ESCOLA POLITÉCNICA
DE PERNAMBUCO

Departamento de Sistemas e Computação

PSS: UM SIMULADOR PARA OTIMIZAÇÃO POR ENXAME DE PARTÍCULAS

Trabalho de Conclusão de Curso

Engenharia da Computação

Emanoel Francisco Spósito Barreiros
Orientador: Prof. Carmelo José Albanez Bastos Filho

UNIVERSIDADE
DE PERNAMBUCO

EMANOEL FRANCISCO SPÓSITO BARREIROS

PSS: UM SIMULADOR PARA OTIMIZAÇÃO POR ENXAME DE PARTÍCULAS

Monografia apresentada como requisito parcial para obtenção do diploma de Bacharel em Engenharia da Computação pela Escola Politécnica de Pernambuco – Universidade de Pernambuco.

Recife, novembro de 2008.

Emanoel Francisco Spósito Barreiros

PSS: Um Simulador para Otimização por Enxame de Partículas

Dedico este trabalho à minha família, em especial a meus pais, Manoel e Rejane, que agora colhem os frutos da excelente educação que me proporcionaram.

Agradecimentos

Primeiramente agradeço a Deus pela força e perseverança com as quais me agraciou. Sem elas não conseguiria terminar esse curso.

Agradeço profundamente a todos que me ajudaram a concluir este trabalho, tanto direta quanto indiretamente. Primeiramente obrigado à minha família, em especial aos meus pais Manoel e Rejane, e meus irmãos Maurício e Manuela, que proporcionaram um ambiente maravilhoso e me ajudaram das mais diversas maneiras para que eu pudesse concluir minha graduação. Não poderia pedir família melhor. Muito obrigado, amo muito todos vocês! Obrigado a meus tios e tias pelo apoio incondicional em tudo que faço. Um grande beijo para meus avós maternos Romildo e Socorro, e paternos, Zito e Nininha. Vovô Zito, onde quer que o senhor esteja um grande abraço. Um “chero” especial para Helaine Lins, que tem me aturado por quase oito anos, em especial esses últimos meses que tive que me dedicar à conclusão deste trabalho. Te amo minha linda!

Obrigado aos professores do Departamento de Sistemas e Computação, que sem dúvida contribuíram para o que sou hoje como profissional e ser humano. Obrigado ao professor Sérgio Soares que me deu a oportunidade de desenvolver um trabalho de iniciação científica. Muito obrigado ao professor Carmelo pela excelente orientação que tive durante todo este trabalho. Obrigado a Danilo Carvalho pela orientação e grande esforço inicial na ferramenta. Obrigado a Marcel Caraciolo e Péricles Miranda pelas muitas madrugadas que dividimos para que este e outros trabalhos pudessem ser concluídos. Obrigado ao professor Alex Dias pela grande ajuda. Valeu galera!

Um grande abraço aos amigos que fiz durante os cinco anos que passei nesta Universidade. Os projetos e as noites em claro serão inesquecíveis!

Obrigado a todos vocês!

Resumo

A tarefa de encontrar soluções para problemas de otimização é, na maioria das vezes, bastante difícil. A meta-heurística de otimização por enxame de partículas tem ganho muita atenção da comunidade científica por sua efetividade. Por isso, figura como uma das mais utilizadas na categoria dos algoritmos evolutivos, mais especificamente algoritmos baseados em comportamentos sociais. A existência de uma ferramenta que facilite a realização de pesquisas e análise dos resultados de novas propostas é de grande valia. Este trabalho apresenta o PSS (*Particle Swarm Optimization Simulation Shell*), uma ferramenta de propósito geral para simulação de algoritmos de otimização por enxame de partículas. A referida ferramenta incorpora os mais importantes conceitos acerca do PSO, de forma que pode ser configurada para simular o comportamento das mais variadas instâncias do algoritmo. Incorpora conceitos de topologia de comunicação, mecanismos de atualização de velocidades, funções de teste e outras variações do PSO. Implementa ferramentas de análise de resultados, reforçando para a comunidade a necessidade da realização de análises estatísticas mais completas e confiáveis. As funcionalidades da ferramenta são testadas em dois estudos de caso, onde ela se mostrou bastante eficaz tanto na geração dos dados como em sua posterior análise.

Abstract

The task of finding solutions for optimization problems is, in most cases, very dificult. The meta-heuristic of particle swarm optimization has gained a lot of attention from the community and stands as one of the most used in the evolutive algorithms class, more specifically the social behavior based algorithms. The existence of a tool that facilitates research and analysis of new solutions results is very worthy. This work presents the PSS (Particle Swarm Optimization Simulation Shell), a general purpose tool for the simulation of particle swarm optimization algorithms. The refered tool incorporates the most important concepts of PSO, in such a way that it can be configured to simulate the behavior of many instances of the algorithm. Embodies the concepts of communication topology, velocity updating mechanisms, test functions and other PSO variations. The simulator implements tools for result analysis, reinforcing to the community the need for the use of a more complete and robust set of statistical analysis tools. Its functionalities are tested in two case studies, where it excels both in the generation of the simulation data and their further analysis.

Sumário

Índice de Figuras

Índice de Tabelas

Lista de Símbolos e Siglas

χ – Fator de Constrição.

API – *Application Programming Interface*

gbest – Global Best (melhor solução encontrada entre todas as partículas de um enxame).

MR – *Multi-Ring*

pbest – Particle Best (melhor solução encontrada pela partícula).

PSO – *Particle swarm optimization*

PSS – *Particle Swarm Optimization Simulation Shell*

vmax – Velocidade máxima que uma partícula poderá atingir em um determinado enxame.

Capítulo 1

Introdução

Encontrar soluções para problemas com múltiplas variáveis sempre foi uma terefa árdua. Por esse mesmo motivo existe uma atenção relevante da comunidade científica para desenvolver e aperfeiçoar algoritmos matemáticos complexos com o intuito de tentar resolver problemas de maneira precisa e rápida. No entanto, tais métodos geralmente tendem a ser computacionalmente caros.

Com o advento da computação, processos que antes demandavam várias horas de exaustivos e complexos cálculos matemáticos (e inevitavelmente suscetíveis a erros) poderiam agora ser automatizados. Quando os pesquisadores começaram a endereçar tais problemas, a estratégia foi simplesmente traduzir a linguagem matemática para a linguagem computacional sob a forma de programas.

Na maioria das vezes, o resultado era uma aplicação que resolvia o problema de forma ineficiente. No entanto, na época em que foram introduzidos, tais programas eram, simplesmente, o que poderia ser feito em termos computacionais. Mesmo que para encontrar a solução para os problemas propostos fosse necessário muito tempo, este processo era muitas vezes mais seguro em termos de precisão matemática e corretude.

Apesar da existência de métodos clássicos de otimização, sua complexidade e visíveis problemas de escalabilidade impulsionavam a comunidade na direção do desenvolvimento de novas alternativas. Em muitas situações não é necessária a obtenção da solução ótima para um determinado problema, é necessário apenas que a solução encontrada esteja dentro de uma margem aceitável de erro pré-estabelecida.

A classe de algoritmos evolutivos tem conseguido resultados bastante expressivos no cenário acima. Particularmente, algoritmos baseados em comportamentos sociais têm ganho especial atenção da comunidade [1][2][3]. Um desses algoritmos é conhecido como Otimização por Enxame de Particulas (PSO, *Particle Swarm Optimization*).

O PSO parte do princípio que um grupo de indivíduos consegue realizar uma busca muito mais abrangente em um determinado espaço do que apenas um único indivíduo. Inicialmente proposto por Kennedy e Eberhart [4], o algoritmo tenta modelar o comportamento de bandos de pássaros em busca de alimento. Basicamente, o algoritmo se resume à modelagem da atualização das posições e velocidades de cada indivíduo do enxame baseados em modelos pré-definidos de comunicação.

Melhoramentos no desempenho do PSO são alcançados realizando, por exemplo, mudanças na forma como as partículas se comunicam (novas topologias) ou como suas velocidades e posições são atualizadas. Tais modificações podem levar a variações muito pequenas no desempenho geral do algoritmo. Ao mesmo tempo, a escolha de uma determinada instância do PSO como solução de um problema real pode ser uma tarefa não trivial. Há ainda uma dificuldade adicional na implementação de uma solução que utilize o PSO, pois esta precisa ser testada e validada para que possa ser utilizada.

Atualmente o processo de coleta dos resultados das simulações e sua análise é penoso, levando, muitas vezes, mais tempo que todo o desenvolvimento da nova implementação. A existência de uma ferramenta que automatize todo este processo de relatórios e coleta de informação é muito importante, poupando muito tempo da equipe e tornando possível sua dedicação integral ao problema proposto.

A utilização de uma ferramenta que possa simular e analisar as mais variadas “configurações” do PSO passa a ser de grande valor. A ferramenta deve fornecer instrumentos estatísticos capazes de calcular desvio padrão, média e teste de significância estatística. Ainda, deve ser capaz de gerar gráficos demonstrativos da evolução do enxame, facilitando a comparação dos experimentos. Uma ferramenta com estas características é o objetivo deste trabalho.

Imagina-se que os relatórios e gráficos fornecidos pela ferramenta ajudarão bastante na análise de significância da configuração do algoritmo em um determinado problema. Atualmente, modificações no algoritmo podem gerar pequenas variações no resultado final (tempo de convergência, qualidade da solução gerada, etc.). A análise de significância estatística será bastante importante

pois vai auxiliar a análise de resultados, reduzindo assim a falsa impressão causada por simulações deficientes ou tendenciosas.

1.1 Objetivos

Este trabalho de conclusão de curso tem como objetivo apresentar uma ferramenta de simulações de algoritmos para otimização por enxame de partículas. Tal ferramenta foi denominada PSS (*Particle Swarm Optimization Simulation Shell*).

Ela fornecerá mecanismos para seleção de topologias, funções de otimização e algoritmos de atualização de posições. Além disso, a ferramenta também possui um mecanismo que permite ao usuário ativar ou desativar uma gama de variações do algoritmo do PSO (por exemplo, o conceito de *charged PSO* [5] ou *cooperative PSO* [6]).

A ferramenta também será capaz de realizar testes de desvio padrão, médias e teste de significância estatística de Mann-Whitney (conhecido também como Teste U) [7]. Com este conjunto de testes o usuário será capaz de analisar os resultados e demonstrar que uma dada configuração do algoritmo é indicada em uma determinada situação.

Ao fim de uma seqüência de simulações a ferramenta deverá ser capaz de gerar gráficos que demonstrem a evolução do enxame, com a opção de se usar escala logarítmica ou linear. Um dos modos de operação da ferramenta será o modo *batch*, em que o PSS deverá executar várias simulações em seqüência e armazenar os dados referentes a estas simulações. Ao fim, o simulador será capaz de gerar gráficos-caixa (também conhecido como *box plot* ou diagrama *Box-and-Whisker*) referente à evolução do enxame.

1.2 Estrutura do Trabalho

Este trabalho está organizado em capítulos, detalhados a seguir.

O capítulo 2 descreve a fundamentação teórica necessária para o entendimento deste trabalho. Será discutida a estrutura clássica do algoritmo de otimização por enxame de partículas. Ele discorre, ainda, sobre os mecanismos de

comunicação entre as partículas (topologias), algumas variações do PSO e equações de atualização de velocidade (clássica e variações).

O capítulo 3 enumera e detalha os requisitos para a ferramenta apresentada neste trabalho. Trata das funções de teste implementadas e suas diversas abordagens, e introduz o conceito de significância estatística e *box plot*.

O capítulo 4 apresenta a ferramenta objeto deste trabalho, pormenorizando toda a sua interface e como o usuário tem acesso às funcionalidades do simulador.

No capítulo 5 são realizados dois estudos de caso da ferramenta. São testadas a eficácia da topologia global em problemas unimodais e multimodais e investigado o impacto da utilização do fator de dispersão na topologia Multi-Ring

No capítulo 6 são apresentadas as conclusões, contribuições e são listados os trabalhos futuros.

Capítulo 2

Inteligência de Enxames e PSO

Neste capítulo são descritos os conceitos básicos sobre PSO, alguns conceitos sobre topologias e equações de atualização de velocidade.

2.1 Otimização por Enxame de Partículas

O embrião dos algoritmos de otimização por enxames de partícula surgiu da curiosidade que cientistas como Reynolds [8] tiveram em descobrir como os bandos de aves se comportavam, quais regras regiam seu movimento, como poderiam mudar de direção tão repentina e sincronizadamente. O PSO foi proposto após a simulação de modelos sociais simplificados baseados em observações feitas nos bandos de aves à procura de alimento. Em teoria, otimização por enxame de partículas tem fortes ligações com os conceitos de algoritmos genéticos e programação evolucionária.

Um enxame pode ser definido com um conjunto de indivíduos que interagem localmente entre si, regidos por um comportamento global, buscando a solução para problemas de forma distribuída [9]. O sociobiólogo E. O. Wilson [10] escreveu:

> Pelo menos em teoria, membros de um grupo podem se beneficiar das descobertas e experiência anteriores de todos os outros membros do grupo durante a busca por alimento. Esta vantagem pode se tornar decisiva, sobrepondo as desvantagens da competição por alimento quando o recurso está imprevisivelmente distribuído em pacotes.

A otimização por enxame de partículas foi fortemente inspirada nesta declaração.

Atualmente, na maioria das implementações do PSO, as partículas movem-se no espaço de busca tendendo para uma combinação entre a melhor posição encontrada pela partícula e a melhor posição encontrada pela vizinhança em que ela está inserida. Esta vizinhança é definida como um conjunto de partículas com as quais a partícula em análise pode se comunicar, podendo este conjunto se estender para todo o enxame ou não (mais informações sobre topologias serão apresentadas

mais adiante neste capítulo). Os primeiros modelos implementavam a vizinhança através da avaliação de uma função Euclidiana. As partículas que estivessem "próximas o suficiente" fariam parte do subconjunto que influenciaria outra determinada partícula. No entanto, este modelo logo foi abandonado por ser extremamente custoso computacionalmente. O enfoque nos modelos biológicos foi logo redirecionado para modelos de otimização matemática.

No momento da inicialização do algoritmo, as partículas são aleatoriamente inicializadas (vetores de velocidade de posição). Cada partícula *i* é representada por três vetores:

a) sua posição num espaço de busca *D*-dimensional: $\overrightarrow{x_i} = (x_{i1}, x_{i2}, \dots, x_{iD})$;

b) a melhor posição que ela encontrou: $\overrightarrow{p_i} = (p_{i1}, p_{i2}, \dots, p_{iD})$;

c) a sua velocidade atual: $\overrightarrow{v_i} = (v_{i1}, v_{i2}, \dots, v_{iD})$.

As partículas então se movem no espaço de busca à procura da melhor solução possível. A cada unidade de tempo o algoritmo atualiza as posições e velocidades das partículas usando as equações (1) e (2).

$$v_{id} = v_{id} + c_1\epsilon_1(p_{id} - x_{id}) + c_2\epsilon_2(p_{gd} - x_{id}) \,, \tag{1}$$

$$x_{id} = x_{id} + v_{id} \,. \tag{2}$$

No algoritmo original, c_1 e c_2 são constantes com o valor 2, ϵ_1 e ϵ_2 são números aleatórios independentes gerados a cada iteração para cada partícula e cada dimensão de 1 a *D*. O vetor $\overrightarrow{p_g}$ é a melhor posição encontrada pela melhor partícula no enxame.

O algoritmo PSO clássico define que deve existir uma condição de parada para o processo, por exemplo, pode ser determinada a quantidade de iterações que o algoritmo deve executar ou que o enxame deve procurar soluções até que seu desempenho não tenha melhora significativa. Também deve ser definida a quantidade de partículas no enxame. Para cada intervalo *t* e para cada partícula *i* do enxame, o algoritmo deve avaliar as equações (1) e (2) e atualizar o vetor de posição $\overrightarrow{x_i}$ da partícula *i* com os valores encontrados, calcular o *fitness*

(desempenho) da partícula *i* e atualizar os valores de $\overrightarrow{p_g}$ e $\overrightarrow{p_i}$. O processo pode ser mais facilmente visualizado na Figura 1.

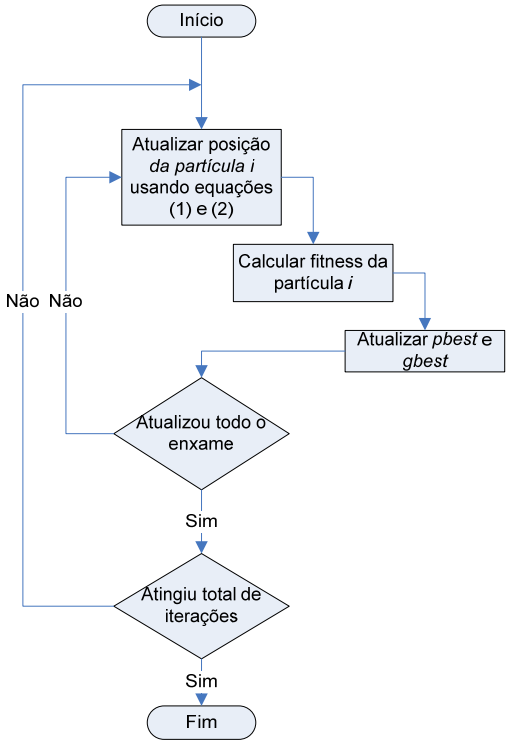


Figura 1. Algoritmo clássico do PSO.

Cada partícula está sob influência de três forças que podem ser representadas matematicamente como vetores, a saber:

a) vetor inércia: representa o movimento atual da partícula, ou seja, a velocidade corrente que impulsiona a partícula para a região onde ela "acredita" estarem as soluções. Na equação (1) é mapeada para $\overrightarrow{v_i}$;

b) vetor memória: representa a componente cognitiva da partícula, uma relação entre a posição atual e a melhor posição encontrada por aquela partícula. Na equação (1) é representada pelo termo $c_1\epsilon_1(\overrightarrow{p_i} - \overrightarrow{x_i})$;

c) vetor cooperação: representa a influência do enxame em uma determinada partícula. É uma relação entre a melhor posição encontrada pelo enxame e a posição atual da partícula. Na equação 1 é representada pelo termo $c_2\epsilon_2(\overrightarrow{p_g} - \overrightarrow{x_i})$.

Um esquema gráfico poder ser observado na Figura 2, onde $\overrightarrow{x_i}$ representa a posição atual da partícula e $\overrightarrow{x_{i+1}}$ representa a posição da partícula após o procesos de atualização.

