Alguns problemas, possuem uma solução exata e a "melhor solução" é única e tem seu valor definido, como por exemplo, a solução de uma equação do segundo grau. Nestes casos, não existe como se otimizar a solução do problema, mas sempre podemos buscar otimizar as formas de como se atingir a solução exata.

Um processo de otimização visando à busca da "melhor solução", se aplica bem a problemas que possuam várias soluções de valor máximo ou mínimo em diferentes regiões do espaço de busca. Como por exemplo, a compra de um computador, onde os computadores apresentam os mais variados preços e os mais variados desempenhos. Cabe neste caso ao comprador determinar a relação preço/desempenho (melhor solução) que ele irá aceitar para realizar a compra.

Quanto ao número de objetivos que serão empregados em uma otimização, os problemas podem ser divididos em duas grandes classes:

- os problemas com um único objetivo: este é o caso mais simples de otimização, pois a comparação direta do valor de cada solução apresentada para o problema permite determinar qual a melhor solução;

- os problemas com mais de um objetivo (Multi_Objetivos): neste caso o processo de otimização é um pouco mais complexo, pois necessita que a pessoa responsável pela otimização faça uma tomada de decisão para a escolha da "melhor solução" dentre as possíveis.

A otimização de problemas Multi_Objetivo é o foco de estudo deste trabalho e é mais bem detalhada no item seguinte.

2.2 – Otimização Multi_Objetivo

A grande maioria dos problemas de otimização no mundo real são caracterizados pela utilização de mais de um objetivo em sua solução. Em determinadas situações, os objetivos em estudos são concorrentes, ou seja, ao se buscar a melhoria de um determinado objetivo existe a degradação de um ou mais dos outros objetivos envolvidos no processo. Este fato pode tornar extremamente difícil o processo de otimização.

Pode-se citar como exemplo de problema de objetivos concorrentes os investimentos financeiros. Para este caso, dentre os objetivos existentes, o investidor pode escolher a minimização dos riscos do investimento e a maximização dos lucros a serem obtidos. É sabido, entretanto, que as aplicações que apresentam baixos riscos, geralmente apresentam baixa rentabilidade. Desta forma, a busca por um aumento de rentabilidade pode levar a um aumento do risco, o que corresponde a uma degradação do objetivo inicial de minimizar o mesmo. Neste processo de otimização, é necessário se encontrar um ponto de equilíbrio entre o risco que se deseja aceitar e o rendimento a se obter, escolha esta que deve ser realizada pelo investidor.

A otimização Multi_Objetivo, também chamada otimização Multi_Critérios, é o caminho natural para a solução desta classe de problemas. Este tipo de otimização tem a sua origem no século dezenove, com os trabalhos de Edgeworth e Pareto (PARETO, 1896).

No problema acima e em vários outros casos, não existe uma solução única para o problema e dificilmente os diferentes objetivos serão otimizados levando-se em conta uma única escolha. Assim, algum tipo de tomada de decisão é necessária para se atingir uma solução ótima. Este fato, faz com que otimização Multi_Objetivo apresente um certo grau de nebulosidade, uma vez que não existe, neste tipo de otimização, uma definição amplamente aceita de ótimo, como no caso de problemas com um único objetivo.

Esta falta de um conceito de ótimo para otimizações Multi_Objetivo acarreta uma dificuldade na comparação dos resultados obtidos, devido ao fato de que a decisão sobre

qual é a melhor solução sempre envolver a tomada de decisão por parte do indivíduo que está realizando o processo de otimização.

As otimizações Multi_Objetivo podem ser definidas como: "o problema de se encontrar o vetor de variáveis que satisfaçam as restrições e otimizem o vetor de funções cujo elementos representam a função objetivo. Esta função forma a descrição matemática dos critérios de avaliação, que são normalmente conflitantes entre si. Assim, o termo ótimo significa, entre as soluções obtidas, o valor da função objetivo que é aceita pelo indivíduo que realiza o processo de otimização" (OSYCZKA, 1985).

2.2.1 – Formulação Matemática

Um problema Multi_Objetivo pode ser expresso, de uma forma geral, pela seguinte equação:

$$Min\ F(X) = \left(f_1(X),\ f_2(X),\ \ldots\ldots\ldots\ldots,\ f_k(X) \right) \qquad \textbf{(2.1)}$$

Com $X \in E$

$$X = (X_1,\ X_2,\ X_3,\ \ldots\ldots\ldots\ldots,\ X_n)$$

onde:

$f_1 (X), f_2 (X), \ldots\ldots\ldots\ldots\ldots, f_k (X)$ são as k funções objetivo

$(X_1,\ X_2,\ X_3, \ldots\ldots\ldots\ldots\ldots, X_n)$ são os n parâmetros de otimização

$E \in R^n$ é o espaço de soluções.

Denotando-se por Y os vetores objetivos obtidos por { F(X) | X ∈ E } tem-se:

$$F : E \rightarrow Y \qquad \textbf{(2.2)}$$

ou seja E é mapeado por F em Y.

$Y \in R^k$ é usualmente denominado como o espaço de atributos, sendo δY definido como a fronteira de Y. Para um problema real, geralmente F é não-linear e multi-modal e E pode ser definido por restrições não-lineares e conter variáveis contínuas e/ou discretas.

Definindo-se f_1^* (X), f_2^* (X), ..., f_k^* (X) como o valor mínimo individual de cada função objetivo, uma solução F* (X) pode ser escrita como:

$$F^*(X) = (f_1^*(X),\ f_2^*(X), \ldots\ldots\ldots\ldots,\ f_k^*(X)) \qquad \textbf{(2.3)}$$

Como F* (X) minimiza simultaneamente todos os objetivos, ela é uma solução ideal. Entretanto, este tipo de solução raramente é factível, principalmente para os problemas reais, onde na maioria das vezes os objetivos são conflitantes entre si.

A Figura 2.1 corresponde a uma representação gráfica dos termos apresentados, utilizando-se como exemplo um problema bidimensional contendo dois objetivos.

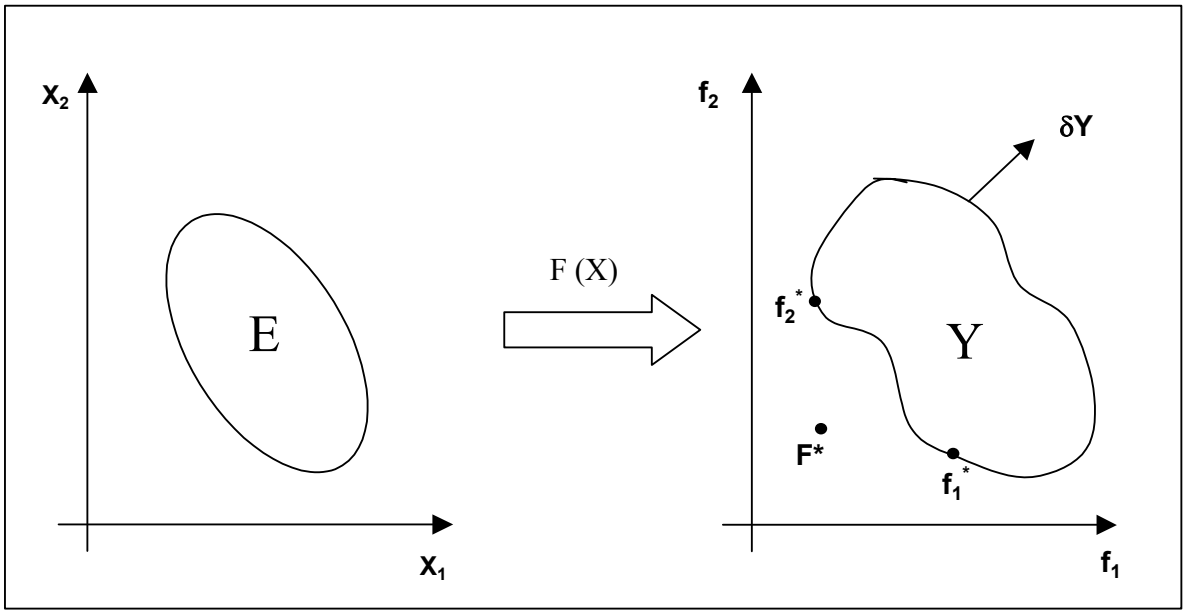


Figura 2.1 – Representação dos elementos de uma otimização Multi_Objetivo bidimensional com dois objetivos.

2.2.2 Definições

A seguir serão apresentadas algumas das definições que são empregadas na área de otimização Multi_Objetivo.

2.2.2.1 Solução não-dominada

Uma propriedade que normalmente é considerada como necessária para toda solução de um problema Multi_Objetivo é que ela não seja dominada.

Uma solução A domina uma solução B se ela for melhor ou igual a B em todos os atributos, sendo que, necessariamente deve existir um atributo no qual o valor de A é melhor que B. Neste caso, B é dito ser uma solução dominada.

Considerando-se um problema de minimização e duas soluções X e Y $\in$ E, é dito que X domina Y se:

$$\forall i \in \{1, 2, 3, ..., k\} : F_i\ (X) \leq F_i\ (Y) \qquad \textbf{(2.4)}$$

$$\exists\ j \in \{1, 2, 3, ..., k\} : F_j\ (X) < F_j\ (Y) \qquad \textbf{(2.5)}$$

2.2.2.2 Superfície de Pareto

O conjunto formado por soluções do problema Multi_Objetivo que não são dominadas por nenhuma outra solução é denominado conjunto de soluções não-dominadas ou conjunto ótimo de Pareto.

A região em R^k formada pelos vetores objetivos das soluções ótimas de Pareto é conhecido como superfície de Pareto. A Figura 2.2 apresenta a superfície de Pareto para um problema bidimensional com dois objetivos.

É claro que qualquer solução que se deseje para um problema Multi_Objetivo deve ser, sempre que possível, um membro da superfície de Pareto. Desta forma, se a solução final é selecionada desta superfície, não existe nenhuma outra solução que seja melhor que ela em todos os atributos e cabe ao indivíduo responsável pela otimização a decisão de qual destes pontos utilizar.

Existe um diferença entre um conjunto de soluções não-dominadas e a superfície de Pareto. Quando fazemos referência ao conjunto de soluções não-dominadas estamos tratando apenas de uma amostra do espaço de busca, enquanto a superfície de Pareto é

definida em relação a todo o espaço de busca. Logo o conjunto de soluções não-dominadas é um subconjunto da superfície de Pareto.

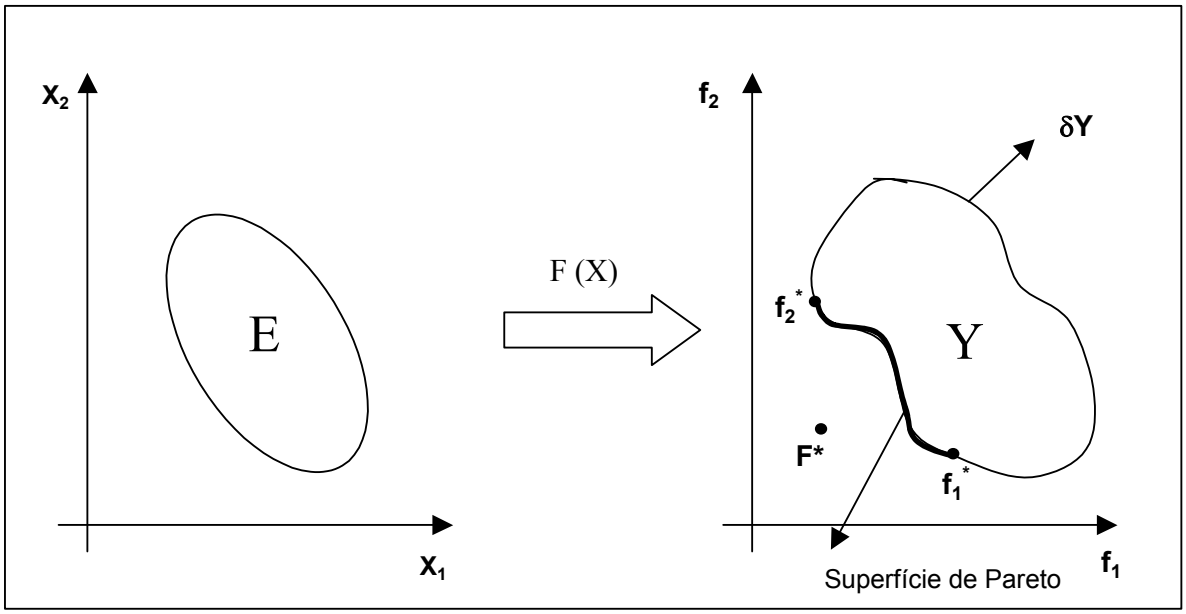


Figura 2.2 - Superfície de Pareto para um problema bidimensional com dois objetivos.

2.2.2.3 Conjunto convexo e não-convexo

O conjunto E é dito convexo se:

$$F(\Phi X + (1-\Phi)Y) \leq \Phi F(X) + (1-\Phi)F(Y) \qquad \textbf{(2.6)}$$

com

X e $Y \in E$

$\Phi \in [0, 1]$

ou seja, E é dito convexo se para qualquer ponto X e Y pertencente ao conjunto, o segmento de linha unindo esses dois pontos também pertence ao conjunto, caso contrário o conjunto E é dito ser não-convexo.

A Figura 2.3 apresenta uma representação simples de um conjunto convexo e um conjunto não-convexo.

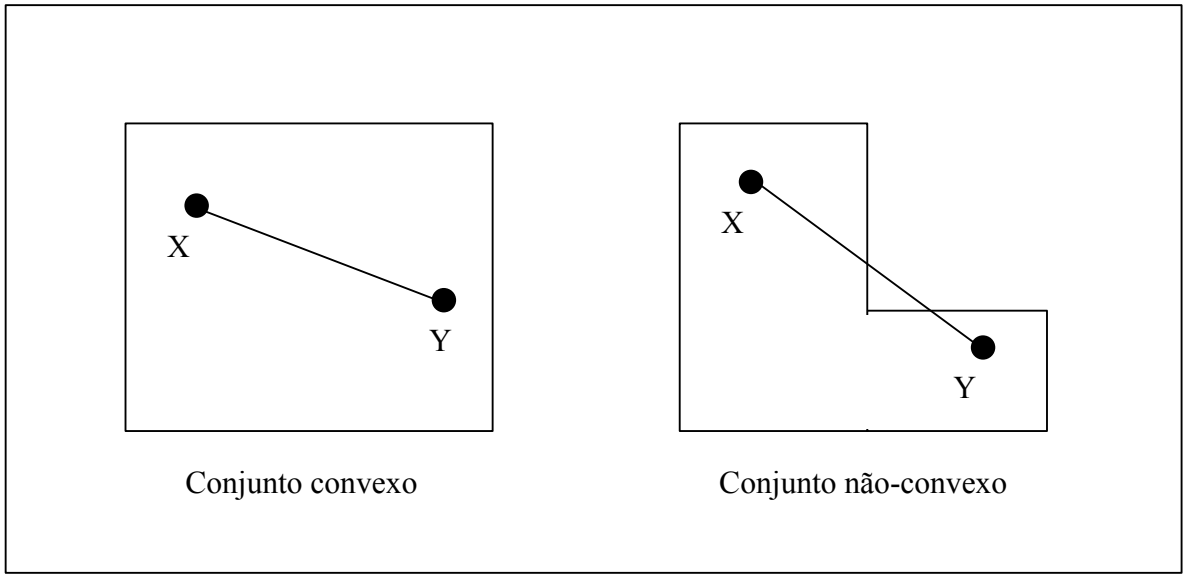


Figura 2.3 – Representação de um conjunto convexo e não-convexo

2.3 Métodos de solução de problemas Multi_Objetivo

Como visto anteriormente, a escolha de uma solução para um problema Multi_Objetivo será sempre dependente do indivíduo responsável pelo processo de otimização.

Assim, os vários tipos de problemas Multi_Objetivo existentes no campo da engenharia e em outros setores, têm sua solução baseada na forma como a pessoa responsável pelo processo de otimização emprega seu conhecimento acerca do problema, principalmente no que se refere à forma como os objetivos do problema serão relacionados .

O emprego deste conhecimento pode ser divido, de forma simples, em quatro grandes grupos (ANDERSSON, 2000):

- sem agregação de conhecimento;
- com agregação de conhecimento a priori;
- com agregação de conhecimento ao longo do processo de otimização;
- com agregação de conhecimento após o processo de otimização.

2.3.1 Sem agregação de conhecimento

Neste método de solução nenhum tipo de conhecimento acerca da relação entre os objetivos envolvidos no problema é utilizado. A forma mais utilizada deste método de solução é a formulação Min-Max.

2.3.1.1 Formulação Min-Max

A formulação Min-Max é baseada na minimização da distância relativa de uma solução encontrada e uma solução ideal do problema (F^*). A distância entre o vetor solução e o vetor ideal é expressa por uma Norma L. Assim, o problema de otimização é formulada da seguinte maneira:

$$Min \left[\sum_{j=1}^{k} \left(\frac{f_j(X) - f_j^*}{f_j^*} \right)^P \right]^{1/P} \qquad \textbf{(2.7)}$$

Com $X \in E$

$1 \leq P \leq \infty$

Na formulação Min-Max, o expoente P fornece as diferentes formas de cálculo de distância.

Para P =1 - formulação simples;

P = 2 – distância Euclidiana;

P = ∞ – norma de Tchebycheff

Nesta formulação é fornecido como resultado final um único ponto na superfície de Pareto. Porém, com alterações no valor da solução ideal e/ou do valor do expoente P, outros pontos na superfície de Pareto podem ser mapeados.

Caso se tenha um processo de otimização demorado (casos em que a função objetivo demanda muito tempo para ser avaliada), a geração da superfície de Pareto por este método se torna extremamente lenta e, em alguns casos, inviáveis.

2.3.2 Com agregação de conhecimento a priori

Neste caso, antes de se iniciar o processo de otimização existe uma análise dos vários objetivos envolvidos de forma a se determinar uma maneira de agregar estes objetivos em uma única função a ser utilizada no processo de otimização. Este talvez seja o método de solução mais utilizado na resolução de problemas Multi_Objetivo.

Existem várias formas de se realizar a agregação dos objetivos, sendo aqui destacadas a soma ponderada, a programação de metas e a lógica nebulosa.

2.3.2.1 Soma Ponderada

Este método consiste na adição de todas as funções objetivos usando diferentes coeficientes de peso para cada uma, visando à formação de uma função objetivo única. Isto significa que o problema de otimização Multi_Objetivo é transformado em um problema de otimização escalar da seguinte forma (STEUER, 1986):

$$Min \sum_{j=1}^{k} \lambda_j \ f_j(X) \qquad \textbf{(2.8)}$$

Com $X \in E$

$$\lambda \in R^k \mid \lambda_j > 0, \quad com \ \sum \lambda_j = 1,$$

onde os λ_j (coeficiente de peso) representam a importância relativa de cada um dos objetivos envolvidos no processo de otimização. A variação destes parâmetros permite que diferentes pontos da superfície de Pareto sejam localizados. Assim, o resultado do processo de otimização pode variar significativamente de acordo com a escolha dos valores dos coeficientes de peso e cabe a quem realiza o processo de otimização definir qual o melhor conjunto a ser utilizado.

Esta é a forma mais fácil e talvez a mais utilizada para a solução de problemas Multi_Objetivo. Porém, como este método utiliza uma combinação convexa dos diferentes objetivos, como na equação 2.8, não é possível se localizar soluções na região não-convexa da superfície de Pareto. Em outras palavras, este método não trabalha bem

em espaços de busca não-convexos (RITZEL, EHEART e RANJITHAN, 1994), o que é uma grande desvantagem do emprego deste método na maioria das aplicações reais, principalmente naquelas onde não se conhece o espaço de busca.

2.3.2.2 Programação de metas

A programação de metas tem origem no trabalho publicado por Charnes e Cooper (1961). Na programação de metas (TAMIZ, JONES e ROMERO, 1998) a pessoa que realiza o processo de busca tem que determinar alvos ou metas que deseja alcançar para cada objetivo. Estes valores de meta devem ser incorporados ao problema como restrições adicionais. As metas podem ser formuladas das seguintes formas:

- maior que ou igual a;
- menor que ou igual a;
- igual a;
- na faixa de.

A função objetivo neste caso, busca minimizar a soma do valor absoluto da diferença entre os valores de meta e o valor atual do objetivo. Em sua formulação mais simples a função objetivo pode ser descrita da seguinte forma:

$$Min \sum_{j=1}^{k} \left| f_j(X) - T_j \right| \qquad \textbf{(2.9)}$$

Com $X \in E$

onde T_j representa o conjunto de alvos ou metas para cada um dos f_j objetivos.

2.3.2.3 Lógica Nebulosa

O conceito de conjuntos nebulosos (ZADEH, 1965) é baseado em uma lógica multi-valorada onde uma declaração pode ser ao mesmo tempo parcialmente verdadeira e parcialmente falsa. Na lógica nebulosa, a função pertinência μ expressa o grau de verdade de uma declaração, na faixa entre $\mu = 0$, indicando uma declaração falsa, até $\mu = 1$ para uma declaração verdadeira.

Em um problema de otimização a função pertinência permite associar um valor normalizado μ_i (f_i (X)) a cada objetivo i de forma a expressar o grau de satisfação em relação a um determinado objetivo.

Estes valores devem ser agregados em um único valor, de forma a se ter um valor geral para o problema. Na lógica binária esta agregação é realizada pelo operador **E**. Entretanto, na lógica nebulosa o operador **E** pode ser implementado por diferentes regras, sendo as mais comuns o operador de mínimo (**Min**) e o operador produto (**Π**). Desta forma, a função objetivo pode ser expressa das seguintes maneiras:

$$F_{fuzzy}\ (X) = Min\left(\mu_1\ (f_1(X), \mu_2\ (f_2(X), ..., \mu_k\ (f_k(X)\right) \qquad \textbf{(2.10)}$$

ou

$$F_{fuzzy}\ (X) = \prod_{j=1}^{k} \mu_j\ (\ f_j\ (X)\ ) \qquad \textbf{(2.11)}$$

e o problema de otimização consiste em maximizar F_{fuzzy} (X) com $X \in E$.

2.3.3 Com agregação de conhecimento ao longo do processo de otimização

Esta classe de métodos necessita que a pessoa que está realizando o processo de otimização forneça informações para a escolha das soluções a medida que o processo evolui.

Estes métodos trabalham baseados na hipótese que a pessoa que realiza a otimização é incapaz de definir, no início do processo, formas de agregação entre os objetivos do problema, geralmente devido à complexidade do mesmo. No entanto, ela é capaz de guiar o processo de busca à medida que o mesmo evolui, ou seja, ela adquire conhecimento sobre o problema a medida que diferentes soluções são apresentadas.

Este método apresenta algumas vantagens tais como:

- não é necessário a manipulação inicial dos objetivos;
- utiliza somente um conhecimento local sobre o problema;
- é possível se adquirir conhecimento sobre o problema ao longo do processo de busca .

Como principais desvantagem da utilização desta forma de agregação de conhecimento, podemos destacar:

- a solução é totalmente dependente da pessoa que realiza o processo de otimização, se existe a troca desta pessoa o processo necessita ser reiniciado;

- exige um grande esforço por parte da pessoa que realiza o processo de busca, uma vez que esta deve analisar todas as soluções geradas a cada passo do processo;
- a solução final depende da qualidade com que a pessoa expressa suas decisões a cada passo do processo.

Como exemplo desta forma de agregação de conhecimento, é apresentado o método STEM a seguir.

2.3.3.1 Método STEM

No método STEM, apresentado por Benayoun et al. (1971), as informações fornecidas são usadas para reduzir o espaço de busca sucessivamente. O problema de otimização é reformulado como a Lp – norma (formulação Min-Max) com fronteiras e pesos para os objetivos. Em sua forma matemática temos:

$$Min \left\{ \sum_{j=1}^{k} \left(W_j^h \left(f_i(X) - f_j^* \right) \right)^P \right\}^{1/P} \qquad \textbf{(2.12)}$$

com $X \in E$

$W_j^h > 0$

$\sum_{i=1}^{k} W_i = 1$

2.3.4 Com agregação de conhecimento após o processo de otimização

Determinadas técnicas de inteligência artificial permitem que se realize um processo de otimização com muito pouco conhecimento acerca do problema a ser resolvido. Como exemplo podemos citar os algoritmos evolucionários, para os quais não é necessário se conhecer o espaço de busca nem relações entre as variáveis empregadas para a solução do problema.

Desta forma é possível o uso dos algoritmos evolucionários para se realizar uma busca através do espaço de soluções, com a finalidade de se formar uma superfície de Pareto que ao final do processo permitirá que a pessoa que realiza o processo de otimização escolha um determinado ponto do conjunto como solução para o problema.

Neste caso não existe a intervenção direta da pessoa que realiza o processo de otimização na localização dos pontos da superfície de Pareto, o que é uma vantagem pois evita o favorecimento de determinadas regiões do espaço de busca. Entretanto, a superfície final gerada pode ter um tamanho extremamente elevado o que irá dificultar o processo de escolha final.

Este método de geração da superfície de Pareto pode utilizar processos com várias execução do algoritmo de otimização, como no caso do Método das Restrições ε ou com uma única execução do algoritmo como no caso dos Algoritmos Genéticos Multi_Objetivos, ambos apresentados a seguir.

2.3.4.1 Método das Restrições ε

Este método é baseado na minimização de um dos objetivos do problema, o de maior importância ou primário, considerando-se os demais objetivos como sendo restrições com limites ε_i. Assim, se realiza o processo de minimização da função objetivo (f_p) do objetivo mais relevante sujeito as restrições adicionais formadas pelos demais objetivos. Fazendo-se alterações nos valores ε_i é possível a geração de uma superfície de Pareto.

Este método pode ser formulado da seguinte maneira: para o objetivo primário p

$$Min \quad f_p\ (X) \tag{2.13}$$

com

$$f_i\ (X) \leq \varepsilon_i \quad para \quad i = 1, 2, 3, ..., k \quad e \quad p \neq i$$

Esta abordagem foi proposta por Ritzel (RITZEL, EHEART e RANJITHAN, 1994) como uma forma simples de aplicação dos algoritmos genéticos na solução de problemas Multi_Objetivo. O processo consiste em fixar o valor da função objetivo de todos os objetivos, com exceção de uma que será utilizada como função de avaliação do algoritmo genético. Com várias execuções do algoritmo genético alterando-se o valor fixado das funções objetivo é possível a formação da superfície de Pareto. O mesmo princípio descrito acima pode ser aplicado a outras ferramentas de otimização, como o Simulated Annealing (SUPPAPITNARM et al. , 1999).

Este processo, obviamente, requer um elevado tempo computacional para a sua realização, o que em determinados problemas do mundo real pode inviabilizar a sua utilização.

