LEANDRO ZAVAREZ BARBOSA

**TÉCNICAS DE OTIMIZAÇÃO BASEADAS NO
PARADIGMA DE ENXAMES DE PARTÍCULAS E
SUA APLICAÇÃO AO PROJETO DE
EQUIPAMENTOS ELETROMAGNÉTICOS**

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Engenharia Elétrica.

São Paulo
2012

LEANDRO ZAVAREZ BARBOSA

**TÉCNICAS DE OTIMIZAÇÃO BASEADAS NO
PARADIGMA DE ENXAMES DE PARTÍCULAS E
SUA APLICAÇÃO AO PROJETO DE
EQUIPAMENTOS ELETROMAGNÉTICOS**

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Engenharia Elétrica.

Área de Concentração:
Sistemas de Potência

Orientador:
Luiz Lebensztajn

São Paulo
2012

Este exemplar foi revisado e alterado em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, de dezembro de 2012.

Assinatura do autor _____

Assinatura do orientador _____

FICHA CATALOGRÁFICA

Barbosa, Leandro Zavarez

Técnicas de otimização baseadas no paradigma de enxames de partículas e sua aplicação ao projeto de equipamentos eletromagnéticos / L.Z. Barbosa. – ed.rev. -- São Paulo, 2012. 107 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Energia e Automação Elétricas.

1.Eletromagnetismo (Otimização) 2.Pareto otimalidade 3.Máquinas elétricas I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Energia e Automação Elétricas II. t.

Aos meus pais: Cleyton e Virgínia
Ao meu irmão: Henrique
À minha noiva: Claudete
Aos meus familiares

AGRADECIMENTOS

Ao Prof. Dr. Luiz Lebensztajn, agradeço por enriquecer nossas conversas com sabedoria e descontração, contribuindo em meu desenvolvimento profissional e pessoal. Agradeço pela paciência, atenção e disponibilidade durante a elaboração deste trabalho. Acima de tudo, agradeço por me aceitar como seu aluno.

Ao Prof. Dr. Leandro dos Santos Coelho pela colaboração no desenvolvimento de alguns algoritmos utilizados.

A todos professores que colaboraram na formação dos conhecimentos requeridos para a conclusão desta dissertação.

Aos meus familiares pelo amor e carinho dedicados em todos os momentos da minha vida.

À minha noiva, pelo amor e dedicação. Especialmente, pela compreensão de privar-se de minha companhia.

Agradeço a Deus por sua companhia silenciosa e sempre presente.

RESUMO

O presente trabalho propõe a utilização do método de otimização baseado no paradigma de enxame de partículas no projeto de um dispositivo eletromagnético, modelado analiticamente. A otimização baseada em enxames de partículas pertence à classe dos algoritmos evolutivos e é baseada no algoritmo de simulação do movimento de pássaros na busca por comida. O trabalho será focado na resolução de problemas de otimização multiobjetivo e apenas alguns casos de otimização mono-objetivo serão resolvidos para demonstrar a funcionalidade do método de otimização. Dois métodos de otimização multiobjetivo são propostos: um é baseado num algoritmo de otimização multiobjetivo que utiliza o paradigma de enxames em conjunto com soluções adotadas pelo algoritmo genético multiobjetivo denominado Nondominated Sorting Genetic Algorithm-II (NSGA-II) e o outro é baseado também no paradigma de enxames utilizando elementos do algoritmo de otimização multiobjetivo intitulado Strength Pareto Evolutionary Algorithm (SPEA). Ambos algoritmos são validados em um problema de otimização baseado no projeto de um motor de corrente contínua sem escovas, um benchmark de otimização.

ABSTRACT

This work proposes the use of the optimization method based on the particle swarm paradigm in the design of electromagnetic device, analytically modeled. Particle swarm optimization belongs to the class of evolutionary algorithms and is based on the movement simulation of birds searching for food. This work will be focused on solving multi-criteria optimization problems and some cases of single-objective optimization problems will be solved only to demonstrate the functionality of optimization method. Two multi-criteria optimization methods are proposed: one based on an optimization algorithm that uses the multiobjective particle swarm paradigm and some concepts extracted from the multiobjective genetic algorithm called Nondominated Sorting Genetic Algorithm-II (NSGA-II) and the other is based on the particle swarm paradigm by using some elements of another multiobjective optimization algorithm entitled Strength Pareto Evolutionary Algorithm (SPEA). Both methods are applied to an optimization problem related to the design of a brushless direct current motor.

SUMÁRIO

Lista de Ilustrações

Lista de Tabelas

Lista de Abreviaturas e Siglas

1	Introdução	14
1.1	Otimização mono-objetivo	16
1.2	Otimização multiobjetivo	17
1.2.1	Dominância de Pareto	18
1.2.2	Resolução de problemas de otimização multiobjetivo . .	21
1.3	Objetivos	26
1.4	Organização	26
2	Estado da arte	28
2.1	Algoritmos Evolutivos	28
2.1.1	Algoritmo Genético	29
2.1.1.1	Nondominated Sorting Genetic Algorithm	30
2.1.1.2	Strength Pareto Evolutionary Algorithm	34
2.1.2	Particle Swarm Optimization	38
3	Particle Swarm Optimization	39

3.1	PSO mono-objetivo	41
3.1.1	Parâmetros do PSO	44
3.2	PSO multiobjetivo	49
3.2.1	Enhanced Multiobjective Particle Swarm Optimization . .	51
3.2.1.1	O algoritmo do Enhanced Multiobjective Particle Swarm Optimization	56
3.2.2	Multiobjective Strength Pareto Particle Swarm Optimization	59
3.2.2.1	Incluindo restrições no problema de otimização	63
4	Resultados e Discussões	67
4.1	Problemas analisados	67
4.1.1	Funções de teste mono-objetivo	67
4.1.2	Funções de teste multiobjetivo	69
4.1.3	Problema de otimização real	70
4.1.3.1	Otimização mono-objetivo	74
4.1.3.2	Otimização multiobjetivo	74
4.2	Resultados da otimização mono-objetivo	75
4.2.1	Ajustes de parâmetros do PSO	75
4.2.2	Otimização mono-objetivo do motor	80
4.3	Resultados da otimização multiobjetivo	83
4.3.1	Algoritmo EMOPSO	83
4.3.2	Algoritmo MSP2SO	86

4.3.2.1	Funções de teste	86
4.3.2.2	Otimização do projeto do motor	89
5	Considerações Finais	95
	Referências	97
	Apêndice A – Projeto do Motor de Corrente Contínua sem escovas	101

LISTA DE ILUSTRAÇÕES

1	Dominância de Pareto para MOP	19
2	Fronteira de Pareto para um problema de otimização cujo objetivo é minimizar as funções f_1 e f_2	20
3	Fronteira de Pareto com soluções mal distribuídas. A linha contínua representa a fronteira de Pareto verdadeira para o problema.	24
4	Fronteira de Pareto com excesso de soluções. A linha contínua representa a fronteira de Pareto verdadeira para o problema. . .	25
5	Cálculo da <i>crowding distance</i>	33
6	Algoritmo simplificado do SPEA.	35
7	Funcionamento da técnica de cluster no SPEA. Cada cluster é representado por uma cor diferente.	37
8	Topologia totalmente conectada. As 8 partículas estão totalmente conectadas conforme indicado pelas linhas em preto. . .	47
9	Topologia em anel, ou melhor local com $k = 2$. As linhas em preto indicam como as partículas estão conectadas.	47
10	Topologia melhor local com $k = 4$. As linhas em preto indicam como as partículas estão conectadas.	48
11	Topologia Roda. As linhas em preto indicam como as partículas estão conectadas.	48

12	Soluções A e B para um problema de otimização a duas funções objetivo.	50
13	Subdivisão do espaço das funções em hipercubos para distribuição das soluções.	52
14	Função densidade de probabilidade para as distribuições normal e de Cauchy.	58
15	Gráfico da função que gera números aleatórios com distribuição de Cauchy a partir de números aleatórios uniformemente distribuídos.	58
16	Densidade de probabilidade para a distribuição beta com $\alpha = \beta = 1/2$	61
17	Redução do número máximo de soluções no arquivo livre por patamares.	65
18	Algoritmo simplificado do funcionamento da atualização do conjunto de Pareto.	66
19	Função de Rastrigin para $n = 2$	68
20	Função Rosenbrock para $n = 2$	69
21	Geometria do motor CC sem escovas a ímãs permanentes com rotor externo	71
22	Fluxograma de cálculo do motor.	73
23	Taxa de acerto média para cada valor de w_{max} considerando $c = 1, 4$	76
24	Iterações até atingir o valor ótimo para cada valor de w_{max} considerando $c = 1, 4$	76

25	Taxa de acerto média para cada valor de c considerando $w_{max} = 1, 1$	77
26	Iterações até atingir o valor ótimo para cada valor de c considerando $w_{max} = 1, 1$	77
27	Taxa de acerto média para cada valor de w_{max} considerando $c = 1, 1$	78
28	Taxa de acerto média para cada valor de w_{max} considerando $c = 1, 2$	78
29	Iterações até atingir o valor ótimo para cada valor de w_{max} considerando $c = 1, 1$	79
30	Iterações até atingir o valor ótimo para cada valor de w_{max} considerando $c = 1, 2$	79
31	Rendimento do motor obtido pelo PSO em cada execução do algoritmo. A média ficou em 95,25 %, com máximo de 95,35 % e mínimo em 95,0 %.	81
32	Iterações até convergência obtido pelo PSO em cada execução do algoritmo. A média ficou em 96, com máximo de 171 e mínimo em 57.	82
33	Histograma das soluções.	82
34	Fronteira de Pareto utilizando RNMOPSO e EMOPSO	84
35	Histograma de algumas variáveis de otimização das soluções não dominadas encontradas.	84
36	Histograma das demais variáveis de otimização das soluções não dominadas encontradas.	85

37	Fronteira de Pareto para T1 com antigos valores para w , c_1 e c_2 .	86
38	Fronteira de Pareto final para T1.	87
39	Fronteira de Pareto final para T2.	88
40	Fronteira de Pareto final para T3.	88
41	Fronteira de Pareto final para o motor CC utilizando função penalidade para tratamento de restrições.	90
42	Fronteira de Pareto final para o motor CC utilizando dois arquivos, livre e válido, para tratamento de restrições.	91
43	Fronteira de Pareto final obtida pelo SQP.	91
44	Comparação entre as fronteira de Pareto para otimização do motor utilizando SQP e o MSP2SO.	92
45	Detalhe das fronteiras de Pareto na região de maior eficiência. .	93
46	Histogramas das soluções.	94

LISTA DE TABELAS

1	Variáveis de construção do motor CC sem escovas proposto por Brisset e Brochet.	72
2	Restrições do problema de otimização mono-objetivo.	72
3	Valores encontrados pelo PSO para as variáveis em 50 execu- ções do algoritmo de otimização	81
4	Tabela de valores para a métrica C	93

LISTA DE ABREVIATURAS E SIGLAS

NSGA-II	Nondominated Sorting Genetic Algorithm-II
SPEA	Strength Pareto Evolutionary Algorithm
TSP	Traveling Salesman Problem
MEF	Método dos Elementos Finitos
MOP	Multiobjective Optimization Problem
NSGA	Nondominated Sorting Genetic Algorithm
SPEA-2	Strength Pareto Evolutionary Algorithm-2
PSO	Particle Swarm Optimization
MOPSO	Multiobjective Particle Swarm Optimization
PAES	Pareto Archived Evolution Strategy
Micro-GA	Micro-Genetic Algorithm
RNMOPSO	Raquel and Naval Multiobjective Particle Swarm Optimization
EMOPSO	Enhanced Multiobjective Particle Swarm Optimization
MSP2SO	Multiobjective Strength Pareto Particle Swarm Optimization
CC	Corrente Contínua
SQP	Sequential Quadratic Problem

1 INTRODUÇÃO

Segundo o dicionário Aurélio otimizar é: “Dar a algo (uma máquina, uma empresa, uma ação, etc.) o rendimento ótimo, criando-lhe as condições mais favoráveis ou tirando (dele ou dela) o melhor partido possível; tornar (algo) ótimo ou ideal”. Ou ainda, em uma forma mais coloquial proposta pelo professor Jocelyn Freitas Bennaton¹: “Otimizar é melhorar até onde pudermos. No ideal, melhorar até o máximo. Até aquela situação ideal na qual, como vulgarmente se diz, ‘se mexer mais, piora”. Otimizar é, portanto, encontrar uma combinação de grandezas, denominadas *variáveis*, para a qual obtemos o melhor de “algo”.

O processo de otimização inicia-se com a definição do problema a ser resolvido, que pode ter origem em qualquer área do conhecimento: engenharia, matemática, física, química, biologia, etc. A otimização está presente, inclusive, em nossas atividades diárias mais comuns: sempre procuramos alcançar nosso “ótimo” no trabalho, nos estudos, na vida.

Em engenharia a otimização é utilizada para maximizar a eficiência de sistemas, minimizar massa, encontrar o melhor caminho, entre tantos outros objetivos. Ela acaba sendo aplicada a diversos problemas reais como projeto de motores elétricos, projeto de instalações prediais, projetos de asas de aviões, disposição de componentes em placas eletrônicas, etc. Um problema

¹O que é otimizar - http://www.lps.usp.br/neo/jocelyn/que_e_otimizar.htm (acessado em 01/03/2012)

clássico de otimização é o problema do caixeiro-viajante (TSP), pela abreviação do nome em inglês *Traveling Salesman Problem* (COOK, 2012). Neste problema o objetivo é minimizar a rota para o caixeiro-viajante percorrer todas as cidades vendendo seus produtos e retornar ao ponto de partida, sendo que uma cidade não pode ser visitada mais de uma vez.

O problema a ser resolvido pode ainda ser de origem teórica. Problemas teóricos são problemas puramente matemáticos criados para teste e comparação de algoritmos de otimização e alguns exemplos destes problemas serão descritos no capítulo 4. Os problemas reais podem ser descritos através de um modelo analítico, através de equações matemáticas, ou também serem modelados utilizando métodos numéricos, como o método dos elementos finitos (MEF). A precisão da resposta do processo de otimização dependerá da modelagem adotada, das considerações e aproximações assumidas, da metodologia de otimização utilizada, etc.

A modelagem pelo método dos elementos finitos é mais precisa, contudo o tempo de cálculo da solução é maior que o requerido por modelos analíticos. A otimização em conjunto com MEF possui um alto custo computacional, pois a modificação das variáveis requer o reprocessamento do modelo. O custo computacional pode ser reduzido através da seguinte abordagem: o problema de otimização é resolvido com MEF para um conjunto de possíveis soluções e, em seguida, um método de aproximação de funções é utilizado para construir um modelo aproximado. O modelo aproximado ou metamodelo será utilizado pelo algoritmo de otimização para o cálculo das funções objetivo e restrições. Contudo a solução encontrada pelo algoritmo será aproximada e pode ser refinada utilizando novamente o MEF.

Um problema que busca encontrar o ponto ideal, ou ponto ótimo, de “algo” é um problema de otimização e encontrar o ponto ideal consiste em maximi-

zar ou minimizar este “algo” através da modificação de algumas grandezas. Essas grandezas passíveis de modificação são as variáveis do problema e este “algo” é denominado *objetivo*. Em um problema de otimização pode-se ter vários objetivos, mas em um primeiro momento neste trabalho, o problema será entendido como alcançar uma única meta. Ao objetivo se atribui um valor numérico, facilitando a comparação entre soluções, e esse valor é uma *função* do valor das variáveis da solução analisada. Essa função é denominada de *função objetivo*. O ponto ótimo do problema de otimização é a combinação de valores das variáveis para a qual o valor da função objetivo atinge o valor máximo (maximizar) ou mínimo (minimizar).

O problema de otimização está sujeito a algumas condições ou *restrições* que definem a validade de uma solução proposta. As soluções que não satisfazem essas restrições, soluções não válidas, podem possuir informações úteis à busca do ponto ótimo do objetivo, portanto simplesmente descartar estas soluções pode não ser a melhor ideia (MICHALEWICZ, 1995). As formas mais comuns de lidar com as restrições são: uso de funções de penalidade ou através da conversão das restrições em objetivos do problema, entre outras. Existem alguns problemas de otimização, com poucas soluções válidas ou ponto ótimo isolado, nos quais se têm a impressão que o verdadeiro desafio é encontrar a região de pontos factíveis e uma vez encontrada esta região a tarefa de determinar o ponto ótimo parece mais simples.

1.1 Otimização mono-objetivo

A otimização é dita mono-objetivo quando se deseja alcançar uma única meta ou objetivo, ou seja, minimizar ou maximizar uma função objetivo única e escalar. Sem perda de generalidade, um problema de otimização mono-objetivo qualquer pode ser matematicamente representado pela eq.(1.1), sendo

$\vec{x} = [x_1, x_2, \dots, x_n]^T$ o vetor das variáveis de entrada do problema, f_o a função objetivo e $g_i(\vec{x})$, $h_i(\vec{x})$ as restrições de desigualdade e igualdade, respectivamente, às quais o problema está sujeito.

$$\begin{aligned} \text{minimizar} \quad & y = f_o(\vec{x}) \\ \text{Sujeito a} \quad & g_i(\vec{x}) \leq 0 \quad i = 1, 2, \dots, r \\ & h_i(\vec{x}) = 0 \quad i = 1, 2, \dots, q \end{aligned} \quad (1.1)$$

Otimizar o problema descrito acima, consiste em encontrar um vetor $\vec{x}^*$, tal que o valor para y seja o menor possível e satisfaça as restrições do problema, ou seja, $g_i(\vec{x}^*) \leq 0$ e $h_i(\vec{x}^*) = 0$. Contudo, reescrevendo cada uma das restrições como objetivos, o problema mono-objetivo pode ser resolvido como um problema de otimização a múltiplos objetivos.

1.2 Otimização multiobjetivo

A resolução de um problema de otimização com múltiplos objetivos implica minimizar ou maximizar diversas funções objetivo com algum tipo de conflito, ou seja, melhorar um dos requisitos implica a deterioração de outro. Por exemplo, na otimização de um motor elétrico, maximizar a eficiência e minimizar a massa geralmente são objetivos conflitantes, pois um motor de menor massa potencialmente não possuirá a maior eficiência possível.