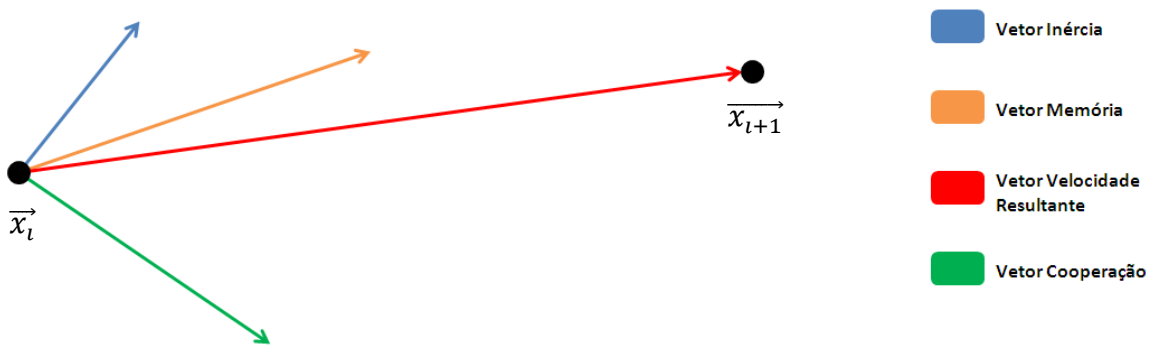


Figura 2. Vetores que influenciam o movimento da partícula.

2.2 Peso de Inércia e Fator de Constrição

Um fenômeno freqüentemente observado nos enxames que utilizavam o algoritmo clássico é a “explosão” de velocidades. Facilmente uma partícula adquiria uma velocidade muito grande muito rapidamente, o que a levava a oscilar dentro do espaço de busca.

Foi proposto por Eberhart e Kennedy [11] um mecanismo que estabelece um limite *vmax* para a velocidade das partículas. Foi observado no entanto, que a determinação do valor *vmax* não era nada trivial, e a escolha errada para este valor poderia implicar em uma perda de desempenho ou taxas de erro inadmissíveis. Por exemplo, espaços de busca maiores demandavam valores maiores de *vmax* para garantir a exploração e espaços de busca menores exigiam menores valores para evitar o problema da “explosão” de velocidades.

O peso de inércia e fator de constrição [12] foram introduzidos com a intenção de se remover completamente o conceito de velocidade limite. A idéia era utilizar a velocidade anterior da partícula como parâmetro para atualização de sua velocidade. A equação de velocidade pode ser reescrita na forma da equação (3) para comportar o conceito de peso de inércia (representado por ω):

$$v_{id} = \omega v_{id} + c_1\epsilon_1(p_{id} - x_{id}) + c_2\epsilon_2(p_{gd} - x_{id}) \; . \qquad (3)$$

Os autores sugerem que o valor de ω seja inicializado em 1, para favorecer o comportamento exploratório inicial da partícula. Durante a execução do algoritmo esse valor pode ser reduzido para que a partícula realize uma busca mais refinada nos arredores da posição onde ela atualmente se encontra, pois assume-se que a partícula já encontrou uma região onde a probabilidade de se encontrar uma solução satisfatória é grande.

O fator de constrição, conceito similar ao peso de inércia, consiste na introdução de um novo parâmetro χ, derivado das constantes existentes na equação de velocidade. O parâmetro χ é calculado de acordo com a equação 4.

$$\chi = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|}, \varphi = c_1 + c_2 \,. \tag{4}$$

Foi demonstrado por Clerc e Kennedy [12] que para valores de c_1 e c_2 tal que $\varphi < 4$, o enxame converge lenta e espiralmente para a solução. Ao passo que quando $\varphi > 4$ a convergência é rápida e garantida. Por simplicidade, assume-se o valor de $\varphi = 4{,}1$ e conseqüentemente c_1 e c_2 iguais a 2,05 para assegurar a convergência. Para aplicar o fator de constrição à equação de atualização de velocidade, esta precisa ser reescrita na forma da equação 5.

$$v_{id} = \chi\left(v_{id} + c_1\epsilon_1(p_{id} - x_{id}) + c_2\epsilon_2\left(p_{gd} - x_{id}\right)\right). \tag{5}$$

Outro fator muito importante e responsável por grande parte do sucesso ou fracasso de uma determinada configuração do enxame é a forma como as partículas se comunicam. A seção 2.3 trata do conceito de topologia.

2.3 Topologias

Algoritmos inteligentes baseados em enxames, e conseqüentemente semelhantes ao PSO, são algoritmos que tentam mapear comportamentos sociais em um ambiente computacional controlado. As partículas que compõem o enxame precisam, de alguma forma, propagar as informações que conseguem coletar, caso contrário, os referidos algoritmos não poderia se apoiar no conceito de sociedade, pois as partículas estariam “voando” pelo espaço de busca à procura de soluções sendo influenciadas apenas por sua própria experiência.

Neste cenário, as topologias, que definem as regras de como as partículas devem se comunicar, desempenham um papel importantíssimo, influenciando completamente o resultado da execução. Na maioria das vezes significa o sucesso ou o fracasso de uma determinada instância do algoritmo.

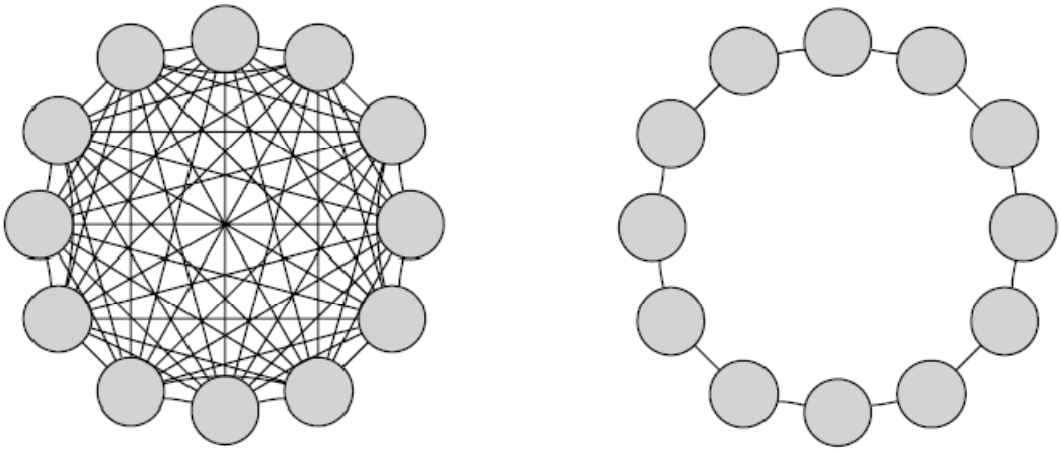
2.3.1 As Topologias Local e Global

As topologias mais conhecidas e utilizadas são as topologias local (ou em anel) e global [4]. A topologia global foi a primeira a ser proposta. Ela diz que cada partícula está conectada com qualquer outra partícula do enxame. Conseqüentemente, uma partícula é influenciada por todas as outras partículas, pois estaria recebendo informações de todo o enxame (modelo *gbest*). Isto apresenta grandes vantagens quando o problema é sabidamente unimodal, pois as partículas tenderão rapidamente para a única solução.

A topologia em anel (ou local) é considerada a variação mais importante do algoritmo de PSO original, onde cada partícula se comunica apenas com seus vizinhos diretos. Uma influência direta desta diferença entre os modelos de comunicação é responsável por fazer enxames com a topologia global terem um desempenho, na maioria das vezes, superior aos enxames com topologia em anel em problemas unimodais (problemas com apenas uma solução), enquanto que problemas multimodais (problemas com várias soluções ótimas ou sub-ótimas) são melhores tratados com o uso de enxames com topologia em anel. Isso se deve ao fato de enxames com topologia em anel explorarem melhor o ambiente, não atraindo todas as partículas para uma solução sub-ótima, o que é bastante comum em problemas multimodais.

Vale salientar que na maioria das aplicações reais do PSO não se conhece bem o problema a ser resolvido, muito menos sabe-se quantas soluções satisfatórias ele pode ter. Conseqüentemente, utilizar a topologia global pode ser muito arriscado se são necessárias soluções mais próximas da ótima. No entanto, aplicar a topologia em anel geralmente leva a uma convergência mais lenta, pois num enxame de N partículas, uma partícula pode ter de esperar de uma a $N/2$ iterações para indiretamente receber informações da melhor partícula do enxame, enquanto que na topologia global, após a primeira iteração todas as partículas já têm conhecimento

da melhor solução do bando. Na Figura 3 o leitor pode ter uma idéia da comunicação entre as partículas nas topologias global e em anel.



(a) (b)

Figura 3. Topologias (a) global e (b) anel (extraído de [10]).

2.3.2 A Topologia Focal

A topologia focal [13] introduz o conceito de partícula focal. Esta partícula desempenha um papel especial no enxame. Ela está conectada a todas as outras partículas no enxame, enquanto que todas as partículas não-focais estão ligadas apenas à partícula focal. A referida partícula funciona como uma mediadora, arbitrando a cada iteração se as informações passadas a ela pelas partículas não-focais devem ou não ser propagadas para todo o enxame.

De maneira mais simples, a partícula focal só atualizará sua posição no espaço se esta atualização acarretar melhora no seu desempenho (melhor *fitness*). Este pequeno detalhe faz toda a diferença, pois se a partícula focal simplesmente repassasse a informação da melhor partícula para todo o enxame, estaria-se falando da topologia global, ou seja, neste caso, um enxame com N partículas e topologia focal seria equivalente a um enxame com $N - 1$ partículas com topologia global. Na Figura 4 o leitor pode verificar graficamente a topologia focal.

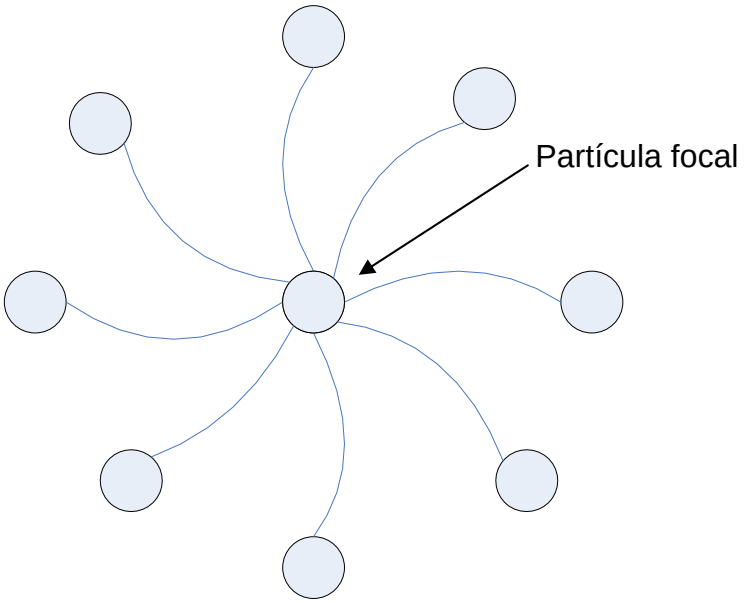


Figura 4. Topologia focal.

2.3.3 A Topologia Hierárquica

A topologia hierárquica [14] estrutura as partículas em uma árvore. Neste modelo, cada partícula é influenciada apenas por sua própria experiência e pela informação da partícula imediatamente superior a ela na hierarquia. Por definição, *h* é a altura da árvore, *d* a sua ordem e *m* a quantidade total de nós na árvore.

O algoritmo básico de atualização da árvore na topologia hierárquica pode ser observado na Figura 6. A cada iteração Um fato interessante que pode ser observado no algoritmo mostrado é que uma partícula pode subir apenas um nível na hierarquia a cada iteração e pode chegar ao topo em no máximo em *h – 1* iterações, a não ser que uma solução melhor tenha sido encontrada neste meio tempo. No entanto, um nó pode descer até o último nível mesmo na primeira iteração. O processo demonstrado na Figura 6 ocorre após a avaliação da função objetivo, mas antes da atualização da velocidade e posição de cada partícula do enxame.

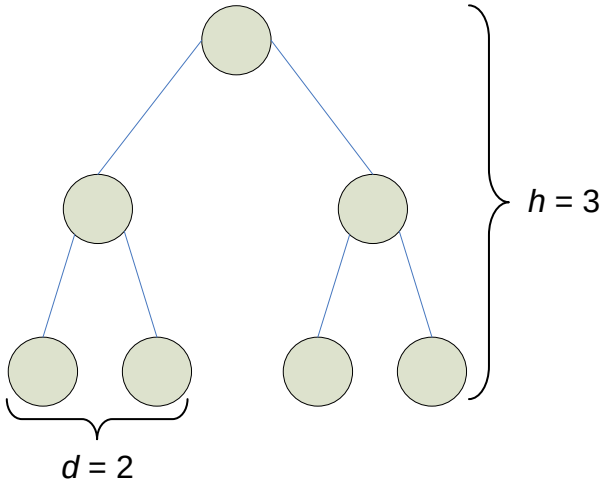


Figura 5. Topologia hierárquica com *m* = 7.

A cada iteração o algoritmo seleciona na árvore, utilizando o caminhamento em largura, todos os nós internos (um por vez). Para cada partícula selecionada, o algoritmo verifica se seus filhos têm desempenho melhor. Em caso positivo, é realizada a troca do pai com o filho corrente, de forma que ao fim deste passo, o lugar antes ocupado pelo pai será agora ocupado pela partícula com o melhor desempenho entre o pai e seus filhos.

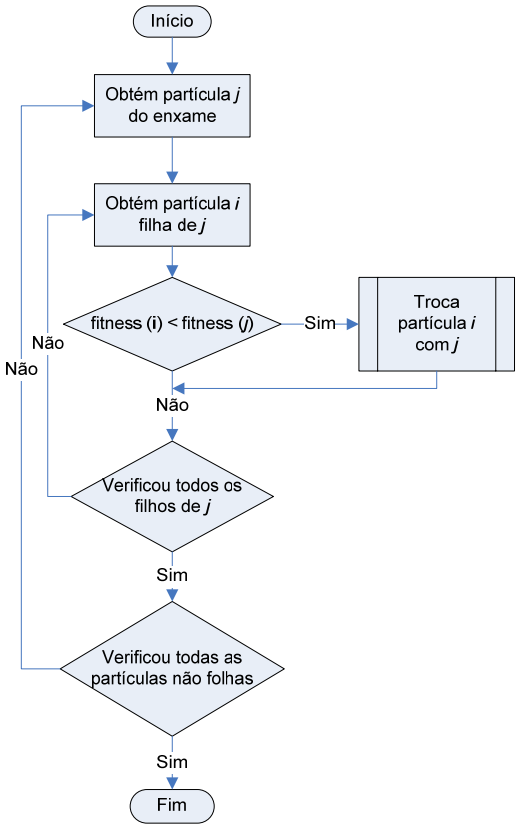


Figura 6. Algoritmo de atualização da árvore na topologia Hierárquica.

2.3.4 As Topologias Von Neumann e *Multi-Ring*

Outra topologia bastante conhecida é a Von Neumann [15]. O esquema gráfico desta topologia pode ser comparado a uma malha, onde a vizinhança de cada partícula é estática, com os quatro vizinhos diretos (superior, inferior, esquerdo e direito). O grafo formado geralmente é fechado, de forma que as partículas de uma extremidade se comunicam com as partículas da extremidade oposta. O leitor pode entender melhor a topologia Von Neumann na Figura 7.

Existem também um conjunto de topologias mistas, que reutilizam alguns dos conceitos discutidos anteriormente mas modelam novos comportamentos. Um exemplo desta abordagem é a topologia *Multi-Ring* (MR) [16]. Nesta topologia, múltiplos anéis coexistem e se comunicam no espaço de busca de acordo com algumas regras. Cada anel conduz a sua evolução independentemente, no entanto, as partículas, além de se comunicarem com seus vizinhos diretos no anel também se comunicam com seus vizinhos diretos nos anéis imediatamente superior e inferior, como pode ser visto na Figura 8, ou seja, uma partícula que pertença a uma camada chamada $rl_{(k)}$ troca informações com as camadas $rl_{(k-1)}$ e $rl_{(k+1)}$, de forma que uma partícula possa tomar proveito da informação social gerada pelas outras camadas.

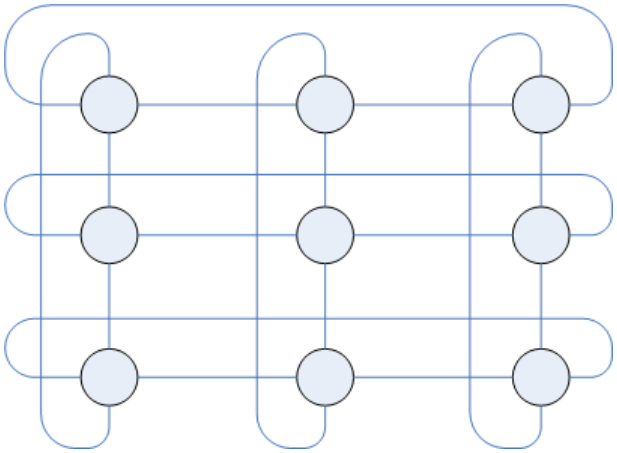


Figura 7. Topologia Von Neumann.

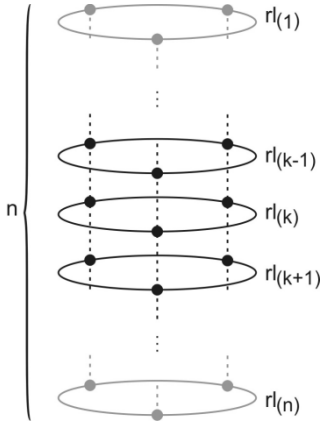


Figura 8. Topologia *Multi-Ring* (extraído de [16]).

É fácil de se observar que caso as partículas permanecessem estáticas, a topologia *Multi-Ring* se resumiria à topologia Von Neumann. Os anéis da topologia *Multi-Ring*, no entanto, têm a habilidade de rotacionar caso o desempenho do anel não seja satisfatório. Caso o anel (camada) $rl_{(k)}$ não consiga melhorar o seu desempenho por uma determinada quantidade de iterações, ele será rotacionado.O grau da rotação é determinado pela equação 6, onde i é o novo índice da partícula, d é a distância da rotação e nl é a quantidade de partículas no anel.

$$i = (i + d)\ mod\ (nl)\ . \tag{6}$$

Logo, a nova vizinhança da partícula *i* será composta por { $rl_{(k)(i-1)}$, $rl_{(k)(i+1)}$, $rl_{(k+1)(i+d)}$, $rl_{(k-1)(i+d)}$}.