2.3.4.2 Algoritmos Genéticos Multi_Objetivo

Nos últimos anos cada vez mais trabalhos vêm sendo publicados, tanto no desenvolvimento de novos tipos de algoritmos genéticos Multi_Objetivo, quanto da aplicação destes algoritmos nos mais diversos tipos de problemas Multi_Objetivo.

A grande vantagem da utilização do algoritmo genético sobre outros métodos é que eles trabalham com uma população de indivíduos, o que permite o desenvolvimento de estratégias que permitam que toda a superfície de Pareto possa ser mapeada empregando-se apenas uma execução do algoritmo genético.

A idéia da utilização de *fitness* para os algoritmos genético baseadas no conceito de Pareto foi proposto por Goldberg (GOLDBERG, 1989). Ele sugere o uso de um método de Classificação (Rank) e seleção para os indivíduos não dominados para mover a população do algoritmo genético em direção à superfície de Pareto em um problema de otimização Multi_Objetivo.

A idéia básica é encontrar um conjunto de indivíduos na população que não sejam dominados pelo resto da população. A estes indivíduos é atribuído o melhor valor de Rank e eles deixam de ser considerados para um futuro teste de dominância. Outro conjunto de indivíduos não dominados é formado com o restante de população e a eles é atribuído o segundo melhor valor de Rank. O processo continua até que todos os indivíduos presentes na população tenham um valor de Rank. A seguir os mecanismos do algoritmo genético (apresentados no próximo capítulo) são aplicados.

Goldberg (1989) também sugere o uso de algum tipo de técnica de nicho (SACCO, 2004) para evitar que o algoritmo genético tenha a convergência para um único ponto da superfície de Pareto.

Capítulo 3

Algoritmos Evolucionários

Algoritmo evolucionário (AE) é um termo genérico utilizado para indicar qualquer algoritmo de otimização baseado em uma população e que utilize operadores inspirados em mecanismos biológicos de evolução, tais como a seleção, reprodução e mutação. Candidatos a solução do problema de otimização fazem o papel de indivíduos desta população e uma função de custo determina a capacidade de um determinado indivíduo sobreviver ou não. A evolução da população se realiza através de repetidas aplicações dos operadores utilizados pelo algoritmo.

Estes algoritmos evolucionários vêm sendo cada vez mais utilizados na solução de problemas reais no campo da engenharia, principalmente devido à sua capacidade de encontrar boas soluções, mesmo que se tenha pouco conhecimento acerca do problema a ser resolvido.

Os algoritmos evolucionários não necessitam de derivadas, como alguns métodos de otimização tradicionais, nem conhecimento acerca do espaço de busca, o que permite que eles sejam utilizados em quase todos os tipos de problemas de otimização.

Dentre os algoritmos evolucionários o algoritmo genético (HOLLAND, 1975), talvez seja o mais conhecido e utilizado. Ele tem sido aplicado com sucesso na otimização de funções, problemas de otimização combinatória, bem como na Biologia, Economia, Finanças entre outros (MITCHELL, 1998). Recentemente, o algoritmo PBIL

(Population_Based Incremental Learning) (BALUJA, 1994), baseado nos princípios do algoritmo genético e do aprendizado competitivo, vem demonstrando boa aplicabilidade na solução de problemas combinatórios complexos (MACHADO, 1999).

Estes algoritmos evolucionários serão apresentados a seguir, com destaque para o algoritmo PBIL que também será objeto deste trabalho.

3.1 O Algoritmo Genético

Os algoritmos genéticos são algoritmos de busca baseados nos mecanismos genéticos e evolucionários dos seres vivos, os princípios conhecidos como seleção natural e sobrevivência do mais apto introduzidos por Charles Darwin no clássico "*The Origin of Species*" (1859).

O algoritmo genético combina o princípio da sobrevivência da função objetivo (*fitness*), que é a forma de avaliação dos indivíduos, com a troca de informações aleatória. Embora a troca de informações seja aleatória, o algoritmo genético é muito diferente de uma simples busca aleatória, devido à sua capacidade de reconhecer tendências acerca das melhores soluções encontradas em cada geração, e utilizar essas informações para direcionar o processo de busca. Uma representação esquemática para o algoritmo genético é mostrada no apêndice A.

Uma das principais diferenças entre os processos convencionais de otimização e os algoritmos genéticos é a utilização, por parte deste, de um conjunto de possíveis soluções para a *fitness* a ser analisada, ao invés de se trabalhar com somente uma única

solução. Tipicamente, uma possível solução é representada por uma cadeia de bits (ou por cadeia de cadeias) correspondendo ao genótipo biológico. Este genótipo define as características do indivíduo quando é decodificado em um fenótipo. Um genótipo é composto de um ou mais cromossomos, onde cada cromossomo é composto de um ou mais genes que tem um certo valor (*allele*) de acordo com o tipo de codificação empregada. O *locus* identifica a posição do gene dentro do cromossomo. Assim cada indivíduo é decodificado no conjunto de parâmetros usados na solução do problema. Finalmente, o conjunto de cromossomos é chamado de população do algoritmo genético. Uma representação gráfica dos termos mencionados é apresentada na Figura 3.1. Comumente é utilizada, no algoritmo genético padrão, uma codificação binária para as variáveis do problema.

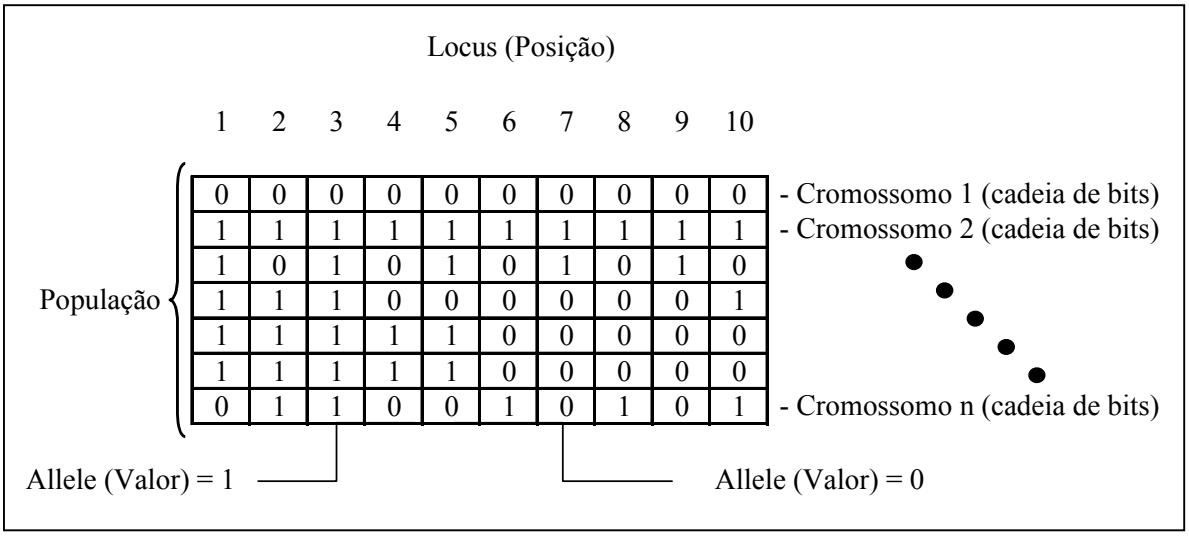


Figura 3.1 – Representação dos componentes do algoritmo genético

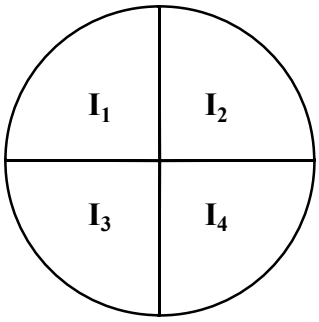
O grupo inicial de possíveis soluções é gerado de forma aleatória dentro do espaço de busca da função, sendo que quanto mais espalhados estiverem estes indivíduos no início, melhor será o processo de busca, uma vez que será possível se obter informações acerca de uma maior região do espaço de busca.

Em cada geração, a *fitness* de cada cromossomo é avaliada a fim de direcionar a aplicação dos operadores genéticos de seleção e recombinação para a criação de uma nova população de soluções.

O operador de seleção é responsável pela escolha dos indivíduos da população que irão participar do processo de recombinação genética. Este operador pode ser implementado computacionalmente de várias maneiras, talvez a mais simples seja a que utiliza uma roleta onde cada indivíduo da população corresponde a uma arco da roleta, de forma proporcional ao valor de sua *fitness*. Supondo, como exemplo, uma população formada por quatro cromossomos com seus respectivos valores de *fitness*, duas roletas para a seleção destes indivíduos são apresentadas na Figura 3.2.

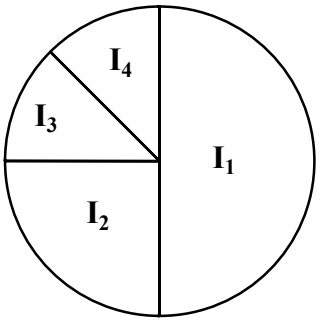
Desta forma, ao se rodar a roleta vão sendo escolhidos os indivíduos que irão participar do processo de recombinação. Com este tipo de seleção, os indivíduos melhores, ou seja, com melhor valor de *fitness*, terão uma maior probabilidade de serem escolhidos, porém sem eliminar a chance de que indivíduos com baixo valor de *fitness* também sejam escolhidos. Este tipo de seleção é denominado *fitness* ou prêmio proporcional.

Indivíduo	*Fitness*
I_1	10
I_2	10
I_3	10
I_4	10
Média	10
Total	40



Roleta para indivíduos de mesma Fitness

Indivíduo	*Fitness*
I_1	20
I_2	10
I_3	5
I_4	5
Média	10
Total	40



Roleta para indivíduo de Fitness diferentes

Figura 3.2 – Representação para seleção por roleta de indivíduos baseada em sua *fitness*

O operador de recombinação utilizado nos algoritmos genéticos, combina a informação contida em dois cromossomos, chamados "pais", selecionados dentre a população de indivíduos e coloca esta informação misturada em um novo cromossomo, chamado "filho", que irá pertencer à próxima geração. Existem várias formas de atuação para o operador de recombinação, de acordo com o tipo de problema e codificação empregados. A forma mais simples do operador, é o operador de recombinação de um ponto, que consiste na escolha, de forma aleatória, de um ponto de corte na cadeia dos cromossomos "pais" e a seguir a troca de parte das cadeias entre eles (Figura 3.3).

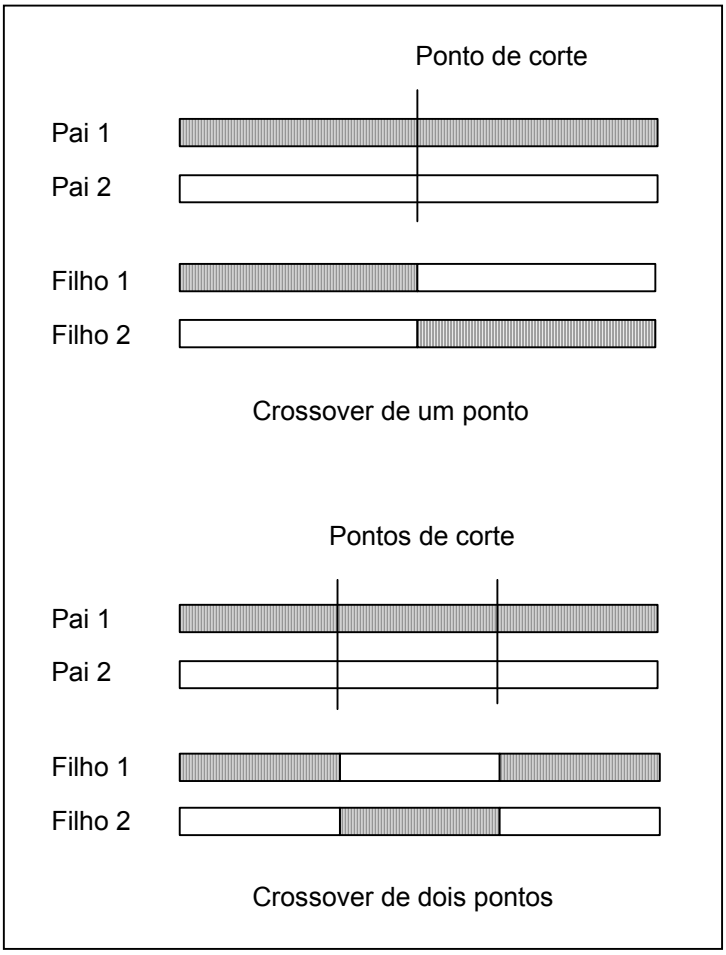


Figura 3.3 – Esquemas de recombinação para o algoritmo genético.

Embora os cromossomos com maior valor de *fitness* tenham uma maior probabilidade de serem selecionados para o processo de recombinação que os cromossomos de menor *fitness*, não se pode garantir que eles estejam presentes na próxima geração. Nem que os cromossomos "filhos" gerados no processo de recombinação serão necessariamente melhores que seus cromossomos "pais". Todavia, devido à pressão de busca seletiva aplicada ao longo das gerações, a tendência é que os cromossomos que venham a ser gerados se tornem melhores.

Um importante fator para o bom desempenho do processo de busca do algoritmo genético é que a diversidade genética, ou seja, o número de indivíduos diferentes presentes na população seja mantida. Quando esta diversidade é perdida pode-se ter uma convergência prematura levando o algoritmo genético a cair em um ponto de ótimo local para o valor da *fitness*. Existem dois mecanismos fundamentais empregados pelo algoritmo genético para manter a diversidade genética. O primeiro, já mencionado, é o mecanismo de seleção probabilística para o processo de recombinação dos cromossomos, o que possibilita que as informações contidas nos cromossomos de menor *fitness* também apareçam nas gerações posteriores.

O segundo mecanismo é a utilização de operador genético de mutação. O operador de mutação introduz mudanças aleatórias nos cromossomos da população. Como exemplo, se um cromossomo é codificado em uma forma binária, o operador de mutação escolhe posições aleatórias ao longo da cadeia e altera o valor do bit, se o valor desta posição for 0 ele é alterado para 1 e se for 1 alterado para 0. Isto faz com que o cromossomo alterado seja posicionado em outra parte do espaço de busca. Visto desta forma, o operador de mutação pode ser considerado como uma caminhada aleatória através do espaço de busca, ajudando a preservar a diversidade genética.

No uso do algoritmo genético nos processos de otimização a definição de uma *fitness* e uma representação eficiente do problema, necessitam de especial atenção a fim de se garantir que informações importantes não sejam perdidas ao longo do processo de busca, o que inviabilizaria a utilização do mesmo.

Como o algoritmo genético emprega mecanismos de seleção estocásticos, é possível se perder a melhor solução encontrada durante o processo de busca. Não existe garantia de que a melhor solução existente na população será selecionada para o processo de recombinação ou que se ela for selecionada, que os operadores de recombinação e/ou mutação não irão destruir alguma das informações que serão passadas aos novos cromossomos gerados. Se a melhor solução for perdida não existe garantia de que ela poderá ser novamente encontrada. Métodos como o elitismo são propostos para a solução deste problema. O elitismo garante que a melhor solução encontrada em uma população seja transferida sem alterações para a próxima população.

Com todas as características descritas, podemos concluir que os algoritmos genéticos constituem uma poderosa ferramenta na otimização e processos de busca em problemas complexos diferindo dos métodos tradicionais em cinco pontos principais:

- Os algoritmos genéticos trabalham com uma codificação do conjunto de parâmetros e não diretamente com eles.
- Os algoritmos genéticos trabalham com uma população de possíveis soluções ao invés de um ponto único.
- Os algoritmos genéticos utilizam somente a função objetivo, não necessitando do emprego das derivadas ou outros conhecimentos auxiliares sobre a função.
- Os algoritmos genéticos utilizam regras de transição probabilísticas e não determinísticas.
- Os algoritmos genéticos necessitam de pouca ou nenhuma informação sobre a relação entre as variáveis do problema.

3.2 O Algoritmo PBIL

O algoritmo PBIL (Population-Based Incremental Learning) (BALUJA, 1994) (BALUJA and CARUANA, 1995) é um método que combina os mecanismo do algoritmo genético com o aprendizado competitivo simples, gerando uma importante ferramenta a ser empregada na otimização de funções numéricas e problemas combinatórios.

O PBIL é uma extensão do "Equilibriun Genetic Algorithm" (EGA) (BALUJA, 1994). O EGA é um algoritmo que tenta descrever a população limite do algoritmo genético por um ponto de equilíbrio, supondo que esta população está sempre se combinando para atingir a convergência. Este processo pode ser visto como um modo de se eliminar a forma explícita do operador de recombinação existente no algoritmo genético.

O objetivo do algoritmo PBIL é criar um vetor probabilidade, contendo valores reais em cada posição, que ao ser utilizado em um processo de decodificação gere indivíduos que apresentem as melhores soluções para a função a ser otimizada. Por exemplo, ao se empregar para a solução de um problema uma codificação binária, o vetor probabilidade irá especificar a probabilidade de cada posição da cadeia de bits conter o valor "1". Desta forma, um bom vetor probabilidade, para um problema codificado com seis bits, a ser obtido ao final do processo de busca é apresentado na Figura 3.4.

0,01	0,99	0,99	0,01	0,01	0,01

Figura 3.4 - Representação de um vetor probabilidade para o algoritmo PBIL

que ao ser decodificado irá gerar, com alta probabilidade, o seguinte cromossomo:

0	1	1	0	0	0

Figura 3.5 – Representação de um indivíduo decodificado a partir do vetor probabilidade da Figura 3.4

Como no algoritmo PBIL toda a população de indivíduos é definida a partir do vetor probabilidade, os operadores empregados para evolução desta população não são definidos diretamente sobre a população, como no caso dos operadores empregados no algoritmos genéticos, mas sim sobre o vetor probabilidade. Os operadores do algoritmo PBIL são derivados daqueles aplicados nos algoritmos genéticos (operador de mutação) e nas redes de aprendizado competitivo (atualização do vetor probabilidade).

Como nos algoritmos genéticos, o algoritmo PBIL também mantém um paralelismo no processo de busca, através da representação de vários pontos distintos do espaço de busca representados por meio de sua população.

A fim de se obter uma maior diversidade da população no início do processo de busca, cada posição do vetor probabilidade é definida com o valor inicial de 0,5. Isto determina que a probabilidade de geração do valor “0” ou “1” em cada posição da cadeia de bits é igual. Esta igualdade na geração de valores faz com que a população inicial gerada pelo PBIL seja aleatória. É extremamente importante para os algoritmos evolucionários que

a sua população inicial seja gerada de forma aleatória, pois permite que o espaço de busca seja bem amostrado no início do processo evitando o favorecimento de uma determinada região do espaço.

De maneira similar ao treinamento de redes de aprendizado competitivo, os valores do vetor probabilidade vão sendo gradualmente alterados do valor inicial de 0,5 para valores próximos a 0,0 ou 1,0, a fim de representar os melhores indivíduos encontrados na população, para cada geração.

A forma de aprendizado utilizada no algoritmo PBIL padrão é semelhante àquela empregada por Kohonen (1990) no "Learning Vector Quantization" (LVQ). Com o LVQ, a rede neural é treinada com exemplos positivos e exemplos negativos, conhecidos *a priori*. De maneira similar, o algoritmo PBIL padrão, realiza a atualização do vetor probabilidade empregando 2 vetores (o melhor e o pior) presentes na população de possíveis soluções da seguinte maneira:

- o melhor vetor, aquele que apresenta o maior valor de *fitness* (classificação correta) altera o vetor probabilidade de forma que este aproxime de sua representação;
- o pior vetor, aquele com o menor valor de *fitness* (classificação incorreta), altera o vetor probabilidade de forma que esta se afaste de sua representação;

Durante o processo de busca, a cada geração, os valores do vetor probabilidade vão sendo alterados de acordo com as seguintes regras:

- Para o melhor vetor

$$P(i) = P(i) \times (1{,}0 - Lr) + vetor(i) \times Lr \qquad \textbf{(3.1)}$$

onde: Lr - taxa de aprendizado;

P (i) - valor do vetor probabilidade na posição i;

vetor (i) - valor do vetor solução escolhido na posição i.

- Para o pior vetor

$$P(i) = P(i) \times (1{,}0 - Lr\_neg) + vetor(i) \times Lr\_neg \qquad \textbf{(3.2)}$$

onde: Lr_neg - taxa de aprendizado negativa;

P (i) - valor do vetor probabilidade na posição i;

vetor (i) - valor do vetor solução escolhido na posição i.

Uma representação esquemática do algoritmo PBIL padrão, bem como uma descrição do aprendizado competitivo é apresentada no apêndice B.

O processo de otimização do algoritmo PBIL padrão tem início com as posições do vetor probabilidade sendo inicializadas com o valor 0,5. Uma população de vetores solução é gerada com base na probabilidade especificada neste vetor probabilidade. Cada vetor solução gerado é avaliado pela função objetivo a ser otimizada, e os indivíduos com a melhor e a pior *fitness* são selecionados para a atualização do vetor probabilidade. Esta atualização consiste em se alterar os valores do vetor probabilidade, de maneira similar à atualização dos pesos das redes de aprendizado competitivo, aproximando-o do melhor vetor solução e afastando-o do pior vetor solução encontrado. A distância com que o vetor probabilidade se aproxima e se afasta dos vetores

selecionados é definida pelos parâmetros denominados taxa de aprendizado e taxa de aprendizado negativo, respectivamente. Após a atualização do vetor probabilidade uma nova população de vetores solução é gerada com base neste novo vetor probabilidade e o ciclo é repetido.

Em Machado (1999) desenvolveu-se uma nova forma de aprendizado para o algoritmo PBIL padrão, na qual os N melhores indivíduos presentes na população são utilizados para a atualização do vetor de probabilidades. No algoritmo PBIL modificado (PBIL_N – Figura 3.6) utiliza-se somente uma taxa de aprendizado, que faz com que o vetor probabilidade seja alterado de forma a se aproximar dos N melhores indivíduos existentes na população. O valor da taxa de aprendizado é alterado de forma proporcional ao valor da *fitness* atribuída a cada um dos N indivíduos selecionados para a atualização do vetor probabilidade.

Assim, a atualização do vetor probabilidade para o algoritmo PBIL_N, em um problema de maximização é realizada da seguinte forma:

$$P(i) = P(i) \times (1{,}0 - Lr\_N) + vetor(i) \times Lr\_N \qquad \textbf{(3.3)}$$

$$Lr\_N = Lr\_N \times \frac{fitness\_N}{fitness\_Melhor} \qquad \textbf{(3.4)}$$

onde: Lr - taxa de aprendizado;

Lr_N - taxa de aprendizado para o indivíduo N;

Fitness_N – função de avaliação do indivíduo N;

Fitness_Melhor – função de avaliação do melhor indivíduo da população;

P (i) - valor do vetor probabilidade na posição i;

vetor (i) - valor do vetor solução escolhido na posição i.

No trabalho anterior, Machado (1999) realizou comparações entre o algoritmo PBIL e o algoritmo PBIL_N demonstrando a viabilidade da utilização deste novo algoritmo. Algumas dessas comparações são apresentadas no Anexo B (Tabelas B.1 e B.2) .

Desta forma, o algoritmo PBIL_N tem como principais parâmetros:

- Número de Indivíduos - determina o número de vetores solução a serem gerados, baseados no vetor probabilidade, em cada geração. Este parâmetro é análogo ao tamanho da população nos algoritmos genéticos.

- Taxa de Aprendizado - determina a distância com que o valor do vetor probabilidade é alterado a fim de se aproximar do vetor solução selecionado na população a cada geração.

Estes dois parâmetros são de fundamental importância no processo de busca do algoritmo PBIL_N, pois o ajuste destes têm impacto direto sobre a exploração do espaço de busca e a aplicação desta exploração. Neste caso, a exploração é a habilidade do algoritmo em pesquisar completamente o espaço de busca, enquanto a aplicação está relacionada a habilidade do algoritmo em utilizar as informações obtidas pela exploração do espaço de busca para orientar as futuras regiões do espaço a serem melhor pesquisadas. Por exemplo, se a taxa de aprendizado é igual a 0 não existe a utilização das informações obtidas pela exploração, e o processo de busca se assemelha

a uma busca aleatória. Por outro lado, se a taxa de aprendizado é muito alta (valores próximos a 1) não é possível se realizar uma boa exploração do espaço de busca pois existe uma convergência rápida do vetor probabilidade. Esta convergência prematura do vetor probabilidade faz com que os novos vetores soluções gerados por este vetor estejam todos localizados em uma região restrita do espaço de busca.

Como se pôde notar, o valor atribuído à taxa de aprendizado afeta diretamente qual parte do espaço de busca será explorado. Para funções que não apresentem ótimos locais, o uso de algoritmos com altas taxas de aprendizado podem trabalhar bem. Porém, para problemas do mundo real onde normalmente não se conhece o espaço de busca e a possibilidade da existência de ótimos locais é alta, a utilização de taxas de aprendizado de valores mais baixos permitirão uma melhor exploração do espaço de busca e como conseqüência melhores resultados.

O algoritmo PBIL_N será utilizado como base para o desenvolvimento do algoritmo PBIL Multi_Objetivo (PBIL_MO), que será apresentado no capítulo seguinte.

```
***Inicializa vetor Probabilidades
        Para i =1 até COMPRIMENTO → P(i) = 0,5

*** Programa Principal
        Enquanto (Não Terminou)
                Para i = 1 até NÚMERO DE INDIVÍDUOS

                *** Gera exemplos
                        Vetor_Solução (i) = gera vetor de acordo com a probabilidade P

                *** Avalia indivíduos
                        fitness (i) = Função Objetivo (Vetor_Solução (i) )

                *** Seleciona os melhores
                        Primeiro_vetor = seleciona o vetor com melhor valor de fitness

                        Segundo_vetor = seleciona o vetor com o segundo melhor valor
                                        de fitness

                *** Atualiza o vetor Probabilidades
                        Para i = 1 até COMPRIMENTO
                                P(i) = P(i) * (1 – Lr_N) + Primeiro_vetor (i) * Lr_N

                        Para i = 1 até COMPRIMENTO
                                P(i) = P(i) * (1 – Lr_N) + Segundo_vetor (i) * Lr_N

                ***Verifica Termino

        ***Fim Enquanto
```

onde:

COMPRIMENTO - número de bits na solução (determinado pela codificação do problema)

Lr_N - taxa de aprendizado

NÚMERO DE INDIVÍDUOS – número de vetores gerados pelo vetor probabilidade (semelhante ao tamanho da população no GA)

Figura 3.6 – Representação esquemática do Algoritmo PBIL_N

Capítulo 4

O Algoritmo PBIL Multi_Objetivo (PBIL_MO)

Neste capítulo são apresentados o algoritmo, baseado no algoritmo PBIL_N, desenvolvido para a solução do problemas Multi_Objetivo - PBIL_MO e os resultados obtidos com este novo algoritmo na solução de algumas funções de teste utilizadas na literatura para a avaliação de algoritmos Multi_Objetivo.

