



UNIVERSIDADE FEDERAL DE PERNAMBUCO  
CENTRO DE TECNOLOGIA E GEOCIÊNCIAS  
DEPARTAMENTO DE ENGENHARIA CIVIL



# **OTIMIZAÇÃO ESTRUTURAL UTILIZANDO O ALGORITMO EVOLUCIONÁRIO DO ENXAME DE PARTÍCULAS**

DISSERTAÇÃO DE MESTRADO APRESENTADA POR:

***Leonardo Correia de Oliveira***

Orientado por: Silvana Maria Bastos Afonso da  
Silva

Recife, PE – Brasil  
Maio de 2008.



UNIVERSIDADE FEDERAL DE PERNAMBUCO  
CENTRO DE TECNOLOGIA E GEOCIÊNCIAS  
DEPARTAMENTO DE ENGENHARIA CIVIL



# **OTIMIZAÇÃO ESTRUTURAL UTILIZANDO O ALGORITMO EVOLUCIONÁRIO DO ENXAME DE PARTÍCULAS**

por

***Leonardo Correia de Oliveira***

Dissertação submetida ao Corpo Docente do Curso de Pós-Graduação da Universidade Federal de Pernambuco, como parte dos requisitos necessários à obtenção do Grau de Mestre em Ciências em Engenharia Civil.

Orientadora: Silvana Maria Bastos Afonso da Silva

Recife, PE – Brasil  
Maio de 2008.

**O48o**

**Oliveira, Leonardo Correia de**

Otimização estrutural utilizando o algoritmo evolucionário do enxame de partículas / Leonardo Correia de Oliveira. - Recife: O Autor, 2008.

xxiv, 130 folhas, il : tabs. grafs.

Dissertação (Mestrado) – Universidade Federal de Pernambuco. CTG. Programa de Pós-Graduação em Engenharia Civil, 2008.

Inclui Bibliografia.

1. Engenharia Civil. 2. Otimização estrutural. 3. Algoritmo evolucionário. 4. Enxame de partículas I. Título.

**624 CDD (22 ed.)**

**UFPE  
BCTG/ 2009-058**

# OTIMIZAÇÃO ESTRUTURAL UTILIZANDO O ALGORITMO EVOLUCIONÁRIO DO ENXAME DE PARTÍCULAS

por

***Leonardo Correia de Oliveira***


Dissertação submetida ao Corpo Docente do Curso de Pós-Graduação da Universidade Federal de Pernambuco, como parte dos requisitos necessários à obtenção do Grau de Mestre em Ciências em Engenharia Civil.

Aprovada por:



---

Prof. Silvana Maria Bastos Afonso da Silva, Ph.D.



---

Prof. Bernardo Horowitz, Ph.D.



---

Prof. Ramiro Brito Willmersdorf, Ph.D.

Recife, PE - Brasil  
Maio de 2008.

# Agradecimentos

Agradeço a Deus pela perseverança e sabedoria concedida para realização deste trabalho; aos meus familiares pelo apoio e compreensão nas horas mais difíceis; aos meus amigos e irmãos de convívio diário, principalmente a Liliane, Sonia, Cláudio, Renato e Samuel, pelo auxílio dado e pelo ombro amigo ofertado nos momentos de desânimo; à minha orientadora, a professora Silvana, pela paciência e pelos ensinamentos; a CAPES, pelo auxílio financeiro; e aos professores e colegas da pós-graduação que, direta ou indiretamente, tornaram este trabalho possível.

# Resumo

Nas ciências em geral, o termo otimização se refere ao estudo de um conjunto de técnicas que têm como objetivo a obtenção de um melhor resultado para uma função e parâmetros (variáveis de projeto) pré-especificados dentro de um conjunto permitido (espaço de projeto). A otimização em geral é feita através de procedimentos numéricos computacionais. A maioria desses procedimentos utiliza algoritmos que fazem uso de gradientes devido principalmente à eficiência computacional dos mesmos no processo de obtenção de pontos de ótimo. No entanto, nas últimas décadas, algoritmos metaheurísticos (algoritmos que não requerem cálculos de gradientes no processo de otimização) têm atraído grande atenção da comunidade científica. Os algoritmos dessa classe geralmente imitam algum fenômeno da natureza e são comumente chamados de algoritmos evolucionários.

Dentre as alternativas existentes nessa classe de algoritmos, podem ser citados: o algoritmo genético (*genetic algorithm* – GA), o recozimento simulado (*simulated annealing* – SA) e o enxame de partículas (*particle swarm* – PS). Embora as técnicas citadas requeiram mais avaliações de funções para encontrar uma solução ótima, quando comparadas com algoritmos que utilizam o cálculo de gradientes, os algoritmos baseados em procedimentos evolucionários apresentam várias vantagens, a saber: facilidade de programação; não necessitam da garantia de continuidade nas funções envolvidas na definição do problema; mais adequado na determinação de um ótimo global ou próximo do global; e adequados na solução de problemas discretos.

Nos últimos três anos, o nosso grupo de pesquisa tem se empenhado na implementação computacional e uso do algoritmo de otimização do enxame de partículas (*Particle Swarm Optimization* - PSO). O algoritmo PSO se desenvolveu de experiências com algoritmos que modelavam o comportamento de muitas espécies de pássaros. A metodologia estudada tem fortes raízes em vida artificial e na psicologia social. Neste trabalho, o procedimento desenvolvido é aplicado a uma diversidade de problemas que têm o intuito de enfatizar a eficácia e versatilidade da metodologia estudada nos diversos tipos de problemas existentes, inclusive em problemas práticos da engenharia.

Várias versões foram desenvolvidas no ambiente MATLAB, onde o algoritmo PSO está implementado, tanto para problemas que envolvem uma única função objetivo como para aqueles que envolvem várias funções (otimização multiobjetivo). As várias opções disponíveis estão configuradas em um ambiente bastante fácil de entender e de operar.

A utilização de modelos substitutos de baixo custo computacional, porém de precisão aferida, constitui uma alternativa bastante promissora a ser utilizadas em tais algoritmos, evitando desta forma uma grande demanda de tempo computacional, característica inerente das metodologias evolucionárias acopladas a simuladores numéricos. Nesta combinação de estratégias, o grande número de avaliações de funções requeridas pelo algoritmo evolucionário não mais se darão através de simulações numéricas do problema real, e sim através de cálculos rápidos que consideram o uso de modelos aproximados. Neste contexto, a técnica escolhida foi o método das Bases Reduzidas. Além da opção de modelos substitutos, uma implementação alternativa utilizando os paradigmas da computação paralela foi realizada objetivando a eficiência computacional. Para essa implementação, as operações realizadas no algoritmo PSO, como atualizações no decorrer das iterações e cálculo de funções de avaliação, foram distribuídas entre as várias unidades de processamento disponíveis.

Ambos os aspectos acima mencionados são de crucial importância, especificamente para o caso da aplicação dos algoritmos PSO em problemas da engenharia prática. As aplicações deste trabalho se dirigiram ao uso de funções empíricas multimodais objetivando mostrar a potencialidade da metodologia em se determinar a solução global, bem como as funções provenientes da simulação numérica de treliças planas sob várias condições de solicitação. Foram conduzidas otimizações uni e multiobjetivo considerando a abordagem PSO, no contexto do uso de modelos reais e substitutos, e também alguns estudos utilizando o algoritmo na versão da implementação computacional paralela.

*Palavras chaves:* 1. Engenharia Civil. 2. Otimização estrutural. 3. Algoritmo evolucionário. 4. Enxame de partículas.

# Abstract

In sciences in general, the term optimization means the study of a set of approaches that aims to obtain the best output for a function and its parameters (design variables), pre-specified in an allowed set (design domain). The optimization of problems in general is carried out by computational numerical procedures. The majority of these procedures consider algorithms which use gradient information due to the computational efficiency to obtain the optimal solution. Meanwhile, in the latest decades, metaheuristic procedures, which don't need gradient information in the optimization procedure, have attracted attention of the scientific community. These algorithms, generally copy a natural phenomenon and they are commonly called evolutionary algorithms.

Among the existent alternatives of this class, can be mentioned genetic algorithm (GA), simulated annealing (SA) and particle swarm (PS). These techniques need more function evaluations to find an optimum design when compared with algorithm which uses gradient information, nevertheless, evolutionary-based algorithms offer many advantages: they are easier to implement than the others; they don't need the continuity guarantee for the functions involved on the problem definition; they are more suitable to find global optimum; and they are suitable to solve discrete problems.

In the last three years, our group has concentrated effort to implement/to use the algorithm of Particle Swarm Optimization (PSO). The optimization tool, which uses the PSO algorithm, was developed by experiences with algorithms that modeled the birds' behaviour. The method studied has strong links with artificial life and social psychology. In this work, the procedure developed is applied in a variety of problems that targets to emphasize the efficacy and the versatility of the method studied in many kinds of the existent problems, including usual problems of engineering.

A variety of versions was developed on MATLAB's environment where the PSO algorithm is implemented, both for problems involving only one objective function and for problems involving many objectives functions (multiobjective optimization). The many options available are configured on a simple way to understand and to use.



The use of surrogate models with low-cost computing, with accuracy measured, consist in a very promising way to be used on this algorithms to avoid a great demand of computational time, which is an inherit characteristic of the evolutionary methods coupled with numerical simulators. In this combination, the great number of functions evaluations required by the evolutionary algorithm won't be done by the numerical simulation of the real problem but by the fast functions computations using surrogate models. In this context, the chosen technique was the Reduced Basis Method (RBM). A part from surrogate models options, aiming computational efficiency, an alternative implementation using parallel programming paradigms was done. For this implementation, the operations on PSO algorithm were distributed among the available processors, as updates during the process and computation of evaluated functions.

Both the points of view mentioned above are crucial, especially for PSO algorithm, to be applied on usual engineering problems. The applications conduct in this work are involved with the use of multi-modals empirical functions, aiming to show the potential of the method to obtain the global optimum, and on functions derived from numerical simulations of plane trusses under a variety of boundary and load conditions. Optimizations uni-objective and multiobjective were conducted considering the PSO approach, using real models and surrogate models, and some studies considering the parallel version of the implementation were also carried out.

*Keywords:* 1. Civil Engineering. 2. Structural optimization. 3. Evolutionary algorithm. 4. Particle swarm

# Sumário:

<b>Listas de Figuras</b>	xiii
<b>Lista de Tabelas</b>	xviii
<b>Lista de Símbolos</b>	xx

## 1 Introdução

1.1 Considerações gerais	01
1.2 Motivação	04
1.3 Objetivos	04
1.4 Metodologia	05
1.5 Organização da dissertação	06

## 2 Otimização

2.1 Introdução	08
2.2 Elementos para formulação de um problema de otimização	09
2.2.1 Variáveis de projeto	09
2.2.2 Função objetivo	10
2.2.3 Restrições	11
2.2.4 Formulação padrão	12
2.2.5 Normalização do problema de otimização	12
2.3 Programação matemática	13
2.3.1 Programação sequencial quadrática	14
2.4 Metaheurística	17
2.5 Otimização multiobjetivo e dominância	19
2.5.1 Otimização multiobjetivo	19
2.5.2 Dominância	20
2.6 Superfície de <i>tradeoff</i> ou rente de Pareto	21
2.7 Convexidade	22
2.8 Métodos para geração de pontos de Pareto	23
2.8.1 Método da soma ponderada dos objetivos ( <i>Weighted Sum Method</i> – WS)	23

2.8.2 Método da intersecção contorno-normal ( <i>Normal-Boundary Intersection</i> – NBI) . . . . .	24
2.8.3 Método da restrição normal normalizada ( <i>Normalized Normal-Constraints</i> – NNC) . . . . .	27
<b>3 Algoritmos evolucionários</b>	
3.1 Introdução . . . . .	30
3.2 Visão geral, classificação das estratégias . . . . .	30
3.2.1 Função aptidão . . . . .	33
3.2.2 Programação evolucionária . . . . .	33
3.2.3 Estratégia evolutiva . . . . .	34
3.2.4 Algoritmo genético . . . . .	35
3.2.5 Programação genética . . . . .	36
3.2.6 <i>Swarm Intelligence</i> . . . . .	37
3.2.6.1 Enxame de partículas ( <i>Particle Swarm</i> – PS) . . . . .	37
3.3 Algoritmos evolucionários em problemas multiobjetivos . . . . .	38
3.3.1 Funções agregadas . . . . .	39
3.3.2 Abordagens populacionais. . . . .	39
3.3.3 Abordagem de Pareto . . . . .	40
<b>4 Otimização via enxame de partículas (<i>Particle Swarm Optimization</i> – PSO)</b>	
4.1 Introdução . . . . .	43
4.2 Otimização via enxame de partículas . . . . .	44
4.2.1 Idéia geral . . . . .	44
4.2.2 Algoritmo básico. . . . .	45
4.2.3 Distribuição inicial . . . . .	46
4.2.4 Atualização das variáveis . . . . .	47
4.2.5 Atualização de parâmetros . . . . .	48
4.2.6 Consideração de restrições no problema . . . . .	51
4.2.7 Partículas com restrições violadas . . . . .	51
4.2.8 Função aptidão considerada . . . . .	52
4.2.9 Critério de convergência . . . . .	53
4.2.9.1 Condições de finalização . . . . .	54

4.2.9.1.1. Primeira condição de finalização . . . . .	54
4.2.9.1.2. Segunda condição de finalização . . . . .	55
4.3 Otimização de múltiplos objetivos com o PSO . . . . .	56
4.3.1 Técnicas adotadas para otimização multiobjetivo via enxame de partículas . . . .	57
4.3.1.1 Agregação evolucionária com ponderação dinâmica ( <i>Evolutionary Dynamic Weighted Aggregation – EDWA</i> ) . . . . .	57
4.3.1.2 Otimização via enxame de partículas com avaliação vetorial ( <i>Vector Evaluated Particle Swarm Optimization – VEPSO</i> ) . . . . .	61
4.3.1.3 Otimização multiobjetivo via enxame de partículas considerando o critério de dominância ( <i>Multiobjective Particle Swarm Optimization – MOPSO</i> ) . .	63
4.3.2 Critério de convergência na otimização de múltiplos objetivos . . . . .	65
4.4 Melhoria de aficiência computacional . . . . .	65
4.4.1 Método das bases reduzidas . . . . .	66
4.4.1.1 Observação crítica . . . . .	66
4.4.1.2 Preliminares . . . . .	68
4.4.1.3 Aproximação . . . . .	69
4.4.1.4 Precisão . . . . .	72
4.4.1.5 Algoritmo computacional . . . . .	73
4.4.2 Programação paralela . . . . .	74
4.4.2.1 Otimização via enxame de partículas em programação paralela . . . . .	74
4.4.2.2 Inicialização e atualização das variáveis . . . . .	75
4.4.2.3 Verificação da convergência . . . . .	75
<b>5 Exemplos</b>	
5.1 Introdução . . . . .	77
5.2 Problemas de otimização considerando funções analíticas . . . . .	77
5.2.1 Otimização uni-objetivo . . . . .	78
5.2.1.1 Exemplo 1: função de Rastrigin . . . . .	78
5.2.1.2 Exemplo 2: função não homogênea . . . . .	84
5.2.2 Otimização multiobjetivo . . . . .	89
5.2.2.1 Exemplo 1: problema MO1 . . . . .	90
5.2.2.2 Exemplo 2: problema MO2 . . . . .	93
5.2.2.3 Exemplo 3: problema MO3 . . . . .	96

5.2.2.4 Exemplo 4: problema MO4 .....	99
5.2.2.5 Exemplo 5: problema MO5 .....	102
5.2.2.6 Exemplo 6: problema MO6 .....	106
5.3 Problemas de otimização estrutural .....	108
5.3.1 Otimização uni-objetivo .....	109
5.3.1.1 Exemplo 1: treliça com 10 barras .....	109
5.3.1.2 Exemplo 2: treliça com 200 barras .....	113
5.3.1.3 Exemplo 3: treliça com 940 barras .....	118
5.3.2 Otimização multiobjetivo .....	121
5.3.2.1 Exemplo: treliça de três barras .....	122
 <b>6 Conclusões</b>	
6.1 Realizações .....	126
6.2 Conclusões gerais .....	128
6.3 Sugestões para trabalhos futuros .....	129
 <b>Bibliografia</b> .....	B.1

# Lista de Figuras:

<b>Figura (2.1)</b>	Nível de preferência e relação de dominância.....	21
<b>Figura (2.2)</b>	Representação da superfície de <i>tradeoff</i> .....	22
<b>Figura (2.3a)</b>	Exemplo de conjunto convexo.....	23
<b>Figura (2.3b)</b>	Exemplo de conjunto não-convexo.....	23
<b>Figura (2.4)</b>	Conjunto viável sobre o mapeamento de $f$ no espaço das funções objetivo.....	26
<b>Figura (2.5)</b>	Representação gráfica do NNC para um problema com dois objetivos..	26
<b>Figura (4.1a)</b>	Frente de Pareto convexa com o sistema rotacionado em $0^\circ$ .....	58
<b>Figura (4.1b)</b>	Frente de Pareto convexa com o sistema rotacionado em $45^\circ$ .....	58
<b>Figura (4.1c)</b>	Frente de Pareto convexa com o sistema rotacionado em $90^\circ$ .....	58
<b>Figura (4.2a)</b>	Frente de Pareto não-convexa com o sistema rotacionado em $0^\circ$ .....	59
<b>Figura (4.2b)</b>	Frente de Pareto não-convexa com o sistema rotacionado em $45^\circ$ .....	59
<b>Figura (4.2c)</b>	Frente de Pareto não-convexa com o sistema rotacionado em $90^\circ$ .....	59
<b>Figura (4.3a)</b>	Menor caminho viável entre duas soluções de Pareto quando a frente de Pareto é convexa.....	60
<b>Figura (4.3b)</b>	Menor caminho viável entre duas soluções de Pareto quando a frente de Pareto é não-convexa.....	60
<b>Figura (4.4a)</b>	Representação do espaço de soluções considerando a variação da solução com o parâmetro $\mu$ .....	67
<b>Figura (4.4b)</b>	Representação do espaço de soluções considerando a aproximação da solução em $\mu^{novo}$ através de uma combinação linear de soluções $u(\mu^i)$ pré-calculadas.....	67
<b>Figura (5.1)</b>	Esboço do comportamento da função de Rastrigrin.....	79
<b>Figura (5.2a)</b>	Gráficos mostrando o ponto inicial e o ponto ótimo resultado da otimização via PSO.....	80
<b>Figura (5.2b)</b>	Gráficos mostrando o ponto inicial e o ponto ótimo resultado da otimização via SQP.....	80
<b>Figura (5.3)</b>	Esboço do comportamento da função não homogênea considerada no exemplo 2.....	85

<b>Figura (5.4a)</b>	Vista X-Y do comportamento da função não homogênea considerada no exemplo 2.....	85
<b>Figura (5.4b)</b>	Vista X-Z do comportamento da função não homogênea considerada no exemplo 2.....	85
<b>Figura (5.5a)</b>	Gráficos mostrando o ponto inicial e o ponto ótimo resultado da otimização via PSO.....	87
<b>Figura (5.5b)</b>	Gráficos mostrando o ponto inicial e o ponto ótimo resultado da otimização via SQP.....	87
<b>Figura (5.6a)</b>	Frentes de Pareto encontradas no problema MO1 via EDWA comparada à encontrada via NNC.....	91
<b>Figura (5.6b)</b>	Frentes de Pareto encontradas no problema MO1 via VEPSO comparada à encontrada via NNC.....	91
<b>Figura (5.6c)</b>	Frentes de Pareto encontradas no problema MO1 via MOPSO comparada à encontrada via NNC.....	91
<b>Figura (5.7a)</b>	Frente de Pareto encontrada no problema MO1, em novos testes, via EDWA comparada à encontrada via NNC, considerando dez variáveis em cada função.....	93
<b>Figura (5.7b)</b>	Frente de Pareto encontrada no problema MO1, em novos testes, via VEPSO comparada à encontrada via NNC, considerando dez variáveis em cada função.....	93
<b>Figura (5.7c)</b>	Frente de Pareto encontrada no problema MO1, em novos testes, via MOPSO comparada à encontrada via NNC, considerando dez variáveis em cada função.....	93
<b>Figura (5.8a)</b>	Frentes de Pareto encontradas no problema MO2 via EDWA comparada à encontrada via NNC.....	94
<b>Figura (5.8b)</b>	Frentes de Pareto encontradas no problema MO2 via VEPSO comparada à encontrada via NNC.....	94
<b>Figura (5.8c)</b>	Frentes de Pareto encontradas no problema MO2 via MOPSO comparada à encontrada via NNC.....	94
<b>Figura (5.9a)</b>	Frente de Pareto encontrada no problema MO2, em novos testes, via EDWA comparada à encontrada via NNC, considerando dez variáveis em cada função.....	96

<b>Figura (5.9b)</b>	Frente de Pareto encontrada no problema MO2, em novos testes, via VEPSO comparada à encontrada via NNC, considerando dez variáveis em cada função.....	96
<b>Figura (5.9c)</b>	Frente de Pareto encontrada no problema MO2, em novos testes, via MOPSO comparada à encontrada via NNC, considerando dez variáveis em cada função.....	96
<b>Figura (5.10a)</b>	Frentes de Pareto encontradas no problema MO3 via EDWA comparada à encontrada via NNC.....	97
<b>Figura (5.10b)</b>	Frentes de Pareto encontradas no problema MO3 via VEPSO comparada à encontrada via NNC.....	97
<b>Figura (5.10c)</b>	Frentes de Pareto encontradas no problema MO3 via MOPSO comparada à encontrada via NNC.....	97
<b>Figura (5.11a)</b>	Frente de Pareto encontrada no problema MO3, em novos testes, via EDWA comparada à encontrada via NNC, considerando dez variáveis em cada função.....	99
<b>Figura (5.11b)</b>	Frente de Pareto encontrada no problema MO3, em novos testes, via VEPSO comparada à encontrada via NNC, considerando dez variáveis em cada função.....	99
<b>Figura (5.11c)</b>	Frente de Pareto encontrada no problema MO3, em novos testes, via MOPSO comparada à encontrada via NNC, considerando dez variáveis em cada função.....	99
<b>Figura (5.12a)</b>	Frentes de Pareto encontradas no problema MO4 via EDWA comparada à encontrada via NNC.....	100
<b>Figura (5.12b)</b>	Frentes de Pareto encontradas no problema MO4 via VEPSO comparada à encontrada via NNC.....	100
<b>Figura (5.12c)</b>	Frentes de Pareto encontradas no problema MO4 via MOPSO comparada à encontrada via NNC.....	100
<b>Figura (5.13a)</b>	Frente de Pareto encontrada no problema MO4, em novos testes, via EDWA comparada à encontrada via NNC, considerando dez variáveis em cada função.....	102
<b>Figura (5.13b)</b>	Frente de Pareto encontrada no problema MO4, em novos testes, via VEPSO comparada à encontrada via NNC, considerando dez variáveis em cada função.....	102



<b>Figura (5.13c)</b>	Frente de Pareto encontrada no problema MO4, em novos testes, via MOPSO comparada à encontrada via NNC, considerando dez variáveis em cada função.....	102
<b>Figura (5.14a)</b>	Frentes de Pareto encontradas no problema MO5 via EDWA comparada à encontrada via NNC.....	103
<b>Figura (5.14b)</b>	Frentes de Pareto encontradas no problema MO5 via VEPSO comparada à encontrada via NNC.....	103
<b>Figura (5.14c)</b>	Frentes de Pareto encontradas no problema MO5 via MOPSO comparada à encontrada via NNC.....	103
<b>Figura (5.15a)</b>	Frente de Pareto encontrada no problema MO5, em novos testes, via EDWA comparada à encontrada via NNC, considerando dez variáveis em cada função.....	105
<b>Figura (5.15b)</b>	Frente de Pareto encontrada no problema MO5, em novos testes, via VEPSO comparada à encontrada via NNC, considerando dez variáveis em cada função.....	105
<b>Figura (5.15c)</b>	Frente de Pareto encontrada no problema MO5, em novos testes, via MOPSO comparada à encontrada via NNC, considerando dez variáveis em cada função.....	105
<b>Figura (5.16a)</b>	Frentes de Pareto encontradas no problema MO6 via EDWA.....	107
<b>Figura (5.16b)</b>	Frentes de Pareto encontradas no problema MO6 via VEPSO.....	107
<b>Figura (5.16c)</b>	Frentes de Pareto encontradas no problema MO6 via MOPSO.....	107
<b>Figura (5.17)</b>	Treliça de dez barras.....	110
<b>Figura (5.18)</b>	Treliça de duzentas barras.....	114
<b>Figura (5.19)</b>	Gráfico da relação entre o número de amostras fornecidas e a convergência dos resultados encontrados pelo OPT_PSRbm para os resultados encontrados pelo OPT_PS.....	116
<b>Figura (5.20)</b>	Treliça de 940 barras.....	118
<b>Figura (5.21)</b>	Treliça de três barras.....	121
<b>Figura (5.22a)</b>	Frentes de Pareto encontradas para o problema estrutural MO via EDWA comparada à encontrada via NNC.....	123
<b>Figura (5.22b)</b>	Frentes de Pareto encontradas para o problema estrutural MO via VEPSO comparada à encontrada via NNC.....	123

<b>Figura (5.22c)</b>	Frentes de Pareto encontradas para o problema estrutural MO via MOPSO comparada à encontrada via NNC.....	123
-----------------------	--	-----

# Lista de Tabelas:

<b>Tabela (3.1)</b>	Algoritmo geral de um AE.....	32
<b>Tabela (3.2)</b>	Algoritmo da Programação Evolucionária.....	34
<b>Tabela (3.3)</b>	Algoritmo da Estratégia Evolutiva.....	35
<b>Tabela (3.4)</b>	Algoritmo Genético Simples.....	36
<b>Tabela (3.5)</b>	Algoritmo Generalizado da Programação Genética.....	37
<b>Tabela (4.1)</b>	Algoritmo do RBM: estágios <i>off-line/on-line</i> .....	73
<b>Tabela (5.1)</b>	Resultados da otimização da função de Rastrigin com o PSO e com o SQP.....	79
<b>Tabela (5.2)</b>	Resultados da otimização da função de Rastrigin modificando o número de dimensões do problema.....	81
<b>Tabela (5.3)</b>	Resultados da otimização da função de Rastrigin considerando diferentes valores fixos de inércia.....	82
<b>Tabela (5.4)</b>	Resultados da otimização da função de Rastrigin considerando variações dinâmicas nos parâmetros de confiança.....	83
<b>Tabela (5.5)</b>	Resultados da otimização da função de Rastrigin considerando diferentes valores fixos para os parâmetros de confiança.....	83
<b>Tabela (5.6)</b>	Resultados da otimização da função do exemplo 2 com o PSO e com o SQP.....	86
<b>Tabela (5.7)</b>	Resultados da otimização da função do exemplo 2 considerando diferentes valores fixos de inércia.....	88
<b>Tabela (5.8)</b>	Resultados da otimização considerando variações dinâmicas nos parâmetros de confiança.....	89
<b>Tabela (5.9)</b>	Resultados da otimização considerando diferentes valores fixos para os parâmetros de confiança.....	89
<b>Tabela (5.10)</b>	Número médio de soluções de Pareto encontradas por cada metodologia considerada – exemplo MO1.....	92
<b>Tabela (5.11)</b>	Número médio de soluções de Pareto encontradas por cada metodologia considerada – exemplo MO2.....	95

<b>Tabela (5.12)</b>	Número médio de soluções de Pareto encontradas por cada metodologia considerada – exemplo MO3.....	98
<b>Tabela (5.13)</b>	Número médio de soluções de Pareto encontradas por cada metodologia considerada – exemplo MO4.....	101
<b>Tabela (5.14)</b>	Número médio de soluções de Pareto encontradas por cada metodologia considerada – exemplo MO5.....	104
<b>Tabela (5.15)</b>	Número médio de soluções de Pareto encontradas por cada metodologia considerada – exemplo MO6.....	107
<b>Tabela (5.16)</b>	Resultados da otimização da treliça de 10 barras com o OPT_PS e com o OPTRUSS.....	111
<b>Tabela (5.17)</b>	Resultados da otimização da treliça de 10 barras considerando diferentes valores fixos de inércia.....	112
<b>Tabela (5.18)</b>	Resultados da otimização considerando variações dinâmicas nos parâmetros de confiança.....	113
<b>Tabela (5.19)</b>	Resultados da otimização considerando variações dinâmicas nos parâmetros de confiança.....	113
<b>Tabela (5.20)</b>	Resultados da otimização da treliça de 200 barras com o OPT_PS e com OPTRUSS.....	115
<b>Tabela (5.21)</b>	Resultados da otimização da treliça de 200 barras com o OPT_PS e com o OPT_PSRbm.....	116
<b>Tabela (5.22)</b>	Amostragem utilizada para a otimização do exemplo utilizando o OPT_PSRbm.....	117
<b>Tabela (5.23)</b>	Resultados da otimização da treliça de 200 barras com o OPT_PS e com o OPT_PSparr .....	118
<b>Tabela (5.24)</b>	Resultados da otimização da treliça de 940 barras com o OPT_PS e com o OPTRUSS.....	120
<b>Tabela (5.25)</b>	Resultados da otimização da treliça de 940 barras com o OPT_PS e com o OPT_PSRbm.....	121
<b>Tabela (5.26)</b>	Número médio de soluções de Pareto encontradas por cada metodologia considerada – exemplo da treliça de três barras.....	123

# Lista de Símbolos

## *Abreviações:*

AE	Algoritmos Evolucionários
BFGS	Método Broyden-Fletcher-Goldfarb-Shanno
CO	Critério de Otimalidade
$COV$	Coeficiente de Variação
CWA	Conventional Weighted Aggregation
ECMI	Envoltória Convexa do Mínimo Individual
EDWA	Evolutionary Dynamic Weighted Aggregation
EMO	Evolutionary Multiobjective Optimization
EP	Evolutionary Programming
ES	Evolutionary Strategies
FEM	Finite Element Method
GA	Genetic Algorithm
GP	Genetic Programming
LU	Linha Utópica
<i>Mean</i>	Média
MMA	Method of Moving Asymptotes
MOGA	Multiobjective Genetic Algorithm
MOPSO	Multiobjective Particle Swarm Optimization
MPI	Message Passing Interface
NBI	Normal-Boundary Intersection
$NBI_{\beta}$	Subproblema NBI relativo ao conjunto de pesos $\beta$
NC	Normal Constraints
NNC	Normalized-Normal Constraints
NU	Normal Utópica
NPGA	Niched-Pareto Genetic Algorithm
NPGA 2	Niched-Pareto Genetic Algorithm 2
NSGA	Non-dominated Sorting Genetic Algorithm

NSGA-II	Non-dominated Sorting Genetic Algorithm II
PAES	Pareto Archived Evolutionary Strategy
PE	Programação Evolucionária
PESA	Pareto Envelope-based Selection Algorithm
PM	Programação Matemática
PS	Particle Swarm
PSO	Particle Swarm Optimization
RBM	Reduced Basis Method
SLP	Sequential Linear Programming
SPEA	Strength Pareto Evolutionary Algorithm
SPEA2	Strength Pareto Evolutionary Algorithm 2
SQP	Sequential Quadratic Programming
<i>StdDev</i>	Desvio Padrão
VEGA	Vector Evaluated Genetic Algorithm
VEPSO	Vector Evaluated Particle Swarm Optimization
WS	Weight Sum Method
<i>cos</i>	Função cosseno
<i>nelem</i>	Número de elementos
<i>nobj</i>	Número de objetivos
<i>np</i>	Número de partículas
<i>rank</i>	Ranque de Pareto
<i>sen</i>	Função seno
<i>tg</i>	Função tangente
<i>vol</i>	Volume

### ***Romanos:***

#### **Escalares:**

<i>A</i>	Área da secção transversal
<i>E</i>	Módulo de elasticidade
<i>F</i>	Força aplicada
	Frequência

$L$	Comprimento da barra
$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda})$	Função Lagrangeana
$M$	Número de populações
$N$	Número de regiões do sub-domínio
$P(\mathbf{x})$	Função aptidão Função objetivo penalizada
$R$	Sub-regiões do domínio
$S$	População
$a$	Fator de aceleração
$c$	Parâmetro de confiança
$\bar{c}$	Valor médio das aptidões da população
$cg_{best}$	Melhor aptidão encontrada na população
$cg_{worst}$	Pior aptidão encontrada na população
$f$	Constante para redução da inércia
$f(\mathbf{x})$	Função objetivo
$g(\mathbf{x})$	Função de restrição de desigualdade
$h(\mathbf{x})$	Função de restrição de igualdade
$i, j$	Índices das coordenadas dos vetores
$k$	Contador de iterações Índice das coordenadas dos vetores
$n_{dom}$	Número de soluções dominadoras
$r$	Valor randômico
$s$	Energia de deformação
$t$	Contador de iterações
$t_{max}$	Valor máximo de iterações permitidas
$w$	Inércia da partícula Pesos agregados
$x$	Variável de projeto
$x^l$	Limite inferior das variáveis de projeto
$x^u$	Limite superior das variáveis de projeto

**Vetores:**

$e$	Erro
$\mathbf{F}$	Vetor de forças aplicadas
$\mathbf{F}(\mathbf{x})$	Vetor de funções objetivos
$\mathbf{f}$	Vetor de esforços no elemento
$\mathbf{g}(\mathbf{x})$	Vetor de funções de restrição de desigualdade
$\mathbf{h}(\mathbf{x})$	Vetor de funções de restrição de igualdade
$\hat{n}$	Versor normal
$\mathbf{p}$	Vetor da posição da partícula com melhor aptidão
$\mathbf{R}$	Vetor de resíduos
$\mathbf{r}$	Direção de busca
$\mathbf{u}$	Vetor de deslocamentos
$\mathbf{v}$	Vetor velocidade da partícula
$\mathbf{x}$	Vetor da posição da partícula
	Vetor de variáveis de projeto
$\mathbf{x}^l$	Vetor de limites inferiores das variáveis de projeto
$\mathbf{x}^u$	Vetor de limites superiores das variáveis de projeto

**Matrizes:**

$\mathbf{C}(\boldsymbol{\mu})$	Operador simétrico
$\mathbf{H}(\mathbf{x})$	Matriz Hessiana
$\mathbf{K}(\boldsymbol{\mu}), \mathbf{K}$	Matriz de rigidez
$\mathbf{S}(\boldsymbol{\mu})$	Matriz de amostras de variáveis
$\mathbf{Z}(\boldsymbol{\mu})$	Matriz de amostras de soluções



## ***Gregos:***

### **Escalares:**

$\alpha$	Coefficiente linear Peso da função $f$ na função substituta Tamanho do passo na direção de busca
$\mu$	Média Parâmetro de penalidade Variáveis de projeto
$\sigma$	Desvio padrão Tensão
$\chi$	Fator de constrição

### **Vetores:**

$\alpha$	Vetor de coeficientes lineares
$\lambda$	Multiplicadores de Lagrange
$\mu$	Vetor das variáveis de projeto
$\zeta$	Amostra de solução

### **Matrizes:**

$\Phi$	Matriz auxiliar na definição dos pontos que compõem a ECMI
--------	--

# Introdução

## 1.1 Considerações gerais

Nos dias atuais é imprescindível um planejamento adequado das atividades nos diversos campos de atuação do homem, é necessário haver a determinação da meta a ser alcançada, a investigação das opções disponíveis para se alcançar tal meta e a observação das especificações a serem seguidas.

O termo “otimização” tem sido bastante empregado ultimamente em vários âmbitos. Até mesmo na vida cotidiana, se faz uso do conceito de otimização quando se deseja, por exemplo, realizar tarefas diárias no menor tempo possível. A natureza induz a certa harmonia para que tudo isso seja feito. O processo de otimização pode ser definido consistindo basicamente no melhor aproveitamento possível dos recursos disponíveis.

Nas engenharias, a otimização desperta bastante interesse principalmente nos procedimentos de otimização de forma que consistem numa abordagem geral para projetar estruturas dos mais variados campos tais como Civil, Mecânica, Aeroespacial, Naval, etc., através da variação dos parâmetros de forma e/ou espessura que definam a estrutura. Neste sentido, um dado projeto inicial é melhorado com relação a um conjunto de funções, objetivos e restrições pré-definidas.

Fazendo uso da linguagem matemática, três perguntas podem ser formuladas da seguinte maneira:

- a) Qual a função objetivo?
- b) Que variáveis estão relacionadas àquela função?

c) Há alguma restrição a ser satisfeita?

Os estudos de técnicas de otimização que permitem chegar à solução de problemas como o imposto acima é datado do final do século XIX e início do século XX com Maxwell e Mitchell atuando no campo da otimização analítica (KIRSCH, 1993). Pode-se considerar essa época como sendo o início da busca para soluções dos problemas de otimização estrutural.

Durante muitas décadas a otimização estrutural pouco evoluiu e somente a partir da década de 40, e mais intensamente na década de 50, houve um impulso maior. Isto foi possível devido ao forte investimento no programa aeroespacial, o qual tinha dentre os seus objetivos a redução do peso estrutural sem que houvesse o comprometimento da integridade estrutural das aeronaves. Foi também na década de 50 que surgiram os primeiros computadores digitais, o qual possibilitou ao projetista o emprego do Método dos Elementos Finitos (*Finite Element Method* – FEM) (COOK, 1981; ZIENKIEWICZ e TAYLOR, 2000), que é uma poderosa ferramenta na análise de estruturas complexas.

Para Vanderplaats (VANDERPLAATS, 1993), foi Schmidt, na década de 60, quem apresentou de forma compreensiva o uso de técnicas de Programação Matemática (PM) para resolver problemas com restrição não-linear de desigualdade, para estruturas no regime elástico submetidas a múltiplas condições de carregamento. Nesta mesma década surgiram outras técnicas que também faziam uso da PM. Mesmo com essa evolução, o número de variáveis de projeto que se conseguia trabalhar era pequeno, limitando assim o uso apenas a estruturas de pequeno porte.

Em contrapartida à técnica de PM, no final da década de 60 surgiu uma técnica chamada de Critério de Otimalidade (CO), a qual foi apresentada em sua forma analítica (PRAGER e SHIELD, 1968; PRAGER e TAYLOR, 1968) e na forma numérica (VENKAYYA et al., 1968). Esta técnica apresenta como mérito principal a facilidade de implementação computacional e a relativa independência do tamanho do problema.

Nas décadas de 70 e 80, o aperfeiçoamento das técnicas de PM e a dos CO aplicadas à otimização estrutural se desenvolveu bastante. A formulação do Método Dual foi interpretada como uma generalização do método CO e foi apresentada como a base para a união dos dois métodos (KIRSCH, 1993).

Todos esses desenvolvimentos, aliados ao crescimento exponencial dos recursos computacionais, e procedimentos numéricos tornaram possível a utilização de procedimentos de otimização para solução de problemas de engenharia prática. Tais procedimentos constituem hoje uma das mais modernas e poderosas tecnologias computacionais aplicadas à engenharia, envolvendo modelos geométricos e numéricos e algoritmos de otimização. Em decorrência dos desenvolvimentos atingidos nesta área, uma grande variedade de códigos computacionais (acadêmicos e comerciais) tem sido desenvolvida. Como consequência natural, o tamanho e complexidade dos problemas que vem sendo resolvidos com ferramentas de otimização expandiram-se devido aos avanços acima descritos (AFONSO, 1995).

A maioria dos procedimentos de otimização utiliza algoritmos que fazem uso do cálculo de gradientes devido principalmente à eficiência computacional destes no processo de obtenção do ponto de ótimo. No entanto, nas últimas décadas, algoritmos metaheurísticos, que não requerem cálculo de gradientes, têm atraído grande atenção de comunidade científica (MICHALEWICZ, 1997).

Os algoritmos metaheurísticos descendem de uma classe de algoritmos conhecida como Programação Evolucionária (PE). A PE é um ramo da ciência da computação que tem por base os mecanismos evolutivos encontrados na natureza. Esses mecanismos estão diretamente relacionados com a teoria da evolução de Darwin, onde o mesmo afirma que a vida na Terra é o resultado de um processo de seleção, feito pelo ambiente, em que somente os mais aptos e adaptados possuem chances de sobreviver e, conseqüentemente, reproduzir-se.

Os primeiros passos dados na área da PE foram dados pelos biólogos e geneticistas que estavam interessados em simular os processos vitais do ser humano. Na década de 60, um grupo de cientistas iniciou um estudo no qual fora implementada uma população de  $n$ -indivíduos onde cada um possuía um genótipo e estava sujeito a operações de seleção, de cruzamento e mutação. Tal estudo foi modelado e passou a ser conhecido como algoritmo genético. Os estudos relacionados à PE não cessaram. Outras idéias surgiram e ganharam importância no meio científico. Novas concepções de meta-heurística emergiram e todas diretamente relacionadas com a PE. Podem ser citados como exemplos os algoritmos de Formações de Cristais, o da Colônia de Formigas e o do Enxame de Partículas. Tais propostas também têm uma forte motivação física ou biológica baseados em componentes da natureza tais como um bando de pássaros, cardumes e enxames entre outros.

Embora tais técnicas requeiram muito mais avaliações de funções para encontrar uma solução otimizada, quando comparados a algoritmos baseados no cálculo de gradientes, algoritmos baseados em procedimentos evolucionários apresentam várias vantagens, a saber: facilidade de programação; não necessitam da garantia de continuidade nas funções envolvidas na definição do problema; são mais adequados na determinação de um ótimo global ou próximo ao global; e adequados para solucionar problemas discretos.

## **1.2 Motivação**

A principal motivação deste trabalho é a implementação de uma metodologia evolucionária para problemas de otimização, que possa ser utilizada nos mais diversos campos da engenharia e das mais variadas configurações possíveis. O interesse no desenvolvimento de tal ferramenta se deve às dificuldades encontradas pela maioria dos algoritmos convencionais quando se deparam com problemas de otimização com algumas particularidades, como funções descontínuas, por exemplo.