Um problema de otimização multiobjetivo (MOP²), com m objetivos, pode

²do inglês, *Muliobjective Optimization Problem*

ser representado matematicamente por:

$$\begin{aligned}
 &\text{minimizar} \quad \vec{y} = F(\vec{x}) := [f_1(\vec{x}), f_2(\vec{x}), \dots, f_m(\vec{x})] \\
 &\text{sujeito a} \quad : \quad g_i(\vec{x}) \leq 0 \quad i = 1, 2, \dots, r \\
 &\quad \quad \quad h_i(\vec{x}) = 0 \quad i = 1, 2, \dots, q
 \end{aligned} \tag{1.2}$$

no qual $\vec{x} = [x_1, x_2, \dots, x_n]^T$ são as variáveis de entrada do problema, $g_i(\vec{x})$, $h_i(\vec{x})$ são as restrições do problema e $f_i(\vec{x})$, $i = 1, \dots, m$ são as funções objetivo. Para simplificar, consideremos apenas a resolução de um problema no qual todas as funções objetivo f_i devem ser minimizadas.

Antes de prosseguirmos com discussão sobre técnicas e algoritmos de resolução de problemas de otimização multiobjetivo, é preciso introduzir alguns conceitos importantes para comparação e classificação das soluções encontradas.

1.2.1 Dominância de Pareto

O conceito de dominância de Pareto (PARETO, 1986) tem sido empregado para comparar as várias soluções obtidas na resolução de MOP. As definições a seguir ajudam a explicar melhor o conceito de dominância de Pareto considerando a resolução de um MOP definido pela eq.(1.2). Nas definições será utilizada a seguinte notação: se p é uma solução para o MOP então $f_1^p, \dots, f_m^p$ e $x_1^p, \dots, x_n^p$ são, respectivamente, os valores das funções objetivo e variáveis desta solução.

Definição 1. Dominância de Pareto *Sejam duas possíveis soluções, p e g , sendo f_i^g o valor de cada função objetivo do MOP para a solução g . Dizemos que p domina g , se e somente se, $f_i^p \leq f_i^g$ para todo $i = 1, \dots, m$ e $p \neq g$.*

Denotamos por:

$$p < g \leftrightarrow f_i^p \leq f_i^g \quad \forall i \in [1, \dots, m] \quad \text{e} \quad p \neq g \quad (1.3)$$

A figura 1 ilustra o caso no qual $p < g$, para um MOP cujo objetivo é minimizar as funções objetivo f_1 e f_2 .

Figura 1: Dominância de Pareto para MOP

Definição 2. Pareto Ótimo Uma solução p é um Pareto ótimo, ou não dominada, em relação ao conjunto de soluções Ω , se e somente se, não existe $p' \in \Omega$ tal que $p' < p$. É importante observar que a não dominância é sempre referente a um conjunto de soluções.

Definição 3. Conjunto Pareto Ótimo Seja o conjunto $\mathcal{P} \subset \Omega$, dizemos que $\mathcal{P}$ é o conjunto Pareto ótimo definido por:

$$\mathcal{P} := \{p \in \Omega : \nexists p' \in \Omega : p' < p\} \quad (1.4)$$

As soluções pertencentes ao conjunto Pareto ótimo ($\mathcal{P}$) representam o conjunto das melhores soluções encontradas e são denominadas como soluções não dominadas do conjunto Ω .

Definição 4. Fronteira de Pareto O conjunto $\mathcal{PF}$ representa a fronteira de

Pareto e é definido por:

$$\mathcal{PF} := \{f_1^p, \dots, f_m^p : \forall p \in \mathcal{P}\} \quad (1.5)$$

Assim, *fronteira de Pareto* ($\mathcal{PF}$) é o conjunto dos valores das funções objetivo do problema para as soluções armazenadas no conjunto Pareto ótimo ($\mathcal{P}$). Na figura 2 é possível observar a fronteira de Pareto formada pelas soluções não dominadas, para um problema de otimização qualquer cujo objetivo é minimizar as funções objetivo f_1 e f_2 .

Figura 2: Fronteira de Pareto para um problema de otimização cujo objetivo é minimizar as funções f_1 e f_2 .

As soluções representadas na fronteira de Pareto são ditas incomparáveis entre si, pois uma solução sempre será melhor que as demais em pelo menos uma das funções objetivo. Com efeito, a resolução de um problema de otimização multiobjetivo pode não possuir uma única solução de resposta e apresentar um conjunto de melhores soluções possíveis, ao contrário do caso mono-objetivo.

Para evitar confusão na utilização dos conceitos de Pareto é importante estabelecer algumas notações. A cada iteração (t) um conjunto de soluções $\Omega(t)$ originará um conjunto Pareto ótimo, denominado $\mathcal{P}_{current}(t)$, e a respectiva fronteira de Pareto ($\mathcal{PF}_{current}(t)$). O conjunto Pareto ótimo e a fronteira de Pa-

reto de todas soluções não dominadas encontradas em todas as iterações são denominados: $\mathcal{P}_{know}(t)$ e $\mathcal{PF}_{know}(t)$, respectivamente, o índice t é utilizado para evidenciar que esses conjuntos podem mudar a cada iteração do algoritmo. Mais duas definições acerca dos conjuntos de Pareto são necessárias:

Definição 5. Pareto ótimo verdadeiro *Seja Ω_T o conjunto de todas as soluções possíveis para o MOP analisado, o conjunto Pareto ótimo verdadeiro $\mathcal{P}_{true}$ é definido por:*

$$\mathcal{P}_{true} := \{g \in \Omega_T : \nexists g' \in \Omega_T : g' < g\} \quad (1.6)$$

Definição 6. Fronteira de Pareto verdadeira *Conjunto de vetores das projeções das soluções do Pareto ótimo verdadeiro no espaço das funções.*

$$\mathcal{PF}_{true} := \{f_1^g, \dots, f_m^g : g \in \mathcal{P}_{true}\} \quad (1.7)$$

O conjunto $\mathcal{P}_{true}$ engloba todas as soluções não-dominadas existentes para o problema de otimização, considerando o universo de todas as soluções possíveis para o problema (Ω_T). Para a maioria dos problemas de otimização reais o conjunto das soluções Pareto ótimas é desconhecido, dificultando a comparação de resultados obtidos por diferentes algoritmos de otimização.

1.2.2 Resolução de problemas de otimização multiobjetivo

A utilização do conceito de Pareto na otimização permite a obtenção de um conjunto de soluções incomparáveis entre si. Contudo, um projetista (engenheiro, pesquisador, etc.) deseja uma solução única para o seu problema de otimização multiobjetivo. Ao contrário da otimização mono-objetivo, na otimização multiobjetivo não existe um consenso referente ao ponto ótimo do problema recaindo sobre o fator humano a responsabilidade de escolha da melhor solução. O responsável pela escolha de qual será a melhor solução

para o problema de otimização é chamado de tomador de decisão³. No entanto, não é o objetivo deste trabalho discutir sobre como será feita a escolha da melhor solução, existem diversos trabalhos sobre a metodologia de tomar decisões a múltiplos objetivos⁴.

O modo de interação entre o tomador de decisão e o algoritmo de otimização serve como base para classificação das metodologias de resolução dos problemas de otimização (VELDHUIZEN; LAMONT, 2000; HWANG; MASUD, 1979). Essa interação pode ocorrer em três etapas do processo de otimização: no início (*a priori*), durante (*progressivo*) ou no final (*a posteriori*) do processo.

A priori Os múltiplos objetivos são transformados em um único objetivo, para essa transformação é preciso priorizar os objetivos e definir quais são as restrições. O problema de otimização resultante será mono-objetivo.

Progressivo Os processos de otimização e tomada de decisão são entrelaçados. A priorização das funções objetivo é revista de acordo com a evolução da otimização

A posteriori A otimização é realizada considerando todas as funções objetivo e restrições do MOP. Um conjunto de soluções, usualmente o conjunto Pareto ótimo, é submetido ao tomador de decisão após a conclusão da otimização.

Na resolução *a priori* deseja-se transformar o problema de otimização multiobjetivo em um problema de otimização mono-objetivo a ser resolvido por um algoritmo de otimização mono-objetivo. Desta forma, as múltiplas funções objetivo são transformadas numa única função objetivo. Contudo, essa trans-

³em inglês *decision maker*

⁴Maiores informações pode ser obtidas no site da Sociedade Internacional de tomada de decisão a multi-critério <http://www.mcdmsociety.org/index.html> (acessado em 13/03/2012)

formação depende da importância de cada uma das funções objetivo, ou seja, o usuário precisa conhecer a importância de cada objetivo.

A transformação das múltiplas funções objetivo em uma única função pode ser realizada através da combinação linear ou não linear das funções objetivo originais do problema. Outra transformação possível é escolher apenas uma única função objetivo para ser otimizada e as demais serem transformadas em restrições do problema.

Na eq.(1.8), as m funções objetivo (f_i) de um problema multiobjetivo qualquer são combinadas linearmente para formar uma única função objetivo (f_o) escalar. Os termos α_i são denominados coeficiente de ponderação.

$$f_o(\vec{x}) = \alpha_1 f_1(\vec{x}) + \alpha_2 f_2(\vec{x}) + \dots + \alpha_m f_m(\vec{x}) \quad (1.8)$$

A solução resultante dependerá dos valores escolhidos para os coeficientes. Outra metodologia de resolução *a priori* é a ordenação lexicográfica das funções objetivo. Ou seja, as funções objetivo são ordenadas em ordem de importância e a solução ótima possuirá o melhor valor encontrado para a função mais importante, sendo as demais funções utilizadas como critério de desempate.

Na resolução *a posteriori* o algoritmo de otimização deve ser capaz de apresentar um conjunto com as melhores soluções, todas as funções objetivo são de igual importância. As melhores soluções usualmente são definidas utilizando o conceito de dominância de Pareto, assim as melhores soluções são todas as soluções não dominadas encontradas pelo algoritmo.

Em geral, os algoritmos que trabalham com o conceito de dominância de Pareto utilizam uma variável para armazenar as informações das soluções pertencentes à fronteira de Pareto, essa variável é denominada arquivo externo.

O arquivo externo contém as informações dos conjuntos $\mathcal{P}$ e $\mathcal{PF}$. A quantidade de soluções ótimas armazenadas no arquivo externo é limitada por dois fatores: custo computacional e desempenho do algoritmo.

O custo computacional diz respeito à quantidade de memória utilizada e ao tempo computacional gasto na comparação das soluções para determinar a dominância de Pareto. A comparação pode ser realizada de maneira a reduzir o tempo computacional, ao invés de realizar uma comparação entre todos os indivíduos da população, basta comparar aqueles para os quais ainda não se tem informação sobre serem dominados ou não dominados pois, se $p < g$ e $g < p'$ logo $p < p'$.

Figura 3: Fronteira de Pareto com soluções mal distribuídas. A linha contínua representa a fronteira de Pareto verdadeira para o problema.

As figuras 3 e 4 representam duas fronteiras de Pareto para um MOP, cujo objetivo é minimizar duas funções objetivo. A fronteira apresentada na figura 3 possui uma grande concentração de soluções numa certa região do espaço das funções objetivo, região A, enquanto que a região B praticamente não possui nenhuma solução. A representatividade da fronteira de Pareto não está boa, existem muitas soluções mapeando a mesma região do espaço. Já na fronteira representada na figura 4, o problema é o grande número de

soluções, a quantidade de soluções poderia ser reduzida sem perder a representatividade da fronteira de Pareto real.

Figura 4: Fronteira de Pareto com excesso de soluções. A linha contínua representa a fronteira de Pareto verdadeira para o problema.

Os algoritmos de otimização utilizam diversas técnicas para garantir a diversidade e limitar o número de soluções na fronteira de Pareto. Algumas técnicas utilizadas aplicam a teoria de *cluster* (ZITZLER; THIELE, 1999) para remover as soluções muito parecidas, outra técnica é a definição de um valor mínimo para a distância entre duas soluções (SRINIVAS; DEB, 1994). Alguns algoritmos ainda utilizam uma estimativa (DEB et al., 2000) da densidade de soluções na fronteira de Pareto mantendo as soluções em regiões com baixa densidade e eliminando algumas soluções das regiões com maior densidade de soluções.

1.3 Objetivos

O algoritmo de otimização baseado no paradigma de enxames de partículas tem sua origem na simulação do movimento de pássaros na busca por comida ou abrigo. Esse algoritmo evolutivo tem sido aplicado a resolução de vários problemas de otimização devido à fácil implementação e bons resultados apresentados. Embora tenha sido inicialmente proposto apenas para otimização mono-objetivo, atualmente existem várias versões para otimização multiobjetivo.

Existem diversos trabalhos com algoritmo baseado no paradigma de enxames aplicado a otimização multiobjetivo de funções matemáticas que podem não representar um problema físico. O objetivo deste trabalho é estudar a utilização de técnicas de otimização baseadas no paradigma de enxames de partículas no projeto de dispositivos eletromagnéticos, mais particularmente de um motor de corrente contínua sem escovas. Os dispositivos poderiam ser representados por modelos analíticos ou pelo método dos elementos finitos. Neste trabalho a análise será realizada por um procedimento analítico, sem qualquer perda de generalidade. O estudo será realizado com o desenvolvimento de duas propostas de variação do algoritmo para otimização multiobjetivo.

1.4 Organização

O texto está organizado de forma que no capítulo 2, fazemos uma breve análise dos algoritmos de otimização mais populares, com ênfase para aqueles de natureza evolucionária e a múltiplos objetivos. No capítulo 3, o algoritmo de otimização baseado no paradigma de enxame de partículas é detalhado em sua versão mono-objetivo. Também são expostas algumas propos-

tas de implementação para otimização multiobjetivo.

O capítulo 4 apresenta os problemas de otimização, tanto mono como multiobjetivo, a serem utilizados para testar os algoritmos propostos. Neste capítulo também são apresentados e discutidos os resultados obtidos nas simulações realizadas. O capítulo 5 apresenta as conclusões deste trabalho.

2 ESTADO DA ARTE

Há uma grande quantidade de algoritmos para a resolução de problemas de otimização e eles podem ser divididos em duas categorias: determinísticos e estocásticos.

Os algoritmos determinísticos, em geral, utilizam derivadas das funções objetivo e exigem muitas vezes conhecimento prévio, como existência de derivadas, continuidade, etc., da função objetivo para garantir convergência para o ótimo global. Por vezes, essas características limitam a aplicação na otimização de problemas reais, pois é comum desconhecer forma e continuidade das funções objetivo.

Apesar do grande número de chamadas da função objetivo, os algoritmos estocásticos são aplicados com grande frequência na resolução de problemas reais, pois uma solução razoável pode vir a ser encontrada em um número razoável de iterações. Isso ocorre devido ao número de soluções potenciais geradas e analisadas a cada iteração.

2.1 Algoritmos Evolutivos

Os algoritmos evolutivos são uma classe dos algoritmos estocásticos e geralmente são derivados de simulações ou observações de fenômenos naturais tais como: evolução de espécies, movimento de pássaros e insetos, resfria-

mento de metais, entre outros. O conjunto de soluções gerado a cada iteração é chamado de população.

São exemplos de algoritmos evolutivos: algoritmo genético (GOLDBERG, 1989), *ant colony optimization* (DORIGO; MANIEZZO; COLORNI, 1996), *simulated annealing* (KIRKPATRICK; GELATT; VECCHI, 1983), *particle swarm optimization* (KENNEDY; EBERHART, 1995), etc. O *simulated annealing* é um algoritmo inspirado num processo metalúrgico cujo objetivo é reduzir os defeitos na estrutura cristalina do metal. O algoritmo genético e o *particle swarm optimization* serão detalhados a seguir.

2.1.1 Algoritmo Genético

O algoritmo genético é um dos mais populares algoritmo de otimização evolutivo. É baseado na teoria da evolução proposta por Darwin, na qual os indivíduos mais aptos possuem maior probabilidade de sobreviver e gerar descendentes.

Os indivíduos representam potenciais soluções para o problema de otimização, o número total de soluções analisadas a cada iteração constitui a população. As variáveis são codificadas nos genes dos indivíduos utilizando uma representação binária ou em números reais. A população é modificada a cada iteração pelos processos de seleção e reprodução, que serão detalhados a seguir, gerando uma nova população ou geração de indivíduos.

A população é iniciada aleatoriamente no espaço de soluções e a cada indivíduo é atribuído um valor de aptidão ou *fitness*, relacionado ao valor da função objetivo normalizado. A seguir são selecionados os indivíduos para a reprodução. A seleção pode ser do tipo torneio, roleta, elitista, etc. Na seleção por roleta, os indivíduos são distribuídos na roleta de acordo com o

valor de aptidão. A seleção elitista reproduz fielmente os melhores indivíduos na próxima geração.

Os mecanismos de reprodução são o *crossover*, ou cruzamento, e a *mutação*, e sua utilização depende das probabilidades de cruzamento e mutação de cada indivíduo. O cruzamento é considerado o operador de reprodução predominante, tendo preferência sobre a mutação.

O cruzamento é o processo de troca de informações, bits, entre os indivíduos selecionados (pais). A troca de informações pode ocorrer a partir de um ponto escolhido, a partir de um conjunto de pontos ou através do cálculo da probabilidade de troca de variáveis entre os pais.

O operador mutação altera aleatoriamente um ou mais componentes dos indivíduos escolhidos. Novas informações podem ser adicionadas com esse processo garantido a diversidade genética da população. O operador também permite ao algoritmo contornar os mínimos locais da função objetivo.

2.1.1.1 Nondominated Sorting Genetic Algorithm

O *Nondominated Sorting Genetic Algorithm* (NSGA), proposto por Deb em 1994 (SRINIVAS; DEB, 1994), é um algoritmo de otimização multiobjetivo baseado no algoritmo genético de otimização mono-objetivo. Como no algoritmo genético, as soluções são denominadas indivíduos e a evolução dos indivíduos utiliza os mecanismos de seleção, reprodução e mutação. O NSGA foi um dos primeiros algoritmos de otimização multiobjetivo a utilizar o conceito de dominância de Pareto.