Apesar de algumas destas funções serem simples, elas são de grande utilidade no teste dos algoritmos, pois possuem superfícies de Pareto conhecidas. O conhecimento da superfície de Pareto permite verificar a capacidade do algoritmo Multi_Objetivo em localizar os pontos da superfície e a distribuição destes pontos ao longo da mesma.

Neste trabalho foram escolhidas quatro funções de teste para a aplicação do algoritmo PBIL_MO. Estas funções estão divididas em dois grupos, no primeiro grupo temos três funções numéricas e no segundo grupo uma função combinatória.

4.1 Algoritmo PBIL Multi_Objetivo (PBIL_MO)

Como os algoritmos evolucionários apresentam a vantagem, por trabalharem com uma população de indivíduos, de poderem formar a superfície de Pareto para um problema Multi_Objetivo empregando-se apenas uma execução do código de otimização e baseado nos bons resultados obtidos por Machado e Schirru (2000) com o algoritmo PBIL_N, surge de forma natural o desejo da criação de um algoritmo evolucionário baseado no PBIL_N para a solução de problemas Multi_Objetivo.

Os algoritmos evolucionários normalmente empregam alguma forma de conhecimento para combinar os vários objetivos de um problema Multi_Objetivo em uma função objetivo escalar. A forma mais utilizada para a definição desta função é a combinação linear desses objetivos utilizando pesos associados a cada um dos objetivos do problema. Entretanto, a utilização de pesos tem demonstrado ser uma tarefa complexa, tanto em relação à definição dos pesos em si quanto a sensibilidade da solução final com relação a pequenas alterações nos valores dos pesos (HORN e NAFPLIOTIS, 1993). Para o algoritmo PBIL_MO é empregado o conceito de dominância, apresentado no Capítulo 2, em substituição à utilização de uma função objetivo escalar pré-definida para a avaliação dos indivíduos da população. Esta relação de dominância entre os indivíduos estabelece uma ordem ou classificação na população. Um indivíduo com uma classificação melhor deve ter uma maior probabilidade de ser escolhido para a evolução da população, no caso do algoritmo PBIL_MO para a atualização do vetor de probabilidades.

Durante o processo de evolução do algoritmo PBIL_MO, não existe a garantia que um bom indivíduo presente em uma determinada geração venha a ser novamente formado em uma geração posterior, desta forma, para a formação de uma superfície de Pareto é necessário que os indivíduos não-dominados gerados ao longo de todo o processo de otimização sejam mantidos em uma estrutura. Esta estrutura deve ser capaz de incluir os indivíduos não-dominados presentes nas novas populações geradas e eliminar os indivíduos, existentes na estrutura, que possam vir a ser dominados.

Para inclusão desses dois novos conceitos ao algoritmo PBIL_N, visando à criação do algoritmo PBIL_MO foram desenvolvidos dois novos módulos, sendo o primeiro

chamado Rank_MO, responsável pela classificação dos indivíduos e o segundo de Sup_MO, que implementa a estrutura responsável pela criação da superfície de Pareto.

Para o algoritmo PBIL_MO, cada indivíduo é avaliado usando-se cada um dos objetivos do problema. A seguir é atribuído o melhor valor de Rank para os indivíduos que não são dominados por nenhum outro indivíduo presente na população e estes indivíduos são eliminados do próximo passo da formação do Rank. Dentre os indivíduos restantes é atribuído o segundo melhor valor de Rank e estes também são eliminados e este procedimento prossegue até que todos os indivíduos possuam um valor de Rank (Figura 4.1). Este procedimento é semelhante a forma utilizada por Goldberg (GOLDBERG, 1989) no algoritmo genético.

No processo de escolha dos indivíduos para a atualização do vetor de probabilidades, caso exista mais de um indivíduo com o mesmo valor de Rank para ser escolhido, é criada uma roleta na qual todos os indivíduos possuem a mesma probabilidade de ser escolhido, uma vez que nenhum dos indivíduos é melhor do que os outros. A roleta é girada e o número de indivíduos necessários é selecionado.

O módulo Sup_MO tem por objetivo armazenar os pontos que não são dominados ao longo de todas as gerações do algoritmo PBIL_MO. Para cada geração, os indivíduos contidos no módulo Sup_MO são atualizados para incluir os pontos que não são dominados e a eliminação dos pontos que passam a ser dominados. Esta procedimento é importante, pois como nos algoritmos genéticos, não existem também no algoritmo PBIL_MO a garantia de que um indivíduo criado em uma determinada geração venha a ser criado novamente em gerações futuras, nem que um indivíduo que não seja

dominado na geração atual não possa vir a ser dominado em uma geração futura e que indivíduos que não sejam dominados nesta geração não sejam dominados por indivíduos de gerações passadas.

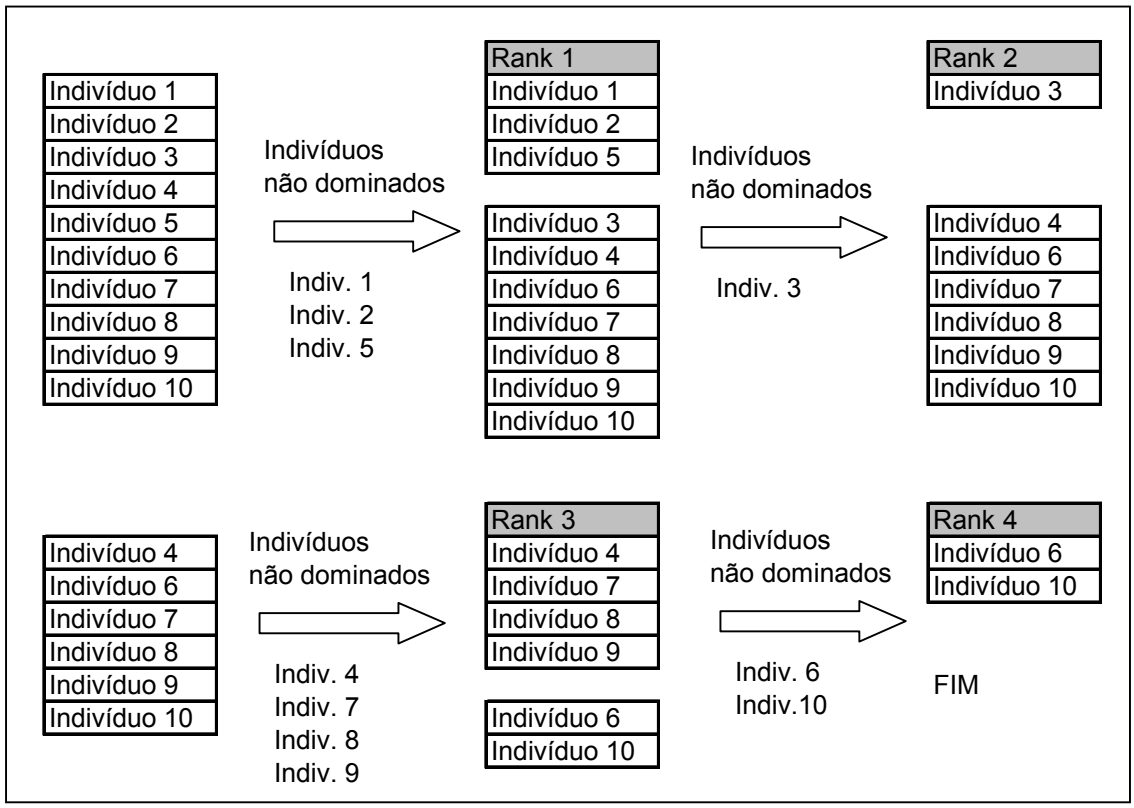


Figura 4.1 Esquema de formação do Rank do Algoritmo PBIL_MO

A estrutura do novo algoritmo desenvolvido é apresentado na Figura 4.2, onde são destacados os dois novos módulos e as alterações realizadas.

O desempenho do algoritmo PBIL_MO foi verificado empregando-se algumas funções de teste apresentadas na literatura. Estas funções bem como os resultados obtidos são apresentados no próximo item.

***Inicializa vetor Probabilidades

Para i =1 até COMPRIMENTO → P(i) = 0,5

*** Programa Principal

Enquanto (Não Terminou)

Para i = 1 até NÚMERO DE INDIVÏDUOS

*** Gera exemplos

Vetor_Solução (i) = gera vetor de acordo com a probabilidade P

***** Avalia indivíduos**

Para j = 1 até NÚMERO OBJETIVOS

Avaliação(i)_(j) = Função Objetivo_(j) (Vetor_Solução (i))

***** Criação do Rank**

Rank_MO (Avaliação) → fitness(i) = Rank (i)

*** Seleciona os melhores

Primeiro_vetor = seleciona o vetor com melhor valor de fitness

Segundo_vetor = seleciona o vetor com o segundo melhor valor de fitness

*** Atualiza o vetor Probabilidades

Para i = 1 até COMPRIMENTO

P(i) = P(i) * (1 – Lr_N) + Primeiro_vetor (i) * Lr_N

Para i = 1 até COMPRIMENTO

P(i) = P(i) * (1 – Lr_N) + Segundo_vetor (i) * Lr_N

***** Verifica os elementos da Superfície**

SUP_MO (Vetor_Solução (i))

***Verifica Termino

***Fim Enquanto

onde:

COMPRIMENTO – número de bits na solução (determinado pela codificação do problema)

Lr_N **- taxa de aprendizado**

NÚMERO DE INDIVÏDUOS – número de vetores gerados pelo vetor probabilidade (semelhante ao tamanho da população no GA)

Figura 4.2 – Representação esquemática do Algoritmo PBIL_MO

4.2 – Funções de teste numéricas

A seguir são apresentados os resultados obtidos com a aplicação do algoritmo PBIL_MO à classe de funções de teste numéricas.

4.2.1 – Funções de Schaffer

As funções de Schaffer (ZITZLER, DEB e THIELE, 2000) são funções bidimensionais definidas pelas seguintes equações:

$$F_1 = x^2$$

$$F_2 = (x-2)^2$$

$$com \qquad -1{,}0 \leq x \leq 3{,}0$$

As Figuras 4.3 e 4.4 apresentam, respectivamente, os gráficos das funções F_1 e F_2 de Schaffer.

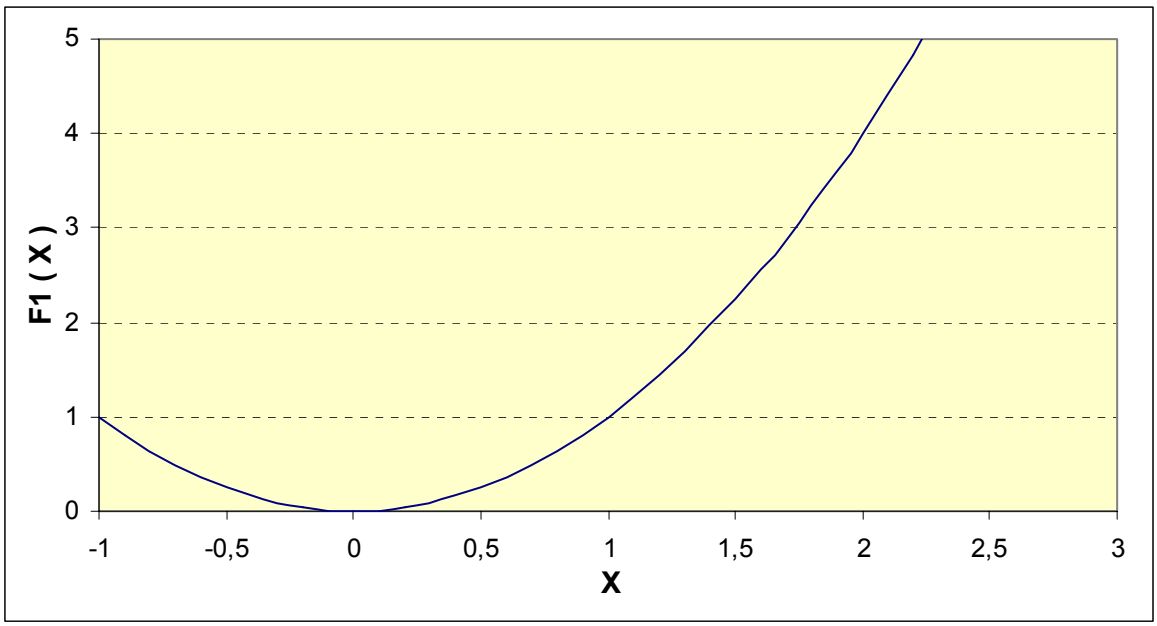


Figura 4.3 - Gráfico da função F_1 de Schaffer

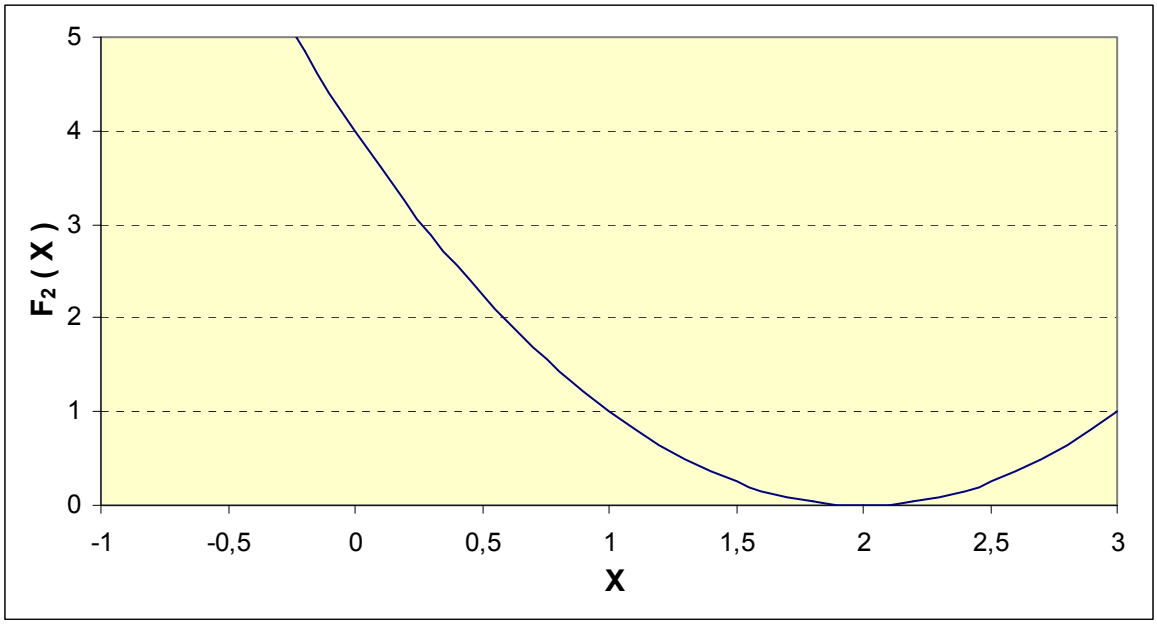


Figura 4.4 - Gráfico da função F_2 de Schaffer

Para esta função de teste deseja-se minimizar o valor das duas funções. Com este objetivo e observando os gráficos anteriores podemos notar que a superfície de Pareto formada se localiza entre X igual a zero e X igual a dois, pontos onde as funções F_1 e F_2 apresentam seus valores de mínimo respectivamente. A Figura 4.5 apresenta a superfície de Pareto deste problema.

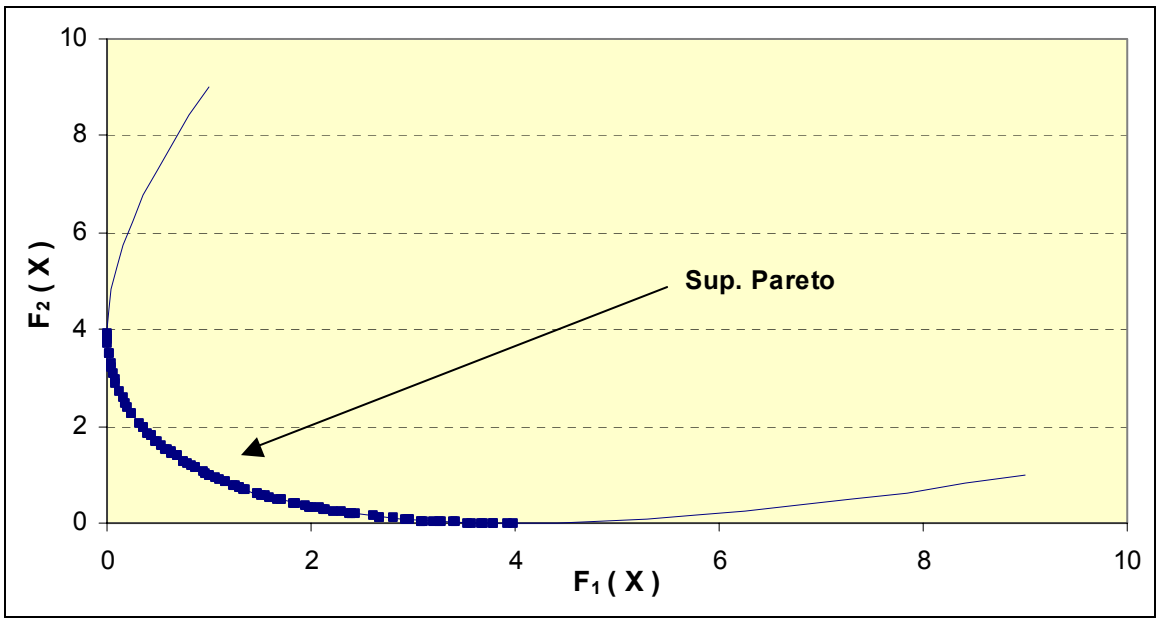


Figura 4.5 - Gráfico da superfície de Pareto para as funções de Schaffer

Para as funções de Schaffer foi utilizada pelo algoritmo PBIL_MO uma codificação binária da população, com 14 bits por indivíduo e um tamanho de população igual a 50 indivíduos. Os indivíduos binários foram decodificados em valores inteiros e depois ajustados para o intervalo de busca das funções. Foi determinado um número máximo de 100 indivíduos na superfície de Pareto.

Várias combinações entre taxas de aprendizado e sementes aleatórias foram utilizadas, com a finalidade de observar o comportamento do algoritmo PBIL_MO, principalmente em relação às variações da taxa de aprendizado que é o principal parâmetro do algoritmo PBIL_MO.

Para este problema os resultados obtidos não apresentaram grandes variações entre si. A Figura 4.6 apresenta o gráfico típico obtido pelo algoritmo PBIL_MO para as funções de Schaffer.

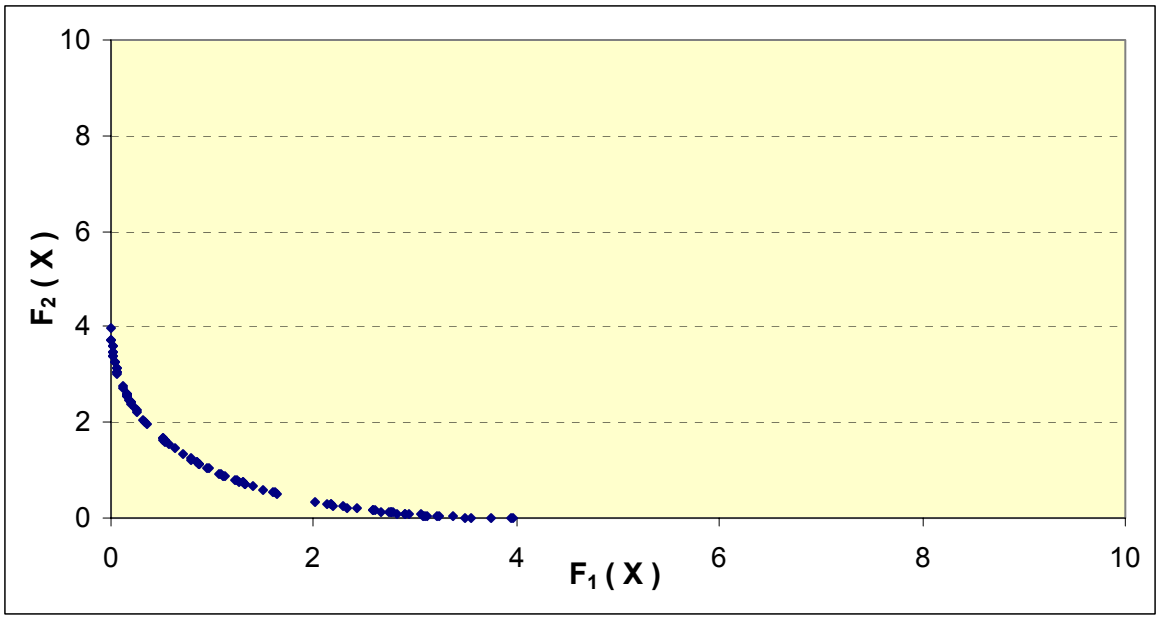


Figura 4.6 - Gráfico da superfície de Pareto para as funções de Schaffer com o algoritmo PBIL_MO

Para as funções de Schaffer, em todos os teste realizados, o algoritmo PBIL_MO foi capaz de mapear vários pontos ao longo da superfície o que permite se ter uma boa visualização da mesma.

4.2.2 – Funções de Schaffer2

As funções de Schaffer2 (ZITZLER, DEB e THIELE, 2000) são funções bidimensionais definidas da seguinte forma:

$$F_1 = \begin{cases} x-1 & se \quad x \leq 1 \\ x-2 & se \quad 1 < x \leq 3 \\ 4-x & se \quad 3 < x \leq 4 \\ x-4 & se \quad x > 4 \end{cases}$$

$$F_2 = (x-5)^2$$

$$com \quad -5{,}0 \leq x \leq 10{,}0$$

O objetivo é minimizar as duas funções e graficamente temos a seguinte superfície de Pareto, que é apresentada na Figura 4.7.

Para as funções de Schaffer2 foram utilizadas pelo algoritmo PBIL_MO uma codificação binária da população, com 17 bits por indivíduo e um tamanho de população igual a 50 indivíduos. Os indivíduos binários foram decodificados em valores inteiros e depois ajustados para o intervalo de busca das funções. Foi determinado um número máximo de 100 indivíduos na superfície de Pareto.

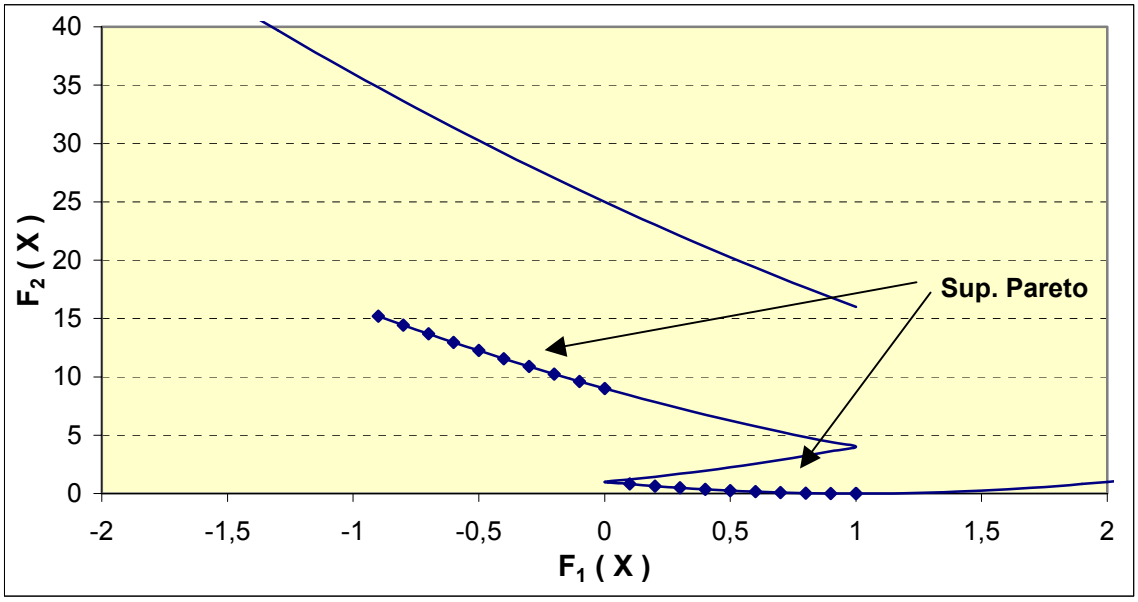


Figura 4.7 - Gráfico da superfície de Pareto para as funções de Schaffer2

Neste caso também foram utilizadas várias combinações entre taxas de aprendizado e sementes aleatórias no teste do algoritmo PBIL_MO.

A Figura 4.8 apresenta um gráfico típico obtido para as funções de teste Schaffer2 empregando-se o algoritmo PBIL_MO. Para estas funções de teste, a utilização de taxas de aprendizado muito elevadas, faz com que uma das partes da superfície de Pareto seja melhor explorada que a outra (Figura 4.9). Cabe destacar que taxas de aprendizados muito elevadas não são normalmente utilizadas em aplicações reais uma vez que conduzem a pontos de ótimo local com a utilização do algoritmo PBIL_N (MACHADO, 1999). Neste caso a utilização destas taxas tem por finalidade verificar se o mesmo comportamento é apresentado pelo algoritmo PBIL_MO.

