Universidade de Pernambuco (UPE)
Escola Politécnica de Pernambuco (POLI)
Programa de Pós-graduação em Engenharia da Computação (PPGEC)
Programa de Pós-graduação em Engenharia de Sistemas (PPGES)

Relatório da Prática de Inteligência de Enxames

Aluno: Carlos Henrique Maciel Sobral Timoteo

Professor: Dr. Carmello Bastos Filho

06 de Dezembro de 2012

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

A motivação deste estudo é aprender como desenvolver um algoritmo de inteligência de enxames, mais especificamente o *PSO* (do inglês *Particle Swarm Optimization*, Otimização por Enxame de Partículas), e adiquirir alguma sensibilidade quanto aos parâmetros da técnica. O objeto de estudo deste trabalho é o algorítmo clássico do PSO e entender quais impactos os parâmetros tem no desempenho da técnica, tais como: tamanho do enxame, importância do conhecimento cognitivo, importância do conhecimento social, a topologia de comunicação entre as partículas, entre outros.

Alguns experimentos serão montados e, se necessário, um estudo estatístico será efetuado para melhorar a compreensão do que os parâmetros podem ajudar ou atrapalhar no funcionamento do algoritmo.

1.1 Algoritmo PSO

O PSO é um algoritmo inspirado no voo de aves em busca de alimentos. Foi idealizado por Kennedy e Eberhart em 1995 [1]. Cada partícula é uma solução candidata para um problema e, por meio de troca de informações, o conjunto de partículas (o enxame) realiza, iterativamente, uma busca para encontrar uma "boa"solução para o problema. A solução é representada pelo posição da partícula no espaço de busca. Não é possível afirmar que a solução final é a melhor pois seria necessário uma exploração total do espaço de busca e um estudo de convergência para o algoritmo implementado.

Como há movimentação de partículas pelo espaço de busca, é clara a presença de uma velocidade e de uma posição. O algoritmo original do PSO apresenta duas equações que definem a atualização da velocidade e da posição iterativamente. Estas equações são, respectivamente

$$\vec{v}[t+1] = \omega\vec{v}[t] + C_1 r_1(\vec{p}_{Best} - \vec{x}[t]) + C_2 r_2(\vec{g}_{Best} - \vec{x}[t]) \tag{1.1}$$

e

$$\vec{x}[t+1] = \vec{x}[t] + \vec{v}[t+1] \tag{1.2}$$

onde $\vec{v}$ e $\vec{x}$ são, respectivamente, a velocidade e a posição da partícula. C_1 e C_2 são números tal que $C_1, C_2 \in \mathbb{R}$ e representam a importância que a partícula dá a, respectivamente, informação cognitiva e a informação do enxame. r_1 e r_2 são números uniformemente distribuídos no intervalo $[0, 1]$ e ω representa a inércia da partícula.

1.2 Objetivo da medição

Serão realizadas três análises do desempenho do algoritmo do PSO. A primeira análise é avaliar o desempenho do algoritmo PSO na versão básica; a segunda análise é avaliar a influência da topologia de comunicação; a terceira análise é avaliar a influência da contribuição inercial. A métrica de comparação será o melhor *fitness* por iteração (ou conjunto de iterações).

1.3 Objetivo do estudo

- Análise 1: Analisar o algoritmo PSO em sua versão básica

  30 partículas

  30 dimensões

  10000 iterações

  Inércia w = 0.8

  Coeficiente cognitivo e social c1 = c2 = 2.05

- Análise 2: Analisar a influência da topologia de comunicação.

  Topologia Global

  Topologia Local

  Topologia Focal

- Análise 3: Analisar a influência da contribuição inercial

  Fator de inércia com decaimento de 0.9 até 0.4

  Coeficiente de constricção de Clerc

- Funções para avaliação do desempenho:

  Griewank

  Rastringin

  Rosenbrock

  Schwefel

  Sphere

No segundo capítulo tem-se a descrição do planejamento dos experimentos assim como o levantamento de hipóteses. No Capítulo três apresenta-se como os testes serão realizados e os resultados comparados. No quarto e último capítulo apresentam-se os resultados obtidos e finaliza-se com as conclusões.

Capítulo 2

Planejamento

Neste capítulo descreve-se o planejamento que foi realizado para a realização dos experimentos.