## **1.3 Objetivos**

O objetivo principal deste trabalho é a realização de estudos para validação das metodologias evolucionárias utilizando o método do enxame de partículas (*Particle Swarm – PS*) (VENTER e SOBIESZCZANSKI-SOBIESKI, 2002) como ferramenta para otimização de problemas da engenharia. Os testes são aplicados em problemas puramente analíticos e a treliças planas, porém várias técnicas foram abordadas levando em consideração a complexidade do problema. As respostas obtidas para os problemas propostos foram obtidas por meio de implementação computacional, desenvolvida no ambiente MATLAB® (LITTLEFIELD, 1999).

Os principais aspectos relacionados ao desenvolvimento deste trabalho estão listados a seguir:

- Apresentar os principais aspectos da otimização evolucionária;
- Apresentar os principais aspectos e definições das considerações feitas sobre a metodologia selecionada como principal foco, a metodologia do enxame de partículas;
- Desenvolver e implementar o algoritmo da metodologia selecionada;

- Validar as novas implementações através de exemplos padrões encontradas na literatura especializada;
- Implementar o algoritmo desenvolvido no contexto da otimização multiobjetivo;
- Implementar novas técnicas visando à diversificação de aplicação e melhoria do desempenho computacional do algoritmo;
- Realizar estudos comparativos entre os métodos implementados e os métodos tradicionais através de exemplos;
- Apresentar orientações sobre escolha de parâmetros, técnicas multiobjetivo mais adequada para o emprego da ferramenta desenvolvida para outras aplicações da engenharia.

#### **1.4 Metodologia**

Os processos de otimização aplicados a problemas da engenharia em geral utilizam um procedimento numérico e necessitam do uso sequencial de simuladores numéricos e da análise de sensibilidade. Os algoritmos evolucionários conseguem driblar os inconvenientes que surgem com tais considerações, pois não necessitam de análises de sensibilidade, que é um procedimento de custo computacional elevado.

O PSO foi estudado e implementado seguindo o roteiro apresentado no capítulo quatro, onde é dada uma descrição detalhada de todas as considerações feitas. A formulação do PSO visava inicialmente o emprego em alguns determinados tipos de problemas, porém com as modificações que foram surgindo ao longo dos anos com o desenvolvimento de pesquisas na área, as limitações foram eliminadas aos poucos.

Com a utilização das técnicas estudadas, foram criadas algumas versões do algoritmo que se adequam a cada tipo de problema. Primeiramente foi desenvolvido o código PSO que visava à otimização de problemas irrestritos de funções analíticas.

Com a verificação da eficácia do algoritmo, a utilização foi estendida a consideração de problemas estruturais (treliças planas) restritos através de algumas modificações no código OPRUSS (AFONSO e HOROWITZ, 1998) criando assim o código computacional OPT\_PS.

Posteriormente, foram feitas algumas investidas em estudos de técnicas visando à melhoria do desempenho computacional, surgindo assim três novas versões dos algoritmos citados, o Parallel\_PSO e Parallel\_OPT\_PS, que utilizam os paradigmas da computação paralela no processo de otimização, usados respectivamente para problemas analíticos e problemas estruturais, e o OPT\_PSRbm, que utiliza uma técnica de modelo substituto, o método das bases reduzidas, para obtenção das múltiplas avaliações das funções requeridas no processo de otimização estrutural (AFONSO e PATERA, 2003).

Por fim, estudos e implementação de técnicas para resolução de problemas de otimização mais próximos da realidade, onde é feita a otimização de múltiplos objetivos simultaneamente foram conduzidas, dando origem ao código PSO\_MO onde estão implementadas as três técnicas distintas, o EDWA (JIN et al., 2001), o VEPSO (SCHAFER, 1984; COELLO e SIERRA, 2004) e o MOPSO (COELLO et al., 2004), aplicados a problemas analíticos e problemas estruturais.

A formulação dos problemas de otimização, especificação das funções objetivo e restrições, variam de acordo com a aplicação em questão. No entanto, todos os problemas de otimização podem ser esquematicamente representados através de uma formulação matemática, como descrita no capítulo dois.

## **1.5 Organização da dissertação**

Esta dissertação consiste em seis capítulos organizados de maneira a facilitar o entendimento dos estudos realizados. A descrição do conteúdo de cada capítulo é dada a seguir.

Após breve introdução vista neste capítulo é apresentada no capítulo dois uma definição mais detalhada do procedimento de otimização. No capítulo dois, as diversas variedades de problemas de otimização são definidas bem como os métodos utilizados na resolução dos mesmos. Um breve histórico sobre o desenvolvimento dos processos de otimização utilizando métodos numéricos é mostrado. Uma definição mais detalhada sobre os métodos mais conhecidos também é apresentada no capítulo dois.

Após a apresentação de algumas metodologias existentes para resolução de problemas de otimização, no capítulo três é dada atenção às metodologias evolucionárias. Uma visão geral das estratégias existentes e as vantagens de suas aplicações também são apresentadas no capítulo.

No capítulo quatro, a metodologia do enxame de partículas, foco principal deste trabalho, é apresentada detalhadamente, desde sua concepção até as técnicas adotadas passando pelas definições e implementações realizadas.

O capítulo cinco trata dos exemplos considerados para aplicação e validação do algoritmo de otimização via enxame de partículas. Os problemas apresentados foram selecionados para demonstrar a eficácia do algoritmo e nos mesmos também foram realizados testes para verificar a influência dos parâmetros que configuram o algoritmo na obtenção dos resultados.

No capítulo seis, último capítulo deste trabalho, as conclusões obtidas com os testes realizados são apresentadas juntamente com as sugestões para trabalhos futuros.

No Apêndice são descritas as implementações realizadas nas diversas versões do código PSO e também é explicado o funcionamento de cada módulo do programa.



# Otimização

## 2.1 Introdução

Nas ciências em geral, o termo otimização, se refere ao estudo de conjunto de técnicas que têm como objetivo a obtenção de um melhor resultado para uma função e parâmetros (variáveis de projeto) pré-especificados dentro de um conjunto permitido (espaço de projeto).

O ser humano, guiado e influenciado pelos cercos naturais, desempenha, quase que instintivamente, todas as funções de um modo que economize energia ou minimize o desconforto e a dor. A motivação é tirar proveito de recursos disponíveis, porém limitados, de maneira a maximizar a produção ou lucro (HAFTKA, e GÜRDAL, 1993).

Normalmente, em vários problemas de engenharia, um grande número de variáveis está envolvido. Cabe ao projetista encontrar uma combinação para estas variáveis que resulte num projeto mais eficiente e idealmente o mais econômico possível. A determinação desta melhor configuração dos parâmetros do projeto, muitas vezes depende da experiência do projetista, porém, nem sempre é possível obtê-la intuitivamente, em função da ausência de uma base de conhecimentos físicos sobre um dado problema específico para justificar a sua intuição sobre o porquê de se utilizar àqueles dados.

No mundo da engenharia estrutural, um bom projeto requer eficiência no tempo de execução e nos vários custos envolvidos. Além disso, este deve atingir uma forma aceitável do ponto de vista da execução e satisfazer as restrições de projeto impostas. Muitas vezes, para se chegar à forma ideal, é necessário se avaliar as várias possibilidades de combinação dos parâmetros do projeto. Esse procedimento iterativo envolve vários processos até se encontrar a combinação ótima. A otimização é utilizada para auxiliar o projetista na determinação da solução para esses problemas, de acordo com os critérios estabelecidos.

A composição de problemas de otimização e suas diversas características e classificações serão discutidas neste capítulo. Algumas metodologias conhecidas, utilizadas para resolução de tais problemas, também serão apresentadas.

## **2.2 Elementos para formulação de um problema de otimização**

### **2.2.1 Variáveis de projeto**

A idéia de melhorar ou otimizar uma estrutura implica em alguma liberdade para modificá-la e assim obter um melhor desempenho da mesma. O potencial de mudanças é tipicamente expresso em termos das variações permitidas num grupo de parâmetros, que são comumente chamados de variáveis de projeto e podem ser denotados por um vetor  $\mathbf{x} = [x_1, x_2, \dots, x_n]^T$ , onde  $n$  é o número total de variáveis de projeto de um problema específico. Na engenharia estrutural, as variáveis de projeto podem ser as dimensões das seções transversais, parâmetros que controlam a geometria, forma da estrutura ou ainda, propriedades dos materiais utilizados. O conjunto de variáveis que fornecem o valor ótimo do projeto avaliado é chamado de ponto ótimo e também pode ser representado por um vetor  $\mathbf{x}^* = [x_1^*, x_2^*, \dots, x_n^*]^T$ .

As variáveis podem assumir valores contínuos ou discretos. Na maioria dos problemas de projetos estruturais, tende-se a negligenciar a natureza discreta das variáveis de projeto na solução do problema de otimização. No caso da presença de variáveis discretas, uma alternativa adotada é o ajuste do projeto ótimo obtido para o valor discreto mais próximo. Esta abordagem é adotada porque resolver um problema de otimização com variáveis discretas é, normalmente, muito mais complicado do que resolver o mesmo considerando as variáveis contínuas. Entretanto, arredondamentos no projeto, para a solução inteira mais próxima, funcionam bem quando os valores admissíveis para as variáveis são espaçados a uma distância cujas mudanças no valor de uma variável para o inteiro mais próximo não modificam substancialmente a resposta da estrutura. Alternativamente, os algoritmos pertencentes à classe evolucionária lidam diretamente com variáveis discretas.

### **2.2.2 Função objetivo**

A noção de otimização implica na existência de alguma função de mérito  $f(x)$  que pode ser melhorada e utilizada como uma medida da eficácia do projeto. A essas funções é dado o nome de função objetivo. Em problemas de otimização estrutural, peso, deslocamentos, tensões, frequências de vibração, carga de flambagem, e custo, são comumente utilizados como função objetivo.

Elas podem ser função de uma variável (unidimensional) ou de várias variáveis (multidimensional). Da mesma maneira, o problema de otimização pode apresentar um ou vários objetivos, sendo o primeiro definido como problema de otimização uni-objetivo ou escalar e o último como problema de otimização multiobjetivo ou vetorial.

Chama-se solução ótima o valor obtido na avaliação do projeto, fornecido pelo ponto ótimo, e este pode ser classificado em:

- ótimo local – melhor solução encontrada em uma região específica do espaço de projeto (próximo ao ponto inicial);
- ótimo global – melhor solução encontrada em todo o espaço de projeto investigado.

Não são todas as técnicas de otimização garantem que a solução encontrada será a ótima global. A maioria apenas converge para uma solução local (BEALE, 1988; CASTRO, 2001).

### 2.2.3 Restrições

Em muitos projetos, algumas condições são impostas de modo a limitar a escolha do projetista. A essas condições dá-se o nome de restrições que podem ser geométricas ou de comportamento (do ponto de vista físico), ou de igualdade ou desigualdade (do ponto de vista matemático). As restrições geométricas, também chamadas restrições de limite, são as limitações impostas diretamente às variáveis de projeto já as restrições de comportamento são aquelas que limitam indiretamente às variáveis de projeto.

As restrições geométricas são determinadas através de valores que impõem limites inferiores e/ou superiores e são por natureza funções de desigualdade (geralmente concebidas na forma  $g_i(\mathbf{x}) \leq 0$ ).

As restrições de comportamento são determinadas através de especificações de funções que dependem das variáveis de projeto, impondo a perpetuação das mesmas em um semi-espço, através de funções de desigualdade, ou em um hiper-plano, através de funções de igualdade (geralmente concebidas na forma  $h_i(\mathbf{x}) = 0$ ). As restrições podem ser funções de uma, de algumas, ou de todas as variáveis de projeto.

O espaço de projeto é delimitado pelas restrições geométricas, e a viabilidade do projeto é determinada pela intersecção entre o espaço delimitado pelas restrições geométricas e as restrições de comportamento. Para um projeto na região viável, uma restrição de desigualdade pode ser ativa ou não, porém, na solução ótima, as restrições de igualdade sempre devem estar ativas. Uma restrição de desigualdade é dita ativa para um ponto  $\mathbf{x}^*$  se  $g_i(\mathbf{x}^*) = 0$  e inativa para o mesmo ponto se  $g_i(\mathbf{x}^*) < 0$  (TORRES, 2001). Uma restrição de igualdade é dita ativa para um ponto  $\mathbf{x}^*$  se  $h_i(\mathbf{x}^*) = 0$ . Para um projeto na região inviável, existem duas possibilidades ao considerar um ponto  $\mathbf{x}$  da mesma,  $g_i(\mathbf{x}) > 0$  ou  $h_i(\mathbf{x}) \neq 0$ .

É importante frisar que o número de funções de restrições de igualdade deve ser menor ou igual ao número de variáveis (NOCEDAL e WRIGHT, 2000). Caso isso não ocorra, tem-se um sistema de equações indeterminado, onde há uma formulação inconsistente ou alguma restrição redundante (isto é, linearmente dependente de outra). No caso das restrições de desigualdade, não há limitação imposta ao número de funções.

### 2.2.4 Formulação padrão

O conceito apresentado anteriormente para variáveis de projeto, função objetivo e de restrições, pode ser sumarizado na formulação matemática de um problema típico de otimização da seguinte maneira:

$$\begin{aligned} &\text{Minimize } f(\mathbf{x}) \\ &\text{sujeito à: } g_j(\mathbf{x}) \leq 0, \quad j = 1 \dots l \\ &\quad \quad \quad h_k(\mathbf{x}) = 0, \quad k = 1 \dots m \\ &\quad \quad \quad x_i^l \leq x_i \leq x_i^u, \quad i = 1 \dots n \end{aligned} \tag{2.1}$$

onde  $\mathbf{x}$ ,  $f(\mathbf{x})$ ,  $\mathbf{g}(\mathbf{x})$ ,  $\mathbf{h}(\mathbf{x})$ ,  $\mathbf{x}^l$  e  $\mathbf{x}^u$  representam, respectivamente, o vetor com as variáveis de projeto, a função objetivo, as restrições de desigualdade e igualdade e os limites inferiores e superiores variáveis de projeto.

O problema é apresentado como uma minimização porque a maioria dos algoritmos de otimização é formulada dessa maneira, uma vez que a maximização pode ser obtida através da minimização da sua forma negativa. Da mesma forma, se na formulação for apresentada alguma restrição de desigualdade de uma forma diferente, pode se chegar à formulação apresentada através de algumas operações matemáticas.

### 2.2.5 Normalização do problema de otimização

Em problemas de otimização, a discrepância entre a magnitude das variáveis de projeto e/ou das funções objetivo e de restrições, pode levar a dificuldades numéricas na solução dos mesmos. Este problema pode ser resolvido através da normalização destas quantidades.

As variáveis de projeto devem ser normalizadas através da razão entre o valor atual e o valor inicial, isto é:

$$\bar{x}_i = x_i / x_{i_0} \tag{2.2}$$

onde  $x_i$  e  $x_{i_0}$  são, respectivamente, a variável  $i$  do ponto  $x$  corrente e a variável  $i$  do ponto inicial.

Similarmente a magnitude da função objetivo pode ser normalizada como mostra a equação (2.3).

$$\bar{f}(x) = f(x) / f(x_0) \quad (2.3)$$

onde  $f(x)$  e  $f(x_0)$  são, respectivamente, a função objetivo avaliada no ponto  $x$  (corrente) e a função objetivo avaliada no ponto inicial.

A normalização das restrições de desigualdade é feita em função do valor limite estabelecido para as mesmas. Um exemplo de normalização é apresentado no esquema a seguir:

$$\bar{g}_j(x) = \frac{g_j(x) - g_j^u}{g_j^u} \leq 0 \text{ com } g_j^u \neq 0 \quad (2.4)$$

onde  $g_j(x)$  e  $g_j^u$  são, respectivamente, o valor da restrição  $j$  correspondente a variável  $x$  e valor limite superior da mesma. Da mesma forma pode ser obtida a normalização para restrições com limite inferior.

A normalização das restrições de igualdade pode ser feita de forma similar ao apresentado para o caso da função objetivo, na equação (2.3).

## 2.3 Programação matemática

A programação matemática pode ser considerada como a primeira linha de métodos para resolução de problemas de otimização através do uso de algoritmos computacionais. Ela trata o problema de forma iterativa e determinística, isto é, através de gradientes, funcionais e operações matriciais (CASTRO, 2001). Devido a isto, normalmente necessita de várias informações iniciais.

Os algoritmos são distinguidos entre algoritmos de ordem zero, primeira ordem e segunda ordem, dependendo se a solução do mesmo exige apenas o valor da função, da primeira derivada ou da segunda derivada (TORRES, 2001).

Para resolver alguns tipos de problemas de otimização, e lidar com problemas restritos de várias variáveis, é comum definir a função Lagrangeana  $\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda})$  do problema original como segue:

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \sum_{j=1}^l \lambda_j g_j(\mathbf{x}) + \sum_{k=1}^m \lambda_{l+k} h_k(\mathbf{x}) \quad (2.5)$$

onde  $\boldsymbol{\lambda}$  é o vetor com os multiplicadores de Lagrange associadas às restrições  $g_j$  e  $h_k$  no ponto  $\mathbf{x}$ . Pode ser mostrado (VANDERPLAATS, 1984; HAFTKA, e GÜRDAL, 1993) que a condição de mínimo local desta função na solução  $\mathbf{x}^*$ , satisfaz as condições necessárias de Karush, Kuhn e Tucker:

1. Viabilidade:

$$\begin{cases} g_j(\mathbf{x}^*) \leq 0, & j = 1 \dots l \\ h_k(\mathbf{x}^*) = 0, & k = 1 \dots m \end{cases} \quad (2.6)$$

2. Estacionaridade:

$$\begin{aligned} \exists \boldsymbol{\lambda} \in \mathbb{R}^{l+m} \mid \nabla f(\mathbf{x}^*) + \sum_{j=1}^l \lambda_j \nabla g_j(\mathbf{x}^*) + \sum_{k=1}^m \lambda_{l+k} \nabla h_k(\mathbf{x}^*) = 0 \\ \text{com } \lambda_j \geq 0 \quad j = 1 \dots l \end{aligned} \quad (2.7)$$

3. Complementaridade:

$$\lambda_j g_j(\mathbf{x}^*) = 0, \quad j = 1 \dots l \quad (2.8)$$

O processo de otimização normalmente é encetado com uma proposta inicial  $\mathbf{x}_0$ , fornecido como entrada. O projeto é então atualizado modificando  $\mathbf{x}$  através da equação (2.9).

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \alpha \mathbf{r}_t \quad (2.9)$$

Na expressão acima,  $t$  é o iterando. O vetor  $\mathbf{r} = [r_1, r_2, \dots, r_n]^T$  é a direção de busca, que indica a direção viável para qual o valor de  $f(\mathbf{x})$  decresce. O escalar  $\alpha$  é o tamanho do passo na direção de  $\mathbf{r}$ .

Os métodos de solução são aplicados de acordo com o tipo de problema de programação matemática identificado. No geral, este tipo é determinado pelas características das funções objetivo e de restrições envolvidas (AFONSO, 1995). A classificação mais simples é:

- Programação linear: quando ambos, objetivo e restrições, são funções lineares ou são assumidas como funções lineares;
- Programação quadrática: quando o objetivo é quadrático, ou é assumido como função quadrática, e as restrições são lineares, ou assumidas assim;
- Programação não-linear: quando ambos, objetivo e restrições, são funções não-lineares.

Os primeiros dois métodos foram desenvolvidos para lidar com classes especiais de problemas de otimização, já os métodos de programação não-linear consistem de uma categoria genérica de algoritmos. As formas mais complexas de programação matemática envolvem funções não-lineares.

O método de programação não-linear é agrupado em duas categorias. Essa classificação depende de como a otimização é efetuada no algoritmo.

1. Métodos indiretos: Esses métodos envolvem técnicas para solucionar o problema original de otimização através da conversão do mesmo em um problema de otimização equivalente. Alguns exemplos de métodos indiretos são:
  - Métodos de penalidade ou barreira;
  - Método dual;
2. Métodos diretos: Esses métodos resolvem diretamente o problema original em vez de convertê-lo num problema equivalente. Alguns exemplos de métodos indiretos são:
  - Métodos gradientes;
  - Métodos sequenciais, recursivos ou sucessivos.

A classificação anterior dá apenas uma idéia dos algoritmos de otimização disponíveis.



Na otimização estrutural, muitos algoritmos podem ser aplicados. Os mais comuns são: o método das assíntotas móveis (*Method of Moving Asymptotes* – MMA); a programação sequencial linear (*Sequential Linear Programming* – SLP); e a programação sequencial quadrática (*Sequential Quadratic Programming* – SQP), entretanto apenas o SQP será apresentado, pois este será usado neste trabalho objetivando comparações com os resultados obtidos através do algoritmo evolucionário de estudo dessa dissertação.

### 2.3.1 Programação sequencial quadrática

Será dada uma breve descrição do algoritmo SQP. Considerando uma expansão de segunda ordem em série de Taylor da função  $f(\mathbf{x})$  em  $\bar{\mathbf{x}}$ , de acordo com:

$$f(\mathbf{x}) = f(\bar{\mathbf{x}}) + \nabla f(\bar{\mathbf{x}})^T \Delta \mathbf{x} + \frac{1}{2} \Delta \mathbf{x}^T \mathbf{H}(\bar{\mathbf{x}}) \Delta \mathbf{x} \quad (2.10)$$

onde  $\Delta \mathbf{x} = \mathbf{x} - \bar{\mathbf{x}}$ ,  $\nabla f(\bar{\mathbf{x}})$  é o vetor gradiente de  $f$  em  $\bar{\mathbf{x}}$ , de quem os elementos são definidos por:

$$\nabla f_i = \frac{\partial f}{\partial x_i} \quad (2.11)$$

e  $\mathbf{H}(\bar{\mathbf{x}})$  é a matriz Hessiana, de quem os elementos  $H_{ij}$  são definidos por:

$$H_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j} \quad (2.12)$$

É assumido que o vetor gradiente e a matriz Hessiana, na equação (2.10) irão proporcionar uma boa aproximação da função verdadeira, especialmente se  $\bar{\mathbf{x}}$  está próximo ao ponto ótimo.

Num algoritmo SQP, uma sequência de subproblemas quadráticos são resolvidos. A função objetivo deste subproblema é tal que o coeficiente do termo linear é formado pelo gradiente da função objetivo do problema principal enquanto para o termo quadrático, uma aproximação da Hessiana da função Lagrangeana (esquema BFGS – método Broyden-Fletcher-Goldfarb-Shanno) (VANDERPLAATS, 1984) do problema principal é usada. Uma descrição detalhada deste algoritmo pode ser encontrada em Powell (POWELL, 1978) e em Vanderplaats (VANDERPLAATS, 1984). Em suma, as principais etapas envolvidas no algoritmo SQP convencional são:

1. Estabelecer uma solução inicial  $\mathbf{x}_0$ ;
2. Configurar uma aproximação inicial para a matriz Hessiana dos termos quadráticos da função objetivo;
3. Resolver o subproblema para encontrar a direção de busca  $\mathbf{r}$ ;
4. Realizar uma busca linear para determinar o tamanho do passo  $\alpha$  na direção  $\mathbf{r}$ ;
5. Atualizar a matriz Hessiana e a solução, remetendo-a para a posição indicada;
6. Checar a convergência; se o mínimo local for encontrado o processo pára, caso contrário, volta para o passo 2.

## 2.4 Metaheurísticas

Um problema de otimização pode ser classificado levando em consideração o domínio das variáveis de projeto, definindo-o como problema combinatório (quando as variáveis são discretas) e problema contínuo. Esforços têm sido feitos para resolver estes tipos de problemas, porém de maneira distinta. Alguns algoritmos são particulares para cada tipo de problema.

O surgimento de uma nova metodologia, chamada de metaheurística, permite que ambos os tipos de problemas citados acima, sejam resolvidos sem muita distinção. Os métodos metaheurísticos têm sido desenvolvidos desde a década de 80 com um objetivo comum: resolver problemas complexos de otimização da melhor maneira possível.

Estão incluídos nestas metodologias algoritmos como o da recozimento simulado (*Simulated Annealing*), os algoritmos genéticos (*Genetic Algorithms*), o do enxame de partículas (*Particle Swarm*) e o algoritmo da colônia de formigas (*Ant Colony Algorithm*) (COLLETTE e SIARRY, 2003). Algumas características em comum desses algoritmos são apresentadas a seguir.

- Todos são, pelo menos em parte, estocásticos;
- Suas origens são combinatórias;
- Eles são inspirados em analogias;
- São capazes de orientar, em tarefas especiais, outro método de busca especializado;
- Eles compartilham as mesmas desvantagens.

Um algoritmo clássico de “melhoria iterativa” (o método de declive) não é capaz, em geral, de localizar um mínimo global, mas somente um mínimo local que corresponde à melhor solução acessível em relação ao ponto inicial. Para melhorar o resultado encontrado, pode-se repetir o processo inúmeras vezes, com diferentes pontos iniciais e tomar como resultado final a melhor entre todas as soluções locais encontradas. Entretanto, esse processo aumentaria bastante o tempo requerido para otimização e ainda assim não haveria uma garantia de que a solução encontrada seria o mínimo global.

Superar esses obstáculos, e obter a solução global para problemas com vários mínimos locais, é a base de todos os métodos metaheurísticos. Nos metaheurísticos de vizinhança (recozimento simulado e busca tabu, por exemplo), a meta é permitir, de tempo em tempo, um movimento de escalada, quer dizer, permitir temporariamente uma piora na solução quando a configuração corrente é modificada. Esse mecanismo de degradação é específico para cada metaheurístico e é o que torna possível a fuga de armadilhas de mínimos locais para explorar outros “vales” no espaço de projeto investigado. Metaheurísticos “espalhados” (tais como algoritmos genéticos) têm mecanismos de fuga (a mutação é um deles) que modificam a solução enfatizando o mecanismo de luta coletiva contra os mínimos locais através do controle paralelo de uma “população” de soluções. O foco dos estudos realizados foi a utilização de algumas técnicas para solucionar problemas de otimização que apresentavam vários ótimos. As metodologias metaheurísticas mais tradicionais serão apresentadas no capítulo 3.

## 2.5 Otimização multiobjetivo e dominância

### 2.5.1 Otimização multiobjetivo

Quando um problema é formulado, muitas vezes se quer atender a vários objetivos, por exemplo, quando se deseja uma estrutura que apresente uma configuração com o menor volume de material e ao mesmo tempo com a menor energia de deformação. Neste caso, o problema é chamado de problema de otimização multiobjetivo (ou problema de otimização com múltiplos critérios) (COLLETTE e SIARRY, 2003) e é comumente apresentado da seguinte forma:

$$\begin{aligned} &\text{Minimize } \mathbf{F}(\mathbf{x}) \\ &\text{sujeito à: } g_j(\mathbf{x}) \leq 0, \quad j = 1 \dots l \\ &\quad h_k(\mathbf{x}) = 0, \quad k = 1 \dots m \\ &\quad x_i^l \leq x_i \leq x_i^u, \quad i = 1 \dots n \end{aligned} \tag{2.13}$$

onde  $\mathbf{F}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_{nobj}(\mathbf{x})]$  é um vetor que contém todos os objetivos a serem avaliados no problema e  $nobj$  é o número total de objetivos encontrados no problema em questão.

Normalmente, os objetivos a serem otimizados são contraditórios (um objetivo é dito contraditório a outro quando uma degradação feita no mesmo implica na melhoria do outro) e devido a isso a busca não encontra uma solução única, mas um conjunto de soluções. Essas soluções são chamadas soluções de Pareto e o conjunto formado por todas elas é a superfície de Pareto ou frente de Pareto (*tradeoff*) (MACÊDO, 2002).

Depois de encontrar as soluções do problema de otimização multiobjetivo, é enfrentada uma dificuldade: deve-se escolher uma solução deste grupo. A solução selecionada pelo usuário refletirá no interesse do mesmo pelos vários objetivos relacionados. Como o tomador de decisões é humano, uma das metas da otimização multiobjetivo é modelar as várias possibilidades de escolhas ou as preferências do tomador de decisões.

O ganho de mais liberdade na modelagem dos problemas é uma grande vantagem da otimização multiobjetivo, porém, enquanto a seleção do método de otimização uni-objetivo já é difícil devido à diversidade de métodos disponíveis, na otimização multiobjetivo a diversidade é ainda maior.

## 2..1 Dominância

Quando o problema de otimização multiobjetivo é resolvido, uma grande quantidade de soluções é encontrada, porém, somente um pequeno subgrupo dessas soluções é de interesse. Para uma solução ser interessante, deve existir uma relação de dominância entre a solução considerada e as outras soluções comparadas a ela. A relação de dominância é definida da seguinte forma:

- Um vetor solução  $\mathbf{x}^1$  domina um vetor solução  $\mathbf{x}^2$  se:

$$\begin{aligned} \text{Para um determinado } \mathbf{F}(\mathbf{x}) &= [f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_{nobj}(\mathbf{x})]^T \\ f_q(\mathbf{x}^1) &\leq f_q(\mathbf{x}^2) \text{ para todos os objetivos, e} \\ f_q(\mathbf{x}^1) &< f_q(\mathbf{x}^2) \text{ para pelo menos um dos objetivos} \end{aligned} \quad (2.14)$$

Soluções que dominam outras soluções e não apresentam soluções que as dominem são denominadas soluções de Pareto (pontos de Pareto ou soluções não dominadas) (COLLETTE e SIARRY, 2003).

Nos problemas de otimização multiobjetivo encontrar um  $\mathbf{x}^*$  que minimize  $\mathbf{F}$  (i.e todas as funções simultaneamente) é uma tarefa muito árdua, senão impossível para quase a totalidade dos problemas devido ao conflito de objetivos, como explicado anteriormente. Uma forma de determinar um  $\mathbf{x}$  que satisfaça em parte os problemas de otimização multiobjetivo está contida na definição de otimalidade de Pareto. Pontos de Pareto são pontos  $\mathbf{x}^p$  tais que satisfaçam a relação indicada na equação (2.14).

Tomando como exemplo um problema com duas funções objetivo, onde se deseja obter o valor mínimo para ambas, quando se aplica o critério de dominância, quatro áreas podem ser definidas e a cada área pode ser associado um nível de preferência, como mostrado na figura (2.1) (COLLETTE e SIARRY, 2003).

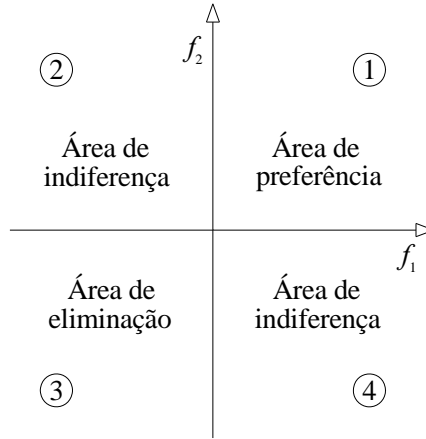


Figura (2.1) – Nível de preferência e relação de dominância.

Considerando que a figura esteja centrada em uma solução A e comparando-se um ponto B à mesma, têm-se as seguintes possibilidades:

- se a solução B pertence à área 1, então a solução A domina B;
- se a solução B pertence à área 3, então a solução A é dominada pela solução B;
- se a solução B pertence à área 2 ou 4, então não se pode dizer qual das soluções é preferível em relação à outra.

É estabelecido um posto, chamado posto de Pareto, baseado no número de soluções pelas quais uma solução é dominada (COLLETTE e SIARRY, 2003). Para entender melhor a determinação deste posto é apresentada a equação (2.15).

$$rank^i = 1 + n_{dom}^i \quad (2.15)$$

Na equação (2.15),  $rank^i$  é o posto de Pareto da solução  $i$  e  $n_{dom}^i$  é o número de soluções que dominam a solução  $i$ .

## 2.7 Superfície de *tradeoff* ou frente de Pareto

O menor número de soluções com  $rank = 1$  selecionadas de acordo com as regras de triagem baseadas nos critérios de dominância de Pareto, produzem a superfície de *tradeoff* ou frente de Pareto.

Considerando mais uma vez um problema com dois objetivos (minimizar  $f_1$  e  $f_2$  sujeitas à  $\mathbf{g}(\mathbf{x}) \leq \mathbf{0}$  e  $\mathbf{h}(\mathbf{x}) = \mathbf{0}$ ):  $S$  denota o grupo de pontos com coordenadas  $(f_1(\mathbf{x}^i), f_2(\mathbf{x}^i))$  quando  $\mathbf{x}^i$  respeita as restrições  $\mathbf{g}$ ,  $\mathbf{h}$  e de limites; e  $P$  denota a frente de Pareto (COLLETTE e SIARRY, 2003). A figura (2.2) representa  $S$  e  $P$ .

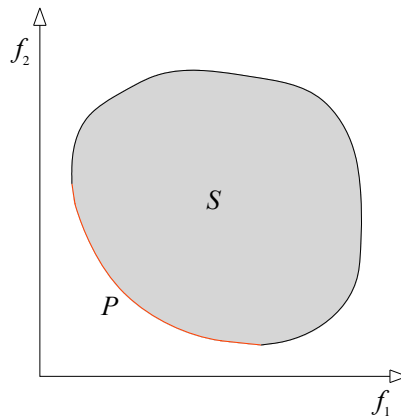


Figura (2.2) – Representação da superfície de *tradeoff*.

Uma propriedade que deve ser destacada é a forma da frente de Pareto, pois a mesma depende do tipo de problema considerado. A forma mais comum de frente de Pareto é a mostrada na figura (2.2). Estas formas são típicas de problemas multiobjetivos com um conjunto convexo de soluções (COLLETTE e SIARRY, 2003).

## 2.7 Convexidade

Um conjunto  $S$  é convexo se, dados dois pontos distintos no conjunto, o segmento que liga estes dois pontos se encontra por completo no conjunto (COLLETTE e SIARRY, 2003). Um exemplo de um conjunto convexo e de um conjunto não-convexo é representado na figura (2.3).

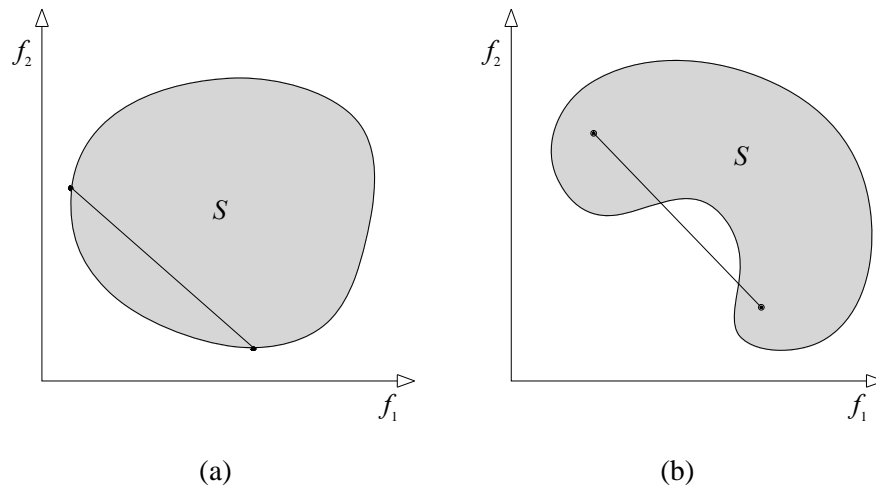


Figura (2.3) – Exemplo de conjunto (a) convexo e (b) não-convexo.

Alguns métodos de otimização multiobjetivo requerem que algumas hipóteses sejam respeitadas. Muitas vezes precisam trabalhar sobre um conjunto convexo  $S$  de valores de  $\mathbf{F}$ . A seguir serão apresentados alguns métodos para determinar a frente de Pareto.

## 2.8 Métodos para geração de pontos de Pareto

Existem várias técnicas para se obter a frente de Pareto (HERNÁNDEZ, 1994; SRINIVAS e DEB, 1994; DAS e DENNIS, 1997; MACÊDO, 2002). Neste capítulo serão apresentados resumos sobre algumas das metodologias mais conhecidas.

### 2.8.1 Método da soma ponderada dos objetivos (*Weighted Sum Method* – WS)

Um dos métodos convencionais de otimização multiobjetivo e talvez o mais conhecido seja o WS. Dentre os métodos desenvolvidos, no qual se substitui as funções objetivo por uma única, denominada de função substituta, este é o mais empregado e de uso mais simples (KOSKI, 1985; AFONSO, 1997; AFONSO e SIENZ, 1999). A técnica do WS baseia-se em atribuir um vetor de coeficientes de ponderação às funções objetivo normalizadas, linearizando-as, ou seja, transformando-as em uma única função objetivo. Sua representação algébrica é dada por:



$$f_{sub} = \sum_{k=1}^p \alpha_k \bar{f}_k \quad (2.16)$$

com

$$\sum_{k=1}^p \alpha_k = 1 \text{ com } \alpha_k \geq 0 \quad (2.17)$$

onde  $f_{sub}$  representa a função substituta,  $\bar{f}_k$  é a função normalizada  $k$  como descrito na equação (2.3) e  $\alpha_k$  é o valor do peso da função associada a ela na função substituta.

Em geral, quando se utiliza essa metodologia, para uma distribuição uniforme de pesos  $\alpha_k$  uma distribuição uniforme na superfície de resposta é obtida, porém não consegue apresentar um bom resultado quando é utilizado sobre um conjunto não-convexo.

### 2.8.2 Método da intersecção contorno-normal (*Normal-Boundary Intersection – NBI*)

O método do NBI (DAS e DENNIS, 1997) é uma técnica criada para encontrar pontos eficientes (ou pontos NBI) do contorno do espaço viável gerado pelos vetores objetivos alcançáveis, que possibilitem a construção de uma curva suave para o problema multiobjetivo. Quando os pontos eficientes estiverem sobre uma parte do contorno suficientemente convexa do espaço viável, esses pontos são definidos como pontos de Pareto, porém, se os pontos avaliados estiverem na parte não-convexa, não há garantia de que tais pontos sejam pontos de Pareto. Apesar disso, esses pontos contribuem para que a frente de Pareto seja definida.

Este método é inovador com relação à obtenção dos pontos eficientes sobre a frente de Pareto, pois, diferente do método descrito anteriormente, não é necessária a transformação das funções objetivo em uma única função.

A idéia central do NBI é encontrar uma porção do contorno do espaço das funções objetivo (DAS e DENNIS, 1997), que contém os pontos ótimos de Pareto. Tais pontos podem ser encontrados resolvendo-se um problema de otimização. No que se seguem algumas terminologias específicas do método precisam ser descritas para um melhor entendimento da metodologia.

Define-se  $\mathbf{F}^*$  como sendo o vetor mínimo local das funções objetivo, denominado de ponto utópico (*Shadow Minima* ou *Utopia Point*) (DAS e DENNIS, 1997), representado por:

$$\mathbf{F}^* = [f_1^*, f_2^*, \dots, f_{nobj}^*]^T \quad (2.18)$$

onde cada  $f_k^*$  representa o mínimo do objetivo  $k$  avaliado individualmente.

Sendo  $\mathbf{x}_k^*$  a solução ótima de  $f_k$ , tem-se  $f_k^* = f_k(\mathbf{x}_k^*)$ . Defini-se a envoltória convexa do mínimo individual (ECMI) como:

$$\left\{ \Phi \boldsymbol{\beta} : \boldsymbol{\beta} \in \mathbb{R}^p, \sum_{k=1}^p \beta_k = 1, \beta_k \geq 0 \right\} \quad (2.19)$$

onde

$$\Phi(i, j) = f_i(x_j^*) - f_i^* \text{ com } i=1 \dots nobj \text{ e } j=1 \dots nobj \quad (2.20)$$

Assim os pontos pertencentes à ECMI são definidos por um conjunto de pontos do  $\mathbb{R}^{nobj}$ , que são definidos pelas combinações convexas de  $\{\mathbf{F}(\mathbf{x}_k^*) - \mathbf{F}^*\}$ , armazenados sob a forma de matriz,  $\Phi$  denominada de “*pay-off*” (DAS e DENNIS, 1997).

O conjunto das funções objetivo no espaço viável será denominado por  $f$  e o seu contorno por  $\partial f$ . A técnica NBI possui o objetivo de encontrar parte do contorno que contém os pontos ótimos de Pareto. A idéia geométrica associada ao método é que tais pontos são encontrados a partir da intersecção da reta normal à ECMI, que aponta para a origem, e o contorno  $\partial f$  como ilustrado na figura (2.4). Nela pode ser observado que a família dos vetores normais uniformemente espaçados intercepta os pontos igualmente espaçados sobre o contorno.

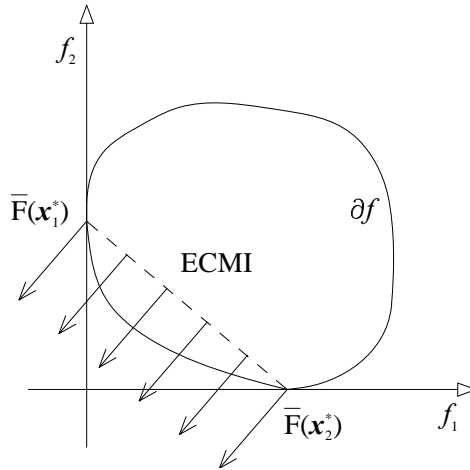


Figura (2.4) – Conjunto viável sobre o mapeamento de  $f$  no espaço das funções objetivo.

Matematicamente, tais pontos podem ser encontrados resolvendo-se um problema de otimização como descrito a seguir.