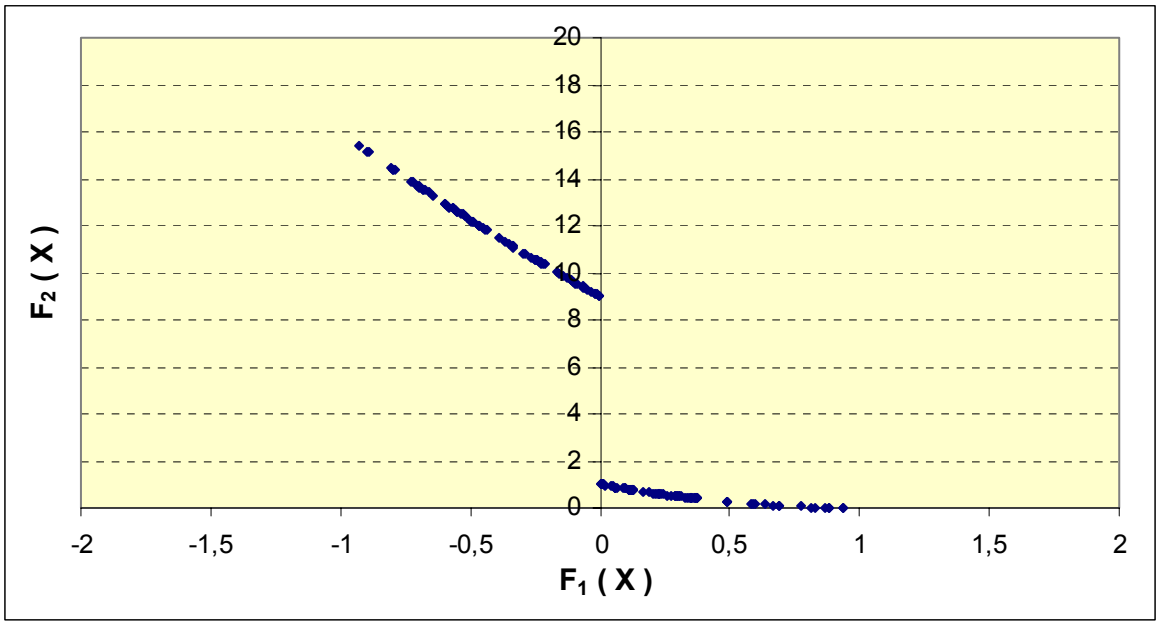


Figura 4.8 - Gráfico da superfície de Pareto para as funções de Schaffer2 com o algoritmo PBIL_MO

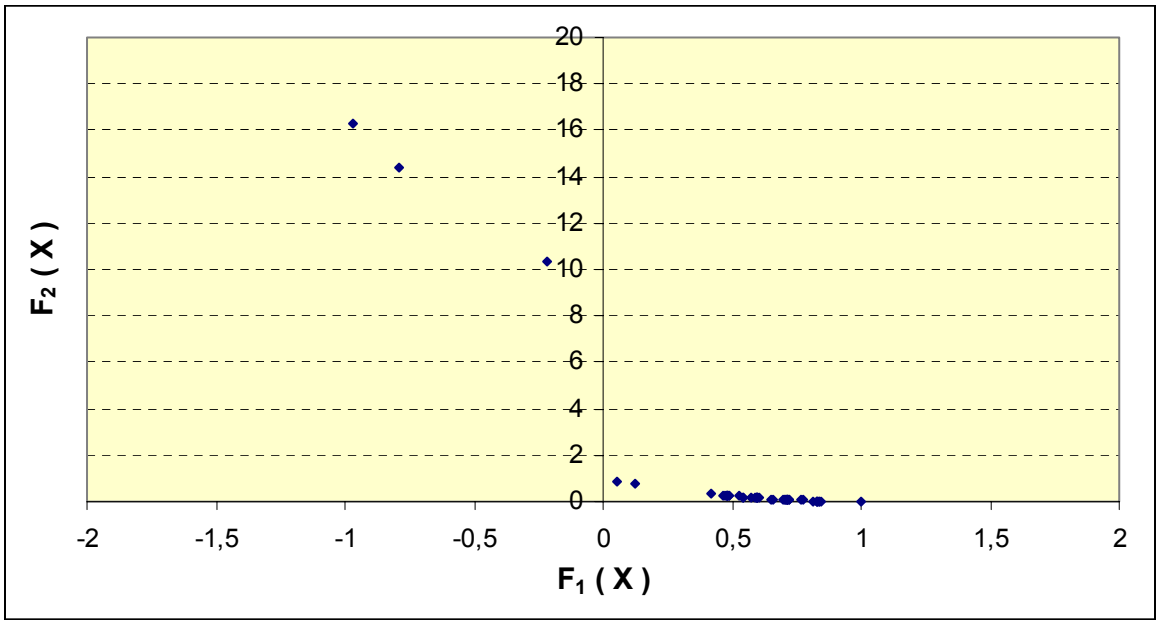


Figura 4.9 - Gráfico da superfície de Pareto para as funções de Schaffer2 com o algoritmo PBIL_MO (Taxas_Altas)

Para as funções de Schaffer2, observamos que a utilização de uma taxa de aprendizado muito elevada faz com que o algoritmo PBIL_MO tenha uma convergência prematura

para uma das regiões da superfície de Pareto, fazendo com que apenas esta região da superfície seja bem explorada. Esta característica de convergência prematura, quando da utilização de taxas de aprendizado elevadas, também é observada no algoritmo PBIL padrão.

No caso de taxas de aprendizado mais baixas, a convergência prematura é evitada, levando a uma melhor exploração do espaço de busca. Este fato faz com que toda a superfície de Pareto seja melhor mapeada.

4.2.3 – Função de Kursawe

As funções de Kursawe (KURSAWE, 1991) são definidas da seguinte forma:

$$F_1(\vec{x}) = \sum_{i=1}^{n-1} (10 - \exp(-0.2\sqrt{x_i^2 + x_{i+1}^2}))$$

$$F_2(\vec{x}) = \sum_{i=1}^{n} (|x_i|^{0.8} + 5\sin(x_i)^3)$$

$$com \quad n = 3$$

$$-5{,}0 \leq x_1, x_2, x_3 \leq 5{,}0$$

O objetivo deste problema é minimizar as duas funções. Graficamente temos na Figura 4.10 a representação da superfície de Pareto.

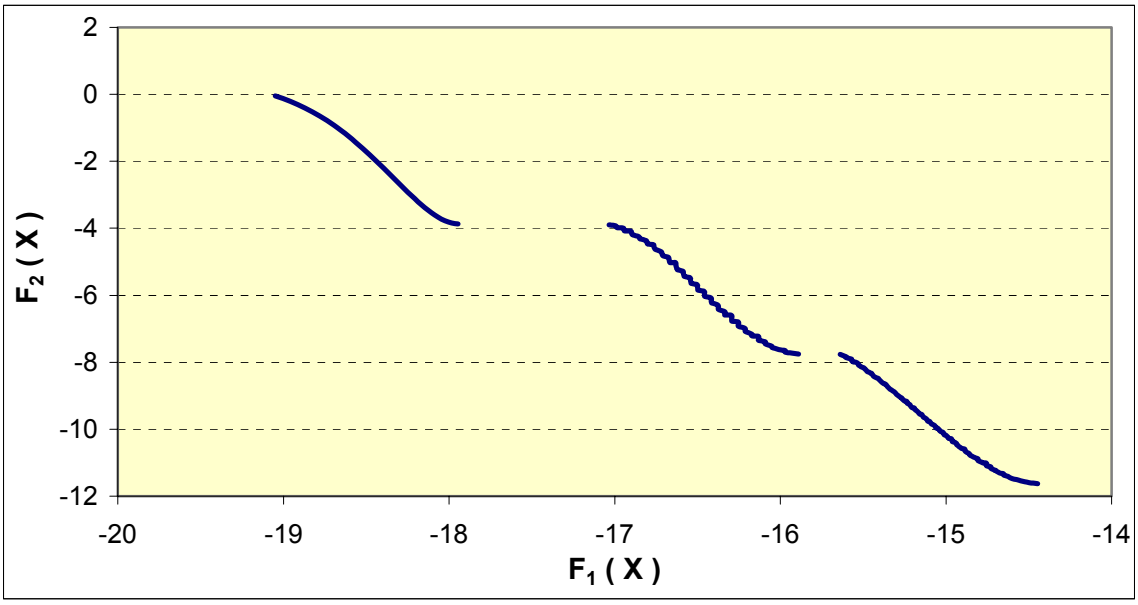


Figura 4.10 - Gráfico da superfície de Pareto para as funções de Kursawe

Para as funções de Kursawe foram utilizadas pelo algoritmo PBIL_MO uma codificação binária da população, com 20 bits por indivíduo, e um tamanho de população igual a 50 indivíduos. Os indivíduos binários foram decodificados em valores inteiros e depois ajustados para o intervalo de busca das funções. Foi determinado um número máximo de 200 indivíduos na superfície de Pareto.

Neste caso também foram utilizadas várias combinações entre taxas de aprendizado e sementes aleatórias no teste do algoritmo PBIL_MO.

Um gráfico típico obtido para as funções de Kursawe, empregando-se valores de taxa de aprendizado mais baixas, é apresentado na Figura 4.11. Neste caso a utilização das taxas de aprendizado muito elevadas também levou a escolha de uma das regiões da superfície de Pareto, o que dificulta uma melhor exploração do espaço de busca. Neste

caso, a utilização de algum método para manter a diversidade da população (como por exemplo o método de Nichos) pode ser empregado a fim de se verificar se é possível o mapeamento de toda a superfície de Pareto mesmo com a utilização de taxas de aprendizado muito elevadas.

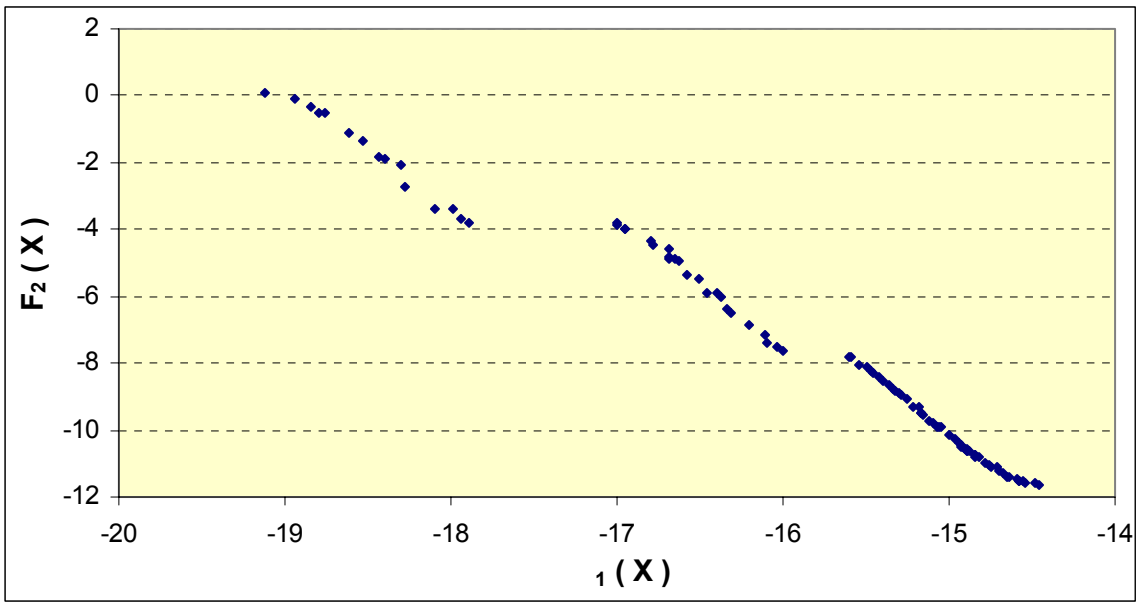


Figura 4.11 - Gráfico da superfície de Pareto para as funções de Kursawe com o algoritmo PBIL_MO

4.3 – Função de teste combinatória

Para o teste do algoritmo PBIL_MO em funções combinatórias foi escolhido um problema de maximização, o problema da sacola (0 – 1) Multi_Objetivo, apresentado a seguir.

4.3.1 - O Problema da Sacola (0 – 1) Multi_Objetivo

O problema da sacola (0 – 1) Multi_Objetivo é um problema NP-Completo. Ele é uma generalização do problema da sacola (0 – 1) simples. No problema da sacola (0 – 1) simples, dada uma sacola de capacidade fixa e um conjunto de n itens, cada um com um valor e um determinado volume, deseja-se determinar quais os itens que devem ser colocados na sacola de modo a se maximizar a soma dos valores destes itens escolhidos sem que a soma dos volumes destes ultrapasse a capacidade da sacola. O problema da sacola (0 - 1) restringe o número de cada item a zero (o item não foi escolhido para ser colocado na sacola) ou um (o item foi escolhido para ser colocado na sacola).

De forma matemática temos:

Maximizar $\sum_{j=1}^{n} p_j \times x_j$

com $\sum_{j=1}^{n} v_j \times x_j \leq C$

onde: C = capacidade da sacola

j = item do conjunto de n itens

p_j = valor do item j

v_j = volume do item j

x_j = vetor de escolha de itens

Já no problema da sacola (0 – 1) Multi_Objetivo, existem k sacolas com suas respectivas capacidades e um conjunto de n itens, onde cada item possui um valor e um determinado volume de acordo com a sacola em que é colocado. Neste caso, deseja-se determinar quais os itens que devem ser colocados ao mesmo tempo nas k sacolas de modo a se maximizar, para cada sacola, a soma dos valores destes itens escolhidos sem que a soma dos volumes destes ultrapasse a capacidade da respectiva sacola.

De forma matemática temos:

Maximizar $\sum_{j=1}^{n} p_{jk} \times x_j$

com $\sum_{j=1}^{n} v_{jk} \times x_j \leq C_k \qquad \forall k = 1, 2, \ldots\ldots, m$

onde: C_k = capacidade da sacola k

j = item do conjunto de n itens

$p_{j\,k}$ = valor do item j colocado na sacola k

$v_{j\,k}$ = volume do item j colocado na sacola k

x_j = vetor de escolha de itens

Para teste do algoritmo PBIL_MO foi escolhido um problema com 2 sacolas e 100 itens (Sac_2/100) apresentado por Zitzler (ZITZLER, LAUMANNS e THIELE, 2001) (ZITZLER e THIELE, 1999), para o qual a superfície de Pareto é mostrada na Figura 4.12 e os dados do problema apresentados no anexo C.

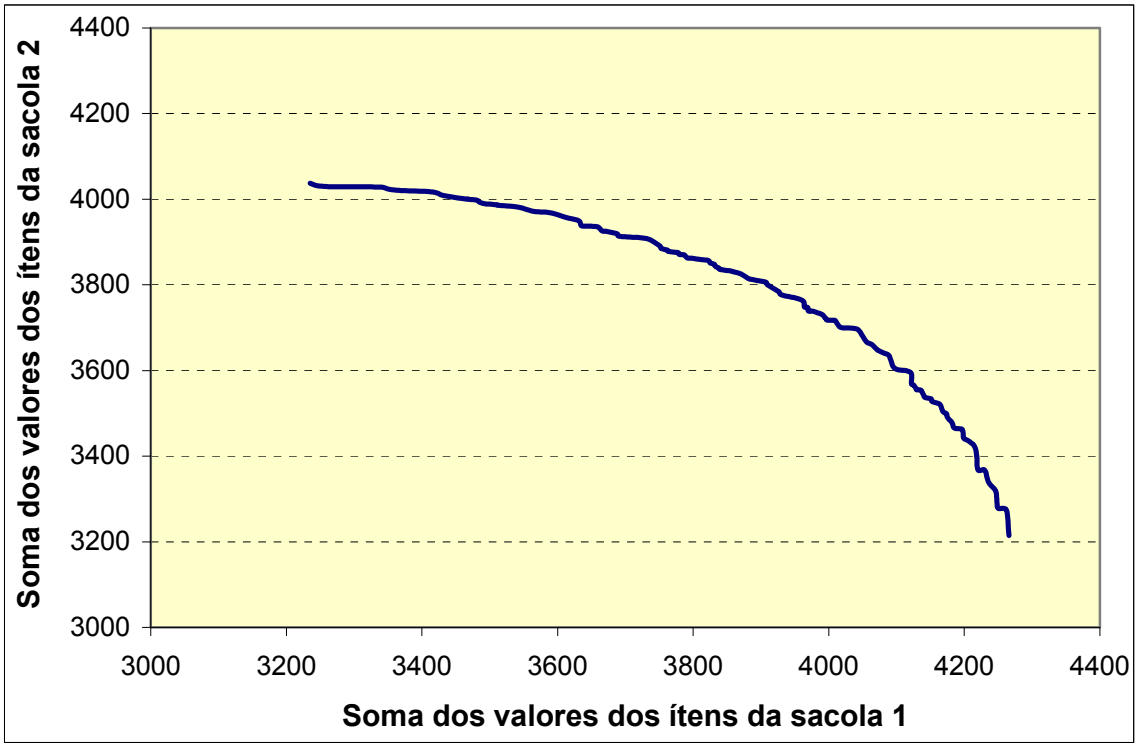


Figura 4.12 – Superfície de Pareto para o problema Sac_2/100

Para a solução do problema da sacola Sac_2/100 com o PBIL_MO utilizamos uma codificação binária. Cada indivíduo da população é representado por um vetor com 100 posições, onde cada posição do vetor corresponde a um item do problema. Neste caso foi utilizada uma população de 50 indivíduos com um número máximo de 100 indivíduos na superfície de Pareto.

Neste caso não é necessário uma decodificação dos indivíduos da população, uma vez que o valor direto dos bits corresponde a seleção (valor igual a um) ou não (valor igual a zero) do item a ser colocado na sacola.

A seguir são apresentados alguns gráficos que apresentam o resultado obtido com o algoritmo PBIL_MO junto com a superfície de Pareto (Figura 4.13) e nas Figuras 4.14 e 4.15 comparado com resultados obtidos por Zitzler (ZITZLER, LAUMANNS e THIELE, 2001) (ZITZLER, DEB e THIELE, 2000) empregando-se o NSGA (Non-dominated Sorting Genetic Algorithm) (SRINIVAS e DEB, 1994) e o NPGA (Niched Pareto Genetic Algorithm) (HORN e NAFPLIOTIS, 1993).

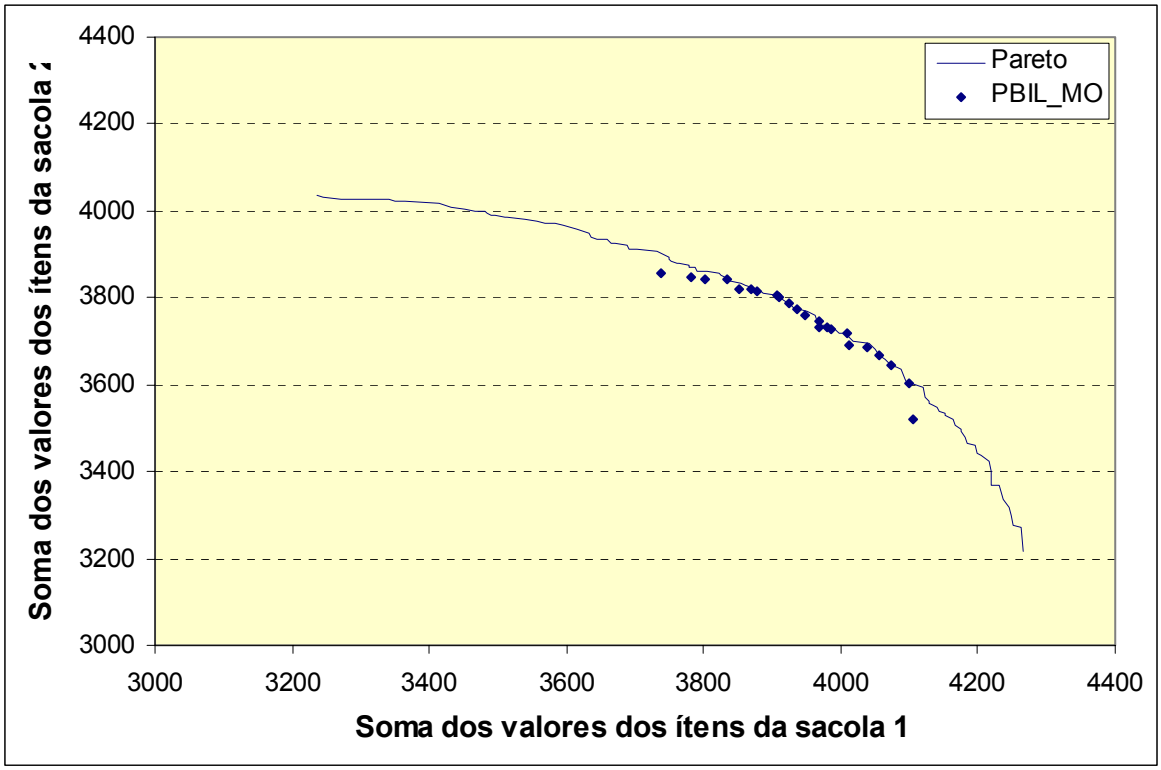


Figura 4.13 – Superfície de Pareto + superfície do PBIL_MO para o Sac_2/100

Da Figura 4.13 podemos observar que a maioria dos pontos que formam a superfície de Pareto obtidos com o algoritmo PBIL_MO são idênticos aos da superfície de Pareto apresentada por Zitzler (ZITZLER, LAUMANNS e THIELE, 2001), porém estes pontos só cobrem uma parte de toda a superfície de Pareto. Este fato pode ser justificado pela não existência no algoritmo PBIL_MO de um método de nicho, por exemplo, para garantir que pontos muito próximos não sejam apresentados como as melhores soluções ao algoritmo PBIL_MO, fazendo com que apenas uma parte do espaço de busca seja explorado.

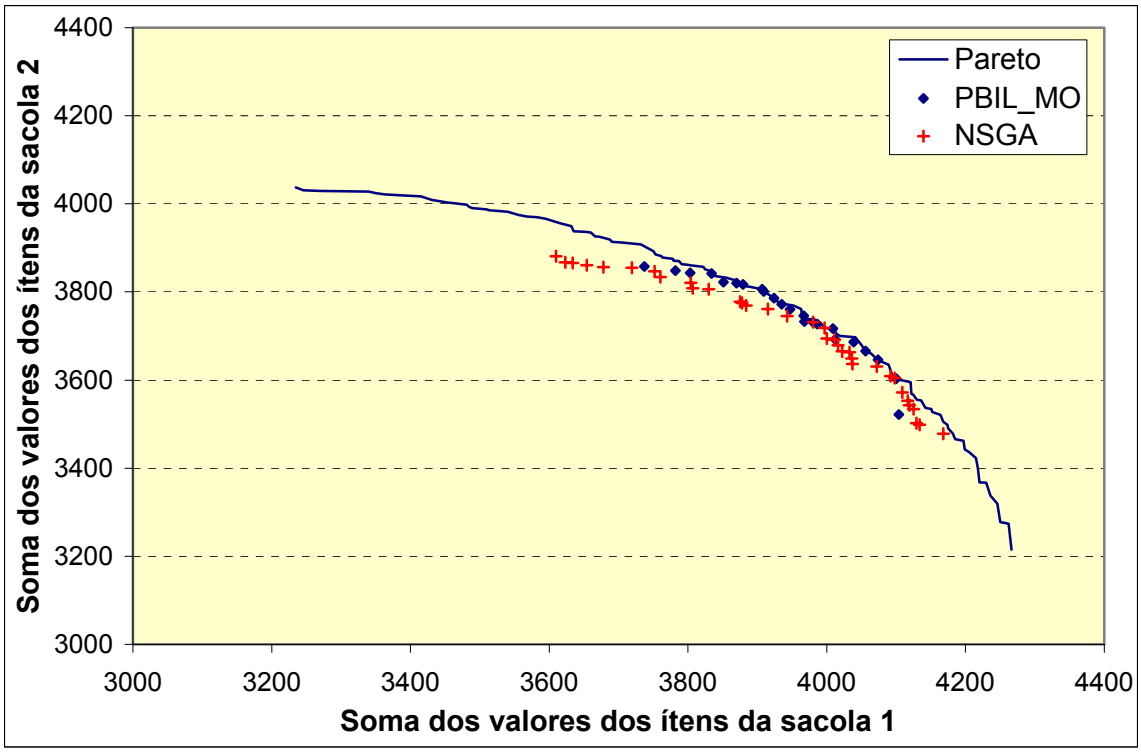


Figura 4.14 – Superfície do PBIL_MO + superfície do NSGA para o Sac_2/100

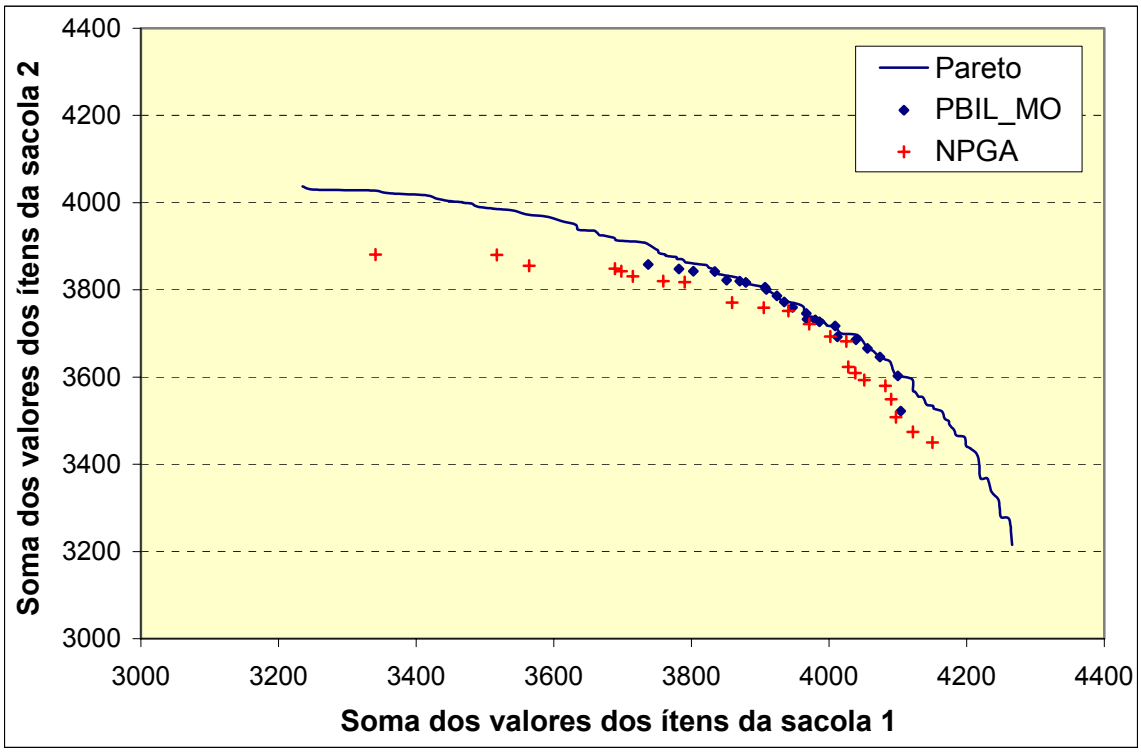


Figura 4.15 – Superfície do PBIL_MO + superfície do NPGA para o Sac_2/100

Das Figuras 4.14 e 4.15 podemos notar que o algoritmo PBIL_MO apresenta resultados mais precisos, ou seja, mais próximos a superfície de Pareto, que os apresentados pelo NSGA e o NPGA. A superfície gerada pelo PBIL_MO, além de ser melhor que a gerada pelos algoritmos genéticos, é obtida de forma mais rápida. Este fato se deve, principalmente, à característica do algoritmo PBIL de obter bons resultados empregando populações de tamanho pequeno quando comparadas com as dos algoritmos genéticos (MACHADO e SCHIRRU, 2000). Porém, o NSGA e o NPGA conseguem distribuir um pouco melhor seus pontos ao longo de toda a superfície de Pareto, resultado dos métodos de nicho utilizados por estes para manter a diversidade da população ao longo de todo o processo de busca. O grande problema da utilização dos métodos de nichos é que muitas vezes pode ser difícil, para um determinado problema, obter-se uma métrica para o agrupamento dos indivíduos, além do tempo computacional requerido por estes métodos.