A principal diferença entre o NSGA e o algoritmo genético mono-objetivo está na forma de se atribuir o valor de *fitness*. No NSGA, a população é dividida em fronteiras não dominadas, segundo critério de dominância de Pa-

reto (seção 1.2.1), e a cada fronteira é atribuída uma nota diferente, intitulada *dummy fitness*, utilizada como sinalizador de probabilidade de reprodução.

O processo de seleção inicia com a identificação dos indivíduos não dominados na população e atribui um valor de *dummy fitness*. Esses indivíduos são momentaneamente removidos da população e constituirão a primeira fronteira não dominada. Um novo grupo de não dominados é selecionado na população restante, a este grupo é atribuído um novo valor de *dummy fitness*, menor que o menor valor recebido pelo grupo anterior. Esse grupo constituirá a próxima fronteira não dominada, sendo momentaneamente removido da população. O processo é repetido até todos indivíduos pertencerem a uma fronteira não dominada.

A diversidade da população é mantida através do mecanismo de *fitness sharing*. O *fitness sharing* reduz o valor do *dummy fitness* atribuído aos indivíduos da mesma fronteira cuja distância (euclidiana) seja inferior ao valor estipulado no parâmetro de nicho (σ_{shared}). O parâmetro de nicho é uma variável do algoritmo definida pelo usuário no início do processo de otimização.

Deb propõe em (DEB et al., 2000; DEB et al., 2002) uma versão atualizada para o NSGA, na qual são analisados os três principais problemas com relação ao desempenho do NSGA:

- Alto custo computacional do algoritmo de classificação das fronteiras não dominadas
- Falta de elitismo
- Necessidade de especificar o parâmetro de nicho (σ_{share})

Esse algoritmo recebeu o nome de NSGA-II. O algoritmo de classificação sugerido permite que, apenas em uma única análise, todas as soluções sejam

classificadas nas respectivas fronteiras não dominadas. Essa metodologia possui, no pior caso, complexidade igual à $O(MN^2)$, sendo M o número de funções objetivo e N o tamanho da população. O novo algoritmo de classificação reduz a complexidade computacional, pois uma implementação simples do algoritmo de classificação proposto no NSGA (SRINIVAS; DEB, 1994) pode ter complexidade igual à $O(MN^3)$, no pior caso. A informação de qual fronteira não dominada um indivíduo pertence é armazenada na variável i_{rank} .

O elitismo é assegurado com um novo algoritmo de seleção, no qual a prioridade de reprodução favorece os indivíduos com menor valor de i_{rank} e maior contribuição para a diversidade da fronteira de Pareto. A contribuição de uma solução na fronteira de Pareto para a diversidade da fronteira é inversamente proporcional a densidade de soluções na região da fronteira de Pareto onde a solução está localizada.

A densidade aproximada de soluções no entorno de uma solução é aproximada pela soma das distâncias médias entre as soluções vizinhas para cada uma das funções objetivo do problema. O valor dessa soma é denominado distância de agrupamento, sendo mais comumente descrito utilizando a denominação em inglês (*crowding distance*). O próprio Deb, em (DEB et al., 2000; DEB et al., 2002), se refere à *crowding distance* como uma estimativa do perímetro do cuboide cujos vértices são os vizinhos mais próximos da solução. A figura 5 ilustra o cálculo da *crowding distance* para um problema de otimização a 2 objetivos. Portanto, quanto maior o valor da *crowding distance* para uma solução numa certa região do espaço das funções, menor será a densidade de soluções nesta região.

A *crowding distance* é calculada entre os indivíduos pertencentes à uma mesma fronteira não dominada (I). Após ordenar as soluções crescentemente para o valor da função objetivo, a distância para uma solução é dada pela dife-

Figura 5: Cálculo da *crowding distance*.

rença normalizada e absoluta do valor da função objetivo entre duas soluções adjacentes. Uma distância infinita é imposta às soluções de contorno, soluções com os valores máximo e mínimo para a função objetivo, assim, essas soluções sempre são mantidas no arquivo externo e possuem maior probabilidade de reprodução. O cálculo da distância é repetido para cada função objetivo e o valor de *crowding distance* será a somatória dos valores de distância individuais para cada função objetivo. A eq.(2.1) representa o cálculo da distância para os l indivíduos de uma fronteira não dominada I , sendo m o número total de funções objetivo e $f_k(i)$ o valor da função objetivo k para o indivíduo i .

$$I_{distance}(i) = \sum_{k=1}^m (f_k(i+1) - f_k(i-1)) \quad i = 2, \dots, l-1 \quad (2.1)$$

Sendo um valor de aptidão para as soluções, a *crowding distance* é utilizada no processo de seleção do NSGA-II. A probabilidade de reprodução será maior quanto maior for o valor de *crowding distance* da solução, favorecendo as soluções localizadas em regiões menos populosas da fronteira de Pareto. Na redução do arquivo externo, necessária quando o número de soluções na fronteira de Pareto é maior que um valor máximo, as soluções eliminadas

serão as soluções com o menor valor de *crowding distance*.

O NSGA-II encontra a melhor distribuição para as soluções na fronteira de Pareto quando comparado a outros algoritmos evolutivos de otimização multiobjetivo, como observado em (DEB et al., 2002; COELLO; PULIDO; LECHUGA, 2004). Uma das explicações para esta característica é exatamente o uso da *crowding distance* como mecanismo de atribuição de aptidão para as soluções encontradas.

2.1.1.2 Strength Pareto Evolutionary Algorithm

O *Strength Pareto Evolutionary Algorithm* (SPEA) proposto em (ZITZLER; THIELE, 1998a; ZITZLER; THIELE, 1999) é um algoritmo de otimização multiobjetivo derivado do algoritmo genético. As principais características do SPEA são:

- Todas as soluções não dominadas encontradas são armazenadas no arquivo externo
- A aptidão é atribuída de acordo com o conceito de dominância de Pareto
- Redução do número de soluções não dominadas utilizando técnica de cluster, as características da fronteira de Pareto são preservadas
- A aptidão de um indivíduo depende única e exclusivamente das soluções armazenadas no arquivo externo
- Todas as soluções do arquivo externo participam da reprodução

Primeiramente, uma cópia dos indivíduos não dominados é adicionada ao conjunto externo de Pareto. A dominância de Pareto é verificada entre as soluções do arquivo externo e as soluções dominadas são removidas. Portanto,

em cada iteração o arquivo externo possui as soluções não dominadas encontradas até o momento. Esse mecanismo garante que as soluções Pareto ótimas não se percam e também, não restringe ao tamanho da população o número de soluções Pareto ótimas encontradas. Caso o número de soluções no arquivo externo exceda o máximo definido, um método de redução por clusters é aplicado, sem alterar as características da fronteira ótima.

Após atribuir a aptidão aos indivíduos da população e do arquivo externo, é feita a seleção para a reprodução. A técnica de seleção utilizada é a seleção por roleta, as técnicas de reprodução utilizadas são o cruzamento e a mutação. A figura 6 apresenta o algoritmo simplificado do SPEA, evidenciando a distinção entre a população e o arquivo externo.

Figura 6: Algoritmo simplificado do SPEA.

A aptidão de uma solução é atribuída em função da dominância de Pareto, por isso é denominada força de Pareto. Seja p_k uma solução não dominada do

arquivo externo A_s , N o número total de soluções na população e d_k o número de soluções na população dominadas pela solução p_k . A força de Pareto s_k , ou aptidão, da solução p_k será dada pela eq.(2.2).

$$s_k = \frac{d_k}{N + 1} \quad (2.2)$$

A força de Pareto para uma solução dominada da população é a soma das forças de Pareto das soluções que a dominam mais um. A eq.(2.3) representa o cálculo de força para uma solução (p_g) dominada qualquer. Assim a força para as soluções dominadas será sempre maior que um, garantindo prioridade na reprodução para as soluções do arquivo externo.

$$s_g = 1 + \sum_k s_k \quad \forall k \in A_s \mid p_k < p_g \quad (2.3)$$

A redução do número de soluções no arquivo externo é realizada pela técnica de análise por grupo (*Cluster Analysis*), a qual divide um conjunto de p elementos em q grupos relativamente homogêneos. O SPEA utiliza um método hierárquico de cluster conhecido como *average linkage method*, no qual os clusters mais próximos são unidos até que o número máximo de clusters permitido seja atingido. O arquivo externo reduzido será formado por um indivíduo de cada cluster formado, caso o cluster possua mais de um indivíduo será escolhido o mais próximo do centroide.

A figura 7 ilustra o processo de redução por cluster utilizado no SPEA. A figura 7(a) mostra a situação inicial, na qual cada indivíduo pertence a um cluster específico. O número de cluster é reduzido, segundo o processo descrito anteriormente, até restar apenas o número limite de 4 clusters, figura 7(b). O cluster indicado pela cor azul contém as soluções 3, 4 e 5, sendo que apenas a solução 4 será mantida no arquivo externo reduzido.

Figura 7: Funcionamento da técnica de cluster no SPEA. Cada cluster é representado por uma cor diferente.

Em 2001, Zitzler et al. (ZITZLER; LAUMANNNS; THIELE, 2001) propuseram modificações aos mecanismos de atribuição de aptidão, redução do arquivo externo e adicionaram uma nova técnica de estimativa de densidade das soluções na fronteira de Pareto. Esse novo algoritmo recebe o nome de SPEA-2. No entanto, para este trabalho apenas será considerado a atribuição de aptidão para as soluções não dominadas do SPEA original, definida pela eq.(2.2).

Percebe-se influências do NSGA no SPEA na forma de atribuição de *fitness*. Há influência do SPEA no NSGA-II, como o uso de arquivo, que não havia no NSGA. Finalmente, o SPEA-2 tem traços do NSGA-II, principalmente na técnica de estimativa de densidade das soluções na fronteira de Pareto.

2.1.2 Particle Swarm Optimization

O *Particle swarm optimization* (PSO) ou otimização por enxames de partículas é um algoritmo de busca baseado em população. Inspirado no comportamento de aves ou peixes ao buscarem alimento ou abrigo é um algoritmo pertencente à classe de algoritmos conhecida como inteligência de enxame. Os algoritmos desta classe se caracterizam por utilizarem indivíduos dotados de pouca ou nenhuma inteligência, porém capazes de organizar estruturas coletivas complexas para a resolução de problemas de otimização. O algoritmo *Ant Colony Optimization*, proposto em (DORIGO; MANIEZZO; COLORNI, 1996) também é um exemplo desta classe.

O PSO proposto em (KENNEDY; EBERHART, 1995) é um algoritmo de otimização mono-objetivo no qual as soluções potenciais analisadas a cada iteração são denominadas partículas. Cada partícula possui uma posição que é representada pelo vetor de variáveis do problema de otimização. O processo de otimização ocorre por meio de simples regras de movimento (no espaço das soluções). A posição de cada partícula varia em função de sua velocidade, que é calculada em função de seu valor anterior, da distância entre a posição atual da partícula e as melhores posições encontradas por si mesma e por toda a população.

O PSO é um algoritmo de fácil implementação, na sua versão original apenas um único operador gera as novas soluções, sendo um dos algoritmos evolutivos mais populares. Existem várias extensões do PSO para otimização multiobjetivo, como pode ser observado no estado da arte publicado em (COELLO; REYES-SIERRA, 2006) por Coello et al.

O PSO é descrito com maiores detalhes no capítulo seguinte.

3 PARTICLE SWARM OPTIMIZATION

O método de otimização *Particle Swarm Optimization* (PSO) desenvolvido por Kennedy e Eberhart (KENNEDY; EBERHART, 1995) possui origem num algoritmo desenvolvido para simular graficamente o movimento aleatório de uma revoada de pássaros. O algoritmo consistia em inicializar cada “pássaro” com uma posição e velocidade aleatórias em uma grade XY . O efeito de movimento do bando é conseguido atualizando a cada iteração a posição de acordo com a velocidade. Também a cada iteração, a velocidade é atualizada igualando a velocidade de dois indivíduos mais próximos.

A regra de atualização da velocidade leva, em poucas iterações, a um movimento unidirecional. Adicionando uma variável estocástica, chamada pelos autores de “loucura”, conseguiu-se mimetizar o movimento de pássaros com eficiência. A variável “loucura” produzia a cada iteração uma alteração nas velocidades $\vec{V} = (v_x, v_y)$ escolhidas aleatoriamente na população.

A habilidade dos pássaros chegarem a um “poleiro”, um ponto conhecido no plano XY , foi proposta por Heppner em (HEPPNER; GRENANDER, 1990) ao adicionar à simulação uma força dinâmica que faz os pássaros se moverem, eliminando a necessidade da variável loucura. A segunda versão do PSO introduz o conceito do “milharal”, uma posição atratora em $XY = (x_m, y_m)$ na qual os pássaros devem chegar. A localização do milharal é desconhecida pela população, por esse motivo cada representante foi programado para avaliar

sua posição segundo a eq.(3.1), uma função de aferição que mede a distância até o “milharal”. A melhor posição para cada representante é armazenada em $p_{best} = (x_{pbest}, y_{pbest})$ e a melhor posição encontrada por toda população em $g_{best} = (x_{gbest}, y_{gbest})$ a cada iteração. O tamanho total da população n é uma constante do algoritmo.

$$Eval = \sqrt{(x_{atual} - x_m)^2} + \sqrt{(y_{atual} - y_m)^2} \quad (3.1)$$

As velocidades, v_x e v_y , são agora atualizadas por um fator de $r_1 \cdot p_{inc}$ e $r_2 \cdot g_{inc}$ dependendo da posição atual em relação a p_{best} e g_{best} , ou seja, p_{best} e g_{best} funcionam como atratores. Sendo p_{inc} e g_{inc} constantes de incremento da velocidade definidas no início do algoritmo, r_1 e r_2 são variáveis estocásticas. As velocidades para cada pássaro são modificadas de acordo com a seguinte regra:

$$x_{atual} < x_{pbest} \Rightarrow v_x = v_x + rand \cdot p_{inc}$$

$$x_{atual} > x_{pbest} \Rightarrow v_x = v_x - rand \cdot p_{inc}$$

$$y_{atual} < y_{pbest} \Rightarrow v_y = v_y + rand \cdot p_{inc}$$

$$y_{atual} > y_{pbest} \Rightarrow v_y = v_y - rand \cdot p_{inc}$$

$$x_{atual} < x_{gbest} \Rightarrow v_x = v_x + rand \cdot g_{inc}$$

$$x_{atual} > x_{gbest} \Rightarrow v_x = v_x - rand \cdot g_{inc}$$

$$y_{atual} < y_{gbest} \Rightarrow v_y = v_y + rand \cdot g_{inc}$$

$$y_{atual} > y_{gbest} \Rightarrow v_y = v_y - rand \cdot g_{inc}$$

O processo de seleção de p_{best} e g_{best} e atualização da velocidade $\vec{V}$ é repetido até o máximo de iterações ser atingido. Se chamarmos a eq.(3.1) de função objetivo, o “milharal” será o mínimo global desta função enquanto

que os pássaros serão as soluções em potencial. O algoritmo 1 descreve a segunda versão do PSO, e será a base para o algoritmo de otimização baseado na simulação do voo de um grupo de aves.

Algoritmo 1 Pseudocódigo básico do PSO

```

1: Inicializa o enxame aleatoriamente
2: Armazena  $p_{best}$  e  $g_{best}$ 
3:  $t = 0$ 
4: enquanto  $t <$  iterações máxima faça
5:   para todo pássaro faça
6:     Atualiza a velocidade atual
7:     Movimenta os pássaros
8:     Avalia a posição segundo a eq.(3.1)
9:   fim
10:  Armazena  $p_{best}$  e  $g_{best}$ 
11: fim
  
```

3.1 PSO mono-objetivo

No algoritmo PSO, proposto para otimização, as possíveis soluções são agora denominadas partículas e o conjunto total de partículas é chamado de enxame, daí o nome *Particle Swarm Optimization* (Otimização por Enxames de Partículas). Cada partícula i possui: uma posição $\vec{x}_i \in R^n$, sendo R^n o espaço n dimensional formado pelas n variáveis do problema, uma velocidade vetorial $\vec{v}_i = (v_{i1}, \dots, v_{in})$, uma “inércia” $\vec{w}_i$, a sua melhor posição x_{pbest}^i e um valor para a função objetivo f_{pbest}^i . O enxame armazena x_{gbest} e f_{gbest} que são, respectivamente, a melhor posição e o valor da função objetivo obtida por qualquer partícula do enxame.

Inércia é o nome dado a variável $\vec{w}_i$ responsável por ponderar o quanto a velocidade da iteração atual contribuirá no cálculo da velocidade na próxima iteração e foi apresentada por Shi e Eberhart em (SHI; EBERHART, 1998a). A otimização, descrita pelo algoritmo 2, segue o princípio do algoritmo 1, ou seja, a cada iteração se verifica qual a melhor posição para cada partícula

x_{pbest}^i , qual a melhor posição do enxame x_{gbest} e é atualizada a velocidade em função da distância da posição atual da partícula em relação a x_{pbest}^i e x_{gbest} . A posição da partícula é atualizada em função da velocidade e o novo valor da função objetivo é calculado. A atualização da velocidade para a partícula i é calculada segundo a eq.(3.2). A partir do cálculo da velocidade, a posição da partícula é atualizada conforme eq.(3.3), sendo sempre $\Delta t = 1$.

$$v_i(t) = w_i v_i(t-1) + c_1 r_1 (x_{pbest}^i - x_i(t)) + c_2 r_2 (x_{gbest} - x_i(t)) \quad (3.2)$$

$$x_i(t+1) = x_i(t) + v_i(t) \Delta t \quad (3.3)$$

A eq.(3.2) pode ser dividida em três partes: a 1ª é referente à velocidade da partícula anterior, a 2ª é referente à melhor posição da própria partícula e a 3ª é referente à melhor posição do enxame todo. Os termos c_1 e c_2 da eq.(3.2) são denominados de *fator de aprendizado cognitivo* e *fator de aprendizado social*, respectivamente. Sua função é ponderar o quanto a velocidade da partícula é influenciada pelo seu próprio desempenho ou pelo desempenho global e estão definidos no intervalo $[0, 4]$. Os termos r_1 e r_2 da eq.(3.2) são variáveis estocásticas com distribuição uniforme no intervalo $[0, 1]$ e alteram a contribuição da melhor posição individual e global no cálculo da velocidade. Os parâmetros w , c_1 e c_2 alteram diretamente o comportamento do algoritmo em relação a diversos fatores, quais sejam: sua capacidade de escapar de mínimos locais, o número de iterações até a convergência, a habilidade de exploração e refinamento das soluções¹.