2.3.5 A Topologia *Four-Clusters*

A topologia *Four-Clusters* [17] consiste em ligar grupos de partículas. Ela é inspirada por pequenas comunidades que se comunicam através de informantes. A quantidade de informantes em cada agrupamento é exatamente a quantidade de agrupamentos de todo o enxame menos um, ou seja, quantidade de agrupamentos que com os quais um determinado agrupamento deve se comunicar.

A Figura 9 demonstra graficamente a topologia *Four-Clusters*. De acordo com a definição da topologia, os agrupamentos marcados com “A”, “B”, “C” e “D” são os chamados *clusters*. As partículas realçadas em cinza são os informantes, que se comunicam com os demais *clusters*.

Os informantes de um *cluster* são estáticos, ou seja, permanecem os mesmos durante toda a execução do algoritmo. Utilizando essa estrutura, a informação flui de um informante diretamente para outro. Uma coisa que pode ser notada, é que uma informação boa não será diretamente transmitida para outros *clusters*, o que faz com que algumas iterações sejam “gastas” em vão.

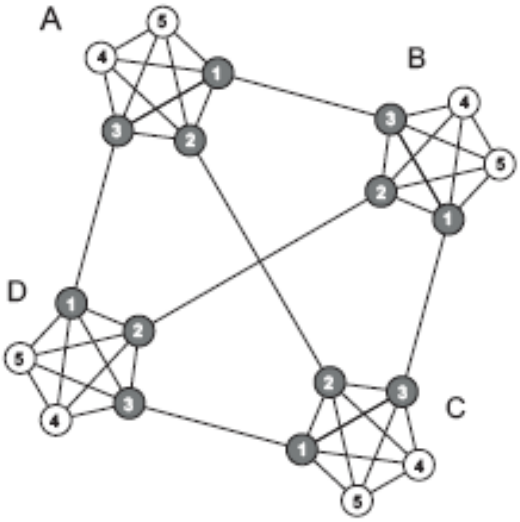


Figura 9. A topologia *Four-Clusters* (extraído de [17]).

2.3.6 A Topologia *Clan*

Clãs são grupos de indivíduos que de alguma forma se relacionam através de ma característica comum a todos os integrantes do clã. Internamente ao clã há um processo que tenta determinar seu líder, que intuitivamente todas as outras partículas tendem a seguir. O significado social do clã pode ser visto como uma pequena parte de uma sociedade maior.

A topologia define que os clãs devem ser comunicar internamente seguindo a topologia Global [4]. A Figura 10 demonstra graficamente os clãs da topolgia. A cada iteração, cada clã procura pela solução e marca a partícula que obteve o melhor desempenho.

A delegação do líder representa a definição da partícula que terá o poder de liderar as outras partículas em direção à solução. Após a execução e a delegação do líder, o enxame pode ficar semelhante ao que é demonstrado na Figura 10.

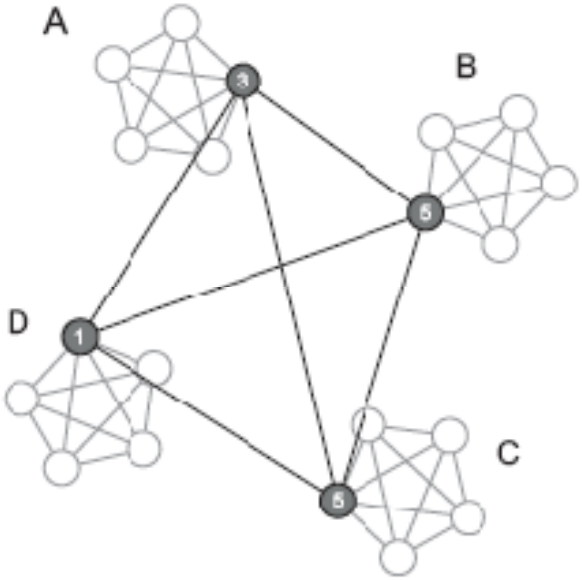


Figura 10. A topologia *Clan* e a conferência de líderes (extraído de [17]).

Os líderes que foram selecionados no passo anterior deverão agora formar um novo enxame, que executará uma nova busca, levando em consideração apenas a informação gerada pelos líderes. A conferência pode ser realizada tanto utilizando a topologia Local quanto a topologia Global.

Uma vez que os líderes tiveram suas posições e velocidades atualizadas, eles devem "retornar" ao seu clã de origem e propagar a informação adquirida. A informação então é diretamente transmitida para cada partícula no clã, logo, todas as partículas de todos os clãs terão sido indiretamente influenciadas pela melhor partícula do enxame.

2.3.7 A Topologia *Random*

A topologia *Random* [18] (aleatória) prega que a vizinhança de uma determinada partícula deve ser aleatória. O autor ressalta que a partícula deve influenciar a si própria, pois toda estratégia em que isso não ocorre não tem um bom desempenho.

Definindo o enxame como $S = \{ P_1, P_2, ..., P_S \}$, toda partícula P_i possui K informantes, incluindo si própria. Na topologia Local, por exemplo, a vizinhaça tem tamanho $K = 3$. A escolhad a vizinhança da partícula P_i deve seguir as seguintes regras:

a) a vizinhança contém P_i;

b) as outras K – 1 partículas são escolhidas aleatoriamente de S.

O leitor pode estar se perguntando: se a vizinhança de P_i já possui a própria partícula P_i porque deve-se sortear sua vizinhaça de S (que contém P_i) e não de S – { P_i }? Em alguns casos é preferível que a partícula não possua qualquer outra influência a não ser dela própria, o que permite que a partícula realize uma busca em profundidade, explorando melhor a região em que ela atualmente se encontra. Escolhendo a vizinhança de S, a probabilidade de P_i não possuir nenhuma influência é de 1 / S^{K-1}. Escolhendo os vizinhos de S – { P_i } essa possibilidade não existe.

A topologia funciona da seguinte maneira:

1. construir uma matriz S x S chamada L. Imediatamente, a diagonal principal da matriz é inicializada em 1, ou seja, todas as partículas são influenciadas por ela própria;
2. para cada linha i sortea-se K números k_1,k_2, ..., k_K aleatoriamente e inicializa $L(i,k_n) = 1$. Há a possibilidade de que todos os elementos sorteados sejam o mesmo elemento.
3. Considerando cada coluna j, se $L(i, j) = 1$, significa que a partícula P_i informa a partícula P_j.

A topologia dispõe de um mecanismo que modifica a vizinhança caso a melhor partícula do enxame não tem melhora após toda uma rodada de avaliações (S avaliações). Outra possibilidade é que a vizinhança seja reinicializada após uma quantidade de iterações. Tomando como base o modelo espalhamento de informação, espera-se uma quantidade de iterações de forma que uma partícula em um extremo do enxame tenha a oportunidade de influenciar uma outra partícula na extremidade oposta do enxame. Como esta estratégia é bastante custosa computacionalmente, reorganiza-se o enxame após N / 2 iterações.

2.4 Variações do PSO

Esta seção apresenta algumas variações do algoritmo do PSO implementadas na ferramenta.

2.4.1 *Charged* PSO

O modelo *Charged* PSO [20] utiliza o conceito de cargas eletrostáticas aplicado às partículas. Neste modelo, cada partícula i possui uma carga Q_i, que estimula um comportamento repulsivo entre as partículas carregadas. Os autores argumentam que os problemas reais são raramente estáticos como os que encontramos na maioria dos trabalhos publicados na área. Por este motivo, os algoritmos atuais não têm um bom desempenho em ambientes de busca dinâmicos, onde, por exemplo, a solução ótima muda de tempos em tempos.

Os autores argumentam ainda, que a configuração ideal do novo algoritmo não é composta apenas de partículas carregadas, mas também de partículas neutras, que correspondem às partículas como hoje conhecemos. As forças de repulsão causadas pelas partícula carregadas não têm efeito sobre as partículas neutras, o que faz com que elas procurem por soluções livremente. Isto também contribui para equilibrar os níveis de busca em amplitude e profundidade do enxame, pois em um problema dinâmico, não é possível identificar quando um é mais necessário que o outro.

No algoritmo PSO clássico mais comumente usado, a velocidade e posição das partículas são atualizadas pelas equações (3) e (2) respectivamente. Para introduzir a idéia de carga, e conseqüentemente evitar colisões, é necessário modificar a equação (3) através da adição de um novo termo de aceleração, descrito pela equação (7),

$$a_i = \sum_{j \neq i} \frac{Q_i Q_j}{r_{ij}^3} \boldsymbol{r}_{ij}, \;\; p_{core} < r_{ij} < p \;, \tag{7}$$

onde $\mathbf{r}_{ij} = \mathrm{x}_i - \mathrm{x}_j$ e $\mathrm{r}_{ij} = |\mathrm{x}_i - \mathrm{x}_j|$. x_i e x_j são vetores que representam as posições de duas partículas *i* e *j*, a_i representa o vetor aceleração da partícula *i*. A equação (3) deve então ser reescrita para incorporar a aceleração da partícula, dando origem à equação (8).

$$v_{id} = \omega v_{id} + c_1 \epsilon_1 (p_{id} - x_{id}) + c_2 \epsilon_2 (p_{gd} - x_{id}) + a_i \;. \tag{8}$$

2.4.2 *Fully Informed* PSO

O modelo *Fully Informed* [21] define que uma determinada partícula deve ser influenciada por todas as partículas que compõem sua vizinhança. No modelo tradicional do PSO, o algoritmo deve escolher dentre as partículas da vizinhança uma que tenha o melhor desempenho e eliminar todas as outras. A quantidade de vizinhos afetará o quão diversa será a influência em uma partícula. No contexto de algoritmos de otimização, muitas influências pode significar que a busca é prejudicada, ao invés de melhorada.

Segundo Mendes *et al* [21], o ponto de partida para o novo modelo foi a equação de atualização de valocidade utilizando o fator de constrição. A equação (9) é uma variação da equação (5), onde χ é o fator de constrição calculado pela equação (4), $\vec{v}_t$ e $\vec{v}_{t+1}$ são as velocidades atual e no instante t e $(t+1)$ da partícula respectivamente, $\vec{X}_t$ é o vetor de posição da partícula no instante t, $\vec{P}_i$ e $\vec{P}_g$ são a melhor posição encontrada pela partícula i e a melhor posição encontrada pela melhor partícula da vizinhança de i.

$$\vec{v}_{t+1} = \chi \left(\vec{v}_t + \varphi\left(\vec{P}_m - \vec{X}_t\right)\right), \tag{9}$$

$$\vec{P}_m = \frac{\varphi_1 \vec{P}_i + \varphi_2 \vec{P}_g}{\varphi_1 + \varphi_2}. \tag{10}$$

Os autores então propõem um modo alternativo para o cálculo de $\vec{P}_m$ visto na equação (11), onde $\vec{\mathrm{U}}[min, max]$ representa uma função que é capaz de gerar um vetor de valores aleatórios distribuídos uniformemente entre min e max, N é o conjunto de vizinhos da partícula e $\vec{P}_k$ é a melhor posição da partícula k. A função W pode representar qualquer função que se considere relevante, como por exemplo, retornar um valor constante.

$$\vec{P}_m = \frac{\sum_{k \in N} W(k)\vec{\varphi}_k \otimes \vec{P}_k}{\sum_{k \in N} W(k)\vec{\varphi}_k}, \tag{11}$$

$$\vec{\varphi}_k = \vec{\mathrm{U}}\left[0, \frac{\varphi_{max}}{|N|}\right] \ \forall k \in N \ . \tag{12}$$

Capítulo 3
Requisitos para a Nova Ferramenta de Simulação

O que motivou o desenvolvimento deste simulador foi a necessidade de uma ferramenta capaz de realizar simulações das mais diversas configurações possíveis do algoritmo do PSO. Além disso, era interessante que o simulador fosse capaz de simular algumas das variações mais importantes do PSO. Mas para que esta ferramenta pudesse ser realmente útil também, era necessário que ela tornasse possível o teste das configurações do PSO nos mais diversos cenários possíveis. Isso só poderia ser atingido se a ferramenta implementasse vários tipos de funções de teste.

Neste capítulo o leitor encontrará um conjunto de requisitos necessários para que o PSS pudesse realmente ser aplicado em pesquisas de alto nível.

3.1 Funções de Teste

As funções de teste desempenham um papel importantíssimo num simulador de técnicas de otimização com o intuito de fornecer um ambiente capaz de testar os algoritmos. Para o PSO, as funções de teste (ou otimização) representam o problema a ser resolvido pelo algoritmo, que deverá otimizá-las de forma a encontrar seu máximo ou mínimo.

As funções de otimização podem ser classificacas em dois grandes grupos, as unimodais e multimodais. Uma função $f(x)$ pode ser definida como unimodal se para um dado valor m ela é monotonamente crescente para $x \leq m$ e monotonamente decrescente para $x \geq m$. Neste caso, o valor máximo é $f(m)$ e não há nenhum outro máximo local. O mesmo pode ser dito se a função é monotonamente decrescente para $x \leq m$ e monotonamente crescente para $x \geq m$. Neste caso, $f(m)$ é o mínimo da função. Um exemplo de função unimodal pode ser visto na Figura 11.

Funções multimodais são aquelas que possuem mais de um máximo ou mínimo local, e podem ter tanto um ou mais máximos e mínimos globais, ou seja, são problemas bem mais complexos que funções unimodais. No caso do PSO, os mínimos locais podem ser considerados armadilhas, que podem atrair partículas, e por conseqüência suas vizinhas (de forma indireta) para regiões do espaço de busca que não contenham as soluções ótimas para um dado problema.

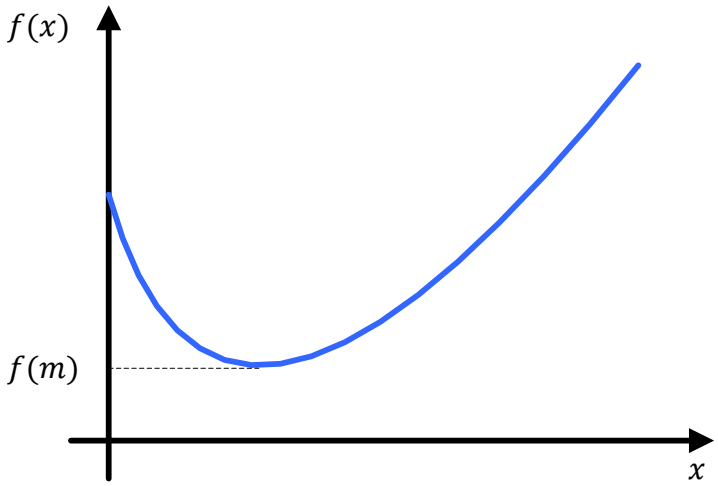


Figura 11. Exemplo de função unimodal.

3.1.1 Funções de Otimização Implementadas

As seguintes funções de otimização foram implementadas na ferramenta. Cada função será acompanhada por algumas de suas características. A maioria terá seus gráficos apresentados com a função adaptada para duas dimensões. Outras, no entanto, não podem ser plotadas pois só existem para espaços com quatro dimensões ou mais. Todos os gráficos das funções apresentadas foram retirados de [19] e podem ser visualizados no Apêndice A. As funções que dão origem a estes gráficos podem ser vistas na Tabela 1. Funções de teste implementadas.

Tabela 1. Funções de teste implementadas

Nome	Definição	*D*	Domínio
Ackley	$f = -20exp\left\{-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right\} - exp\left\{\frac{1}{D}\sum_{i=1}^{D}\cos 2\pi x_i\right\} + 20 + e$	*n*	$(-32{,}32)^D$

Six-hump Camel-cack	$f = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	*n*	$(-5,5)^D$
Goldstein&Price	$f = \{1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x1x2 + 3x22 \times 30 + 2x1 - 3x2218 - 32x1 + 12x12 + 48x2 - 36x1x2 + 27x22$	*n*	$(-2,2)^D$
Griewank	$f = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	*n*	$(-600,600)^D$
Rastrigin	$f = \sum_{i=1}^{D}\{x_i^2 - 10\cos(2\pi x_i) + 10\}$	*n*	$(-5,12,5,12)^D$
Rosenbrock	$f = \sum_{i=1}^{D-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	*n*	$(-30,30)^D$
Schwefel 1.2	$f = \sum_{i=1}^{D}\left(\sum_{j=1}^{i} x_j\right)^2$	*n*	$(-100,100)^D$
Schwefel 2.6	$f = -\sum_{i=1}^{D} x_i \sin\left(\sqrt{x_i}\right)$	*n*	$(-500,500)^D$
Esfera	$f = \sum_{i=1}^{D} x_i^2$	*n*	$(-100,100)^D$
Beale	$f = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$	*2*	$(-4,5,4,5)^2$
Bohachevsky1	$f = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$	*2*	$(-100,100)^2$
Bohachevsky2	$f = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1)\cos(4\pi x_2) + 0.3$	*2*	$(-100,100)^2$
Bohachevsky3	$f = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1 + 4\pi x_2) + 0.3$	*2*	$(-100,100)^2$
Booth	$f = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$	*2*	$(-10,10)^2$
Branin	$f = \left(x_2 - \frac{5}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10$	*2*	$(-5,15)^2$
DIxon&Price	$f = (x_1 - 1)^2 + \sum_{i=1}^{D} i(2x_i^2 - x_{i-1})^2$	*n*	$(-10,10)^D$
Easom	$f = -\cos(x_1)\cos(x_2)\, e^{-(x_1-\pi)^2-(x_2-\pi)^2}$	*2*	$(-100,100)^2$
Matyas	$f = 0.2(x_1^2 + x_2^2) - 0.48x_1x_2$	*2*	$(-10,10)^2$
Michalewicz	$f = -\sum_{i=1}^{D} \sin(x_i)\sin^{20}\left(\frac{ix_i^2}{\pi}\right)$	*n*	$(0,\pi)^D$