Capítulo 5

O Problema da Recarga de Reatores Nucleares

5.1 - O problema da recarga

Os reatores nucleares a água pressurizada ou PWR (*Pressurized Water Reactor)*, como por exemplo os reatores das usinas nucleares Angra 1 (que será utilizado neste estudo) e Angra 2, utilizam como combustível, para a geração de energia elétrica, urânio enriquecido ($^{235}U_{92}$). O urânio enriquecido se encontra na forma de pastilhas combustíveis dispostas dentro de uma vareta chamada de vareta combustível. Um determinado número de varetas, de acordo com o reator, mais um conjunto de estruturas (bocais, grades espaçadoras, tubos guia de barras de controle e tubo de instrumentação) irão formar os elementos combustíveis do reator. No núcleo do reator da usina Angra 1 existem 121 elementos combustíveis (Figura 5.1), que possuem 235 varetas combustíveis cada um.

Uma característica importante do núcleo da usina Angra 1 é a existência de eixos de simetria (Figura 5.1) que podem ser divididos em dois tipos:

- Eixos de simetria principal – são os dois eixos que cortam o núcleo no sentido Norte-Sul e Leste-Oeste e dividem o núcleo do reator em quatro regiões chamadas quadrantes.
- Eixos de simetria secundário ou diagonal – são os dois eixos que cortam os quadrantes do núcleo em suas diagonais e junto com os eixos principais dividem o núcleo do reator em oito regiões, as quais iremos chamar octantes.

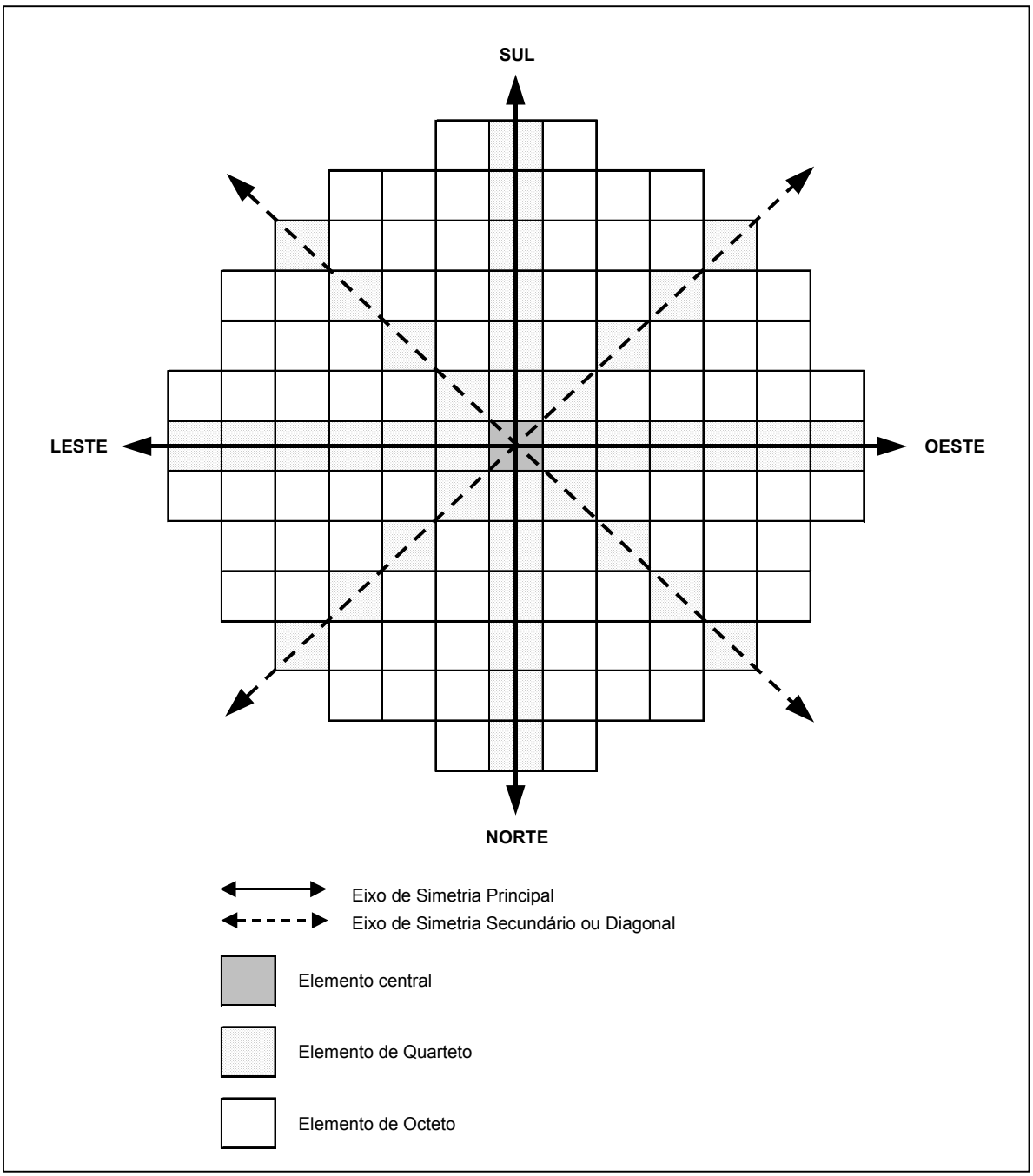


Figura 5.1 – Representação do núcleo de Angra 1

Utilizando-se os eixos de simetria do núcleo, pode-se classificar os 121 elementos combustíveis que formam o núcleo de Angra 1 em três grupos distintos:

- Elemento central – é o elemento combustível localizado no ponto onde todos os eixos de simetria se cruzam (ponto central do núcleo);

- Elementos de quarteto – conjunto de 4 elementos combustíveis localizados em posições do núcleo por onde passam os eixos de simetria. Para Angra 1 existem 10 quartetos, que formam um total de 40 elementos combustíveis.
- Elementos de octeto – conjunto de 8 elementos combustíveis localizados em posições do núcleo por onde não passam os eixos de simetria. Para Angra 1 existem 10 octetos, que formam um total de 80 elementos combustíveis.

A medida que o tempo passa, a concentração de material físsil ($^{235}U_{92}$) existente nos elementos combustíveis diminui até um valor para o qual não é mais possível se manter o reator operando produzindo energia à potência nominal. Este período de tempo que o reator opera com um determinado conjunto de elementos combustíveis é chamado de ciclo de operação.

Ao final de um ciclo de operação o reator da usina é desligado e todos os elementos combustíveis presentes no núcleo são descarregados e colocados em uma piscina de combustíveis usados (PCU). Deste conjunto de elementos descarregados, alguns não irão retornar ao reator (elementos mais queimados – ficarão em definitivo na PCU) e serão substituídos por elementos combustíveis novos.

A maioria dos reatores do tipo PWR utiliza um esquema de carregamento onde 1/3 dos elementos combustíveis são trocados a cada recarga. Os elementos combustíveis remanescentes, tendo estado em diferentes posições do núcleo por, geralmente, um ou dois ciclos de operação, apresentam características neutrônicas diferentes entre si.

O conjunto de elementos novos mais os elementos combustíveis descarregados do núcleo e que não foram descartados formarão o conjunto de elementos a serem utilizados no processo de recarga do reator.

De posse destes elementos combustíveis surge o problema da recarga, que consiste em determinar o arranjo entre os elementos combustíveis parcialmente queimados e os elementos combustíveis novos no núcleo do reator, ou seja um padrão de carregamento, de forma a se otimizar o próximo ciclo de operação do reator, garantindo que restrições operacionais e de segurança sejam respeitadas.

Para se especificar um padrão de carregamento é necessário se determinar para cada posição do núcleo (CARTER, 1997):

1º - o elemento combustível que ocupa a posição;

2º - a rotação do elemento combustível (se ele não for novo);

3º - o número de venenos queimável contidos no elemento (se este possuir).

O objetivo da recarga é minimizar o custo da energia gerada pela usina nuclear. Entretanto, como a avaliação deste custo envolve vários fatores complexos (operacionais e não operacionais) não existe uma formulação simples que possa ser utilizada na prática e em geral costuma-se trabalhar com funções objetivo que estão relacionadas ao custo tais como:

- maximizar a reatividade no final do ciclo;
- maximizar a queima dos elementos combustíveis;

- minimizar o número de elementos combustíveis novos a serem utilizados;

com restrições que incluem:

- coeficiente de temperatura máximo do moderador;
- queima máxima do elemento combustível;
- fator de pico máximo;
- minimização dos danos de radiação ao vaso do reator.

Para um reator do tipo PWR contendo 193 elementos combustíveis (com exemplo Angra 2), onde o elemento central é fixo e supondo que 64 (1/3 x 192 = 64) elementos novos, com características idênticas, são carregados, existem $\mathbf{128!\ {}^{192}C_{64}}$ possíveis formas de carregamento do núcleo (CARTER, 1997), o que corresponde a um número aproximado de 10^{267} soluções. Isto significa que existem $\mathbf{{}^{192}C_{64}}$ formas de se escolher onde colocar os elementos combustíveis novos e 128! formas de se arranjar os elementos combustíveis restantes.

A existência dos eixos de simetria é de grande importância na solução do problema da recarga, uma vez que o uso destas simetrias reduz consideravelmente o número de combinações do problema. Desta forma podemos trabalhar somente com um dos quadrantes do núcleo (simetria de 1/4) ou com um dos octantes do núcleo (simetria de 1/8) e depois utilizando as regras de rebatimento do núcleo montarmos a configuração completa do mesmo. Nestes casos, o número de possíveis soluções se encontra na ordem de 10^{52} (para 1/4 de núcleo) e 10^{67} (para 1/8 de núcleo), valores que ainda assim tornam impossível que todos estes padrões de carregamento sejam avaliados para a

escolha do padrão que será efetivamente utilizado no próximo ciclo de operação da usina.

Reatores nucleares possuem características altamente não lineares (POON e PARKS, 1992). Isto significa que toda função objetivo e restrições utilizadas para definir e quantificar os padrões de carregamento gerados possuirão, inevitavelmente, características não lineares. Uma conseqüência deste fato é a criação de numerosos pontos de mínimo (ou máximo) locais. Galperin (GALPERIN, 1995) em seu trabalho, mostra a existência de um número incrivelmente grande de ótimos locais, cerca de um ótimo para cada cem configurações. A implicação direta desta descoberta é que para um espaço de busca contendo 10^{67} combinações existem aproximadamente 10^{65} pontos de ótimos locais. Este fato, torna indispensável que qualquer técnica que venha a ser empregada na solução do problema da recarga tenha bom desempenho de busca em espaços multi-modais.

Para a operação segura de uma usina nuclear é necessário que um padrão de carregamento seja minuciosamente examinado utilizando-se códigos computacionais extremamente detalhados e, conseqüentemente, demorados. A utilização direta destes códigos computacionais em um processo de otimização da recarga torna o processo bastante lento. Como alternativa a este problema, alguns pesquisadores utilizam códigos mais simplificados para a avaliação dos padrões de carregamento.

Outro fator que dificulta a solução do problema da recarga é a necessidade que alguns algoritmos de otimização tradicionais têm de utilizar informações sobre derivadas, que

neste caso, ou não estão disponíveis ou não são sempre confiáveis em face da não linearidade das funções envolvidas.

A combinação destes atributos:

- alto número de combinações;
- objetivos e restrições não lineares;
- multi-modalidade;
- elevado custo computacional;
- falta de informações sobre derivadas

descrevem o formidável problema da recarga, que vem desafiando os métodos de otimização tradicionais e estimulando os pesquisadores a desenvolverem/aplicarem cada vez mais métodos de otimização "inteligentes" na solução deste problema. Algumas destas aplicações são apresentadas no item a seguir.

5.2 - Algoritmos evolucionários aplicados à recarga

Como podemos observar, o problema da recarga apresenta grandes dificuldades para a utilização dos métodos tradicionais de otimização. Por isso ao longo das ultimas décadas cada vez mais pesquisadores do campo da inteligência artificial vêm buscando formas de se empregar os algoritmos evolucionários em sua solução.

No passado, a aplicação de métodos estocásticos para a otimização de problemas, especialmente os combinatórios foi o centro das atenções de uma considerável parte dos pesquisadores. Estas técnicas provaram ser capazes de encontrar boas aproximações das

soluções exatas de vários problemas, de forma mais eficiente que os métodos tradicionais de otimização. Uma das principais ferramentas para este trabalho foi o SA (*Simulated Annealing*) (KIRKPATRICK, GELATT e VECCHI, 1983).

A maior vantagem do SA sobre os métodos tradicionais de otimização é sua habilidade de não ser capturado por pontos de ótimo local. O algoritmo emprega uma busca randômica que não aceita somente mudanças que diminuam o valor da função objetivo (F) mas também aceita mudanças que permitam seu aumento, com uma probabilidade descrita por:

$$P = \exp\left(-\frac{\delta F}{T}\right) \qquad \textbf{(5.1)}$$

onde: δ F é a variação do aumento da função F

T é um parâmetro de controle definido pelo usuário

Quando considerado como uma ferramenta para a otimização do processo de recarga de reatores PWR o SA possui algumas características atrativas, tais como:

- é um método eficiente em problemas de dimensões elevadas;
- não necessita de informações sobre derivadas;
- é capaz de escapar de pontos de ótimo local.

Um código computacional empregando o SA acoplado a um modelo físico do reator baseado na teoria da perturbação, para a otimização da recarga, foi desenvolvido na universidade do estado da Carolina do Norte. Este código é chamado de FORMOSA

(Fuel Optimization for Reloads: Multiple Objectives by Simulated Annealing) (KROPACKZEK e TURINSKY, 1991).

No FORMOSA, durante o processo de otimização, os próximos candidatos a padrão de carregamento são gerados pela modificação de uma, duas ou três posições escolhidas aleatoriamente no padrão de carregamento atual. Se uma única posição é escolhida, a orientação e/ou o número de barras de veneno queimável do elemento combustível (se este for novo) são trocados aleatoriamente e o novo padrão de carregamento gerado é avaliado. Se duas ou três posições são selecionadas, uma troca binária (A → B / B → A) ou ternária (A → B / B → C / C → A) de elementos é realizada e novas orientações para os elementos combustíveis são selecionados de forma aleatória.

Os resultados apresentados por Kropackzek e Turinsky demostraram a viabilidade da utilização do SA na solução do problema da recarga, tanto que o programa FORMOSA passou a ser comercializado por seus criadores.

Poon e Parks (1992) propuseram a substituição do Simulated Annealing pelo Algoritmo Genético (GA) e desenvolveram um novo algoritmo para a solução do problema da recarga chamado FORMOGA.

Para a representação dos indivíduos presentes na população, o FORMOGA utiliza três cromossomos. O primeiro composto por números que identificam o elemento combustível, o segundo identifica o número de barras de veneno queimável em cada elemento e o terceiro indica a rotação dos mesmos. Esta representação foi usada em uma modelagem de 1/4 de núcleo e com uma função objetivo que minimizava o valor

de F_{XY} (Fator de Pico que é definido como a razão do pico de densidade de potência pela densidade média de potência no plano horizontal onde ocorreu o pico local de potência)

O FORMOGA utiliza um processo de Rank para a seleção dos indivíduos que irão participar do processo de recombinação. Como Poon e Parks não utilizam uma codificação binária para a representação dos indivíduos da população do algoritmo genético do FORMOGA, é necessário a utilização de operadores especiais de recombinação (alguns desses operadores são apresentados no Anexo D). O FORMOGA utiliza um dos seguintes operadores:

- CX (*Cycle Crossover*);
- PMX (*Partially Mapped Crossover*);
- HCMX (*Heuristic Copy & Match Crossover*).

Estes operadores de recombinação atuam somente sobre o primeiro cromossomo e os demais dados referentes a cada elemento combustível existente nos dois outros cromossomos são trocados de forma idêntica à realizada no primeiro. Existe ainda um operador de mutação que opera sobre os dois outros cromossomos promovendo alterações de forma aleatória dentro dos valores permitidos para cada parâmetro.

Poon e Parks concluíram que embora fossem necessário a continuidade da pesquisa, a substituição do SA pelos algoritmos genéticos representava um ganho, principalmente no caso da utilização de um processamento paralelo para a avaliação dos indivíduos, o que levaria a redução do tempo gasto no processo de otimização.

Em 1995 DeChaine e Feltus (1995) desenvolveram um sistema usando uma abordagem modular chamado CIGARO (*Code Independent Algoritms Reactor Optimization System*), cuja parte central do código é formado pelo padrão de carregamento e a estrutura do genótipo, que transforma as informações deste padrão em uma cadeia de bits que pode ser trabalhada pelo algoritmo genético empregado.

Na codificação do CIGARO, o genótipo utiliza uma única cadeia de bits, onde cada posição de carregamento tem um conjunto de bits que determina qual dos elementos combustíveis deve ser colocado nesta posição. Estes bits ao serem decodificados correspondem ao valor de K_∞ que deve ser colocado nesta posição. No caso em que o valor de K_∞ especificado não está disponível, o valor mais próximo disponível é utilizado.

Neste caso, o algoritmo genético utiliza para o processo de seleção dos indivíduos uma roleta proporcional ao valor da fitness, uma seleção por torneio ou um método de Rank. O operador de recombinação utiliza o método de cruzamento de dois pontos.

O CIGARO acoplado ao sistema de cálculos neutrônicos CASMO-3 / SIMULATE-3 foi utilizado na otimização da recarga de um reator Westinghouse de 157 elementos combustíveis. Foi utilizada uma modelagem de 1/8 de núcleo com uma função objetivo que visava a maximização do fator de multiplicação efetivo (K_{eff}) de início de ciclo.

Chapot (2000), Machado (2001) e Lima, Machado e Schirru (2002) utilizaram o algoritmo genético, Sistema de Colônia de Formigas e o algoritmo PBIL_N respectivamente, acoplados ao código de cálculos neutrônicos RECNOD (CHAPOT,

2000) visando maximizar a concentração de boro no início do ciclo. Em todos os estudos realizados, os autores concluíram que a utilização dos algoritmos evolucionários são ferramentas computacionais viáveis para estudos de otimização de recargas de forma automática.

Quando fazemos uma revisão dos trabalhos apresentados visando a otimização do problema da recarga, podemos destacar um ponto bastante interessante, qual seja, a grande diversidade de funções objetivo que são utilizadas. Como exemplo de algumas delas, podemos citar (PARKS, 1996):

- a maximização da reatividade no final do ciclo;
- a maximização da queima de descarga dos elementos combustíveis;
- minimização do fator de pico;
- minimização do enriquecimento inicial;
- minimização do número de elementos combustíveis novos;
- minimização do número de barras de veneno queimável.

De certa maneira, é obvio que se pudéssemos encontrar uma solução que otimizasse todos esses objetivos ao mesmo tempo, esta poderia ser considerada como uma solução ideal para o problema da recarga. Entretanto, como em outros problemas reais da engenharia, também no problema da recarga a otimização de um determinado objetivo só é alcançado em detrimento dos outros objetivos.

Parks (1996) em seu trabalho propõe a utilização de um algoritmo genético Multi_Objetivo (MOGA) para a solução do problema da recarga. Como objetivos da recarga são utilizados a minimização do enriquecimento, a maximização da queima de

descarga dos elementos combustíveis e a minimização do fator de pico de potência radial.

Parks (1996) utiliza a mesma codificação e forma de avaliação dos indivíduos empregada no FORMOSA. Em seu processo de seleção dos indivíduos do algoritmo genético, Parks utiliza o conceito de ótimo de Pareto e dominância de forma a determinar um valor de Rank para os mesmos sem utilizar qualquer informação da importância relativa de cada objetivo e um arquivo para armazenar os indivíduos não dominados ao longo do processo de busca. Ele conclui que a facilidade que o algoritmo genético Multi_Objetivo possui para trabalhar com várias combinações de objetivos e de ser acoplado a diferentes códigos de física de reatores transformam o MOGA em uma ferramenta viável para a otimização do problema da recarga.

Capítulo 6

O PBIL_MO aplicado ao Problema da Recarga

6.1 – O Caso de Estudo

Para a aplicação do algoritmo PBIL_MO ao problema da recarga foi selecionada a recarga do ciclo 7 da usina nuclear Angra 1 em um esquema de carregamento do núcleo do tipo baixa-fuga, onde deseja-se que elementos combustíveis novos sejam colocados em posições mais internas do núcleo e, por conseqüência, os elementos já irradiados sejam colocados em posições mais externas do núcleo. Para este ciclo de operação da usina nuclear Angra 1 não foram utilizados venenos queimáveis, que são utilizados para manter os valores de F_{xy} abaixo do limite de especificação técnica da usina.

Neste estudo foi utilizado o programa computacional de cálculos neutrônicos RECNOD (CHAPOT, 2000) modelado para uma simetria de 1/8 de núcleo, a duas dimensões, sendo os cálculos executados em duas etapas de queima: início de vida e equilíbrio de xenônio. Uma vez que o RECNOD não calcula o fator de pico de potência radial (F_{xy}) este parâmetro foi substituído pela máxima potência média relativa dos elementos combustíveis (P_{rm}).

Os objetivos escolhidos para a otimização do PBIL_MO foram definidos de acordo com os dados de saída fornecidos pelo RECNOD, quais sejam: a concentração crítica de boro e a potência média relativa. Para a concentração crítica de boro, quanto maior o seu valor ao final do ciclo, maior será o comprimento do ciclo de operação da usina. Desta forma, o primeiro objetivo do problema da recarga foi definido como sendo a maximização da concentração crítica de boro. A otimização do carregamento do núcleo

por uma estratégia de baixa-fuga possibilita esta maximização. Entretanto, como no ciclo 7 não são utilizados venenos queimáveis, existe a possibilidade da ocorrência de elevados picos de potência no núcleo que ultrapassem o limite de especificação técnica. Assim, o segundo objetivo a ser empregado no algoritmo PBIL_MO foi a minimização da máxima potência média relativa dos elementos combustíveis.

Em resumo o PBIL_MO utilizará dois objetivos para a solução do problema da recarga do ciclo 7 da usina Angra 1:

1º - Maximização da concentração crítica de boro

2º - Minimização da máxima potência média relativa dos elementos combustíveis.

6.2 – O Sistema PBIL_MO – RECNOD

O sistema PBIL_MO – RECNOD é o ambiente computacional desenvolvido neste trabalho para realizar o estudo de otimização Multi_Objetivo para o problema da recarga utilizando-se o algoritmo PBIL_MO e o código RECNOD. O sistema é formado pelo algoritmo de otimização PBIL_MO, uma interface de comunicação, um banco de configurações e o código RECNOD (Figura 6.1).

A interface de comunicação possui a função de receber os padrões de carregamento gerado pelo algoritmo PBIL_MO verificar, através de uma consulta ao banco de configurações, se estes padrões já foram avaliados pelo código de física de reatores (função consulta banco). Caso o indivíduo já tenha sido avaliado os resultados são obtidos do banco de configurações e não é necessário se avaliar novamente este

indivíduo com o código de física de reatores. Caso o indivíduo não tenha sido avaliado, a interface monta um arquivo de entrada para o código de física de reatores, executa uma rodada do código e obtém os resultados (função executa código).

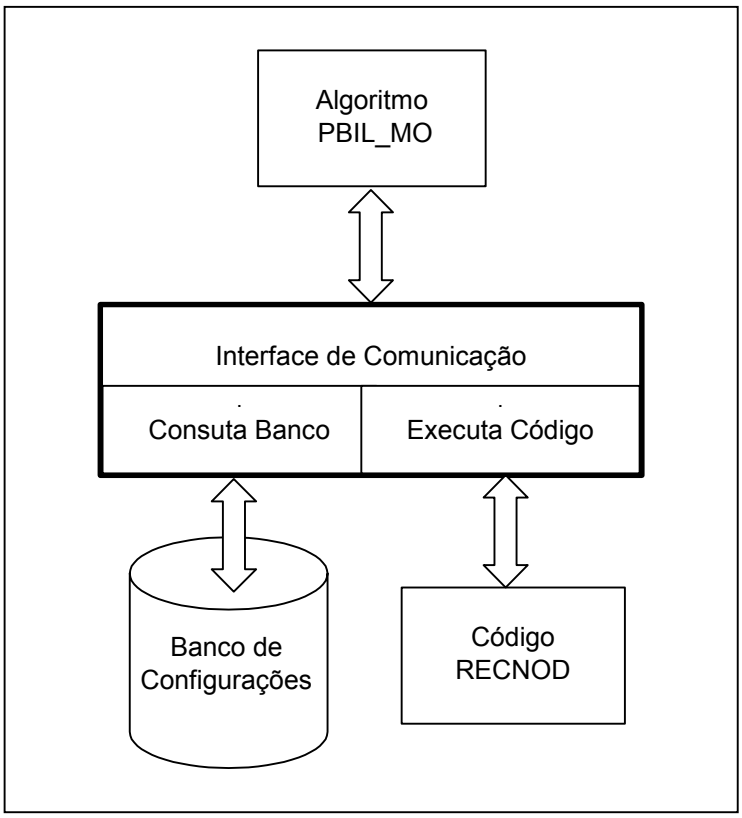


Figura 6.1 – Sistema PBIL_MO - RECNOD

O banco de configurações é responsável por armazenar os indivíduos que já foram avaliados pelo código de física de reatores. Isso é de grande importância, no sentido de se reduzir o tempo computacional gasto na avaliação dos indivíduos gerados. Dependendo do códigos de física de reatores empregado e o número de etapas de queima a serem realizadas, o tempo gasto para a avaliação pode chegar a dois minutos por padrão de carregamento. Caso este padrão já tenha sido avaliado, os valores obtidos do código de física de reatores estarão armazenados no banco e podem ser recuperados através de uma busca que dura poucos segundos.

Como exemplo, para um algoritmo empregando uma população de 30 indivíduos o tempo gasto para cada geração do algoritmo seria de uma hora. Se forem utilizadas 504 gerações no processo de otimização o tempo total gasto no processo seria de 504 horas ou 21 dias. Supondo que 15 % dos indivíduos sejam repetidos ao longo do processo, com a utilização do banco de configurações, tem-se uma redução de 3,15 dias para a realização do processo de otimização.

Outra função da interface de comunicação é transformar os indivíduos gerados pelo algoritmo PBIL_MO, que não estão no banco de configurações, em um arquivo de entrada para o código de física de reatores, executar o código de física de reatores e ao final da execução extrair os resultados obtidos. A seguir, o padrão de carregamento e os resultados obtidos são armazenados no banco de configurações.