2.1 Função para otimização

2.1.1 Griewank

A função Griewank é definida tal que

$$f(\vec{x}) = \sum_{i=1}^{n} x_i^2 \big/ 4000 - \prod_{i=1}^{n} \cos(x_i \big/ \sqrt{i}) + 1 - 180 \tag{2.1}$$

onde $x_i \in [0; 600]$ e n indica o número de dimensões que está se utilizando. Esta função possui mínimo global $f(\vec{x}) = 0$ quando $\vec{x} = \vec{0}$. A Figura 2.1 apresenta o formato da função quando $n = 2$

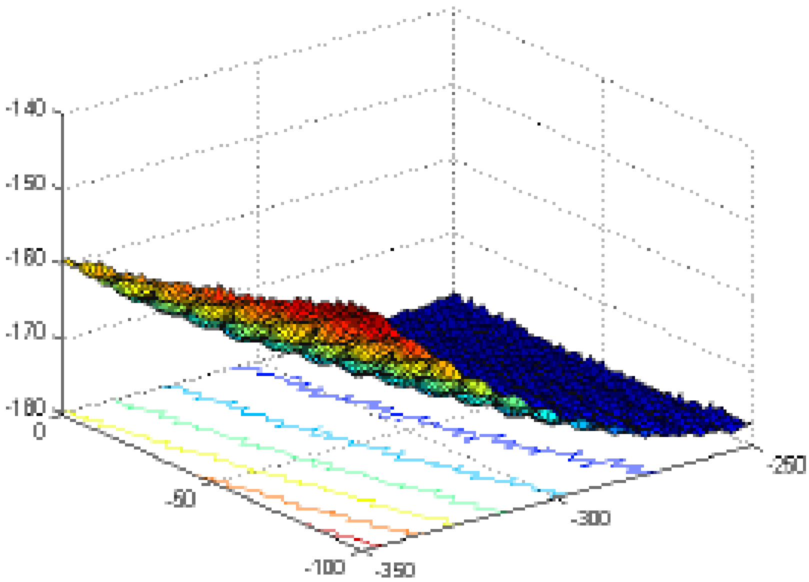

Figura 2.1: Função Griewank para n = 2.

Possuindo a equação (2.1) apenas mínimo, o objetivo de se utilizar a técnica de PSO será para minimizar, isto é, encontrar o valor mínimo da função Griewank.

2.1.2 Rastringin

A função Rastrigin é definida tal que

$$f(\vec{x}) = An + \sum_{i=1}^{n} \left[x_i^2 - A\cos(2\pi x_i)\right] \tag{2.2}$$

onde $A = 10$, $x_i \in [-5,12; 5,12]$ e n indica o número de dimensões que está se utilizando. Esta função possui mínimo global $f(\vec{x}) = 0$ quando $\vec{x} = \vec{0}$. A Figura 2.2 apresenta o formato da função quando $n = 2$

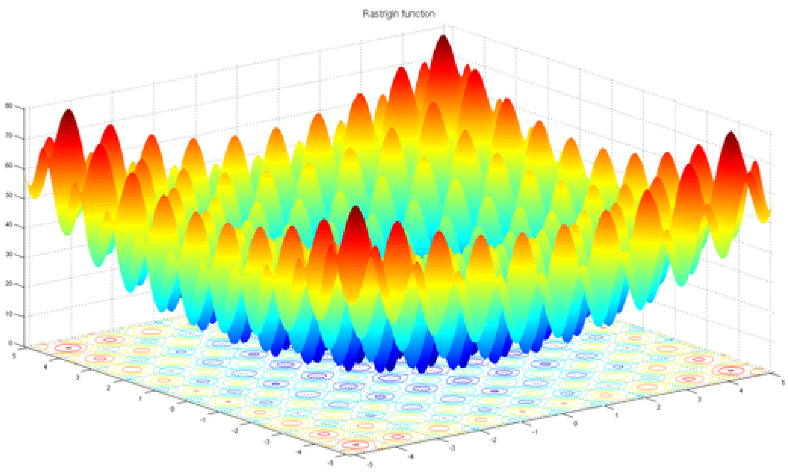

Figura 2.2: Função Rastrigin para n = 2.

Possuindo a equação (2.2) apenas mínimo, o objetivo de se utilizar a técnica de PSO será para minimizar, isto é, encontrar o valor mínimo da função Rastrigin.