Perm	$f = \sum_{i=1}^{D}\left[\sum_{j=1}^{D}(j^i + 0.5)\left(\left(\frac{x_j}{j}\right)^i - 1\right)\right]^2$	*n*	$(-n, n)^D$
Shubert	$f = \left[\sum_{i=1}^{5} i\cos\left((i+1)x_1 + i\right)\right]\left[\sum_{i=1}^{5} i\cos\left((i+1)x_2 + i\right)\right]$	*2*	$(-10,10)^2$
Soma de Quadrados	$f = \sum_{i=1}^{D} ix_i^2$	*n*	$(-10,10)^D$
Trid	$f = \sum_{i=1}^{D}(x_i - 1)^2 - \sum_{i=2}^{D} x_i x_{i-1}$	*n*	$(-n^2, n^2)^D$
Zakharov	$f = \sum_{i=1}^{D} x_1^2 + \left(\sum_{i=1}^{D} 0.5ix_i\right)^2 + \left(\sum_{i=1}^{D} 0.5ix_i\right)^4$	*n*	$(-5,10)^D$
Colville	$f = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 + (x_3 - 1)^2 + 90(x_3^2 -$ *x42+10.1x3−12+x4−12+19.8x2−1x4−1*	*4*	$(-10,10)^D$
Levy	$f = \sin^2(\pi x_1) + \sum_{i=1}^{D}[(y_i - 1)^2(1 + 10\sin^2(\pi y_i +$ *1+yD− 121+*$\sin 22\pi yD$, $y_i = 1 + \frac{x_i - 1}{4}$	*n*	$(-10,10)^D$
Power Sum	$f = \sum_{i=1}^{D}\left[\left(\sum_{j=1}^{D} x_j^i\right) - b_i\right]^2$, $b_i = [8.0 \quad 18.0 \quad 44.0 \quad 114.0]$	*4*	$(0,4)^4$
Penalized P8	$f = \frac{\pi}{D}\{10\sin^2(\pi y_1) + \sum_{i=1}^{D-1}(y_i - 1)^2\{1 + 10\sin^2(\pi y_{i+1})\} +$ *yD−12+i=1Dμ(x1,10,100,4)* , $y_i = 1 + \frac{1}{4}(x_i + 1)$, $\mu(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, x_i > a \\ 0, -a \le x_i \le a \\ k(-x_i - a)^m, x_i < -a \end{cases}$	*n*	$(-50,50)^D$
Penalized P16	$f = 0.1\{\sin^2(3\pi x_0) + \sum_{i=1}^{D-1}(x_i - 1)^2\{1 + \sin^2(3\pi x_{i+1})\} +$ *xD−121+*$\sin 22\pi xD$*+ i=1Dμ(xi,5,100,4)* , $\mu(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, x_i > a \\ 0, -a \le x_i \le a \\ k(-x_i - a)^m, x_i < -a \end{cases}$	*n*	$(-50,50)^D$
Shekel 5	$f = -\sum_{i=1}^{5}\left\{\sum_{j=1}^{4}(x_j - a_{ij})^2 + c_i\right\}^{-1}$, $c = [0{,}1 \quad 0{,}2 \quad 0{,}2 \quad 0{,}4 \quad 0{,}4 \quad 0{,}6 \quad 0{,}3 \quad 0{,}7 \quad 0{,}5 \quad 0{,}5]$,	*4*	$(0,10)^4$

	$a = \begin{bmatrix} 4{,}0 & 1{,}0 & 8{,}0 & 6{,}0 & 3{,}0 & 2{,}0 & 5{,}0 & 8{,}0 & 6{,}0 & 7{,}0 \\ 4{,}0 & 1{,}0 & 8{,}0 & 6{,}0 & 7{,}0 & 9{,}0 & 5{,}0 & 1{,}0 & 2{,}0 & 3{,}6 \\ 4{,}0 & 1{,}0 & 8{,}0 & 6{,}0 & 3{,}0 & 2{,}0 & 3{,}0 & 8{,}0 & 6{,}0 & 7{,}0 \\ 4{,}0 & 1{,}0 & 8{,}0 & 6{,}0 & 7{,}0 & 9{,}0 & 3{,}0 & 1{,}0 & 2{,}0 & 3{,}6 \end{bmatrix}$		
Shekel 7	$f = -\sum_{i=1}^{7}\left\{\sum_{j=1}^{4}(x_j - a_{ij})^2 + c_i\right\}^{-1}$	*4*	$(0{,}10)^4$
Shekel 10	$f = -\sum_{i=1}^{10}\left\{\sum_{j=1}^{4}(x_j - a_{ij})^2 + c_i\right\}^{-1}$	*4*	$(0{,}10)^4$

3.2 Significância Estatística

Em estatística, um resultado é considerado estatisticamente significativo se ele tem uma baixa probabilidade de ter ocorrido por acaso. Uma diferença estatisticamente significativa simplesmente indica que há uma diferença entre os dois grupos de amostras analisados, não significa que a diferença é grande ou significante no sentido comum da palavra.

O nível de significância de um teste define o grau de certeza que se tem a respeito de uma hipótese que se quer confirmar. Em outras palavras, significa a probabilidade de se tomar a decisão de rejeitar a hipótesa nula, quando na realidade ela é verdadeira.

Em estatística, uma hipótese nula (H_0) é uma hipótese plausível que pode explicar um determinado conjunto de dados. Uma hipótese nula é testada para determinar se um conjunto de dados fornece informações suficientes para que seja procurada uma hipótese alternativa. Quando usada, a hipótese nula é considerada suficiente para explicar os dados sempre que não haja provas em contrário na forma de uma evidência estatística, ou seja, quando é observado uma certeza de pelo menos 95% de que a hipótese nula não consegue explicar o conjunto de dados. Semelhante a uma prova por absurdo, a hipótese nula é formulada apenas para que possa ser quebrada por um teste estatístico.

É possível para um experimento falhar ao ato de rejeitar a hipótese nula. Neste caso, a hipótese formulada consegue explicar os dados observados e, logo, não é necessária a busca de nenhuma hipótese alternativa. Ao rejeitar a hipótese nula, o teste indica que esta não consegue explicar os dados e que, conseqüentemente, uma hipótese alternativa deve ser buscada, mas não indica que

qualquer uma das hipóteses alternativas previamente indentificadas tenha maior ou menor chance de ser verdadeira.

Quando um determinado teste rejeita a hipótese nula erroneamente, isto é, acredita-se que a hipótese nula não explica os dados quando na realidade ela é verdadeira, este é um erro de decisão conhecido como erro Tipo I (também conhecido como erro de primeiro tipo) [22], ou determinação de falso positivo. Por outro lado, se o teste sugere que a hipótese nula é verdadeira quando na verdade ela deveria ser rejeitada, ele incorreu no erro de Tipo II (também conhecido como erro de segundo tipo) [22], ou determinação de um falso negativo.

Como um exemplo, suponha que o teste deseja identificar se um determinado indivíduo estará ou não infectado por uma bactéria *x* se exposto a ela por mais de trinta segundos. Neste exemplo, a hipótese nula é representada pelo suposição de que um indivíduo não estará infectado se exposto à bactéria por mais de trinta segundos. Ao executar o teste clínico, podem ser obtidos dois resultados: positivo ou negativo para a infecção. Os casos em que o teste apontou corretamente que um indivíduo estava infectado ou descartou a possibilidade de que este indivíduo estivesse infectado podem ser considerados como verdadeiros positivos. No entanto, o teste pode classificar como não infectado um indivíduo que estava de fato infectado. Este caso é o erro de Tipo I, pois o teste não rejeitou a hipótese nula. Outra possibilidade é que o indivíduo seja considerado infectado quando na realidade não foi infectado. Este caso é chamado de erro Tipo II, caracterizado pela rejeição da hipótese nula erroneamente. A Tabela 2 mostra o exemplo descrito anteriormente.

3.2.1 Inferência para Duas Populações e Teste de Mann-Whitney

Como este trabalho trata da implementação de uma ferramenta capaz de simular várias configurações diferentes é bastante interessante que ela também possa informar se uma determinada configuração tem melhor desempenho que outra. Esta funcionalidade é uma resposta à freqüente pergunta em Ciência: o método A é melhor que o método B?

Tabela 2. Exemplo de falso positivo e falso negativo.

		Condição Real	
		Infectado	**Não infectado**
Resultado do teste	**Infectado**	Verdadeiro positivo	Falso positivo (erro Tipo I)
	Não infectado	Falso negativo (erro Tipo II)	Verdadeiro negativo

A dificuldade está em caracterizar corretamente a equivalência de duas populações. Suponha que em um determinado cenário, cálculos posteriores demonstrem que as médias e desvios padrão para duas populações A e B sejam iguais, ou seja, $\mu_A = \mu_B$ e $\sigma_A = \sigma_B$. No entanto, a igualdade destes indicadores não catacteriza as populações A e B como iguais. É necessário que seja feita uma análise mais detalhada a respeito desta afirmação.

Os testes a serem realizados sobre as populações dependem intimamente da natureza das distribuições. Os chamados testes paramétricos são melhor direcionados para populações que possuem uma distribuição conhecida (e.g. a distribuição normal). Se um teste paramétrico for aplicado em uma população que não apresenta uma distribuição não conhecida, será observado uma queda significativa no grau de confiabilidade deste teste. Para populações com distribuição que não seguem a normal, os testes não-paramétricos são mais aconselhados pois consideram outros parâmetros da distribuição.

Os teste *t* de Wilcoxon [23] é bastante indicado para populações que seguem a normal e possuem a mesma variância. No entanto, estas condições devem ser verificadas antes que o teste possa ser aplicado, caso contrário o teste não terá a mesma confiabilidade.

No domínio de aplicações do PSO, não é possível garantir que as distribuições seguirão a normal, por este motivo, a ferramenta realiza um teste da classe não-paramétricos.

O teste de Mann-Whitney é baseado nos postos dos valores obtidos das populações quando combinadas. Isso é feito ondenando os valores do menor para o maior independende de que população cada valor é originado. A estatística do teste

é simplesmente a soma dos postos associados aos valores amostrados de uma das populações em análise. Se este valor for grande, é uma indicação de que os valores desta população tendem a ser maiores do que os valores da outra população. Nesse caso, podemos rejeitar a hipótese nula de que as duas populações são equivalentes.

De maneira geral, deseja-se testar duas amostras independentes, X_1, ..., X_n, de P_1, e Y_1, ..., Y_m, de P_2. Seja $N = n + m$ e combina-se as duas amostra em apenas uma, ordena-se os N valores do menor para o maior e chama-se $S_1 < S_2 < ... < S_m$ os postos dos Y_i (tratamentos) e $R_1 < R_2 < ... < R_n$ os postos dos X_i (controles). A estatística W_S pode ser escrita na forma da equação (13) [23], a sua média pode ser verificada na equação (14) [23] e a variância na equação (15) [23]. Se as populações P_1 e P_2 forem equiprováveis, espera-se que a distribuição de W_S seja em torno da média calculada pela equação (14).

$$W_S = S_1 + S_2 + \cdots + S_m \ , \tag{13}$$

$$E(W_S) = \frac{m(N+1)}{2} \ , \tag{14}$$

$$Var(W_S) = \frac{nm(N+1)}{12} \ . \tag{15}$$

Para o caso em que há empates, devem ser contabilizadas as quantidades de empates, de forma que tem-se d_1 observações empatadas no menor valor, d_2 observações empatadas no segundo menor valor etc. até d_e observações empatadas no maior valor, onde e é a quantidade de valores distintos. A distribuição W_S depende destes valores. A média de W_S pode continuar sendo calculada a partir da equação (14), mas a variância deve ser ajustada para levar em consideração os empates. Esse ajuste pode ser visto na equação (16).

$$Var(W_S) = \frac{nm(N+1)}{12} - \frac{nm}{12N(N-1)} \sum_{i=1}^{e} (d_i^3 - d_i) \ . \tag{16}$$

Para os casos de empate, tabelas deveriam ser construídas para cada uma das configurações de empate, o que é impraticável. Para os casos em que não há empate e os valores de m e n são pequenos (ou seja, as proporções $\frac{d_i}{N}$ são próximas de 1) ou a quantidade de empates for pequena, as tabelas ainda podem ser consultadas diretamente a partir do valor W_S. Para os casos em que os valores de m e n são muito grandes ou há muitos empates a aproximação normal será adequada.

Nestes casos, é utilizada a estatística Z, definida pela equação (17). O valor calculado de Z, então, deve ser utilizado para a consulta nas tabelas.

Na prática, a análise é feita baseada em um valor denominado α, que determina a probabilidade de a hipótese nula estar correta. O valor índice chamado de *p*-valor ($\hat{\alpha}$) é consultado em tabelas como a presente em [7] e comparado com α e caso seja menor ou igual a este valor, o teste deve rejeitar a hipótese nula.

$$Z = \frac{W_S - E(W_S)}{\sqrt{Var(W_S)}}. \tag{17}$$

3.2.2 Comparação Entre o Teste *t* e o Teste de Mann-Whitney

O teste *t* assume que as duas populações sendo analisadas são normais. Uma violação para esta regra muda a distribuição da estatística utilizada no teste e muda as probabilidades para os erros tipo I e II. O teste *t* é pouco sensível à diferença das variâncias se $m = n$, mas ele será bastante afetado se $m \neq n$.

Os testes *t* e de Mann-Whitney podem ser comparados quanto à sua robustez em termos de um valor chamado eficiência relativa assintótica, o qual leva às seguintes conclusões:

a) o teste *t* é mais poderoso quando as populações têm distribuições normais, mas a perda de eficiência do teste de Mann-Whitney é pequena (da ordem de 5%) nesse caso;
b) haverá pouca diferença entre os dois testes para distribuições próximas da normal;
c) o teste *t* e de Mann-Whitney têm a mesma eficiência se as populações forem uniformes, mas o teste de Mann-Whitney é três vezes mais eficiente se as populações forem exponenciais.

3.3 Gráficos Box Plot

O *box plot*, também conhecido como gráfico *Box-and-Whisker*, foi inventado pelo estatístico americano John Tukey em 1977. É bastante útil para se analisar várias informações estatísticas em relação à distribuição dos dados. Por isso

optamos por incluir como parte da ferramenta PSS. Um exemplo de *box plot* pode ser visualizado na Figura 12.

O valor da média é representado por um círculo cheio na cor preta. A mediana é representada por uma linha cheia que divide a caixa em duas partes. Tendo como base a mediana e os valores máximo e mínimo que a distribuição aprensenta, é possível desenhar os quartis *Q1* e *Q3*. Eles são posicionados no ponto que divide a metade superior e a metade inferior em duas partes, repectivamente. Logo, acima do limite superior da caixa, estão os 25% maiores valores da distribuição e abaixo da linha inferior da caixa estão os 25% menores valores da distribuição.

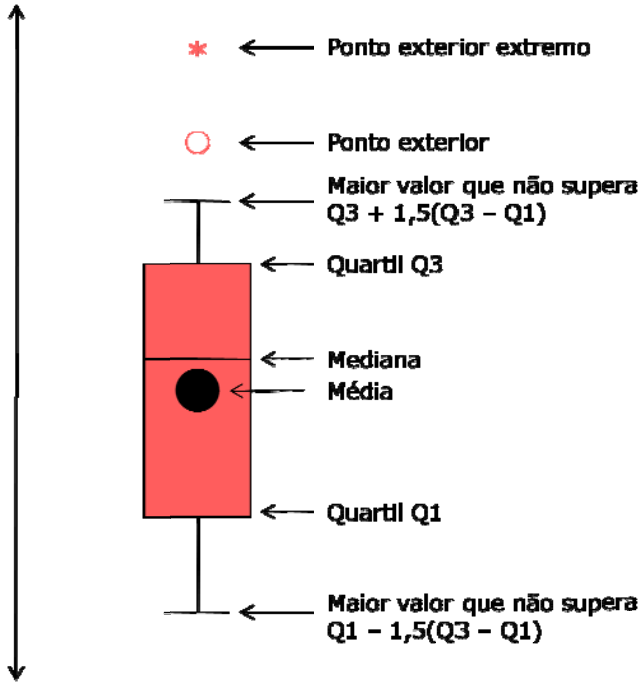


Figura 12. Exemplo de *boxplot*.

O segmento horizontal abaixo do limite inferior da caixa representa o menor valor que não foi menor que ( ). Já o segmento horizontal acima do limite superior da caixa marca o maior valor que não foi maior que

). Os valores calculados acima são usados também para determinar os pontos exteriores (também conhecidos como *outliers*) e exteriores extremos (também conhecidos por *far outliers*). Todos os pontos que estiverem localizados nos intervalos]] e [

[são considerados pontos externos, e são representados pelo círculo vazado.

Todos os pontos que estiverem localizados nos intervalos [-∞;$Q1 - 3(Q3 - Q1)$] e [$Q3 + 3(Q3 - Q1)$;+∞] são considerados pontos externos extremos e são representados pelos asteriscos. No PSS, os pontos externos extremos são representados por triângulos.

Capítulo 4

Apresentação da Ferramenta

Neste capítulo será apresentada a ferramenta desenvolvida durante este trabalho. Intitulada PSS (*Particle Swarm Optimization Simulation Shell*), ela é capaz se realizar simulações de algoritmos de otimização por enxame de partículas. Incorpora o conceito de topologia de comunicação entre partículas, funções de otimização e algoritmos de atualização de velocidade. E após uma simulação bem sucedida, o usuário pode aproveitar os dados gerados para criar gráficos de evolução das partículas e realizar comparações entre resultados de diferentes configurações do algoritmo utilizando testes de significância estatística.

4.1 Casos de Uso, Escopo e Requisitos

Os casos de uso do PSS podem ser agrupados sob a visão de dois atores: o usuário e o próprio sistema. Esta divisão é necessário pois nem todas as ações que o sistema executa foram inicializadas pelo usuário. O diagrama de casos do uso do PSS pode ser visualizado na Figura 13.

Do ponto de vista do usuário, os seguintes casos de uso podem ser listados:

a) realizar simulação;
b) realizar teste de Mann-Whitney;
c) gerar gráfico de *box plot*;
d) gerar gráfico de evolução.

Do ponto de vista do sistema, os seguintes casos de uso podem ser citados:

a) salvar arquivo de simulação;
b) salvar arquivo informativo de configuração;
c) calcular média e desvio padrão;

No Apêndice B o leitor pode ler todo o documento de casos de uso do PSS. No Apêndice C, o leitor tem acesso ao documento de escopo do PSS. No Apêndice D o leitor encontrará o documento de requisitos do PSS.