¹Mais detalhes do algoritmo PSO podem ser obtidos em: <http://www.swarmintelligence.org/index.php> (acessado em 01/05/2010)

Algoritmo 2 Pseudocódigo detalhado PSO

```

1: Inicia enxame aleatoriamente           % Posição e Velocidade
2: Calcula o aptidão  $f(x)$  para cada partícula
3: para todo partículas faça
4:    $f_{pbest}^i = f(x_i)$ 
5:    $x_{pbest}^i = x_i$ 
6: fim
7: função Encontra Melhor( $f(x), x$ )
8:    $f_{gbest} = \min(f(x))$ 
9:    $x_{gbest} = x_g$  sendo  $f(x_g) = f_{gbest}$ 
10: fim
11:  $t = 1$ 
12: enquanto  $t < maxiter$  faça           %  $maxiter = n^\circ$  máximo de iterações
13:   para cada partícula faça
14:     Atualiza  $V$ 
15:     Atualiza  $X$ 
16:     Calcula  $f(x)$ 
17:     função Atualiza  $pbest(f_{pbest}^i, x_{pbest}^i, f(x_i(t)), x_i(t))$ 
18:       se então  $f(x_i(t)) < f_{pbest}^i$ 
19:          $f_{pbest}^i = f(x_i(t))$ 
20:          $x_{pbest}^i = x_i(t)$ 
21:       fim
22:     fim
23:   fim
24:   função Atualiza  $gbest(f_{gbest}, x_{gbest}, f(x(t)), x(t))$ 
25:     se então  $\min(f(x(t))) < f_{gbest}$ 
26:        $f_{gbest} = \min(f(x))$ 
27:        $x_{gbest} = x_g$  sendo  $f(x_g) = f_{gbest}$ 
28:     fim
29:   fim
30:    $t = t + 1$ 
31: fim

```

3.1.1 Parâmetros do PSO

O algoritmo possui parâmetros que podem ser modificados e ajustados para melhorar seu desempenho em encontrar o ótimo global para a função objetivo analisada. Alguns desses parâmetros são: inércia, fatores de aprendizado, tamanho do enxame, organização do enxame ou topologia, etc. Nesta seção será detalhado o que representa cada parâmetro e como ele atua na evolução do algoritmo.

Inércia O parâmetro inércia (w) foi adicionado ao PSO por Shi e Eberhart em (SHI; EBERHART, 1998a) e sua função é ponderar a importância da velocidade da iteração atual no cálculo da velocidade da iteração seguinte. O parâmetro foi chamado de inércia pelos autores por dificultar a alteração da velocidade das partículas, contudo as partículas não possuem massa. O valor de w influencia a capacidade de exploração e refinamento do algoritmo conforme demonstrado por Shi e Eberhart em (SHI; EBERHART, 1998b).

Um grande valor de w significa dificultar a mudança do movimento de uma partícula em direção à sua melhor posição ou à melhor posição do enxame, a partícula tende a manter seu movimento na direção anterior. Essa característica é desejada nas primeiras iterações, pois possibilita a ampla *exploração* do espaço de soluções, contudo nas iterações finais o movimento das partículas deve privilegiar o *refinamento* da solução realizando apenas pequenos movimentos. Isso é conseguido fazendo com que w seja decrescente ao longo das iterações. Tipicamente w diminui linearmente ou exponencialmente com o valor da iteração.

Fatores de aprendizado Os fatores de aprendizado, como são conhecidas as variáveis c_1 e c_2 , possuem a função de ponderar a influência da me-

lhor posição individual e coletiva no movimento da partícula, conforme pode ser observado na eq.(3.2). O fator c_1 é denominado fator de aprendizado cognitivo por determinar a influência do melhor valor para a função objetivo encontrado pela própria partícula. O fator c_2 é denominado fator de aprendizado social por ponderar a importância do melhor valor para a função objetivo encontrado pelo enxame do qual a partícula faz parte. Os valores de c_1 e c_2 são usualmente definidos no intervalo $[0, 4]$ e tipicamente é considerado $c_1 = c_2$. Os fatores de aprendizado também influenciam o desempenho do método de otimização, contudo o valor de c_1 e c_2 irá depender do valor escolhido para w conforme demonstrado em (EBERHART; SHI, 2000), (CLERC; KENNEDY, 2002) e (KAMEYAMA, 2009).

Tamanho do enxame O tamanho do enxame é a quantidade total de partículas utilizadas na busca pelo ótimo, ou em outras palavras, é a quantidade de soluções avaliadas a cada iteração do algoritmo. Um grande número de partículas implica maior amostragem do espaço de soluções, o que pode levar a uma convergência mais rápida para o ótimo global. Contudo, um número muito grande de partículas pode aumentar o tempo de execução do algoritmo devido ao grande número de chamadas da função objetivo e gerar instabilidade, não permitindo a convergência do enxame para a solução ótima ou até uma convergência precoce.

Topologia do enxame A maneira como o desempenho de uma partícula influencia o desempenho das demais é determinado pela topologia do enxame, ou simplificada, pelo modo que as partículas se conectam. Uma partícula está conectada a um conjunto de partículas que define sua vizinhança e a sua velocidade passa a ser atualizada em função da melhor posição obtida nesta vizinhança, ao invés da melhor posição do enxame. A melhor posição obtida pela vizinhança é denominada x_{leader} e

a melhor posição obtida pelo enxame é denominada x_{gbest} . O algoritmo 2 apresentado em (KENNEDY; EBERHART, 1995) e (SHI; EBERHART, 1998a) considera todas as partículas conectadas entre si, por isso $x_{leader} = x_{gbest}$.

Assumindo que o enxame pode ter uma topologia qualquer, a equação (3.2) da velocidade deve ser modificada substituindo x_{gbest} por x_{leader} resultando na eq.(3.4). Os subconjuntos do enxame, ou vizinhanças, não são isolados compartilhando membros entre si, preservando a interconexão de todo enxame. Isso garante que a informação de boas regiões no espaço de solução será passada aos demais membros do enxame que não pertencem diretamente à vizinhança da partícula que o localizou.

$$v_i(t) = w_i v_i(t-1) + c_1 r_1 (x_{pbest} - x_i(t)) + c_2 r_2 (x_{leader} - x_i(t)) / \Delta u \quad (3.4)$$

Ao alterar o fluxo de informação de boas soluções para o enxame, a topologia influencia a convergência do algoritmo, como demonstrado em (KENNEDY; MENDES, 2002). A topologia pode ser representada por um grafo no qual os vértices são as partículas e as arestas representam as conexões entre as partículas. Existem diversos tipos de topologia para o enxame conforme apresentados em (KENNEDY; MENDES, 2002), (CELLO; REYES-SIERRA, 2006) e (KAMEYAMA, 2009) e alguns exemplos são:

- **Totalmente Conectado:** Neste tipo de topologia todas as partículas estão conectadas entre si e $x_{leader} = x_{gbest}$. A figura 8 ilustra um exemplo de topologia totalmente conectada para um enxame com 8 partículas, S_i é a vizinhança da partícula p_i . Essa topologia apresenta rápida convergência sendo suscetível de convergir para mínimos locais (KENNEDY; MENDES, 2002) dependendo da característica da função objetivo, contudo a diversidade do enxame pode ser garantida através da escolha correta do valor de w (SHI; EBERHART,

1998b).

Figura 8: Topologia totalmente conectada. As 8 partículas estão totalmente conectadas conforme indicado pelas linhas em preto.

- **Melhor local:** A partícula possui como vizinhança as k partículas mais próximas. Um caso especial dessa topologia é a topologia em anel na qual $k = 2$, como está demonstrado na figura 9. Na figura 10 está demonstrada a vizinhança quando $k = 4$. Nas figuras 9 e 10 estão indicadas as vizinhanças S_i , em vermelho, e S_j , em azul, para as partículas p_i e p_j , respectivamente.

Figura 9: Topologia em anel, ou melhor local com $k = 2$. As linhas em preto indicam como as partículas estão conectadas.

- **Roda:** Esta topologia é caracterizada por uma partícula p_0 possuir como vizinhança todo o enxame e para as demais partículas a vizinhança será uma parte do enxame, sempre incluindo a partícula p_0 . A figura 11 ilustra esta topologia para o caso em que a vizinhança

Figura 10: Topologia melhor local com $k = 4$. As linhas em preto indicam como as partículas estão conectadas.

para as partículas, exceto p_0 , é formada pelos 2 vizinhos mais próximos, mais a partícula p_0 .

Figura 11: Topologia Roda. As linhas em preto indicam como as partículas estão conectadas.

3.2 PSO multiobjetivo

As primeiras extensões do algoritmo PSO à otimização multiobjetivo foram propostas em 2000 por Moore et al. (MOORE; CHAPMAN; DOZIER, 2000) e em 2002 por Coello et al. (COELLO; PULIDO; LECHUGA, 2002). Coello et al. em (COELLO; REYES-SIERRA, 2006) apresentam o estado da arte da aplicação do PSO em otimização multiobjetivo. Neste último trabalho são elencadas mais de 25 propostas diferentes para o uso do PSO em otimização a multiobjetivo.

As propostas foram classificadas de acordo com a técnica de abordagem do problema de otimização multiobjetivo em:

- Combinação das funções objetivo em uma única função objetivo
- Ordenação lexicográfica das funções objetivo
- Divisão do enxame
- Conceito de dominância de Pareto
- Metodologias combinadas
- Outras metodologias

Em geral, os algoritmos propostos apresentaram bom desempenho quando comparados com outros métodos amplamente utilizados como o NSGA, o SPEA, etc. Outra característica observada é a rápida convergência apresentada pelos algoritmos, assim como a sua versão em mono-objetivo, reque-rendo em alguns casos a adoção de técnicas para introduzir variação no enxame.

Uma importante questão ao estender o PSO para resolver problemas de otimização multiobjetivo é a escolha da partícula líder para atualizar a velocidade (COELLO; REYES-SIERRA, 2006). Suponha um problema multiobjetivo

no qual se deseja minimizar duas funções, f_1 e f_2 , do qual a figura 12 representa duas possíveis soluções. Não é possível determinar qual das duas soluções, A ou B , é a melhor, pois ambas são não dominadas ($f_1(A) < f_1(B)$ e $f_2(B) < f_2(A)$), ou seja, são potenciais líderes.

Figura 12: Soluções A e B para um problema de otimização a duas funções objetivo.

Algumas metodologias utilizadas na otimização multiobjetivo com PSO dividem o enxame em sub-enxames, sendo cada sub-enxame focado numa das funções objetivo, alguns exemplos de aplicações deste tipo, citados em Coello et al. (COELLO; REYES-SIERRA, 2006), são: o método desenvolvido por Parsopoulos et al. (PARSOPOULOS et al., 2004) e Chow e Tsui (CHOW; TSUI, 2004).

A maioria das propostas, no entanto, considera critérios para seleção de líder baseados na dominância de Pareto, escolhendo o líder entre as soluções não dominadas. Assim, é necessário definir uma medida de *qualidade*, também conhecida como *fitness*, que indique o quão boa a solução não dominada será como líder. Alguns exemplos dessas medidas são a *crowding distance* (NSGA-II) e a força de Pareto (SPEA).

A seguir são propostos dois algoritmos de otimização multiobjetivo baseados no PSO. O primeiro algoritmo é uma variação do algoritmo de Raquel e

Naval (RAQUEL; NAVAL JR., 2005) que utiliza um arquivo externo para armazenar as soluções não dominadas e a *crowding distance* como *fitness*. O líder para cada partícula é escolhido aleatoriamente entre as soluções do arquivo externo com os maiores valores de *crowding distance*, por exemplo entre 20% das soluções com os maiores valores de *crowding distance*.

O segundo algoritmo proposto é igualmente derivado do PSO com alguns elementos do SPEA para otimização multiobjetivo. Tal como no SPEA, as soluções não dominadas são armazenadas no arquivo externo e classificadas de acordo com a força de Pareto, sendo as novas soluções geradas atualizando velocidade e posição dos indivíduos, como no PSO. As restrições são tratadas de duas formas diferentes, função penalidade e arquivo livre, detalhadas mais a frente.

3.2.1 Enhanced Multiobjective Particle Swarm Optimization

O *Multiobjective Particle Swarm Optimization* (MOPSO), proposto por Coello et al. (COELLO; PULIDO; LECHUGA, 2004), utiliza-se da mesma idéia do PSO mono-objetivo para gerar as novas soluções. Novas soluções são geradas a partir da atualização da posição das partículas, segundo a eq.(3.3), pela velocidade que é uma função da velocidade anterior, diferença entre a posição atual da partícula e as melhores posições individual e global do enxame. Contudo, o MOPSO não utiliza os fatores de aprendizado cognitivo e social, c_1 e c_2 respectivamente, sendo a velocidade calculada conforme a eq.(3.5).

$$v_i(t) = wv_i(t-1) + r_1(x_{pbest}^i - x_i(t)) + r_2(x_{gbest} - x_i(t))/\Delta u \quad (3.5)$$

A capacidade de otimização multiobjetivo é resultado do uso de um arquivo externo para armazenar as soluções não dominadas, segundo o critério

de dominância de Pareto, dentre as quais uma será escolhida a cada iteração como melhor global (x_{gbest}). A cada iteração é analisada a dominância no conjunto formado pelas soluções não dominadas, encontradas pelas partículas, e as soluções armazenadas no arquivo externo, sendo o arquivo externo atualizado formado pelas soluções não dominadas deste conjunto.

O *fitness* das soluções no arquivo externo é atribuído segundo a metodologia de grades adaptativas do *Pareto Archived Evolution Strategy* (PAES) (KNOWLES; CORNE, 2000). Segundo a qual, o espaço das funções, conhecido até o momento, é subdividido em hipercubos sendo as soluções alocadas nestes hipercubos de acordo com os valores das funções objetivo, conforme demonstrado na figura 13 extraída de (COELLO; PULIDO; LECHUGA, 2004). Caso a solução adicionada esteja fora das fronteiras atuais da grade, o espaço das soluções é redividido e as soluções são realocadas.

Figura 13: Subdivisão do espaço das funções em hipercubos para distribuição das soluções.

A cada hipercubo é atribuído um valor de *fitness* proporcional ao número

de soluções nele contidas, o *fitness* é menor quanto maior for o número de soluções. A posição x_{gbest} é escolhida aleatoriamente entre as soluções contidas em um hipercubo sorteado. Neste sorteio os hipercubos com maiores valores de *fitness* possuem maior probabilidade de serem sorteados. O valor de *fitness* também será utilizado para definir quais soluções remover do arquivo externo, quando o número máximo é atingido, privilegiando sempre as soluções em regiões menos povoadas do espaço das funções (maior valor de *fitness*).

A melhor posição individual (x_{pbest}^i) é determinada segundo o critério de dominância de Pareto em relação à posição atual da partícula ($x_i(t)$). Se $x_i(t) < x_{pbest}^i(t-1)$, sendo $x_{pbest}^i(t-1)$ a melhor posição encontrada até o momento, então $x_{pbest}^i(t) = x_i(t)$, caso contrário $x_{pbest}^i(t) = x_{pbest}^i(t-1)$. Quando ambas são não dominadas, a $x_{pbest}^i(t)$ é escolhida aleatoriamente entre a posição atual e a melhor posição armazenada.

Um mecanismo, denominado mutação, é utilizado para evitar a convergência do enxame para ótimos locais e garantir a capacidade de exploração do espaço das soluções. A mutação causa uma perturbação aleatória na posição da partícula afetada. Tanto a intensidade quanto o número de partículas afetadas decresce rapidamente com o aumento das iterações. Nas iterações iniciais todo o enxame sofre mutação podendo assumir qualquer posição no espaço das soluções, considerando os limites válidos das variáveis. Garantindo ao algoritmo a capacidade de exploração do espaço de soluções.

O MOPSO mostrou um bom desempenho, comparado com o NSGA-II, *Pareto Archived Evolution Strategy* (PAES) (KNOWLES; CORNE, 2000) e *Micro-Genetic Algorithm* (Micro-GA) (COELLO; PULIDO, 2001; COELLO; ZACATENCO; PULIDO, 2001) em (COELLO; PULIDO; LECHUGA, 2004), conseguindo cobrir a fronteira de Pareto inteira para quase todas as funções de teste analisadas.

Contudo o NSGA-II obteve uma melhor distribuição das soluções na fronteira de Pareto, resultado atribuído ao emprego da *crowding distance* como medida de comparação entre as soluções não dominadas.

Baseados nestes resultados, Raquel e Naval (RAQUEL; NAVAL JR., 2005) propuseram uma variação do algoritmo MOPSO que substitui a técnica de grades adaptativas pela *crowding distance* na atribuição de *fitness* para as soluções da fronteira de Pareto. A *crowding distance*, conforme detalhado na seção 2.1.1.1, estima a densidade de soluções ao redor de cada solução da fronteira de Pareto no espaço das funções objetivo.

A melhor posição global é escolhida aleatoriamente para cada partícula entre as soluções não dominadas com maior valor de *crowding distance*, ou seja, o enxame é direcionado para as regiões mais esparsas da fronteira de Pareto. Quando o arquivo externo é maior que o máximo permitido, as soluções com os menores valores são removidas. Nos demais aspectos, o algoritmo de Raquel e Naval (RNMOPSO) é semelhante ao MOPSO. O algoritmo 3 detalha o funcionamento do RNMOPSO.