Dado  $\Phi\beta$  que contém as constantes,  $\Phi\beta$  representa um ponto sobre a ECMI. Seja  $\hat{n}$  o vetor unitário normal à ECMI, direcionado desta na direção da origem do sistema, passando pelos pontos  $\Phi\beta$ . Então,  $\Phi\beta + t\hat{n}$ , com  $t \in \mathbb{R}$ , representa o conjunto de pontos sobre àquela normal. A intersecção entre as normais à ECMI e o contorno que define o espaço  $\{F(x) | x \in C\}$ , onde  $C = \{x : h(x) = 0, g(x) \leq 0, x^l \leq x \leq x^u\}$ ,  $\partial f$  mais próximo da origem é a solução global do seguinte problema de programação não-linear:

$$\text{Maximize } t \quad (2.21)$$

sujeito às restrições definidas para o problema esquematizado na equação (2.1) e às restrições adicionais

$$\Phi\beta + t\hat{n} = \bar{F}(x) \quad (2.22)$$

sendo esta a equação de restrição garantindo o mapeamento de  $x$  por  $\bar{F}(x)$  sobre a normal, onde  $\bar{F}(x) = F(x) - F^*$ .

O problema apresentado nas equações (2.21), (2.22) e as restrições (2.1) passam a ser definido com um subproblema NBI, representado por  $\text{NBI}_\beta$ , considerando que  $\beta$  seja o parâmetro que caracteriza o subproblema. Resolvendo esse subproblema para um conjunto de parâmetros  $\beta$ , encontra-se um conjunto de pontos sobre  $\partial f$  que poderão fornecer uma curva suavizada. Esses pontos serão pontos de Pareto caso estejam numa região convexa de  $\partial f$ , caso contrário, não serão pontos de Pareto, mas serão úteis na suavização da curva o que não ocorre no método descrito anteriormente.

### 2.8.3 Método da restrição normal normalizada (*Normalized Normal-Constraints* – NNC)

Este método foi introduzido por Messac et. al. (MESSAC et al., 2003) e é uma melhoria sobre o método da Restrição Normal (*Normal-Constraints* - NC) de Yahaya e Messac (ISMAIL-YAHAYA e MESSAC, 2002) removendo problemas numéricos de escala com a normalização das funções objetivo. Este método mostrou-se capaz de gerar uma propagação uniforme de pontos de Pareto. O NNC trabalha de forma similar ao NBI e sua representação (MESSAC et al., 2003) é mostrada na figura (2.5).

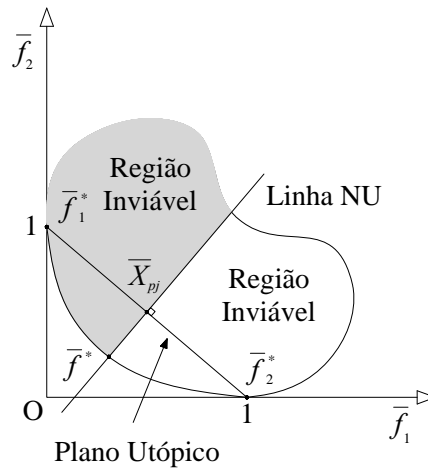


Figura (2.5) – Representação gráfica do NNC para um problema com dois objetivos.

A normalização das funções objetiva consideradas neste método assume a forma descrita na equação (2.23)

$$\bar{f}_k = \frac{f_k - \min f_k}{\max f_k - \min f_k} \quad (2.23)$$

onde  $\bar{f}_k$ ,  $\max f_k$  e  $\min f_k$  são, respectivamente, o valor normalizado e o valor máximo e mínimo do objetivo  $k$ .

No espaço dos objetivos do projeto normalizado, todos os pontos mínimos locais das funções objetivo estão a uma determinada distância do ponto utópico e este se encontra na origem, por definição. Os pontos de mínimo das funções objetivo são obtidos pela minimização separada das mesmas.

Considerando um problema com dois objetivos, como apresentado na figura (2.5), um seguimento de reta que liga os dois pontos de mínimo é chamado de Linha Utópica e é equivalente à ECMI no NBI. Sobre este segmento são determinados vários pontos através da divisão do mesmo.

Na figura (2.5) um ponto genérico interceptando o segmento é utilizado para definir a uma linha normal à linha utópica. Esta linha normal (NU) diminui a região viável, e como pode ser visto, quando se minimiza  $\bar{f}_2$ , o resultado obtido será  $\bar{f}^*$ . Transladando a linha normal, pela linha utópica, um conjunto de pontos de Pareto será obtido.

Na formulação matemática do NNC para otimização multiobjetivo, considerando  $p$  objetivos, o vetor  $\mathbf{x}$  define as variáveis de projeto e  $f_k$  define a  $k$ -ésima função objetivo. O vetor  $\mathbf{x}_k^*$  é o ponto ótimo da função  $f_k$ .

Os vetores do plano utópico  $\bar{\mathbf{N}}_q$  são definidos como a direção de  $\bar{\mathbf{F}}(\mathbf{x}_q^*)$  até  $\bar{\mathbf{F}}(\mathbf{x}_p^*)$  dada pela equação (2.24).

$$\bar{\mathbf{N}}_q = \bar{\mathbf{F}}(\mathbf{x}_p^*) - \bar{\mathbf{F}}(\mathbf{x}_q^*) \text{ com } q = 1 \dots (p-1) \quad (2.24)$$

Um conjunto de  $n$  pontos distribuídos no plano utópico é gerado da seguinte forma:

$$\bar{\mathbf{X}}_j = \sum_{k=1}^p \gamma_{j,k} \bar{\mathbf{F}}(\mathbf{x}_k^*) \text{ com } j = 1 \dots n \quad (2.25)$$

onde

$$0 \leq \gamma_{j,k} \leq 1 \text{ e } \sum_{k=1}^p \gamma_{j,k} = 1 \quad (2.26)$$

São gerados pontos distribuídos no espaço das funções objetivo normalizadas. Para cada ponto  $\bar{X}_j$  gerado é obtido um ponto de Pareto correspondente, solucionado o seguinte problema:

$$\text{Maximize } \bar{f}_p \quad (2.27)$$

sujeito às restrições definidas para o problema esquematizado na equação (2.1) e às restrições adicionais

$$\bar{\mathbf{N}}_q^T(\bar{\mathbf{F}} - \bar{\mathbf{X}}_j) \leq 0 \text{ com } q = 1 \dots (p-1) \quad (2.28)$$

O NNC permite que pontos de Pareto numa região não-convexa sejam obtidos.

As metodologias apresentadas anteriormente são apenas alguns dos métodos convencionais existentes para obtenção da superfície de Pareto. Outras metodologias, utilizadas nos métodos evolucionários, serão apresentadas nos capítulos 3 e 4.

# Algoritmos evolucionários

## 3.1 Introdução

Uma visão geral dos algoritmos evolucionários será apresentada, enfatizando suas características na solução de problemas de otimização. O procedimento geral de um algoritmo evolucionário genérico será apresentado como descrito em (INNOCENTE, 2006). Uma descrição dos algoritmos uni/multiobjetivo, baseada em uma perspectiva unificadora, será mostrada num esquema geral que constitui uma base comum a todas as variações de estratégias evolucionárias relatadas na literatura especializada.

Os principais componentes de um algoritmo evolucionário são: um processo de inicialização, um mecanismo para determinação da hereditariedade, operadores de mutação e um mecanismo de seleção (PULIDO, 2005). Cada um deles será brevemente discutido neste capítulo, também será explicado como esses componentes anteriormente mencionados são aplicados no correspondente algoritmo evolucionário multiobjetivo.

## 3.2 Visão geral, classificação das estratégias

Naturalmente, aqueles indivíduos que melhor se adaptam ao meio ambiente onde vivem tem mais oportunidade de competir pelos recursos e reproduzir (DARWIN, 1859). Nesse processo (conhecido como “a sobrevivência dos mais fortes”), a seleção natural desempenha o papel principal.

No entanto, em (DARWIN, 1859), Darwin também identificou que algumas raras modificações no fenótipo (mutações) afetam diretamente o desempenho de um indivíduo num determinado ambiente. Evolução natural pode ser vista como um problema de otimização, onde o objetivo é adaptar melhor as espécies para seu meio ambiente. Portanto, não é surpreendente que os cientistas têm tido inspiração na natureza, uma vez que o Neodarwinismo tem sido o mais importante modelo para explicar a evolução natural (PULIDO, 2005). Neodarwinismo salienta que todas as diversidades da vida no nosso planeta podem ser explicadas através de quatro processos:

1. Reprodução: este mecanismo garante a herança do material genético da geração atual para a próxima geração;
2. Mutação: acontece quando ocorre um erro na cópia do material genético (durante a reprodução). Uma mutação é benéfica para um organismo se produz um aumento da capacidade de adaptação ao seu meio ambiente;
3. Concorrência: é o processo natural, em que os organismos têm um confronto contínuo para sobreviver e para garantir a continuidade do seu código genético para a próxima geração;
4. Seleção: em um meio ambiente que só pode acolher um número limitado de indivíduos, apenas os organismos que competirem mais eficazmente pelos recursos sobrevivem e reproduzem.

Algoritmos evolucionários (AE) são capazes de gerar soluções para problemas difíceis do mundo real, principalmente devido a sua analogia com o neodarwinismo (o fato de que naturalmente, as populações evoluem através de gerações utilizando o mecanismo de seleção natural para sobrevivência do mais apto) (PULIDO, 2005). A evolução deste tipo de soluções depende da aplicação adequada dos seguintes pontos:

- A codificação das estruturas de dados utilizados para representar as soluções do problema. Cada solução é conhecida como um “indivíduo” e um conjunto de indivíduos é chamado de “população”;
- Operadores que modificam as características do indivíduo (os chamados operadores de mutação);
- Uma função de aptidão que desempenha o papel do ambiente;
- A sobrevivência do mais forte é implementada através da utilização de um processo de seleção, que desempenha o papel de pressão ambiental.



Hoje em dia, AE são muito populares, principalmente porque as apresentam várias vantagens em relação à otimização com o emprego de técnicas tradicionais, tais como:

- São conceitualmente muito simples;
- Têm uma grande aplicabilidade;
- Podem facilmente explorar os paradigmas da computação paralela;
- Normalmente podem adaptar os seus próprios parâmetros;
- São menos suscetíveis a serem aprisionados em um mínimo/máximo local (devido a serem de técnicas de base populacional);
- Não requerem a garantia da diferenciabilidade e continuidade das funções envolvidas;
- Podem cooperar com outras técnicas de busca/otimização (otimização híbrida).

Um esquema geral de um algoritmo evolucionário é apresentado na tabela (3.1) como visto em (PULIDO, 2005).

Tabela (3.1): Algoritmo geral de um AE

- 
1. Inicializar a população com soluções geradas aleatoriamente;
  2. Avaliar cada solução;
  3. Enquanto o critério de parada não for atingido fazer:
    4. Seleção dos indivíduos para reprodução;
    5. Aplicação do operador de mutação nos indivíduos selecionados;
    6. Avaliar os novos candidatos;
    7. Seleção do melhor indivíduo (entre todos) para a nova geração (elitismo).
- 

Tradicionalmente, os AE têm sido agrupados em três principais modelos:

- Programação evolucionária (*Evolutionary Programming* - EP);
- Estratégia evolutiva (*Evolution Strategies* - ES);
- Algoritmo genético (*Genetic Algorithm* - GA).

Alguns pesquisadores acrescentam um quarto modelo, enquanto outros afirmam ser apenas uma variação do algoritmo genético (INNOCENT, 2006).

- Programação genética (*Genetic Programming* - GP).

Além desses modelos, alguns incluem o método do enxame de partículas dentro dos AE (INNOCENT, 2006). O modelo do enxame de partículas é um dos métodos baseados na “*Swarm Intelligence*” porque não considera as metáforas da evolução natural e sim a metáfora da inteligência que surge de um comportamento de cooperação social entre um grupo de indivíduos. Todos os modelos de AE foram inicialmente desenvolvidos independentemente, prosseguindo por diferentes propósitos (INNOCENT, 2006).

Mais adiante cada um dos modelos citados anteriormente será apresentado.

### **3.2.1 Função Aptidão**

A função aptidão é um tipo específico de função que quantifica a otimalidade de uma solução nos algoritmos evolucionários, para que os indivíduos possam ser classificados e comparados entre si. Nesse contexto, os indivíduos que apresentam melhor aptidão são autorizados a “misturar sua raça”, ou seja, são autorizados a propagar seus dados através de qualquer uma das várias técnicas de transferência de informações existentes, a depender do algoritmo considerado.

Uma função aptidão ideal se correlaciona estreitamente com o objetivo do problema, e ainda pode ser computada rapidamente. Em muitos casos a definição da função aptidão não é simples. No capítulo 4 será apresentada a maneira como a função aptidão é considerada neste trabalho.

### **3.2.2 Programação evolucionária**

Lawrence J. Fogel concebeu a utilização de uma forma de evolução simulada para resolver problemas em meados da década de 1960 (FOGEL, 1964). A técnica que ele desenvolveu foi chamada de Programação Evolucionária. A inteligência nesta técnica pode ser vista como um comportamento adaptativo (PULIDO, 2005). Esta abordagem enfatiza a interação entre pais e filhos. Na EP, o operador de cruzamento não é necessário, uma vez que a EP é uma abstração do processo evolutivo em um nível de espécie (duas espécies diferentes não podem ser recombinadas) (PULIDO, 2005). A programação evolucionária usa seleção probabilística (normalmente simulações estocásticas). Atualmente, existem várias variações desta técnica.

O algoritmo básico programação evolucionária é apresentada na tabela (3.2) como visto em (PULIDO, 2005).

Tabela (3.2): Algoritmo da Programação Evolucionária

- 
1. Inicializar a população com soluções geradas aleatoriamente;
  2. Enquanto o critério de parada não for atingido fazer:
    3. Aplicar mutação;
    4. Avaliar os novos candidatos;
    5. Selecionar as soluções que serão mantidas;
    6. Gerar as novas populações;
- 

### 3.2.3 Estratégia evolutiva

Em meados dos anos 60, Hans-Paul Schwefel (SCHWEFEL, 1965), Peter Bienert (BIENERT, 1967), e posteriormente Ingo Rechenberg (RECHENBERG, 1973) desenvolveram um método de ajuste discreto aleatório, inspirados no mecanismo de mutação existente na natureza. A técnica desenvolvida foi chamada de Estratégia Evolutiva e foi inicialmente utilizada para resolver problemas de alta complexidade da hidrodinâmica (PULIDO, 2005).

A versão original é chamada (1+1)–ES (PULIDO, 2005) e usa um único progenitor que gera um único descendente. Se este descendente tiver uma melhor condição do que seu progenitor, ele sobrevive e se torna o progenitor da geração seguinte, caso contrário ele é eliminado (isso é chamado extinção seletiva).

No (1+1)–ES, um novo indivíduo é gerado usando:

$$\mathbf{x}^{t+1} = \mathbf{x}^t + N(0, \sigma) \quad (3.1)$$

onde  $t$  se refere à geração corrente (ou iteração), e  $N(0, \sigma)$  é um vetor de coordenadas Gaussianas com média igual a zero e desvio padrão igual a  $\sigma$ .

Estratégia evolutiva não envolve apenas as variáveis do problema, mas também seu próprio parâmetro (desvio padrão), em um processo chamado “auto-adaptação”. A recombinação existe, porém é normalmente um operador secundário (PULIDO, 2005). O mecanismo de seleção é normalmente determinístico. O algoritmo generalizado é apresentado na tabela (3.3) como visto em (PULIDO, 2005).

Tabela (3.3): Algoritmo da Estratégia Evolutiva

- 
1. Inicializar a população  $G$  com soluções geradas aleatoriamente;
  2. Fazer  $t = 0$ ;
  3. Avaliar  $G(t)$ ;
  4. Enquanto o critério de parada não for atingido fazer:
    5. Reproduzir  $G(t)$  para gerar  $G(t+1)$ ;
    6. Aplicar mutação em  $G(t+1)$ ;
    7. Avaliar  $G(t+1)$ ;
    8. Selecionar os sobreviventes;
    9. Fazer  $t = t + 1$ ;
- 

#### 3.2.4 Algoritmo genético

Algoritmos genéticos (originalmente chamados planos de reprodução) foram introduzidos por John H. Holland (HOLLAND, 1992) em meados dos anos 60. O principal interesse de Holland era estudar a adaptação natural com o intuito de aplicá-la em mecanismos de aprendizado artificial (*machine learning*). Hoje em dia, os algoritmos genéticos são os tipos mais populares de algoritmos evolucionários (PULIDO, 2005).

Um pseudocódigo generalizado de um modelo simples do GA é apresentado na tabela (3.4) como visto em (PULIDO, 2005).

Tabela (3.4): Algoritmo Genético Simples

- 
1. Inicializar a população  $G$  com soluções geradas aleatoriamente;
  2. Fazer  $t = 0$ ;
  3. Enquanto o critério de parada não for atingido fazer:
    4. Avaliar  $G(t)$ ;
    5. Selecionar  $G_1$  a partir de  $G(t)$ ;
    6. Fazer o cruzamento e aplicar a mutação em  $G_1$  para gerar  $G(t+1)$ ;
    7. Fazer  $t = t + 1$ ;
- 

No algoritmo anterior,  $G_1$  representa o grupo de indivíduos que melhor se adaptou ao meio (indivíduos com as melhores aptidões).

### 3.2.5 Programação genética

Nichal Lynn Cramer (CRAMER, 1985) e mais tarde, John R. Koza (KOZA, 1989) propuseram (num caminho diferente e independente) o uso de uma representação ramificada onde foi implementado um operador de cruzamento que intercambiava os sub-ramos entre diferentes programas computacionais implementados (com certas restrições impostas pela sintaxe da linguagem de programação utilizada) (PULIDO, 2005).

A diferença básica entre as propostas está no fato de Cramer usar uma função aptidão interativa (i.e., o usuário deve abastecer, manualmente, o valor da aptidão de cada ramo da população), enquanto Koza foi capaz de automatizar. A proposta de Koza prevaleceu e é hoje conhecida como programação genética (KOZA, 1992).

A programação genética aplica a abordagem do algoritmo genético para o espaço de eventuais programas computacionais. Programas computacionais são a *lingua franca* para expressar as soluções numa grande variedade de problemas. Uma grande variedade de problemas aparentemente diferentes, em diferentes áreas, que podem ser reformulados como uma busca de um programa computacional para resolver o problema (KOZA, 2003).

O algoritmo generalizado da programação genética é apresentado na tabela (3.5) como visto em (PULIDO, 2005).

Tabela (3.5): Algoritmo Generalizado da Programação Genética

- 
1. Inicializar a população  $G$  com soluções geradas aleatoriamente;
  2. Fazer  $t = 0$ ;
  3. Enquanto o critério de parada não for atingido fazer:
    4. Avaliar a população  $G(t)$ , com os programas computacionais adequados;
    5. Selecionar  $G_1(t)$  a partir de  $G(t)$ ;
    6. Aplicar os operadores genéticos em  $G_1(t)$  para gerar  $G(t+1)$ ;
  7. Fazer  $t = t + 1$ ;
- 

### 3.2.6 *Swarm Intelligence*

“*Swarm Intelligence*” é o ramo na área da inteligência artificial que envolve o estudo do comportamento coletivo que surge de um sistema descentralizado e auto-organizado (INNOCENT, 2006). Estudos do comportamento social dos organismos em multidões inspiraram a concepção de eficientes algoritmos de otimização. Por exemplo, a simulação da coreografia de um bando de aves levou a concepção do algoritmo de otimização do enxame de partículas, e os estudos no comportamento de vasculhar levou a concepção do algoritmo de otimização da colônia de formigas (ENGELBRECHT, 2002).

Nesses algoritmos, os indivíduos se comportam de acordo com suas experiências passadas e com a interação com outros indivíduos. Estas interações dão a característica de comportamento global ao algoritmo (SIERRA, 2006).

#### 3.2.6.1 Enxame de partículas (*Particle Swarm - PS*)

A otimização via enxame de partículas é um método, capaz de lidar com problemas de otimização em um espaço  $n$ -dimensional, que foi inspirado no comportamento social observado em alguns animais (como exemplos: bandos de aves, cardume de peixes, e sociedades de insetos), e na sociedade humana (INNOCENTE, 2006). Assim, o modelo do enxame de partículas tem fortes raízes em ambas, vida artificial (vida artificial é um campo relativamente novo que engloba todos os paradigmas baseados sobre a metáfora de organismos vivos, a vida artificial não pretende modelar a vida e sim criar novas formas de

vida que exibiriam inteligência, através de princípios de exploração subjacentes a organismos vivos) e psicologia social. De fato, o mesmo comportamento de alguns animais é relevante para ambas, vida artificial e psicologia social, daí existem várias ligações entre esses dois campos (INNOCENT, 2006). Mais detalhes serão dados no capítulo 4.

### **3.3 Algoritmos evolucionários em problemas multiobjetivos**

A otimização multiobjetivo vem sendo avaliada intensamente a pouco mais de duas décadas e suas aplicações em problemas reais estão aumentando continuamente. Conforme mencionado no capítulo anterior, uma importante tarefa na otimização multiobjetivo é identificar o conjunto de soluções de Pareto (ABRAHAM et al., 2005). Como foi descrito anteriormente, um algoritmo evolucionário é caracterizado por uma população de candidatos a solução e a presença dos operadores de reprodução habilita a combinação entre as soluções existentes para gerarem novas soluções. Como resultados, são encontradas diversas soluções ótimas de Pareto em uma única execução do algoritmo invés de executar várias avaliações separadamente, que é o caso de alguns dos processos estocásticos convencionais.

O principal desafio em uma otimização multiobjetivo, no presente contexto (abordagem evolucionária), é minimizar a distância entre as soluções geradas para a frente de Pareto e maximizar a diversidade de soluções de Pareto desenvolvidas (ABRAHAM et al., 2005). Boas soluções de Pareto são obtidas com a orientação correta no processo de busca através da elaboração cuidadosa dos operadores de reprodução e a forma das estratégias de consentimento. Para obter uma heterogeneidade, atenção especial deve ser dada ao processo de seleção. Cuidados especiais também devem ser dados para prevenir que soluções não-dominadas sejam perdidas (ABRAHAM et al., 2005).

Algoritmos multiobjetivos evolucionários podem render um conjunto de potenciais soluções ótimas. A primeira implementação do que hoje é chamado de algoritmo evolucionário multiobjetivo foi o algoritmo genético com avaliação vetorial (*Vector Evaluation Genetic Algorithm* – VEGA) de Schaffer, que foi introduzida em meados dos anos 80, visando resolver principalmente problemas no mecanismo de aprendizado artificial (SCHAFFER, 1984; SCHAFFER, 1985; SCHAFFER e GREFENSTETTE, 1985). Desde então, um montante considerável de pesquisas vêm sendo realizadas nessa área, agora conhecida como otimização evolucionária multiobjetiva (*evolutionary multiobjective optimization* - EMO).

A crescente importância deste campo está refletida na publicação significativa (principalmente nos últimos dez anos) de artigos em conferências internacionais, em revistas e livros (ABRAHAM et al., 2005).

Em grosso modo podemos dividir em três as estratégias para a otimização evolucionária multiobjetiva, tal como apresentada em (ABRAHAM et al., 2005):

- Funções agregadas (*Aggregating functions*);
- Abordagens populacionais (*Population-based approaches*);
- Abordagem de Pareto (*Pareto-based approaches*).

A seguir será apresentada uma breve explicação sobre cada uma destas estratégias.

### **3.3.1 Funções agregadas**

Talvez a mais simples abordagem para manusear múltiplos objetivos com qualquer técnica seja combinar todos os objetivos em um único objetivo, seja através de uma adição, uma multiplicação ou qualquer outra combinação de operações aritméticas.

Essas técnicas são normalmente conhecidas como “funções agregadas” porque elas combinam (ou “agregam”) todos os objetivos do problema em um único objetivo tornando o problema escalar. De fato, a abordagem de agregar é o mais antigo método de programação para otimização multiobjetivo (KUHN e TUCKER, 1951).

### **3.3.2 Abordagens populacionais**

Nestas técnicas, a população de um algoritmo evolucionário é usada para diversificar a busca, o conceito de dominância de Pareto não é diretamente incorporado no processo de seleção. O exemplo clássico deste tipo de abordagem é o algoritmo genético com avaliação vetorial, proposta por Schaffer (SCHAFFER, 1985). VEGA consiste basicamente de um algoritmo genético com um mecanismo de seleção modificado. Em cada geração, um número de sub-populações é gerada executando uma seleção proporcional de acordo com cada função objetivo por vez. Assim, para um problema com  $k$ -objetivos,  $k$  sub-populações, cada uma de tamanho  $M/k$ , são geradas (assumindo um tamanho total  $M$  para população) (ABRAHAM et al., 2005). Estas sub-populações são então misturadas para obtenção de uma nova população de tamanho  $M$ , onde são feitos os cruzamentos e aplicados os operadores de mutação do algoritmo genético.



VEGA apresenta alguns problemas, dentre eles o mais sério é que seu esquema de seleção se opõe ao conceito de dominância de Pareto. Se, por exemplo, existir um indivíduo que represente uma solução compromisso com relação a todos os objetivos, mas não é a melhor solução encontrada na população quando considerada a atuação isolada de cada objetivo, esta solução é descartada (ABRAHAM et al., 2005). Deve ser observado, no entanto, que tais indivíduos devem ser preservados porque eles representam uma solução ótima de Pareto. Schaffer sugeriu algo heurístico para lidar com esses problemas. Por exemplo, usar uma abordagem heurística preferencial para seleção de indivíduos não dominados em cada geração, preservando assim indivíduos que representam soluções que satisfazem o critério de Pareto (ABRAHAM et al., 2005). O cruzamento entre “espécies” podia ser incentivado através do acréscimo de seleções heurísticas de parceiro invés de usar uma seleção aleatória de parceiro como é feito no algoritmo genético tradicional (ABRAHAM et al., 2005). Não obstante, o fato da dominância de Pareto não estar diretamente incorporada no processo de seleção no algoritmo, sua principal desvantagem é mantida (ABRAHAM et al., 2005).

Um aspecto interessante do VEGA é que, apesar de suas desvantagens, ele ainda permanece sendo usado por muitos pesquisadores principalmente porque é mais apropriado para problemas onde se quer um processo de seleção tendenciosa e onde se deve lidar com um grande número de objetivos (ABRAHAM et al., 2005).

### **3.3.3 Abordagem de Pareto**

Tendo como base os principais inconvenientes do VEGA, Goldberg apresenta (GOLDBERG, 1989) outra abordagem para solucionar problemas multiobjetivos. O procedimento consiste num esquema de seleção baseado no conceito de otimalidade de Pareto. Goldberg não só sugeriu o que passaria a ser o padrão de algoritmo evolucionário multiobjetivo por muitos anos, mas também indicou que ruídos estocásticos fariam tais algoritmos inúteis a menos que algum mecanismo especial fosse adotado para bloquear a convergência. Um processo de separação (*niching*) com a partilha das funções de aptidão foi sugerida por Goldberg como uma maneira de manter a diversidade e impedir a convergência no algoritmo genético para uma única solução (ABRAHAM et al., 2005).

Abordagens de Pareto podem ser historicamente estudadas como abrangendo duas gerações. A primeira geração é caracterizada pelo uso das funções de aptidão e o processo de separação combinadas ao ranque de Pareto (como definido por Goldberg ou adotando uma ligeira variação) (ABRAHAM et al., 2005). Os algoritmos mais representativos desta primeira geração são os listados a seguir:

1. Triagem não-dominada no algoritmo genético (*Non-dominated Sorting Genetic Algorithm* – NSGA) (SRINIVAS e DEB, 1994);
2. Algoritmo genético com processo da separação de Pareto (*Niched-Pareto Genetic Algorithm* – NPGA) (HORN et al., 1994);
3. Algoritmo genético multiobjetivo (*Multiobjective Genetic Algorithm* – MOGA) (FONSECA e FLEMING, 1993).

A segunda geração de algoritmos evolucionários multiobjetivo surgiu com a introdução da noção de elitismo. No contexto da otimização multiobjetivo, elitismo normalmente (embora não necessariamente) se refere ao uso de uma população externa (também chamada de população secundária) para reter os indivíduos não-dominados. Contudo, o uso dessa população externa levanta algumas questões:

- Como a população secundária interage com a população principal?
- O que deve ser feito quando a população secundária estiver cheia?
- Deve ser imposto algum critério adicional para fazer parte da população secundária invés de usar apenas a dominância de Pareto?

É perceptível que também pode ser introduzido através da utilização de uma seleção na qual os pais competem com os filhos, e aqueles que são não-dominados (e possivelmente cumprem alguns critérios adicionais, tais como fornecer uma melhor distribuição das soluções) são selecionados para as gerações seguintes (ABRAHAM et al., 2005).

Os algoritmos mais representativos desta segunda geração de algoritmos evolucionários multiobjetivo, são os listados a seguir:

1. Algoritmo evolucionário da resistência de Pareto (*Strength Pareto Evolutionary Algorithm* – SPEA) (ZITZLER e THIELE, 1999);
2. Algoritmo evolucionário da resistência de Pareto 2 (*Strength Pareto Evolutionary Algorithm 2* – SPEA2) (ZITZLER et al., 2001);
3. Estratégia evolutiva de arquivo de Pareto (*Pareto Archived Evolution Strategy* – PAES) (KNOWLES e CORNE, 2000);

4. Triagem não-dominada no algoritmo genético II (*Non-dominated Sorting Genetic Algorithm II* – NSGA-II) (DEB et al., 2000<sup>1</sup>; DEB et al., 2000<sup>2</sup>; DEB et al., 2002);
5. Algoritmo genético com processo da separação de Pareto 2 (*Niched-Pareto Genetic Algorithm 2* – NPGA 2) (ERICKSON et al., 2001);
6. Algoritmo de seleção baseado no empacotamento de Pareto (*Pareto Envelope-based Selection Algorithm* – PESA) (CORNE et al., 2000);
7. Micro algoritmo genético (*Micro-Genetic Algorithm*) (COELLO e PULIDO, 2001<sup>1</sup>; COELLO e PULIDO, 2001<sup>2</sup>).

A otimização evolucionária multiobjetiva é uma área bastante promissora, porém é esperado que os algoritmos da próxima geração provavelmente visem apenas à melhoria da eficiência dos algoritmos já existentes.

# 4

## Otimização via Enxame de Partículas (*Particle Swarm Optimization* – PSO)

### 4.1 Introdução

A otimização via enxame de partículas foi idealizada originalmente por um psicólogo e um engenheiro em 1995 e desenvolveu-se de experiências com algoritmos que modelavam o comportamento de muitas espécies de pássaros (POMEROY, 2003).

Várias versões são apresentadas na literatura especializada para o algoritmo PSO. O funcionamento do algoritmo tomado como padrão para os estudos realizados será apresentado colocando em destaque a eficácia do mesmo na obtenção de soluções em problemas críticos de otimização (problemas com funções não diferenciáveis, problemas de otimização global, etc.).

Os procedimentos adotados para obtenção dos resultados serão descritos. As adaptações realizadas para adequar o algoritmo a problemas diversos também serão apresentadas, bem como técnicas aqui empregadas para melhoria da eficiência computacional.

## **4.2 Otimização via enxame de partículas**

### **4.2.1 Idéia geral**

O termo PSO se refere a uma família, relativamente nova, de algoritmos que podem ser usados para encontrar soluções otimizadas em problemas numéricos e qualitativos (POMEROY, 2003). O PSO foi originalmente desenvolvido por um psicólogo (James Kennedy) e um engenheiro elétrico (Russel Eberhart) em 1995 (KENNEDY e EBERHART, 1995) e foi inspirado em experimentos com algoritmos que modelavam o comportamento de revoada visto em muitas espécies de pássaros. Embora houvesse um número considerável de algoritmos com atenção na ocasião, Kennedy e Eberhart se tornaram particularmente interessados nos modelos desenvolvidos pelo biólogo Frank Heppner.

Os pássaros do modelo exibido por Heppner apresentavam o mesmo comportamento de revoada apresentado em outros modelos produzidos, mas Heppner adicionou algo diferente: os pássaros de Heppner eram atraídos para uma área de pouso, um poleiro. Na simulação, os pássaros começariam a voar sem um destino em particular e espontaneamente uma revoada estaria formada até que um dos pássaros voasse sobre uma área de pouso (POMEROY, 2003).

As regras que modelam o comportamento do bando de pássaros são simples: os pássaros escolhem a direção e a velocidade de vôo (essencialmente, cada pássaro tenta permanecer próximo aos outros enquanto também tenta não colidir com os mesmos); um dos pássaros se desvia do restante do bando com o intuito de pousar num poleiro, o que acarreta num movimento dos pássaros mais próximos para mesma direção; quando esses pássaros descobrem um poleiro, eles pousam conduzindo o restante do bando para o mesmo local. Dadas as especialidades dos pesquisadores, não é difícil perceber o que deixou Eberhart e Kennedy fascinado com os pássaros de Heppner (POMEROY, 2003).

Encontrar um poleiro é análogo a encontrar uma solução em um campo de possíveis soluções, e a maneira que uma das aves “leva” seus vizinhos em direção a ele, aumentando a probabilidade dos mesmos encontrá-lo, está ligada a cognição social da mente (a mente, e, portanto a inteligência, é social) que dá a idéia de convergência.

A essência (ou pelo menos um aspecto importante) da perspectiva da “mente é social” é o fato dos indivíduos aprenderem principalmente com o êxito de seus vizinhos (POMEROY, 2003). É como nos compararmos com outros e imitarmos o comportamento dos que obtêm sucesso naquilo que nos interessa. Pensando bem nisso, o que é requerido é um equilíbrio entre explorar (tipo 1: procurar uma solução individualmente) e explorar (tipo 2: tirar vantagem do sucesso de outros indivíduos) (POMEROY, 2003). Com pouca exploração do tipo 1, a convergência se dá na primeira solução boa que for encontrada. Com pouca exploração do tipo 2, nunca se dará a convergência (os indivíduos manterão a busca ou cada qual parará em qualquer lugar que bem entender).

Há outro modo de olhar isso, no lugar de explorar e explorar (comportamentos), o que deve ser convenientemente equilibrado é a individualidade e a socialização (características que influenciam o comportamento).

De maneira ideal, se quer que os indivíduos prefiram ser individualistas (semelhante aos pássaros de Heppner que não querem colidir uns nos outros) e também se deseja que os indivíduos queiram saber onde boas soluções têm sido encontradas pelos outros, então se pode “aprender com o que os outros fizeram”.

#### **4.2.2 Algoritmo básico**

O algoritmo de otimização via enxame de partículas, faz uso de um vetor velocidade para atualizar a posição de cada partícula. A posição de cada partícula é atualizada com base no comportamento social da população de indivíduos, o *swarm* (a movimentação amontoada de um grande número de insetos ou outras pequenas criaturas). O processo é estocástico por natureza e faz uso de uma memória de cada partícula bem como do conhecimento adquirido pela multidão como um todo.

O roteiro de um algoritmo básico do PSO é o descrito a seguir:

1. A população é inicializada com uma distribuição aleatória das partículas por todo o espaço de projeto;
2. O vetor velocidade para cada partícula é calculado;
3. A posição de cada partícula é atualizada usando a posição prévia e o vetor velocidade atualizado;
4. Checa-se a convergência;

5. Se a convergência foi atingida, o processo é interrompido; caso contrário retorna ao passo 2.

#### 4.2.3 Distribuição inicial

A distribuição inicial das partículas é dada de tal forma que os indivíduos estejam distribuídos aleatoriamente no espaço de projeto que é restringido pelos limites superiores e inferiores das variáveis de projeto. Cada partícula considerada representa um projeto a ser avaliado. De maneira semelhante à inicialização do vetor posição das partículas, o vetor velocidade de cada partícula é também inicializado aleatoriamente, tendo como limites os valores máximos e mínimos das mudanças locais.

O esquema utilizado para estas inicializações (posição e velocidade) é mostrado nas equações apresentadas a seguir.

$$\mathbf{x}_0^i = \mathbf{x}_{\min} + r_1 (\mathbf{x}_{\max} - \mathbf{x}_{\min}) \quad (4.1)$$

$$\mathbf{v}_0^i = \mathbf{v}_{\min} + r_2 (\mathbf{v}_{\max} - \mathbf{v}_{\min}) \quad (4.2)$$

Nas equações, anteriormente apresentadas,  $\mathbf{x}_0^i$  e  $\mathbf{v}_0^i$  são, respectivamente, o vetores posição e velocidade iniciais da partícula  $i$ ,  $r_1$  e  $r_2$  são números randômicos entre 0 e 1,  $\mathbf{x}_{\min}$  e  $\mathbf{x}_{\max}$  são, respectivamente, os vetores com os limites mínimos e máximos das posições (variáveis de projeto), e  $\mathbf{v}_{\min}$  e  $\mathbf{v}_{\max}$  são os vetores com os valores mínimos e máximos para mudanças locais na variável de projeto. Os valores limites das variáveis de projeto são valores determinados pelo usuário. Os valores limites das mudanças locais são determinados em função dos limites das variáveis de projeto como mostrado na equação (4.3) (INNOCENTE, 2006).

$$\mathbf{v}_{\min} = \frac{(\mathbf{x}_{\min} - \mathbf{x}_{\max})}{2 \cdot \Delta t} \text{ e } \mathbf{v}_{\max} = \frac{(\mathbf{x}_{\max} - \mathbf{x}_{\min})}{2 \cdot \Delta t} \quad (4.3)$$

Esta forma de determinar os valores limites das mudanças locais garante uma melhor busca no espaço de projeto. A determinação de tais limites pode ser feita pelo usuário, porém, o processo de busca pode ficar sujeito a falhas, como lentidão da saída das partículas de regiões não promissoras (caso os valores fornecidos fossem baixos demais) ou áreas promissoras não seriam investigadas (caso os valores fossem altos demais).

Devido às mudanças dinâmicas através do processo de otimização, como já mencionado, uma distribuição inicial precisa das partículas não é importante, contanto que a distribuição das partículas cubra todo o espaço de projeto em consideração.

#### 4.2.4 Atualização das variáveis

A atualização do vetor posição das partículas, mostrada a seguir, segue um esquema baseado na equação cinemática da função horária do espaço em um movimento uniforme.

$$\mathbf{x}_{t+1}^i = \mathbf{x}_t^i + \mathbf{v}_{t+1}^i \Delta t \quad (4.4)$$

Na equação (4.4),  $\mathbf{x}_{t+1}^i$  representa a posição atualizada da partícula  $i$  na iteração  $t+1$ ,  $\mathbf{x}_t^i$  representa a posição da partícula  $i$  na iteração  $t$  e  $\mathbf{v}_{t+1}^i$  representa a velocidade atualizada da partícula  $i$  na iteração  $t+1$ . O fator  $\Delta t$ , bem como na equação do movimento uniforme, representa o tempo. No caso do PSO, o tempo é representado pelas iterações, o que torna o valor de  $\Delta t$  unitário.

A atualização do vetor velocidade de cada partícula depende do algoritmo PSO em consideração. O esquema adotado (o que é normalmente utilizado) foi apresentado por Shi e Eberhart (SHI e EBERHART, 1998) e é mostrado na equação (4.5).

$$\mathbf{v}_{t+1}^i = w\mathbf{v}_t^i + c_1 r_1 \frac{(\mathbf{p}^i - \mathbf{x}_t^i)}{\Delta t} + c_2 r_2 \frac{(\mathbf{p}_t^b - \mathbf{x}_t^i)}{\Delta t} \quad (4.5)$$



Na equação anterior,  $r_1$  e  $r_2$  são números randômicos entre 0 e 1,  $v_t^i$  representa o vetor velocidade da partícula  $i$  na iteração  $t$ ,  $p^i$  representa a posição com a melhor aptidão encontrada pela partícula  $i$ , e  $p_t^b$  representa a posição com a melhor aptidão entre todas as partículas da população (VENTER e SOBIESZCZANSKI-SOBIESKI, 2002) e  $\Delta t$  representa o tempo (já definido anteriormente).

Na equação há três parâmetros relacionados ao tipo de problema, a inércia ( $w$ ) e os dois parâmetros de confiança ( $c_1$  e  $c_2$ ). A inércia influencia fortemente a capacidade de exploração do algoritmo: valores altos facilitam um comportamento mais global enquanto valores baixos um comportamento mais local (VENTER e SOBIESZCZANSKI-SOBIESKI, 2002). Os parâmetros de confiança indicam quanta confiança a partícula tem em si mesma ( $c_1$ ) e quanta confiança ela tem na população ( $c_2$ ) (VENTER e SOBIESZCZANSKI-SOBIESKI, 2002). A atuação combinada destes parâmetros está relacionada com a exploração do domínio de projeto, a inércia e a confiança em si próprio estão relacionadas com a exploração tipo 1 (procurar uma solução individualmente) e a confiança na população está relacionada com a exploração tipo 2 (tirar vantagem do sucesso de outros indivíduos).

Após a atualização dos vetores posição e velocidade, as coordenadas atualizadas são conferidas. Caso excedam os limites permitidos as coordenadas violadas assumem o valor dos limites impostos, por exemplo, se  $x_i > x_{i\_max}$  implica que  $x_i$  assumirá o valor  $x_{i\_max}$ .

#### 4.2.5 Atualização de parâmetros

Como parâmetros de confiança, a literatura (KENNEDY e EBERHART, 1995) propõe que seja adotado um equilíbrio entre a confiança da partícula em si próprio e na população, usando um valor igual para ambas. Kennedy e Eberhart (KENNEDY e EBERHART, 1995) adotam  $c_1 = c_2 = 2$ .

Alguns estudos recomendam uma variação nos parâmetros de confiança com a finalidade de investigar apenas o efeito de troca de importância da aprendizagem. A soma dos dois parâmetros de confiança recebe o nome de fator de aceleração (INNOCENTE, 2006). A consideração da variação dos parâmetros ao decorrer do processo de busca é limitada pelo fator de aceleração onde se recomenda que  $a = c_1 + c_2 = 4$  (INNOCENTE, 2006).