A construção modular do sistema PBIL_MO – RECNOD permite que qualquer um de seus componentes seja substituído com facilidade. Para a substituição do banco de configurações e/ou sua estrutura basta alterar a função de busca e para o código de físicas de reatores a função executa código na interface de comunicação.

Para a solução do problema da recarga pelo algoritmo PBIL_MO foi definida uma modelagem genética e um processo de decodificação dos indivíduos. Estes itens são apresentados nos tópicos seguintes.

6.2.1 – A modelagem genética

A modelagem genética é a forma com que um indivíduo da população do algoritmo PBIL_MO representa um padrão de carregamento para o núcleo do reator, ou seja, é a definição da estrutura que codifica um candidato a solução do problema.

No algoritmo PBIL_MO cada indivíduo é formado por uma cadeia binária (cromossomo) e cada elemento combustível é representado por um gene neste cromossomo. Cada gene do cromossomo do PBIL_MO utiliza cinco bits para sua representação. Com esta definição, para um processo de otimização contendo X elementos de quarteto e Y elementos de octeto, o comprimento total do cromossomo (K) será definido por :

$$K = 5 \times X + 5 \times Y \qquad \textbf{(6.1)}$$

Desta forma utilizando a simetria de 1/8 de núcleo para Angra 1 e definindo o elemento central como fixo, são utilizados 20 genes – 10 representando os elementos de quarteto e 10 os de octeto. O comprimento total do cromossomo é de 100 bits, sendo os 50 primeiros bits para representar os elementos de quarteto e os 50 restantes os elementos de octeto.

A associação entre os elementos combustíveis e os genes é feita pela Tabela_Elementos de acordo com os elementos disponíveis para o processo de otimização e a forma utilizada por cada código de física de reatores para identificar os elementos combustíveis. A Tabela_Elemento está localizada dentro da função “Executa Código”

da interface de comunicação e uma vez definida, esta tem que ser mantida fixa ao longo de todo o processo de otimização.

A Tabela 6.1 mostra a representação da Tabela_Elementos utilizada pelo RECNOD no ciclo 7 da usina Angra 1. No RECNOD cada elemento combustível é caracterizado pela posição em que estava no reator no ciclo anterior é o tipo de elemento combustível que identifica o elemento combustível na biblioteca de dados nucleares do RECNOD.

Tabela 6.1 – Representação da Tabela_Elementos do PBIL_MO para o Ciclo 7 de Angra 1

Identificação do Elemento Combustível	Número do gene	Posição no Ciclo Anterior	Tipo de Elemento Combustível
Central	X	(1, 1)	4
Quarteto	1	(2, 1)	6
Quarteto	2	(3, 1)	4
Quarteto	3	(4, 1)	4
Quarteto	4	(5, 1)	4
Quarteto	5	(6, 1)	2
Quarteto	6	(7, 1)	2
Quarteto	7	(2, 2)	5
Quarteto	8	(3, 3)	4
Quarteto	9	(4, 4)	1
Quarteto	10	(5, 5)	2
Octeto	11	(3, 2)	4
Octeto	12	(4, 2)	4
Octeto	13	(5, 2)	4
Octeto	14	(6, 2)	3
Octeto	15	(7, 2)	5
Octeto	16	(4, 3)	6
Octeto	17	(5, 3)	6
Octeto	18	(6, 3)	3
Octeto	19	(5, 4)	5
Octeto	20	(6, 4)	2

Uma decodificação direta dos cromossomos do PBIL_MO, na grande maioria das vezes, irá gerar indivíduos inválidos, ou seja, com a repetição de determinado elemento combustível o que determina que este elemento deve ser colocado em mais de uma posição do núcleo. Como exemplo, tendo-se um indivíduo formado por 4 genes com 2 bits/gene com a configuração apresentada na Figura 6.2 ao ser decodificado se torna inválido. Para a solução deste problema é utilizado um processo de decodificação, baseado no modelo Random Keys (BEAN, 1994) apresentado no próximo item.

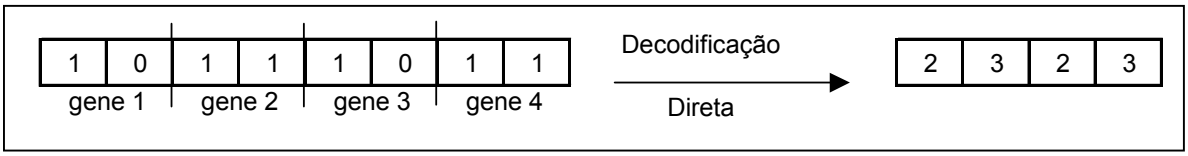


Figura 6.2 – Representação da decodificação direta dos cromossomos do PBIL_MO

6.2.2 – Processo de decodificação do PBIL_MO

Com a finalidade de se evitar que quase a totalidade dos indivíduos gerados pelo algoritmo PBIL_MO sejam inválidos, adotou-se o processo de decodificação baseado no Random Keys (BEAN, 1994). A opção por este modelo de decodificação foi baseado no trabalho anterior de Machado (1999) que obteve excelentes resultados na solução de problemas combinatórios.

Este processo de decodificação é bastante simples e garante que todos os indivíduos gerados por qualquer algoritmo evolucionário, utilizando uma codificação binária, serão sempre válidos.

No modelo Random Keys a cadeia de bits que forma o gene é decodificada em um valor inteiro e forma o vetor de inteiros V_I. O gene que possuir o menor valor no vetor V_I aparece primeiro no vetor de ordenação final O_F. No caso de valores iguais no vetor V_I o gene de menor posição é colocado primeiro no vetor O_F. A Figura 6.3 apresenta o funcionamento do modelo Random Keys.

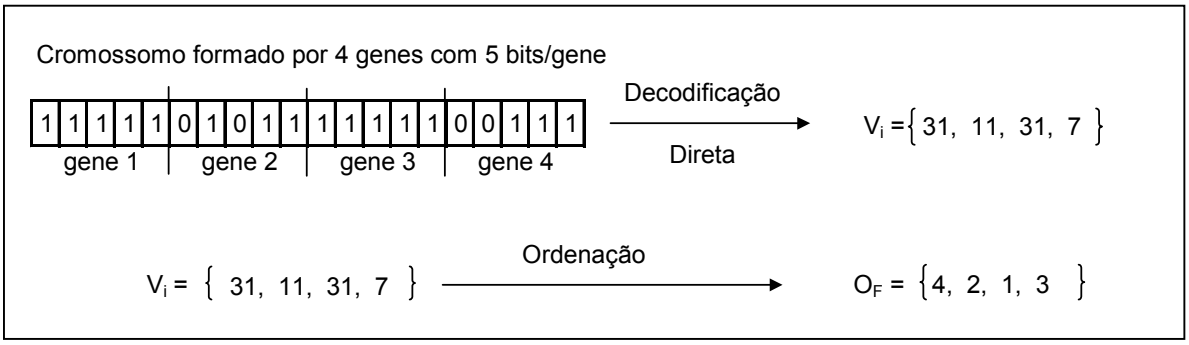


Figura 6.3 – Funcionamento do modelo Random Keys

6.2.3 – O código RECNOD

O programa computacional de cálculos neutrônicos RECNOD (CHAPOT, 2000) foi criado com o objetivo de substituir o código ANC (CHAPOT, 2000) em um sistema de otimização de recargas baseado na técnica dos algoritmos genéticos. Chapot teve como motivação para o desenvolvimento do RECNOD a busca por um código de física de reatores bidimensional a dois grupos de energia, mais rápido do que o ANC, de modo a reduzir os custos computacionais envolvidos no processo de otimização de reatores a água pressurizada, ou seja, buscava-se reduzir o tempo para a avaliação dos padrões de carregamento que eram gerados pelo algoritmo genético.

Os altos custos computacionais do método de diferenças finitas (MDF) utilizados para a resolução da equação de difusão de nêutrons bidimensional, para poucos grupos de energia, levaram à criação de uma metodologia de cálculo menos rigorosa, porém mais eficiente: os métodos nodais. Este métodos partem do pressuposto que o núcleo do reator possa ser decomposto em sub-regiões, relativamente grandes chamadas nodos. A técnica dos métodos nodais está calcada em duas hipóteses:

- Os parâmetros nucleares são supostos uniformes dentro de um nodo;
- Em cada nodo o fluxo pode ser representado por uma expansão em funções polinomiais.

O código RECNOD utiliza um método de expansão de fluxo alternativo designado como FEM-M cuja diferença fundamental com o FEM clássico, é que enquanto o FEM calcula fluxos pontuais (no centro e nos pontos médios dos lados do nodo), o FEM-M faz uso de fluxos médios (Figura 6.4).

Um código nodal de física de reatores de uso comercial para estudos de otimização de recarga de reatores, deve conter os seguintes módulos (CHAPOT, 2000):

- módulo de Fluxo-Potência-Reatividade;
- módulo de pesquisa de criticalidade com boro solúvel;
- módulo queima de combustível;
- módulo com os modelos de realimentação;
- módulo para reconstrução da distribuição de densidade de potência pino a pino;

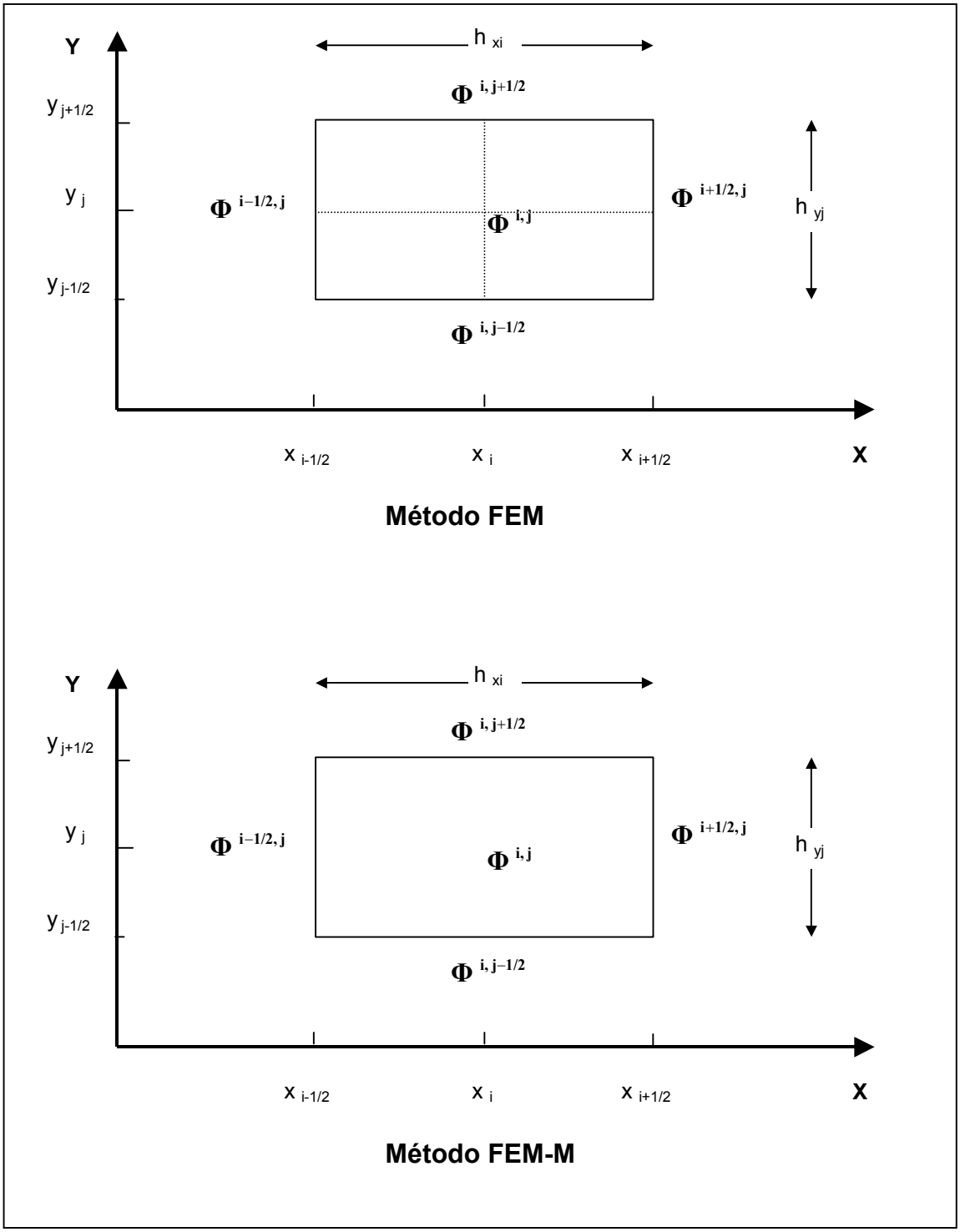


Figura 6.4 – Esquema de um Nodo dos Métodos FEM e FEM-M

Apesar do código RECNOD não poder ser utilizado de forma comercial, pois não possui os módulos como os modelos de realimentação e nem aqueles para a reconstrução da distribuição de potência pino a pino, Chapot (2000) demostrou que o

RECNOD foi capaz de realizar com eficiência estudos de otimização para o problema da recarga.

6.3 Resultados Obtidos

Para testar o sistema PBIL_MO – RECNOD foram utilizados como parâmetros de configuração os seguintes dados:

- Número de Gerações = 500;
- Tamanho da população = 20 indivíduos;
- Número de indivíduos para atualização do vetor de probabilidades = 2;
- Número de genes = 20;
- Número de bits por gene = 5;

e um conjunto de 5 taxas de aprendizado combinados com 5 valores de semente aleatória, perfazendo um total de 25 experimentos.

6.3.1 Caso 1

Neste caso, são utilizados os objetivos de maximização da concentração crítica de boro e a minimização da máxima potência média relativa, com o algoritmo PBIL_MO com a escolha dos indivíduos para atualização por uma roleta com arco igual para todos os indivíduos não dominados a cada geração.

Chapot (2000), em seu trabalho, determinou o limite de 1,395 com relação a máxima potência média relativa, para considerar um padrão de carregamento como válido, a fim de garantir que limites de especificações técnicas não fossem violados. Entretanto, neste primeiro caso de estudo não foi utilizado nenhum tipo de restrição com relação os

objetivos empregados, com a finalidade de se mapear toda a superfície de Pareto, uma vez que restrições e penalizações restringem o espaço de busca. A Figura 6.5 apresenta uma curva característica obtida pelo algoritmo PBIL_MO.

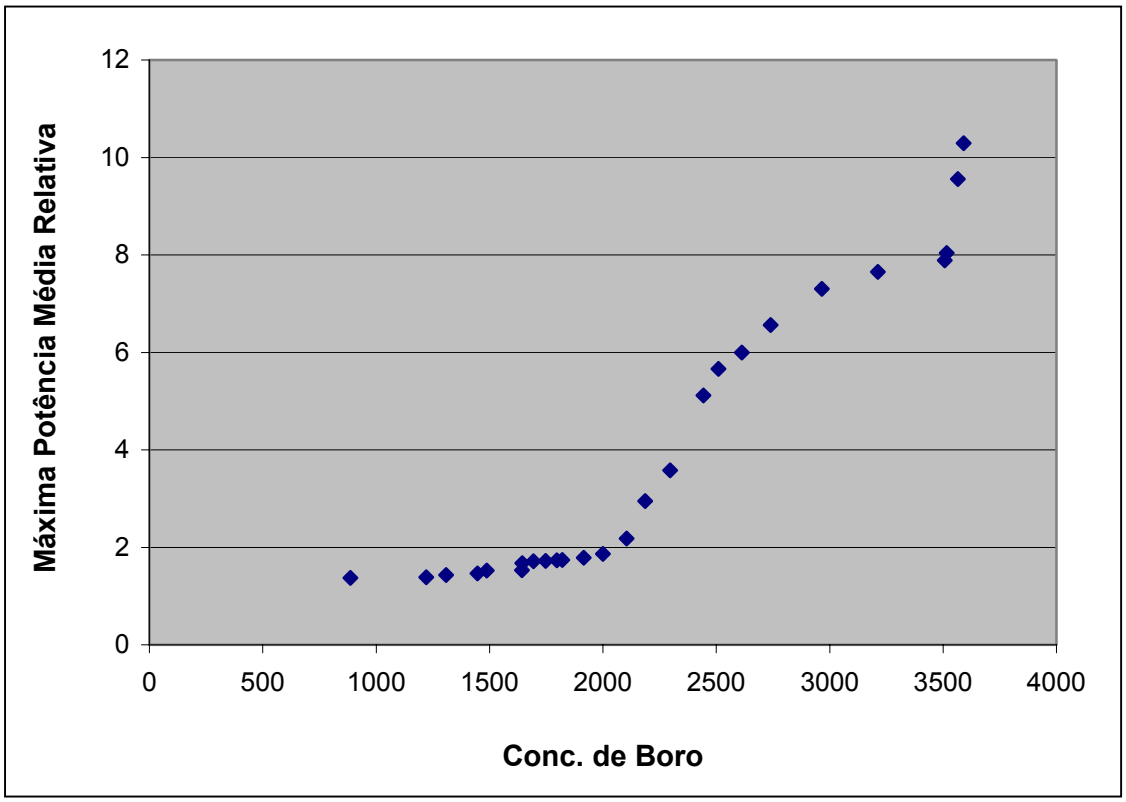


Figura 6.5 – Superfície de Pareto Caso 1 - Tpop = 20 e 500 gerações

A fim de se verificar o comportamento do algoritmo PBIL_MO, em relação ao número de gerações foi realizado, com o mesmo conjunto de taxas de aprendizado e sementes novas rodadas empregando-se 20 indivíduos na população, porém com o aumento do número de gerações para 1000.

A Figura 6.6 apresenta uma curva característica obtida para estes testes, onde podemos observar um aumento do número de indivíduos na superfície bem como a melhor

distribuição destes indivíduos ao longo da superfície de Pareto. Entretanto, cabe destacar que esta melhoria no resultado só é obtida com o aumento considerável do tempo computacional.

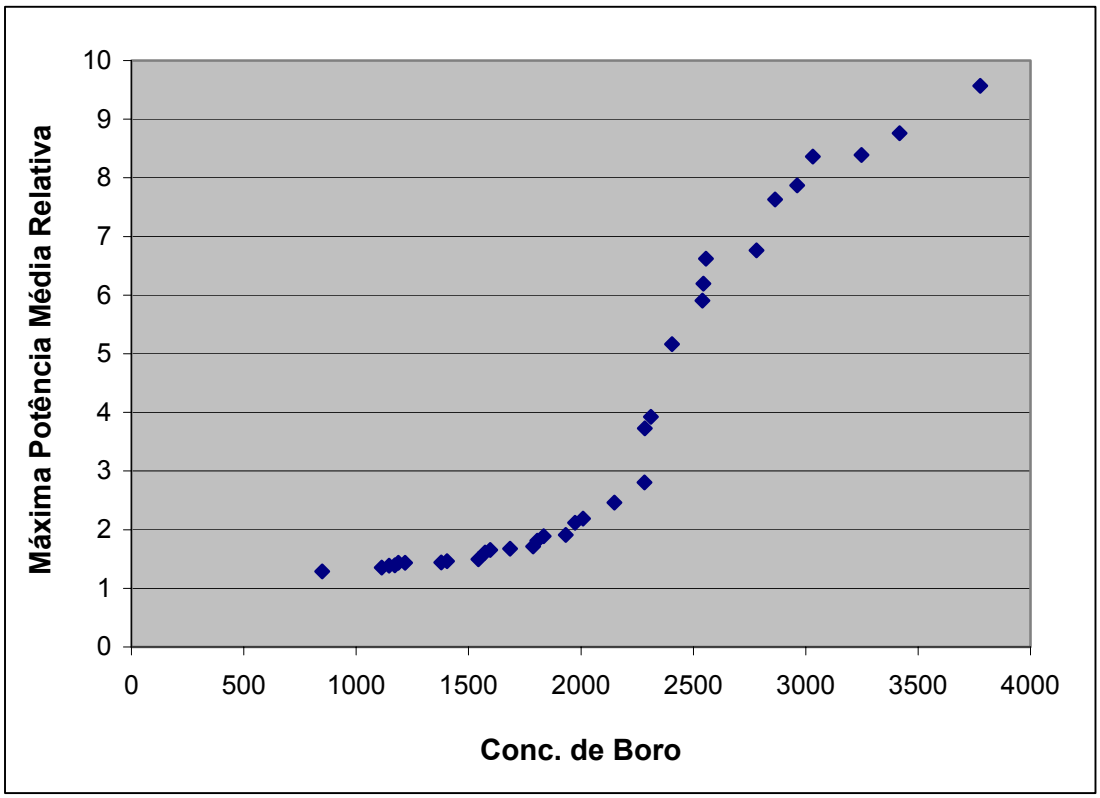


Figura 6.6 – Superfície de Pareto Caso 1 - Tpop = 20 e 1000 gerações

Um terceiro teste empregando-se o mesmo esquema de escolha dos indivíduos para atualização do vetor de probabilidades, foi realizado com o objetivo de analisar o comportamento do algoritmo PBIL_MO em função do aumento do tamanho da população. Neste caso, para se realizar a comparação, utiliza-se o mesmo número de avaliações da fitness para os dois testes, onde o número de avaliações corresponde ao produto do tamanho da população pelo número de gerações realizadas. Neste caso, no primeiro teste com 20 indivíduos ao longo de 500 gerações temos 10000 avaliações, o

que corresponde a realização de 200 gerações para a população de 50 indivíduos que será utilizada no segundo teste.

Neste terceiro teste também obteve-se um aumento do número de indivíduos que passaram a formar a superfície de Pareto, bem como um melhor mapeamento da mesma. Isto demonstra que o aumento do tamanho da população permite uma melhor amostragem do espaço de busca e por conseqüência uma busca mais efetiva do mesmo. A Figura 6.7 apresenta uma curva característica obtida neste terceiro teste.

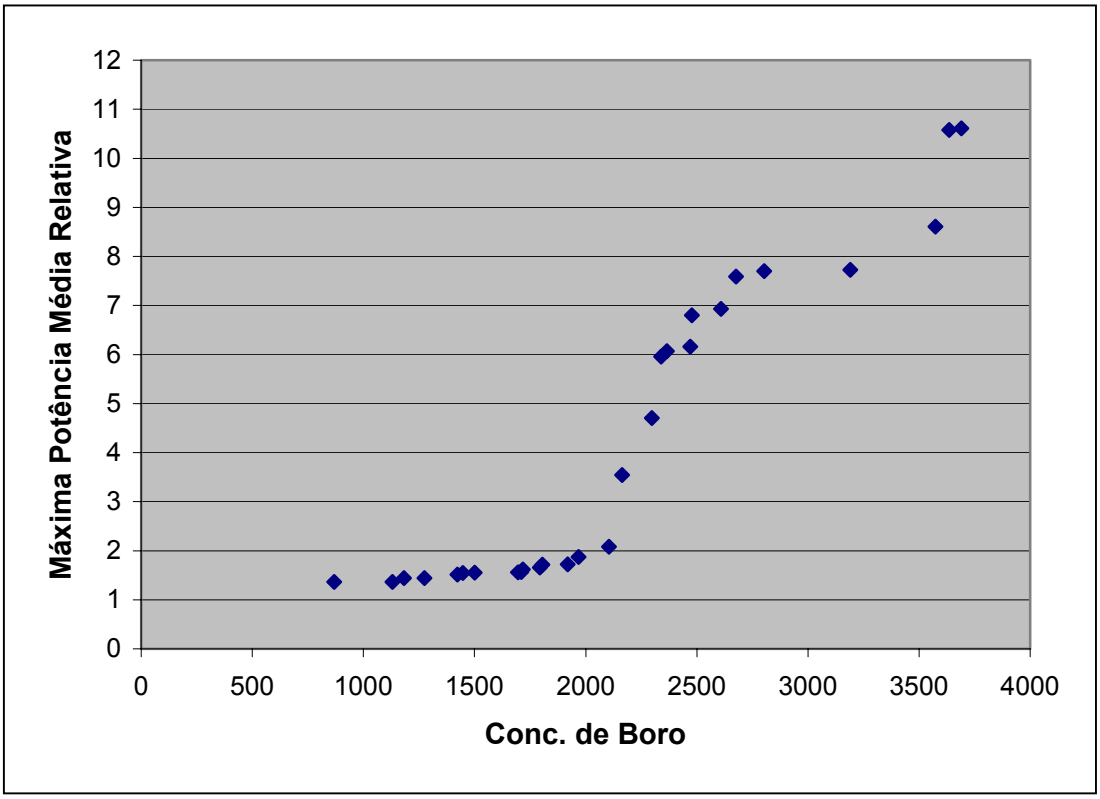


Figura 6.7 – Superfície de Pareto Caso 1 - Tpop = 50 e 200 gerações

Da observação das Figuras 6.6 e 6.7 podemos observar que os resultados obtidos com o aumento do tamanho da população são bastante próximos aos obtidos com o aumento

do número de gerações, com a vantagem de não implicarem no aumento do tempo computacional .

6.3.2 Caso 2

A fim de verificar o comportamento do algoritmo PBIL_MO para o caso de problemas onde exista uma restrição, foi definido um novo objetivo para o problema da recarga com relação a máxima potência média relativa, qual seja, a minimização da potência a partir do valor limite de 1,395. Indivíduos com valores superiores a este valor limite deixam de ser considerados no critério de dominância. O segundo objetivo foi mantido como sendo a maximização da concentração crítica de boro.

Neste caso, temos a busca restringida a uma parte do espaço de busca e podemos observar que a retirada dos valores elevados de concentração crítica de boro promoveu a retirada dos valores elevados de máxima potência média relativa (Figura 6.8), demostrando o conflito existente entre o objetivo de se maximizar a concentração de boro e minimizar a potência.

A Tabela 6.2 apresenta uma comparação dos pontos obtidos neste experimento com o valor obtido por Chapot (2000) com o algoritmo genético e com o PBIL_N desenvolvido por Machado (1999). Como era esperado, com a superfície gerada pelo algoritmo PBIL_MO o especialista passa a dispor de várias configurações que podem ser utilizadas para o carregamento do núcleo, cabendo a ele decidir a que melhor atende o planejamento para o próximo ciclo de operação.