2.1.3 Rosenbrock

A função Rosenbrock é definida tal que

$$f(\vec{x}) = \sum_{i=1}^{n} \left(100\left(x_i^2 - x_{i+1}\right)^2 + (x_i - 1)^2\right) + 390 \tag{2.3}$$

onde $x_i \in [-100; 100]$ e n indica o número de dimensões que está se utilizando. Esta função possui mínimo global $f(\vec{x}) = 0$ quando $\vec{x} = \vec{0}$. A Figura 2.3 apresenta o formato da função quando $n = 2$

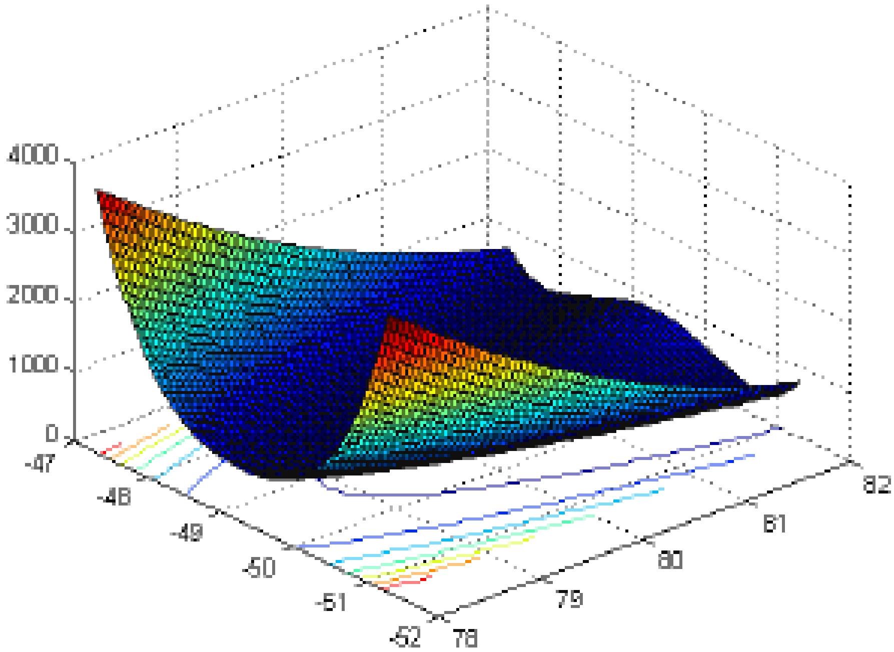

Figura 2.3: Função Rosenbrock para n = 2.

Possuindo a equação (2.3) apenas mínimo, o objetivo de se utilizar a técnica de PSO será para minimizar, isto é, encontrar o valor mínimo da função Rosenbrock.

2.1.4 Schwefel

A função Schwefel é definida tal que

$$f(\vec{x}) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_i^2 \right)^2 - 450 \qquad (2.4)$$

onde $x_i \in [-100; 100]$ e n indica o número de dimensões que está se utilizando. Esta função possui mínimo global $f(\vec{x}) = 0$ quando $\vec{x} = \vec{0}$. A Figura 2.4 apresenta o formato da função quando $n = 2$

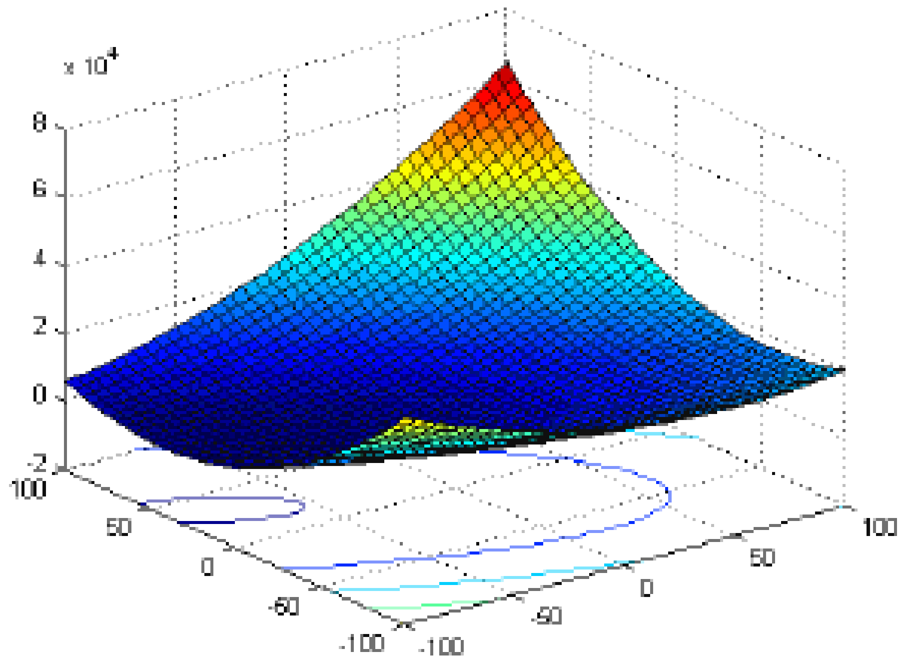

Figura 2.4: Função Schwefel para n = 2.