A atualização sugerida por (INNOCENTE, 2006) considera uma variação linear dos parâmetros de confiança com o passar do tempo, como apresentado na equação (4.6).

$$c_1^{new} = c_1^l + \frac{t}{t_{max}} \times (c_1^u - c_1^l) \quad (4.6)$$

onde  $c_1^{new}$ ,  $c_1^u$  e  $c_1^l$  são, respectivamente, o valor atualizado e os limites inferior e superior do parâmetro de confiança que o indivíduo tem em si próprio, e os valores de  $t$  e  $t_{max}$  representam, respectivamente, a iteração corrente e o número máximo de iterações permitidas no processo de otimização.

Conseqüentemente, pela determinação imposta com o fator de aceleração, o valor atualizado do parâmetro de confiança que o indivíduo tem na população é encontrado a partir da equação (4.7).

$$c_2^{new} = 4 - c_1^{new} \quad (4.7)$$

Adicionalmente aos valores sugeridos como valores de parâmetros de confiança, no que diz respeito ao parâmetro de inércia  $w$ , Shi e Eberhart (SHI, e EBERHART, 1998) sugerem usar  $0.8 < w < 1.4$ , começando o procedimento com um valor alto (garantindo um comportamento mais global à exploração) que é dinamicamente reduzido durante o processo de otimização. Eles sugeriram um decréscimo linear para inércia.

Venter e Sobieszczanski-Sobieski (VENTER e SOBIESZCZANSKI-SOBIESKI, 2002), sugerem uma diferente implementação, baseada no uso do coeficiente de variação do valor da função objetivo. A finalidade é mudar o valor da inércia de maneira independente do problema. Também é sugerido usar  $0.35 \leq w \leq 1.4$ , começando com  $w=1.4$  (para iniciar o processo com uma busca mais global) e dinamicamente reduzi-lo a um valor não menor do que 0.35. A idéia é terminar o processo com uma busca mais local. O valor de  $w$  é ajustado usando a equação (4.8).

$$w_{new} = w_{old} f_w \quad (4.8)$$

onde  $w_{new}$  é o valor ajustado de  $w$ ,  $w_{old}$  é o antigo valor de  $w$  e  $f_w$  é uma constante entre 0 e 1. Valores pequenos para  $f_w$  podem resultar numa redução dramática em  $w$ , que, por sua vez, resultaria numa busca mais local.  $f_w = 0.975$  é adotado nos problemas analisados, resultando numa característica de busca mais global (INNOCENTE, 2006).

O valor de  $w$  não é ajustado a cada iteração. Em vez disso, o coeficiente de variação ( $COV$ ) do valor da função objetivo de um subconjunto das melhores partículas da população é monitorado. Se o coeficiente de variação cai abaixo de um determinado valor limite (*threshold value*), é assumido que o algoritmo está convergindo na direção de uma solução ótima e a equação (4.8) é aplicada, caso contrário o valor de  $w$  permanece o mesmo (VENTER e SOBIESZCZANSKI-SOBIESKI, 2002). A equação geral para calcular a  $COV$  de um conjunto de pontos é fornecido pela equação (4.9).

$$COV = StdDev / Mean \quad (4.9)$$

Em estatística, coeficiente de variação é uma medida de dispersão e como mostrado anteriormente é dado pela razão entre o desvio padrão (*StdDev*) e a média (*Mean*). O subgrupo de partículas da população, que é monitorada, é formado por 20% das partículas de toda a população e o limiar adotado para o  $COV$  é um.

A idéia apresentada por Venter e Sobieszczanski-Sobieski (VENTER e SOBIESZCZANSKI-SOBIESKI, 2002) foi a idéia aqui adotada na implementação do algoritmo PSO para atualização do parâmetro de inércia.

#### 4.2.6 Consideração de restrições no problema

O algoritmo básico do PSO é definido para problemas irrestritos. Como a maioria dos problemas de engenharia apresenta restrições é importante acrescentar ao algoritmo a capacidade de lidar com tais funções. As metodologias utilizadas para resolver tais problemas, consistem na transformação dos problemas restritos em problemas irrestritos. Foi decidido lidar com problemas restritos através do método das funções de penalidade. Uma função de penalidade quadrática foi considerada. Esta técnica é freqüentemente usada no algoritmo genético (VENTER e SOBIESZCZANSKI-SOBIESKI, 2002). Em tais casos, a função objetivo é penalizada como mostrado na equação (4.10).

$$P(\mathbf{x}) = \bar{f}(\mathbf{x}) + \mu\alpha(\mathbf{x}) \quad (4.10)$$

A função objetivo é penalizada quando uma ou mais de uma das restrições é violada. No esquema anterior  $P(\mathbf{x})$  é a nova função objetivo (penalizada),  $\bar{f}(\mathbf{x})$  é a função objetivo normalizada do problema, como é apresentada na equação (2.3),  $\mu$  é o parâmetro de penalidade (é adotado  $\mu = 10^8$ ) e  $\alpha(\mathbf{x})$  é dado através da equação (4.11).

$$\alpha(\mathbf{x}) = \sum_{i=1}^{nr} \max[0; g_i(\mathbf{x})]^2 \quad (4.11)$$

Na equação anterior,  $\mathbf{g}(\mathbf{x})$  representa o conjunto de todas as restrições do problema em questão.

#### 4.2.7 Partículas com restrições violadas

Quando se trata com problemas restritos de otimização, uma atenção especial deve ser dada a atualização de partículas que violam as restrições. Um recurso adicional ao algoritmo básico do PSO deve ser construído tal como sugerido por Venter e Sobieszczanski-Sobieski (VENTER e SOBIESZCZANSKI-SOBIESKI, 2002).

Além da construção da função de penalidade, o vetor velocidade da partícula que violou alguma das restrições deve ser atualizado de forma a garantir a volta da mesma para o espaço viável do projeto. Uma alternativa apresentada é calcular uma direção viável, que seria uma direção que induziria a partícula a melhorar a função objetivo dentro do espaço viável do projeto. Infelizmente, isso requeria informações do gradiente de cada partícula. Um dos atrativos do algoritmo PSO é o fato de não serem necessárias informações sobre gradientes.

No lugar das informações sobre o gradiente é proposta uma modificação simples na atualização do vetor velocidade para partículas com restrições violadas. Considerando uma partícula com pelo menos uma restrição violada na iteração  $t$ , a velocidade dessa partícula na iteração  $t$  é reiniciada para o vetor nulo, e o vetor velocidade na iteração  $t+1$  (VENTER e SOBIESZCZANSKI-SOBIESKI, 2002) é obtido através da equação (4.12).

$$\mathbf{v}_{t+1}^i = c_1 r_1 \frac{(\mathbf{p}^i - \mathbf{x}_t^i)}{\Delta t} + c_2 r_2 \frac{(\mathbf{p}_t^b - \mathbf{x}_t^i)}{\Delta t} \quad (4.12)$$

A velocidade da partícula  $i$  na iteração  $t+1$  é então influenciada apenas pela melhor posição já encontrada por si mesma e pela melhor posição encontrada pela população até a presente iteração. Na maioria dos casos o novo vetor velocidade apontará em direção à região viável do espaço de projeto. O resultado é o retorno, da partícula com restrições violadas, para região viável do projeto.

#### 4.2.8 Função aptidão considerada

Como apresentado no capítulo 3, a definição da função aptidão é uma etapa que merece atenção especial. No algoritmo PSO considerado neste trabalho, dois tipos de funções aptidão são adotados, a depender do tipo de problema.

Para o caso de um problema de otimização irrestrita, a função aptidão considerada é a própria função objetivo. No caso de problemas de otimização com a consideração de restrições, a função aptidão considerada é a função  $P(\mathbf{x})$  apresentada na equação (4.10).

#### 4.2.9 Critérios de convergência

Critérios de convergência eficientes devem ser considerados em qualquer algoritmo de otimização. Tradicionalmente, métodos iterativos são dotados de algum critério de parada eficiente. Atender ao(s) critério(s) de convergência significa que cálculos adicionais não serão necessários após uma solução boa o suficiente ter sido encontrada, ou seja, quando uma melhoria significativa na solução encontrada é improvável. Um critério de parada eficiente tanto serve para diminuir o custo computacional como para aumentar a confiança na solução encontrada.

Técnicas tradicionais, adequadas a métodos tradicionais, envolvem (INNOCENTE, 2006):

- a diferença entre a melhor solução encontrada na iteração atual e a solução encontrada na iteração precedente;
- a distância entre as duas últimas posições (normalmente, a norma Euclidiana);
- um número limitado de iterações.

Contudo, estas técnicas não são convenientes para métodos de base populacional por muitas razões, algumas das quais são as seguintes (INNOCENTE, 2006):

- métodos de base populacional apresentam inúmeros candidatos à solução por iteração;
- a melhor solução encontrada na iteração corrente poderia permanecer imutável por algum tempo antes de sofrer uma melhoria;
- a melhor solução encontrada na corrente iteração e a encontrada na iteração precedente poderia corresponder a partículas diferentes, então a distância entre suas localizações não é exatamente uma medida de convergência.

A essência de base populacional do PSO permite a concepção de medidas que dão uma idéia de agregação que as partículas têm atingido até um determinado momento, que, por sua vez, dá uma idéia de convergência ao método.

Apesar do fato que a agregação de partículas não necessariamente significa a convergência do método, mais melhorias na melhor solução encontrada, após a implosão completa das partículas, são improváveis devido à perda de diversidade. Seguindo a metáfora de inspiração na psicologia social do método, isso indicaria que os indivíduos na população teriam chegado a um acordo, ou ainda, que a diversidade de crenças está perdida. As condições para finalizar o procedimento do PSO são propostas a seguir.

#### 4.2.9.1 Condições de finalização

Para condições de finalização do processo de otimização são considerados dois procedimentos, que são realizados com frequências diferentes. O primeiro procedimento é relacionado à agregação (realizado com uma frequência maior), e o segundo procedimento é relacionado ao monitoramento da melhor aptidão da população (realizado com uma frequência menor).

A descrição de cada um dos procedimentos será apresentada em seguida.

##### 4.2.9.1.1 Primeira condição de finalização

Como citado anteriormente, a primeira condição de finalização trata da agregação de partículas. Essa condição é dividida em duas relações:

1. Relação 1: essa relação avalia a razão da diferença entre a aptidão de cada partícula e a melhor aptidão encontrada pelo enxame nas últimas 100 iterações em relação à diferença entre a pior e a melhor aptidão encontrada, na iteração onde é realizada a verificação (INNOCENTE, 2006). Se essa razão for menor do que um valor determinado, a probabilidade de convergência aumenta. O esquema pode ser representado da seguinte forma:

$$\frac{\sum_{k=t-99}^t \frac{\sum_{i=1}^m (c_k^i - cgbest_k)}{m}}{100 \times (cgworst_t - cgbest_t)} = \frac{\sum_{k=t-99}^t (\bar{c}_k - cgbest_k)}{100 \times (cgworst_t - cgbest_t)} \leq 1 \times 10^{-12} \quad (4.13)$$

onde  $\bar{c}_k$  representa o valor médio das aptidões de todas as partículas,  $cgbest_k$  é a melhor aptidão encontrada na população e  $cgworst_k$  é a pior aptidão encontrada na população na iteração  $k$ ;

2. Relação 2: essa relação avalia a razão da diferença entre a atual melhor aptidão encontrada e a melhor aptidão encontrada na 100ª iteração precedente em relação à diferença entre a pior e a melhor aptidão encontrada na iteração onde é realizada a verificação (INNOCENTE, 2006). O esquema adotado é representado da seguinte forma:

$$\frac{abs(cgbest_t - cgbest_{t-99})}{100 \times (cgworst_t - cgbest_t)} = \frac{cgbest_{t-99} - cgbest_t}{100 \times (cgworst_t - cgbest_t)} \leq 1 \times 10^{-15} \quad (4.14)$$

onde  $cgbest_{t-99}$  é o valor da melhor aptidão encontrada na iteração  $t - 99$ .

Apesar de a primeira condição ser avaliada com mais frequência, ela não é avaliada em todas as iterações. Para realização das verificações da primeira condição de finalização, é necessário que:

- tenha acontecido, no mínimo, 10% do máximo de iterações a serem realizadas;
- seja dado um intervalo entre duas verificações consecutivas.

Nos estudos realizados, foi considerado um intervalo de 50 iterações entre duas avaliações consecutivas.

#### 4.2.9.1.2 Segunda condição de finalização

A segunda condição de finalização trata da verificação das mudanças observadas na melhor aptidão da população. Nessa condição, é conferido se ocorreu alguma melhoria significativa na melhor função aptidão encontrada até o momento, num determinado número de iterações.

A verificação segue o seguinte esquema:

$$cgbest_{t-0,25 \cdot t_{\max}} - cgbest_t = 0 \quad (4.15)$$



onde  $t_{\max}$  é o número máximo de iterações a serem realizadas e  $cgbest_{t-0,25 \cdot t_{\max}}$  é a melhor aptidão encontrada na iteração  $t - 0,25 \cdot t_{\max}$ .

Essa condição é avaliada com uma frequência menor, como já mencionado antes, e do mesmo modo que a primeira condição de finalização, alguns critérios precisam ser atendidos antes da realização da segunda avaliação.

Um dos critérios, para realização da segunda avaliação, está ligado diretamente à primeira condição de finalização. Para realização da verificação da segunda condição de finalização, é necessário que:

- tenha acontecido, no mínimo, 25% do número máximo de iterações a serem realizadas;
- dez avaliações, da primeira condição de finalização, sejam realizadas sem serem atendidas.

Deve ser observado que ambas as condições justificam o encerramento do processo iterativo. Se a primeira for atingida, implica que as partículas atingiram um elevado grau de agregação, e que a taxa de melhoria da solução atingiu um limiar inferior admissível. Cumprindo a segunda, implica que, embora as partículas ainda não tenham atingido um elevado grau de agregação, uma melhoria significativa na melhor solução encontrada é improvável (INNOCENTE, 2006).

### 4.3 Otimização de múltiplos objetivos com o PSO

No capítulo anterior, foram apresentadas as vantagens da utilização dos algoritmos evolucionários em problemas de otimização multiobjetivo, neste capítulo, será tratada a eficiência do algoritmo de otimização via enxame de partículas na resolução de tais problemas.

Serão mostradas as principais técnicas utilizadas para que a otimização multiobjetivo via enxame de partículas pudesse ser realizada, além das modificações no algoritmo tradicional para adaptá-lo a tais técnicas.

#### **4.3.1 Técnicas adotadas para otimização multiobjetivo via enxame de partículas**

Várias técnicas já são conhecidas para resolução de problemas de otimização evolucionária multiobjetivo, em particular, as que são utilizadas nos algoritmos genéticos. Adaptações de algumas dessas técnicas para a metodologia do PSO vêm sendo estudadas por vários pesquisadores interessados na área. Aqui serão apresentadas três técnicas, as quais foram estudadas e adotadas nos procedimentos de otimização multiobjetivo via enxame de partículas.

##### **4.3.1.1 Agregação evolucionária com ponderação dinâmica (*Evolutionary Dynamic Weighted Aggregation – EDWA*)**

Um método eficiente e eficaz, chamado agregação evolucionária com ponderação dinâmica, é proposta em (JIN et al., 2001). A idéia original do EDWA (JIN et al., 2001) é direta, i.e., pesos são atribuídos às funções objetivos (normalizadas) que por sua vez são agregadas em uma única função objetivo.

Os pesos para os diferentes objetivos são modificados durante a otimização. Com isso, o otimizador irá percorrer todas as posições da frente de Pareto. O EDWA (JIN et al., 2001) é uma modificação da metodologia convencional de agregação com ponderação (*Conventional Weight Aggregation - CWA*), que visa aproveitar a diversidade de soluções apresentadas em cada iteração.

No CWA é necessário efetuar uma otimização para obtenção de cada solução da frente de Pareto. Isso não é aceitável em muitos dos problemas do mundo real porque normalmente é gasto muito tempo para executar várias otimizações. A eficiência não é o único problema da CWA, deve ser salientado que soluções localizadas em regiões não-convexas da frente de Pareto não são encontradas.

Contudo, não é tão simples como se poderia imaginar explicar os motivos que impedem as soluções localizadas na região não-convexa da frente de Pareto de serem obtidas pela CWA. Uma teoria bastante interessante é sugerida por Jin et al. (JIN et al., 2001).

A CWA é capaz de obter soluções ótimas de Pareto, dependendo da estabilidade da solução de Pareto correspondente a combinação de pesos dados. Se a solução é um mínimo estável, então a solução pode ser encontrada pela CWA. Para dar uma explicação sobre essa estabilidade, será considerado um problema de otimização com dois objetivos para facilitar a visualização.

Se a combinação de pesos for  $(w_1, w_2) = (0,1)$  a frente de Pareto é apresentada na forma convencional, como mostrado na figura (4.1a), e o ponto B é o mínimo estável na frente de Pareto. Isto é, o ponto B será a solução obtida pela CWA. A modificação da combinação de pesos na otimização é equivalente a rotacionar o sistema de coordenadas junto com a frente de Pareto. Deste modo, quando  $w_1$  é diminuído e  $w_2$  é aumentado, é o mesmo que rotacionar o sistema de coordenadas no sentido anti-horário. Se a combinação dada for  $(w_1, w_2) = (0.5, 0.5)$ , implica dizer que o sistema de coordenadas é rotacionado em  $45^\circ$  (o ângulo de rotação do sistema é determinado por  $\tan^{-1}(w_1/w_2)$ ). Nesse caso, C é o mínimo estável da frente de Pareto que será obtido, como é mostrado na figura (4.1b). Obviamente, se a combinação dada for  $(w_1, w_2) = (1, 0)$ , A é o mínimo estável e o sistema de coordenadas é rotacionado em  $90^\circ$  como é mostrado na figura (4.1c).

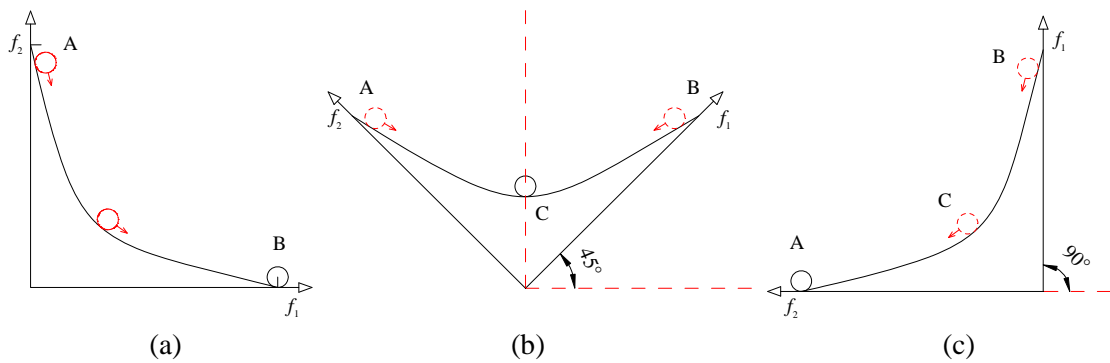


Figura (4.1) – Frente de Pareto convexa. (a) Sistema rotacionado em  $0^\circ$ ; (b) Sistema rotacionado em  $45^\circ$ ; (c) Sistema rotacionado em  $90^\circ$ .

Por essa razão, diferentes soluções de Pareto são obtidas usando a metodologia convencional de agregação com ponderação com diferentes combinações de pesos, se a frente de Pareto é convexa e desde que os pesos sejam sempre positivos e a máxima rotação seja  $90^\circ$ . Desconsiderando o tempo utilizado, toda a frente de Pareto pode ser obtida executando as otimizações quantas vezes forem necessárias (JIN et al., 2001).

Considerando agora, a frente de Pareto não-convexa, como é apresentado na figura (4.2a), todas as soluções localizadas na região não-convexa são instáveis quando a combinação de pesos é modificada. Semelhante a situação anterior, a figura (4.2a) corresponde à combinação de pesos  $(w_1, w_2) = (0,1)$  e a solução encontrada será o ponto B. Para todas as combinações que correspondem à rotação do sistema de coordenadas entre  $0^\circ$  e  $45^\circ$ , a solução obtida será B, enquanto que para as combinações correspondentes a rotação do sistema entre  $45^\circ$  e  $90^\circ$ , a solução obtida será A. A combinação de pesos que corresponde ao sistema rotacionado em  $45^\circ$ , como mostrado na figura (4.2b), o resultado da otimização pode ser ou A ou B, dependendo da condição inicial e da dinâmica do otimizador. Conclui-se que, somente A e B são mínimos estáveis na frente de Pareto não importando qual combinação de pesos é adotada (JIN et al., 2001).

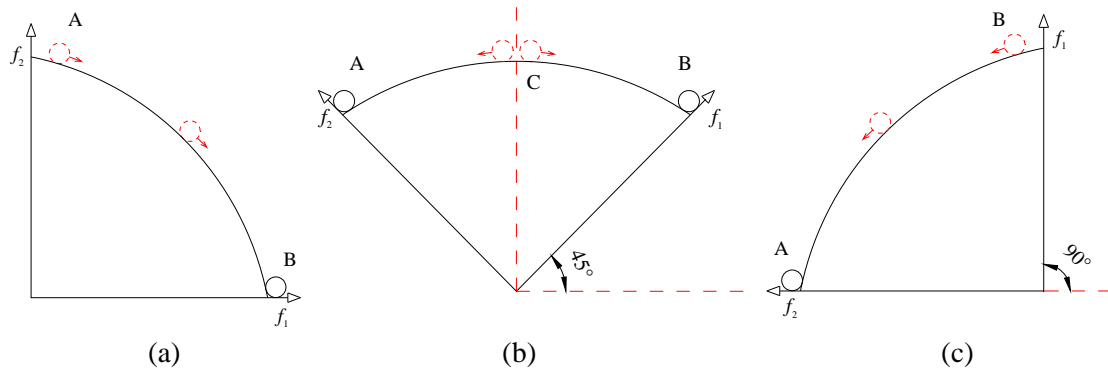


Figura (4.2) – Frente de Pareto não-convexa. (a) Sistema rotacionado em  $0^\circ$ ; (b) Sistema rotacionado em  $45^\circ$ ; (c) Sistema rotacionado em  $90^\circ$

Como foi apontado anteriormente, se for realizada uma rotação de  $90^\circ$  na frente de Pareto, o otimizador irá de um ótimo estável para outro. Para eliminar os inconvenientes anteriormente apontados, foi criada uma versão melhorada do CWA aqui chamada de agregação evolucionária com ponderação dinâmica (*Evolutionary Dynamic Weight Aggregation* - EDWA) (JIN et al., 2001). A diferença básica é aplicação da rotação do sistema de coordenadas, que foi descrita anteriormente, de forma suave, ou seja, os pesos podem ser modificados gradualmente para forçar o otimizador a se manter em movimento na frente de Pareto. Nesse caso todas as soluções na frente de Pareto podem ser obtidas (JIN et al., 2001).

Para uma frente de Pareto convexa, a mudança dinâmica nos pesos não é tão necessária porque a frente de Pareto não é o menor caminho viável entre duas soluções estáveis (JIN et al., 2001), como é mostrado na figura (4.3a). Já no caso da frente de Pareto não-convexa, a estratégia de mudanças de coordenadas de forma dinâmica deve ser mantida porque a frente de Pareto fornece o menor caminho viável de uma solução para outra (JIN et al., 2001), como ilustrado na figura (4.3b).

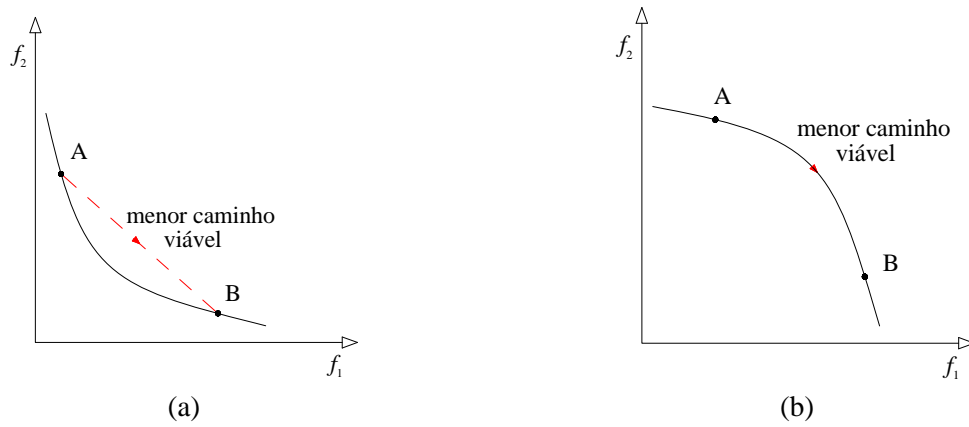


Figura (4.3) – Menor caminho viável entre duas soluções de Pareto. (a) Quando a frente de Pareto é convexa; (b) Quando a frente de Pareto é não-convexa.

A mudança dinâmica nos pesos é caracterizada nas equações apresentadas a seguir (para um problema de minimização com dois objetivos):

$$w_1(t) = |\sin(2\pi t/F)| \quad (4.16)$$

$$w_2(t) = 1.0 - w_1(t) \quad (4.17)$$

É perceptível que o  $w_1(t)$  varia de 0 até 1 periodicamente. Nas equações apresentadas acima,  $t$  é o iterando e  $F$  é a frequência das mudanças nos pesos. A frequência  $F$  não deve ser tão elevada para que o algoritmo seja capaz de convergir para um ponto na frente de Pareto. Por outro lado, parece razoável deixar os pesos mudarem de 0 para 1 pelo menos duas vezes durante todo processo de otimização (JIN et al., 2001).

#### **4.3.1.2 Otimização via enxame de partículas com avaliação vetorial (*Vector Evaluated Particle Swarm Optimization - VEPSO*)**

A otimização via enxame de partículas com avaliação vetorial, é uma variação do PSO, inspirada no algoritmo genético com avaliação vetorial (VEGA) (SCHAFFER, 1984; SCHAFFER, 1985; SCHAFFER e GREFENSTETTE, 1985).

No VEGA, frações da próxima geração ou sub-populações são selecionadas da geração precedente de acordo com os objetivos, separadamente. Depois que todas essas sub-populações são embaralhadas juntas, o cruzamento e a mutação são aplicados para gerar a nova população. Estas idéias foram adotadas e modificadas para se ajustarem a estrutura do PSO.

O algoritmo do VEPSO é conceitualmente simples. É semelhante a utilizar uma população para otimizar cada função objetivo proposta no problema. A questão-chave nesse algoritmo é o fato da aptidão de um indivíduo numa população depender de um indivíduo de outra população. Isso reforça a capacidade do algoritmo de melhor estudar e explorar o espaço de busca para assim detectar com mais precisão frentes de Pareto convexas, não-convexas, parcialmente convexas e/ou não-convexas e/ou descontínuas (SCHAFFER, 1984; COELLO e SIERRA, 2004). As principais características do VEPSO são explicadas em pormenores a seguir.

A metodologia do VEPSO assume que  $M$  populações ( $S_1, S_2, \dots, S_M$ ), com  $N$  indivíduos cada, são designadas a otimizar simultaneamente  $M$  funções objetivo. Como foi citado anteriormente, cada população é avaliada exclusivamente de acordo com uma das funções objetivos. No processo de otimização informações sobre a busca são trocadas entre populações, essas informações são usadas para influenciar a comoção no espaço de busca (OMKAR et al., 2007).

Na avaliação de cada população, todos os parâmetros de uma otimização tradicional via enxame de partículas são considerados (OMKAR et al., 2007). A inicialização de cada população (determinação do vetor posição e do vetor velocidade) é feita aleatoriamente como é mostrado nas equações seguintes:

$$\mathbf{x}_0^{i[j]} = \mathbf{x}_{\min} + r_1(\mathbf{x}_{\max} - \mathbf{x}_{\min}) \quad (4.18)$$

$$\mathbf{v}_0^{i[j]} = \mathbf{v}_{\min} + r_1(\mathbf{v}_{\max} - \mathbf{v}_{\min}) \quad (4.19)$$

onde  $\mathbf{x}_0^{i[j]}$  é o vetor posição e  $\mathbf{v}_0^{i[j]}$  é o vetor velocidade da partícula  $i$  ( $i = 1 \dots N$ ) da população  $j$  ( $j = 1 \dots M$ ).

Anteriormente foi comentado que a questão chave do algoritmo está no fato das aptidões dos indivíduos dependerem de indivíduos de populações diferentes. Essa dependência é criada na atualização do vetor velocidade (OMKAR et al., 2007). Uma mudança na atualização do vetor velocidade foi implementada e é apresentada na equação (4.20).

$$\mathbf{v}_{t+1}^{i[j]} = \chi \cdot \left\{ w\mathbf{v}_t^{i[j]} + \frac{c_1 r_1 (\mathbf{p}^{i[j]} - \mathbf{x}_t^{i[j]})}{\Delta t} + \frac{c_2 r_2 (\mathbf{p}_t^{b[s]} - \mathbf{x}_t^{i[j]})}{\Delta t} \right\} \quad (4.20)$$

Na equação apresentada anteriormente,  $\mathbf{v}_{t+1}^{i[j]}$  é a velocidade atualizada da partícula  $i$  da população  $j$  na iteração  $t+1$ ,  $\mathbf{v}_t^{i[j]}$  e  $\mathbf{x}_t^{i[j]}$  são, respectivamente, a velocidade e a posição da partícula  $i$  da população  $j$  na iteração  $t$  e  $\mathbf{p}^{i[j]}$  é a melhor posição encontrada pela partícula  $i$  da população  $j$ . O termo  $p_t^{b[s]}$  representa a posição da melhor aptidão entre todas as partículas da população  $s$  (onde  $s = 1 \dots M$  e  $s \neq j$ ), e é o responsável pela troca de informações entre populações, fazendo com que as partículas da população  $j$  sejam guiadas também em direção à melhor solução encontrada para a função objetivo  $s$ .

O fator  $\chi$  representa o fator de constrição e pode ser selecionado de várias maneiras, resultando em diferentes esquemas de migração de informações entre as múltiplas populações (OMKAR et al., 2007). O fator  $\chi$  é um tipo de confiança que as partículas da população  $j$  têm na velocidade atualizada influenciada por  $\mathbf{p}_t^{b[s]}$ . O algoritmo foi implementado considerando  $0.7 \leq \chi \leq 1.5$ .

#### **4.3.1.3 Otimização multiobjetivo via enxame de partículas considerando o critério de dominância (*Multiobjective Particle Swarm Optimization – MOPSO*)**

Fazer uma analogia do PSO com algoritmos evolucionários multiobjetivo torna clara a idéia de que utilizar o esquema de ranque de Pareto poderia ser o caminho mais direto para estender a abordagem tornando-a capaz de lidar com problemas de otimização multiobjetivo (COELLO et al., 2004). Um histórico com as melhores aptidões encontradas pelas partículas (i.e., pelos indivíduos) poderia ser usado para armazenar as soluções não-dominadas geradas no algoritmo multiobjetivo (isto seria similar a noção de elitismo usada no GA) (COELLO et al., 2004). O uso do mecanismo de atração global combinado ao histórico de soluções não-dominadas previamente encontradas motivaria a convergência dos indivíduos em direção às soluções não-dominadas (COELLO et al., 2004). O algoritmo principal do MOPSO surgiu com a incorporação dessa idéia ao algoritmo PSO convencional.



O algoritmo PSO para consideração de uma única função objetivo não foi modificado na sua essência, porém, algumas adaptações foram feitas para realização das tarefas de acordo com as idéias sugeridas por Coello et al. (COELLO et al., 2004). Uma adaptação implementada foi a criação de um repositório que armazena as soluções não-dominadas encontradas, outra foi uma modificação na atualização da velocidade e por fim uma consideração após a atualização da posição das partículas foi definida.

O repositório corresponde ao arquivo histórico com soluções não-dominadas encontradas ao longo do processo de otimização. As partículas são avaliadas, de acordo com o critério de dominância de Pareto, e as que apresentam  $rank = 1$  são armazenadas no repositório. A atualização do repositório é feita a cada iteração. Para atualizar o repositório, o processo de avaliação é realizado entre as partículas, já atualizadas, e as partículas do repositório, caso haja alguma partícula do repositório que seja dominada por uma partícula externa, a mesma é eliminada do repositório dando lugar à outra. O tamanho do repositório pode aumentar ou diminuir em cada iteração.

A adaptação feita na atualização da velocidade tem o intuito de tornar a atualização da velocidade das partículas influenciada pelas soluções do repositório, tal como representado na equação (4.21).

$$\mathbf{v}_{t+1}^i = w\mathbf{v}_t^i + c_1r_1 \frac{(\mathbf{p}^i - \mathbf{x}_t^i)}{\Delta t} + c_2r_2 \frac{(\mathbf{p}^{rep} - \mathbf{x}_t^i)}{\Delta t} \quad (4.21)$$

Na equação (4.21),  $\mathbf{p}^{rep}$  representa uma solução não-dominada, selecionada aleatoriamente, do repositório, as demais variáveis já foram anteriormente descritas.

Após a atualização dos vetores posição e velocidade, as coordenadas são verificadas, como citado anteriormente. A consideração de verificação do vetor posição, diz respeito à violação das coordenadas do mesmo. Quando essas excedem os limites, além de assumirem os valores limites, a velocidade é multiplicada por  $(-1)$ , direcionando a busca da partícula para o sentido oposto ao da violação dos limites.

A atualização das posições de cada partícula é feita da maneira convencional como descrito na equação (4.4).

### 4.3.2 Critério de convergência na otimização de múltiplos objetivos

A criação do repositório no MOPSO foi a modificação mais significativa no PSO para resolver problemas com múltiplos objetivos. Apesar de não ser diretamente utilizado em todas as técnicas, é de grande auxílio, pois separa as soluções não-dominadas do resto da população e define critérios para convergência do processo iterativo.

A condição definida para finalização do processo de otimização diz respeito ao número de soluções encontradas no repositório. Uma quantidade mínima é estipulada, caso essa não seja atingida e o número de partículas, no repositório, não seja modificado por um determinado período, a convergência é assumida e a frente de Pareto é definida com as soluções encontradas no repositório.

Da mesma maneira que a versão para problemas uni-objetivo, o teste de convergência não é realizado em todas as iterações do processo, ele é realizado de acordo com os critérios para realização da primeira condição de finalização, com uma pequena modificação considerada, o intervalo entre duas avaliações consecutivas após a realização de 10% do máximo de iterações permitidas foi reduzido de 50 para 10.

## 4.4 Melhoria da eficiência computacional

O algoritmo PSO é bastante eficaz, porém devido ao alto número de avaliações de funções que realiza nos processos se faz necessário o uso de medidas que melhorem sua eficiência. No presente trabalho estudaram-se duas medidas para providenciar essa melhoria: a substituição da avaliação das funções reais por funções substitutas calculadas através do Método das Bases Reduzidas (*Reduced Basis Method* - RBM) (AFONSO e PATERA, 2003); e a implementação do algoritmo original em uma versão paralelizada (Parallel\_PSO).

A programação paralela foi utilizada para os diversos problemas onde o PSO foi aplicado, já o RBM foi utilizado apenas em problemas de otimização estrutural. A seguir serão detalhados os procedimentos de adaptação do PSO a cada uma das metodologias.

#### 4.4.1 Método das bases reduzidas

Na engenharia, os problemas abordados têm se tornado cada vez mais realistas, no entanto, o projeto ótimo de problemas reais quase sempre leva a problemas complexos de otimização. Frequentemente, os parâmetros de interesse de um problema não são relativos ao próprio campo de solução das equações governantes, mas a alguma grandeza que depende deste campo, além disso, muitas vezes a obtenção destas soluções é demasiadamente cara e/ou o número de soluções é bastante elevado (como no caso do uso de estratégias evolucionárias), podendo até tornar inviável a resolução do problema. Desta forma, o uso das ferramentas de otimização pode ser desmotivado devido ao imenso esforço computacional requerido.

Neste contexto, o denominado Método das Bases Reduzidas será aqui considerado. A metodologia consiste numa projeção do tipo Galerkin em um espaço de ordem reduzida que contém soluções (bases) para o problema de interesse em pontos selecionados do espaço de projeto (AFONSO e PATERA, 2003; ALBUQUERQUE, 2005).

A implementação numérica do RBM consiste de dois estágios: o primeiro é um estágio de pré-processamento, também chamado de estágio *off-line*, no qual a base e as funções associadas são computadas em um conjunto de pontos prescritos no espaço de parâmetros do projeto; e o segundo é um estágio de tempo real, também chamado de estágio *on-line*, no qual as grandezas de interesse são computadas para um novo valor de parâmetros (AFONSO e PATERA, 2003; ALBUQUERQUE, 2005). No segundo estágio a análise do erro da solução aproximada é também conduzida para verificar a precisão da solução.

O segundo estágio é de custo computacional relativamente baixo, uma vez que requer a solução de sistemas de equações de ordem muito inferior do que àqueles do problema original.

##### 4.4.1.1 Observação crítica

A idéia central do RBM (PRUD'HOMME, 2002; AFONSO e PATERA, 2003) é o reconhecimento de que o campo de variáveis depende de um espaço menor de soluções.

Unir eficiência e confiança envolve vários atributos, a saber: redução de dimensão, operações de separabilidade (AFONSO e PATERA, 2003) (dividindo um domínio  $\mathbb{V}^*$  em várias sub-regiões  $r$  de domínio  $\mathbb{V}^{*r}$ ), um estimador de erro e uma estratégia computacional *off-line/on-line* eficaz (AFONSO et al., 2006).

Para aproximar o campo de soluções e conseqüentemente as grandezas dependentes destes, não é necessário representar todas as funções do espaço infinito-dimensional associado à equação governante, mas sim um espaço com dimensões muito menores induzido pela dependência paramétrica (AFONSO et al., 2006).

Se imaginarmos  $\mathbb{Y}$  como um espaço tri-dimensional, então  $\mathbf{u}$  - como uma função de  $\boldsymbol{\mu}$  - pode ser representado por uma curva ou superfície conforme a figura (4.4a). A partir daí, pode-se afirmar que o espaço de interesse não se trata de um espaço infinito-dimensional e sim de um espaço reduzido onde se observa a dependência do parâmetro variável  $\boldsymbol{\mu}$  (PRUD'HOMME et al., 2002; AFONSO e PATERA, 2003;) como ilustrado na figura (4.4b).

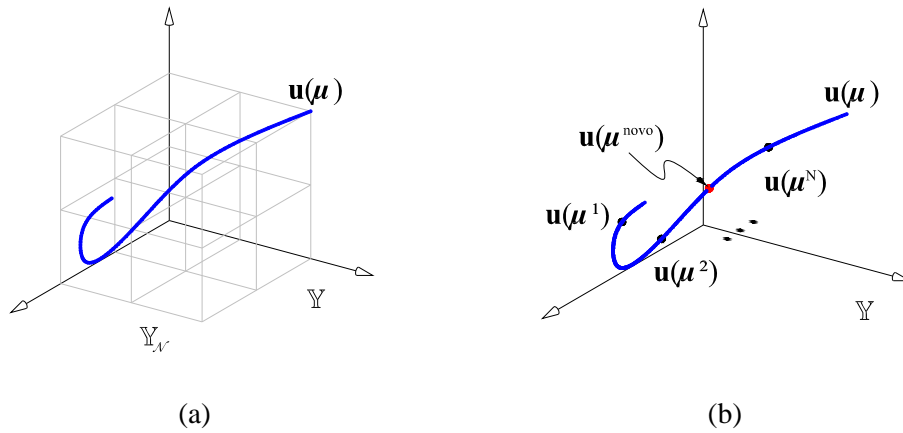


Figura (4.4) – Representação do espaço de soluções: (a) Variação da solução com o parâmetro  $\boldsymbol{\mu}$  ; e (b) aproximação da solução em  $\boldsymbol{\mu}^{Novo}$  através de uma combinação linear de soluções  $\mathbf{u}(\boldsymbol{\mu}^i)$  pré-calculadas.

#### 4.4.1.2 Preliminares

O RBM será aplicado a problemas lineares elásticos em treliças. A equação governante pode ser obtida utilizando a aproximação tipo Galerkin e aplicando o procedimento do método dos elementos finitos (*Finite Elements Method* – FEM) (COOK, 1981; ZIENKIEWICZ e TAYLOR, 2000), considerando  $\boldsymbol{\mu}$  (que representa as variáveis de projeto) pertencente a um espaço  $\mathbb{D}^\mu$ , tem-se:

$$\mathbf{K}(\boldsymbol{\mu})\mathbf{u}(\boldsymbol{\mu}) = \mathbf{F} \quad (4.22)$$

onde o vetor deslocamento  $\mathbf{u}(\boldsymbol{\mu}) \in \mathbb{R}^n$ , a matriz de rigidez  $\mathbf{K}(\boldsymbol{\mu}) \in \mathbb{R}^{n \times n}$  e o vetor de carregamentos  $\mathbf{F} \in \mathbb{R}^n$  tendo  $n$  como o total de graus de liberdade da treliça.

A energia de deformação da estrutura é uma grandeza que não pertence ao campo de soluções da equação governante, mas é uma grandeza de interesse, e depende de  $\mathbf{u}(\boldsymbol{\mu})$  como definido na equação (4.23).

$$s(\boldsymbol{\mu}) = \frac{1}{2} \mathbf{u}(\boldsymbol{\mu})^T \cdot \mathbf{F}(\boldsymbol{\mu}) \quad (4.23)$$

Na presente aplicação, cada parâmetro  $\boldsymbol{\mu}$  representará a rigidez axial ( $EA/L$ ) específico para cada região  $r$  da treliça, definido *a priori*. No processo de otimização, cada região é definida por uma única variável de projeto e um esquema de associação (*linking rule*).

Para não corromper o processo de otimização, a meta é construir uma aproximação para os deslocamentos e conseqüentemente para energia de deformação satisfazendo os requisitos de eficiência e de confiança. A primeira exigência será obtida aplicando uma decomposição para o problema original (computacionalmente caro) enquanto a segunda será realizada através de cálculos de um estimador de erro *a posteriori*.