O desempenho do RNMOPSO em relação ao MOPSO foi comparado em (RAQUEL; NAVAL JR., 2005) mostrando que o RNMOPSO realmente possui a capacidade de gerar uma fronteira de Pareto mais diversificada, contudo o desempenho médio de ambos algoritmos é incomparável, pois não é possível dizer que a fronteira de Pareto obtida por um deles domina a obtida pelo outro.

Algoritmo 3 RNMOPSO - Pseudocódigo

```

1: Inicializa o enxame
2: Armazena as soluções não dominadas no arquivo externo  $A$ 
3: Calcula a crowding distance para cada partícula  $\in A$ 
4:  $t = 0$ 
   %  $G_{max}$  é nº máximo de iterações
5: enquanto  $t < G_{max}$  faça
6:   para Partículas do enxame faça
7:     Atualiza a velocidade segundo a eq.(3.5)
8:     Atualiza a posição segundo a eq.(3.3)
9:     se  $t < G_{max} * PMUT$  então
       %  $PMUT$  é a probabilidade de mutação
10:      Faz a mutação de  $x_i$ 
11:    fim
12:  fim
13:  Armazena as novas soluções não dominadas em  $A$ 
14:  Reavalia  $A$  e remove as soluções dominadas
15:  Calcula a crowding distance para cada partícula  $\in A$ 
       %  $A_s$  é o nº máximo de elementos permitidos em  $A$ 
16:  se  $size(A) > A_s$  então
17:    Remover as soluções com menor crowding distance
18:  fim
19:   $t = t + 1$ 
20: fim

```

3.2.1.1 O algoritmo do Enhanced Multiobjective Particle Swarm Optimization

O Enhanced Multiobjective Particle Swarm Optimization (EMOPSO), publicado em (COELHO; BARBOSA; LEBENSZTAJN, 2009) e (COELHO; BARBOSA; LEBENSZTAJN, 2010), aqui proposto é uma variação do RNMOPSO cujas principais diferenças são:

- Uso da distribuição truncada de Cauchy para ponderar os fatores de aprendizado cognitivo e social, c_1 e c_2 respectivamente
- Não utilização do operador mutação, pois no PSO não há reprodução

Essas modificações no RNMOPSO foram propostas com o objetivo de melhorar o equilíbrio entre a capacidade de exploração e refino das soluções. O algoritmo 4 detalha o funcionamento do EMOPSO.

A velocidade das partículas é atualizada conforme a eq.(3.6), sendo cd_1 e cd_2 , variáveis aleatórias que utilizam a distribuição truncada (normalizada) de Cauchy no intervalo $[0, 1]$. A distribuição de Cauchy é semelhante à normal como pode ser observado na figura 14, contudo se aproxima tão lentamente do eixo que sua variância tende a infinito. A figura 15 mostra que as variáveis cd_1 e cd_2 podem assumir valores muito maiores do que 1, essa característica permite às partículas “saltarem” grandes distância dispensando o uso do mecanismo de mutação para garantir a diversidade do enxame. Na figura 15 também é possível observar quais seriam os valores aleatórios, a partir de um gerador com distribuição uniforme, para uma variável aleatória com distribuição normal.

$$v_i(t) = w_i v_i(t-1) + c_1 cd_1(x_{pbest}^i - x_i(t)) + c_2 cd_2(x_{gbest} - x_i(t)) \quad (3.6)$$

Algoritmo 4 EMOPSO - Pseudocódigo

```

1: Inicializa o enxame com posições e velocidades aleatórias
2: Avalia as partículas e armazena as soluções não dominadas no arquivo
   externo A
3: para todo Partículas  $\in$  arquivo A faça
4:   Calcula a crowding distance para cada partícula  $\in A$ 
5: fim
6: Ordena A, decrescente para a crowding distance
7:  $t = 0$ 
8: enquanto  $t < G_{max}$  faça           % Em que  $G_{max}$  = n° máximo de iterações
9:   Seleciona aleatoriamente o  $x_{gbest}$  da porção superior de A, por exemplo
   entre o top 20%
10:  para Partículas do enxame faça
11:    Atualiza a velocidade segundo a eq.(3.6)
12:    Atualiza a posição segundo a eq.(3.3)
13:    Calcula a aptidão
14:  fim
15:  Armazena as novas soluções não dominadas em A, caso não sejam
   dominadas por nenhuma solução pertencente A
16:  para todo Partículas  $\in A$  faça
17:    Calcula a crowding distance para cada partícula  $\in A$ 
18:  fim
19:  Ordena A, decrescente para a crowding distance
20:  se  $size(A) > A_s$  então           %  $A_s$  é o n° máximo de elementos permiti-
   dos em A
21:    Remover as soluções inferiores do arquivo, ou seja, com baixo valor
   para a crowding distance
22:  fim
23:   $t = t + 1$ 
24: fim

```

Figura 14: Função densidade de probabilidade para as distribuições normal e de Cauchy.

Figura 15: Gráfico da função que gera números aleatórios com distribuição de Cauchy a partir de números aleatórios uniformemente distribuídos.

Os mecanismos de escolha das melhores posições individuais e global, atribuição de *fitness* para as soluções do arquivo externo e redução do arquivo externo utilizados no EMOPSO são semelhantes aos do RNMOPSO.

3.2.2 Multiobjective Strength Pareto Particle Swarm Optimization

A ideia do *Multiobjective Strength Pareto Particle Swarm Optimization* (MSP2SO) é unir o bom desempenho e simplicidade do PSO mono-objetivo com o SPEA em um algoritmo de otimização multiobjetivo que preserve as boas características do SPEA. As mudanças na população seguem o paradigma do PSO e as possíveis soluções são chamadas de partículas. As soluções não dominadas, segundo o critério de Pareto, são armazenadas no arquivo externo como no SPEA.

A cada iteração a dominância das soluções do arquivo externo em relação ao enxame é verificada: caso uma solução encontrada pelo enxame não seja dominada por nenhuma outra, tanto do arquivo externo quanto do enxame, ela é adicionada ao arquivo externo e caso uma solução do arquivo externo seja dominada por alguma solução do enxame ela é excluída do arquivo. Assim, o arquivo externo representa o conjunto de soluções não dominadas dentre todas as soluções encontradas até o momento. Quando o número de soluções no arquivo é maior que o permitido a redução é feita pela técnica de *cluster* do SPEA.

A cada solução do arquivo é atribuído um valor de *fitness* como no SPEA, ou seja, para cada solução é calculada a força de Pareto s_k segundo a eq.(2.2), repetida na eq.(3.7).

$$s_k = \frac{d_k}{N + 1} \quad (3.7)$$

Sendo N o tamanho do enxame e d_k o número de partículas dominadas no enxame pela solução p_{gbest}^k . O arquivo externo será representado pelo conjunto $X_{gbest} = [p_{gbest}^k], k = 1, \dots, A_s \leq A_{max}$, sendo p_{gbest}^k cada uma das soluções não dominadas, A_s o número de soluções no arquivo externo e A_{max} o número

máximo de soluções permitido no arquivo externo. A posição da solução não dominada p_{gbest}^k será representada por x_{gbest}^k .

A atualização da velocidade deve ser modificada para considerar o conjunto de soluções ótimas (X_{gbest}) e garantir que o enxame se mova através do espaço das soluções em busca da fronteira de Pareto verdadeira. Contudo, a filosofia de atualização da velocidade não mudará: um componente ligado a velocidade anterior da partícula, outro ligado a melhor posição da própria partícula e por último uma parcela ligada a melhor posição global. A eq.(3.2) pode ser reescrita como segue:

$$v_i(t) = w_i v_i(t-1) + c_1 r_1 \Delta_{local}^i + c_2 r_2 \Delta_{global} \quad (3.8)$$

Todas as soluções de X_{gbest} serão consideradas como atratores para o enxame, portanto o termo Δ_{global} possui a contribuição de todas as soluções não dominadas. Então calculamos a diferença entre a posição atual da partícula x_i e cada uma das posições das soluções do arquivo externo x_{gbest}^k . Este vetor de diferenças é ponderado por $1/(1 + s_k)$ e por r_2^k , uma variável estocástica baseada na distribuição beta com $\alpha = \beta = 1/2$, também conhecida como distribuição arco-seno. A função de densidade de probabilidade para a distribuição beta, com $\alpha = \beta = 1/2$, está ilustrada na figura 16.

O cálculo de Δ_{global} para cada partícula é dado por:

$$\Delta_{global} = \sum_{k=1}^{A_s} r_2^k \frac{(x_{gbest}^k - x_i(t))}{(1 + s_k)} \quad (3.9)$$

A melhor posição de cada partícula para cada função objetivo é armazenada em x_{lpbest}^m . Essa informação será utilizada no cálculo de Δ_{local}^i segundo a eq.(3.10), na qual: F é o número total de funções objetivo, x_{lpbest}^m é a posição do melhor resultado para a função objetivo f_m , $x_i(t)$ é a posição atual para a partícula i e r_m^f é uma variável aleatória também baseada na distribuição beta

Figura 16: Densidade de probabilidade para a distribuição beta com $\alpha = \beta = 1/2$.

com $\alpha = \beta = 1/2$.

$$\Delta_{local}^i = \sum_{m=1}^F r_m^f (x_{i_{pbest}}^m - x_i(t)) \quad (3.10)$$

A equação completa para a velocidade é expressa pela eq.(3.11). A posição da partícula é atualizada normalmente conforme a eq.(3.3) do PSO mono-objetivo.

$$v_i(t) = w_i v_i(t-1) +$$

$$c_1 r_1 \left[\sum_{m=1}^F r_m^f (x_{i_{pbest}}^m - x_i(t)) \right] + c_2 r_2 \left[\sum_{k=1}^{A_s} r_2^k \frac{(x_{gbest}^k - x_i(t))}{(1 + s_k)} \right] \quad (3.11)$$

Considerando a metodologia exposta até o momento, o algoritmo 5 representa o pseudocódigo para a MSP2SO aqui proposto.

Algoritmo 5 MSP2SO

```

1: Inicializa o enxame com posições e velocidades aleatórias
2: Avalia o valor das funções objetivo
3: Avalia as partículas e armazena as soluções não dominantes no arquivo
   externo A
4: Armazena a melhor posição para cada função objetivo de cada partícula
5:  $t = 0$ 
6: enquanto  $t < G_{max}$  faça           %  $G_{max}$  = n° máximo de iterações
7:   para todo Partículas  $\in A$  faça
8:     Calcula a força de Pareto  $S_k$ 
9:   fim
10:  para Partículas do enxame faça
11:    Atualiza a velocidade segundo a eq.(3.11)
12:    Atualiza a posição segundo a eq.(3.3)
13:    Calcula a aptidão
14:  fim
15:  Atualiza A com as soluções não dominadas
16:   $t = t + 1$ 
17: fim

```

3.2.2.1 Incluindo restrições no problema de otimização

As restrições existem na maioria dos problemas de otimização reais, por isso é importante que o algoritmo seja capaz de lidar de maneira eficaz com elas. Duas formas para tratamento de restrições são propostas: uso de função penalidade e o uso de arquivo externo livre de restrições, no qual soluções inválidas também pode ser consideradas como não dominadas. O uso de um arquivo externo livre, proposto neste trabalho, é semelhante a metodologia de tratamento das restrições proposta por Deb em (DEB et al., 2002), contudo não há preferência de escolha entre soluções válidas ou inválidas para a geração de novas soluções.

Função penalidade Uma função penalidade quantifica o quanto a solução não atende às restrições do problema de otimização. No geral, é uma soma ponderada do quanto a solução viola cada uma das restrições. Neste trabalho, o valor da penalidade será utilizado para deteriorar o resultado em cada objetivo e esse efeito será mais significativo com o avanço das iterações.

As soluções inválidas também serão admitidas no arquivo externo, desde que o valor para a função penalidade seja menor que um máximo permitido, decrescente com as iterações. Assim, o conjunto de soluções não dominadas final não conterá nenhuma solução inválida.

Arquivo externo livre O arquivo externo passa a ser formado por dois conjuntos: o primeiro contém as soluções não dominadas entre todas as soluções *válidas* encontradas, sendo denominado *arquivo válido*, o outro conterá as soluções não dominadas considerando a resolução do problema *livre* de restrições, sendo denominado *arquivo livre*. Essa metodologia altera apenas o processo de definição de quais soluções são

armazenadas no arquivo externo (X_{gbest}), não modificando a forma como as novas soluções são geradas, ou seja, a posição e a velocidade das partículas ainda são atualizada conforme a eq.(3.3) e eq.(3.11), respectivamente. Deste modo, as soluções válidas presentes em ambos arquivos, válido e livre, influenciarão duplamente a atualização da velocidade das partículas.

Antes de verificar a dominância de Pareto entre as soluções são formados dois conjuntos com as soluções tanto do enxame quanto as soluções já armazenadas no arquivo: um *conjunto válido*, que contém apenas as soluções válidas, e um outro *conjunto livre* que contém todas as soluções e no qual as restrições são desconsideradas.

O arquivo livre é formado pelas soluções não dominadas do conjunto livre e, portanto, poderá conter soluções válidas e inválidas como não dominadas. O cálculo da força de Pareto para essas soluções é feito segundo a eq.(3.12), na qual d_k é o número de soluções dominadas no enxame pela solução k contida no arquivo livre e N é o tamanho total do enxame.

$$s_{k_{A_L}} = \frac{d_k}{(N + 1)} \quad (3.12)$$

Todas as soluções do arquivo livre são dominadas por todas as soluções do arquivo válido, uma vez que podem violar as restrições do problema. Assim, o cálculo da força de Pareto para o arquivo válido é realizado segundo a eq.(3.13), na qual A_L é o número de soluções não dominadas do arquivo livre e d_{k_v} é a quantidade de soluções válidas dominadas pela solução k do arquivo válido.

$$s_{k_{A_V}} = \frac{d_{k_v} + A_L}{(N + 1)} \quad (3.13)$$

O número de elementos no arquivo válido (A_v) deve ser menor ou igual

ao tamanho máximo permitido para o arquivo externo (A_{max}), caso contrário utiliza-se a técnica de *cluster*, tal qual se fazia no SPEA, para a redução do número de soluções. As soluções do arquivo livre são importantes nas iterações iniciais, pois permitem ao algoritmo a exploração do espaço das soluções. Contudo, o tamanho máximo para o arquivo livre ($A_{L_{max}}(t)$) deve ser decrescente para que nas iterações finais somente soluções válidas sejam não dominadas. A figura 17 ilustra um caso de redução no tamanho máximo do arquivo livre ao longo das iterações por patamares.

Figura 17: Redução do número máximo de soluções no arquivo livre por patamares.

A eq.(3.14) define quais são os limites para a quantidade de soluções no arquivo livre e a figura 18 ilustra a atualização do conjunto de Pareto, o arquivo externo, utilizando os arquivos *válido* e *livre*.

$$A_L \leq \begin{cases} A_{L_{max}}, & \text{se } A_v > A_{L_{max}}, \\ A_v, & \text{se } A_v \leq A_{L_{max}}. \end{cases} \quad (3.14)$$

Figura 18: Algoritmo simplificado do funcionamento da atualização do conjunto de Pareto.

4 RESULTADOS E DISCUSSÕES

4.1 Problemas analisados

Dois tipos de problemas de otimização foram analisados: funções de teste analíticas e um problema de otimização real. As funções de teste são problemas de otimização nos quais o objetivo é encontrar o ótimo, mínimo ou máximo, de uma função matemática cujos pontos de ótimo são conhecidos. Comumente utilizados para teste e comparação de algoritmos de otimização. O segundo problema analisado é a otimização, mono e multiobjetivo, de um motor de corrente contínua sem escovas (BRISSET; BROCHET, 2005), que será descrito posteriormente.

4.1.1 Funções de teste mono-objetivo

Encontrar o ponto de mínimo das funções de Rastrigin (TÖRN; ZILINSKAS, 1989) e Rosenbrock (ROSENBROCK, 1960) são dois problemas comumente utilizados para testes de algoritmos de otimização mono-objetivo. Essas funções foram escolhidas por resultarem em problemas de otimização que apresentem uma grande quantidade de mínimos locais e um grande platô, respectivamente.

Rastrigin A função de Rastrigin é uma função não linear multimodal caracterizada pela existência de vários mínimos locais e apenas um único mínimo

global. A função é definida por (4.1), sendo $x_i \in [-5, 12; 5, 12]$ e o único mínimo global em $x = (0, \dots, 0)$ com $f(x) = 0$. A figura 19 mostra a função de Rastrigin para o caso bidimensional, ou seja, $n = 2$.

$$f(x) = 10 \cdot n + \sum_{i=1}^n [x_i^2 - 10 \cos(2\pi x_i)] \quad (4.1)$$

Figura 19: Função de Rastrigin para $n = 2$

Rosenbrock A função de Rosenbrock é caracterizada pela existência de um grande vale no qual está localizado seu único ponto mínimo, localizado em $x = (1, \dots, 1)$ sendo $f(x) = 0$. A função é definida por (4.2), na qual $x_i \in [-\infty, +\infty]$, mas neste trabalho consideraremos apenas $x_i \in [-6, 6]$. A figura 20 ilustra a função para o caso de $n = 2$.

$$f(x) = \sum_{i=1}^{n-1} [(1 - x_i)^2 + 100(x_{i+1} - x_i^2)^2] \quad (4.2)$$

A função de Rastrigin testa a capacidade do algoritmo de escapar de mínimos locais, permitindo a deterioração da função objetivo para explorar o espaço de soluções. A função de Rosenbrock testa a capacidade do algoritmo

Figura 20: Função Rosenbrock para $n = 2$

em gerar novas soluções mesmo com pequenas variações no valor da função objetivo.

4.1.2 Funções de teste multiobjetivo

O teste de algoritmos de otimização multiobjetivo evolutivos são comumente realizados utilizando as funções propostas por Deb e Zitzler em (ZITZLER; DEB; THIELE, 2000). Considerando apenas o caso a dois objetivos, todas as funções de teste propostas por Deb e Zitzler estão estruturadas da seguinte maneira:

$$\begin{aligned}
 \text{minimizar} \quad & T_i(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x})) \\
 \text{Sendo} \quad & f_2(x) = g(\vec{x}) \cdot h(f_1(\vec{x}), g(\vec{x})) \\
 & \vec{x} = (x_1, \dots, x_m)
 \end{aligned} \tag{4.3}$$

Deb e Zitzler propuseram um conjunto de funções de teste, contudo neste trabalho apenas as funções T_1 , T_2 e T_3 são consideradas para teste do algoritmo de otimização MSP2SO, descrito na seção 3.2.2. Para todas as funções será considerado: $m = 30$, $x_i \in [0, 1]$ e a solução ótima é formada com $g(.) = 1$.