Possuindo a equação (2.4) apenas mínimo, o objetivo de se utilizar a técnica de PSO será para minimizar, isto é, encontrar o valor mínimo da função Schwefel.

2.1.5 Sphere

A função Sphere é definida tal que

$$f(\vec{x}) = \sum_{i=1}^{n} x_i^2 - 450 \tag{2.5}$$

onde $x_i \in [-100; 100]$ e n indica o número de dimensões que está se utilizando. Esta função possui mínimo global $f(\vec{x}) = 0$ quando $\vec{x} = \vec{0}$. A Figura 2.5 apresenta o formato da função quando $n = 2$

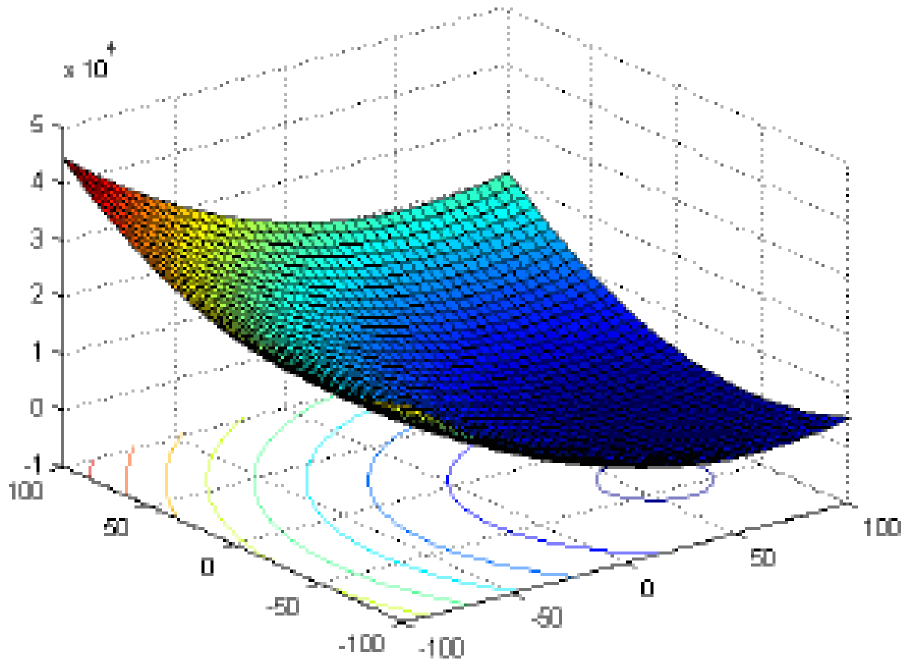

Figura 2.5: Função Sphere para n = 2.

Possuindo a equação (2.5) apenas mínimo, o objetivo de se utilizar a técnica de PSO será para minimizar, isto é, encontrar o valor mínimo da função Sphere.

2.2 Tabela de Experimentos

Os parâmetros que serão variados nos experimentos serão:

1. **Topologia** de comunicação do enxame
2. **Fator de Inércia** com decaimento
3. **Coeficiente de constricção** de Clerk
4. Função de **fitness**.

O critério de parada adotado para o algorítmo será o número de iterações que será fixado em 10000 (dez mil).

A Topologia do enxame será:

Tabela 2.1: Topologias de comunicação do enxame

Topologias do enxame
Global
Local
Focal

Já o coeficiente de inércia (ω) e o coeficiente do conhecimento cognitivo e social (respectivamente C_1 e C_2) serão testados de acordo com a seguinte tabela:

Tabela 2.2: Variação dos coeficientes

Inércia (ω)	C_1 e C_2
$\omega = 0,9 \rightarrow 0,4$ (Clerc 2002) [2]	$C_1 = C_2 = 2,05$ (constante) (Clerc 2002) [2]

As Funções de *fitness* serão:

Tabela 2.3: Tipos de funções de *fitness*

Funções
Griewank
Rastringin
Rosenbrock
Schwefel
Sphere

Portanto unindo as informações das Tabelas 2.1, 2.2 e 2.3 tem-se, no total, trinta configurações diferentes para realizar experimentos. Cada experimento será realizado trinta vezes e, então, serão comparados.