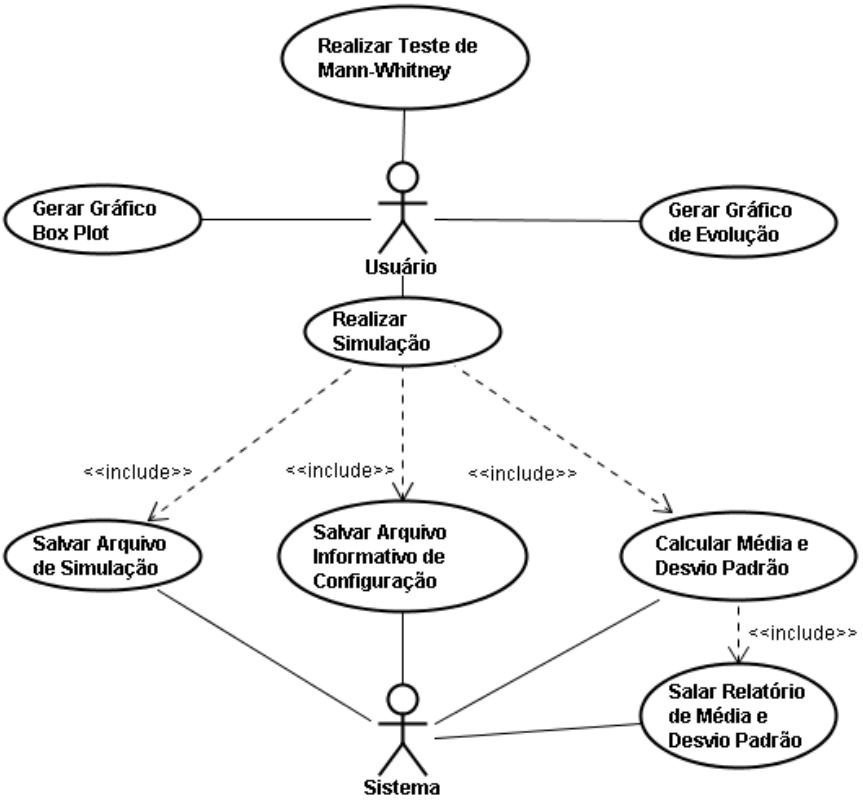


Figura 13. Diagrama de casos de uso do PSS.

4.2 Descrição das Principais Classes do PSS

Nesta seção o leitor encontrará a descrição e os relacionamentos entre os principais componentes do PSS. Todos os diagramas de classe estão sendo fornecidos em formato digital com este trabalho. Devido ao pouco espaço, a maioria dos diagramas estaria ilegível. Por este motivo, todos os diagramas estão sendo fornecidos em formato digital no endereço <http://www.dsc.upe.br/~efsb/pss.zip>

4.2.1 Partículas e Suas Variações

A classe que representa a partícula no PSS é denominada *Particle*. Algumas topologias necessitam que sejam criadas classes específicas para representar suas partículas. Até o momento, as classes dessa natureza existentes no pacote "br.upe.dsc.pss.swarm.particles" são: *Particle*, *ParticleRandom* e *ParticleTree*. O diagrama da Figura 14 de exibe essas classes.

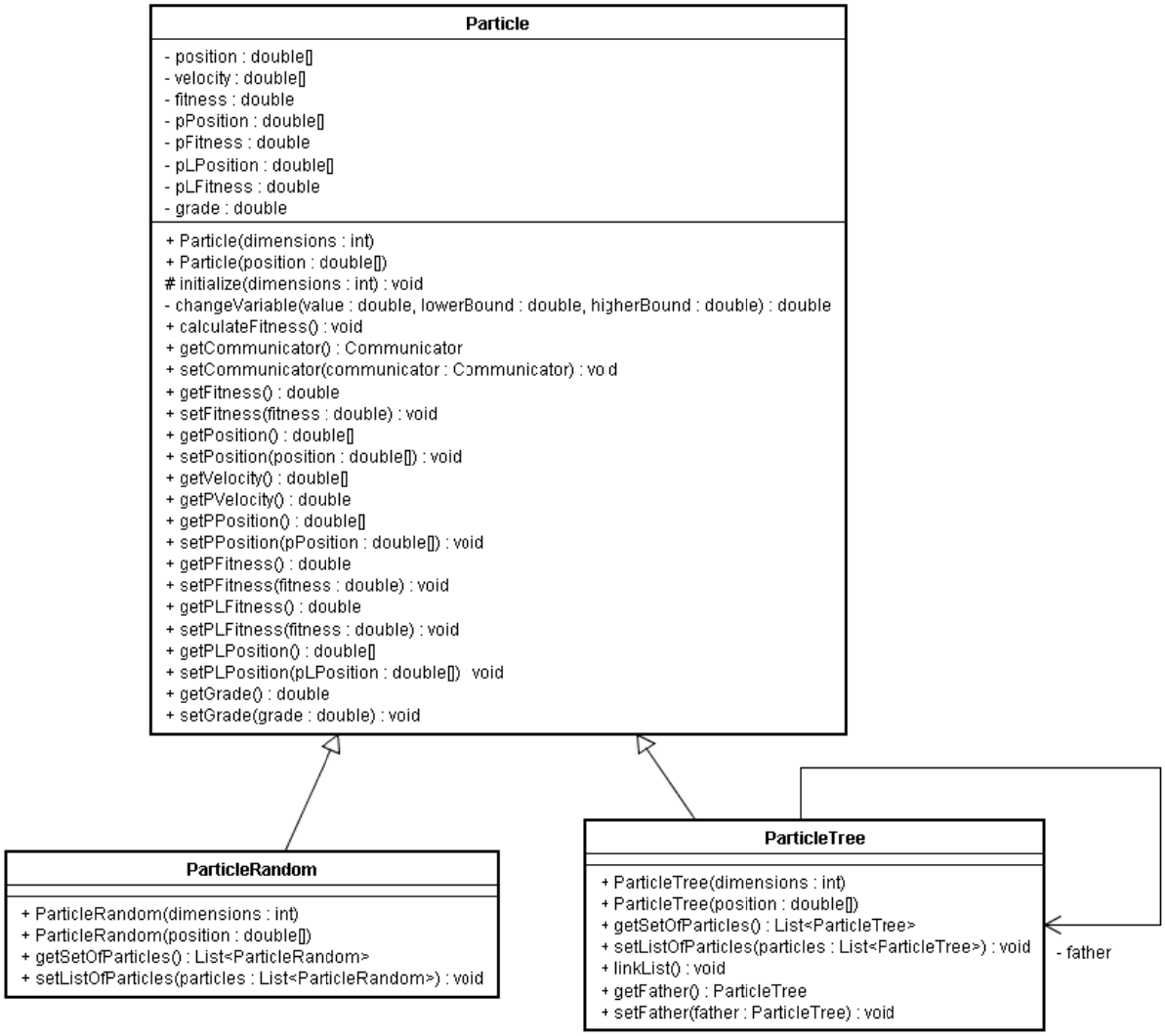


Figura 14. Diagrama de classes para o pacote br.upe.dsc.pss.swarm.particles.

4.2.2 Topologias

Todas as topologias incluídas no PSS devem herdar da classe *Topology*, que contém as definições comuns a todas as topologias, sem excessão. Cada topologia que necessite de novas funcionalidades deve estendê-la através do mecanismo padrão do Java. Algumas classes do pacote "br.upe.dsc.pss.swarm.topologies" podem ser vistas na Figura 15.

É interessante ressaltar que a topologia *Multi-Ring* estende da topologia Von Neumann, pois são bastante semelhantes. A diferença é apenas o mecanismo de rotação, presente apenas na topologia *Multi-Ring*. Outro ponto interessante é que a estratégia de rotação, representada pelas classes *Rotarion* e *NoRotation*, está desacoplada da topologia, o que permite que novos mecanismos de rotação sejam

implementados e adicionados ao sistema sem que haja a necessidade de modificar qualquer outra classe.

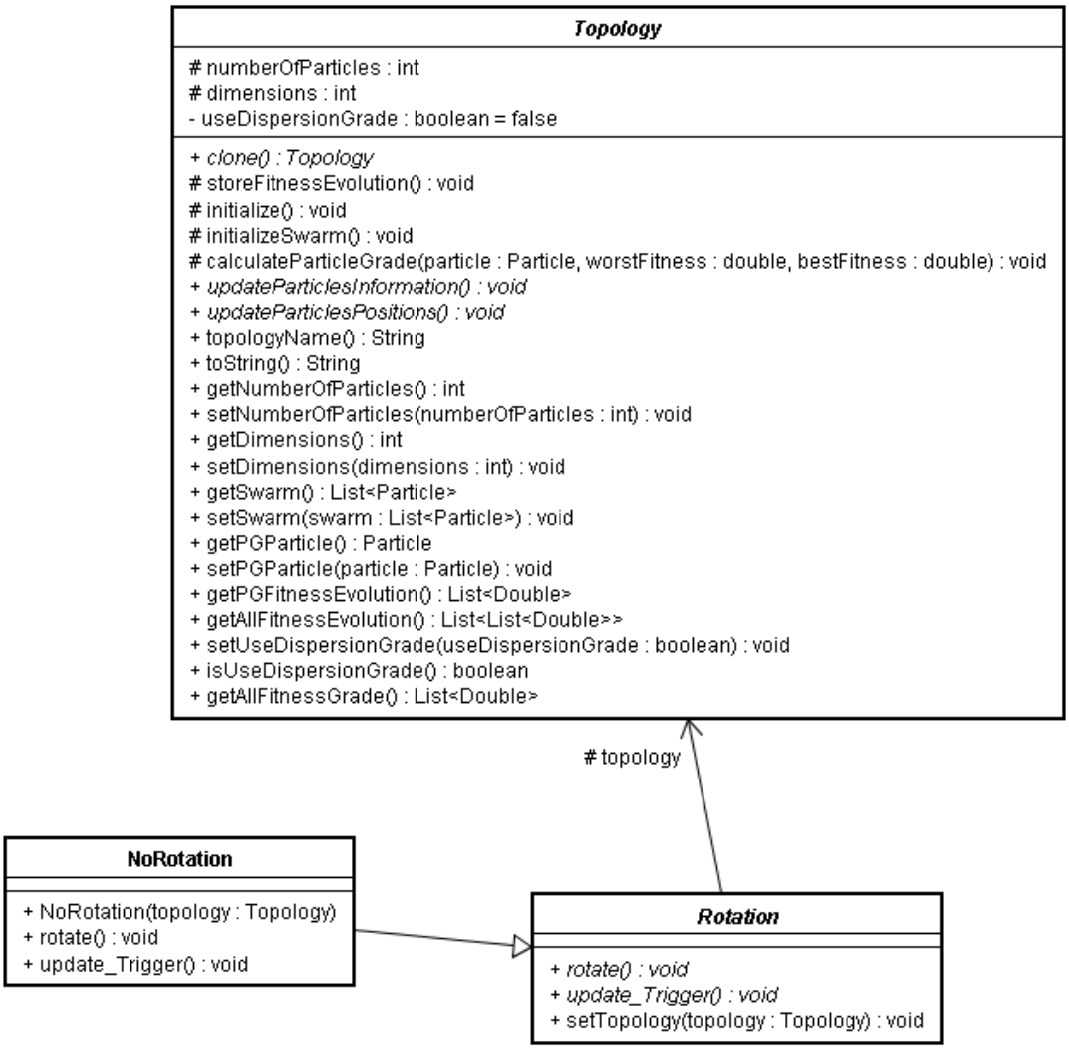


Figura 15. As classes do pacote br.upe.dsc.pss.swarm.topologies.

4.2.3 Implementação do Algoritmo

O algoritmo é implementado através das classes *SwarmAlgorithm*, *Pso*, *PsoImpl*, *Execution* e *PsoExecution*. A classe *PsoExecution* é responsável por gerenciar a execução do algoritmo e determinar quando os relatórios de evolução do enxame devem ser gravados. A classes *PsoImpl* é responsável por executar de fato o algoritmo. Ela utiliza a configuração feita pelo usuário e realiza os passos do algoritmo.

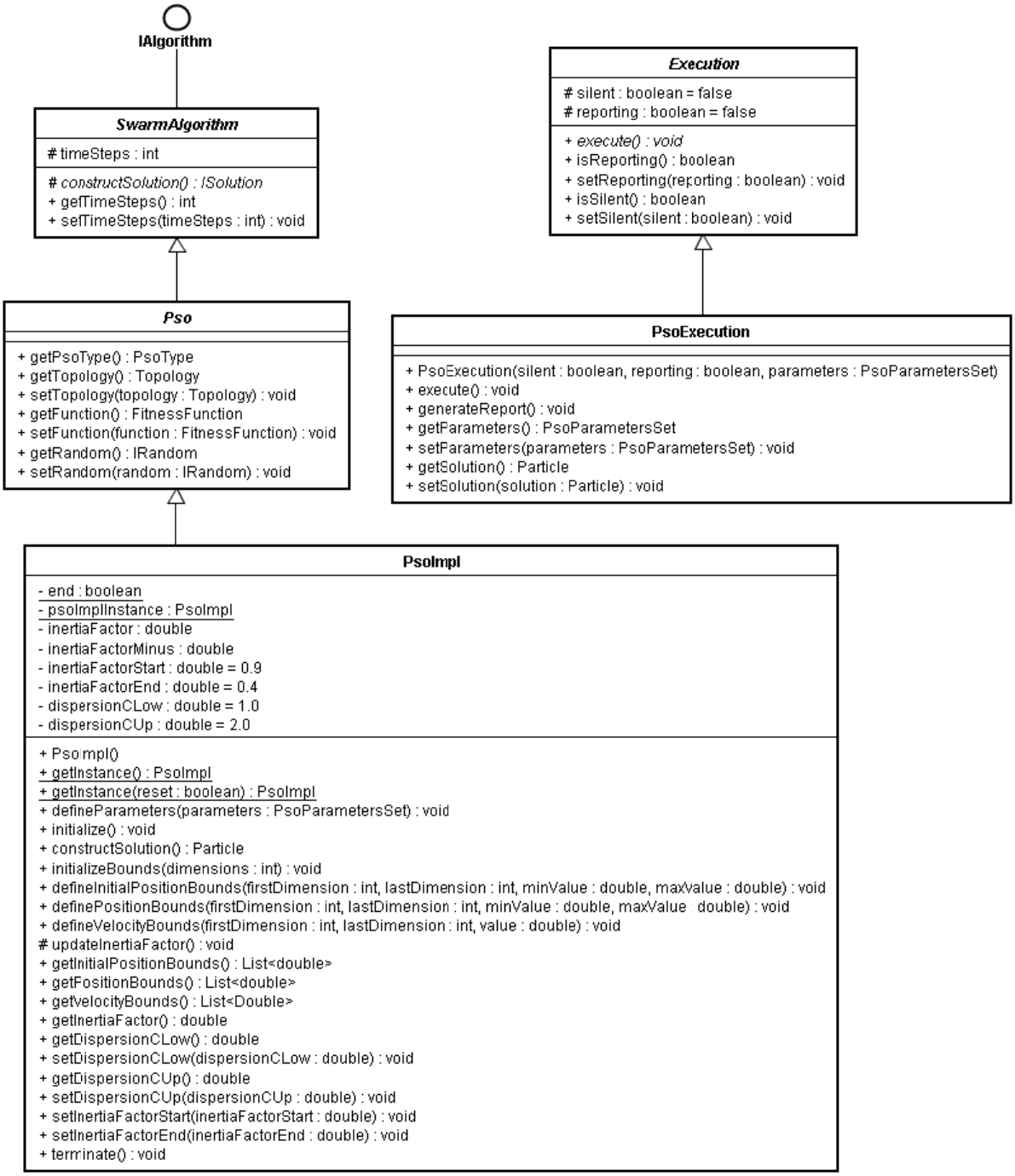


Figura 16. Classes que implementam o algoritmo do PSO.

4.2.4 Execução do Algoritmo

O algoritmo do PSO segue é executado seguindo o diagrama de seqüência encontrado na Figura 17. A *GUI* inicia todo o processo quando o usuário pressiona o botão *Start*. A *GUI* então inicializa a classe *PsoImpl*. Por simplicidade, várias chamadas são omitidas entre a classe *GUI* e a classe *PsoImpl*, mas todas têm a

intenção de inicializar a classe *PsoImpl*. Em seguida a *GUI* inicializa a classe *PsoSimulation* e inicia o algoritmo. A classe *PsoSomulation* por sua vez cria um novo diretório para a gravação dos relatórios de simulação. A classe *PsoExecution* é requerida e por sua vez repassa a chamada para a classe *PsoImpl*, que é a classe que implementa o algoritmo e encontra a solução para o problema (função de teste).

Este processo é repetido uma quantidade de vezes estabelecida na *GUI* e salva o relatório de simulação no diretório criado. A classe *PsoSimulation* cria o relatório final chamando a classe *ResultsSimulator*. Esse relatório contém as médias e desvios padrão para o conjunto de simulações que se acabou de se executar.

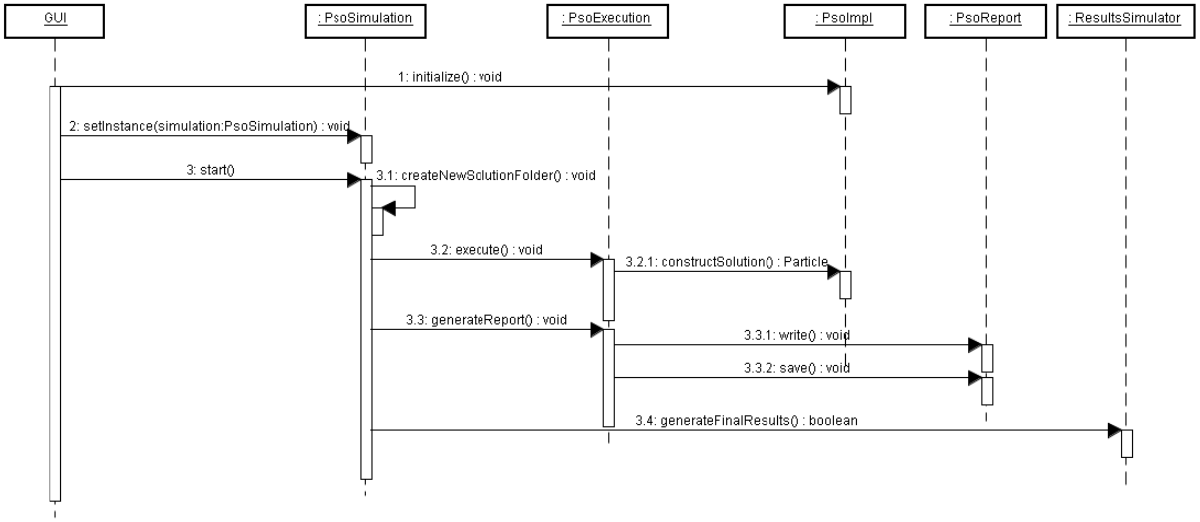


Figura 17. Diagrama de seqüência do processo de solução do problema.

4.3 Parametrização e Telas do PSS

A Figura 18 exibe a tela principal do PSS. É possível perceber que a interface gráfica é bastante simples e intuitiva, dando ao usuário a capacidade de operar facilmente a maior parte dos parâmetros de configuração da ferramenta a partir de sua tela principal. Os campos estão agrupados de forma que o usuário possa acessá-los mais intuitivamente.

A Figura 19 mostra em mais detalhe o grupo de campos denominado *Algorithm* (Algoritmo). Neste grupo o usuário pode especificar qual o algoritmo a ser usado para a atualização das velocidades das partículas. Como pode ser visto na

Figura 19, o usuário pode escolher entre as estratégias *Basic* (Básico), *Constricted* (Fator de Constrição) ou *Inertia* (Peso Inercial).