T1 A função-teste T_1 possui uma fronteira convexa e as soluções são distribuídas uniformemente no espaço de busca. As funções f_1 , g , h são definidas:

$$\begin{aligned} f_1(x) &= x_1 \\ g(x_2, \dots, x_m) &= 1 + 9 \cdot \left[\sum_{i=2}^m \left(\frac{x_i}{m-1} \right) \right] \\ h(f_1, g) &= 1 - \sqrt{\frac{f_1}{g}} \end{aligned} \quad (4.4)$$

T2 A função-teste T_2 apresenta uma fronteira ótima côncava e é a contraparte de T_1 definida por:

$$\begin{aligned} f_1(x) &= x_1 \\ g(x_2, \dots, x_m) &= 1 + 9 \cdot \left[\sum_{i=2}^m \left(\frac{x_i}{m-1} \right) \right] \\ h(f_1, g) &= 1 - \left(\frac{f_1}{g} \right)^2 \end{aligned} \quad (4.5)$$

T3 A função-teste T_3 apresenta uma fronteira ótima formada por várias partes convexas não contínuas, embora o espaço de busca seja contínuo, definida por:

$$\begin{aligned} f_1(x) &= x_1 \\ g(x_2, \dots, x_m) &= 1 + 9 \cdot \left[\sum_{i=2}^m \left(\frac{x_i}{m-1} \right) \right] \\ h(f_1, g) &= 1 - \sqrt{\frac{f_1}{g}} - \left(\frac{f_1}{g} \right) \cdot \sin(10\pi f_1) \end{aligned} \quad (4.6)$$

A descontinuidade na fronteira de Pareto é causada pela introdução da função *seno* na função h .

4.1.3 Problema de otimização real

O desempenho dos algoritmos em problemas reais será testado utilizando o benchmark proposto por Brisset e Brochet em (BRISSET; BROCHET, 2005) que apresenta o projeto de um motor para ser utilizado na tração de um veículo movido a energia solar, mais detalhes também podem ser obtidos em (MOUS-SOUNI; BRISSET, 2010). O motor escolhido para a aplicação é um motor de

corrente contínua (CC) sem escovas com ímãs permanentes de terras raras (SmCo), fluxo radial, rotor externo devido à instalação do motor nas rodas do veículo, e enrolamentos concentrados cuja geometria está representada na figura 21. O motor é acionado por um inversor trifásico conectado a uma bateria.

Figura 21: Geometria do motor CC sem escovas a ímãs permanentes com rotor externo

O problema de otimização é construído a partir do equacionamento analítico dos fenômenos elétricos, magnéticos e térmicos relacionados ao funcionamento do motor CC sem escovas. O projeto do inversor também é considerado no equacionamento. O projeto do motor possui no total 11 variáveis, mostradas na tabela 1. A partir da definição das variáveis, os autores propuseram 3 problemas distintos de otimização: um a 5 variáveis (variáveis de 1 a 5), outro a 10 variáveis (1 a 10) e por último leva-se em conta todas as variáveis.

Os problemas de otimização estão sujeitos às restrições mostradas na tabela 2. Para o caso de otimização com 5 variáveis, apenas as restrições de 1 a 6 são consideradas, sendo as restrições 7 e 8 consideradas apenas nos

ID	Símbolo	Variáveis	Limites		Unidade
			Inferior	Superior	
1	D_s	Diâmetro estator	150	330	mm
2	B_e	Indução no entreferro	0,5	0,76	T
3	δ	Densidade de corrente	2	5	A/mm^2
4	B_d	Indução média no dente	0,9	1,8	T
5	B_{cs}	Indução magnética média na coroa do estator	0,6	1,6	T
6	L_m	Comprimento do circuito magnético	30	200	mm
7	R_{rs}	Razão entre comprimento rotor-estator	0,8	1,2	
8	e	Entreferro	0,3	2	mm
9	U_{dc}	Tensão de alimentação	50	200	V
10	B_{cr}	Indução magnética média na coroa do rotor	0,6	1,6	T
11	p	Número de polos	4	6	

Tabela 1: Variáveis de construção do motor CC sem escovas proposto por Brisset e Brochet.

ID	Símbolo	Variáveis	Restrição	Unidade
1	M_{tot}	Massa total do motor	≤ 15	kg
2	D_{ext}	Diâmetro externo do motor	≤ 340	mm
3	D_{int}	Diâmetro interno do motor	≥ 76	mm
4	I_{max}	Corrente máxima	≥ 125	A
5	T_a	Temperatura nos ímãs	≤ 120	$^{\circ}C$
6	h_d	Altura da ranhura	≥ 0	mm
7	L_{tot}	Comprimento total do eixo do motor	≤ 100	mm
8	t_1	Tempo de subida da corrente na fase 1	$\leq \frac{1}{30 \cdot f}$	s

Tabela 2: Restrições do problema de otimização mono-objetivo.

casos de otimização a 10 ou 11 variáveis. A restrição I_{max} está relacionada à desmagnetização dos ímãs permanentes, ou seja, é o valor de corrente na fase que gera uma indução suficiente para desmagnetizar os ímãs.

A figura 22 apresenta o fluxograma de cálculo do desempenho do motor, cujo equacionamento está no apêndice A.

Figura 22: Fluxograma de cálculo do motor.

4.1.3.1 Otimização mono-objetivo

O problema de otimização do motor possui como objetivo maximizar a eficiência através da função objetivo $f_o = \eta$, sendo η o rendimento, ou então minimizar $f_o = 1 - \eta$, sujeito às restrições da tabela 2. O caso a 5 variáveis foi resolvido utilizando um algoritmo determinístico chamado *sequential quadratic programming* (SQP) obtendo um valor máximo para o rendimento igual a 95,32 % (MOUSSOUNI; BRISSET, 2010).

O SQP é um dos algoritmos de maior sucesso na resolução de problemas de otimização não lineares com restrições. Ele é um algoritmo de resolução iterativo que utiliza derivadas para localizar o ponto ótimo da função objetivo e é semelhante ao método de Newton para os problemas sem restrições.

4.1.3.2 Otimização multiobjetivo

Uma variação do problema multiobjetivo é proposta e resolvida em (MOUSSOUNI; BRISSET; BROCHET, 2007) e considera como segundo objetivo minimizar a massa total do motor M_{tot} . Sendo f_1 a função que calcula a eficiência e f_2 a massa total do motor, o problema de otimização pode ser descrito como:

$$\begin{cases} \max f_1 \\ \min f_2 \end{cases}$$

As restrições são as mesmas do problema de otimização mono-objetivo, apresentadas na tabela 2, exceto pela massa total. Em (MOUSSOUNI; BRISSET, 2010) é possível encontrar alguns resultados para otimização multiobjetivo com 5 variáveis utilizando diferentes algoritmos de otimização tanto para otimização multiobjetivo quanto mono-objetivo.

4.2 Resultados da otimização mono-objetivo

4.2.1 Ajustes de parâmetros do PSO

Ao utilizar o PSO é preciso definir alguns parâmetros e constantes inerentes ao método, conforme discutido em 3.1.1. Existem na literatura valores recomendados para cada um, por exemplo, utilizar uma população com 20 indivíduos, velocidade máxima $V_{max} = x_{max} - x_{min}$, entre outros. Para determinar um bom conjunto de valores para as constantes w , c_1 e c_2 será realizado um estudo de influência na taxa de acerto e iterações necessárias para a convergência. Para o estudo será considerado como problema de otimização encontrar o mínimo global das funções de teste Rastrigin e Rosenbrock, detalhadas na seção 4.1.1.

O algoritmo PSO implementado utiliza uma topologia de população totalmente conectada, ou seja, a vizinhança de uma partícula é o enxame inteiro. Sendo assim, o líder é o mesmo para o enxame inteiro e na eq.(3.4) x_{leader} é substituído por x_{gbest} . O tamanho da população foi mantido constante igual a 20 e os fatores de aprendizado c_1 e c_2 serão considerados iguais, $c = c_1 = c_2$, deixando a cargo das variáveis estocásticas r_1 e r_2 ponderar a importância de x_{pbest} e x_{leader} no cálculo da velocidade para a próxima iteração.

A influência de w e c é analisada separadamente, assim, quando w varia, o valor de c será mantido constante, tornando a análise mais simples por diminuir o número de combinações possíveis. O algoritmo foi executado 50 vezes para cada valor dos parâmetros, com os seguintes critérios de parada: o número máximo de iterações (3000), ou a convergência até o ótimo da função com uma tolerância igual a 10^{-6} .

O valor do parâmetro inércia w foi adotado decrescente ao longo das iterações, segundo a eq.(4.7) seguindo a recomendação de Shi e Eberhart

(SHI; EBERHART, 1998a). Apenas o valor de w_{max} foi modificado no intervalo $[0,5; 1,4]$ com passo igual a 0,1, nos casos analisados, sendo $w_{min} = 0,4$ mantido constante.

$$w(t) = w_{max} - \left(\frac{w_{max} - w_{min}}{num_max_iter} \right) t \quad (4.7)$$

Figura 23: Taxa de acerto média para cada valor de w_{max} considerando $c = 1,4$.

Figura 24: Iterações até atingir o valor ótimo para cada valor de w_{max} considerando $c = 1,4$.

As figuras 23 e 24 mostram a taxa de acerto e iterações médias, respectivamente, para cada valor de w_{max} utilizando $c = 1,4$. Analisando a figura 23,

uma boa escolha para o valor w_{max} fica no intervalo $[1,0; 1,1]$, pois a taxa de acerto para ambas as funções foi maior que 90 %. Neste gráfico é possível observar também que um valor de w maior no início favorece o desempenho do algoritmo em funções com muitos mínimos locais.

Figura 25: Taxa de acerto média para cada valor de c considerando $w_{max} = 1,1$.

Figura 26: Iterações até atingir o valor ótimo para cada valor de c considerando $w_{max} = 1,1$.

Fixando $w_{max} = 1,1$ foi feita a variação de c no intervalo $[0,4; 2,1]$ e os resultados podem ser observados nas figuras 25 e 26. Na figura 25 conclui-se

que um bom valor para escolha de $c \in [0,9; 1,5]$, e analisando a figura 26, um bom valor de $c \in [0,9; 1,2]$ devido ao aumento do número de iterações. Como para os valores de $c = 1,1$ e $c = 1,2$ a taxa de acerto foi de 100 % para as 50 simulações e uma nova análise para os valores de w_{max} foi executada para verificar qual a melhor combinação para os parâmetros.

Figura 27: Taxa de acerto média para cada valor de w_{max} considerando $c = 1,1$.

Figura 28: Taxa de acerto média para cada valor de w_{max} considerando $c = 1,2$.

As figuras 27, 28, 29 e 30 apresentam os resultados obtidos para a variação de w_{max} com $c = 1,1$ e $c = 1,2$. Comparando as figuras 27 e 28 observa-se

que a taxa de acerto para $w_{max} \in [0,8; 1,2]$ varia em função do valor de c , sendo menor com $c = 1,1$. Contudo, não há alteração significativa no número médio de iterações gasto para atingir o ótimo das funções entre $c = 1,1$ ou $c = 1,2$, conforme observado nas figuras 29 e 30. Pelos resultados apresentados os valores escolhidos foram $w_{max} = 1,1$ e $c = c_1 = c_2 = 1,2$.

Figura 29: Iterações até atingir o valor ótimo para cada valor de w_{max} considerando $c = 1,1$.

Figura 30: Iterações até atingir o valor ótimo para cada valor de w_{max} considerando $c = 1,2$.

4.2.2 Otimização mono-objetivo do motor

O problema de otimização do motor, detalhado na seção 4.1.3, mono-objetivo a 5 variáveis foi utilizado para analisar o desempenho do algoritmo PSO. Para o cálculo da função objetivo e restrições foram utilizados os arquivos fornecidos em (MOUSSOUNI; BRISSET, 2010) adaptados para permitirem o cálculo para a população do enxame de uma única vez. O PSO utilizado possui as mesmas características do PSO utilizado para a aferição dos parâmetros na seção 4.2.1, com os valores de $w_{max} = 1,1$ e $c_1 = c_2 = 1,2$ definidos baseados nos resultados obtidos para as funções de Rastrigin e Rosenbrock. As variáveis de otimização, as restrições e a função objetivo foram tratadas em valores por unidade variando, portanto, no intervalo $[0, 1]$. Como critério de parada foi adotado a seguinte regra:

- x_{gbest} não varia após 50 iterações
- $f_{gbest} > 90 \%$, sendo f_{gbest} o rendimento da melhor solução conhecida

O problema de otimização do motor está sujeito às restrições apresentadas anteriormente na seção 4.1.3.1. Quando uma solução viola essas restrições a sua aptidão é penalizada de um fator $p + 1$, no qual p é a soma de todas as violações em valores por unidade. O termo $+1$ foi adicionado para garantir que as soluções válidas sempre terão melhor aptidão do que as soluções não válidas. A aptidão de uma solução válida é dada por $1 - \eta$, onde η é o rendimento do motor, e para uma solução não válida por $(1 - \eta) + (p + 1)$.

Os resultados do rendimento para cada execução podem ser vistos na figura 31 e as iterações necessárias até convergência na figura 32. O algoritmo de otimização precisa de aproximadamente 100 iterações para obter um valor de eficiência superior a 95 %. A tabela 3 apresenta os valores máximo, mínimo e médio obtidos para o rendimento para as 5 variáveis.

Figura 31: Rendimento do motor obtido pelo PSO em cada execução do algoritmo. A média ficou em 95,25 %, com máximo de 95,35 % e mínimo em 95,0 %.

Variável	Mínimo	Máximo	Média	Unidade
η	95,0	95,35	95,25	%
D_s	192,76	210,80	198,83	mm
B_e	0,64	0,76	0,70	T
δ	2,0	2,92	2,28	A/mm ²
B_e	1,8	1,8	1,8	T
B_{cs}	0,6	1,47	0,98	T

Tabela 3: Valores encontrados pelo PSO para as variáveis em 50 execuções do algoritmo de otimização

A figura 33 apresenta o histograma das variáveis do motor para as soluções obtidas nas 50 execuções. A variável indução média no dente (B_d) não está representada, pois todas as soluções possuem valor igual a 1,8 T. As soluções possuem valores para a indução média na coroa do estator distribuídos ao longo de todo intervalo válido para a variável, ou seja, entre [0,6; 1,6] T.

Figura 32: Iterações até convergência obtido pelo PSO em cada execução do algoritmo. A média ficou em 96, com máximo de 171 e mínimo em 57.

Figura 33: Histograma das soluções.

4.3 Resultados da otimização multiobjetivo

4.3.1 Algoritmo EMOPSO

Ambos os algoritmos MOPSO e RNMOPSO foram testados em funções de teste em (COELLO; PULIDO; LECHUGA, 2004) e (RAQUEL; NAVAL JR., 2005), e apresentaram resultados satisfatórios. Por este motivo neste trabalho comparamos o desempenho dos algoritmos RNMOPSO e EMOPSO na resolução de um problema de otimização real: o problema de otimização multiobjetivo do motor a 10 variáveis, descrito na seção 4.1.3. Os seguintes valores foram utilizados para os parâmetros de controle dos algoritmos RNMOPSO e EMOPSO: tamanho do exame $N = 30$, fatores de aprendizado $c_1 = c_2 = 1,0$, tamanho máximo do arquivo externo $A_{max} = 200$ e número máximo de iterações $G_{max} = 200$.

A figura 34 apresenta a fronteira de Pareto obtida após 30 execuções dos métodos RNMOPSO e EMOPSO, essa fronteira é formada pelas soluções não dominadas no conjunto formado pelas soluções de cada uma das fronteiras de Pareto encontradas a cada execução. É possível observar que as soluções obtidas pelo EMOPSO dominam as soluções obtidas pelo RNMOPSO, ou seja, para um dado valor de massa de uma solução, a solução obtida pelo EMOPSO possui sempre maior valor de eficiência do que a solução obtida pelo RNMOPSO.

As figuras 35 e 36 apresentam o histograma para algumas variáveis de otimização das soluções dominantes obtidas pelo algoritmo EMOPSO. A maioria das soluções dominantes possuem o valor da indução média no dente (B_d) muito próximo de $1,8 T$ e a indução magnética média no ferro do rotor (B_{cr}) muito próximo de $1,6 T$, o que é esperado no projeto clássico de máquinas elétricas.

O histograma para a densidade de corrente (δ) mostra que a maioria das

Figura 34: Fronteira de Pareto utilizando RNMOPSO e EMOPSO

Figura 35: Histograma de algumas variáveis de otimização das soluções não dominadas encontradas.

soluções possuem o valor de densidade no valor mínimo. Possivelmente devido às restrições de corrente máxima de pico e temperatura. As demais variáveis possuem valores distribuídos ao longo do intervalo permitido.

Figura 36: Histograma das demais variáveis de otimização das soluções não dominadas encontradas.

4.3.2 Algoritmo MSP2SO

O algoritmo MSP2SO, detalhado na seção 3.2.2, foi inicialmente testado utilizando as funções de teste T_1 , T_2 e T_3 propostas por Zitzler e Deb (ZITZLER; DEB; THIELE, 2000) e detalhadas na seção 4.1.2. Uma vez validado o funcionamento do algoritmo, ele foi utilizado na resolução do problema de otimização multiobjetivo do motor a 5 variáveis, descrito na seção 4.1.3.2. Os resultados obtidos são apresentados a seguir.