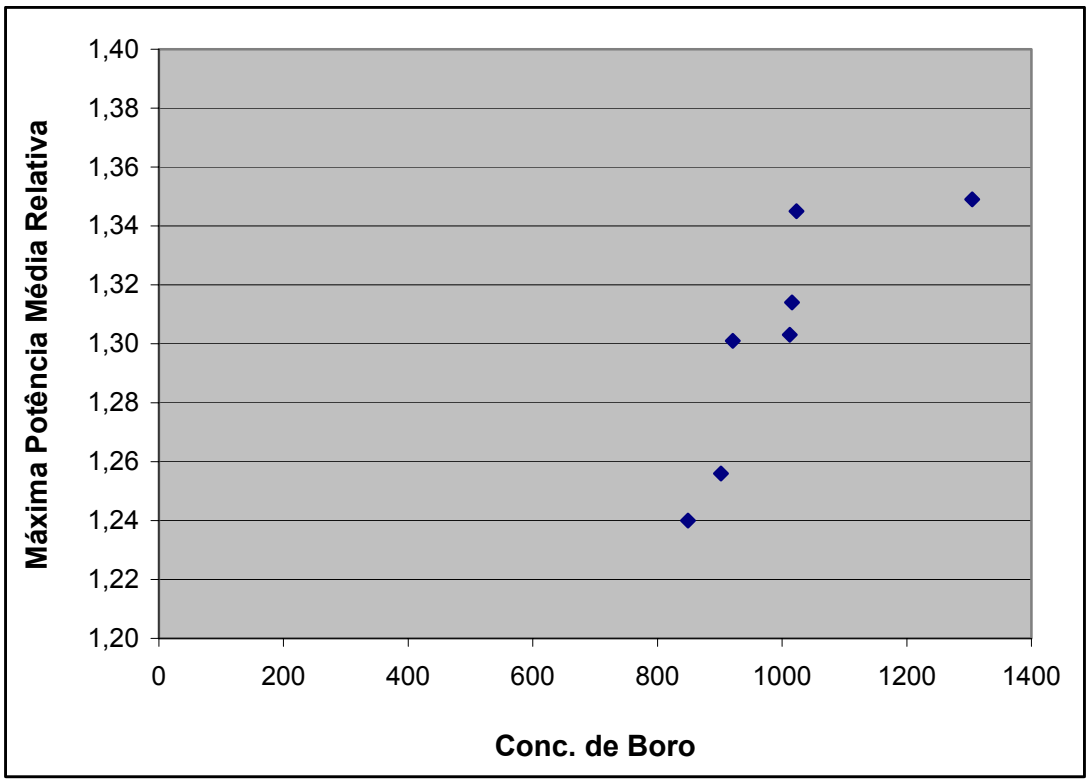


Figura 6.8 – Superfície de Pareto Caso 2 – Restrição de máxima potência

Tabela 6.2 – Comparação entre AG, PBIL_N e PBIL_MO para a Recarga

Algoritmo	Pontos (Conc. de Boro / Máx. Potência)	Avaliações
Algoritmo Genético	1026 / 1,390	4000
PBIL_N	1242 / 1,361	6000
PBIL_MO	849 / 1,24 902 / 1,256 921 / 1,301 1012 / 1,303 1016 / 1,314 1023 / 1,345 1305 / 1,349	10000

6.3.3 Caso 3

Neste caso foram testadas duas novas formas de escolha dos indivíduos utilizados para a atualização do vetor de probabilidades do PBIL_MO. Nos dois casos deseja-se privilegiar os indivíduos não dominados que melhor atendam aos objetivos definidos. Ou seja, dentre os indivíduos não dominados os que possuem um maior valor de concentração crítica de boro e a menor máxima potência média relativa. Na primeira abordagem foi utilizada uma roleta proporcional ao valor de cada objetivo – indivíduos melhores possuem arcos maiores na roleta (Figura 3.2) - e na segunda abordagem foi escolhido diretamente dentre os indivíduos não dominados o melhor para cada um dos objetivos.

Para o teste, destas duas novas formas de seleção dos indivíduos, também foram realizados 25 experimentos, com tamanho da população igual a 20 e número de gerações igual a 500.

Pare estes casos foi observado que o algoritmo PBIL_MO promoveu um aumento do número de indivíduos nas extremidades da superfície de Pareto, principalmente para o segundo processo de seleção, mantendo um mapeamento razoável da mesma, ou seja, os indivíduos continuaram sendo distribuídos ao longo de toda a superfície. As Figuras 6.9 e 6.10 apresentam as curvas características obtidas neste caso de estudo.

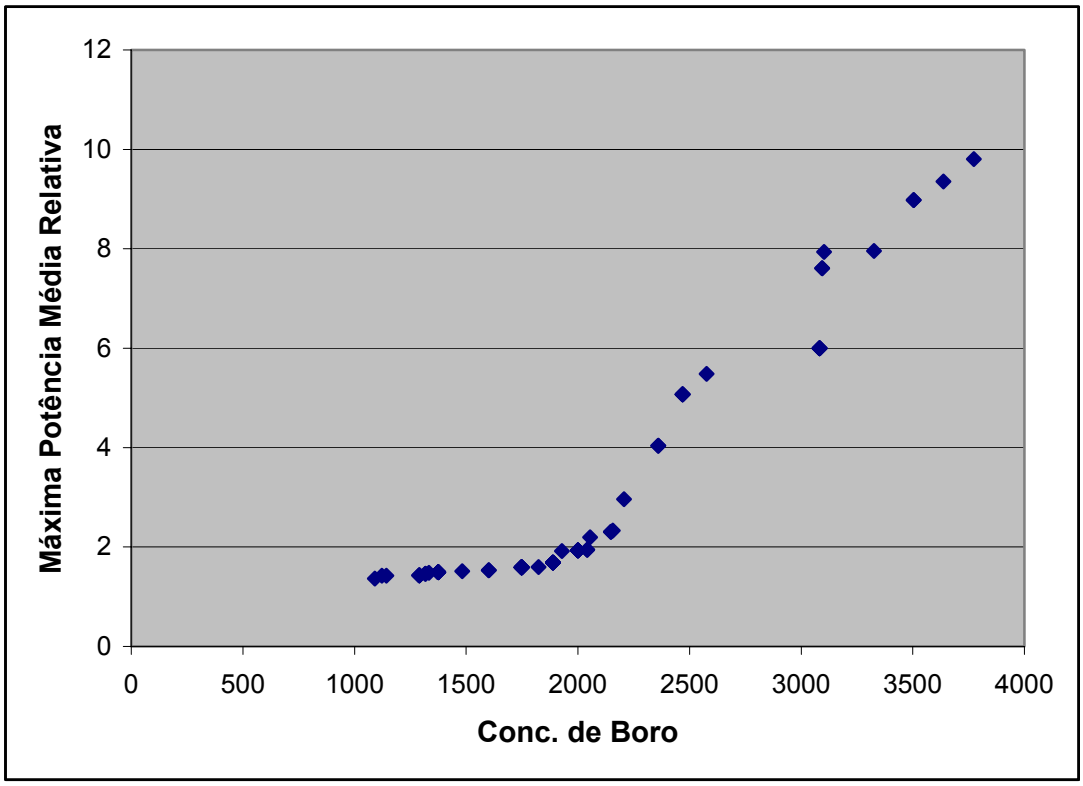


Figura 6.9 – Superfície de Pareto Caso 3 – Roleta Proporcional

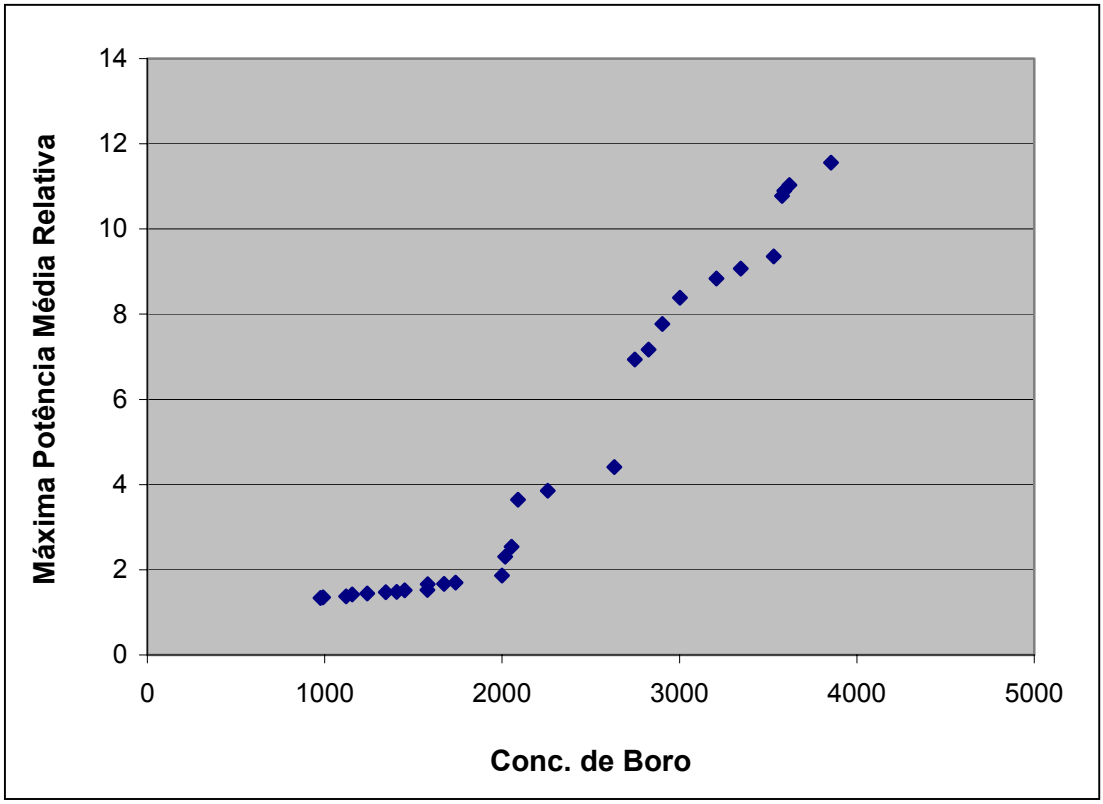


Figura 6.10 – Superfície de Pareto Caso 3 – Seleção dos Melhores

Capítulo 7

Conclusões e Propostas de Trabalhos Futuros

Quando buscamos a criação de um sistema "inteligente", o que se pretende é auxiliar na obtenção de soluções de problemas complexos no campo da engenharia. Problemas cujas soluções demandam esforço considerável por parte dos pesquisadores e engenheiros para serem encontradas. O que se deseja não é substituir o especialista humano, mas sim fornecer uma ferramenta onde seu conhecimento acerca do problema seja usado para se obter a solução, mas sem que este conhecimento seja um fator limitante para a obtenção desta solução.

Os excelentes resultados obtidos pelo algoritmo PBIL_MO com as funções de teste para problemas Multi_Objetivo, tanto nas funções numéricas como no problema combinatório, vieram confirmar a viabilidade da utilização desta nova ferramenta de otimização. Dos resultados obtidos no problema combinatório, podemos notar que o algoritmo PBIL_MO sempre foi capaz de gerar pontos mais próximos da superfície de Pareto que os obtidos com os algoritmos genéticos Multi_Objetivo e com um tempo de processamento bem menor. Outra característica observada nestes testes, foi a boa capacidade do PBIL_MO de distribuir seus pontos ao longo de todas as superfícies de Pareto, mesmo sem a utilização de uma técnica de nicho, o que permite uma boa visualização destas superfícies.

No problema da recarga, existem padrões de carregamento que são sabidamente inviáveis de serem utilizados e que nem chegam a ser testados pelo especialista humano. Entretanto, pode existir em algum destes padrões informações que conduzam a

uma excelente solução. O surgimento, ao longo do processo de otimização, desses padrões de carregamento que na prática não podem ser utilizados para o carregamento do reator, serve como uma forma de demonstrar o potencial que o algoritmo PBIL_MO possui de explorar todo o espaço de busca, independentemente da complexidade do problema. Esta capacidade de exploração é de fundamental importância para o mapeamento de toda a superfície de Pareto em um determinado problema.

Para um algoritmo evolucionário, mantendo-se o tamanho da população fixo, é esperado que o aumento do número de gerações conduza a melhores resultados, uma vez que um maior número de pontos podem ser avaliados. Entretanto, este aumento do número de gerações implica um aumento do tempo computacional, pois mais indivíduos necessitam ser avaliados, o que pode inviabilizar, dependendo do tempo necessário para a avaliação dos indivíduos, este aumento de gerações. Uma alternativa a este problema é o aumento do tamanho da população, com a redução do número de gerações, a fim de se manter o mesmo número de avaliações de indivíduos. Os testes realizados com o algoritmo PBIL_MO no problema da recarga, demostraram que o aumento do número de gerações produz superfícies melhores, entretanto, para os casos em que se aumentou o tamanho da população com a redução do número de gerações foram geradas superfícies equivalentes as obtidas com o aumento do número de gerações, com a vantagem de se ter um esforço computacional menor.

Mesmo no caso em que o algoritmo PBIL_MO foi aplicado a um problema com restrição, este foi capaz de obter bons resultados. Neste caso parte do espaço de busca deixou de ser explorado devido à penalidade imposta aos indivíduos que não atendiam a restrição de máxima potência média relativa. Mesmo assim, o algoritmo PBIL_MO

obteve um indivíduo com maior concentração de boro e máxima potência média relativa menor (1305 / 1,349) que o obtido com o algoritmo genético (1039 / 1,384) e com o algoritmo PBIL_N (1242 / 1,361).

Uma característica importante do sistema PBIL_MO - RECNOD desenvolvido para a solução do problema da recarga de reatores PWR é sua modularidade. Esta modularidade permite a fácil troca dos componentes do sistema, tais como o banco de dados e o código de física de reatores, sem que o funcionamento do sistema tenha que ser alterado.

Um dos grandes problema da utilização dos algoritmos evolucionários na solução do problema da recarga é o elevado custo computacional (tempo gasto) para a avaliação dos indivíduos da população pelo código de física de reatores. O custo total da otimização aumenta, logicamente, com o aumento do tamanho da população, número de iterações a serem realizadas e características do problema. Uma grande contribuição que este trabalho veio trazer para reduzir este problema foi a introdução de um banco de dados contendo todos os resultados dos indivíduos já avaliados pelo código de física de reatores. Como a busca de um indivíduo no banco é muitas vezes menor que o tempo gasto pelo código de física de reatores para avaliar este indivíduo, tem-se uma redução do tempo gasto ao longo do processo de otimização.

Outra grande dificuldade na utilização de algoritmos evolucionários para a solução do problema da recarga é a definição da função de avaliação dos indivíduos presentes na população. Esta função deve ser definida de tal forma que mantenha um equilíbrio entre os indivíduos que atendam ao Limite de Especificação Técnica da usina e os que não

atendem a este Limite. Este equilíbrio é de fundamental importância para manter a diversidade da população e permitir que o espaço de busca seja bem explorado. Diversas funções de avaliação vêm sendo estudadas ao longo dos anos, mas nenhuma se mostrou ser eficiente para os diversos tipos de otimização do problema da recarga. O conceito de dominância utilizado pelo algoritmo PBIL_MO para a seleção dos melhores indivíduos, elimina a necessidade da definição de uma função de avaliação combinando os objetivos, o que é uma grande vantagem na solução de problemas complexos, onde esta função não esteja bem definida.

Em vista dos resultados obtidos pode-se afirmar que os objetivos desta tese foram atingidos. Foi demonstrado que o novo algoritmo desenvolvido foi capaz de realizar com sucesso processos de otimização em problemas Multi_Objetivo. No caso do problema da recarga de reatores nucleares do tipo PWR, gerar superfícies de Pareto, sem a necessidade de se especificar uma fitness complexa e com a vantagem sobre o algoritmo genético e o PBIL_N de fornecer mais de um padrão de carregamento ao especialista em recarga, que pode optar pelo padrão que melhor atende as necessidades do próximo ciclo de operação da usina.

Uma proposta as ser estudada futuramente, é um processo de paralelização da interface de comunicação do sistema PBIL_MO – Código de Física de Reatores, visando distribuir a avaliação dos indivíduos presentes na população por um conjunto de computadores trabalhando em paralelo, acarretando uma redução do tempo de processamento.

A baixa diversidade é um fator crítico quando o problema é de grande complexidade, como no caso do problema da recarga, uma só população tende a uniformizar-se com o tempo. Para a solução deste problema pode-se empregar alguma estratégia que mantenha esta diversidade como os métodos de nicho. Em um trabalho futuro pode-se desenvolver-se uma técnica de nicho a ser aplicada no algoritmo PBIL_MO. O grande problema a ser resolvido nas técnicas de nicho é definir uma métrica a ser utilizada para este tipo de problema (SACCO, 2004).

E por fim, como continuidade a este trabalho de pesquisa, dentro do problema da recarga deve-se buscar definir novos objetivos a fim de se testar o desempenho do algoritmo PBIL_MO e a substituição do código RECNOD por um código de uso comercial como o ANC.

APÊNDICE A

A.1 – Representação do Algoritmo Genético

```
**** Inicialização da População

        Para i =1 até TamahoPop faça
                vetor solução (i) = Gere indivíduo aleatório

*** Programa Principal
        enquanto ( Não terminou)

        **** Avalia indivíduo
                Para i =1 até TamanhoPop faça
                        avaliação (i) = avalie solução (vetor solução (i))

        **** Aplicar operadores genéticos
                *** Recombinação
                        Para i = 1 até Num_Cross
                                Seleciona Pai 1
                                Seleciona Pai 2
                                Gera Filho 1 e Filho 2
                *** Mutação
                        Seleciona indivíduo
                        Para i =1 até Num_Mut
                                Seleciona posição no indivíduo
                                Altera valor do Bit

        **** Verifica Fim

onde:
        TamanhoPop  - Tamanho da população do AG
        Num_Cross   - Número de vezes que se realiza o processo de Recombinação
        Num_Mut     - Número de Bits a serem alterados
```

Figura A.1 – Representação esquemática do Algoritmo Genético

A.2 – Teorema fundamental dos Algoritmos Genéticos

Uma importante estrutura contida nos algoritmos genéticos são os esquemas. Um esquema é o modelo de similaridade que descreve um subconjunto da cadeia de bits com similaridade em certas posições desta cadeia. A motivação para se considerar importante as similaridades nas cadeias de bits é a utilização destas similaridades para se obter mais informações que ajudem a guiar o processo de busca.

Para um alfabeto ternário { 0, 1, * } formado pela adição do alfabeto binário { 0 , 1 } com o símbolo * ("tanto faz", que indica que a posição pode conter o valor 0 ou 1), o esquema de comprimento de 5 bits, H = * 0 0 0 0 representa duas cadeias de bits, quais sejam, { 1 0 0 0 0, 0 0 0 0 0 } e o esquema H = * 1 1 1 * descreve um conjunto formado por quatro membros { 1 1 1 1 1, 1 1 1 1 0, 0 1 1 1 1, 0 1 1 1 0 }. Desta forma, para uma cadeia de bits de comprimento L empregando um alfabeto contendo k caracteres, o número de possíveis esquemas é dado por $(k + 1)^L$.

Todos os esquemas não são criados de mesma forma e alguns trazem mais informações que os outros. Para se quantificar estas informações, são apresentadas duas propriedades dos esquemas: a ordem do esquema o(H), que é o número de posições fixas (valor 0 ou 1) presentes na cadeia de bits, por exemplo, para o esquema H = 0 1 1 * 1 * * o valor de o(H) = 4 e o tamanho do esquema $\delta(H)$, que é a distância entre a primeira e a última posição fixa da cadeia de bits, para o esquema anterior $\delta(H) = 5 - 1 = 4$.

Ao longo do processo de busca, os operadores genéticos vão alterando o número de esquemas presentes na população de possíveis soluções.

O efeito do operador de seleção no número de esquemas esperado na próxima população é determinado da seguinte forma: seja m o número de esquemas H em um passo t, representado por m = m(H, t). Durante a seleção, as cadeias de bits são escolhidas de acordo com o valor de sua *fitness* segundo uma probabilidade $P_i = \frac{f_i}{\sum_j f_j}$.

O número de representantes de esquemas H na população no tempo t+1, m(H, t+1) é dado pela equação:

$$m\ (H,t+1) = m(H,t) \times n \times \frac{f\ (H)}{\sum_j f_j} \qquad (\text{A.1})$$

onde: n = tamanho da população;

f(H) = média da *fitness* das cadeias de bits representadas pelo esquema H no tempo *t*.

Considerando-se que a média da *fitness* de toda a população pode ser dada por:

$\bar{f} = \frac{\sum_{j=1}^{n} f_j}{n}$, então pode-se reescrever a equação (A.1) da seguinte forma:

$$m\ (H,t+1) = m(H,t) \times \frac{f\ (H)}{\bar{f}} \qquad (\text{A.2})$$

Da equação (A.2) conclui-se que um esquema cresce com a razão entre a média da *fitness* dos esquemas e a média da *fitness* da população. Em outras palavras, esquemas com valores de *fitness* acima da média da população serão aumentados na próxima geração.

Supondo que um determinado esquema H se mantenha acima da média da população, com uma quantidade $c\,\overline{f}$, com c constante, pode-se reescrever a equação (A.2) da seguinte forma:

$$m\ (H,t+1) = m\left(H,t\right) \times \frac{\overline{f} + c\,\overline{f}}{\overline{f}} \ = \ m(H,t) \times (\,1 + c\,) \qquad (\text{A.3})$$

Partindo-se de t=0 e supondo um valor estacionário para c, tem-se:

$$m\ (H,t) = m\left(H,\ 0\right) \times (\,1 + c\,)^{t} \qquad (\text{A.4})$$

O efeito do operador de *crossover* sobre a formação dos esquemas, parte do principio que, escolhendo-se aleatoriamente um ponto de corte na cadeia de bits, a probabilidade de um esquema ser destruído é dada por

$$P_d = \frac{\delta\left(H\right)}{(\,L-1\,)} \qquad (\text{A.5})$$

onde, L – comprimento da cadeia de bits

e a probabilidade de sobrevivência P_s, por:

$$P_s = 1\text{-}\,P_d \Rightarrow P_s = 1 - \frac{\delta\left(H\right)}{(\,L-1\,)} \qquad (\text{A.6})$$

Se o *crossover* é executado por uma escolha aleatória, com uma probabilidade P_c, a probabilidade de sobrevivência pode ser reescrita como:

$$P_s \ \geq 1\text{-}\ P_c \times \frac{\delta\left(H\right)}{(\,L-1\,)} \qquad (\text{A.7})$$

O efeito combinado do operador de seleção e *crossover* é obtido pela multiplicação do número de esquemas esperados para a seleção pela probabilidade de sobrevivência após o *crossover*, representado na seguinte equação:

$$m(H,t+1) \geq m(H,t) \times \frac{f(H)}{\bar{f}} \left[1 - P_c \times \frac{\delta(H)}{(L-1)} \right] \qquad (A.8)$$

O último operador a ser considerado é a mutação. Considerando a mutação como a alteração aleatória de uma única posição com probabilidade P_m. Para a sobrevivência do esquema H, todas as posições fixadas têm que sobreviver. Sendo a sobrevivência de uma posição dada por $(1 - P_m)$ e que cada uma das mutações é estatisticamente independente, um determinado esquema sobrevive quando cada uma das o(H) posições fixadas do esquema sobrevivem. Multiplicando a probabilidade de sobrevivência $(1 - P_m)$ por ela mesmo o(H) vezes, temos a probabilidade de mutação dada por $(1 - P_m)^{o(H)}$. Para valores pequenos de P_m ($P_m << 1$) a probabilidade de sobrevivência do esquema pode ser aproximada por:

$$P_s \cong (1 - o(H)\ P_m) \qquad (A.9)$$

Assim, temos que o número de esquemas esperado, na próxima geração, após a atuação dos operadores seleção, *crossover* e mutação é dada por:

$$m(H,t+1) \geq m(H,t) \times \left[1 - P_c \times \frac{\delta(H)}{(L-1)} - o(H) \times P_m \right] \times \left(\frac{f(H)}{\bar{f}} \right) \qquad (A.10)$$

Da equação (A.10) concluímos que para esquemas curtos, de baixa ordem e acima da média da população, apresentam um aumento no seu número de uma geração para a outra. Esta equação representa o teorema fundamental dos algoritmos genéticos e

demonstra que com a evolução do processo de busca existe uma convergência dos esquemas presentes na população.

APÊNDICE B

B.1 Representação do Algoritmo PBIL Padrão

O algoritmo PBIL padrão pode ser representado da seguinte maneira:

```
**** Inicialização do vetor de Probabilidades

        Para i =1 até COMPRIMENTO do P(i) = 0.5

        enquanto ( Não terminou)

        **** gera população
                Para i =1 até N_EXEMPLOS faça

                        vetor solução (i) = gera vetor exemplo de acordo com a probabilidade P

                        avaliação (i) = avalie solução (vetor solução (i))

                primeiro melhor vetor = procure o melhor vetor de acordo com a avaliação
                segundo melhor  vetor = procure o segundo melhor vetor de acordo com a avaliação

        **** atualiza vetor probabilidade com o melhor vetor
                Para i = 1 até COMPRIMENTO faça
                        P(i) = P(i) * (1 - Lr) + melhor vetor (i) * Lr

onde:
        COMPRIMENTO – número de bits na solução (determinado pela codificação do problema)
        Lr -  taxa de aprendizado
        N_EXEMPLOS – número de vetores gerados pelo vetor probabilidade (semelhante ao tamanho da
        população no GA)
```

Figura B.1 – Representação esquemática do Algoritmo PBIL padrão

B.2 Relação entre o PBIL e o Algoritmo Genético

Uma das características importantes que difere os algoritmos genéticos dos métodos convencionais de otimização é o paralelismo no processo de busca. Neste contexto, paralelismo não ser refere a habilidade de implementação de algoritmos em paralelo e sim referindo-se a habilidade de representar um grande número de possíveis soluções em uma população para uma única geração. Com o andamento do processo de busca, a população do algoritmo genético tende a convergir em torno de um bom vetor solução (vetor com alto valor de fitness) pertencente ao espaço de busca. Como resultado, é de se esperar que ao final do processo de busca a maioria das respectivas posições de bits nas cadeias de soluções (cromossomos) tenham convergido para o mesmo valor.

O algoritmo PBIL utiliza um vetor probabilidade que é protótipo para a criação dos vetores solução que possuam alto valor de fitness dentro do espaço de busca a ser explorado. Com o processo de busca do algoritmo PBIL os valores contidos no vetor probabilidade são movidos de um valor inicial de 0.5 para valores próximos a 0.0 ou 1.0. Desta maneira, de forma semelhante ao algoritmo genético, o algoritmo PBIL também converge de uma população inicial com alta diversidade para um único ponto onde as probabilidades são próximas a 0.0 ou 1.0. Neste ponto, existe um alto grau de similaridade na geração dos vetores solução, por parte do algoritmo PBIL.

Devido à utilização, no algoritmo PBIL, de um único vetor de probabilidade, ele aparentemente parece ter um menor poder de representação que o algoritmo genético, que utiliza toda a população na representação de um grande número de pontos simultaneamente, onde o vetor probabilidade da população 1 e 3 (Figura B.2) são

iguais, apesar das populações serem completamente diferentes. No caso dos algoritmos genéticos estas duas populações não seriam tratadas de forma semelhante.