Capítulo 3

Operação

O experimento consiste de três análises. A primeira do desempenho do algoritmo em sua versão básica, a segunda da influência da topologia de comunicação e a terceira da influência da contribuição inercial para convergência do algorítmo PSO para cada caso de teste definido. Também será realizado o teste não paramétrico de Wilcoxon para comparar os casos em que uma simples inspeção visual não for suficiente para saber qual configuração foi melhor que a outra.

O tipo de erro utilizado no teste de hipótese não paramétrico de Wilcoxon é do Tipo II, o que significa que é o caso em que a hipótese nula é aceita, mesmo que realmente ela não seja verificada. Na prática, aceitam-se os falsos-positivos.

O mesmo computador será utilizado na realização de todos os testes, variando apenas o *seed* para o gerador pseudo-aleatório e os parâmetros do PSO. Como se trata do mesmo algoritmo sendo executado antes e depois de algumas mudanças, pode-se realizar o pareamento dos dados sem problemas.

O algoritmo programado foi escrito na linguagem de programação C: uma linguagem de nível intermediário, e otimizado para que o desempenho do programa fosse o melhor possível. O código-fonte escrito é bastante reduzido e modularizado. O nível de complexidade do programa na notação *Big-O* é $O(n^2)$. Para se executar o programa é necessário passar como parâmetro o perfil do teste a ser computado.

Como saída do programa, tem-se o melhor resultado já visitado pelo enxame para cada 500 iterações. A configuração do computador utilizado para realizar os experimentos é: Intel Core 2 Duo 2,00GHz, 2GB de memória RAM, sistema operacional Windows 7.

Cada configuração foi executada trinta vezes, no mesmo computador. Apesar do computador possuir mais de um núcleo, o programa desenvolvido não utilizou disso para acelerar seu processamento, executando assim o seu código em apenas um núcleo.

Capítulo 4

Resultados

Este capítulo apresenta os resultados obtidos para 9 das dezoito configurações investigadas. Nesse caso, utilizamos a função de *fitness* Rastringin e exibimos as primeiras iterações.

4.1 Resultados obtidos

As figuras a seguir são o *bloxplot* de cada configuração. A nomenclatura utilizada foi a seguinte: uma configuração **global-decay** significa:

- Topologia **global**
- $\omega = 0,9 \rightarrow 0,4$

Já uma configuração **local-clerc** significa:

- Topologia **local**
- Coeficiente de constricção de Clerc

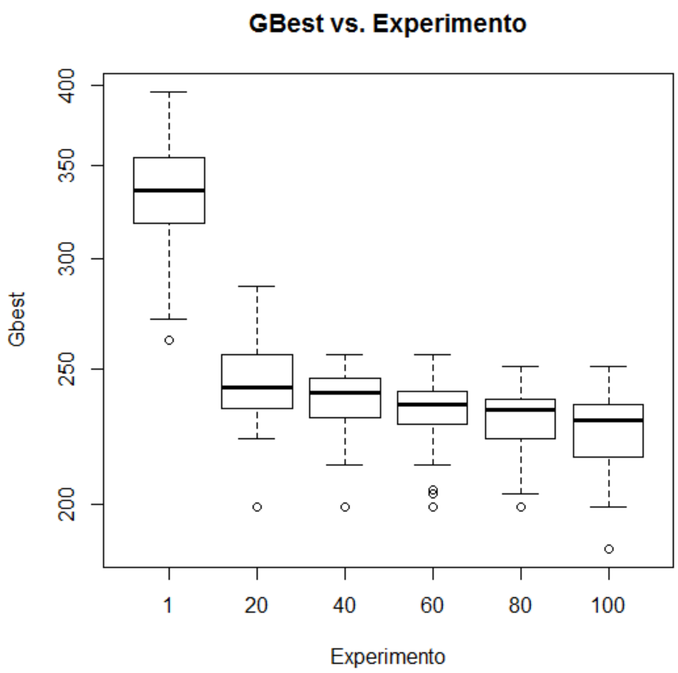

Figura 4.1: Boxplot para configuração focal-normal

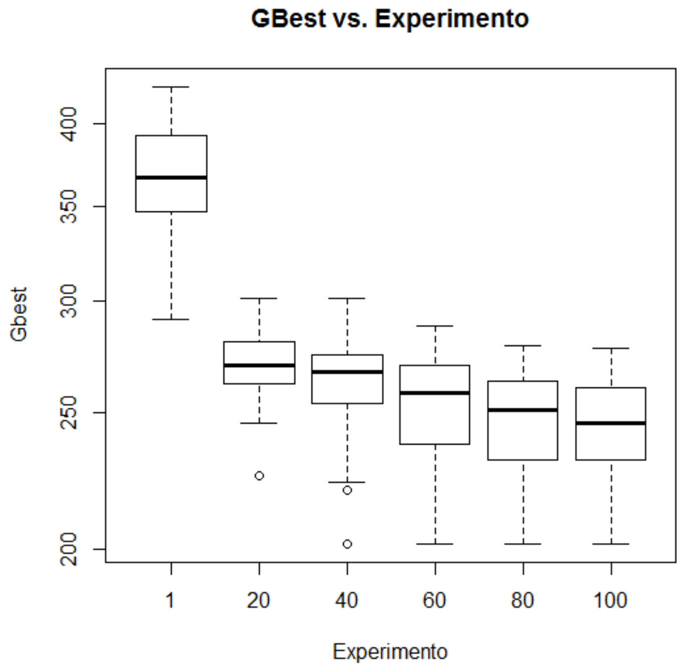

Figura 4.2: Boxplot para configuração local-normal

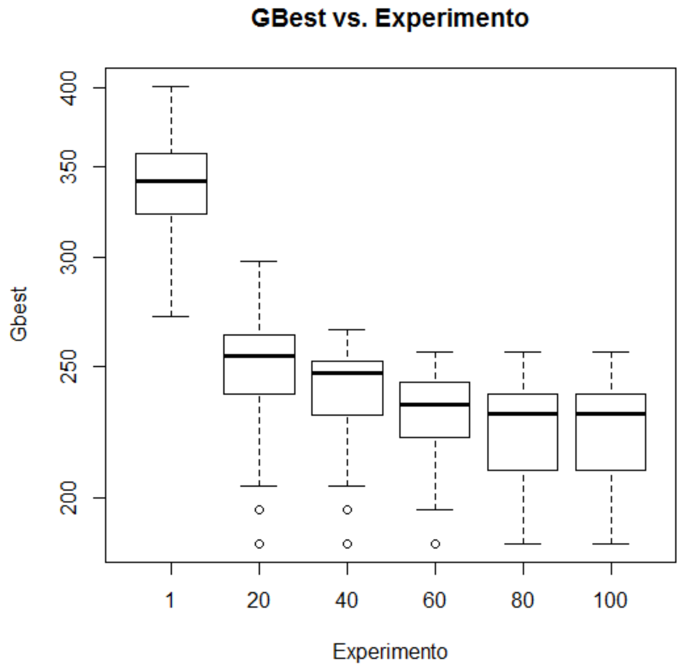

Figura 4.3: Boxplot para configuração global-normal

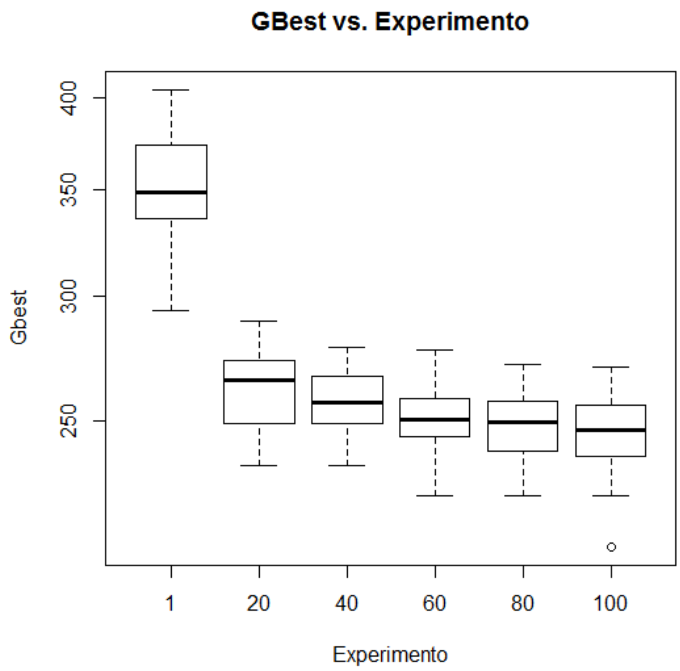

Figura 4.4: Boxplot para configuração focal-decay

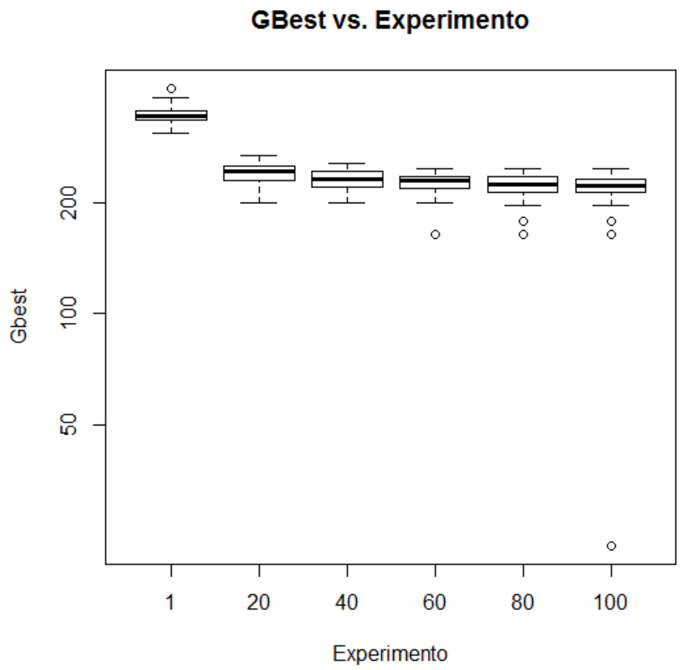

Figura 4.5: Boxplot para configuração focal-clerc

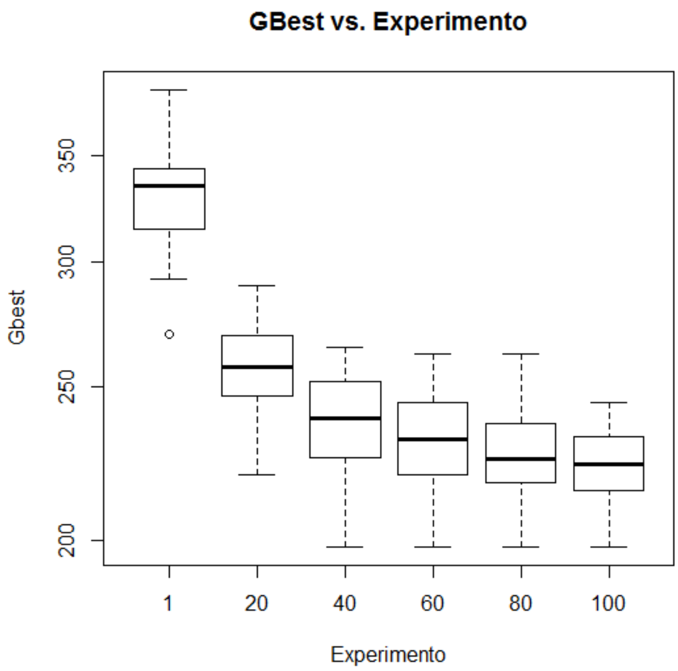

Figura 4.6: Boxplot para configuração global-decay

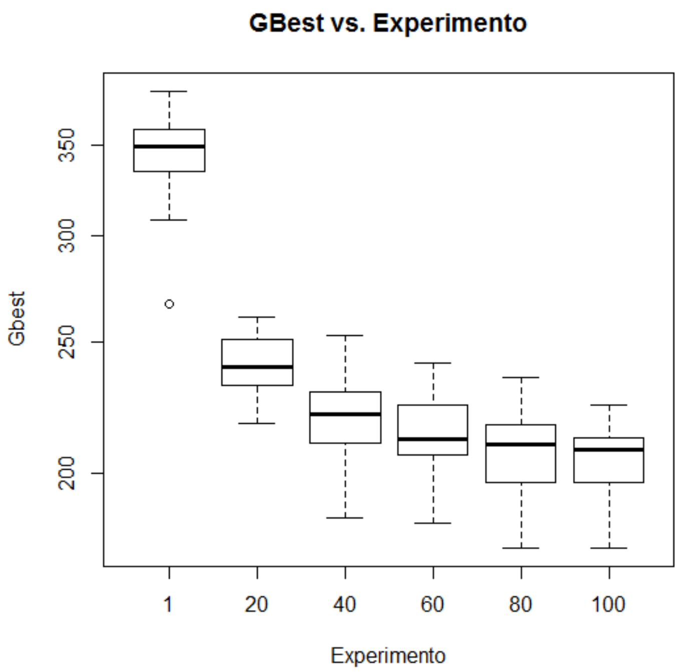