4.3.2.1 Funções de teste

Inicialmente foram considerados os seguintes valores para os parâmetros do MSP2SO: enxame com 20 partículas, w linearmente decrescente com as iterações segundo a eq.(4.7) com $w_{max} = 1,1$ e $w_{min} = 0,4$ e fatores de aprendizado $c_1 = c_2 = 1,2$. Contudo os resultados não foram satisfatórios, a fronteira de Pareto obtida ficou muito distante da fronteira de Pareto ótima, como pode ser observado na figura 37.

Figura 37: Fronteira de Pareto para T1 com antigos valores para w , c_1 e c_2 .

Várias combinações de parâmetros foram testadas para a função de teste $T1$ até se chegar à combinação utilizada: enxame com 200 partículas, w li-

nearmente decrescente com as iterações segundo a eq.(4.7) com $w_{max} = 0,4$ e $w_{min} = 0,001$, fatores de aprendizado $c_1 = c_2 = 3,0$ e número máximo de soluções no arquivo externo $A_{max} = 80$.

O algoritmo com esses parâmetros foi executado 30 vezes para cada uma das funções de teste. As soluções das fronteiras de Pareto obtidas a cada execução serviram para formar uma fronteira de Pareto com as 100 soluções não dominadas encontradas.

As figuras 38, 39 e 40 apresentam as fronteiras de Pareto finais para as funções de teste $T1$, $T2$ e $T3$, marcadas com x em vermelho. A análise dos

Figura 38: Fronteira de Pareto final para T1.

resultados permite concluir que o algoritmo consegue obter uma boa aproximação da fronteira de Pareto verdadeira, indicada pela linha em azul. Também é possível observar um maior acúmulo de soluções para os maiores valores da função objetivo $F2$ para a função de teste $T3$ indicando uma dificuldade do algoritmo em encontrar soluções na região com menores valores para a função objetivo $F2$.

O desempenho para a função $T3$ pode ser melhorado com o ajuste dos parâmetros w , c_1 e c_2 , entretanto o objetivo desta análise é testar a capacidade

Figura 39: Fronteira de Pareto final para T2.

Figura 40: Fronteira de Pareto final para T3.

do algoritmo MSP2SO de encontrar uma boa aproximação para a fronteira de Pareto verdadeira.

4.3.2.2 Otimização do projeto do motor

Após os testes realizados com as funções de Zitzler e Deb, o MSP2SO foi utilizado para resolver o problema de otimização multiobjetivo a 5 variáveis do projeto do motor CC descrito na seção 4.1.3. Inicialmente foi considerado o uso de função penalidade para lidar com as restrições do problema, essa função penaliza os objetivos adicionando um valor positivo proporcional à soma das violações das restrições e à quantidade de iterações.

Algumas simulações foram realizadas para determinar quais seriam os valores para os parâmetros w_{max} , w_{min} , c_1 e c_2 , como ponto de partida foram considerados os valores utilizados para resolver as funções de teste. A comparação entre as fronteiras de Pareto obtidas demonstrou que uma boa combinação de parâmetros seria: $w_{max} = 3$, $w_{min} = 0,001$, $c_1 = 0,3$ e $c_2 = 0,3$. O tamanho do enxame foi mantido em 200, o tamanho máximo do arquivo externo igual a $A_{max} = 100$ e o número máximo de iterações $G_{max} = 300$.

Definidos os valores para os parâmetros, a otimização foi realizada por 30 vezes armazenando a fronteira de Pareto resultante de cada execução. Todas as soluções dessas fronteiras foram agrupadas em um único conjunto para analisar a dominância de Pareto e formar uma fronteira de Pareto final que é mostrada na figura 41. Conforme pode ser observado na figura 41, os resultados obtidos são coerentes com o resultado obtido na otimização mono-objetivo, pois existem soluções ótimas cujo rendimento está em torno de 95% e a massa é inferior a 15 kg.

Como em algumas simulações na primeira iteração nenhuma solução válida foi obtida, algumas execuções redundaram em erro. As soluções não válidas não eram consideradas não dominadas, portanto não permitidas no arquivo externo. A fim de evitar o erro nas execuções as soluções não válidas

Figura 41: Fronteira de Pareto final para o motor CC utilizando função penalidade para tratamento de restrições.

são aceitas no arquivo externo e são removidas com o avanço das iterações, conforme descrito na seção 3.2.2.1. Esta foi a motivação para criar um segundo arquivo, livre de restrições.

O algoritmo foi executado novamente por 30 vezes, considerando o novo tratamento para as restrições, e totalizando as soluções das 30 fronteiras encontradas foram obtidas 958 soluções. Sobre esse conjunto de soluções foi analisada a dominância de Pareto resultando em 75 soluções não dominadas que formam a fronteira de Pareto exibida na figura 42.

Como referência, foi utilizada a fronteira de Pareto obtida com o método de otimização conhecido como SQP. A implementação do SQP utilizada é a disponível no *toolbox* de otimização do Matlab¹. Como o SQP é um otimizador mono-objetivo, a função objetivo é a função definida pela eq.(4.8) que é a soma ponderada das funções objetivo do problema multiobjetivo.

$$f_o = \alpha M_{tot} + (1 - \alpha)(1 - \eta) \quad (4.8)$$

¹<http://www.mathworks.com/products/matlab/>

Figura 42: Fronteira de Pareto final para o motor CC utilizando dois arquivos, livre e válido, para tratamento de restrições.

A fronteira de Pareto é obtida variando-se o valor do coeficiente de ponderação (α) da eq.(4.8) no intervalo $[0, 1]$. E, para escapar de mínimos locais, foram considerados diversos pontos de partida para um mesmo valor de α . Tomou-se ainda um cuidado: o valor da massa total foi normalizado em relação a 15 kg. O resultado obtido com SQP está na fronteira de Pareto representada pela figura 43.

Figura 43: Fronteira de Pareto final obtida pelo SQP.

Colocando as fronteiras de Pareto obtidas em uma mesma figura, conforme se vê na figura 44, é possível observar que a fronteira obtida pelo SQP parece ser um pouco melhor que as demais. Contudo, pouco se pode afirmar sobre a comparação entre as fronteiras obtidas pelo MSP2SO. Para comparar

Figura 44: Comparação entre as fronteiras de Pareto para otimização do motor utilizando SQP e o MSP2SO.

as fronteiras obtidas será utilizada a métrica C , ou cobertura (em inglês *coverage*) introduzida por Zitzler e Thiele em (ZITZLER; THIELE, 1998b). A métrica C , ao comparar dois conjuntos de soluções $\mathcal{A}$ e $\mathcal{B}$, mede quantos elementos do conjunto $\mathcal{B}$ são dominados por algum elemento do conjunto $\mathcal{A}$ e é definida pela eq.(4.9).

$$C(\mathcal{A}, \mathcal{B}) = \frac{|\{b \in \mathcal{B}; \exists a \in \mathcal{A} : a \geq b\}|}{|\mathcal{B}|} \cdot 100 [\%] \quad (4.9)$$

A tabela 4 mostra que a fronteira de Pareto do SQP é a melhor, pois domina 95,54% da obtida com uso de penalidade e 91,04% da obtida com o arquivo livre. Esse resultado já era esperado, contudo a fronteira obtida pelo MSP2SO com arquivo livre possui aproximadamente 9% de soluções não dominadas por nenhuma solução da fronteira do SQP. Observando a figura 45 é

$C(\mathcal{A}, \mathcal{B}) \%$	SQP	Penal.	Livre
SQP	-	92,54	91,04
Penal.	0	-	13,98
Livre	0	71,64	-

Tabela 4: Tabela de valores para a métrica C

possível verificar que o MSP2SO conseguiu localizar soluções com maior valor de eficiência e maior massa do que as soluções encontradas pelo SQP. Ou seja, a ponderação das funções objetivo pode não localizar toda a fronteira possível, mesmo utilizando um eficiente algoritmo otimizador, como o SQP, com 100 valores diferentes para o coeficiente de ponderação.

Figura 45: Detalhe das fronteiras de Pareto na região de maior eficiência.

Ainda da tabela 4, é possível concluir que o uso do arquivo livre permitiu ao MSP2SO encontrar melhores soluções, pois 71,64% da fronteira de Pareto obtida com uso de penalidade é dominada pela fronteira obtida com o uso do arquivo livre.

A figura 46 apresenta o histograma das variáveis para as soluções de todas as fronteiras de Pareto. O histograma é apresentado em porcentagem de soluções, pois há diferenças no número de soluções em cada fronteira. Para

todas metodologias a distribuição das soluções se comporta de forma semelhante para a maioria das variáveis, sendo: a maioria das soluções possui valor de indução média no dente (B_d) próximo de $1,8\text{ T}$ (valor máximo previsto) e indução média no entreferro (B_e) entre $[0,66; 0,68]\text{ T}$, ambos resultados são coerentes com as premissas do projeto clássico de máquinas elétricas. Entretanto, para a indução média na coroa do estator (B_{cs}) o MSP2SO concentra a maioria das soluções no intervalo $[0,6; 0,7]\text{ T}$ enquanto o SQP apresenta valores distribuídos no intervalo $[0,6; 1,0]\text{ T}$.

Figura 46: Histogramas das soluções.

5 CONSIDERAÇÕES FINAIS

A utilização do método de otimização baseado no paradigma de enxames se mostrou satisfatória, tanto para mono-objetivo quanto para multiobjetivo. A análise da influência dos parâmetros no desempenho do PSO, através das funções objetivo de teste, permitiu a escolha consciente dos valores desses parâmetros que foram utilizados na resolução do problema de otimização mono-objetivo para o motor sem escovas.

Os resultados obtidos pelo PSO, para o problema de otimização mono-objetivo do motor sem escovas a 5 variáveis são equivalentes aos encontrados pelos autores do artigo (BRISSET; BROCHET, 2005). O rendimento ficou acima de 95 % para todas as soluções encontradas nas 50 execuções do algoritmo de otimização. Sendo necessário uma média de 100 iterações, com um enxame de 20 partículas, para a convergência para as soluções ótimas. Entretanto, essa solução ótima foi encontrada pelo enxame em apenas 50 iterações, pois o algoritmo só é interrompido após o valor da função objetivo da melhor solução permanecer inalterado por 50 iterações.

Na otimização multiobjetivo um dos algoritmos propostos, o EMOPSO, já foi publicado no artigo de um congresso (COELHO; BARBOSA; LEBENSZTAJN, 2009) e posteriormente em uma revista (COELHO; BARBOSA; LEBENSZTAJN, 2010). O EMOPSO demonstrou desempenho superior ao algoritmo no qual é baseado para o problema de otimização a 10 variáveis do motor sem escovas,

pois as soluções da fronteira de Pareto do EMOPSO dominam as soluções da fronteira de Pareto obtida pelo RNMOPSO, conforme pode ser observado na figura 34. Esse resultado mostra que é possível descartar o operador mutação ao se utilizar uma distribuição de números aleatórios para ponderar os fatores de aprendizado, cognitivo e social, capaz de gerar números aleatórios suficientemente grandes, como o faz a distribuição de Cauchy, para causar um “salto” na partícula. Ao utilizar essa distribuição, o EMOPSO também reduz a dependência em relação aos valores atribuídos aos fatores de aprendizado cognitivo e social.

No MSP2SO o uso do arquivo de soluções não dominadas como melhor global para atualizar a velocidade das partículas mostrou-se uma técnica eficiente para gerar novas soluções, possibilitando uma partícula ser influenciada por múltiplas soluções da fronteira de Pareto. Contribuiu também para o sucesso do algoritmo a utilização de um arquivo que permita as soluções inválidas serem consideradas não dominadas, pois capacita o algoritmo a explorar uma região com alta concentração de soluções inválidas na busca por uma solução válida Pareto ótima.

Ainda sobre o MSP2SO, é fácil notar a dependência entre o problema de otimização e o valor dos parâmetros inércia e fatores de aprendizado, pois existe uma grande diferença entre os valores adotados na resolução das funções de teste e na otimização do motor. Para reduzir esta interação, uma sugestão de trabalho futuro é modificar o EMOPSO adicionando o arquivo livre para tratar as restrições e definindo como melhor global todo o arquivo externo.

Os algoritmos propostos precisam evidentemente serem testados em outros problemas definindo melhor seu comportamento em diferentes situações de complexidade e para diferentes classes de problemas de otimização.

REFERÊNCIAS

BRISSET, S.; BROCHET, P. Analytical model for the optimal design of a brushless dc wheel motor. Compel-the International Journal for Computation and Mathematics in Electrical and Electronic Engineering, v. 24, p. 829–848, 2005. ISSN 0332-1649.

CHOW, C. kin; TSUI, H. tat. Autonomous agent response learning by a multi-species particle swarm optimization. In: Evolutionary Computation, 2004. CEC2004. Congress on. Oregon, US: IEEE Press, 2004. v. 1, p. 778 – 785 Vol.1.

CLERC, M.; KENNEDY, J. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. Evolutionary Computation, IEEE Transactions on, v. 6, n. 1, p. 58–73, 2002. ISSN 1089-778X. Disponível em: <http://dx.doi.org/10.1109/4235.985692>.

COELHO, L. dos S.; BARBOSA, L.; LEBENSZTAJN, L. Multiobjective particle swarm approach for the design of a brushless dc wheel motor. Magnetics, IEEE Transactions on, v. 46, n. 8, p. 2994 –2997, aug. 2010. ISSN 0018-9464.

COELHO, L. dos S.; BARBOSA, L. Z.; LEBENSZTAJN, L. Multiobjective particle swarm approach for the design of a brushless dc wheel motor. In: 17th Conference on the Computation of Electromagnetic Fields. Florianópolis, BR: Proceedings of 17th Conference on the Computation of Electromagnetic Fields, 2009. único, p. 245–246.

COELLO, C.; PULIDO, G.; LECHUGA, M. An extension of particle swarm optimization that can handle multiple objectives. In: Workshop on Multiple Objective Metaheuristics. Paris, FR: [s.n.], 2002.

_____. Handling multiple objectives with particle swarm optimization. Evolutionary Computation, IEEE Transactions on, v. 8, n. 3, p. 256 – 279, june 2004. ISSN 1089-778X.

COELLO, C. A. C.; REYES-SIERRA, M. Multi-objective particle swarm optimizers: A survey of the state-of-the-art. International Journal of Computational Intelligence Research, v. 2, 2006. ISSN 0973-1873.

COELLO, C. A. C.; ZACATENCO, S. P.; PULIDO, G. T. Multiobjective optimization using a micro-genetic algorithm. In: Genetic and Evolutionary Computation Conf.(GECCO'2001). San Francisco, US: [s.n.], 2001. p. 274–282.

COELLO, C. C. C.; PULIDO, G. T. A micro-genetic algorithm for multiobjective optimization. In: ZITZLER, E. et al. (Ed.). Evolutionary Multi-Criterion Optimization. Springer Berlin / Heidelberg, 2001, (Lecture Notes in Computer Science, v. 1993). p. 126–140. Disponível em: http://dx.doi.org/10.1007/3-540-44719-9_9.

COOK, W. The Traveling Salesman Problem. 2012. Online-Acessado em: 01 mar. 2012. Disponível em: <http://www.tsp.gatech.edu/index.html>.

DEB, K. et al. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. In: SCHOENAUER, M. et al. (Ed.). Parallel Problem Solving from Nature PPSN VI. Springer Berlin / Heidelberg, 2000, (Lecture Notes in Computer Science, v. 1917). p. 849–858. Disponível em: http://dx.doi.org/10.1007/3-540-45356-3_83.

_____. A fast and elitist multiobjective genetic algorithm: Nsga-ii. Evolutionary Computation, IEEE Transactions on, v. 6, n. 2, p. 182 –197, apr 2002. ISSN 1089-778X.

DORIGO, M.; MANIEZZO, V.; COLORNI, A. Ant system: optimization by a colony of cooperating agents. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, v. 26, n. 1, p. 29 –41, feb 1996. ISSN 1083-4419.

EBERHART, R.; SHI, Y. Comparing inertia weights and constriction factors in particle swarm optimization. In: Evolutionary Computation, 2000. Proceedings of the 2000 Congress on. La Jolla, US: IEEE Press, 2000. v. 1, p. 84 –88 vol.1. Disponível em: <http://dx.doi.org/10.1109/CEC.2000.870279>.

GOLDBERG, D. Genetic algorithms in search, optimization, and machine learning. [S.l.]: Addison-wesley, 1989.

HEPPNER, F.; GRENNANDER, U. A stochastic nonlinear model for coordinated bird flocks. In: KRASNER, E. (Ed.). The ubiquity of chaos. [S.l.]: AAAS Publications, 1990. p. 233–238.

HWANG, C.; MASUD, A. Multiple objective decision making, methods and applications: a state-of-the-art survey. Springer-Verlag, 1979. (Lecture notes in economics and mathematical systems). ISBN 9780387091112. Disponível em: <http://books.google.com.br/books?id=Hz-yAAAAIAAJ>.

KAMEYAMA, K. Particle swarm optimization - a survey. IEICE Transactions on Information and Systems, E92-D, n. 7, p. 1354–1361, 2009. Cited By (since 1996) 5.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: Neural Networks, 1995. Proceedings., IEEE International Conference on. Perth, AU: IEEE Press, 1995. v. 4, p. 1942 –1948 vol.4.

KENNEDY, J.; MENDES, R. Population structure and particle swarm performance. In: Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on. Honolulu, US: IEEE Press, 2002. p. 1671–1676.

KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. Optimization by simulated annealing. Science, v. 220, n. 4598, p. 671–680, 1983. Disponível em: <http://www.sciencemag.org/content/220/4598/671.abstract>.

KNOWLES, J. D.; CORNE, D. W. Approximating the nondominated front using the pareto archived evolution strategy. Evolutionary Computation, MIT Press, Cambridge, MA, US, v. 8, p. 149–172, June 2000. ISSN 1063-6560. Disponível em: <http://dx.doi.org/10.1162/106365600568167>.

MICHALEWICZ, Z. Do not kill unfeasible individuals. In: Proceedings of the Fourth Intelligent Information Systems Workshop (IIS'95). [S.l.: s.n.], 1995. p. 110–123.