Esta parece ser a limitação fundamental do algoritmo PBIL. Entretanto, o algoritmo genético não é capaz de manter cada uma destas populações ao longo do processo de busca. Devido a erros de amostragem, durante o processo de seleção de indivíduos dentro da população, esta irá convergir para uma ou outra representação; o algoritmo genético não é capaz de manter, por muito tempo, múltiplos pontos em diferentes regiões do espaço de busca.

População 1	População 2	População 3
0 0 1 1	1 0 1 0	1 0 1 0
1 1 0 0	1 1 0 0	0 1 0 1
1 1 0 0	1 1 0 0	1 0 1 0
0 0 1 1	1 1 0 0	0 1 0 1
vetor probab.	vetor probab.	vetor probab.
0.5, 0.5, 0.5, 0.5	1.0, 0.75, 0.25, 0.0	0.5, 0.5, 0.5, 0.5

Figura B.2 - Representação de uma população de 4 bits e seus respectivos vetores probabilidade

Esta característica é apresentada no seguinte teorema (BALUJA, 1994):

"..... Em uma população de tamanho finito, mesmo que não exista uma vantagem na seleção entre duas alternativas concorrentes ..... a população irá convergir para uma ou outra alternativa em um tempo finito. ....."

De forma similar, o algoritmo PBIL irá convergir para um vetor probabilidade que represente uma das alternativas, representando o vetor, ao final do processo de busca, somente uma das populações.

B.3 Aprendizado competitivo e LVQ

A forma de aprendizado atualmente utilizada no algoritmo PBIL é semelhante a forma de aprendizado das redes de aprendizado competitivo empregando um algoritmo LVQ (Learning Vector Quantization) desenvolvido por Kohonen (1990).

O aprendizado competitivo CL (Competitive Learning) é freqüentemente usado para agrupar um número de pontos sem classificação, em grupos distintos. A participação dos pontos em cada grupo é baseada na similaridade destes pontos com respeito às características em estudo. O processo de classificação é não-supervisionado, e não existe a priori conhecimento de quantos grupos existem, ou que características podem distinguir cada grupo. O que se deseja é que o procedimento do CL seja capaz de determinar as características mais relevantes para a formação das classes, agrupando os pontos nos grupos baseado nestas características.

O CL é freqüentemente estudado no contexto das Redes Neurais Artificiais. O algoritmo usado para o treinamento destas redes é descrito a seguir:

Os pesos iniciais da rede são escolhidos de forma aleatória, e estão sujeitos a restrições de normalização. A ativação da unidade de saída é calculada pela seguinte fórmula:

$$Y_i = \sum_j W_{ij} \; X_i \qquad \text{(B.1)}$$

Onde: Y_i – valor do componente i do vetor de saída da rede;

W_{ij} – é o peso da conexão entre i e j;

X_i – valor do componente i do vetor de entrada da rede.

Em uma rede de aprendizado competitivo, somente a unidade de saída com a maior ativação será acesa para cada ponto apresentado. A unidade de saída vencedora corresponde à classificação do ponto de entrada. Durante o processo de treinamento da rede, os pesos das unidades de saída vencedoras são alterados para aproximarem-se dos pontos apresentados. O ajuste dos pesos é realizado de acordo com a seguinte regra:

$$\Delta W_{ij} = Lr \times (X_i - W_{ij}) \qquad \text{(B.2)}$$

onde: Lr – taxa de aprendizado da rede.

O processo de treinamento das redes CL envolve a apresentação, repetidas vezes, de cada ponto a rede até que esta se estabilize ou outro critério seja usado para determinar quando se parar o processo de treinamento.

Após o término do treinamento da rede, o vetor de pesos para cada unidade de saída pode ser considerado um vetor de protótipo para uma das classes desconhecidas. Esta é a noção da criação do vetor protótipo que será o centro da discussão do algoritmo PBIL.

Embora o método de treinamento descrito acima seja não-supervisionado, o aprendizado competitivo supervisionado também vem sendo explorado no treinamento de redes neurais artificiais. Um destes métodos é o LVQ (Learning Vector Quantization.

No LVQ o método pelo qual a unidade de saída é escolhida é o mesmo que o utilizado nas redes CL, porém, no LVQ a classe para cada ponto utilizado no treinamento dos pesos da rede é conhecida a priori.

A rede é treinada com os exemplos positivos e negativos. Se a classificação feita pela rede é correta, os pesos da unidade de saída ativada são reforçados; se a classificação é incorreta, os pesos são enfraquecidos.

Este processo corresponde ao primeiro tipo de LVQ descrito por Kohonen. Este processo pode ser descrito da seguinte forma:

Supondo que o vetor de Voronoi W_c, vetor que apresenta a menor distorção na representação de um vetor de entrada, é a representação mais próxima para um vetor de entrada da rede denotado por X_i. Sendo $\&W_c$ a classe associada com o vetor de Voronoi W_c e $\&X_i$ a classe a qual pertence o vetor de entrada X_i. O vetor de Voronoi é ajustado como a seguir:

Se a classificação é correta => $\&W_c = \&X_i$ então

$$W_c(n+1) = W_c(n) + \alpha_n \left[X_i - W_c(n)\right] \qquad (B.3)$$

Onde $0 < \alpha_n < 1$.

Se a classificação é incorreta => $\&W_c \neq \&X_i$ então

$$W_c(n+1) = W_c(n) - \alpha_n \left[X_i - W_c(n)\right] \qquad (B.4)$$

B.4 Comparação entre o Algoritmo Genético e o PBIL_N

A seguir é apresentada uma comparação entre o algoritmo genético e o algoritmo PBIL_N para o problema do TSP (Traveling Salesman Problem).

O problema do TSP pode ser apresentado da seguinte forma: Dado um determinado número de cidades N, o objetivo do problema é encontrar a menor distância a ser percorrida para se visitar todas as N cidades, sem que nenhuma cidade seja visitada mais de uma vez, retornando-se ao final das visitas a cidade de partida. Em outras palavras, deseja-se dar uma volta completa passando-se por todas as N cidades uma única vez pelo menor percurso possível.

Nas tabelas B.1 e B.2, apresentadas a seguir, são apresentados os resultados obtidos empregando-se o código GENESIS (GREFENSTETTE, 1990) que é uma forma de implementação do algoritmo genético, e os obtidos com o algoritmo PBIL_N (MACHADO, 1999). Na primeira tabela, são apresentados os resultados obtidos para o TSP Oliver 30, TSP com 30 cidades. Este TSP possui uma baixa complexidade e a maioria dos algoritmos evolucionários conseguem obter a solução exata deste problema que é igual a 423,7. Na tabela B.2 são apresentados os resultados para o TSP Rykel 48, um TSP com 48 cidades e que possui como resultado final o valor de 14422.

Tabela B.1- Resultados obtidos para o TSP Oliver 30

	Algoritmo	
Avaliações	AG	PBIL-N
2500	752,8	830,1
5000	606,5	764,6
7500	557,2	550,8
10000	495,2	423,9
12500	473,6	423,7
15000	452,2	423,7
17500	435,2	423,7
20000	423,7	423,7

Da tabela B.1, verifica-se que os dois algoritmos foram capazes de obter a solução para o TSP Oliver 30, porém o algoritmo PBIL_N necessita de um número de avaliações bem menor que o algoritmo genético, o que corresponde a um tempo computacional menor.

Tabela B.2 - Resultados obtidos para o TSP Rykel 48

Avaliações (x 10^3)	Algoritmo	
	AG	PBIL-N
50	26441	35244
100	18786	28365
150	16917	21734
200	16535	15430

Para o TSP Rykel 48, nenhum dos dois algoritmo foi capaz de atingir a solução do problema, mas o resultado apresentado pelo algoritmo PBIL_N é superior ao do AG.

ANEXO C

C.1 - Dados do Problema da Sacola (0 – 1) Multi_Objetivo (2 sacolas, 100 itens)

Sacola 1:
Capacidade: +2732

item 1: Peso: +94 Valor: +57 item 2: Peso: +74 Valor: +94 item 3: Peso: +77 Valor: +59 item 4: Peso: +74 Valor: +83 item 5: Peso: +29 Valor: +82 item 6: Peso: +11 Valor: +91 item 7: Peso: +73 Valor: +42 item 8: Peso: +80 Valor: +84 item 9: Peso: +81 Valor: +85 item 10: Peso: +82 Valor: +18	item 11: Peso: +75 Valor: +94 item 12: Peso: +42 Valor: +18 item 13: Peso: +44 Valor: +31 item 14: Peso: +57 Valor: +27 item 15: Peso: +20 Valor: +31 item 16: Peso: +20 Valor: +42 item 17: Peso: +99 Valor: +58 item 18: Peso: +95 Valor: +57 item 19: Peso: +52 Valor: +55 item 20: Peso: +81 Valor: +97	item 21: Peso: +68 Valor: +79 item 22: Peso: +16 Valor: +10 item 23: Peso: +79 Valor: +34 item 24: Peso: +30 Valor: +100 item 25: Peso: +16 Valor: +98 item 26: Peso: +90 Valor: +45 item 27: Peso: +21 Valor: +19 item 28: Peso: +49 Valor: +77 item 29: Peso: +70 Valor: +56 item 30: Peso: +78 Valor: +25	item 31: Peso: +77 Valor: +60 item 32: Peso: +21 Valor: +22 item 33: Peso: +84 Valor: +84 item 34: Peso: +19 Valor: +89 item 35: Peso: +65 Valor: +12 item 36: Peso: +38 Valor: +46 item 37: Peso: +25 Valor: +20 item 38: Peso: +43 Valor: +85 item 39: Peso: +99 Valor: +42 item 40: Peso: +75 Valor: +94	item 41: Peso: +80 Valor: +20 item 42: Peso: +10 Valor: +65 item 43: Peso: +44 Valor: +27 item 44: Peso: +26 Valor: +34 item 45: Peso: +21 Valor: +27 item 46: Peso: +74 Valor: +91 item 47: Peso: +20 Valor: +17 item 48: Peso: +22 Valor: +56 item 49: Peso: +81 Valor: +23 item 50: Peso: +89 Valor: +89

item 51: Peso: +15 Valor: +18 item 52: Peso: +35 Valor: +11 item 53: Peso: +24 Valor: +91 item 54: Peso: +16 Valor: +79 item 55: Peso: +43 Valor: +14 item 56: Peso: +75 Valor: +99 item 57: Peso: +25 Valor: +45 item 58: Peso: +76 Valor: +73 item 59: Peso: +48 Valor: +81 item 60: Peso: +75 Valor: +96	item 61: Peso: +15 Valor: +51 item 62: Peso: +23 Valor: +96 item 63: Peso: +10 Valor: +63 item 64: Peso: +81 Valor: +40 item 65: Peso: +81 Valor: +93 item 66: Peso: +67 Valor: +87 item 67: Peso: +58 Valor: +71 item 68: Peso: +77 Valor: +54 item 69: Peso: +49 Valor: +74 item 70: Peso: +16 Valor: +15	item 71: Peso: +65 Valor: +32 item 72: Peso: +74 Valor: +57 item 73: Peso: +14 Valor: +70 item 74: Peso: +41 Valor: +62 item 75: Peso: +74 Valor: +12 item 76: Peso: +74 Valor: +71 item 77: Peso: +17 Valor: +57 item 78: Peso: +12 Valor: +97 item 79: Peso: +95 Valor: +48 item 80: Peso: +29 Valor: +33	item 81: Peso: +75 Valor: +42 item 82: Peso: +61 Valor: +15 item 83: Peso: +59 Valor: +39 item 84: Peso: +37 Valor: +91 item 85: Peso: +75 Valor: +17 item 86: Peso: +90 Valor: +63 item 87: Peso: +17 Valor: +81 item 88: Peso: +79 Valor: +49 item 89: Peso: +15 Valor: +60 item 90: Peso: +88 Valor: +90	item 91: Peso: +76 Valor: +87 item 92: Peso: +93 Valor: +25 item 93: Peso: +98 Valor: +15 item 94: Peso: +80 Valor: +30 item 95: Peso: +33 Valor: +76 item 96: Peso: +39 Valor: +76 item 97: Peso: +96 Valor: +53 item 98: Peso: +71 Valor: +59 item 99: Peso: +39 Valor: +40 item 100: Peso: +49 Valor: +59

Sacola 2:
Capacidade: +2753

item 1: Peso: +55 Valor: +20 item 2: Peso: +10 Valor: +19 item 3: Peso: +97 Valor: +20 item 4: Peso: +73 Valor: +66 item 5: Peso: +69 Valor: +48 item 6: Peso: +23 Valor: +100 item 7: Peso: +62 Valor: +13 item 8: Peso: +47 Valor: +87 item 9: Peso: +90 Valor: +62 item 10: Peso: +62 Valor: +73	item 11: Peso: +96 Valor: +53 item 12: Peso: +88 Valor: +79 item 13: Peso: +95 Valor: +17 item 14: Peso: +61 Valor: +93 item 15: Peso: +94 Valor: +78 item 16: Peso: +16 Valor: +22 item 17: Peso: +91 Valor: +85 item 18: Peso: +61 Valor: +86 item 19: Peso: +27 Valor: +56 item 20: Peso: +18 Valor: +56	item 21: Peso: +96 Valor: +44 item 22: Peso: +77 Valor: +86 item 23: Peso: +14 Valor: +94 item 24: Peso: +36 Valor: +93 item 25: Peso: +17 Valor: +57 item 26: Peso: +56 Valor: +31 item 27: Peso: +83 Valor: +20 item 28: Peso: +41 Valor: +35 item 29: Peso: +52 Valor: +70 item 30: Peso: +69 Valor: +79	item 31: Peso: +97 Valor: +58 item 32: Peso: +45 Valor: +24 item 33: Peso: +94 Valor: +84 item 34: Peso: +45 Valor: +12 item 35: Peso: +47 Valor: +17 item 36: Peso: +28 Valor: +43 item 37: Peso: +82 Valor: +35 item 38: Peso: +13 Valor: +47 item 39: Peso: +82 Valor: +92 item 40: Peso: +39 Valor: +38	item 41: Peso: +28 Valor: +93 item 42: Peso: +41 Valor: +50 item 43: Peso: +55 Valor: +27 item 44: Peso: +12 Valor: +100 item 45: Peso: +50 Valor: +36 item 46: Peso: +32 Valor: +30 item 47: Peso: +97 Valor: +23 item 48: Peso: +87 Valor: +22 item 49: Peso: +36 Valor: +56 item 50: Peso: +11 Valor: +73

item 51: Peso: +20 Valor: +55 item 52: Peso: +37 Valor: +32 item 53: Peso: +87 Valor: +75 item 54: Peso: +91 Valor: +42 item 55: Peso: +19 Valor: +82 item 56: Peso: +22 Valor: +80 item 57: Peso: +89 Valor: +55 item 58: Peso: +54 Valor: +48 item 59: Peso: +20 Valor: +93 item 60: Peso: +78 Valor: +28	item 61: Peso: +52 Valor: +26 item 62: Peso: +35 Valor: +42 item 63: Peso: +18 Valor: +96 item 64: Peso: +96 Valor: +93 item 65: Peso: +54 Valor: +16 item 66: Peso: +10 Valor: +39 item 67: Peso: +39 Valor: +46 item 68: Peso: +17 Valor: +80 item 69: Peso: +51 Valor: +24 item 70: Peso: +40 Valor: +87	item 71: Peso: +25 Valor: +37 item 72: Peso: +84 Valor: +73 item 73: Peso: +54 Valor: +81 item 74: Peso: +31 Valor: +38 item 75: Peso: +97 Valor: +98 item 76: Peso: +37 Valor: +13 item 77: Peso: +63 Valor: +91 item 78: Peso: +39 Valor: +85 item 79: Peso: +60 Valor: +17 item 80: Peso: +87 Valor: +59	item 81: Peso: +63 Valor: +58 item 82: Peso: +37 Valor: +56 item 83: Peso: +13 Valor: +93 item 84: Peso: +31 Valor: +66 item 85: Peso: +84 Valor: +64 item 86: Peso: +90 Valor: +17 item 87: Peso: +84 Valor: +10 item 88: Peso: +57 Valor: +33 item 89: Peso: +21 Valor: +28 item 90: Peso: +64 Valor: +97	item 91: Peso: +63 Valor: +25 item 92: Peso: +21 Valor: +42 item 93: Peso: +95 Valor: +17 item 94: Peso: +83 Valor: +23 item 95: Peso: +81 Valor: +37 item 96: Peso: +45 Valor: +46 item 97: Peso: +68 Valor: +52 item 98: Peso: +89 Valor: +33 item 99: Peso: +100 Valor: +26 item 100: Peso: +14 Valor: +90

APÊNDICE D

D.1 – Tipos de operadores de Crossover

A seguir são apresentados alguns tipos de operadores de Crossover que são utilizados pelos algoritmos genéticos. Os três primeiros são aplicados a algoritmos genéticos que empregam cadeias binárias de comprimento fixo. Os demais são operadores especiais para representações que não empregam codificações binárias. O símbolo | | representa o ponto de corte para as cadeias.

D.1.1 - Crossover de um ponto:

Pai 1 = 0000000000000| |000000
Pai 2 = 1111111111111| |111111

⇒

Filho 1 = 0000000000000| |111111
Filho 2 = 1111111111111| |000000

D.1.2 - Crossover de 2 pontos:

Pai 1 = 000000| |0000000| |00000
Pai 2 = 111111| |1111111| |11111

⇒

Filho 1 = 000000| |1111111| |00000
Filho 2 = 111111| |0000000| |11111

D.1.3 - Crossover Uniforme:

Pai 1 = 111111111111 Pai 2 = 000000000000

Filho 1 = 101010101010 Filho 2 = 010101010101

D.1.4 - Partially Matched Crossover (PMX)

A = 9 8 4| |5 6 7| | 1 3 2 10 A’ = 9 8 4(2 3 10(1 6 5 7

B = 8 7 1| |2 3 10| |9 5 4 6 B’ = 8 10 1(5 6 7(9 2 4 3

D.1.5 - Order Crossover (OX)

A = 9 8 4(5 6 7(1 3 2 10

B = 8 7 1(2 3 10(9 5 4 6 (

B’ = 8 H 1(2 3 10(9 H 4 H (

B’ = 2 3 10(H H H(9 4 8 1

A’ = 5 6 7(2 3 10(1 9 8 4

B’ = 2 3 10(5 6 7(9 4 8 1

D.1.6 - Cycle Crossover (CX)

D = 1 2 3 4 5 6 7 8 9 10

C = 9 8 2 1 7 4 5 10 6 3

C' = 9 _ _ 1 _ 4 _ _ 6 _

C' = 9 2 3 1 5 4 7 8 6 10

D' = 1 8 2 4 7 6 5 10 9 3

REFERÊNCIAS BIBLIOGRÁFICAS

ANDERSSON, J., 2000, *A survey of multiobjective optimization in engineering design*, Technical Report: Department of Mechanical Engineering – Linköping University, Linköping, Sweden.

ANEEL, 2005, *Banco de Informações de Geração*, Agência Nacional de energia elétrica – www.aneel.gov.br - Abril.

BALUJA, S., 1994, *Population-Based Incremental Learning: A method for integrating genetic search based function optimizations and competitive learning*, Technical Report CMU-CS-94-163, June.

BALUJA, S., CARUANA, R., 1995, *Removing the genetics from de standard genetic algorithm*, Technical Report CMU-CS-95-141, May.

BEAN, J.C., 1994, *Genetic Algorithms and Random Keys for Sequencing and Optimization*, ORSA Journal on Computing, vol. 6, nº 2, Spring.

BENAYOUN, R., de MONTGOLFIER, J., TERGNY, J., and LARITCHEV, O., 1971, *Linear programming with multiple objective functions: Step Method (STEM)*, Mathematical Programming, vol 1, pp. 366-375.

BOYD, S. , VANDENBERGHE, L., 2004, *Convex Optimization*, Cambridge University Press, ISBN 0521833787.

CARTER, J.N., 1997, *Genetic Algorithms for incore fuel management and other recent developments in optimization*, Advances in nuclear science and technology, vol 25, Plenum Press, New York.

CHAPOT, J.L.C., 2000, *Otimização automática de recargas de reatores a água pressurizada utilizando algoritmos genéticos*, Tese de Doutorado, COPPE/UFRJ, Rio de janeiro, Brasil, Junho.

CHARNES, A., COOPER, W.W., 1961, *Management models and industrial applications of linear programming*, vol 1, John Wiley, New York.

DARWIN, C.R., 1859, *On the Origin of Species by Means of Natural Selection*. London, John Murray.

DeCHAINE, M. D., FELTUS, M. A, 1995, *Nuclear fuel management optimization using genetic algorithms*, Nuclear Tecnology, v. 111, pp. 109-114, July.

GALPERIN, A., 1995, *Exploration of the search space of the in-core fuel management problem by Knowledge-Based techniques*. Nuclear Science and Engineering, vol. 119, February, 144-152.

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

GREFFENSTETTE, J.J., 1990, *A User's Guide to GENESIS version 5.0*, Washington, D.C., Navy Centre for Applied Research in Artificial Intelligence.

HAUPT, R.L., HAUPT, S.E., 1998, *Pratical Genetic Algorithms*. Wiley-Interscience publication, New York.

HOLLAND, J.H., 1975, *Adaptation In Natural and Artificial Systems*, Ann Arbor, University of Michigan Press.

HORN, J., NAFPLIOTIS, N., 1993, *Multiobjective Optimization using the Niched Pareto Genetic Algorithm*. Technical Report IlliGAl Report 93005, University of Illinois, Urbana, Illinois, USA.

IAEA, 2005, *PRIS – Power Reactor Information System*, International Atomic Energy Agency, www.iaea.org, Abril.

KIRKPATRICK, S., GELATT, C.D., VECCHI, M.P., 1983, *Optimization by Simulated Annealing*. Science, vol. 220, May, 671-680.

KOHONEN, T., 1990, *The self-organizing map*. In: *Proceeding of the IEEE*, vol. 78, nº 2, pp. 1464 - 1480, September

KROPACKZEK, D. J., TURINSKY, P. J., 1991, *In-core nuclear fuel management optimization for presurized water reactors utilizing Simulated Annealing*, Nuclear Tecnology, vol 95, nº 9.

KROPACKZEK, D.J., TURINSKY, P.J., 1991, *In-core Nuclear Fuel Management Optimization for Pressurized Water Reactors Utilizing Simulated Annealing*. Nuclear Technology, vol 95 nº 9, July, 9-31.

KURSAWE, F., 1991, *A variant of evolution strategies for vector optimization.* In H. P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature. 1st Workshop, PPSN I*, volume 496 of *Lecture Notes in Computer Science*, pages 193-197, Berlin, Germany, october. Springer-Verlag.

LIMA, A.M.M., MACHADO, M.D., SCHIRRU, R., 2002, *Comparação do Modelo de Ilhas – Implementação Paralela – e do Modelo Serial do Algoritmo Evolucionário de Otimização PBIL para Recarga de Reatores Nucleares PWR*, Anais do XIII ENFIR.

MACHADO, L., 2001, *Otimização de Recarga do Combustível Nuclear por Agentes Artificiais*, Tese de D.Sc., Programa de Engenharia Nuclear, COPPE/UFRJ, Rio de Janeiro, Brasil, Março.

MACHADO, M. D., SCHIRRU, R, 2000 *Um Novo Algoritmo Evolucionário com Aprendizado LVQ na Otimização de Problemas Combinatoriais*. In: Anais do XII ENFIR, Rio de Janeiro, Brasil.

MACHADO, M. D., SCHIRRU, R., MARTINEZ, A. S., PEREIRA, C. M. N. A., DOMINGOS, R. P., MACHADO, L., 1999, *Intelligent Soft Computing in Nuclear Engineering in Brasil.* Progress in Nuclear Energy, Inglaterra, v. 35, n. 3-4, p. 367-391.

MACHADO, M.D., 1999, *Um Novo Algoritmo Evolucionário com Aprendizado LVQ para a Otimização de Problemas Combinatórios como a Recarga de Reatores Nucleares*, Tese de M.Sc., Programa de Engenharia Nuclear, COPPE/UFRJ, Rio de Janeiro, Brasil, Abril.

MITCHELL, M., 1998, *An Introduction to Genetic Algorithms*, Cambridge, MA, MIT Press.

OSYCZKA, A., 1985, *Multicriteria Optimization for engineering desing*. In J.S. GERO ED., Desing Optimization, pp. 193-227. Academic Press.

PARETO, V., 1896, *Cours d'Economie Politique*. Volume I e II F. Rouge, Lausanne

PARKS, G. T., 1996, *Multiobjective Pressurized Water Reactor Reload Core Design by Nondominated Genetic Algorithm Search*, Nuclear Science and Engineering, vol. 124, pp. 178-187.

POON, P. W., PARKS, G. T., 1992, *Optimising PWR reload core Desingns*. In: Parallel Problem Solving from Nature 2, Elsevier Science Publishers B.V., pp. 371-380.

RITZEL, B.J., EHEART, J.W., RANJITHAN, S., 1994, *Using genetic algorithms to solve a multiple objective groundwater pollution containment problem.* Water Resource Research 30, may, 1580-1603.

SACCO, W.F., 2004, *Aplicação do método clearing nebuloso aos algoritmos genéticos na otimização de projeto de reatores nucleares*. Tese de Doutorado COPPE/UFRJ, Rio de Janeiro, Brasil, fevereiro.

SRINIVAS, N., DEB, K., 1994, *Multiobjctive Optimization Using Nondominated Sorting in Genetic Algorithm*. Evolutionary Computation 2, 3 (fall), 221-148.

STEUER, R.; 1986, "Multiple criteria optimization: theory, computation and application". New York, John Wiley & Sons, Inc.

SUPPAPITNARM A., SEFFEN, K. A., PARKS, G.T., CLARKSON, P.J., LIU, J. S., 1999, *Design by multiobjective optimization using simulated annealing*. International conference on engineering desing ICDE 99, Munich, Germany.

TAMIZ, M., JONES, D., and ROMERO, C., 1998, "Goal programming for decision making: an overview of the current state-of-the-art", European Journal of Operational Research, vol 111, pp. 569-581.

WNA, 2005, *Nuclear Power in the World Today*, Word Nuclear Association – www.world-nuclear.org - January.

ZADEH, L. A., 1965, *Fuzzy sets*, Information and Control, Vol. 8, pp. 338 – 352.

ZITZLER, E., DEB, K., and THIELE, L., 2000, *Comparison of Multiobjective Evolutionary Algorithms: Empirical Results.* Evolutionary Computation, 8(2), pages 173-195, Summer 2000.

ZITZLER, E., LAUMANNS, M., and THIELE, L., 2001, *SPEA2: Improving the Strength Pareto Evolutionary Algorithm*. Technical Report 103, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, May.

ZITZLER, E., THIELE, L., 1999, *Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach*. IEEE Transactions on Evolutionary Computation, 3(4), pages 257-271, November.