Para realizar cálculos “rápidos”, a matriz de rigidez  $\mathbf{K}(\boldsymbol{\mu})$  é decomposta na forma apresentada da seguinte forma:

$$\mathbf{K} = \sum_{r=1}^R \mu_r \mathbf{K}_r \quad (4.24)$$

no qual  $R$  é o número total de regiões ou sub-domínios da treliça e  $\mathbf{K}_r$  representa a matriz de rigidez (global) restrita a região geométrica  $r$  e é independente de  $\boldsymbol{\mu}$ .

Como será apresentado mais a frente, a descrição dada anteriormente permite a implementação de dois estágios *off-line/on-line* no algoritmo computacional.

#### 4.4.1.3 Aproximação

Como citado anteriormente, para chegar até  $\mathbf{u}(\boldsymbol{\mu})$ , e conseqüentemente a  $s(\boldsymbol{\mu})$ , não é mais necessário representar todas as possíveis funções em  $\mathbb{Y}$ , apenas é necessário aproximar essas funções no espaço reduzido  $\mathbb{Y}_{\mathcal{N}}$ . Pode-se, por esta razão, simplesmente calcular a solução  $\mathbf{u}$  em vários pontos do espaço, correspondentes a diferentes combinações de parâmetros em  $\boldsymbol{\mu}$  e então, para qualquer  $\boldsymbol{\mu}^{novo}$  encontrar o  $\mathbf{u}(\boldsymbol{\mu}^{novo})$  através de uma combinação linear das soluções conhecidas.

Para construir uma aproximação do campo de soluções, inicialmente deve-se construir a amostra de pontos sob a qual será montada a base. O conjunto de amostras é representado da seguinte forma:

$$\mathcal{S}^N = \{(\mu_1, \dots, \mu_R)^1, \dots, (\mu_1, \dots, \mu_R)^N\} \quad (4.25)$$

onde cada  $(\mu_1, \dots, \mu_R)^i = \boldsymbol{\mu}^i$  está contido no espaço  $\mathbb{D}^\mu$  (isto significa que  $\mu_r^{low} \leq \mu_r^i \leq \mu_r^{up}$  no qual  $\mu_r^{low}$  e  $\mu_r^{up}$  são, respectivamente, os limites inferiores e superiores do espaço de projeto  $\mathbb{D}^\mu$ ) e  $N$  é o número de amostras.

Para cada componente de  $\mathcal{S}^N$  calcula-se a solução  $\mathbf{u}(\boldsymbol{\mu})$  através do FEM, utilizando a equação (4.22) que permite a construção de um espaço de base reduzida  $\mathbb{W}^N$  representado por:

$$\mathbb{W}^N = span\{\mathbf{u}(\boldsymbol{\mu}^1), \dots, \mathbf{u}(\boldsymbol{\mu}^N)\} \quad (4.26)$$

Para simplificar a notação é definido  $\zeta^i \in \mathbb{R}^n$  como:

$$\zeta^i = \mathbf{u}(\boldsymbol{\mu}^i) \quad (4.27)$$

tornando possível a representação de  $\mathbb{W}^N$  através de:

$$\mathbb{W}^N = \text{span}\{\zeta^i\} \text{ com } i=1 \dots N \quad (4.28)$$

A equação anterior significa que cada componente de  $\mathbb{W}^N$  pode ser expresso como uma combinação linear das soluções  $\zeta^i$ .

Tomando a definição acima, o problema aproximado com bases reduzidas pode ser formulado como apresentado a seguir.

Para  $\boldsymbol{\mu} \in \mathbb{D}$  encontrar:

$$s^N(\boldsymbol{\mu}) = (\mathbf{u}^N(\boldsymbol{\mu}))^T \cdot \mathbf{F} \quad (4.29)$$

como aproximação de  $s(\boldsymbol{\mu})$ , onde  $\mathbf{u}^N(\boldsymbol{\mu})$  é a projeção de Galerkin de  $\mathbf{u}(\boldsymbol{\mu})$  em  $\mathbb{W}^N$ . Por esta razão, levando em conta as observações feitas anteriormente, a aproximação com base reduzida para os deslocamentos  $\mathbf{u}^N(\boldsymbol{\mu})$ , é expressa de acordo com a equação (4.30):

$$\mathbf{u}^N(\boldsymbol{\mu}) = \sum_{i=1}^N \alpha^i(\boldsymbol{\mu}) \cdot \zeta^i \text{ com } \alpha^i \in \mathbb{R} \quad (4.30)$$

ou na forma matricial, de acordo com a equação (4.31):

$$\mathbf{u}^N(\boldsymbol{\mu}) = \mathbf{Z}\boldsymbol{\alpha}(\boldsymbol{\mu}) \quad (4.31)$$

na qual  $\boldsymbol{\alpha}(\boldsymbol{\mu})$  é o vetor que contém os coeficientes lineares  $\alpha^i(\boldsymbol{\mu})$  e  $\mathbf{Z} = [\zeta^1, \dots, \zeta^N]$ .

Partindo da equação fundamental do FEM (equação (4.22)) e substituindo o vetor de deslocamentos  $\mathbf{u}$  por  $\mathbf{u}^N$  definido acima, tem-se:

$$\mathbf{K}(\boldsymbol{\mu})\mathbf{Z}\boldsymbol{\alpha}(\boldsymbol{\mu}) = \mathbf{F}(\boldsymbol{\mu}) \quad (4.32)$$

Pode ser efetuada a multiplicação de ambos os membros da equação por  $\mathbf{Z}^T$ , para obter:

$$\mathbf{Z}^T\mathbf{K}(\boldsymbol{\mu})\mathbf{Z}\boldsymbol{\alpha}(\boldsymbol{\mu}) = \mathbf{Z}^T\mathbf{F}(\boldsymbol{\mu}) \quad (4.33)$$

chamando

$$\mathbf{K}^N(\boldsymbol{\mu}) = \mathbf{Z}^T\mathbf{K}(\boldsymbol{\mu})\mathbf{Z} \in \mathbb{R}^{N \times N} \quad (4.34)$$

e

$$\mathbf{F}^N = \mathbf{Z}^T\mathbf{F} \in \mathbb{R}^N \quad (4.35)$$

com isso, tem-se

$$\mathbf{K}^N(\boldsymbol{\mu})\boldsymbol{\alpha}(\boldsymbol{\mu}) = \mathbf{F}^N(\boldsymbol{\mu}) \quad (4.36)$$

o que permite a obtenção dos coeficientes lineares  $\boldsymbol{\alpha}(\boldsymbol{\mu})$ .

Tomando como base as definições anteriores, a aproximação de base reduzida para a grandeza energia de deformação a ser investigada, provem da seguinte formulação:

$$s^N(\boldsymbol{\mu}) = (\mathbf{u}^N(\boldsymbol{\mu}))^T \cdot \mathbf{F} \Rightarrow s^N(\boldsymbol{\mu}) = \boldsymbol{\alpha}(\boldsymbol{\mu})^T \cdot \mathbf{F}^N \quad (4.37)$$



#### 4.4.1.4 Precisão

A verificação da precisão para o método é realizada por um estimador de erro. Para calcular o erro de um modo eficiente computacionalmente, a seguinte equação residual deve ser resolvida:

$$\mathbf{C}(\boldsymbol{\mu})\hat{\mathbf{e}}(\boldsymbol{\mu}) = \mathbf{R}(\boldsymbol{\mu}) \quad (4.38)$$

na qual  $\mathbf{C}(\boldsymbol{\mu})$  é um operador simétrico definido através de:

$$\mathbf{e}(\boldsymbol{\mu})^T \mathbf{K}(\boldsymbol{\mu}) \mathbf{e}(\boldsymbol{\mu}) \leq \hat{\mathbf{e}}(\boldsymbol{\mu})^T \mathbf{C}(\boldsymbol{\mu}) \hat{\mathbf{e}}(\boldsymbol{\mu}) \quad (4.39)$$

Na equação acima  $\mathbf{e}(\boldsymbol{\mu})$  é o erro exato e  $\hat{\mathbf{e}}(\boldsymbol{\mu})$  é o erro aproximado e  $\mathbf{R}$  é o resíduo.

Para calcular  $\mathbf{C}$  é considerada a chamada estratégia do condicionador pontual (PRUD'HOMME et al., 2002) apresentada a seguir:

$$\mathbf{C}(\boldsymbol{\mu}) = g(\boldsymbol{\mu})\hat{\mathbf{K}} \quad (4.40)$$

no qual  $g(\boldsymbol{\mu}) = \min \{ \mu^r \}$  com  $r = 1 \dots R$  e:

$$\hat{\mathbf{K}} = \sum_{r=1}^R \mathbf{K}_r \quad (4.41)$$

É importante enfatizar que os cálculos, da solução e do erro, são feitos de forma rápida uma vez que este utiliza termos pré-calculados (independentes de  $\boldsymbol{\mu}$ ). Isto é possível devido às definições apresentadas nas equações (4.34), (4.35) e (4.36) junto à forma decomposta da matriz de rigidez definida na equação (4.24).

#### 4.4.1.5 Algoritmo computacional

O maior custo computacional no RBM está na formação via FEM da matriz  $\mathbf{K}^N$  independente de  $\boldsymbol{\mu}$ , isto é, usando a matriz  $\mathbf{K}$  da equação (4.24), tem-se:

$$\mathbf{K}^N = \mathbf{Z}^T \mathbf{K} \mathbf{Z} \quad (4.42)$$

Como as matrizes são independentes dos parâmetros elas são computadas somente uma vez e usadas subsequentemente para explorar o espaço de parâmetros  $\mathbb{D}$ . Como consequência da subdivisão, a implementação computacional dos cálculos das saídas com bases reduzidas é conduzida seguindo os estágios *off-line/on-line* descrito descritos na tabela (4.1).

Tabela (4.1): Algoritmo do RBM: estágios *off-line/on-line*

---

*OFF-LINE* – independente de  $\boldsymbol{\mu}$

1. Selecionar uma amostra:  $\mathcal{S}^N = \{(\mu_1, \dots, \mu_R)^1, \dots, (\mu_1, \dots, \mu_R)^N\}$ ;
2. Construir a matriz de soluções obtidas via FE:  $\mathbf{Z} = [\boldsymbol{\zeta}^1, \dots, \boldsymbol{\zeta}^N]$ ;
3. Construir a matriz de cada região com bases reduzidas:  $\mathbf{K}_r^N = \mathbf{Z}^T \mathbf{K}_r \mathbf{Z}$ .

*ON-LINE* – para um novo vetor  $\boldsymbol{\mu}$

1. Formar a matriz com bases reduzidas:  $\mathbf{K}^N(\boldsymbol{\mu}) = \sum_{r=1}^R \mu_r \mathbf{K}_r^N$ ;
  2. Resolver:  $\mathbf{K}^N(\boldsymbol{\mu}) \boldsymbol{\alpha}(\boldsymbol{\mu}) = \mathbf{F}^N(\boldsymbol{\mu})$ ;
  3. Avaliar:  $s^N(\boldsymbol{\mu}) = \boldsymbol{\alpha}(\boldsymbol{\mu})^T \cdot \mathbf{F}^N$ .
- 

Como podem ser observadas, durante o estágio *on-line*, três computações maiores são conduzidas. Calcular a matriz  $\mathbf{K}^N$ , é requerido  $R \cdot N^2$  operações. Então para um sistema de equações lineares  $N \times N$  são necessárias aproximadamente  $2/3 N^3$  para obter  $\boldsymbol{\alpha}$ . Finalmente,  $2N$  operações adicionais são necessárias para obter  $s^N$ . Como consequência, o número dominante de operações necessárias para conduzir o estágio *on-line* e obter as saídas com bases reduzidas é  $R \cdot N^2 + N^3$ .

#### **4.4.2 Programação paralela**

A computação paralela é caracterizada pelo uso de várias unidades de processamento ou processadores na execução de uma computação de forma mais rápida e baseia-se no fato de que o processo de resolução de um problema pode ser dividido em tarefas menores e essas realizadas simultaneamente através de algum tipo de coordenação. O desenvolvimento de microprocessadores e tecnologia de rede tem permitido uma maior disponibilidade da utilização de computação paralela. Para tirar vantagens desses avanços, um dos padrões de comunicação de dados em computação paralela, o MPI (*Message Passing Interface*) (PACHECO, 1998), se tornou bastante popular.

A paralelização tem permitido que a resolução de problemas que requerem dias ou semanas, numa avaliação feita por um único processador, seja resolvida em questão de horas, numa avaliação feita por um conjunto de processadores.

Muitas vezes para modificar algoritmos que tem sua essência em programação sequencial para programação paralela pode ser uma tarefa bastante árdua, pois esse procedimento leva em conta a quantidade de processadores que devem ser utilizados e como as informações devem ser trocadas entre eles. O algoritmo do PSO apresenta características que se adequam facilmente a utilização de programação paralela para realização de processo de otimização.

##### **4.4.2.1 Otimização via enxame de partículas em programação paralela**

A característica principal do algoritmo do PSO é a avaliação de uma diversidade de combinações de parâmetros de projeto no processo de otimização. Devido a isto, o PSO adequa-se facilmente a programação de um algoritmo com uma estrutura paralelizada onde cada unidade de processamento é responsável pela avaliação de uma única combinação de parâmetros.

O roteiro do algoritmo Parallel\_PSO segue basicamente o esquema apresentado no roteiro descrito no item 4.2.2 para o PSO em programação sequencial. A modificação está apenas na maneira como os procedimentos do algoritmo PSO são realizados (como explicado mais adiante).

Na programação do Parallel\_PSO, os parâmetros de confiança e a inércia são os mesmos considerados por toda a população como é feito na programação sequencial, e a quantidade de indivíduos na população indica o número de processadores que serão utilizados no processo de otimização.

#### **4.4.2.2 Inicialização e atualização das variáveis**

A população é inicializada com uma distribuição aleatória das partículas por todo o espaço de projeto. Cada unidade de processamento determina de maneira randômica as coordenadas do vetor posição, através da equação (4.1), e as coordenadas do vetor velocidade, através da equação (4.2).

A posição de cada partícula é atualizada usando a posição prévia e o vetor velocidade atualizado seguindo o esquema apresentado na equação (4.4). O vetor velocidade é atualizado de maneira convencional, de acordo com equação (4.5) ou de acordo com a equação (4.12) caso haja alguma violação de restrição.

A atualização da velocidade apresenta uma particularidade, pois necessita da informação de outra unidade de processamento, o  $p_t^b$  descrito anteriormente no item 4.2.4. Para determinação do  $p_t^b$  é feita uma troca de informação entre todos os processadores. Primeiramente todas as unidades enviam informação sobre suas aptidões para uma unidade específica onde é determinado o  $p_t^b$ . Posteriormente esta informação é enviada como resposta para as demais unidades.

#### **4.4.2.3 Verificação da convergência**

A verificação da convergência do procedimento é realizada de acordo com as condições de finalização apresentadas no item 4.2.9.1. Aqui se faz necessária mais uma vez a troca de informações, principalmente na verificação do grau de agregação das partículas.

Para verificação das condições de finalização, cada unidade armazena um histórico que contém as informações necessárias sobre as regiões investigadas ao longo do processo e outro que contém as informações das melhores aptidões encontradas. Os cálculos que não necessitam de informações de outras unidades são realizados e após a realização destas operações são iniciadas as trocas de informações necessárias.

Além dos parâmetros acima mencionados, o  $cgbest_k$  e o  $cgworst_k$ , são dois termos que necessitam da troca de informações entre as unidades para que sejam determinados. O numerador do primeiro membro da equação (4.13) também é outro termo que necessita da troca de informações. Todos os demais cálculos podem ser realizados sem a necessidade da troca de informações.

# 5

## Exemplos

### 5.1 Introdução

Com o intuito de demonstrar a eficácia do algoritmo PSO, e de todos os procedimentos adotados para adaptar o mesmo aos diversos tipos de problemas aqui estudados, serão apresentados alguns exemplos nos quais o algoritmo foi utilizado.

Como apresentado no capítulo anterior, para utilizar o algoritmo PSO, a determinação de alguns parâmetros é necessária, à medida que os exemplos forem apresentados, o comportamento do algoritmo diante da variação de tais parâmetros será discutido. Os testes foram realizados em problemas que envolvem a otimização de funções analíticas e de estruturas treliçadas. Para validar a utilização do PSO, será apresentado um comparativo feito entre os resultados obtidos através de códigos computacionais que fazem uso do SQP como otimizador.

### 5.2 Problemas de otimização considerando funções analíticas

Para motivar a utilização do algoritmo PSO, foram estudados problemas de otimização de funções analíticas que apresentam algumas especificidades. Primeiramente são mostradas as análises realizadas em problemas de otimização uni-objetivo e posteriormente as análises realizadas em problemas de otimização multiobjetivo. As particularidades de cada problema serão descritas à medida que os mesmos forem apresentados.

### 5.2.1 Otimização uni-objetivo

Nesta seção serão apresentados os resultados obtidos nos estudos realizados com as técnicas consideradas no PSO para problemas de otimização uni-objetivo.

Foram escolhidas duas funções, as quais apresentam características bastante específicas para validar a utilização do PSO. Para isso, é mostrado um comparativo entre os resultados obtidos com o PSO e os resultados obtidos com um algoritmo convencional que faz uso do SQP para realizar o processo de otimização.

Além dos estudos para validar o uso do PSO, também foram feitas algumas análises mais detalhadas sobre a relação entre os parâmetros do PSO e o processo de otimização. Para análise desses problemas, o algoritmo PSO foi executado trinta vezes. Em cada um, a distribuição das partículas foi sempre iniciada de forma randômica. O número de vezes que o algoritmo convencional foi executado é indicado em cada exemplo.

#### 5.2.1.1 Exemplo 1: função de Rastrigin

A função de Rastrigin (INNOCENTE, 2006) é uma função de  $n$ -variáveis, geralmente utilizada para testar os algoritmos genéticos, e é dada por:

$$f(\mathbf{x}) = An + \sum_{i=1}^n (x_i^2 - A \times \cos(2\pi x_i)) \quad (5.1)$$

A função de Rastrigin apresenta como característica principal a presença de vários ótimos locais, o que confunde a busca para algoritmos que fazem uso de gradiente.

Como citado anteriormente, um otimizador tradicional foi utilizado com o intuito de comparar os resultados obtidos e comprovar a viabilidade do PSO. O otimizador escolhido foi o SQP que faz parte do *optimization toolbox* encontrado no MATLAB®.

Para facilitar a visualização do comportamento da função, fez-se  $n=2$ . Um esboço é mostrado na figura (5.1). A programação numérica desta é encontrada no MATLAB® e a sua configuração padrão apresenta  $A=10$ .

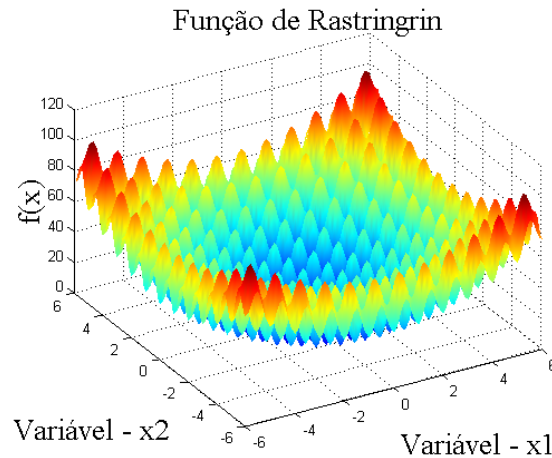


Figura (5.1) – Esboço do comportamento da função de Rastrigrin.

Resolvendo analiticamente o problema de otimização da função, o ponto de ótimo global encontrado foi  $x_1 = x_2 = 0$  que resulta em  $f(\mathbf{x}) = 0$ .

Na otimização com o PSO, foi considerada a seguinte configuração de parâmetros internos:

- os limites para as variáveis do projeto foram  $x_i^l = -5$  e  $x_i^u = 5$  com  $i = 1, 2$ ;
- o número máximo de iterações permitidas foi 3000;
- os parâmetros de vizinhança considerados foram  $c_1 = c_2 = 2$ ;
- a inércia do foi iniciada com  $w = 1.4$  podendo ser reduzido a 0.35, como descrito no capítulo 4;
- a quantidade de partículas consideradas foi  $np = 20$ .

O SPQ foi executado 49 vezes iniciando a busca de pontos diferentes no domínio do projeto. Os pontos iniciais foram determinados através de amostras uniformemente distribuídas (*uniform grid sampling*) no espaço de projeto investigado. Os resultados obtidos estão na tabela (5.1).

Tabela (5.1) – Resultados da otimização da função de Rastrigin com o PSO e com o SQP.

OTIMIZADOR	$x_1^*$	$x_2^*$	$f(\mathbf{x}^*)$	Avaliações de funções	$\mu(f(\mathbf{x}^*))$	$\sigma(f(\mathbf{x}^*))$
PSO	$2.952 \times 10^{-11}$	$5.173 \times 10^{-10}$	0	7000	0	0
SQP	$6.091 \times 10^{-8}$	$4.698 \times 10^{-8}$	$1.066 \times 10^{-14}$	45		



Na tabela (5.1), o resultado apresentado para o PSO mostra o melhor resultado obtido dentre todos os encontrados nas execuções realizadas, também são indicados a média –  $\mu(f(\mathbf{x}^*))$  – e o desvio padrão –  $\sigma(f(\mathbf{x}^*))$  – entre os mesmos. O resultado apresentado utilizando o SQP mostra apenas a melhor resposta encontrada pelo algoritmo nas várias tentativas de execução.

O número de avaliações de funções realizadas pelo PSO foi bem superior ao número de avaliações realizadas pelo SQP, porém deve ser levado em consideração que o resultado apresentado pelo SQP depende do ponto de partida adotado.

A figura (5.2) apresenta gráficos que mostram os pontos iniciais e o resultado encontrado na busca. Na figura (5.2a) o gráfico representa uma configuração para a execução do algoritmo PSO na busca da solução ótima da função, pois como explicado no capítulo 4, a distribuição inicial aleatória das posições por todo o domínio do projeto não interfere no resultado final do processo. É importante perceber que o algoritmo não encontrou dificuldades na obtenção do ótimo global, ou próximo ao global. Na figura (5.2b) cada ponto no gráfico representa uma busca realizada, e como pode ser visto o resultado obtido pelo SQP, tal como esperado, depende do ponto de partida fornecido ao algoritmo. Como pode ser observada, em todos os casos, a solução corresponde a um ponto de ótimo local mais próximo do ponto de partida.

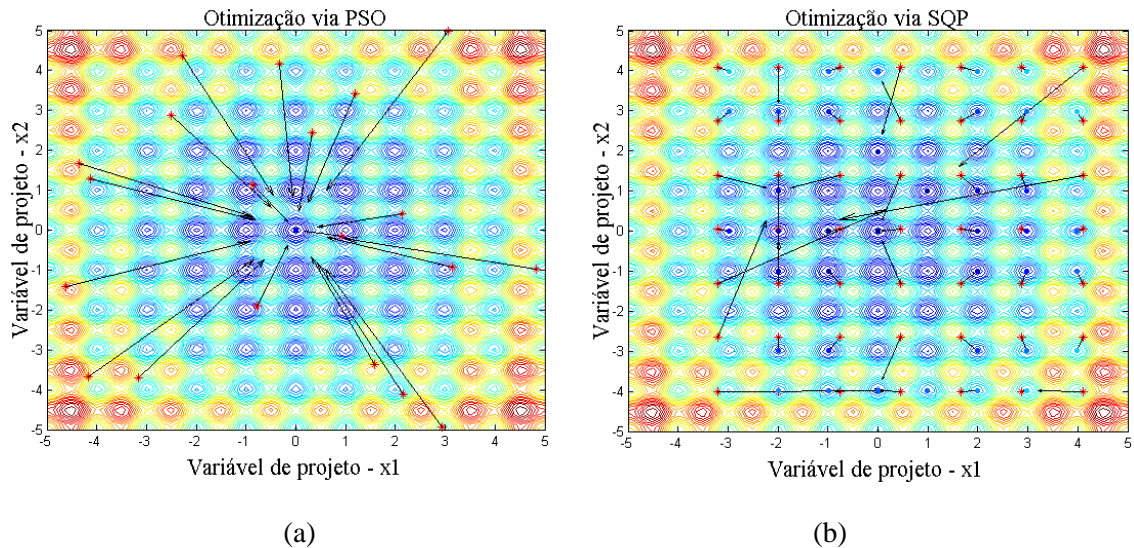


Figura (5.2) – Gráficos mostrando o ponto inicial e o ponto ótimo resultado da otimização (a) via PSO e (b) via SQP.

Como verificado com os resultados acima, o PSO apresentou-se bastante eficaz na resolução de problemas que apresentam vários ótimos locais. Para melhor estudar o desempenho do algoritmo para este tipo de problema, uma série de variações nos parâmetros internos foi considerada. Tais considerações serão discutidas a seguir.

- Influência da quantidade de variáveis

Como citado na definição problema, a função de Rastrigin é uma função de  $n$ -variáveis. Anteriormente foi tomado  $n = 2$ , apenas para facilitar a visualização.

A quantidade de variáveis em um problema pode dificultar o encontro da solução adequada. Para testar a eficácia do PSO diante de problemas com várias variáveis de projeto o algoritmo foi executado tomando-se valores diferentes para  $n$ .

Os resultados obtidos encontram-se listados na tabela (5.2).

Tabela (5.2) – Resultados da otimização da função de Rastrigin modificando o número de dimensões do problema.

$n$	$f(\mathbf{x}^*)$	$\mu(f(\mathbf{x}^*))$	$\sigma(f(\mathbf{x}^*))$
5	0	0	0
10	0	0	0
15	0	0	0
20	0	0	0
25	0	0	0
30	0	$2.173 \times 10^{-3}$	$1.117 \times 10^{-2}$

A tabela (5.2) apresenta apenas o melhor resultado encontrado, a média e o desvio padrão para cada caso considerado.

De acordo com a mesma, apenas quando o número de variáveis considerado foi igual a trinta, o PSO apresentou resultados que não representam exatamente o ótimo global, mesmo assim o PSO apresenta uma boa eficácia para problema com muitas variáveis.

- Consideração de valor fixo da inércia

No capítulo 4 foi citada a influência do valor da inércia ( $w$ ) no processo de busca, onde valores baixos tornariam a exploração mais local e valores altos a tornariam mais global. Como padrão, a inércia foi considerada variando de acordo com a evolução da busca, idéia apresentada por Venter e Sobieszczanski-Sobieski (VENTER e SOBIESZCZANSKI-SOBIESKI, 2002).

Para verificar a influência da inércia, valores fixos para mesma foram considerados na execução do algoritmo. Os valores considerados foram selecionados no intervalo estabelecido quando considerada a atualização dinâmica ao longo do processo de busca.

A tabela (5.3) apresenta os resultados encontrados para os diversos valores de  $w$  considerados.

Tabela (5.3) – Resultados da otimização da função de Rastrigin considerando diferentes valores fixos de inércia.

$w$	$x_1^*$	$x_2^*$	$f(x^*)$	$\mu(f(x^*))$	$\sigma(f(x^*))$
0.35	$-9.540 \times 10^{-10}$	$3.117 \times 10^{-9}$	0	0.3648	0.6118
0.50	$-2.896 \times 10^{-9}$	$-1.428 \times 10^{-9}$	0	0.1990	0.4048
0.65	$5.665 \times 10^{-10}$	$-2.696 \times 10^{-9}$	0	0.3980	0.4958
0.80	$-7.642 \times 10^{-10}$	$1.022 \times 10^{-10}$	0	0.1658	0.3771
1.00	$6.078 \times 10^{-10}$	$-7.956 \times 10^{-10}$	0	$3.317 \times 10^{-2}$	0.1817
1.20	$2.131 \times 10^{-9}$	$1.533 \times 10^{-9}$	0	0	0
1.40	$1.393 \times 10^{-9}$	$-6.350 \times 10^{-10}$	0	0	0

Com os resultados apresentados na tabela (5.3) pode-se perceber que quando se mantém a exploração local, ou seja, quando o valor admitido para  $w$  é pequeno, nem sempre o ótimo global é encontrado. Isto se deve ao fato da busca ser iniciada com uma exploração local, o que é recomendado apenas para o final do processo uma vez que essa imposição favorece uma exploração mais detalhada de uma determinada região do domínio de projeto. Como a função apresenta várias regiões de ótimo que são bastante acentuadas, apesar dessas regiões serem bastante próximas, a saída desses vales pode ser impedida devido a essa exploração refinada feita desnecessariamente logo no início da busca.

- Consideração de diferentes parâmetros de confiança

Para testar a influência dos parâmetros de confiança,  $c_1$  e  $c_2$ , no processo de otimização, algumas combinações com os mesmos foram adotadas respeitando o fator de aceleração descrito no capítulo 4. Nos testes realizados os valores de  $c_1$  foram pré-determinados e, conseqüentemente,  $c_2 = 4 - c_1$ .

As variações foram feitas em duas etapas, a primeira considerava uma atualização dinâmica para as confianças em cada iteração e a segunda considerava um valor fixo para as mesmas durante toda a busca.

Na primeira etapa dos testes, o valor de  $c_1$  variava linearmente com o decorrer das iterações como sugerido por (INNOCENTE, 2006). Testes com o valor de  $c_1$  diminuindo bem como com o valor de  $c_1$  aumentando foram conduzidos, gerando dessa forma dois casos para a etapa 1, tal como apresentado na tabela (5.4).

Na segunda etapa dos testes, o valor de  $c_1$  eram pré-estabelecidos e mantidos durante todo o processo de busca. Os valores determinados geraram quatro casos para a etapa 2, tal como apresentado na tabela (5.5).

Tabela (5.4) - Resultados da otimização da função de Rastrigin considerando variações dinâmicas nos parâmetros de confiança.

$c_1$	$x_1^*$	$x_2^*$	$f(\mathbf{x}^*)$	$\mu(f(\mathbf{x}^*))$	$\sigma(f(\mathbf{x}^*))$
Aumentando (partindo de 0.5)	$1.891 \times 10^{-11}$	$1.875 \times 10^{-11}$	0	0	0
Diminuindo (partindo de 3.5)	$-1.825 \times 10^{-12}$	$-1.998 \times 10^{-10}$	0	0	0

Tabela (5.5) - Resultados da otimização da função de Rastrigin considerando diferentes valores fixos para os parâmetros de confiança.

$c_1$	$x_1^*$	$x_2^*$	$f(\mathbf{x}^*)$	$\mu(f(\mathbf{x}^*))$	$\sigma(f(\mathbf{x}^*))$
1.0	$-6.347 \times 10^{-10}$	$-8.006 \times 10^{-11}$	0	0	0
1.5	$-9.420 \times 10^{-10}$	$-5.511 \times 10^{-13}$	0	0	0
2.5	$-8.040 \times 10^{-10}$	$-8.099 \times 10^{-11}$	0	0	0
3.0	$7.717 \times 10^{-10}$	$-7.096 \times 10^{-10}$	0	0	0

Como apresentam as tabelas acima, os resultados das otimizações não foram influenciados pela variação nos parâmetros de confiança neste problema. Este fato é um pouco estranho, pois era esperado que o comportamento do algoritmo na otimização fosse semelhante ao apresentado na avaliação feita para  $w$ , uma vez que  $c_1$  também é responsável pela exploração da partícula no espaço do projeto.

Quando foram feitas as modificações nos parâmetros de confiança, o valor da inércia voltou a ser atualizado como considerado no modelo padrão adotado descrito no início do problema. Isso indica que a inércia foi o parâmetro que mais influenciou o comportamento do algoritmo na análise.

### 5.2.1.2 Exemplo 2: função não homogênea

Neste exemplo uma função não homogênea (uma função não diferenciável em todo o domínio) e que apresenta pontos ótimos locais é considerada. Este tipo de problema praticamente torna inviável o uso de algoritmos que necessitam do cálculo de derivadas para obtenção do ponto de ótimo.

A função escolhida é uma função de duas variáveis apenas e é encontrada entre as funções utilizadas para testar a eficácia dos algoritmos evolucionários no MATLAB®. A função é apresentada a seguir.

$$f(x) = \begin{cases} x_1^2 + x_2^2 - 25 & \text{se } x_1 < -7 \\ -2\sin(x_1) - \frac{x_1 x_2^2}{10} + 15 & \text{se } x_1 < -3 \\ 0.5x_1^2 + 20 + |x_2| & \text{se } x_1 < 0 \\ 0.3\sqrt{x_1} + 25 + |x_2| & \text{se } x_1 \geq 0 \end{cases} \quad (5.2)$$

Para verificar a eficácia do PSO considerando este tipo de problema, foram realizados procedimentos semelhantes aos considerados no exemplo 1. Os resultados obtidos com o PSO foram mais uma vez comparados com os resultados obtidos utilizando o SQP. Um esboço do comportamento da função é mostrado na figura (5.3).

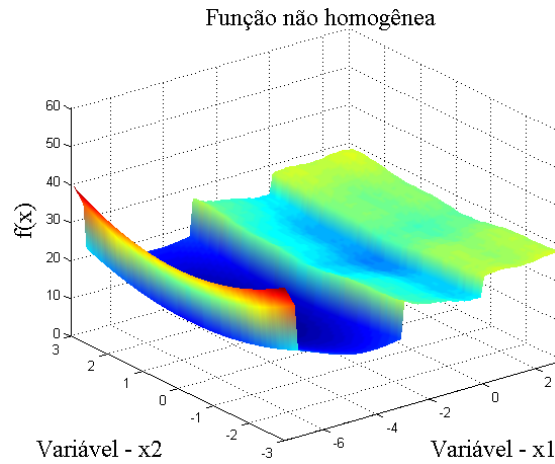


Figura (5.3) – Esboço do comportamento da função não homogênea considerada no exemplo 2.

Na mesma figura (5.4) duas vistas do comportamento da função são mostradas para uma melhor visualização dos trechos onde são apresentadas as singularidades.

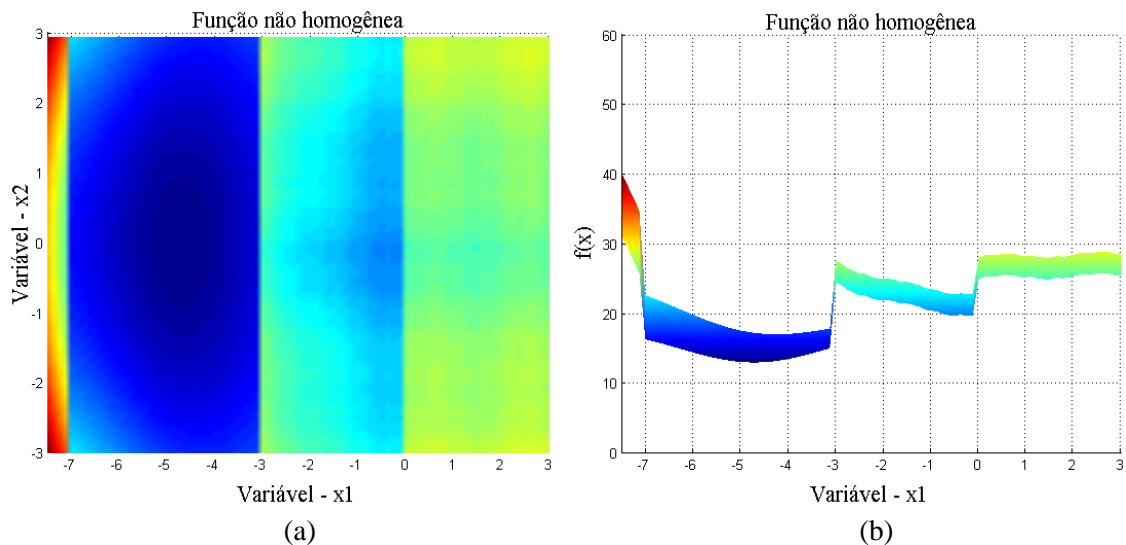


Figura (5.4) – (a) Vista X-Y e (b) Vista X-Z do comportamento da função não homogênea considerada no exemplo 2.

Resolvendo analiticamente o problema de otimização da função ponto de ótimo global encontrado foi  $x_1 = -3\pi/2$  e  $x_2 = 0$  que resulta em  $f(\mathbf{x}) = 13$ .

Na otimização com o PSO, foi considerada seguinte configuração de parâmetros internos:

- os limites para as variáveis do projeto foram  $-7.5 \leq x_1 \leq 3$  e  $-3 \leq x_2 \leq 3$ ;

- o número máximo de iterações permitidas foi 3000;
- os parâmetros de vizinhança considerados foram  $c_1 = c_2 = 2$ ;
- a inércia do foi iniciada com  $w = 1.4$  podendo ser reduzido a 0.35, como descrito no capítulo 4;
- a quantidade de partículas consideradas foi  $np = 20$ .

O SQP foi executado 36 vezes iniciando a busca de pontos diferentes no domínio do projeto. Os pontos iniciais foram determinados através de amostras uniformemente distribuídas (*uniform grid sampling*) no espaço de projeto investigado. Os resultados obtidos encontram-se na tabela (5.6).

Tabela (5.6) – Resultados da otimização da função do exemplo 2 com o PSO e com o SQP.

OTIMIZADOR	$x_1^*$	$x_2^*$	$f(\mathbf{x}^*)$	Avaliações de funções	$\mu(f(\mathbf{x}^*))$	$\sigma(f(\mathbf{x}^*))$
PSO	-4.712389	$1.596 \times 10^{-9}$	13.0	7000	13.0	0
SQP	-4.712389	$-5.283 \times 10^{-8}$	13.0	30		

Na tabela (5.6), o resultado apresentado para o PSO mostra a melhor solução obtida dentre todos os encontrados nas execuções realizadas, a média e o desvio padrão dos mesmos. O resultado apresentado para o SQP mostra a melhor resposta encontrada pelo algoritmo nas várias tentativas de execução.

Vale ressaltar mais uma que o número de avaliações de funções realizadas pelo PSO foi bem superior ao número de avaliações realizadas pelo SQP, porém deve ser levado em consideração que o resultado apresentado pelo SQP depende do ponto de partida adotado.

Na figura (5.5) são mostrados os pontos de partida para a busca e o resultado final encontrado. Na figura (5.5a) o gráfico representa uma configuração para a execução do algoritmo PSO na busca da solução ótima da função, confirmando mais uma vez que a distribuição inicial randômica das posições por todo o domínio do projeto não interfere no resultado final do processo.

Na figura (5.5b) cada ponto no gráfico representa uma busca realizada, e mais uma vez pode ser visto que o resultado obtido pelo SQP depende do ponto de partida fornecido ao algoritmo. Como pode ser observado, na maioria dos casos, a solução corresponde a um ponto de ótimo local mais próximo do ponto de partida

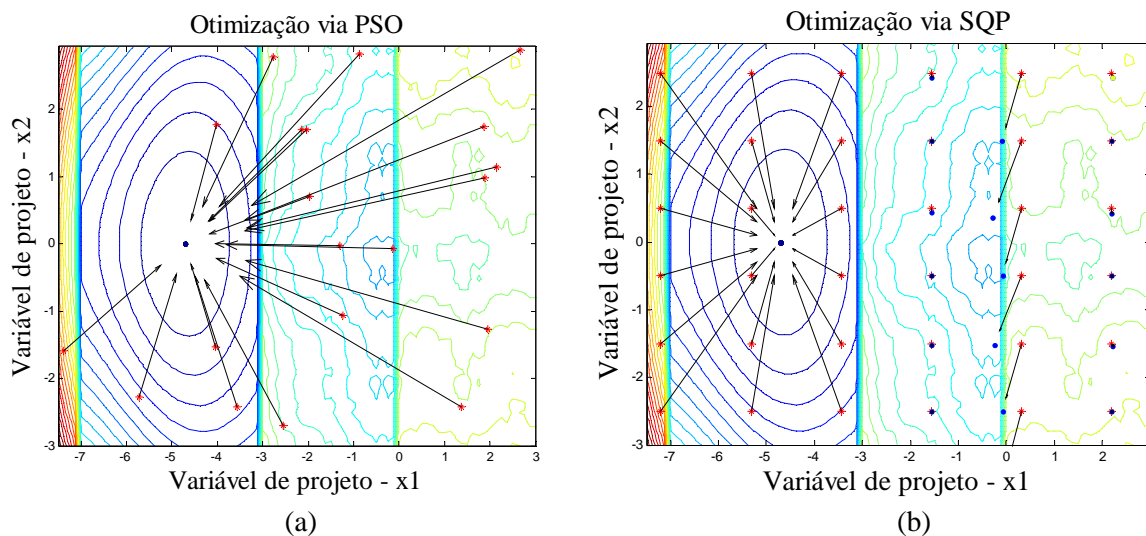


Figura (5.5) – Gráficos mostrando o ponto inicial e o ponto ótimo resultado da otimização (a) via PSO e (b) via SQP.

De acordo com os resultados acima, o PSO mais uma vez apresentou-se bastante eficaz na resolução do problema investigado. Para melhor estudar o desempenho do algoritmo para este tipo de problema, algumas considerações foram feitas, semelhante ao exemplo anterior. Tais considerações serão discutidas a seguir.

- Consideração de valor fixo da inércia

Para verificar a influência da inércia para um problema considerando este tipo de função, valores fixos para a mesma foram considerados na execução do algoritmo. Os valores fixos considerados variam entre os limites estabelecidos para inércia quando a mesma variava durante a busca.

A tabela (5.7) apresenta os resultados encontrados para os diversos valores de  $w$  considerados.



Tabela (5.7) – Resultados da otimização da função do exemplo 2 considerando diferentes valores fixos de inércia.