MOORE, J.; CHAPMAN, R.; DOZIER, G. Multiobjective particle swarm optimization. In: Proceedings of the 38th annual on Southeast regional conference. New York, NY, USA: ACM, 2000. (ACM-SE 38), p. 56–57. ISBN 1-58113-250-6. Disponível em: <http://doi.acm.org/10.1145/1127716-1127729>.

MOUSSOUNI, F.; BRISSET, S. A Benchmark for a Mono and Multi Objective Optimization of the Brushless DC Wheel Motor. 2010. Online-Accessado em 30/ago/2010. Disponível em: <http://l2ep.univ-lille1.fr/come/benchmark-wheel-motor.htm>.

MOUSSOUNI, F.; BRISSET, S.; BROCHET, P. Some results on the design of brushless dc wheel motor using sqp and ga. International Journal of Applied Electromagnetics and Mechanics, IOS Press, v. 26, n. 3, p. 233–241, 2007.

PARETO, V. Cours D'Economie Politique. Lausanne: F. Rouge, 1986.

PARSOPOULOS, K. E. et al. Multiobjective optimization using parallel vector evaluated particle swarm optimization. In: In Proceedings of the IASTED International Conference on Artificial Intelligence and Applications (AIA 2004). Madri, ES: ACTA Press, 2004. p. 823–828.

RAQUEL, C. R.; NAVAL JR., P. C. An effective use of crowding distance in multiobjective particle swarm optimization. In: Proceedings of the 2005 conference on Genetic and evolutionary computation. New York, NY, US: ACM, 2005. (GECCO '05), p. 257–264. ISBN 1-59593-010-8. Disponível em: <http://doi.acm.org/10.1145/1068009.1068047>.

ROSENBROCK, H. H. An automatic method for finding the greatest or least value of a function. The Computer Journal, v. 3, n. 3, p. 175–184, 1960. Disponível em: <http://comjnl.oxfordjournals.org/content/3/3/175.abstract>.

SHI, Y.; EBERHART, R. A modified particle swarm optimizer. In: Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on. Anchorage, US: IEEE Press, 1998. p. 69 –73.

SHI, Y.; EBERHART, R. C. Parameter selection in particle swarm optimization. In: Proceedings of the 7th International Conference on Evolutionary Programming VII. London, UK: Springer-Verlag, 1998. (EP '98), p. 591–600. ISBN 3-540-64891-7. Disponível em: <http://portal.acm.org/citation.cfm?id=647902.738978>.

SRINIVAS, N.; DEB, K. Multiobjective optimization using nondominated sorting in genetic algorithms. Evolutionary Computation, v. 2, p. 221–248, 1994.

TÖRN, A. A.; ZILINSKAS, A. Global optimization. [S.l.]: Springer-Verlag, Berlin, 1989. (Lecture Notes in Computer Science, v. 350).

VELDHUIZEN, D.; LAMONT, G. Multiobjective evolutionary algorithms: Analyzing the state-of-the-art. Evolutionary computation, MIT Press, v. 8, n. 2, p. 125–147, 2000.

ZITZLER, E.; DEB, K.; THIELE, L. Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary Computation, v. 8, n. 2, p. 173–195, 2000. Disponível em: <http://www.mitpressjournals.org/doi/abs/10.1162/106365600568202>.

ZITZLER, E.; LAUMANN, M.; THIELE, L. Spea2: Improving the strength pareto evolutionary algorithm. Eidgenössische Technische Hochschule Zürich (ETH), Institut für Technische Informatik und Kommunikationsnetze (TIK), 2001.

ZITZLER, E.; THIELE, L. An Evolutionary Algorithm for Multiobjective Optimization: The Strength Pareto Approach. [S.l.], May 1998.

_____. Multiobjective optimization using evolutionary algorithms - a comparative case study. Fifth International Conference on Parallel Problem Solving from Nature (PPSN-V), Springer, Berlin, GE, p. 292–301, 1998.

_____. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. Evolutionary Computation, IEEE Transactions on, v. 3, n. 4, p. 257 –271, nov 1999. ISSN 1089-778X.

APÊNDICE A – PROJETO DO MOTOR DE CORRENTE CONTÍNUA SEM ESCOVAS

Em (BRISSET; BROCHET, 2005) os autores propõem o projeto ótimo de um motor de corrente contínua sem escovas a ímãs permanentes para teste de algoritmos de otimização. Diferentemente de *benchmarks* existentes até o momento que se utilizam do método dos elementos finitos, os autores propõem o projeto do motor através de um equacionamento analítico. O uso de modelos analíticos é útil na fase inicial do projeto de uma máquina elétrica quando os valores de muitos parâmetros ainda são desconhecidos.

O motor foi projetado para equipar um veículo movido a energia solar durante uma corrida e será instalado em uma roda. Assim o motor escolhido é um motor a ímãs permanentes de terras raras (SmCo) com enrolamentos concentrados, fluxo radial e rotor externo. O motor será acionado por um inversor trifásico conectado a uma fonte de tensão de corrente contínua.

A seguir são apresentadas as equações de dimensionamento do motor que formam um sistema não linear de 78 equações e 78 incógnitas, incluindo variáveis e constantes. A ordenação correta das equações permite a resolução sequencial, na ordem apresentada, restando apenas a resolução iterativa de um sub-sistema de 7 equações.

1. Definição dos parâmetros de otimização a 5 variáveis:

• Parâmetros variáveis:

- (a) $D_s \in [0, 150; 0, 330]$ (m) % Diâmetro do estator
- (b) $B_e \in [0, 5; 0, 76]$ (T) % Indução magnética no entreferro
- (c) $\delta \in [2 \cdot 10^6; 5 \cdot 10^6]$ (A/m²) % Densidade de corrente
- (d) $B_d \in [0, 9; 1, 8]$ (T) % Indução magnética média no dente
- (e) $B_{cs} \in [0, 6; 1, 6]$ (T) % Indução magnética média no ferro do estator

• Parâmetros fixos:

- (a) $L_m = 0,045$ (m) % Comprimento do circuito magnético do motor
- (b) $R_{rs} = 1,11$ % Razão entre o comprimento do rotor e do estator
- (c) $e = 0,8 \cdot 10^{-3}$ (m) % Entreferro
- (d) $U_{dc} = 120$ (V) % Tensão da fonte de alimentação
- (e) $B_{cr} = 1,2$ (T) % Indução magnética média na armadura
- (f) $p = 6$ % Número de polos

2. Definição dos parâmetros de otimização a 10 variáveis:

• Parâmetros variáveis:

- (a) $D_s \in [0, 150; 0, 330]$ (m) % Diâmetro do estator
- (b) $B_e \in [0, 5; 0, 76]$ (T) % Indução magnética no entreferro
- (c) $\delta \in [2 \cdot 10^6; 5 \cdot 10^6]$ (A/m²) % Densidade de corrente
- (d) $B_d \in [0, 9; 1, 8]$ (T) % Indução magnética média no dente
- (e) $B_{cs} \in [0, 6; 1, 6]$ (T) % Indução magnética média no ferro do estator
- (f) $L_m \in [0,030; 0,200]$ (m) % Comprimento do circuito magnético do motor

(g) $R_{rs} \in [0, 8; 1, 2]$ % Razão entre o comprimento do rotor e do estator

(h) $e \in [0, 2 \cdot 10^{-3}; 2 \cdot 10^{-3}]$ (m) % Entreferro

(i) $U_{dc} \in [50; 200]$ (V) % Tensão da fonte de alimentação

(j) $B_{cr} \in [0, 6; 1, 6]$ (T) % Indução magnética média na armadura

•Parâmetros fixos:

(a) $p = 6$ % Número de polos

3. Definição dos parâmetros de otimização a 11 variáveis:

•Parâmetros variáveis:

(a) $D_s \in [0, 150; 0, 330]$ (m) % Diâmetro do estator

(b) $B_e \in [0, 5; 0, 76]$ (T) % Indução magnética no entreferro

(c) $\delta \in [2 \cdot 10^6; 5 \cdot 10^6]$ (A/m²) % Densidade de corrente

(d) $B_d \in [0, 9; 1, 8]$ (T) % Indução magnética média no dente

(e) $B_{cs} \in [0, 6; 1, 6]$ (T) % Indução magnética média no ferro do estator

(f) $L_m \in [0, 030; 0, 200]$ (m) % Comprimento do circuito magnético do motor

(g) $R_{rs} \in [0, 8; 1, 2]$ % Razão entre o comprimento do rotor e do estator

(h) $e \in [0, 2 \cdot 10^{-3}; 2 \cdot 10^{-3}]$ (m) % Entreferro

(i) $U_{dc} \in [50; 200]$ (V) % Tensão da fonte de alimentação

(j) $B_{cr} \in [0, 6; 1, 6]$ (T) % Indução magnética média na armadura

(k) $p \in [4, 6]$ % Número de polos

4. Especificação da carga e rotação:

• $C = 20$ (Nm) % Carga aplicada

- $\Omega = 721.\pi/30 \text{ (rad/s)}$ % Rotação em regime
- $\Omega_{max} = 1442.\pi/30 \text{ (rad/s)}$ % Rotação em vazio

5. Propriedades dos materiais:

$$\begin{aligned}
 kr &= 0,5; & \rho_{cu} &= 1,72.10^{-8} \text{ (}\Omega.m\text{)} & q_t &= 2,5 \text{ (W/kg)} \\
 B_r &= 1,042 \text{ (T)} & d_t &= 7850 \text{ (kg/m}^3\text{)} & f_t &= 50 \text{ (Hz)} \\
 B_c &= 0,05 \text{ (T)} & d_a &= 7400 \text{ (kg/m}^3\text{)} & B_t &= 1,5 \text{ (T)} \\
 \mu_a &= 1,05 & d_{cu} &= 8950 \text{ (kg/m}^3\text{)} & k_{foi} &= 0,95 \\
 \mu_0 &= 4\pi 10^{-7} \text{ (Tm/A)} & d_{cr} &= 7850 \text{ (kg/m}^3\text{)} & P_m &= 15 \text{ W} \\
 k_{fui} &= 0,8 & \alpha_a &= -5.10^{-4} \text{ (K}^{-1}\text{)} & \alpha_{cu} &= 3,8.10^{-3} \text{ (K}^{-1}\text{)} \\
 h &= 10 \text{ (W/m}^2\text{K)} & T_{ext} &= 50 \text{ (}^\circ\text{C)}
 \end{aligned}$$

6. Alguns cálculos do dimensionamento elétrico do motor:

- $N_e = 3/2 * p$ % Número de ranhuras
- $k_v = \Omega_{max}/\Omega$ % k_v Razão entre a força eletromotriz e metade da tensão de alimentação
- $E = U_{dc}/2/k_v$ % E Força eletromotriz
- $I = C * \Omega/2/E$ % I Corrente de alimentação
- $n = 4 * E/B_e/D_s/L_m/\Omega$ % n Número de espiras

7. Cálculo do dimensional do motor:

- $S_{enc} = 3/2 * n * I/\delta/kr$ % S_{enc} Área total das ranhuras
- $\alpha = \pi/p$ % α Passo polar do estator
- $\beta = \pi/p$ % β Passor polar do rotor.
- $\alpha_i = \alpha/5$ % α_i Ângulo de abertura do dente intermediário
- $l_i = B_e/B_d * \alpha_i * D_s/2$ % l_i Largura do dente intermediário
- $l_d = B_e/B_d * \alpha * D_s/2$ % l_d Largura do dente

- $e_b = (B_e/B_d * (\alpha/2 - \arcsin(l_d/D_s)) * D_s/2) + D_s/2 * (1 - \cos(\arcsin(l_d/D_s)))$
% e_b Altura central da sapata polar
- $B_a = B_e * \alpha * D_s/2 / \beta / (D_s/2 + e) / R_{rs} / k_{fui}$ % B_a Indução magnética no imã
- $h_{cs} = 1/2 * B_d/B_{cs} * l_d$ % h_{cs} Altura do núcleo do rotor
- $h_{cr} = 1/2 * B_a/B_{cr} * \beta * (D_s/2 + e)$ % h_{cr} Altura do núcleo do estator
- Cálculo da altura da ranhura (h_d) dada pela eq.(A.1).

$$S_{enc} \approx h_d [2\pi (D_s/2 - e_b) - \pi h_d - N_e (l_i + l_d)] \quad (A.1)$$

- $N_e = 3/2 * p$ % N_e Número de ranhuras
- $a = \pi i$
- $b = -(2 * \pi * (D_s/2 - e_b) - N_e * (l_i + l_d))$
- $c = S_{enc}$ % S_{enc} Área total das ranhuras
- $\Delta = b^2 - 4 * a * c$ % Cálculo do determinante
- $h_d = (-b - \sqrt{abs(\Delta)}) / 2/a$ % h_d Altura da ranhura. Uma das restrições.
- $h_c = (e_b / \cos(\alpha/2) - D_s/2 * (1 / \cos(\alpha/2) - 1))$ % h_c Altura da extremidade da sapata polar
- $h_i = (D_s/2 * (1 - \cos(\alpha_i/2)) + h_c * \cos(\alpha_i/2))$ % h_i Altura central da sapata do dente intermediário
- $D_{int} = D_s - 2 * (e_b + h_d + h_{cs})$ % D_{int} Diâmetro interno do motor. Uma das restrições.
- $R_{tb} = ((l_d - l_i)/4 + (D_s/2 - e_b - h_d/2) * \pi/2/N_e)$ % R_{tb} Raio médio de uma cabeça de bobina
- $L_{ds} = (L_m/k_{foi} + \pi * R_{tb})$ % L_{ds} Comprimento médio de uma meia volta

- $L_{tot} = (L_m/k_{foi} + 2*((D_s/2 - e_b - h_d/2)*\pi/N_e - l_i/2)) \% L_{tot}$ Comprimento axial total do motor. Uma das restrições.

8.Cálculo das perdas elétricas e temperatura:

- $f = p * \Omega/2/\pi \% f$ Frequência de acionamento
- $M_{cs} = d_t * \pi * h_{cs} * (-h_{cs} + 2 * (D_s/2 - e_b - h_d)) * L_m \% M_{cs}$ Massa do núcleo do estator
- $M_{ds} = d_t * N_e * ((l_d + l_i) * h_d + D_s/2 * (\alpha_i * (h_i + h_c)/2 + \alpha * (e_b + h_c)/2)) * L_m \% M_{ds}$ Massa do dente do estator
- $P_f = q_t * ((f/f_t)^1, 5) * (M_{cs} * ((B_{cs}/B_t)^2) + M_{ds} * ((B_d/B_t)^2)) \% P_f$ Perdas no ferro
- Sistema de equações a ser resolvido para determinar os valores de: $T_a, T_{cu}, h_a, D_{ext}, R_{ph}, P_j$ e S_{ext} .
 - $T_{cu} = T_a \% T_{cu}$ Temperatura no cobre
 - $(B_a - B_r(1 + \alpha_a * T_a))/\mu_a * h_a + B_e * e \% T_a$ Temperatura na armadura ou nos ímãs. Uma das restrições.
 - $D_{ext} = D_s + 2(e + h_a + h_{cr}) \% D_{ext}$ Diâmetro externo do motor. Uma das restrições.
 - $R_{ph} = \rho_{cu}(1 + \alpha_{cu} * T_{cu}) * n/2 * L_{ds} * \delta/I \% R_{ph}$ Resistência por fase
 - $P_j = 2 * R_{ph} * I^2 \% P_j$ Perdas por efeito Joule
 - $S_{ext} = \pi/2 * D_{ext}^2 + \pi * D_{ext} * L_{tot} \% S_{ext}$ Área total externa do motor

9.Cálculo da corrente de acionamento

- $I_{max} = -4 * p/n * (B_c/\mu_0 * \beta/\alpha * (1 + 2 * e/D_s) * R_{rs} * k_{fui} * e + (B_c - B_r * (1 + \alpha_a * T_a)) * h_a/\mu_0/\mu_a); \% I_{max}$ Valor máximo para a corrente, correntes acima deste valor podem causar a desmagnetização dos ímãs.

- Cálculo da indutância do motor, considerando rotor travado substituição dos ímãs por ar e 2 fases conectadas em série

$$- \phi_{enc}/I_{rc} = \mu_0/16 * (n^2)/N_e * h_d * L_m / ((D_s/2 - e_b - h_d/2) * \pi / N_e - (l_i + l_d)/2)$$

% ϕ_{enc}/I_{rc} Fluxo magnético na ranhura por unidade de corrente

$$- \phi_{ent}/I_{rc} = \mu_0 * 3/16 * (n^2)/N_e * L_m / (e + h_a) * \alpha * D_s/2$$

% ϕ_{ent}/I_{rc} Fluxo magnético no entreferro por unidade de corrente

$$- \phi_{bec}/I_{rc} = \mu_0 * 3/16 * (n^2)/N_e * h_c * L_m / (D_s/2 - h_c/2) / (\pi / N_e - (\alpha + \alpha_i)/2)$$

% ϕ_{bet}/I_{rc} Fluxo magnético através do entreferro e da sapata polar

$$- L_{ph} = 3/2 * \phi_{ent}/I_{rc} + 2 * (\phi_{enc}/I_{rc} + \phi_{bec}/I_{rc})$$

% L_{ph} Indutância por fase

- $t_1 = -L_{ph}/R_{ph} * \log(abs(1 - 3 * R_{ph} * I/U_{dc}/(2 - 1/k_v)))$ % t_1 Tempo de subida da corrente na fase 1

10. Cálculo da massa de alguns componentes:

- $M_a = d_a * p * \beta * h_a * (h_a + 2 * (D_s/2 + e)) * L_m * rrs$ % M_a Massa dos ímãs

- $M_{cr} = d_{cr} * \pi * h_{cr} * (h_{cr} + 2 * (D_s/2 + e + h_a)) * L_m * R_{rs}$; % Massa do núcleo do rotor

- $M_{cu} = d_{cu} * 3/2 * n * I/\delta * L_{ds}$ % M_{cu} Massa do enrolamento

11. Funções objetivo:

- $\eta = (C * \omega - P_m)/(C * \omega + P_j + P_f)$ % η Objetivo 1: Rendimento do motor

- $M_{tot} = M_a + M_{cr} + M_{cs} + M_{ds} + M_{cu}$ % M_{tot} Objetivo 2: Massa total do motor