Escolhendo a opção *Basic*, o usuário está informando ao sistema que deseja que ele utilize a Equação (1) para a atualização de velocidades. A opção *Basic* se trata da estratégia adotada por Kennedy e Eberhart em seu primeiro trabalho [4]. Ao escolher a segunda opção, denominada *Constricted*, o usuário informa ao sistema que no momento de atualizar as velocidades das partículas ele deve utilizar as Equações (4) e (5). A outra opção disponível é a utilização do conceito de peso inercial. O sistema então deverá usar Equação (3) para atualizar as velocidades das partículas.

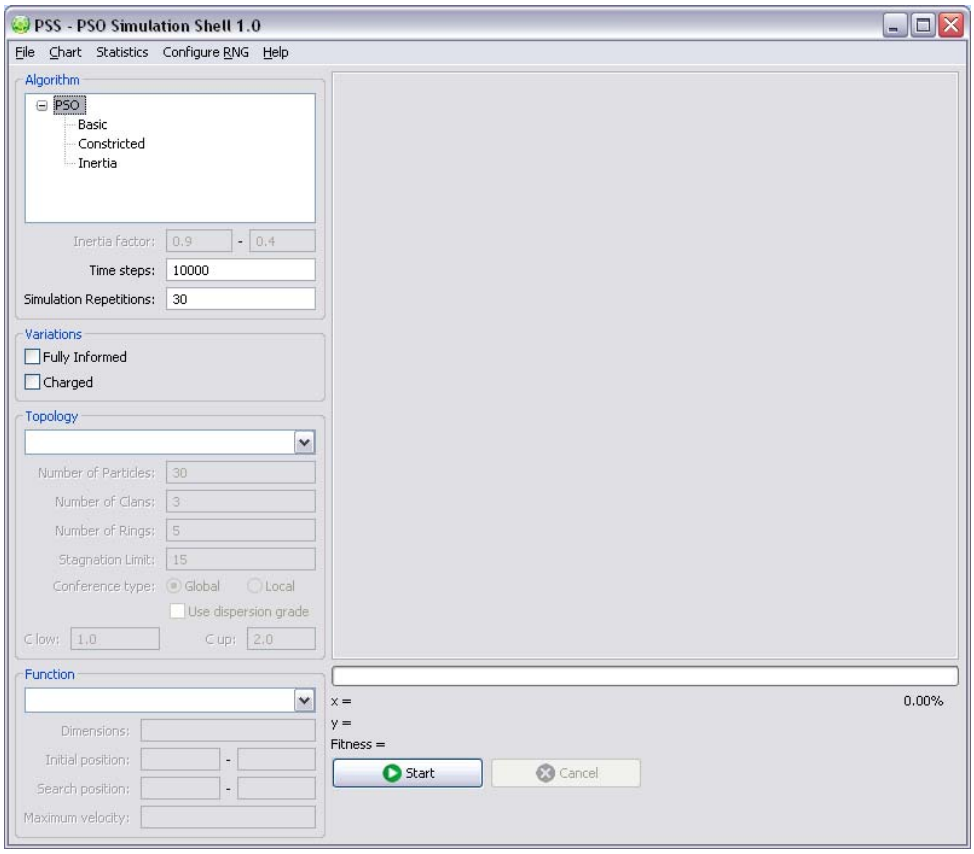


Figura 18. Visão da tela principal do PSS.

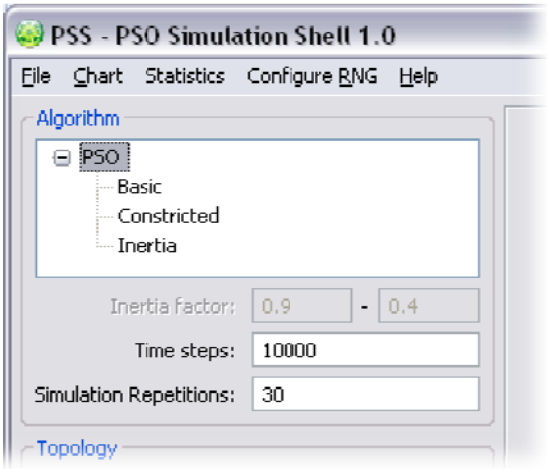


Figura 19. Grupo *Algorithm*.

Ainda no grupo *Algorithm*, o usuário pode determinar a quantidade de iterações do algoritmo, representado pelo campo *Time Steps* (Intervalos de Tempo). O valor especificado neste é bastante importante para o desempenho do algoritmo, pois em alguns casos, uma configuração pode levar mais "intervalos de tempo" para chegar a uma solução satisfatória. O valor padrão para este campo é 10000, pois vários trabalhos na literatura utilizam este valor, mas nada impede que o usuário o altere para o valor que desejar.

O campo *Simulation Repetitions* (Repetições de Simulação) informa quantas vezes o sistema deve realizar a mesma simulação. Caso o usuário informe apenas o valor 1 neste campo, o sistema executa a simulação com a quantidade de passos determinada em *Time Steps* e espera pela próxima ação do usuário. Caso o usuário determine um valor *n* tal que $n > 1$, o sistema realizará a simulação *n* vezes com a mesma configuração especificada pelo usuário. Esta possibilidade é muito útil pois a análise dos dados gerados por uma única simulação pode não fornecer conclusões confiáveis quanto ao desempenho de uma determinada configuração do algoritmo. A não existência desta funcionalidade implicaria que o usuário deveria realizar individualmente cada simulação.

A escolha da atualização de velocidade via *Inertia* (Peso Inercial) implica que sejam definidos os valores inicial e final dos fatores de inércia, representados pelos campos *Inertia Factor* (Fator Inercial). Assim, quando o usuário seleciona o tipo de PSO *Inertia*, os campos *Inertia Factor* são ativados e a ferramenta permite a edição por parte do usuário, como pode ser verificado na Figura 20.

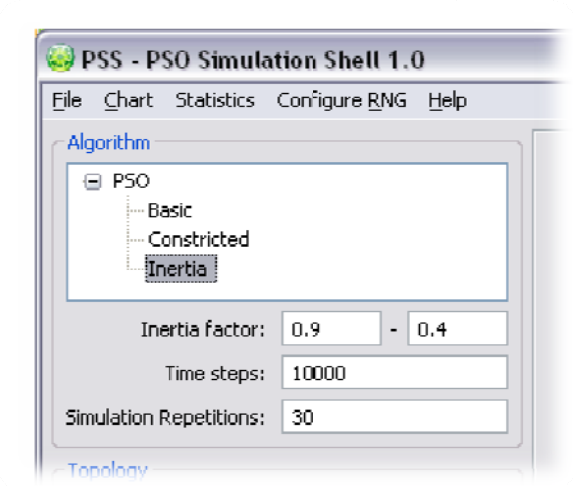


Figura 20. Tipo *Inertia* selecionado.

O grupo *Variations* (Variações) pode ser visto na Figura 21. O usuário pode escolher se deseja executar o algoritmo com as variações *Fully Informed* [21] e/ou a variação *Charged* [20].

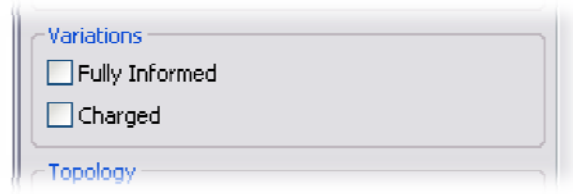


Figura 21. Grupo *Variations.*

O grupo *Topology* (Topologia), que pode ser visto na Figura 22, oferece parâmetros para que o usuário possa selecionar a topologia a ser usada pelo enxame. A utilização de algumas topologias, no entanto, exige a inicialização de alguns parâmetros específicos.

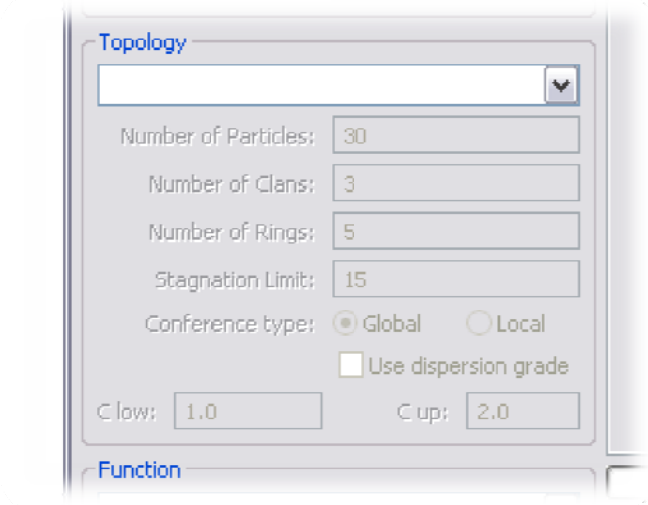


Figura 22. Grupo *Topology*.

Ao selecionar qualquer topologia, o campo *Number of Particles* (Quantidade de Partículas) é habilitado, como pode ser visto na Figura 23. Neste campo, o usuário define a quantidade de partículas que o enxame deve conter. Ainda nesta figura, o leitor pode observar a topologia selecionada foi a *Clan* [24], o que faz com que os campos *Number of Clans* (Quantidade de Clãs) e *Conference Type* (Tipo da Conferência) estejam ativados. No campo *Number of Particles* o usuário pode configurar a quantidade de partículas que cada clã da topologia deve possuir. O campo *Conference Type* é necessário para configurar o tipo de comunicação entre os líderes dos clãs no momento da conferência.

O campo *Use Dispersion Grade* (Usar Fator de Dispersão) permite que o usuário ative ou desative a utilização do conceito de *grade* introduzido por Cai *et al* [25]. Ao selecionar o campo *Use Dispersion Grade*, os campos c_{up} e c_{low} passam a estar habilitados, como pode ser visto na Figura 24. Os valores definidos nestes campos influenciarão diretamente no cálculo do coeficiente social das partículas. Os valores sugeridos de 1,0 e 2,0 para c_{up} e c_{low}, respectivamente, se mostraram bastantes eficazes nas simulações.

O próximo grupo é chamado *Function* (Função), que pode ser observado na Figura 25. Neste grupo o usuário encontrará campos relativos à configuração do problema que o algoritmo deve solucionar. O primeiro campo é o campo relativo à função de otimização a ser utilizada na simulação. Todas as função citadas na seção 3.1.1 foram implementadas e estão disponíveis no sistema.

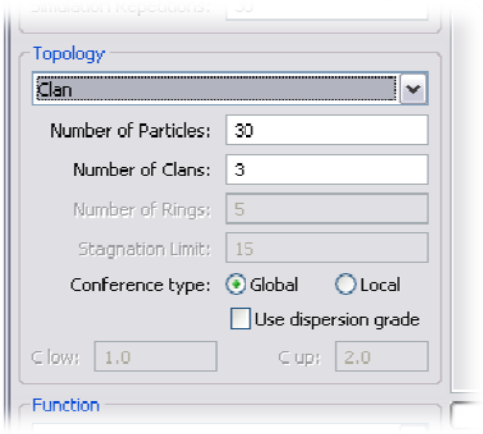


Figura 23. Grupo *Topology* com a topologia *Clan* selecionada.

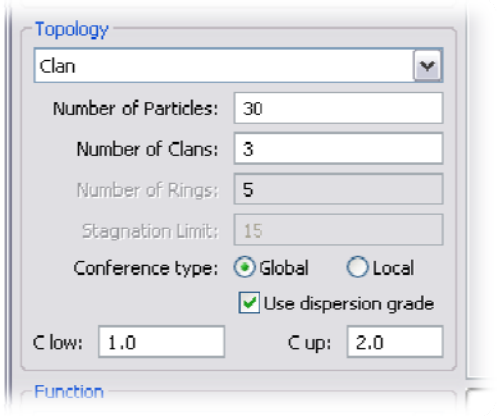


Figura 24. Campo *Use Dipersion Grade* selecionado.

Pode-se observar ainda na Figura 25 que os campos relativos à configuração da função permanecerão desabilitados até que o usuário selecione uma função de otimização. A partir do momento que o usuário seleciona a função, todos os demais campos são habilitados para edição, como pode ser visto na Figura 26 (a).

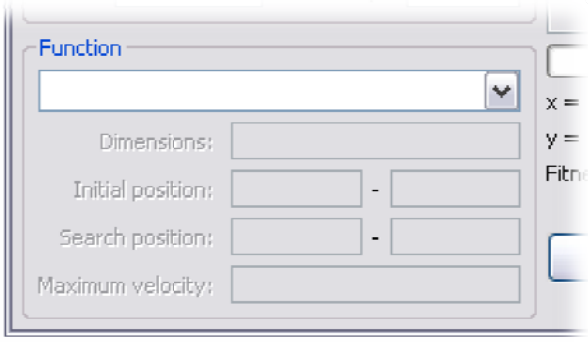
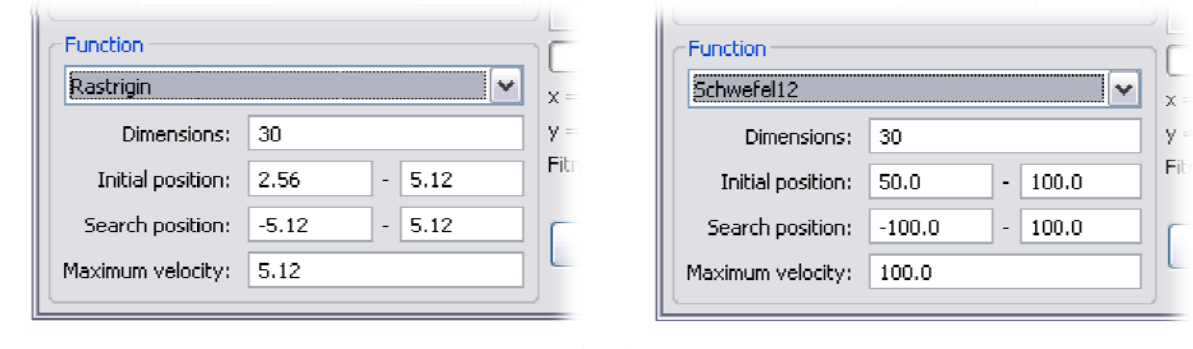


Figura 25. Grupo *Function*.

No campo *Dimensions* (Dimensões) o usuário deve informar a quantidade de dimensões para a função de otimização selecionada. Na maioria das vezes as função têm um número ilimitado de dimensões. Neste caso, o sistema sugere o valor 30 para este campo. Algumas outras funções têm uma limitação em relação à sua dimensionalidade, como por exemplo as funções Matyas (gráfico na Figura 69), Easom (gráfico na Figura 68) e Goldstein & Price (gráfico na Figura 57) que possuem apenas versões com duas dimensões e a função Colvile, válida apenas em um espaço 4-dimensional (por este motivo não possui gráfico).



(a) (b)

Figura 26. Grupo *Function* com as função de otimização Rastringin (a) e Schwefel 1.2 selecionadas.

O campo *Initial position* (Posição Inicial) é responsável por fornecer ao algoritmo uma região onde devem ser posicionadas as partículas quando do início da simulação. Isto é bastante útil pois ao mesmo tempo que o usuário pode fazer com que as partículas sejam inicializadas em qualquer ponto do espaço de busca, ele também pode fazer com que as partículas sejam inicializadas em uma pequena região. Esta funcionalidade permite que vários cenários diferentes sejam simulados, fornecendo mais flexibilidade à ferramenta.

Os campos rotulados como *Search position* (Posição de Busca) indicam ao algoritmo os limites do espaço de busca. Para cada uma das dimensões da função, o algoritmo não deverá realizar a busca fora dos limites definidos pelos valores nestes campos.

O campo *Maximum velocity* (Velocidade Máxima) define a velocidade máxima que uma partícula pode atingir durante a execução do algoritmo. Em muitos casos, algumas partículas sofrem o que é chamado de explosão de velocidade, caracterizada pelo rápido aumento da velocidade da partícula. Esse aumento indiscriminado pode fazer com que a partícula extrapole o espaço de busca em apenas uma unidade de tempo. Mesmo com a introdução do peso inercial e do fator de constrição a limitação de velocidade não foi totalmente descartada. Uma estratégia mista é preferível.

Vale a pena ressaltar que os parâmetros de configuração podem mudar bastante de uma função pra outra. O sistema fornece ao usuário valores padrão para cada função disponível para simulação, agilizando e padronizando o processo geração de dados. Na Figura 26 (a), o usuário selecionou a função Rastrigin (seu gráfico pode ser visualizado na Figura 59). Para esta função, o sistema sugere o valor 30 para o campo *Dimensions* (a função é *n*-dimensional), os valores 2,56 e 5,12 para os campos *Initial Position* e o valor 5,12 para o campo *Maximum velocity*. Já na Figura 26 (b) o usuário selecionou a função Schwefel 1.2.

Como pode ser visto, o sistema sugeriu diferentes valores para os campos descritos anteriormente. Os valores padrão para cada função foram obtidos através trabalhos como de Bratton [26], que sugere um padrão de teste para o algoritmo de otimização por enxames de partículas.

4.4 Realização de Simulação com o PSS

Depois de configurar todos os parâmetros necessários e clicar no botão *Start*, o sistema inicia a simulação. O sistema em simulação pode ser visto na Figura 27. Durante a simulação é exibida uma barra de progresso que apresenta o progresso da simulação corrente. Os valores de *x* e *y* apresentados logo abaixo da barra de progresso são os valores correntes para as duas últimas dimensões da melhor posição visitada pelo enxame, ou seja, a que no momento apresenta o melhor desempenho medido pelo seu *fitness.* Este *fitness* é apresentado logo abaixo, no campo *Fitness*.