$w$	$x_1^*$	$x_2^*$	$f(\mathbf{x}^*)$	$\mu(f(\mathbf{x}^*))$	$\sigma(f(\mathbf{x}^*))$
0.35	-4.712389	$3.148 \times 10^{-9}$	13.0	13.0	0
0.50	-4.712389	$3.890 \times 10^{-10}$	13.0	13.0	0
0.65	-4.712389	$1.412 \times 10^{-8}$	13.0	13.0	0
0.80	-4.712389	$-2.200 \times 10^{-8}$	13.0	13.0	0
1.00	-4.712389	$-1.094 \times 10^{-8}$	13.0	13.0	0
1.20	-4.712389	$4.693 \times 10^{-9}$	13.0	13.0	0
1.40	-4.712389	$1.461 \times 10^{-9}$	13.0	13.0	0

Com os resultados apresentados na tabela (5.7) pode-se perceber que para a função selecionada neste exemplo o algoritmo manteve-se estável para qualquer consideração feita na inércia. Diferente da função considerada no exemplo anterior, as várias regiões de ótimo desta função são suaves e próximas, o que impede o aprisionamento das partículas nesses vales.

- Consideração de diferentes parâmetros de confiança

Neste exemplo também foi testada a influência dos parâmetros de confiança,  $c_1$  e  $c_2$ , no processo de otimização. Como no exemplo anterior, nos testes realizados foram pré-determinados os valores de  $c_1$  e calculados  $c_2 = 4 - c_1$ .

As variações também foram realizadas em duas etapas, a primeira considerava uma atualização dinâmica para as confianças em cada iteração e a segunda considerava um valor fixo para as mesmas durante toda a busca. A tabela (5.8) mostra os resultados da primeira etapa e a tabela (5.9) mostra os resultados da segunda etapa.

Tabela (5.8) - Resultados da otimização considerando variações dinâmicas nos parâmetros de confiança.

$c_1$	$x_1^*$	$x_2^*$	$f(\mathbf{x}^*)$	$\mu(f(\mathbf{x}^*))$	$\sigma(f(\mathbf{x}^*))$
Aumentando (partindo de 0.5)	-4.712389	$9.778 \times 10^{-10}$	13.0	13.0	0
Diminuindo (partindo de 3.5)	-4.712389	$-4.584 \times 10^{-8}$	13.0	13.0	0

Tabela (5.9) - Resultados da otimização considerando diferentes valores fixos para os parâmetros de confiança.

$c_1$	$x_1^*$	$x_2^*$	$f(\mathbf{x}^*)$	$\mu(f(\mathbf{x}^*))$	$\sigma(f(\mathbf{x}^*))$
1.0	-4.712389	$-2.464 \times 10^{-8}$	13.0	13.0	0
1.5	-4.712389	$-1.228 \times 10^{-8}$	13.0	13.0	0
2.5	-4.712389	$3.205 \times 10^{-8}$	13.0	13.0	0
3.0	-4.712389	$4.140 \times 10^{-8}$	13.0	13.0	0

Como apresentam as tabelas acima, os resultados indicam um comportamento estável para função considerada neste exemplo de maneira semelhante ao apresentado na avaliação feita para a inércia.

## 5.2.2 Otimização multiobjetivo

Aqui serão apresentados os resultados obtidos nos estudos realizados com as técnicas consideradas no PSO para otimização de problemas multiobjetivo. Foram selecionados seis problemas com o intuito de mostrar a viabilidade da utilização do PSO. Para isso, foi feito um comparativo entre os resultados encontrados com o PSO e os resultados obtidos com uma das técnicas mais robustas para a otimização multiobjetivo que utilizam a informação do gradiente e que fazem uso do SQP.

Os cinco primeiros problemas (MO1 a MO5) foram retirados de Vrahantis e Parsopoulos (PARSOPOULOS e VRAHATIS, 2002) e o último (MO6) foi retirado de Mostaghim e Teich (MOSTAGHIM e TEICH, 2004).

As técnicas utilizadas na otimização multiobjetivo via enxame de partículas foram as apresentadas no capítulo 4, a saber, EDWA, MOPSO e VEPSO. Para efeito de comparação, o método escolhido que faz uso de informação do gradiente das funções foi o NNC.

Os problemas selecionados apresentam apenas dois objetivos para facilitar a visualização dos resultados obtidos. Para análise desses problemas, o algoritmo PSO foi executado vinte vezes em cada e o algoritmo que faz uso do SQP foi executado 23 vezes, que foi o número de vezes necessárias para obtenção de pontos suficientes para a formação da curva de Pareto.

### 5.2.2.1 Exemplo 1: problema MO1

O problema MO1 é dado por:

Minimize  $\mathbf{F}(f_1(\mathbf{x}), f_2(\mathbf{x}))$ , onde:

$$f_1(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^n x_i^2 \text{ e } f_2(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^n (x_i - 2)^2 \quad (5.3)$$

Para este problema a frente de Pareto a ser encontrada é convexa. O número de variáveis consideradas é  $n = 20$ , para todos os otimizadores utilizados neste problema.

A figura (5.6) mostra a forma da frente de Pareto encontrada por cada um deles. Foi selecionado apenas um dos resultados obtidos nas vinte execuções de cada metodologia do PSO e sobreposto ao resultado obtido com o NNC.

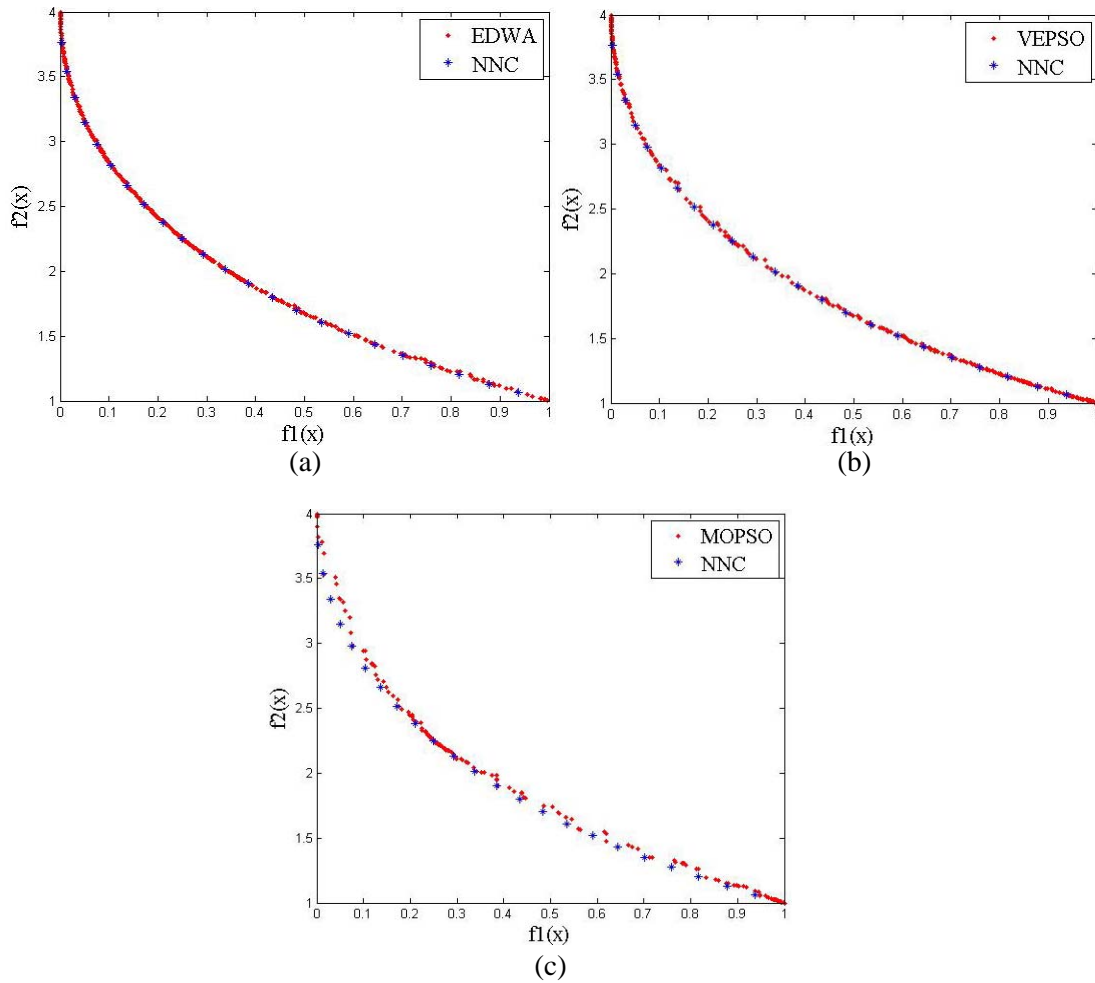


Figura (5.6) – Frentes de Pareto encontradas no problema MO1 via (a) EDWA, (b) VEPSO, (c) MOPSO comparadas à encontrada via NNC.

Nas técnicas adotadas com o PSO nas aplicações de otimização multiobjetivo, os parâmetros internos determinados antes da otimização apresentam a seguinte configuração:

- os limites para as variáveis do projeto foram  $0 \leq x_i \leq 1$ ;
- o número máximo de iterações permitidas foi 1000;
- os parâmetros de vizinhança considerados foram  $c_1 = c_2 = 2$ ;
- a inércia do foi iniciada com  $w = 1.4$  podendo ser reduzido a 0.35, como descrito no capítulo 4;
- a quantidade de partículas consideradas foi  $np = 20$ .

Os parâmetros específicos de cada técnica considerados foram os seguintes: no EDWA, o valor da frequência considerada foi  $F = 100$ ; e no VEPSO o valor do fator constrição considerado foi  $\chi = 1.5$ . No MOPSO não há necessidade de parâmetros adicionais.

Na tabela (5.10) são apresentadas as quantidades de soluções de Pareto encontradas com as técnicas utilizadas.

Tabela (5.10) – Número médio de soluções de Pareto encontradas por cada metodologia considerada – exemplo MO1.

Metodologia	Nº médio de soluções de Pareto
EDWA	368
VEPSO	178
MOPSO	165
NNC	23

O valor indicado para as técnicas do PSO representa a média do número de soluções em todas as execuções, e como citado anteriormente, o valor indicado para o NNC também representa o número de vezes que o mesmo foi executado.

De acordo com os resultados encontrados, as técnicas evolucionárias foram capazes de identificar a frente de Pareto. Um destaque, neste problema, deve ser dado ao EDWA, pois a melhor curva (que apresenta o maior número de pontos pertencentes a frente de Pareto) foi obtida com o mesmo.

Como visto, o MOPSO foi capaz de determinar a superfície de *tradeoff*, porém a mesma não foi tão bem determinada quanta as superfícies encontradas pelos outros métodos. Foram realizados alguns testes com o intuito de identificar a o porquê da dificuldade do MOPSO em definir essa superfície, mas o motivo não foi encontrado.

Um resultado mais satisfatório, ou seja, com uma boa distribuição dos pontos na superfície de Pareto, só foi obtido quando o número de variáveis consideradas em cada função foi diminuído para 10.

Por questão de comparação entre o desempenho, os novos testes foram executados com todas as versões do algoritmo PSO para problemas multiobjetivo. A figura (5.7) mostra os resultados obtidos com cada uma das metodologias do PSO sobrepostas aos resultados obtidos com o NNC, todos considerando dez variáveis.

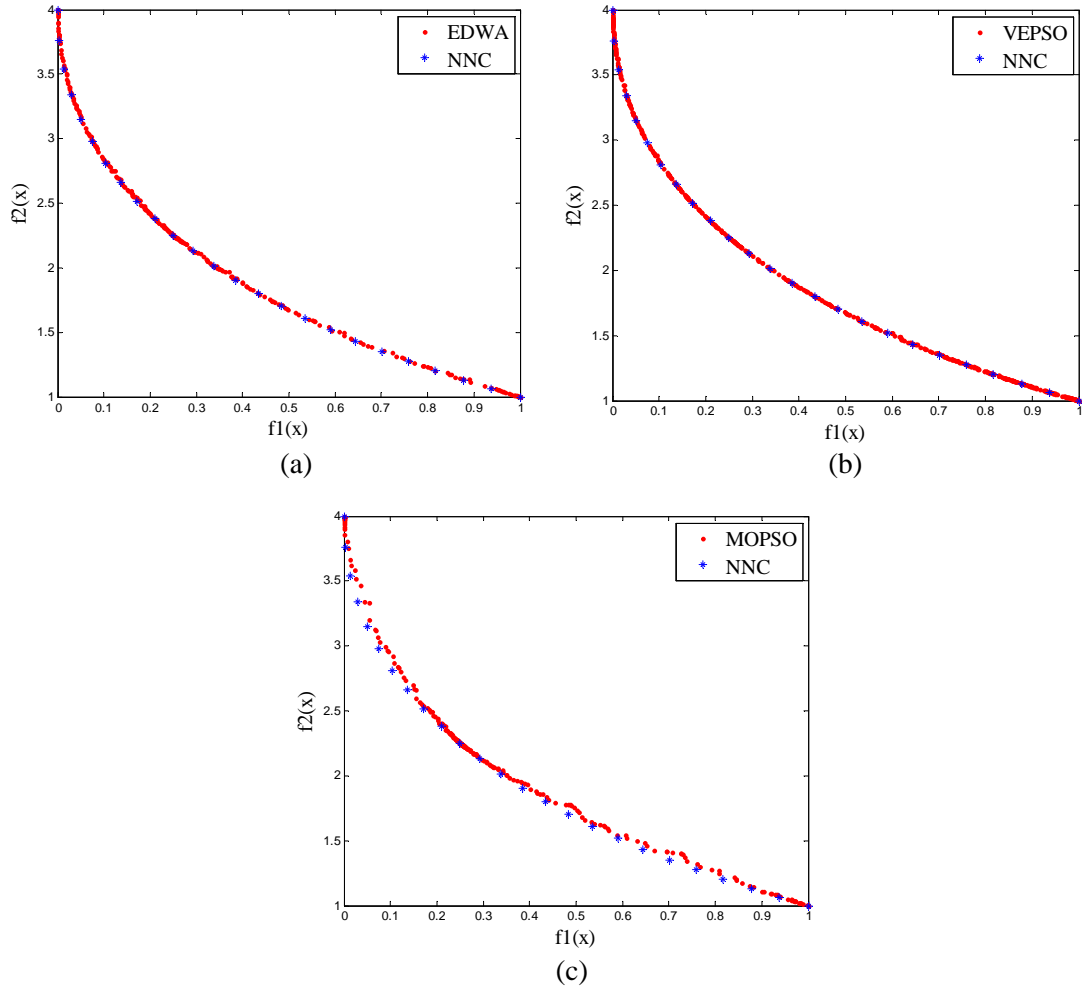


Figura (5.7) - Frente de Pareto encontrada no problema MO1, em novos testes, via (a) EDWA, (b) VEPSO, (c) MOPSO comparadas à encontrada via NNC, considerando dez variáveis em cada função.

### 5.2.2.2 Exemplo 2: problema MO2

O problema MO2 é dado por:

Minimize  $\mathbf{F}(f_1(\mathbf{x}), f_2(\mathbf{x}))$ , onde:

$$f_1(\mathbf{x}) = x_1 \text{ e } f_2(f_1, g) = g \left( 1 - \sqrt{f_1/g} \right) \text{ onde } g = g(\mathbf{x}) = 1 + \frac{9}{n-1} \sum_{i=2}^n x_i \quad (5.4)$$

Para este problema a frente de Pareto a ser encontrada é convexa e o que diferencia este exemplo do exemplo anterior é a distribuição dos pontos na superfície de Pareto encontrada pelas metodologias evolucionárias. No exemplo anterior a distribuição era mais uniforme, ou seja, não havia concentração de pontos em qualquer região da curva, já neste exemplo a distribuição é não-uniforme. O número de variáveis consideradas neste problema também é  $n = 20$ , para todos os otimizadores utilizados.

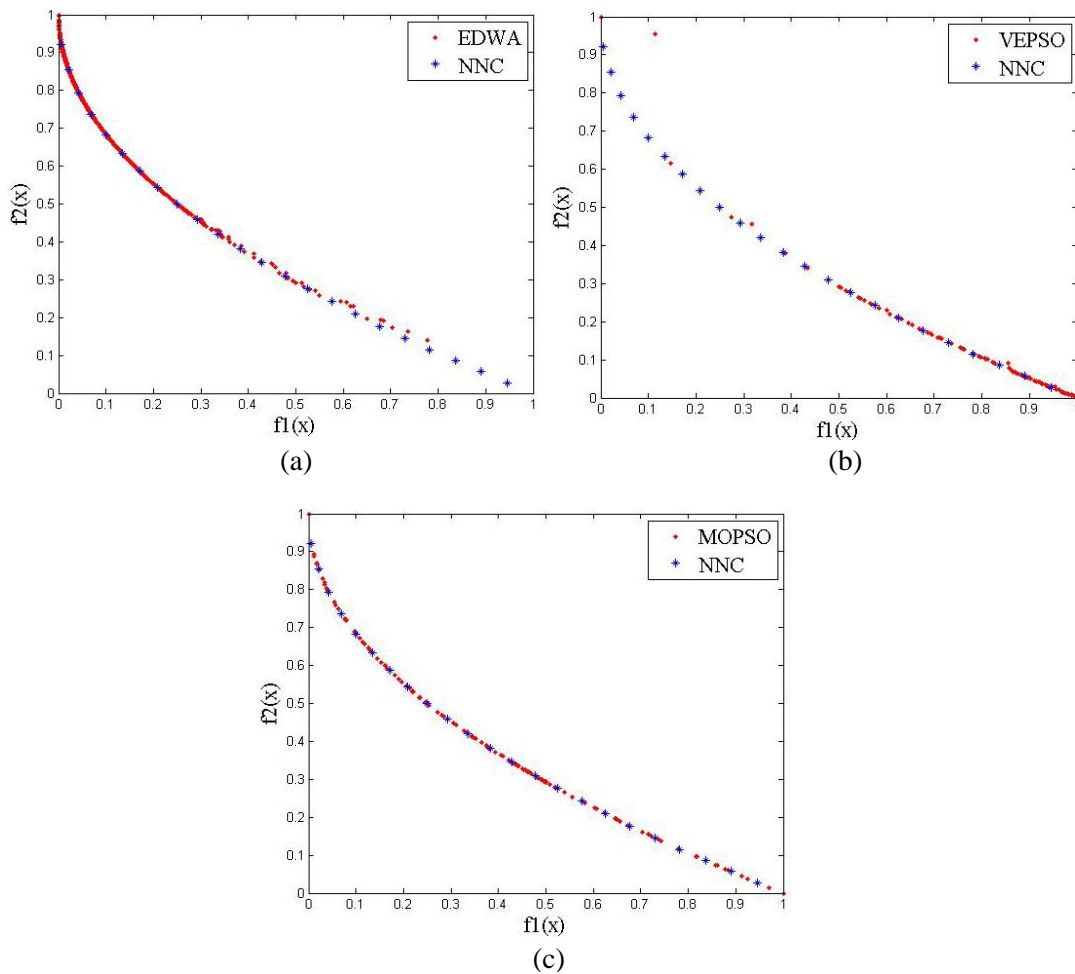


Figura (5.8) – Frentes de Pareto encontradas no problema MO2 via (a) EDWA, (b) VEPSO, (c) MOPSO comparadas à encontrada via NNC.

Na figura (5.8) é mostrada a forma da frente de Pareto encontrada por cada um deles. Para os resultados obtidos via PSO, foi selecionado apenas um dos resultados obtidos nas vinte execuções de cada metodologia do PSO e sobreposto ao resultado obtido com o NNC.

Nas técnicas adotadas com o PSO multiobjetivo a configuração considerada para os parâmetros internos foram os mesmos adotados no exemplo anterior. Os parâmetros específicos de cada técnica também são semelhantes aos considerados no exemplo 1. Na tabela (5.11) são apresentadas as quantidades de soluções de Pareto encontradas.

Tabela (5.11) – Número médio de soluções de Pareto encontradas por cada metodologia considerada – exemplo MO2.

Metodologia	Nº médio de soluções de Pareto
EDWA	715
VEPSO	47
MOPSO	137
NNC	23

Como no problema anterior, o valor indicado para as técnicas do PSO representa a média do número de soluções encontradas em todas as execuções, e o valor indicado para o NNC também representa o número de vezes que o mesmo foi executado.

De acordo com os resultados encontrados, as técnicas evolucionárias não atenderam as expectativas tão bem quanto se esperava, pois uma parte da curva não ficou bem determinada. O motivo deste problema não foi identificado, porém foram feitos testes com o intuito de detectar o motivo que causou o inconveniente.

Foram executadas novas análises modificando os parâmetros internos, porém uma boa distribuição só foi obtida com a diminuição de variáveis consideradas no problema. Os novos resultados foram encontrados considerando  $n=10$ . Para comparação entre o desempenho, os novos testes foram executados com todas as versões do algoritmo PSO para problemas multiobjetivo. A figura (5.9) mostra os resultados obtidos com cada uma das metodologias do PSO sobrepostas aos resultados obtidos com o NNC, todos considerando dez variáveis.



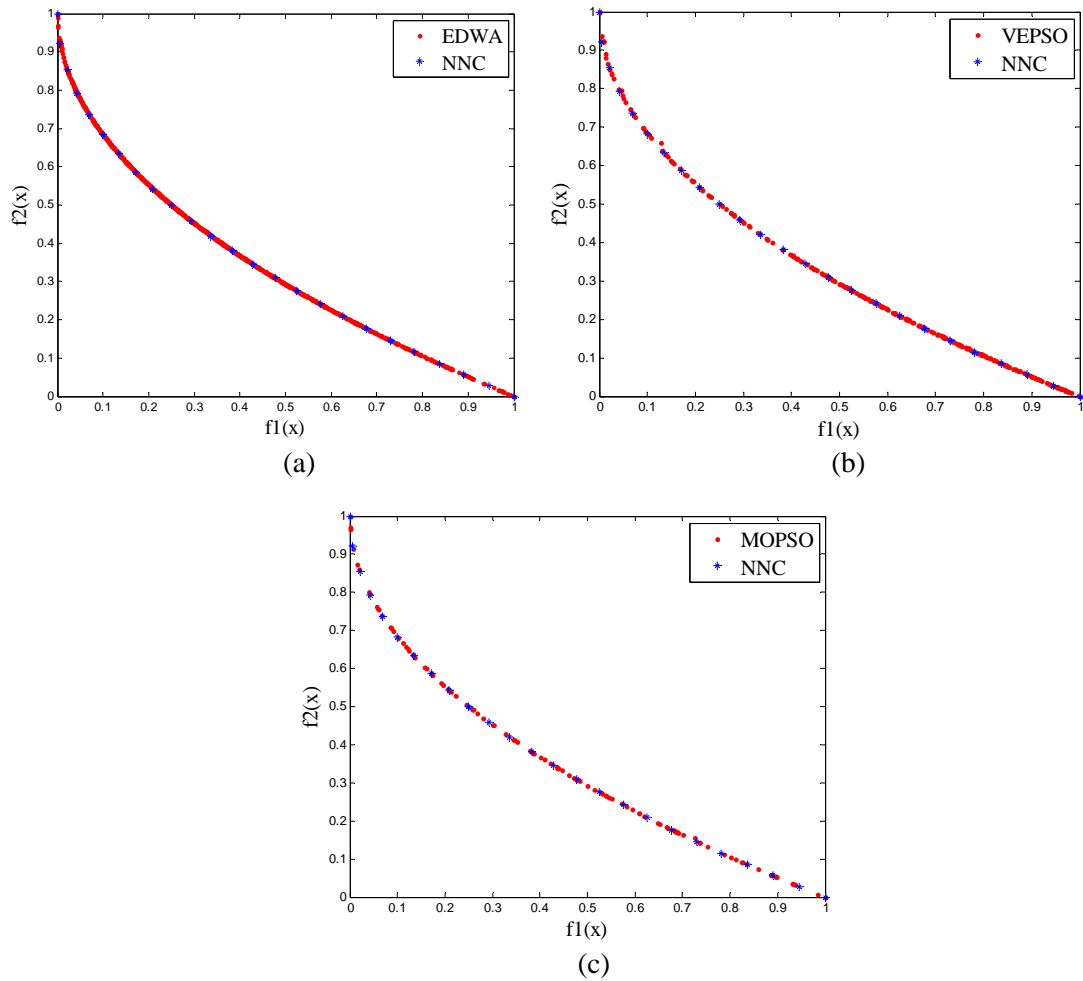


Figura (5.9) - Frente de Pareto encontrada no problema MO2, em novos testes, via (a) EDWA, (b) VEPSO, (c) MOPSO comparadas à encontrada via NNC, considerando dez variáveis em cada função.

Neste problema, o MOPSO demonstrou-se mais eficaz, quando considerando o número de variáveis igual a vinte, pois apesar de apresentar uma curva com menos pontos do que a curva encontrada pelo EDWA, os pontos apresentam-se mais espalhados, o que facilita a identificação da frente de Pareto. O resultado obtido com os novos testes realizados, o VEPSO apresentou uma curva bem distribuída, porém, a qualidade superior dos resultados obtidos pelo EDWA é destacada.

### 5.2.2.3 Exemplo 3: problema MO3

O problema MO3 é dado por:

Minimize  $\mathbf{F}(f_1(\mathbf{x}), f_2(\mathbf{x}))$ , onde:

$$f_1(\mathbf{x}) = x_1 \text{ e } f_2(f_1, g) = g(1 - (f_1/g)^2) \text{ onde } g = g(\mathbf{x}) = 1 + \frac{9}{n-1} \sum_{i=2}^n x_i \quad (5.5)$$

Para este problema a frente de Pareto a ser encontrada é não-convexa e o número de variáveis consideradas também é  $n=20$ , para todos os otimizadores utilizados.

Na figura (5.10) é mostrada a forma da frente de Pareto encontrada por cada um deles. Entre os resultados obtidos com o PSO, apenas um dentre os vinte obtidos nas execuções de cada metodologia do PSO foi selecionado e sobreposto aos resultados obtidos com o NNC.

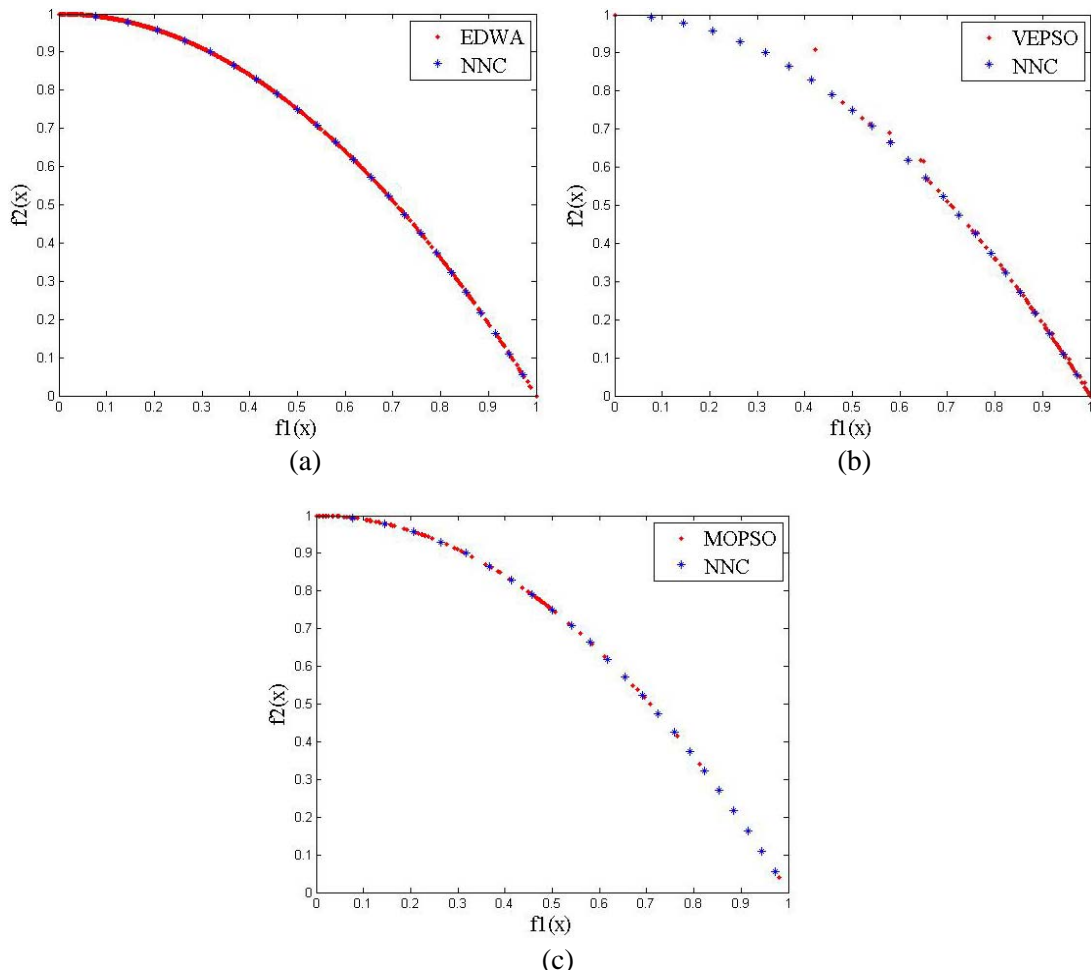


Figura (5.10) – Frentes de Pareto encontradas no problema MO3 via (a) EDWA, (b) VEPSO, (c) MOPSO comparadas à encontrada via NNC.

Nas técnicas adotadas com o PSO multiobjetivo a configuração considerada para os parâmetros internos foram os mesmos adotados no exemplo 1. Os parâmetros específicos de cada técnica também são semelhantes aos considerados anteriormente.

Na tabela (5.12) são apresentadas as quantidades de soluções de Pareto encontradas com as técnicas utilizadas no PSO e na NNC.

Tabela (5.12) – Número médio de soluções de Pareto encontradas por cada metodologia considerada – exemplo MO3.

Metodologia	Nº médio de soluções de Pareto
EDWA	284
VEPSO	11
MOPSO	32
NNC	23

Novamente, o valor indicado para as técnicas do PSO representam uma média do número de soluções em todas as execuções. A quantidade de pontos de Pareto encontradas para esse exemplo foram inferiores aos mostrados nos exemplos anteriores porque em alguns casos os algoritmos não apresentaram bons resultados e até mesmo encontrando apenas uma solução.

Como pode ser visto na figura (5.10), as curvas encontradas com o VEPSO e com o MOPSO não foram tão satisfatórias. O que ocasionou tais dificuldades não foi identificado, porém novos testes foram realizados para tentar encontrar o que causou tal inconveniente.

Para os novos testes foram consideradas novas configurações dos parâmetros internos do PSO, porém resultados satisfatórios só foram obtidos quando o número de variáveis do problema foi diminuído (quando foi considerado  $n = 10$ ). Para comparação entre o desempenho, os novos testes foram executados com todas as versões do algoritmo PSO para problemas multiobjetivo. A figura (5.11) apresenta os resultados obtidos, com cada uma das metodologias do PSO sobrepostas aos resultados obtidos com o NNC, todos considerando dez variáveis.

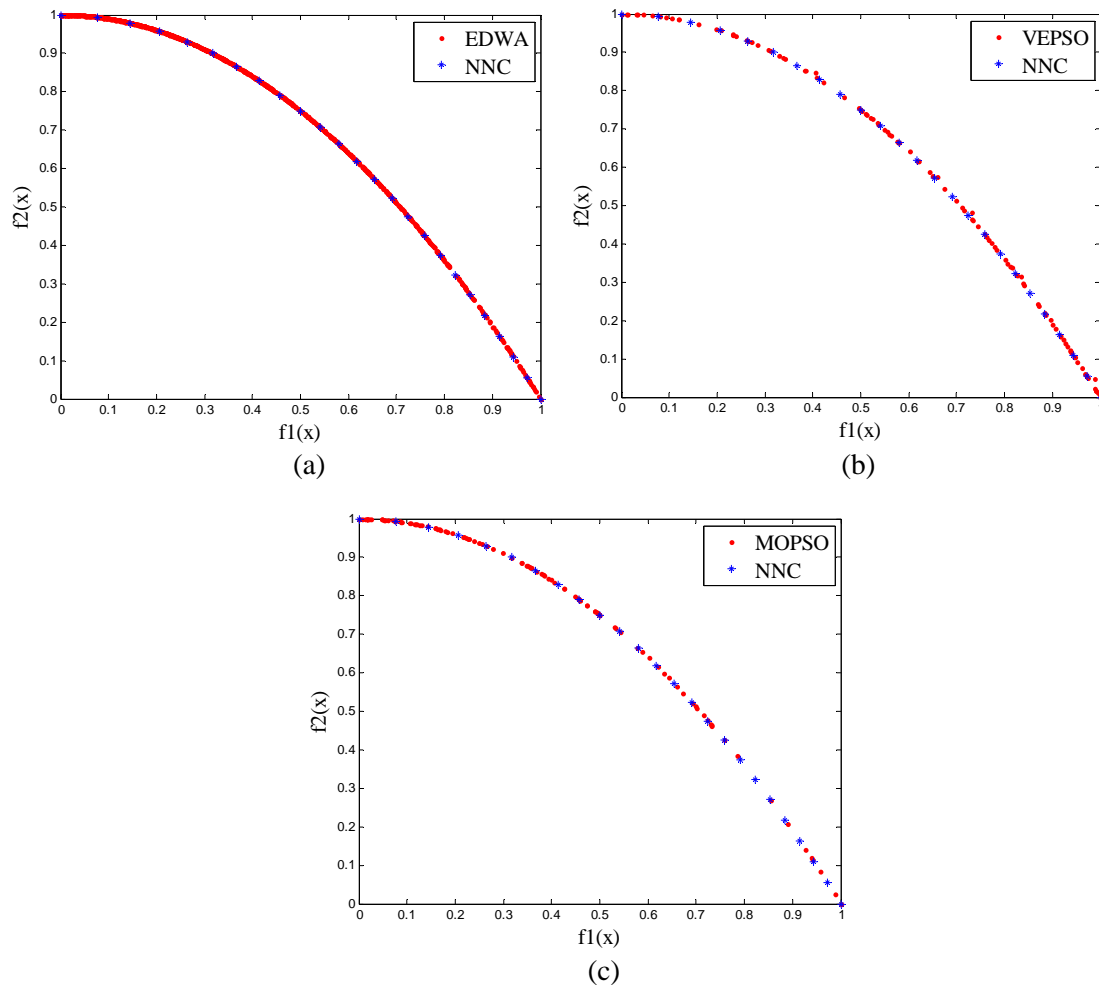


Figura (5.11) - Frente de Pareto encontrada no problema MO3, em novos testes, via (a) EDWA, (b) VEPSO, (c) MOPSO comparadas à encontrada via NNC, considerando dez variáveis em cada função.

Neste problema, o EDWA mostrou-se o mais eficaz, mesmo com os novos resultados obtidos pelo VEPSO e pelo MOPSO, considerando dez variáveis o EDWA novamente apresentou uma curva com a melhor distribuição.

#### 5.2.2.4 Exemplo 4: problema MO4

O problema MO4 é dado por:

Minimize  $\mathbf{F}(f_1(\mathbf{x}), f_2(\mathbf{x}))$ , onde:

$$f_1(\mathbf{x}) = x_1 \text{ e } f_2(f_1, g) = g \left( 1 - \sqrt[4]{f_1/g} - (f_1/g)^4 \right) \quad (5.6)$$

$$\text{onde } g = g(\mathbf{x}) = 1 + \frac{9}{n-1} \sum_{i=2}^n x_i$$

Para este problema a frente de Pareto a ser encontrada possui um trecho convexo e outro não-convexo e o número de variáveis consideradas também é  $n = 20$ , para todos os otimizadores utilizados.

Na figura (5.12) é mostrada a forma da frente de Pareto encontrada por cada um deles. Dos resultados obtidos com o PSO, foi selecionado apenas um de cada metodologia do PSO, obtidos nas vinte execuções, e sobreposto ao resultado obtido com o NNC.

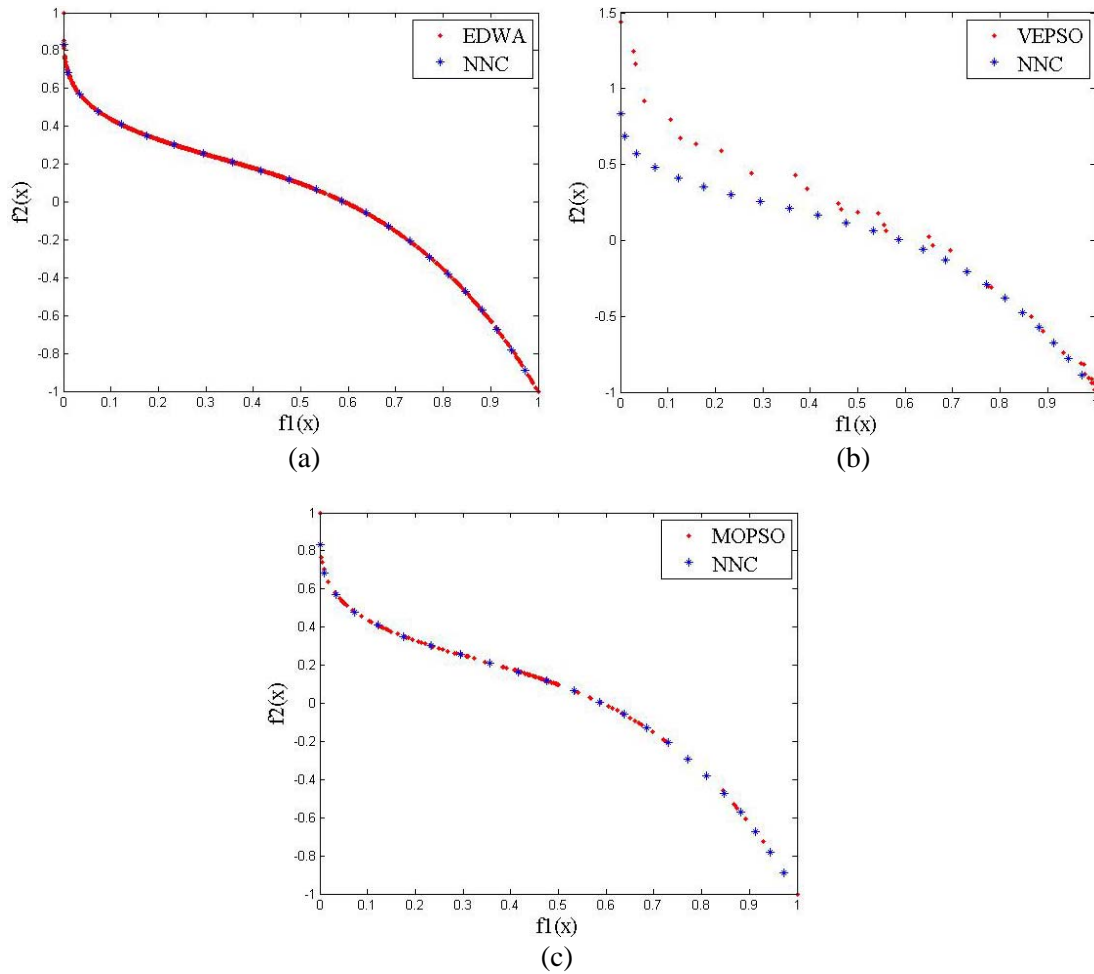


Figura (5.12) – Frentes de Pareto encontradas no problema MO4 via (a) EDWA, (b) VEPSO, (c) MOPSO comparadas à encontrada via NNC.

Nas técnicas adotadas com o PSO multiobjetivo a configuração considerada para os parâmetros internos foram os mesmos adotados no exemplo 1. Os parâmetros específicos de cada técnica também são semelhantes aos considerados anteriormente. Na tabela (5.13) são apresentadas as quantidades de soluções de Pareto encontradas com as técnicas utilizadas no PSO e na NNC.

Tabela (5.13) – Número médio de soluções de Pareto encontradas por cada metodologia considerada – exemplo MO4.

Metodologia	Nº médio de soluções de Pareto
EDWA	1089
VEPSO	57
MOPSO	119
NNC	23

O valor indicado para as técnicas do PSO representam uma média do número de soluções em todas as execuções. Os valores mostrados desta vez foram muito menores do que os mostrados nos exemplos anteriores apenas no VEPSO, além disso, como pode ser visto na figura (5.12), as curvas encontradas com o VEPSO e com o MOPSO não foram tão satisfatórias. Da mesma forma que nos casos anteriores, o motivo deste problema ter ocorrido não foi identificado. Devido a isso, novos testes foram realizados com o intuito de tentar identificar o motivo que causou tais problemas.

Para os novos testes foram consideradas novas configurações dos parâmetros internos do PSO, porém resultados satisfatórios mais uma vez só foram obtidos quando o número de variáveis do problema foi diminuído para dez. Para comparação entre o desempenho, os novos testes foram executados com todas as versões do algoritmo PSO para problemas multiobjetivo. A figura (5.13) apresenta os resultados obtidos com uma dos resultados obtidos, com cada uma das metodologias sobrepostas aos resultados obtidos com o NNC, todos considerando dez variáveis.