Figura 4.7: Boxplot para configuração global-clerc

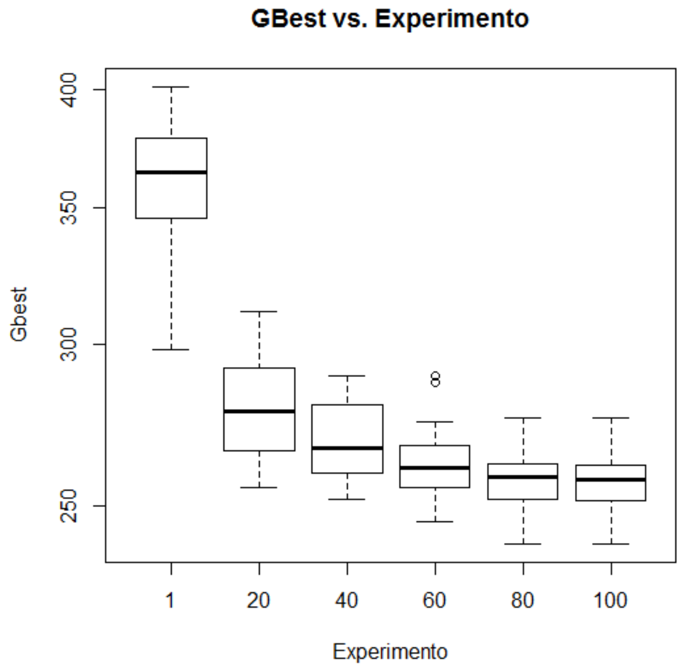

Figura 4.8: Boxplot para configuração local-decay

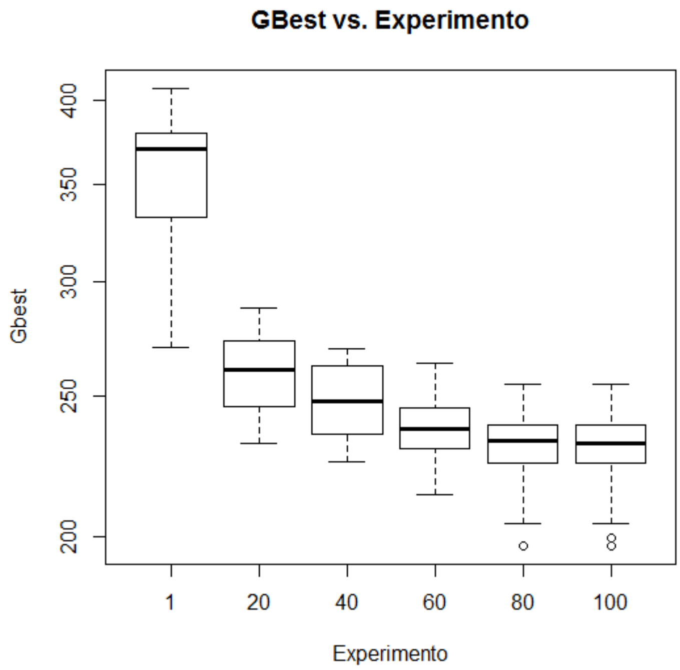

Figura 4.9: Boxplot para configuração local-clerc

4.2 Comparação de execução

Após estes experimentos serem executados foi realizado o teste de Wilcoxon Pareado com 95% de intervalo de confiança. O resultado está na Tabela 4.1.

Tabela 4.1: Teste de Wilcoxon Pareado para a funcao Rastringin

	Global	Focal	Local
Normal	-	P	P
Decaimento	M	-	P
Clerc	M	M	-

Capítulo 5

Conclusão

O algoritmo PSO é bastante utilizado para otimizar funções. Além do algoritmo clássico há variações da técnica. Para este trabalho realizou-se um estudo de convergência para dezoito configurações do PSO clássico. Observou-se que todas as configurações foram capazes de convergir apesar de fazê-lo de modo diferente.

Observou-se que a topologia global obteve um resultado igual ou melhor que a topologia local e que a contribuição inercial teve influência nos resultados.

Referências Bibliográficas

[1] R. Kennedy, J.; Eberhart. Particle swarm optimization. *Proceedings of IEEE International Conference on Neural Networks*, 1995.

[2] J. Clerc, M.; Kennedy. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 2002.