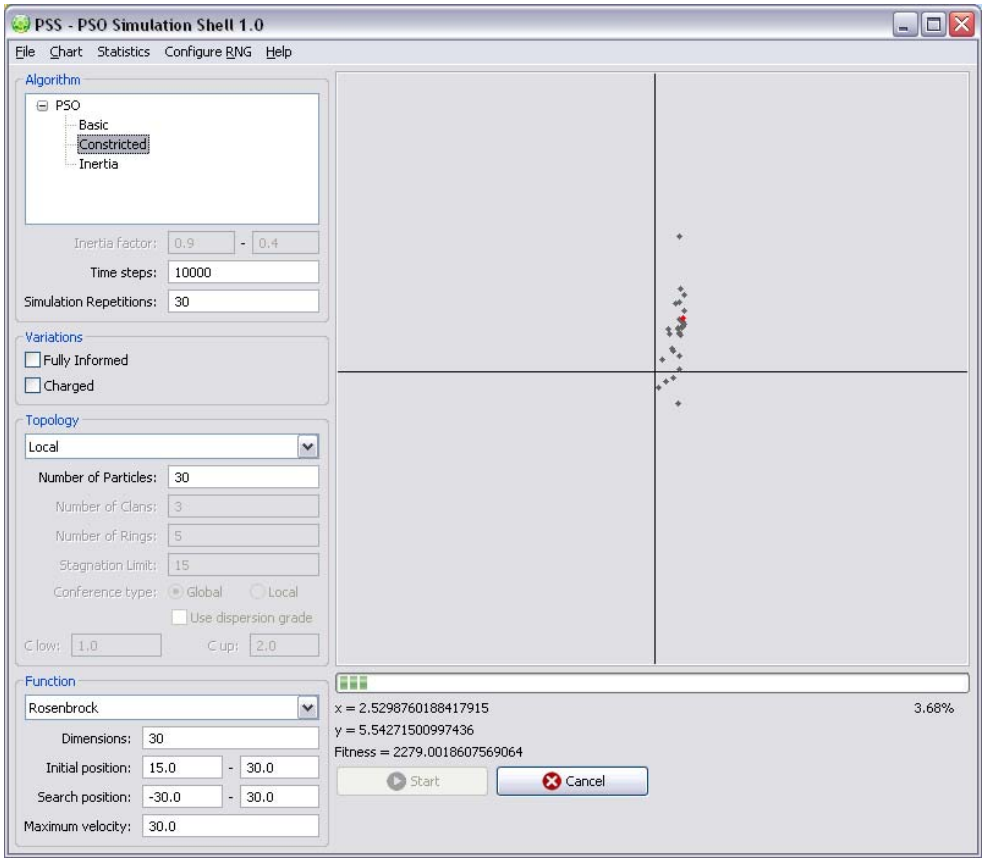


Figura 27. O PSS em modo de simulação.

Para cada simulação, o sistema cria um arquivo de relatório contendo a evolução do enxame naquela simulação. Este relatório contém as seguintes informações:

a) início da simulação (dia e hora);
b) o fim da simulação (dia e hora);
c) o tempo total de execução daquela simulação;
d) o fitness da melhor partícula do enxame em cada iteração do algoritmo.

Como o usuário pode escolher realizar várias simulações em seqüência automaticamente, o sistema poderá gerar vários relatórios. Por exemplo, na simulação demonstrada pela Figura 27, o sistema irá gerar trinta relatórios de evolução. Se o sistema gravar todos os relatórios em um único diretório, o usuário não será capaz de, em um momento posterior, identificar facilmente quais relatórios lhe interessam para análise, apesar de os relatórios serem nomeados segundo o padrão: “PsoEvolutionReport_hh-mm-ss.txt”, onde “hh” representa a hora com dois

dígitos, “mm” os minutos com dois dígitos e “ss” os segundos com dois dígitos do momento em que a simulação foi iniciada.

A solução para o problema foi criar um diretório à parte para cada conjunto de simulações. Digamos que o usuário iniciou a simulação da Figura 27 no dia 04/10/2008 precisamente às 13:03:27. O sistema então cria um diretório de nome “04-20-2008_13-03-27” sob o diretório padrão de relatórios chamado “*reports*” (relatórios). Neste diretório estarão todos os relatórios resultado desta simulação.

Ainda, após o término do ensaio (independente da quantidadade de simulações realizadas), o sistema gera dois arquivos. O primeiro arquivo contém informações acerca da configuração utilizada para o ensaio. Seu nome segue o seguinte padão: “PsoReport_hh-mm-ss.txt”, onde “hh” representa a hora com dois dígitos, “mm” os minutos com dois dígitos e “ss” os segungos com dois dígitos do momento em que a simulação foi finalizada. As seguintes informações compõem o relatório:

a) tipo do PSO (*Inertia*, *Constricted* ou *Basic*);

b) função de otimização selecionada;

c) quantidade de iterações para cada simulação;

d) topologia selecionada;

e) gerador de números aleatórios selecionado;

f) semente inicial do gerador de números aleatórios.

O segundo arquivo gerado, chamado “FinalResults.txt” contém a análise de média e devio padrão para as simulações geradas. As médias e desvios padrão são calculados levando em consideração cada uma das iterações de cada uma das simulações do ensaio. Se o usuário decidiu realizar simulações com dez mil iterações, o arquivo “FinalResults.txt” conterá dez mil pares de valores de média e desvio padrão. Este relatório é muito importante para a análise de uma determinada configuração do algoritmo pois analisando-o, pode-se, por exemplo, dizer se o enxame se apresentou coeso durante a busca ou se o desempenho de uma topologia é satisfatório para um determinado tipo de problema.

Durante a simulação, o sistema fornece ao usuário uma visão bastante interessante do processo de convergência do enxame. A Figura 28 exibe a região da tela principal do PSS responsável por esta visualização. Como a grande maioria das funções são *n*-dimensionais, o desenho de uma gráfico de evolução com *n* dimensões não é possível. A solução encontrada foi apresentar o gráfico com apenas as duas últimas dimensões das partículas, o que já permite ao usuário ter uma boa noção do comportamento do enxame. A partícula com o melhor desempenho é apresentada como um ponto em vermelho, enquanto as demais são apresentadas como pontos em cinza.

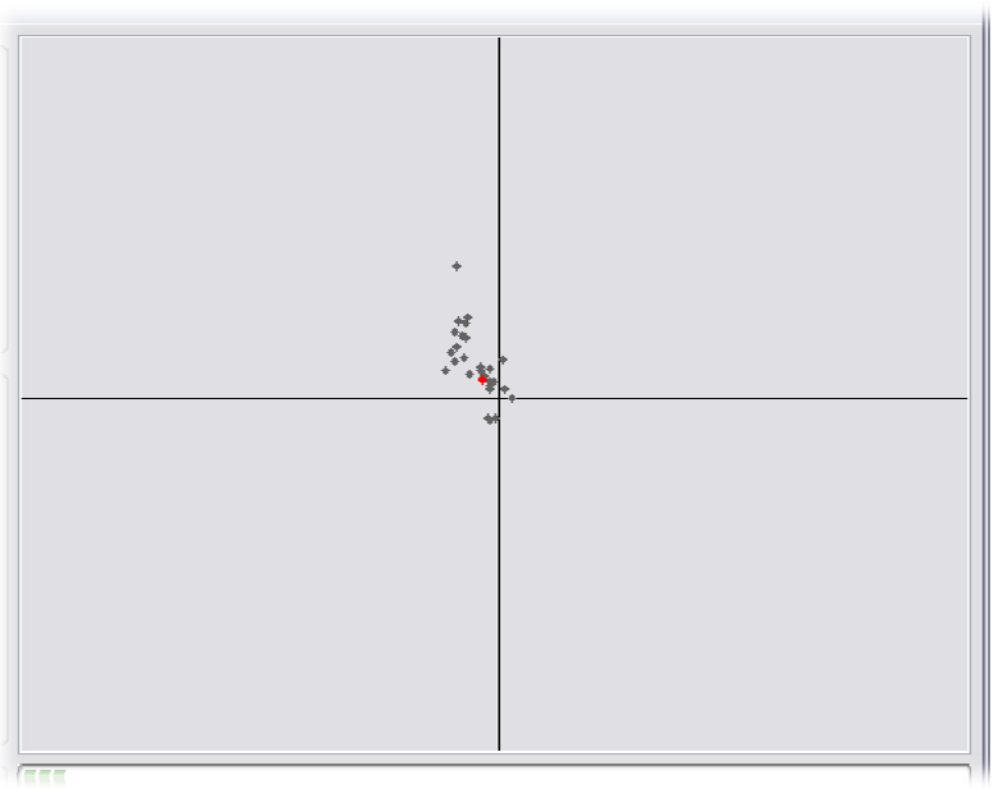


Figura 28. Região de visualização das partículas em movimento.

4.5 Análise de Resultados com o PSS

Depois de gerada a massa de dados pelas simulações o usuário pode realizar algumas análises estatísticas. A Figura 29 mostra as opções que o usuário tem sob o menu *Chart* (Gráfico).

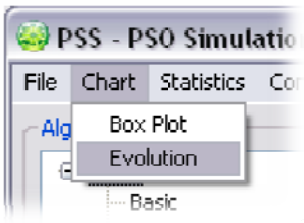


Figura 29. Opções em no menu *Chart.*

A opção *Box Plot*, irá levar o usuário a uma tela onde lhe são solicitados algumas informações para que seja gerado o *box plot*. A referida tela pode ser visualizada na Figura 30.

Figura 30. Tela para geração do *box plot*.

Nesta tela, o usuário deve informar qual o diretório que contém os relatórios de evolução para a geração do *box plot*. O sistema solicita uma pasta, pois como será gerado o *box plot*, é necessário que haja uma massa de dados para comparação devido à natureza deste gráfico.

Nesta tela o usuário pode ver o campo *Folder* (Pasta), onde o usuário poderá informar manualmente o caminho do diretório onde estão os relatórios do ensaio. Também há a possibilidade de o usuário utilizar o botão *Browse* (Navegar), que abrirá um diálogo e o usuário pode navegar pela estrutura de diretórios de seu computador até o diretório do ensaio mais facilmente. Ao confirmar a seleção, o sistema preencherá o campo *Folder* com a escolha do usuário.

Em alguns casos, o gráfico gerado pode não ficar com a apresentação adequada, como por exemplo o gráfico na Figura 31. Para esses casos o sistema oferece a possibilidade de gerar o gráfico com o eixo das ordenadas logartímico. Para ativar essa funcionalidade o usuário só precisa checar o campo *Use Logarithmic Y Axis* (Usar Eixo Y Logarítmico). O resultado disso é um gráfico mais compreensível, como o visto na Figura 32 que usou a mesma massa de dados utilizada para gerar o gráfico da Figura 31.

Como estão sendo tratados ensaios nos quais a quantidade de iterações do algoritmo gira na casa das dezenas de milhares, é preciso ter algum mecanismo capaz de reduzir a quantidade de amostras no gráfico sem prejudicar a visualização da distribuição. O campo *Offset* (Intervalo) permite que o usuário informe ao sistema o intervalo para obtenção das amostrar quando da leitura dos arquivos de relatório de evolução. O sistema sugere o valor 500, que é um valor bastante razoável para as 10000 iterações que o sistema também sugere. Neste cenário, o gráfico vai ser populado com valores das interações múltiplas de 500, ou seja, {0, 500, 1000, 1500, 2000, ... , 10000}, o que é consistente com o gráfico mostrado na Figura 31, onde são exibas vinte e uma amostras.

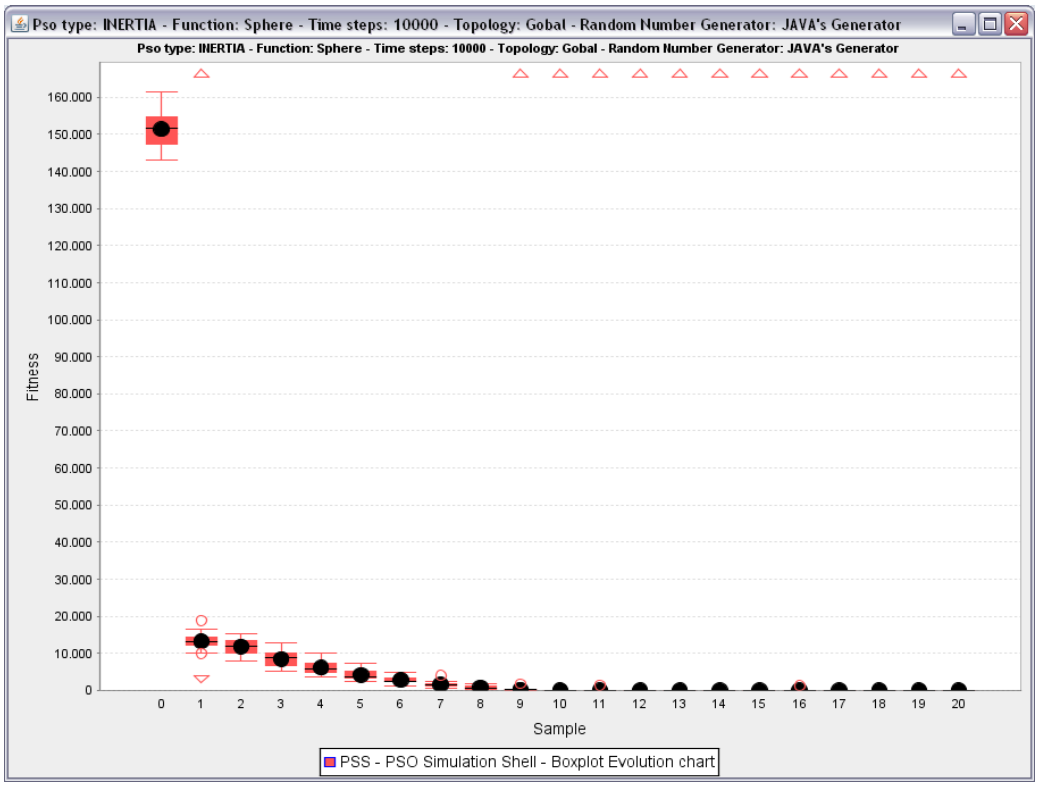


Figura 31. Exemplo de gráfico *box plot* com eixo linear.

Outro gráfico gerado pela ferramenta é o gráfico de evolução. Este gráfico tem um objetivo um pouco diferente do *box plot*, que compara uma grande massa de dados. Ele mostra a evolução de uma única partícula durante o tempo. A tela que permite gerar este gráfico é bem semelhante à tela do *box plot*, como pode ser visualizado na Figura 33.

Os dados a serem informados são praticamente os mesmos, mas o gráfico resultante é bastante diferente. O usuário deve informar um arquivo em vez de um diretório, pois este gráfico exibe apenas a evolução de uma simulação. Um exemplo de gráfico de evolução pode ser visto na Figura 34. A opção de eixo logarítmico também está presente para este gráfico. Um gráfico com esta opção ativada e apresentando a mesma massa de dados do gráfico na Figura 34 pode ser visto na Figura 35.

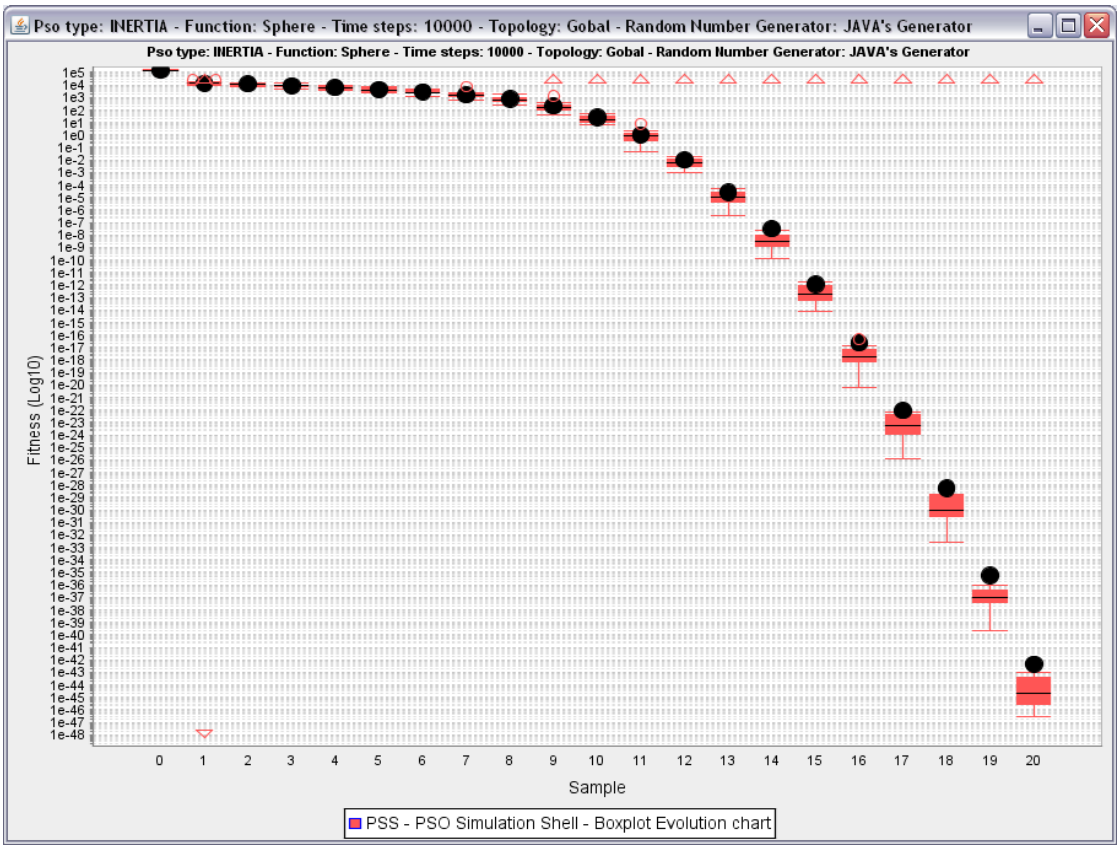


Figura 32. Exemplo de gráfico *box plot* com eixo logarítmico.

Figura 33. Tela para geração do gráfico de evolução.

A outra categoria de análise presente na ferramenta é a *Statistics* (Estatística), como pode ser visto na Figura 36. Abaixo do menu *Statistics* o usuário

pode ver a opção *Mann-Whitney*. Ao clicar nesta opção será apresentada ao usuário a tela para geração da análise do teste de Mann-Whitney [7], também conhecido como Teste U.