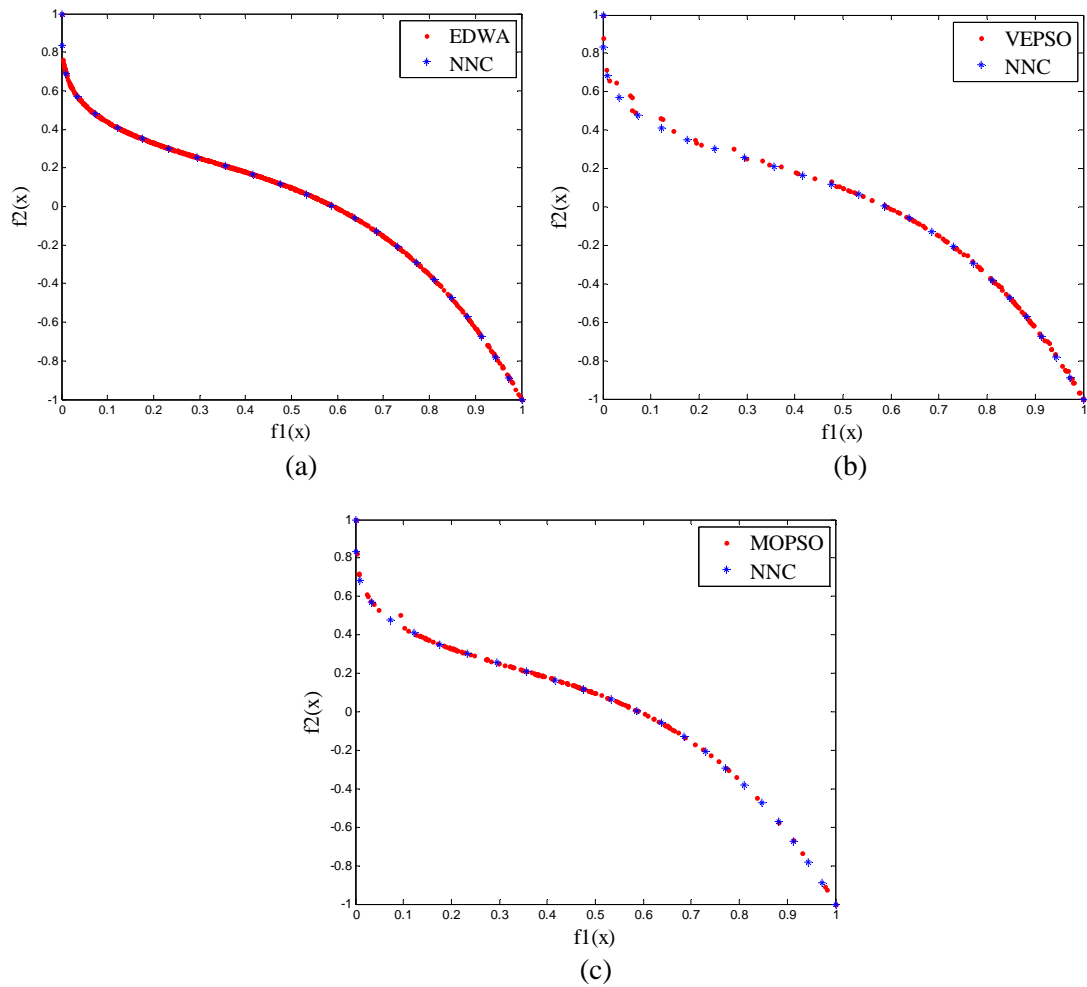


Figura (5.13) - Frente de Pareto encontrada no problema MO4, em novos testes, via (a) EDWA, (b) VEPSO, (c) MOPSO comparadas à encontrada via NNC, considerando dez variáveis em cada função.

Outra vez o EDWA comportou-se de forma excelente varrendo toda a frente de Pareto, o mesmo mostrou-se o mais eficaz dentre todas as metodologias utilizadas.

#### 5.2.2.5 Exemplo 5: problema MO5

O problema MO5 é dado por:

Minimize  $\mathbf{F}(f_1(\mathbf{x}), f_2(\mathbf{x}))$ , onde:

$$f_1(\mathbf{x}) = x_1 \text{ e } f_2(f_1, g) = g \left( 1 - \sqrt{f_1/g} - (f_1/g) \sin(10\pi f_1) \right) \quad (5.7)$$

$$\text{onde } g = g(\mathbf{x}) = 1 + \frac{9}{n-1} \sum_{i=2}^n x_i$$

Para este problema a frente de Pareto a ser encontrada possui trechos convexos (a superfície de Pareto é descontínua) e o número de variáveis consideradas também é  $n = 20$ , para todos os otimizadores utilizados.

Na figura (5.14) é mostrada a forma da frente de Pareto encontrada por cada um deles. Dos resultados obtidos foi selecionado apenas um entre as vinte execuções de cada metodologia do PSO e sobreposto ao resultado obtido com o NNC.

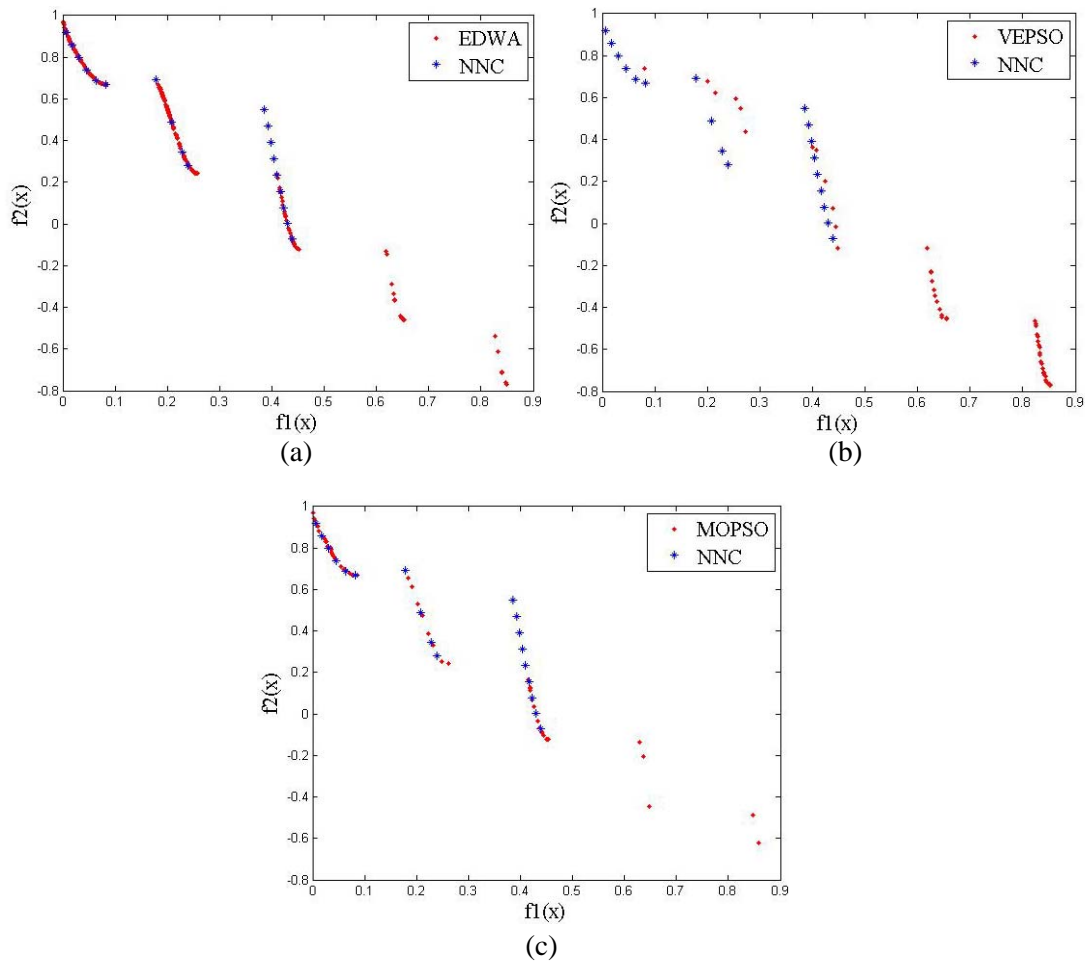


Figura (5.14) – Frentes de Pareto encontradas no problema MO5 via (a) EDWA, (b) VEPSO, (c) MOPSO comparadas à encontrada via NNC.



Nas técnicas adotadas com o PSO multiobjetivo a configuração considerada para os parâmetros internos foram os mesmos adotados no exemplo 1. Os parâmetros específicos de cada técnica também são semelhantes aos considerados anteriormente.

Na tabela (5.14) são apresentadas as quantidades de soluções de Pareto encontradas com as técnicas utilizadas no PSO e na NNC.

Tabela (5.14) – Número médio de soluções de Pareto encontradas por cada metodologia considerada – exemplo MO5.

Metodologia	Nº médio de soluções de Pareto
EDWA	246
VEPSO	40
MOPSO	99
NNC	19

O valor indicado para as técnicas do PSO representam uma média do número de soluções em todas as execuções. Os valores mostrados desta vez foram menores do que os mostrados nos exemplos anteriores como pode ser visto na figura (5.13), porém as curvas encontradas apresentam um resultado melhor do que o obtido via NNC.

Neste problema o NNC encontrou apenas 19 pontos diferentes como indicado na tabela (5.14), além disso, nos resultados apresentados na figura (5.13) pode-se perceber que alguns pontos encontrados não são pontos de Pareto, pois representam soluções dominadas. Este fato é devido à descontinuidade da frente de Pareto.

O NNC não foi o único a apresentar dificuldades em encontrar todos os trechos da superfície de *tradeoff*, o VEPSO e o MOPSO também apresentaram resultados que não foram satisfatórios. Como em todos os exemplos anteriores, testes com essas metodologias para identificar o motivo de tais dificuldades foram realizados, com exceção do NNC, pois como a superfície é formada por vários trechos (a superfície é descontínua) e devido a limitações do NNC, o método fornece como resultados vários pontos dominados até encontrar uma solução não dominada do novo trecho.

Mesmo após os novos testes com as mudanças nos parâmetros do PSO, a causa das falhas na determinação da frente de Pareto não foi identificada, porém quando foi feita uma diminuição no número de variáveis consideradas em cada função do problema os algoritmos apresentaram resultados bastante satisfatórios quando comparado com o caso em que  $n = 20$ .

Nos novos testes cada metodologia foi executada vinte vezes considerando  $n=10$ . Por questão de comparação entre o desempenho, os novos testes foram feitos com todas as versões do algoritmo PSO para problemas multiobjetivo. Os resultados obtidos pelas metodologias do PSO sobrepostos aos resultados obtidos com o NNC, todos considerando dez variáveis, são apresentados na figura (5.15).

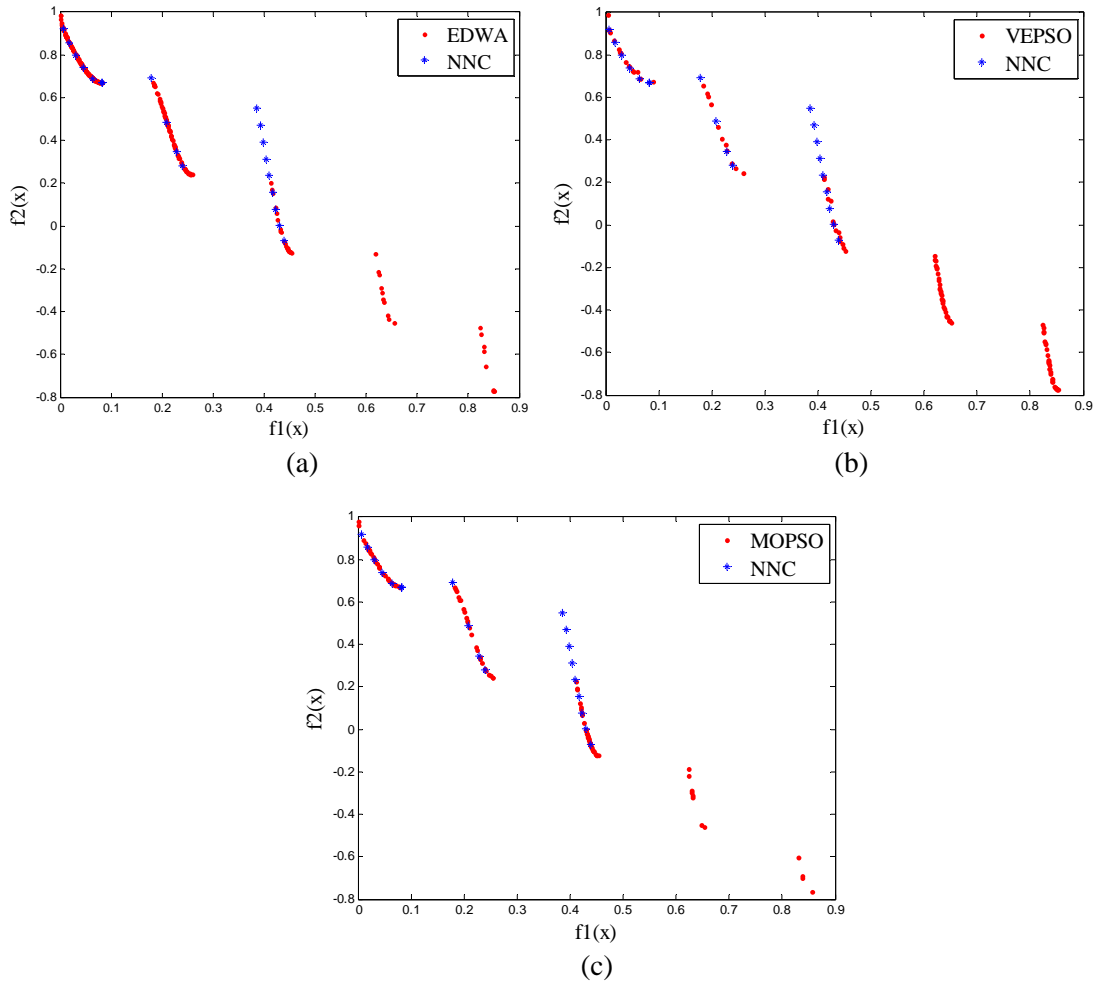


Figura (5.15) - Frente de Pareto encontrada no problema MO5, em novos testes, via (a) EDWA, (b) VEPSO, (c) MOPSO comparadas à encontrada via NNC, considerando dez variáveis em cada função.

Neste problema, o EDWA mais uma vez demonstrou-se o mais eficaz. Os resultados obtidos com os novos testes realizados com o VEPSO e com o MOPSO apresentaram uma curva melhor distribuída, em relação aos testes iniciais.

### 5.2.2.6 Exemplo 6: problema MO6

O problema MO6 é dado por:

Minimize  $\mathbf{F}(f_1(\mathbf{x}), f_2(\mathbf{x}))$ , onde:

$$f_1(\mathbf{x}) = 1 - e^{(-4x_1)} \sin^6(6\pi x_1) \text{ e } f_2(g, h) = g \times h \quad (5.8)$$

$$\text{onde } g = g(\mathbf{x}) = 1 + 9\sqrt[4]{\sum_{i=2}^n x_i} / 9 \text{ e } h = h(f_1, g) = 1 - \sqrt{f_1 / g}$$

Para este problema a frente de Pareto a ser encontrada é não-convexa, descontínua e o número de variáveis consideradas é  $n = 10$ , para todos os otimizadores utilizados.

Nas técnicas adotadas com o PSO multiobjetivo a configuração considerada para os parâmetros internos foram os mesmos adotados no exemplo 1. Os parâmetros específicos de cada técnica também são semelhantes aos considerados anteriormente.

Na figura (5.16) é mostrada a forma da frente de Pareto encontrada por cada um dos métodos utilizados na busca de soluções para este problema. Foi selecionado apenas um entre os vinte resultados obtidos em cada metodologia, porém, neste problema não foi mostrado o comparativo das técnicas utilizadas no PSO e o NNC porque além do mesmo apresentar apenas uma solução esta não é uma solução de Pareto, este problema está relacionado com a descontinuidade da frente de Pareto e com a presença de vários mínimos locais na função  $f_1$ . As soluções obtidas podem ser conferidas em (MOSTAGHIM e TEICH, 2004), de onde este problema foi retirado.

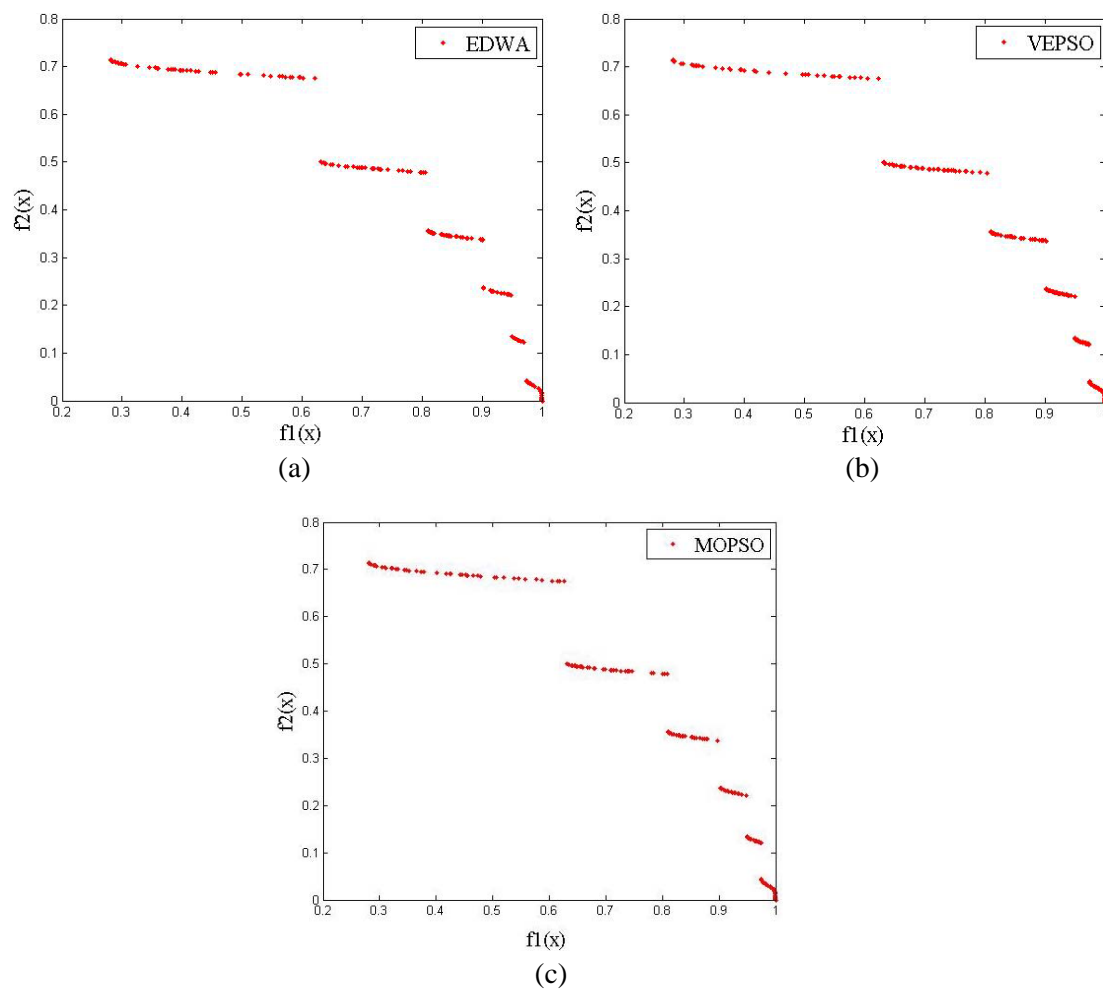


Figura (5.16) – Frentes de Pareto encontradas no problema MO6 via (a) EDWA, (b) VEPSO e (c) MOPSO.

Na tabela (5.15) são apresentadas as quantidades de soluções de Pareto encontradas com as técnicas utilizadas no PSO e na NNC.

Tabela (5.15) – Número médio de soluções de Pareto encontradas por cada metodologia considerada – exemplo MO6.

Metodologia	Nº médio de soluções de Pareto
EDWA	220
VEPSO	132
MOPSO	196
NNC	1

Como visto na tabela (5.15), o NNC encontrou o mesmo ponto em todas as vezes que foi executado. Tal como em todos os casos, o valor indicado para as técnicas do PSO representam uma média do número de soluções em todas as execuções. Os valores encontrados desta vez foram bastante satisfatórios levando em consideração que todas as técnicas encontraram uma quantidade suficiente para determinação da frente de Pareto e com uma boa distribuição ao longo da mesma. Neste exemplo o EDWA mais uma vez demonstrou ser a técnica mais eficaz em meio a todas as consideradas, pois foi o que apresentou uma maior média de quantidade de pontos não-dominados encontrados.

### **5.3 Problemas de otimização estrutural**

Para motivar a utilização do algoritmo PSO em problemas práticos da engenharia, foram estudados alguns casos de otimização estrutural, envolvendo treliças planas.

As aplicações conduzidas nos problemas apresentados nesta seção consistem de treliças planas submetidas a carregamento estático. Este tipo de estrutura foi selecionado porque é a aplicação mais estudada para validar técnicas de otimização em projetos estruturais (PARENTE JR., 1995; AFONSO e HOROWITZ, 1998). Este fato é devido, dentre outros motivos, a análise rápida e direta desse sistema estrutural; a simplicidade da formulação do problema de otimização; a abrangência do porte da estrutura; e ao seu uso prático.

Primeiramente foram realizadas as análises em problemas de otimização uni-objetivo e posteriormente as análises realizadas em problemas multiobjetivo. As funções objetivo e restrições impostas serão descritas à medida que os problemas forem apresentados.

A geometria e o carregamento aplicado serão indicados através de figuras apresentadas em cada exemplo. Nas estruturas consideradas todos os elementos são feitos do mesmo material e o módulo de elasticidade adotado em todos os problemas foi  $E = 203 \text{ GPa}$ .

### 5.3.1 Otimização uni-objetivo

Foram escolhidas três treliças para validar o uso do PSO em problemas de otimização estrutural, para isso, é mostrado um comparativo entre os resultados obtidos com o OPT\_PS e os resultados obtidos com o OPTRUSS (AFONSO e HOROWITZ, 1998) que faz uso do SQP para realizar o processo de otimização.

Além dos estudos para validar o uso do PSO, foi estudado o comportamento do algoritmo em relação às modificações feitas nos parâmetros do mesmo. Também são apresentados os resultados obtidos considerando o método das bases reduzidas (*Reduced Basis Method* – RBM) colocando em destaque a vantagem da utilização de um modelo substituto robusto em relação ao tempo computacional do mesmo sobre outras abordagens.

Para análise desses problemas, o algoritmo PSO foi executado trinta vezes em cada e o OPTRUSS foi executado apenas uma vez para obtenção do resultado que serve de referência para as comparações a serem realizadas.

#### 5.3.1.1 Exemplo 1: treliça com 10 barras

Como primeiro exemplo foi considerado uma treliça de dez barras apenas por questão de didática, pois é um problema bastante simples. Neste problema o objetivo a ser otimizado é o volume de material utilizado na estrutura e as restrições impostas ao problema são limitações nas tensões das barras da treliça.

O volume da treliça é encontrado através da equação (5.9).

$$vol = \sum_{i=1}^{nelem} A_i L_i \quad (5.9)$$

onde  $A$  e  $L$  representam, respectivamente, a área e o comprimento de cada barra da treliça e  $nelem$  é o número de elementos que compõem a treliça.

As tensões nas barras são encontradas através da equação (5.10).

$$\sigma^e = N^e / A^e \quad (5.10)$$

onde  $N^e$  e  $A^e$  são, respectivamente, o esforço normal e a seção transversal do elemento  $e$  da treliça.

Os esforços em cada elemento são obtidos com o pós-processamento dos resultados obtidos na análise realizada através do FEM, considerando a equação (5.11).

$$\mathbf{f}^e = \mathbf{K}^e \times \mathbf{u}^e \quad (5.11)$$

onde  $\mathbf{K}^e$  é a matriz de rigidez do elemento, e  $\mathbf{f}^e$  e  $\mathbf{u}^e$  são, respectivamente, o vetor de esforços e deslocamentos nodais do elemento  $e$ .

No problema o valor máximo permitido para tensão em qualquer barra é  $\sigma = 25 \text{ MPa}$  (tanto para tração quanto para compressão).

São consideradas três variáveis de projeto as quais representam as seções transversais das barras tal como especificado nas regiões destacadas na figura (5.17). As variáveis estão limitadas entre  $0.1 \text{ cm}^2 \leq x_i \leq 10 \text{ cm}^2$ . A figura (5.17) mostra a forma da estrutura estudada. Cada barra da treliça está associada a uma das variáveis como mostra a figura, e na mesma também pode ser visto o carregamento ao qual a treliça está submetida.

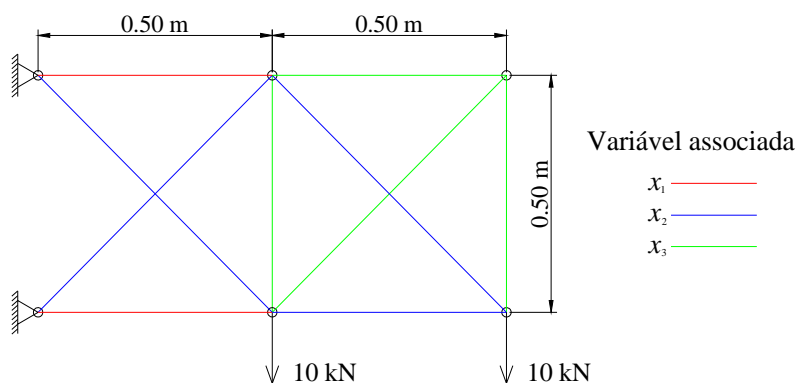


Figura (5.17) – Treliça de dez barras.

Na otimização com o PSO, foi considerada seguinte configuração de parâmetros internos:

- o número máximo de iterações permitidas foi 3000;
- os parâmetros de vizinhança considerados foram  $c_1 = c_2 = 2$ ;

- a inércia do foi iniciada com  $w = 1.4$  podendo ser reduzido a 0.35, como descrito no capítulo 4;
- a quantidade de partículas consideradas foi  $np = 20$ .

Na tabela (5.16) são apresentados os resultados obtidos com o código OPT\_PS e com o OPTRUSS.

Tabela (5.16) – Resultados da otimização da treliça de 10 barras com o OPT\_PS e com o OPTRUSS.

CÓDIGO	$x_1^* (cm^2)$	$x_2^* (cm^2)$	$x_3^* (cm^2)$	$f(\mathbf{x}^*) (m^3)$	Avaliação de funções	$\mu(f(\mathbf{x}^*))$	$\sigma(f(\mathbf{x}^*))$
OPT_PS	8.047558	5.724117	0.10	$2.327 \times 10^{-3}$	20500	$2.373 \times 10^{-3}$	$8.402 \times 10^{-5}$
OPTRUSS	8.047566	5.724122	0.10	$2.327 \times 10^{-3}$	11		

O resultado apresentado com o OPT\_PS mostra o melhor resultado obtido dentre todos os encontrados nas execuções realizadas, também são indicados os valores da média e do desvio padrão dos mesmos. Os resultados demonstram que o OPT\_PS convergiu para solução encontrada pelo OPTRUSS.

O número de avaliações de funções realizadas pelo OPT\_PS foi bem superior ao número de avaliações realizadas pelo OPTRUSS. Não há como discutir a eficiência do OPTRUSS perante o OPT\_PS para este problema, mesmo o primeiro fazendo uso de gradientes para encontrar a solução ótima do problema, porém fica registrada a eficácia do PSO em problemas estruturais.

Como pode ser observado, os valores da média e do desvio padrão indicam que o OPT\_PS não encontrou o resultado ótimo obtido pelo OPTRUSS em todas as vezes, porém o pior resultado obtido pelo OPT\_PS apresenta um erro de 8,39% no valor da função objetivo.

Para estudar o desempenho do PSO para este tipo de problema, algumas variações nos parâmetros internos do algoritmo foram consideradas. Os resultados obtidos serão discutidos a seguir.



- Consideração de valor fixo da inércia

Com o intuito de verificar a influência da inércia em problemas de otimização estrutural, diferentes valores fixos para a mesma foram considerados e novas execuções foram realizadas.

Como considerado nas verificações feitas de maneira semelhante para as funções analíticas, os valores fixos considerados variam entre os limites estabelecidos para inércia. A tabela (5.17) apresenta os resultados encontrados para os diversos valores de  $w$  considerados.

Tabela (5.17) – Resultados da otimização da treliça de 10 barras considerando diferentes valores fixos de inércia.

$w$	$x_1^* (cm^2)$	$x_2^* (cm^2)$	$x_3^* (cm^2)$	$f(\mathbf{x}^*) (m^3)$	$\mu(f(\mathbf{x}^*))$	$\sigma(f(\mathbf{x}^*))$
0.35	8.047558	5.724117	0.10	$2.327 \times 10^{-3}$	$2.366 \times 10^{-3}$	$7.947 \times 10^{-5}$
0.50	8.047558	5.724117	0.10	$2.327 \times 10^{-3}$	$2.379 \times 10^{-3}$	$8.786 \times 10^{-5}$
0.65	8.047558	5.724117	0.10	$2.327 \times 10^{-3}$	$2.386 \times 10^{-3}$	$9.104 \times 10^{-5}$
0.80	8.047558	5.724117	0.10	$2.327 \times 10^{-3}$	$2.405 \times 10^{-3}$	$9.818 \times 10^{-5}$
1.00	8.047558	5.724117	0.10	$2.327 \times 10^{-3}$	$2.381 \times 10^{-3}$	$8.733 \times 10^{-5}$
1.20	8.047558	5.724117	0.10	$2.327 \times 10^{-3}$	$2.327 \times 10^{-3}$	$8.298 \times 10^{-9}$
1.40	8.047558	5.724117	0.10	$2.327 \times 10^{-3}$	$2.353 \times 10^{-3}$	$6.754 \times 10^{-5}$

Considerando as modificações feitas no valor de  $w$ , o algoritmo apresentou comportamento semelhante ao modelo padrão, que consiste no valor de  $w$  atualizado dinamicamente ao longo do processo de busca, para este tipo de problema uma vez que a ordem de grandeza dos desvios-padrão apresentados não foi alterada.

- Consideração de diferentes parâmetros de confiança

Para testar a influência dos parâmetros de confiança,  $c_1$  e  $c_2$ , no processo de otimização de problemas estruturais, novos testes foram realizados onde os valores de  $c_1$  eram determinados e fazia-se  $c_2 = 4 - c_1$ .

Como nos testes realizados com as funções analíticas, as variações foram realizadas em duas etapas, a primeira considerava uma atualização dinâmica para os parâmetros em cada iteração e a segunda considerava um valor fixo para os mesmos durante toda a busca.

A tabela (5.18) mostra os resultados obtidos na primeira etapa e a tabela (5.19) mostra os resultados obtidos na segunda.

Tabela (5.18) – Resultados da otimização considerando variações dinâmicas nos parâmetros de confiança

$c_1$	$x_1^* (cm^2)$	$x_2^* (cm^2)$	$x_3^* (cm^2)$	$f(\mathbf{x}^*) (m^3)$	$\mu(f(\mathbf{x}^*))$	$\sigma(f(\mathbf{x}^*))$
Aumentando (partindo de 0.5)	8.047558	5.724117	0.10	$2.327 \times 10^{-3}$	$2.438 \times 10^{-3}$	$9.845 \times 10^{-5}$
Diminuindo (partindo de 3.5)	8.047558	5.724117	0.10	$2.327 \times 10^{-3}$	$2.373 \times 10^{-3}$	$8.403 \times 10^{-5}$

Tabela (5.19) – Resultados da otimização considerando variações dinâmicas nos parâmetros de confiança

$c_1$	$x_1^* (cm^2)$	$x_2^* (cm^2)$	$x_3^* (cm^2)$	$f(\mathbf{x}^*) (m^3)$	$\mu(f(\mathbf{x}^*))$	$\sigma(f(\mathbf{x}^*))$
1.0	8.047558	5.724117	0.10	$2.327 \times 10^{-3}$	$2.392 \times 10^{-3}$	$9.365 \times 10^{-5}$
1.5	8.047558	5.724117	0.10	$2.327 \times 10^{-3}$	$2.366 \times 10^{-3}$	$8.016 \times 10^{-5}$
2.5	8.047558	5.724117	0.10	$2.327 \times 10^{-3}$	$2.347 \times 10^{-3}$	$5.960 \times 10^{-5}$
3.0	8.047558	5.724117	0.10	$2.327 \times 10^{-3}$	$2.345 \times 10^{-3}$	$6.013 \times 10^{-5}$

Para as considerações feitas nos parâmetros de confiança o algoritmo apresentou neste tipo de problema um comportamento semelhante aos apresentados nas considerações feitas anteriormente no mesmo, a consideração da configuração padrão e a determinação dos valores da inércia, pois os desvios-padrão são da mesma ordem de grandeza.

### 5.3.1.2 Exemplo 2: treliça com 200 barras

Neste segundo exemplo de otimização de problemas estruturais, uma treliça de duzentas barras foi considerada com o objetivo de demonstrar a eficácia do PSO em problemas com um elevado número de graus de liberdade.

Neste problema o objetivo a ser otimizado mais uma vez é o volume de material utilizado na estrutura e as restrições impostas ao problema são limitações nos deslocamentos dos nós livres.

O volume da treliça é dado pela equação (5.9) que foi apresentada no problema anterior e os deslocamentos são encontrados através da resolução da equação (4.22), considerando  $\mathbf{u}$  (o vetor de deslocamentos dos nós da treliça) as incógnitas do sistema.

No problema, os deslocamentos encontrados estão limitados entre  $-0.001\text{cm} \leq u_i \leq 0.001\text{cm}$ . Neste problema também são consideradas três variáveis de projeto as quais representam as seções transversais das barras e estão limitadas entre  $0.1\text{ cm}^2 \leq x_i \leq 100\text{ cm}^2$ . A estrutura e a associação entre as barras da treliça e as variáveis de projeto são mostradas na figura (5.18). Na figura também pode ser visto o carregamento ao qual a estrutura está submetida.

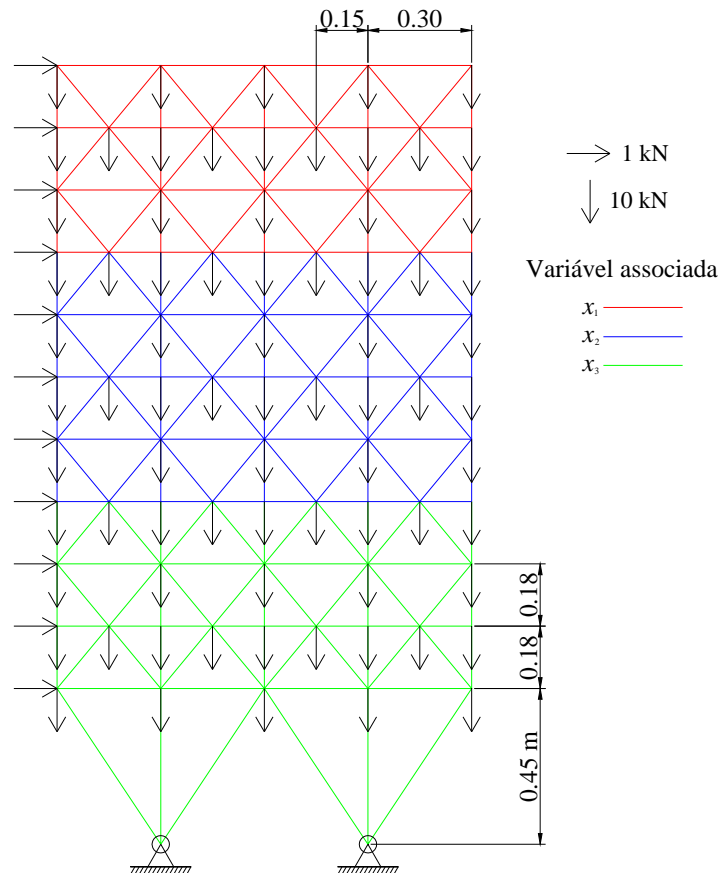


Figura (5.18) – Treliça de duzentas barras.

Na otimização com o PSO, a configuração de parâmetros internos considerada foi igual à considerada no exemplo anterior. Na tabela (5.20) são apresentados os resultados obtidos com o OPT\_PS e com o OPTRUSS.

Tabela (5.20) – Resultados da otimização da treliça de 200 barras com o OPT\_PS e com OPTRUSS.

CÓDIGO	$x_1^* (cm^2)$	$x_2^* (cm^2)$	$x_3^* (cm^2)$	$f(\mathbf{x}^*) (m^3)$	Avaliação de funções	$\mu(f(\mathbf{x}^*))$	$\sigma(f(\mathbf{x}^*))$
OPT_PS	0.336403	0.590382	1.23462	$3.230 \times 10^{-2}$	53600	$3.230 \times 10^{-2}$	$1.358 \times 10^{-14}$
OPTRUSS	0.336403	0.590383	1.23462	$3.230 \times 10^{-2}$	41		

O resultado apresentado para o OPT\_PS mostra o melhor resultado obtido dentre os encontrados nas análises realizadas nesse problema, também são mostrados o valor da média e do desvio padrão. Como pode ser percebido, o resultado do OPT\_PS convergiu para a solução encontrada pelo OPTRUSS.

Neste exemplo, o número de avaliações de funções realizadas pelo OPT\_PS também foi bem superior ao número de avaliações realizadas pelo OPTRUSS e mais uma vez é mostrada não há como discutir a eficiência do OPTRUSS perante o OPT\_PS, porém fica mais uma vez registrada a eficácia do PSO em problemas estruturais.

Neste problema, devido ao grande número de graus de liberdade apresentado pela estrutura, o tempo de análise para obtenção dos resultados foi bastante longo. Com o intuito de testar a eficácia do algoritmo do OPT\_PS com técnicas que visam à melhoria da eficiência computacional o método das bases reduzidas (RBM) e a codificação do algoritmo utilizando paradigmas da programação paralela, os quais apresentados no capítulo anterior, foram aqui implementados criando o OPT\_PSRbm e OPT\_PSpár.

Um dos problemas encontrados na implementação do RBM foi a determinação do número de amostras necessárias para obtenção dos resultados. Vários testes foram realizados, modificando a quantidade de amostras até encontrar o número suficiente. Os testes realizados com o RBM seguiram o padrão dos testes realizados com o FEM, tanto para o número de execuções quanto para os parâmetros do OPT\_PSRbm.

A figura (5.19) mostra um gráfico que representa a relação entre o número de amostras consideradas e a convergência dos resultados obtidos com RBM para os resultados encontrados com o FEM.

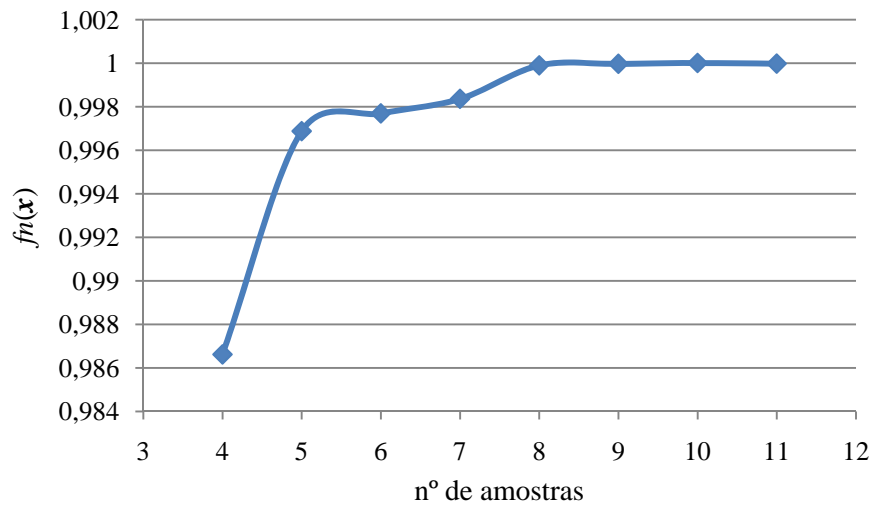


Figura (5.19) – Gráfico da relação entre o número de amostras fornecidas e a convergência dos resultados encontrados pelo OPT\_PSrbm para os resultados encontrados pelo OPT\_PS.

No gráfico  $f_n(\mathbf{x}) = f_{RBM}(\mathbf{x})/f_{FEM}(\mathbf{x})$  e como pode ser observado foram necessárias dez amostras para que o resultado obtido utilizando o RBM convergisse para o resultado obtido com o FEM. As amostras consideradas não podia ser múltiplas uma das outras isoladamente (a amostra  $S^i$  devia ser diferente de  $\alpha S^n$ , com  $i \neq n$ ) porém uma amostra podia ser uma combinação linear de outras amostras.

A tabela (5.21) mostra os resultados da otimização do problema aqui estudado, obtidos com o algoritmo OPT\_PSrbm utilizando dez amostras (que estão indicadas na tabela (5.22)). Também são apresentados os resultados obtidos com o OPT\_PS.

Tabela (5.21) – Resultados da otimização da treliça de 200 barras com o OPT\_PS e com o OPT\_PSrbm.

	OPT_PS	OPT_PSrbm
$x_1^* (cm^2)$	0.336403	0.336403
$x_2^* (cm^2)$	0.590382	0.590382
$x_3^* (cm^2)$	1.234617	1.234617
$f(\mathbf{x}) (m^3)$	$3.230 \times 10^{-2}$	$3.230 \times 10^{-2}$
$\mu(f(\mathbf{x}))$	$3.230 \times 10^{-2}$	$3.230 \times 10^{-2}$
$\sigma(f(\mathbf{x}))$	$1.358 \times 10^{-14}$	$1.974 \times 10^{-15}$
$\bar{t}$	6.6733	1

Tabela (5.22) – Amostragem utilizada para a otimização do exemplo utilizando o OPT\_PSrbm.

VARIÁVEL	AMOSTRA									
	$S_1$	$S_2$	$S_3$	$S_4$	$S_5$	$S_6$	$S_7$	$S_8$	$S_9$	$S_{10}$
$\mu_1$	0.1	10.0	0.1	10.0	10.0	0.1	0.1	0.1	5.0	0.1
$\mu_2$	0.1	0.1	10.0	10.0	0.1	10.0	0.1	5.0	0.1	10.0
$\mu_3$	0.1	0.1	0.1	0.1	10.0	10.0	5.0	0.1	0.1	5.0

Na tabela acima o valor de  $\bar{t}$  representa o tempo normalizado em relação ao tempo levado para otimização considerando a técnica de aproximação. Pelos resultados mostrados fica clara a viabilidade do emprego do RBM no processo de otimização com o algoritmo PSO uma vez que a redução no tempo foi bastante significativa.

No caso da implementação do algoritmo na versão paralelizada, o problema encontrado foi indisponibilidade do *cluster* de computadores devido a adversidades durante o período de testes. Em consequência, a programação foi feita no ambiente MATLAB (R2007b – versão 7.5.0.342), que contém o *parallel computing toolbox*. O computador utilizado para execução possuía em sua configuração um processador Intel® CORE™ QUAD (Q6600 @ 2.40 GHz), o que possibilitava a utilização do *toolbox* citado anteriormente.

Os testes realizados com o OPT\_PSpar, não apresentam a mesma riqueza de detalhes que as outras análises devido a utilização apenas de quatro unidades para o processamento das operações, onde cada unidade representava uma partícula. Para comparar os resultados obtidos, novos testes com o OPT\_PS considerando apenas quatro partículas foram realizados. Devido ao fato do número partículas ter sido reduzido, a busca foi prejudicada tanto na versão paralelizada quanto na versão sequencial, porém os resultados foram satisfatórios. Na tabela (5.23) são apresentados os resultados obtidos nos testes.

Tabela (5.23) – Resultados da otimização da treliça de 200 barras com o OPT\_PS e com o OPT\_PSparr.

	OPT_PS	OPT_PSparr
$x_1^* (cm^2)$	0,339024	0,310053
$x_2^* (cm^2)$	0,747384	0,604206
$x_3^* (cm^2)$	1,113443	1,245421
$f(\mathbf{x}) (m^3)$	$3,302 \times 10^{-2}$	$3,234 \times 10^{-2}$
$\bar{t}$	2,8970	1

Com os resultados apresentados percebe-se a viabilidade da implementação dos PSO na versão paralelizada, uma vez que a redução no tempo foi significativa.

Fica claro que o ganho de eficiência com o uso das técnicas testadas são opções a serem consideradas na melhoria da eficiência computacional do PSO.

### 5.3.1.3 Exemplo 3: treliça com 940 barras

Como terceiro exemplo de otimização de problemas estruturais, uma treliça de 940 barras foi adotada. Neste problema, além de um elevado número de graus de liberdade, o problema formulado tem o intuito de demonstrar a vantagem da utilização do PSO em relação aos algoritmos convencionais em alguns problemas de engenharia.

O objetivo foi minimizar o deslocamento máximo sofrido pela estrutura, enquanto a função de restrição considerada foi o volume constante. Os algoritmos tradicionais que usam informações de gradientes encontram dificuldades na otimização deste problema porque o gradiente da função objetivo muda de direção constantemente mesmo para pequenas perturbações nas variáveis de projeto durante o processo de busca.

Nestes algoritmos o gradiente aponta na direção de minimização da função, porém, neste problema, a minimização do deslocamento de um determinado ponto pode fazer com que outro ponto que apresentava um deslocamento menor do que o máximo encontrado anteriormente passe a ser o máximo na estrutura quando o primeiro for diminuído.

A mudança da localização do ponto de máximo deslocamento faz com que o gradiente seja recalculado apontando em outra direção, até mesmo contrária à da busca realizada até o momento. Isso caracteriza uma não homogeneidade na função objetivo, ou seja, a função não é continuamente diferenciável.

No problema, o volume é restrito ao valor  $vol = 1.5 \text{ m}^3$ . Neste problema também são consideradas três variáveis de projeto as quais representam as seções transversais das barras da treliça e são limitadas entre  $0.1 \text{ cm}^2 \leq x_i \leq 100 \text{ cm}^2$ . Na figura (5.20) são mostradas a estrutura analisada e a associação entre as barras da treliça e as variáveis de projeto. Na figura também é mostrado o carregamento ao qual a estrutura está submetida.

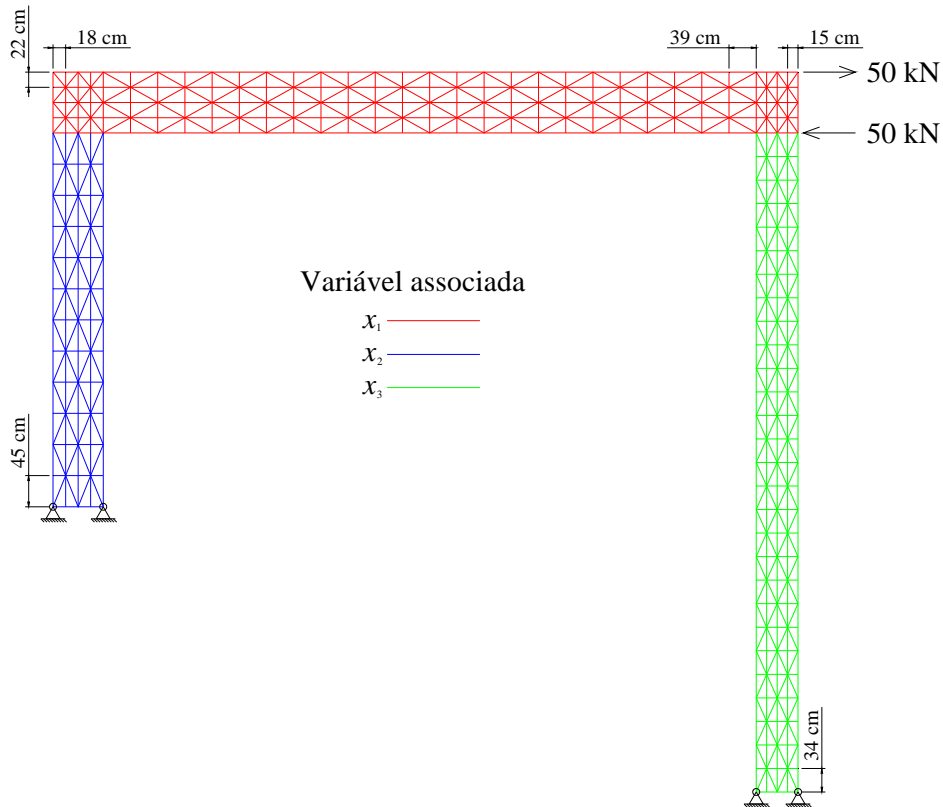


Figura (5.20) – Treliça de 940 barras

Na otimização com o OPT\_PS, a configuração de parâmetros internos considerada foi igual à considerada nos exemplos anteriores com exceção do número máximo de iterações que foi aumentado para 10000. Na tabela (5.24) são apresentados os resultados obtidos com o OPT\_PS e com o OPTRUSS.



Tabela (5.24) – Resultados da otimização da treliça de 940 barras com o OPT\_PS e com o OPTRUSS.

CÓDIGO	$x_1^*$ ( $cm^2$ )	$x_2^*$ ( $cm^2$ )	$x_3^*$ ( $cm^2$ )	$f(\mathbf{x}^*)$ ( $m^3$ )	Avaliação de funções	$\mu(f(\mathbf{x}^*))$	$\sigma(f(\mathbf{x}^*))$
OPT_PS	10.964569	0.137972	0.189120	0.195083	135800	0.201039	$7.322 \times 10^{-4}$
OPTRUSS	9.399377	1.161802	1.601969	0.201797	2777		

É importante ressaltar que o resultado obtido pelo OPTRUSS nesse exemplo foi o melhor obtido em várias tentativas de execução do algoritmo, pois a objetivo selecionado não representa uma função continuamente diferenciável.

Os algoritmos apresentaram respostas diferentes para o problema, porém, diante dos resultados mostrados nos problemas anteriores (que validam a utilização do PSO) e dada à dificuldade do problema para o algoritmo convencional, os resultados fornecidos pelo OPT\_PS são mais confiáveis. Apesar disso, a diferença entre o valor da função objetivo é de apenas 3.33%.

Neste problema, o número de avaliações de funções realizadas pelo OPT\_PS também foi bem superior ao número de avaliações realizadas pelo OPTRUSS. No entanto, diante do número de avaliações realizadas pelo OPTRUSS, considerando o fato do mesmo ter sido executado várias vezes para se obter o resultado mostrado, e considerando também que o OPTRUSS faz uso do cálculo de gradientes (que é uma operação computacionalmente cara), fica claro que em problemas estruturais desta categoria (com alguma descontinuidade) o uso de algoritmos como o PSO são bastante indicados.

De maneira semelhante ao problema anterior o número de graus de liberdade da estrutura é bastante elevado o que tornou a análise bastante demorada, porém neste problema foi considerada apenas o RBM para melhoria da eficiência computacional devido aos problemas citados no exemplo anterior

Mais uma vez, para testar a viabilidade do uso do OPT\_PSRbm, foram feitos testes e os resultados encontram-se na tabela (5.25).

Tabela (5.25) – Resultados da otimização da treliça de 940 barras com o OPT\_PS e com o OPT\_PSrbm.

	OPT_PS	OPT_PSrbm
$x_1^* (cm^2)$	10.964569	10.964322
$x_2^* (cm^2)$	0.137972	0.137127
$x_3^* (cm^2)$	0.189120	0.189902
$f(\mathbf{x}) (m^3)$	0.195083	0.195884
$\mu(f(\mathbf{x}))$	0.201039	0.244701
$\sigma(f(\mathbf{x}))$	$7.322 \times 10^{-3}$	0.177830
$\bar{t}$	36.9496	1

Observa-se que neste caso em particular, que os resultados obtidos utilizando o RBM apresentam um erro da ordem de  $10^{-3}$  em relação aos resultados obtidos utilizando o FEM, o que pode ser considerado satisfatório, principalmente quando é observado o ganho computacional (o tempo gasto pelo OPT\_PSrbm foi 36 vezes menor do que o tempo gasto pelo OPT\_PS).

### 5.3.2 Otimização multiobjetivo

Para validar o uso do PSO em problemas de otimização estrutural multiobjetivo foi selecionado um exemplo retirado de Messac et al. (MESSAC et al., 2003), onde o problema foi formulado visando demonstrar o desempenho do NNC em problemas de otimização de múltiplos objetivos. Os resultados obtidos com o EDWA, o VEPSO e o MOPSO foram comparados aos obtidos com o NNC. O problema apresenta apenas dois objetivos para facilitar a visualização da superfície de Pareto obtida.

Como citado no início, todos os elementos da estrutura são feitos do mesmo material, porém, neste exemplo, o módulo de elasticidade considerado foi  $E = 200 \text{ GPa}$  porque esse foi o valor considerado em Messac et al. (MESSAC et al., 2003).

### 5.3.2.1 Exemplo: treliça de três barras

Apesar da simplicidade da estrutura, este não é um problema tão trivial como aparenta. Para este exemplo, um dos objetivos considerados é o volume de material utilizado na estrutura o outro objetivo é uma combinação linear entre os deslocamentos horizontal e vertical do nó livre da treliça. A combinação linear é dada pela equação (5.11).

$$\Delta = 0.25 \times u_H + 0.75 \times u_V \quad (5.11)$$

No problema as tensões nas barras são limitadas ao valor de  $\sigma = 200 \text{ MPa}$  (tanto para tração quanto para compressão). Neste problema as variáveis estão associadas às seções transversais das barras da treliça e cada barra representa uma variável. As variáveis estão limitadas entre  $0.1 \text{ cm}^2 \leq x_i \leq 2 \text{ cm}^2$ . A figura (5.21) mostra a estrutura analisada e o carregamento ao qual está submetida.

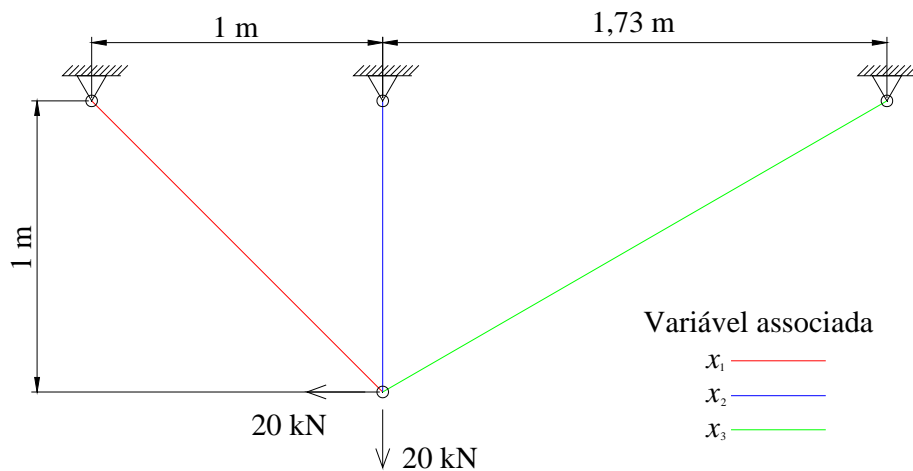


Figura (5.21) – Treliça de três barras.

Na otimização com o PSO, foi considerada seguinte configuração de parâmetros internos:

- o número máximo de iterações permitidas foi 3000;
- os parâmetros de vizinhança considerados foram  $c_1 = c_2 = 2.0$ ;
- a inércia do foi iniciada com  $w = 1.4$  podendo ser reduzido a 0.35, como descrito no capítulo 4;

- a quantidade de partículas consideradas foi  $np = 20$ .

O algoritmo do PSO foi executado vinte vezes com cada uma das metodologias e o algoritmo convencional foi executado 25 vezes, que foi o número de vezes necessárias para obtenção de pontos suficientes para a formação da curva de Pareto.

Na tabela (5.26) é apresentada a quantidade de pontos de Pareto obtidos por cada uma das metodologias.

Tabela (5.26) – Número médio de soluções de Pareto encontradas por cada metodologia considerada – exemplo da treliça de três barras.

Metodologia	Nº médio de soluções de Pareto
EDWA	335
VEPSO	295
MOPSO	191
NNC	25

O valor indicado para as técnicas do PSO representa a média do número de soluções em todas as execuções, e como citado anteriormente nos exemplos de otimização multiobjetivo, o valor indicado para o NNC também representa o número de vezes que o mesmo foi executado.

A figura (5.23) mostra a forma da frente de Pareto encontrada. Dos resultados obtidos foi selecionado apenas um entre as vinte execuções de cada metodologia do PSO e sobreposto ao resultado obtido com o NNC.

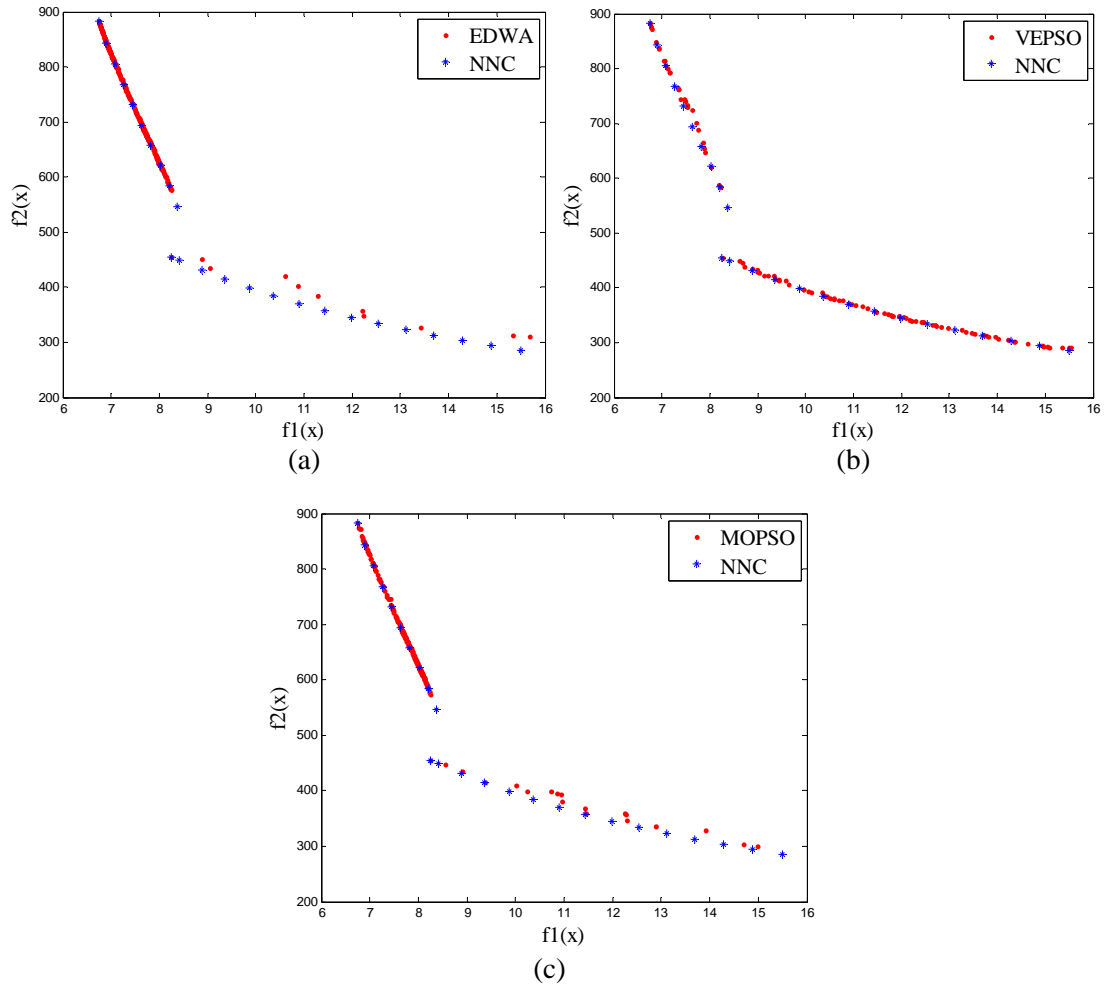


Figura (5.22) – Frentes de Pareto encontradas via (a) EDWA, (b) VEPSO, (c) MOPSO comparadas à encontrada via NNC.

Apesar da grande quantidade de soluções encontradas pelas metodologias evolucionárias, de acordo com os gráficos apresentados na figura (5.23), uma parte da curva não ficou bem definida. Dentre as três metodologias empregadas, o VEPSO foi quem apresentou o melhor resultado pois a frente de Pareto obtida é a mais próxima à encontrada pelo NNC, para qual o problema foi originalmente formulado. Era esperado que o EDWA encontrasse toda frente de Pareto com sucesso pois foi o EDWA quem apresentou os melhores resultados nos exemplos de otimização multiobjetivo de funções analíticas.

Apesar de ser considerado como gabarito para os resultados deste exemplo, uma falha na frente de Pareto encontrada pelo NNC foi detectada. Em um dos trechos mostrados, a curva apresenta pontos dominados. Esta falha não torna inválidos os resultados obtidos pois se deve a descontinuidade da curva e a deficiência do NNC em lidar com tais problemas.

## Conclusões

### 6.1 Realizações

O presente trabalho teve como objetivo apresentar a metodologia evolucionária de otimização via enxame de partículas como uma ferramenta para solução de problemas de otimização. Os diversos tópicos tratados nesta dissertação foram investigados com o intuito de desenvolver uma ferramenta versátil e confiável. Vários códigos foram criados considerando diversos aspectos dentro da teoria do enxame de partículas.

A primeira versão implementada foi utilizada para resolução de problemas de otimização de funções analíticas. Com algumas modificações o algoritmo foi capaz de realizar a otimização de problemas estruturais. Com o avanço dos estudos foram criadas versões que visavam a melhoria da eficiência computacional utilizando modelos substitutos (considerando o método das bases reduzidas) (PRUD'HOMME et al., 2002; AFONSO e PATERA, 2003;) e considerando paradigmas da computação paralela. Por fim, versões do algoritmo PSO para otimização de várias funções objetivos atuando simultaneamente foram implementadas baseadas nas teorias apresentadas na literatura.

A seguir são rapidamente descritos os códigos implementados e suas propriedades.

- OPT\_PS: Este código foi criado inicialmente para lidar com problemas de otimização uni-objetivo considerando funções analíticas. Os problemas considerados tratavam de funções que apresentavam dificuldades para algoritmos que utilizassem informações de gradiente no processo de otimização (descontinuidade da função no domínio de projeto, presença de vários ótimos locais na região investigada, etc.). Os problemas considerados não apresentavam restrições, porém com algumas modificações o código passou a lidar com tais problemas. A partir dessas modificações, junto a adaptações feitas em algumas funções criadas no código anteriormente implementado pela orientadora deste trabalho, o OPTRUSS (AFONSO e HOROWITZ, 1998), testes com problemas estruturais (treliças planas) passaram a ser realizados;
- OPT\_PSrmb: Esta implementação considera a utilização de um modelo substituto (o método das bases reduzidas) (PRUD'HOMME et al., 2002; AFONSO e PATERA, 2003;) para melhoria da eficiência computacional. Este código foi criado a partir de uma modificação do OPT\_PS para testes em problemas estruturais. Nesta versão, as várias chamadas, das avaliações de funções requeridas pelo algoritmo PS, são feitas usando o modelo de bases reduzidas, representando um ganho de tempo computacional significativo em todos os processos;
- OPT\_PSparr: Esta versão foi implementada visando, da mesma forma que o OPT\_PSrmb, a melhoria na eficiência computacional considerando os paradigmas da computação paralela. O OPT\_PSparr, foi desenvolvido a partir do OPT\_PS, e também foi criado pra tratar de problemas estruturais. O OPT\_Psparr encontra-se na fase final de testes;



- EDWA, VEPSO e MOPSO: Estes códigos foram criados com o intuito de fazer com que a otimização via enxame de partículas fosse aplicada a problemas de otimização multiobjetivo. A primeira versão, o EDWA (JIN et al., 2001), apresenta uma ligação com um dos métodos mais tradicionais para problemas de otimização deste tipo, o método da soma ponderada, onde são atribuídos pesos aos objetivos do problema que somados criam uma única função a ser otimizada, porém, no EDWA os pesos agregados são dinamicamente modificados durante o processo de otimização. Outra versão implementada para lidar com este tipo de problema é o VEPSO (SCHAFER, 1984; COELLO e SIERRA, 2004), que é uma versão adaptada do VEGA. O último código criado, porém não menos importante, foi o MOPSO (COELLO et al., 2004), que é a metodologia mais recente dos métodos de otimização multiobjetivo considerando a teoria do enxame de partículas.

## 6.2 Conclusões gerais

Neste trabalho, vários problemas de otimização foram tratados, e com os resultados obtidos pôde se chegar às seguintes conclusões a respeito da utilização da teoria do enxame de partículas:

- Em problemas que apresentam vários ótimos na região investigada o uso do algoritmo PSO foi mais indicado, em relação a algoritmos que consideram informações de gradiente no processo de otimização;
- A presença de descontinuidades na função a ser otimizada, que representa um grande inconveniente para algoritmos que fazem uso da informação de gradientes, não representou problema algum para o algoritmo PSO;
- A consideração de diversas variáveis de projeto no problema não representou dificuldades para o PSO em problemas uni-objetivo;
- O uso do PSO em problemas considerando múltiplos objetivos apresentou resultados bastante satisfatórios, uma vez que as metodologias utilizadas foram bem sucedidas nas diversidades de problemas impostos;

- Apesar dos inconvenientes encontrados pelo VEPSO (SCHAFFER, 1984; COELLO e SIERRA, 2004) e pelo MOPSO (COELLO et al., 2004), ao lidar com um número elevado de variáveis de projeto, o EDWA (JIN et al., 2001) demonstrou-se uma ferramenta poderosa a ser utilizadas em problemas de otimização de múltiplos objetivos, conseguindo resolver problemas com mais de cem variáveis de projeto;
- O PSO mostrou ser uma ferramenta indicada também para problemas práticos da engenharia, apesar de não ser mais eficiente computacionalmente do que algoritmos que fazem uso de informações de gradientes, o PSO demonstrou-se bastante eficaz;
- A utilização de técnicas para melhoria de eficiência computacional foi de grande utilidade para o PSO, principalmente a utilização de modelos substitutos.

### **6.3 Sugestões para trabalhos futuros**

Este trabalho constitui estudos preliminares da metodologia PSO. A partir dos resultados obtidos apresentados nesta dissertação, alguns estudos com o intuito de melhorar o desempenho do PSO, tomando como ponto de partida o presente trabalho, são citadas as seguintes sugestões para trabalhos futuros:

- Desenvolver e implementar um modelo de otimização híbrido utilizando o PSO para explorar globalmente o espaço de projeto e identificar áreas promissoras, e a partir desta informação utilizar um algoritmo que use a informação de gradiente, para realizar uma busca local mais eficazmente;
- Aplicação do método em mais problemas da engenharia considerando estruturas submetidas a diferentes tipos de solicitações, tal como carregamento dinâmico;
- Aplicação do método em problemas estruturais 3D;
- Incluir procedimentos de otimização topológica (BENDSOE, 1995; CHENG, 1995; OBERNDORFER et al., 1996);
- Aplicação do método para otimização de problemas complexos da engenharia, que apresentam funções descontínuas e domínios discretos, tais como problemas de otimização da produção de campos de petróleo;

- Investigação de outras estratégias para lidar com problemas restritos;
- Investigação de técnicas para melhoria da convergência do algoritmo.

# Bibliografia

ABRAHAM, A., JAIN, L. and GOLDBERG, R. (2005) (editors), *Evolutionary Multiobjective Optimization – Theoretical Advances and Applications*, Springer.

AFONSO, S. M. B. (1995), *Shape Optimization of Shells Under Static and Free Vibration Conditions*, PhD thesis, Department of Civil Engineering, University of Wales, Swansea.

AFONSO, S. M. B. (1997), *Structural Shape Optimization Considering Multiobjective Functions*, in: Proceedings of III SIMMEC – III Simpósio Mineiro de Mecânica Computacional, pp. 251-260, Ouro Preto, MG, Novembro.

AFONSO, S. M. B. e HOROWITZ, B. (1998), *Utilização do MATLAB no Desenvolvimento de Ferramentas Educacionais no Ensino da Otimização Estrutural*, in: V Congresso de Engenharia Mecânica do Norte-Nordeste. Fortaleza: UFC. – vol. II, pp. 243-430.

AFONSO, S. M. B. and PATERA, A. T. (2003), *Structural Optimization in the Framework of Reduced Basis Method*, in: Proceedings of CILAMCE 2003 – XXIV Iberian Latin American Congress on Computational Methods in Engineering, Ouro Preto, Brazil.

AFONSO, S. M. B., LYRA, P. R. M. e ALBUQUERQUE, T. M. (2006), *Thermal-Structural Analysis and Optimization Using Reduced-Basis Approximations*, XXVII CILAMCE – 207<sup>th</sup> Iberian Latin-American Congress on Computational Methods in Engineering, Belém – Pará, Brazil, September.

AFONSO, S. M. B. e SIENZ, J. (1999), *Investigation of Different Strategies to Solve Multicriteria Structural Shape Optimization Problems*, 1<sup>st</sup> ASMO UK/ISSMO Conference, Ikley, U.K., July.

ALBUQUERQUE, T. M. M. (2005), *Análise e Otimização de Problemas Térmicos e Estruturais Bidimensionais Através do Método da Base Reduzida*, Dissertação de Mestrado, UFPE, Recife.

BEALE, E. M. L. (1988), *Introduction to optimization*, John Wiley & Sons, London, Great Britain.

BENDSOE, M. P. (1995), *Optimization for structural topology, shape and material*, Springer-Verlag, New York.

BIENERT, P. (1967), *Aufbau einer optimierungsautomatik für drei parameter*, master's thesis, Universidad Técnica de Berlin.

CASTRO, R. E. (2001), *Otimização de estruturas com Multi-objetivo via Algoritmo Genético*, Tese de Doutorado, COPPE/UFRJ, Rio de Janeiro.

CHENG, G. (1995), *Some aspects of truss topology optimization*, Structural optimization 10, 173-179, Springer-Verlag.

COLLETTE, Y. and SIARRY, P. (2003), *Multiobjective Optimization – Principles and Case Studies*, Springer-Verlag, 1<sup>st</sup> ed.

COELLO, C. A. C. and PULIDO, G. T. (2001), *A Micro-Genetic Algorithm for Multiobjective Optimization*, in Zitzler, E., Deb, K., Thiele, L., Coello Coello, C. A. and Corne, D. (eds.), First International Conference on Evolutionary Multi-Criterion Optimization, pp. 126-140, Springer-Verlag, Lecture Notes in Computer Science n° 1993.

COELLO, C. A. C. and PULIDO, G. T. (2001), *Multiobjective Optimization Using a Micro-Genetic Algorithm*, in Spector, L., Goodman, E. D., Wu, A., Langdon, W. B., Voigt, H-M., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M. H. and Burke, E. (eds.), Proceedings of the Genetic and Evolutionary Computation Conference

(GECCO'2001), pp. 274-282, San Francisco, California, 2001, Morgan Kaufmann Publishers.

COELLO, C. A. C. and SIERRA, M. R. (2004), *A Study of the Parallelization of a Co-Evolutionary Multi-Objective Evolutionary Algorithm*, Lecture Notes Artif Intell, Springer Verlag, pp 688-697.

COELLO, C. A. C., PULIDO, G. T. and LECHUGA, M. S. (2004), *Handling Multiple Objective with Particle Swarm Optimization*, IEEE Transactions on Evolutionary Computation, vol. 8, n° 3, June.

COOK, R. D. (1981), *Concepts and Applications of Finite Elements Analysis*, John Wiley & Sons, Singapore.

CORNE, D. W., KNOWLES, J. D. and OATES, M. J. (2000), *The Pareto Envelope-based Selection Algorithm for Multiobjective Optimization*, in Schoenauer, M., Deb, K., Rudolph, G., Yao, X. Lutton, E., Merelo, J. J. and Schwefel, H-P. (eds.), Proceedings of the Parallel Problem Solving from Nature VI Conference, pp. 839-848, Paris, France, 2000, Springer, Lecture Notes in Computer Science n° 1917.

CRAMER, N. L. (1985), *A Representation for the Adaptive Generation of Simple*, in J.J. Grefenstette, editor, Proceedings of an International Conference on Genetic, pages 183–187, Pittsburgh, PA., Carnegie-Mellon University.

DARWIN, C. (1859), *On the Origin of Species by Means of Nature Selection, or the Preservation of Favoured Races in the Struggle for Life*.

DAS, I. e DENNIS, J. E. (1997), *A Closer Look at Drawbacks of Minimizing Weighted Sums of Objectives for Pareto Set Generation in Multicriteria Optimization Problems*, Structural Optimization, vol. 14, pp. 63-69.

DEB, K., AGRAWAL, S., PRATAB, A. and MEYARIVAN, T. (2000), *A Fast Elitism Non-dominated Sorting Algorithm for Multi-objective Optimization: NSGA-II*, KanGAL report 200001, Indian Institute of Technology, Kanpur, India.

DEB, K., AGRAWAL, S., PRATAB, A. and MEYARIVAN, T. (2000), *A Fast Elitism Non-dominated Sorting Algorithm for Multi-objective Optimization: NSGA-II*, in Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J.J. and Schwefel, H-P. (eds.), *Proceedings of the Parallel Problem Solving from Nature VI Conference*, pp. 849-858, Paris, France, 2000, Springer, Lecture Notes in Computer Science n° 1917.

DEB, K., PRATAB, A., AGRAWAL, S. and MEYARIVAN, T. (2002), *A Fast Elitism Multiobjective Genetic Algorithm: NSGA-II*, *IEEE Transactions on Evolutionary Computation*, 6(2): 182-197, April 2002.

ENGELBRECHT, A. P. (2002) (editor), *Computational Intelligence: An Introduction*, John Wiley & Sons, England.

ERICKSON, M., MAYER, A. and HORN, J. (2001), *The Niched Pareto Genetic Algorithm 2 Applied to the Design of Groundwater Remediation Systems*, in Zitzler, E., Deb, K., Thiele, L., Coello Coello, C. A. and Corne, D. (eds.), *First International Conference on Evolutionary Multi-Criterion Optimization*, pp. 681-695, Springer-Verlag, Lecture Notes in Computer Science n° 1993.

FOGEL, L. J. (1964), *On the Organization of Intellect*, PhD thesis, University of California, Los Angeles, California.

FONSECA, C. M. and FLEMING, P. J. (1993), *Genetic Algorithm for Multiobjective Optimization: Formulation, Discussion and Generalization*, in Forrest, S. (ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms*, pp. 416-423, San Mateo, CA, Morgan Kaufmann Publishers.

GOLDBERG, D. E., (1989), *Genetic Algorithm in Search, Optimization and Machine Learning*, Addison-Wesley Publishing Company, Reading, MA.

HAFTKA, R. T. and GÜRDAL, Z. (1993), *Elements of Structural Optimization – Third revised and expanded edition*, Solid Mechanics and its Applications, vol. 1, Kluwer Academic Publishers, Netherlands.

HERNÁNDEZ, S. (1994), *Geometry and Optimization Techniques for Structural Design*, in: Kodiyalan S. and Saucena M. eds., pp. 341-361.

HOLLAND, J. H. (1992), *Adaptation in Natural and Artificial Systems*, MIT Press, Cambridge, Massachusetts, second edition, 1992.

HORN, J., NAFPLIOTIS, N. and GOLDBERG, D. E. (1994), *A Niche Pareto Genetic Algorithm for Multiobjective Optimization*, in Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, vol. 1, pp. 82-87, Piscataway, NJ, June 1994, IEEE Service Center.

INNOCENTE, M. S. (2006), *Population-Based Methods: Particle Swarm Optimization – Development of a General-Purpose Optimizer and Applications*, master's thesis, Universitat Politècnica de Catalunya, Barcelona.

ISMAIL-YAHAYA, A. e MESSAC, A. (2002), *Effective generation of the Pareto frontier using the normal constraint method*, 40<sup>th</sup> Aerospace Sciences Meeting and Exhibit, held in Reno, Nevada, paper n°. AIAA 2000-0178.

JIN, Y., OLHOFFER, M. and SENDHOFF, B. (2001), *Dynamic Weighted Aggregation for Evolutionary Multi-Objective Optimization: Why Does It Work and How?*, in: Proceedings of the Genetic and Evolutionary Computation Conference – GECCO 2001, July 7, San Francisco, California, pp. 1042-1049.

JIN, Y., OKABE, T. e SENDHOFF, B. (2001), *Adapting Weighted Aggregation for Multi-Objective Evolution Strategies*, in: DEB, K., THIELE, L., e ZITZLER E., eds., First International Conference on Evolutionary Multi-criterion Optimization, Lecture Notes in Computer Science, pages 96-110, Zurich, Switzerland, Springer.



KENNEDY, J. and EBERHART, R. C. (1995), *Particle Swarm Optimization*, Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia, pp. 1942-1948.

KIRSCH, U. (1993), *Structural Optimization: Fundamentals and Applications*, Springer-Verlag, Berlin.

KOSKI, J. (1985), *Defectiveness of Weighting Method in Multicriterion Optimization of Structures*, Communications in Applied Numerical Methods, vol. 1, pp. 333-337.

KOZA, J. R. (1989), *Herarchical Genetic Algorithms Operating on Populations of Computer Programs*, in N. S. Sridharan, editor, Proceedings of 11th International Joint Conference on Artificial Intelligence, pages 768–774, San Mateo, California.

KOZA, J. R. (2003), *Genetic Algorithms and Genetic Programming*, 2003 lectures, Stanford University, Stanford, California.

KOZA, J. R. (1992), *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, Massachusetts.

KNOWLES, J. D. and CORNE, D. W. (2000), *Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy*, Evolutionary Computation, 8(2):149-172.

KUHN, H. W. and TUCKER, A. W. (1951), *Nonlinear Programming*, in Neyman, J. (ed), Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, pp. 481-492, Berkeley, CA, University of California Press.

LITTLEFIELD, B. e HANSELMAN, D. (1999), *MATLAB 5: Versão do Estudante – Guia do Usuário*, Makron Books.

MACEDO, C. M. H. (2002), *Otimização de Treliças Planas sob Várias Solicitações com Ênfase em Problemas Multiobjetivos*, Dissertação de Mestrado, Dept. de Engenharia Civil, Universidade de Pernambuco, Recife-PE, Brasil.

MESSAC, A., ISMAIL-YAHAYA, A. e MATTSON, C. A. (2003), *The Normalized Normal Constraint Method for Generating the Pareto Frontier*, Structural Optimization, vol. 25, nº 2, pp. 86-98.

MICHALEWICZ, Z. e DASGUPTA, D. (1997), (eds.) *Evolutionary Algorithms in Engineering Applications*, Springer Verlag.

MOSTAGHIM, S. e TEICH, J. (2004), *Covering Pareto-optimal Fronts by Subswarms in Multi-objective Particle Swarm Optimization*, in IEEE.

NOCEDAL, J. and WRIGHT, S. J. (2000), *Numerical Optimization*, Springer-Verlag, Rensselaer, NY.

OBERNDORFER, J. M., ACHTZIGER, W. e HÖRNLEIN, H. R. E. M. (1996), *Two approaches for truss topology optimization: a comparison for practical use*, Structural optimization 11, 137-144, Springer-Verlag.

OMKAR, S. N., MUDIGERE, D., NAIK, G. N. e GOPALAKRISHNAN, S. (2007), *Vector Evaluated Particle Swarm Optimization (VEPSO) for Multi-Objective Design of Composite Structures*, Available online at [www.sciencedirect.com](http://www.sciencedirect.com), Department of Aerospace Engineering, Indian Institute of Science, Bangalore 560 012, India.

PACHECO, P. S. (1998), *A User's Guide to MPI*, Department of Mathematics, University of San Francisco, San Francisco, Ca 94117, March.

PAGER, W. e SHIELD, R. T. (1968), *Optimal Design of Multi-purpose Structures*, Journal of Solid Structures, vol. 4, pp. 469-475.

PAGER, W. e TAYLOR, J. E. (1968), *Problems of Optimal Structural Design*, Journal of Applied Mechanics, vol. 35, pp. 102-106.

PARENTE JR., E. (1995), *Otimização de Estruturas Sujeitas a Instabilidade Global: Aplicação a Treliças Planas*, Dissertação de Mestrado, PUC-RJ, Rio de Janeiro-RJ, Brasil, Abril.

PARSOPOULOS, K. E. e VRAHATIS, M. N. (2002), *Particle Swarm Optimization Method in Multiobjective Problems*.

POMEROY, P. (2003), *An Introduction to Particle Swarm Optimization*, in:  
<http://www.adaptativeview.com/articles/ipsoprnt.html> (Dezembro, 2005).

POWELL, M. J. D. (1978), *Algorithms for Nonlinear Constraints that Use Lagrangian Functions*, Math. Prog., 14, pp. 224-248.

PRUD'HOMME, C., ROVAS, D. V., VEROY, K., MACHIELS, L., MADAY, Y., PATERA, A. T. and TURICINI, G. (2002), *Reliable Real-time Solution of Parametrized Partial Differential Equations: Reduced-basis Output Bound Method*, Journal of Fluids Eng., vol. 124, pp. 70-79.

PULIDO, G. T. (2005), *Uso de Auto-Adaptación y Elitismo para Optimización Multiobjetivo Mediante Cúmulos de Partículas*, tesis de doctorado, Centro de Investigación y Estudios Avanzados del Instituto Politécnico Nacional, Unidad Zacatenco, Departamento de Ingeniería Eléctrica, Sección de Computación, México D.F., México.

RECHENBERG, I. (1973), *Evolutionstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*, Frommann-Holzboog, Stuttgart.

SCHAFFER, J. D. and GREFENSTETTE, J. J. (1985), *Multiobjective Learning Via Genetic Algorithms*, in Proceedings of the 9<sup>th</sup> International Joint Conference on Artificial Intelligence (IJCAI - 85), pp. 593-595, Los Angeles, CA, AAAI.

SCHAFFER, J. D. (1984), *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*, PhD thesis, Vanderbilt University, Nashville, TN.

SCHAFFER, J. D. (1985), *Multiple Objective Optimization with Vector Evaluated Genetic Algorithm*, in Genetic Algorithm and their Applications: Proceedings of the First International Conference on Genetic Algorithms, pp. 93-100, Hillsdale, NJ, Lawrence Erlbaum.

SCHWEFEL, H-P. (1965), *Kybernetische evolution als strategie der experimentellen forschung inder strömungstechnik*, Dipl.-Ing. thesis, German.

SHI, Y. and EBERHART, R. C. (1998), *A Modified Particle Swarm Optimizer*, Proceedings of the 1998 IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, May 4-9.

SHI, Y. H. and EBERHART, R. C. (1998), *Parameter Selection in Particle Swarm Optimization*, Evolutionary Programming VII, Lecture Notes in Computer Science, pp. 591-600.

SIERRA, M. M. R. (2006), *Uso de Coevolución y Herancia para Optimización Multi-Objetivo Mediante Cúmulos de Partículas*, tesis de doctorado, Centro de Investigación y Estudios Avanzados del Instituto Politécnico Nacional, Unidad Zacatenco, Departamento de Ingeniería Eléctrica, Sección de Computación, México D.F., México.

SRINIVAS, N. and DEB, K. (1994), *Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms*, Evolutionary Computation, 2(3): 221-248, Fall.

TORRES, J. S. (2001), *Otimização de Pórticos de Concreto Armado Utilizando o Sistema Computacional ANSYS*, Dissertação de Mestrado, Dept. de Engenharia Civil, Universidade de Pernambuco, Recife-PE, Brasil, Novembro.

VANDERPLAATS, G. N. (1984), *Numerical Optimization Techniques for Engineering Design: with Applications*, McGraw-Hill, New York.

VANDERPLAATS, G. N. (1993), *Thrity Years of Modern Structural Optimization*, Advances in Engineering of Software 16, pp. 81-88.

VENKAYYA, V. B., KHOT, N. S. e REDDY, V. S. (1968), Optimization of Structures Based on the Study of Strain Energy Distribution, AFFDL-TR-68-150.

VENTER, G. and SOBIESZCZANSKI-SOBIESKI, J. (2002), *Particle Swarm Optimization*, in: 43<sup>rd</sup> AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, April 22-25, Denver, Colorado.

ZIENKIEWICZ, O. C. e TAYLOR, R. L. (2000), *The Finite Element Method*, New York, McGraw-Hill.

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

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