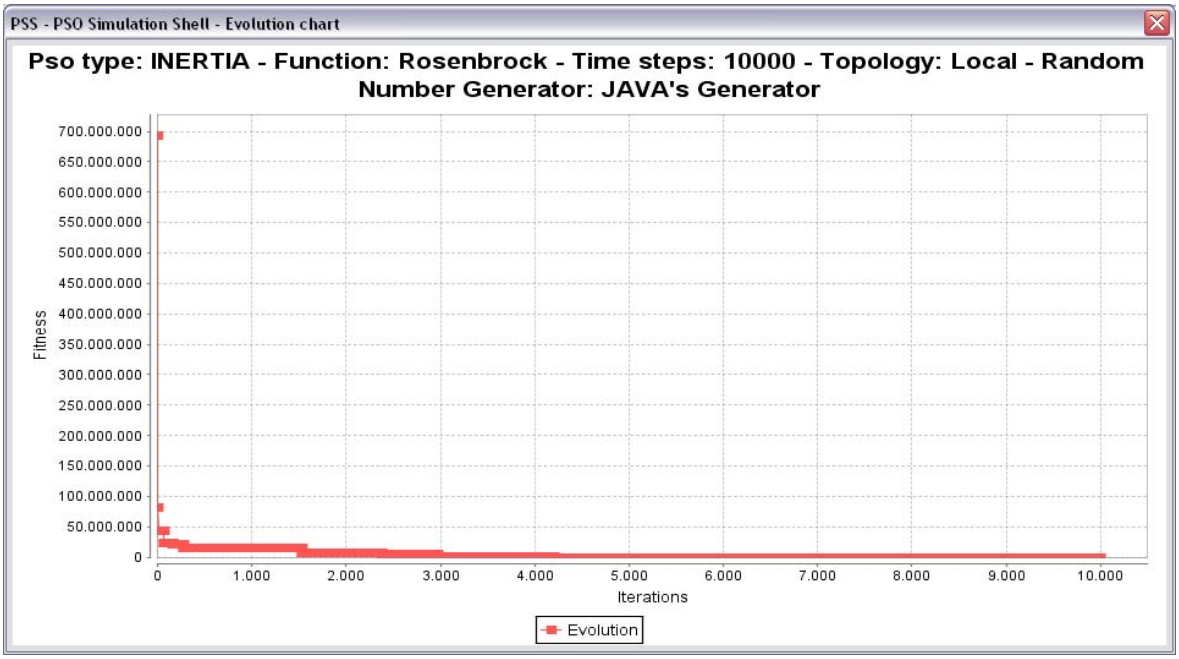


Figura 34. Gráfico de evolução com eixo linear.

O leitor pode observar a tela responsável pela geração da análise do Teste U na tela da Figura 37. Nos dois campos superiores, denominados *Control Sample* (Amostra de Controle) e *Test Sample* (Amostra de Teste) o usuário deve informar os diretórios onde se encontram os resultados das simulações. O sistema exibe diálogos que auxiliam na escolha dos diretórios e preenche os referidos campos com a escolha do usuário.

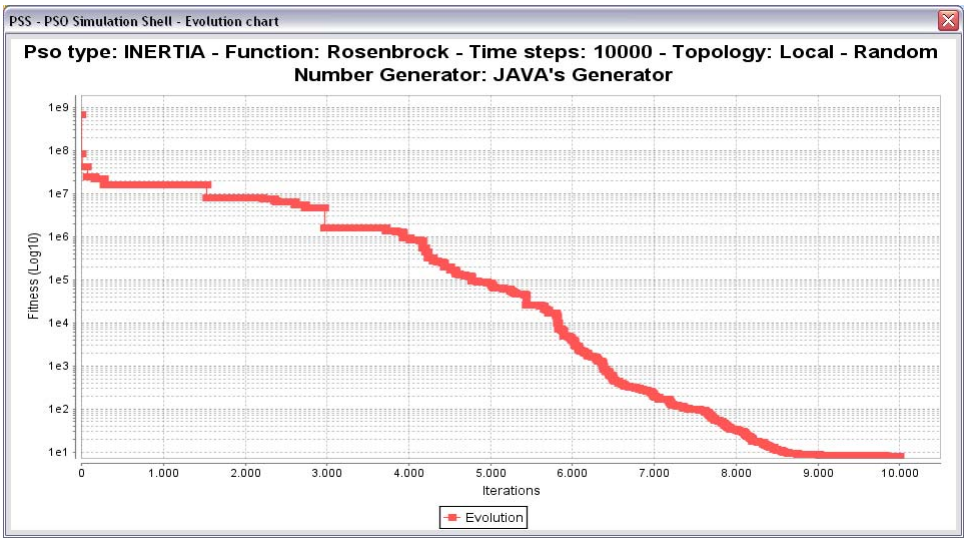


Figura 35. Gráfico de evolução com eixo logarítmico.

A seguir o usuário deve informar qual o hipótese que ele deseja testar. Existem três hipóteses disponíveis:

a) *Not Equal*: com esta hipótese o usuário deseja testar se as duas classes de amostras fazem ou não parte da mesma distribuição;
b) *Test is less then control*: com esta hipótese o usuário deseja testar se a amostra de teste é estocasticamente menor que a amostra de controle;
c) *Test is greater then control*: com esta hipótese o usuário deseja testar se a amostra de teste é estocasticamente maior que a amostra de controle.

O campo opcional *Use normal approximation* (Usar aproximação normal) indica ao sistema que o usuário deseja realizar o cálculo do teste utilizando uma aproximação normal, como visto na equação (17). Esta opção é útil quando o usuário deseja fazer uma análise sobre um conjunto muito grande de dados, pois o cáculo exato do *p*-valor é bastante custoso. Nestes casos, a aproximação normal fornece um valor bastante confiável e é consideravelmente mais simples de calcular.

O campo *Equality tolerance* (Tolerância de igualdade) é responsável por dizer ao sistema qual a tolerância para que ele considere que duas amostras são iguais. No caso demontrado na Figura 37, a presença do valor zero informa ao sistema que não há tolerância e as amostras devem ser consideradas iguais se todas as suas casas decimais forem iguais. Caso tivesse sido informado o valor 0,0001, os valores 1,3214, 1,3215 e 1,3213 seriam considerados iguais, o que configura um caso de empate.

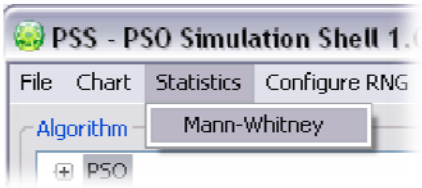


Figura 36. Menu *Statistics.*

O campo *Confidence level* (Nível de confiança) informa ao sistema o valor do nível confiança que deve ser utilizado para se dizer se o teste é significativo ou não. O valor 0,05 (5%) significa que a hipótese sendo testada deve ser rejeitada caso o *p*-

valor possua valor maior ou igual a 0,05, ou seja, para que o teste seja considerado significativo, é necessário que se tenha pelo menos 95% de certeza de que o resultado obtido não pode ser atribuído ao acaso.

Para a implementação do teste de Mann-Whitney, foi utilizada a API *Java Statistical Classes* [27].

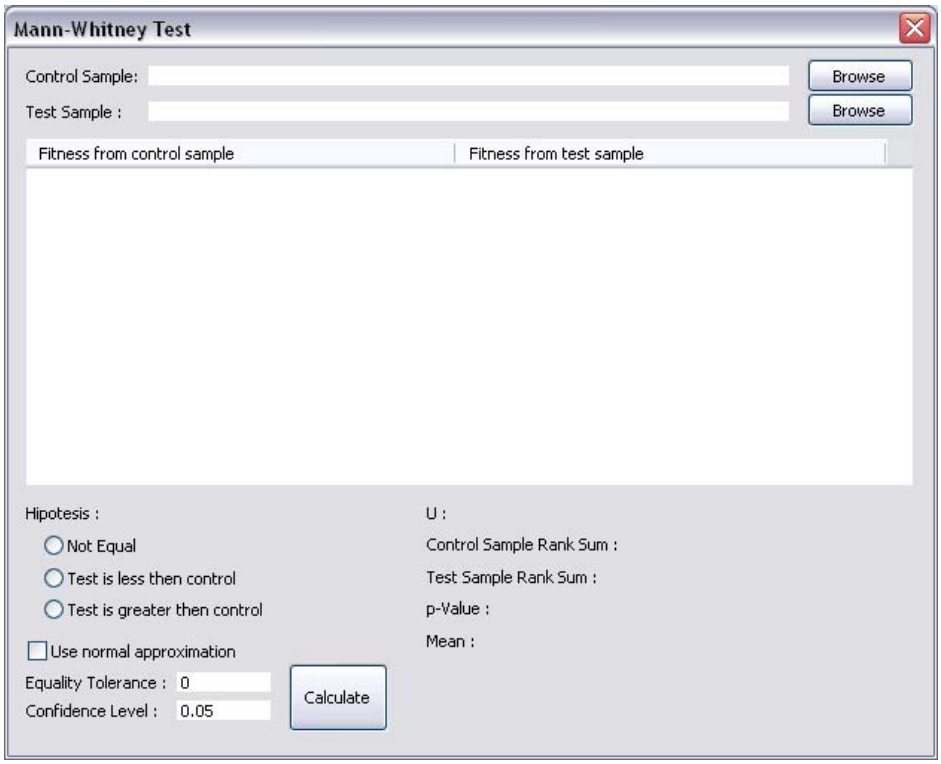


Figura 37. Tela do teste de Mann-Whitney.

Capítulo 5

Estudo de Caso

Neste capítulo serão realizados alguns estudos de caso com a finalidade de demonstrar as funcionalidades da ferramenta desenvolvida em um problema de simulação típico.

5.1 Testando a Topologia Global em Funções Unimodais e Multimodais

O estudo de caso a seguir visa investigar se a topologia Global [4] tem desempenho melhor em problemas unimodais devido à sua forma de comunicação. Intuitivamente, pode aceita-se que a topologia global tenha um desempenho pior que outras topologias menos agressivas em problemas multimodais pois as soluções tendem a ficar “presas” em mínimos locais.

O experimento foi configurado de forma que o algoritmo utilizou a topologia Global com 30 partículas. O tipo do PSO escolhido foi o *Inertia*, com os fatores para coeficiente de inércia inicial igual a 0,9 e final igual a 0,4. Foram utilizadas as funções Rosenbrock, Ackley, Esfera, Rastrigin e Schwefel 1.2. Os dados foram colhidos após 30 simulações, cada uma com 10000 iterações. Os parâmetros para as funções de teste são encontrados na Tabela 3.

Tabela 3. Configuração das funções de teste.

Função	Dimensões	Posição Inicial	Domínio	Velocidade Máxima
Rosenbrock	30	[-15;30]	[-30;30]	30,0
Ackley	30	[16;32]	[-32;32]	32,0
Esfera	30	[50;100]	[-100;100]	100,0
Rastrigin	30	[2,56;5,12]	[-5,12;5,12]	5,12

Schwefel 1.2	30	[50;100]	[-100;100]	100,0

A Tabela 4 demonstra os valores de média e desvio padrão do desempenho do algoritmo com a topologia Global para cada uma das funções de teste do experimento.

Tabela 4. Resultado da execução do algoritmo com topologia Global.

Função	Média (Aproximada)	Desvio Padrão (Aproximado)
Rosenbrock	46,0632799733997081	36,0829720163367895224 83
Ackley	1,2404891928478415E-14	3,787901940163675548E-15
Esfera	4,4586424473509220E-43	2,32030791954147028E-42
Rastrigin	15,786676317078907	4,976392017651984467363
Schwefel 1.2	4,83326351878064402	3,131132986412863861147

O leitor pode observar claramente, que a topologia Global tende a ter um desempenho muito superior em funções unimodais. No experimento acima, apenas a função Esfera é unimodal. Os gráficos de *box plot* podem ser vistos apartir da Figura 38 até a Figura 42. Os gráficos estão usando o eixo das ordenadas logarítmico e intervalo para capturas das amostras de 500.

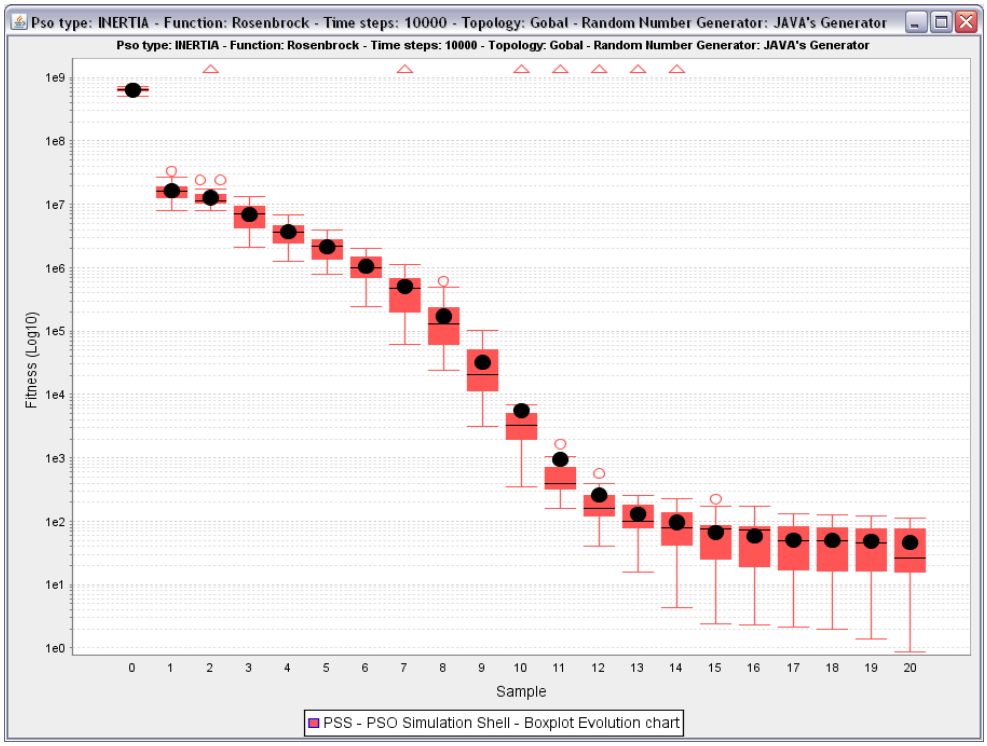


Figura 38. Evolução da topologia Global para a função Rosenbrock

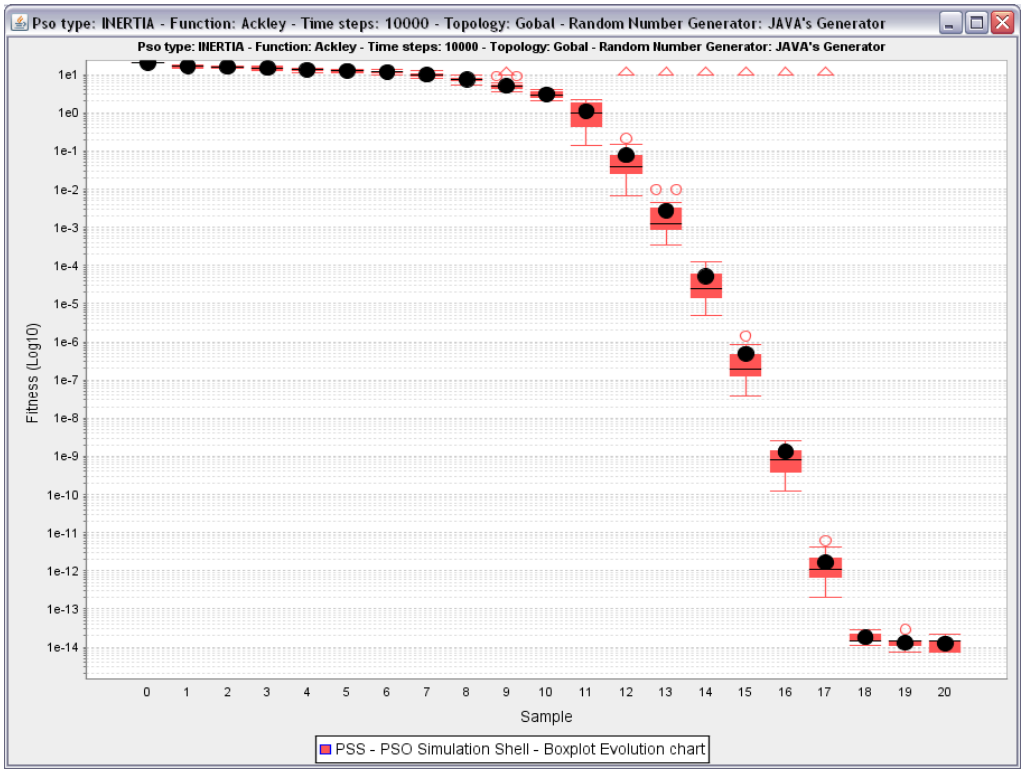


Figura 39. Evolução da topologia Global para a função Ackley.

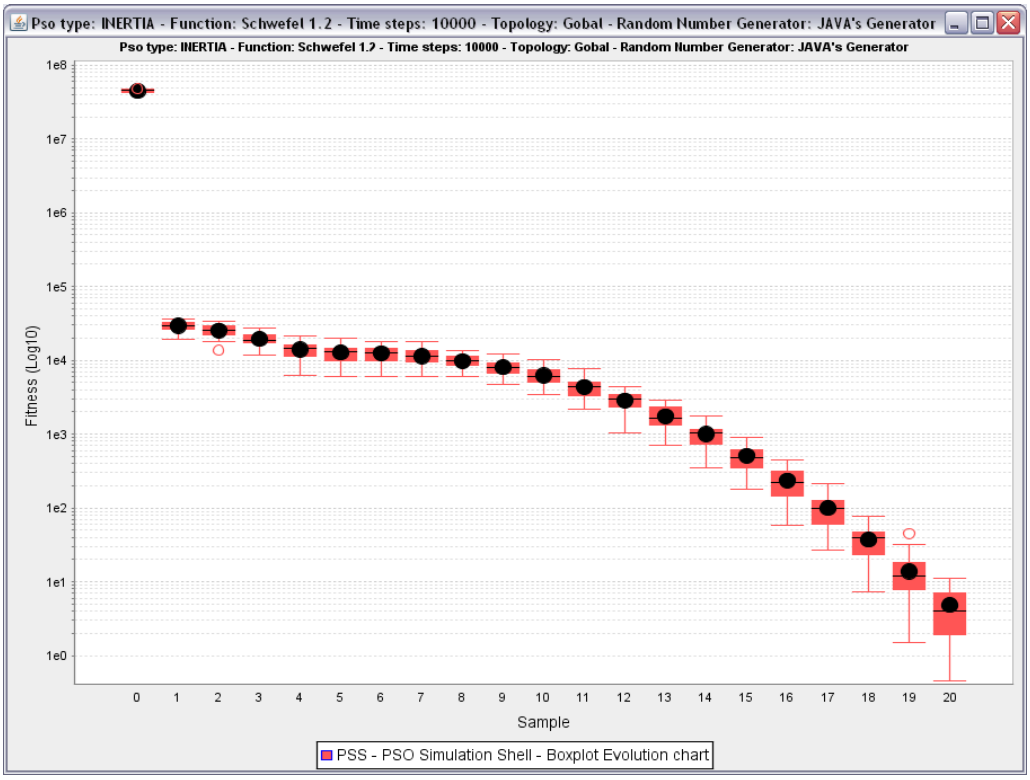


Figura 40. Evolução da topologia Global para a função Schwefel 1.2.

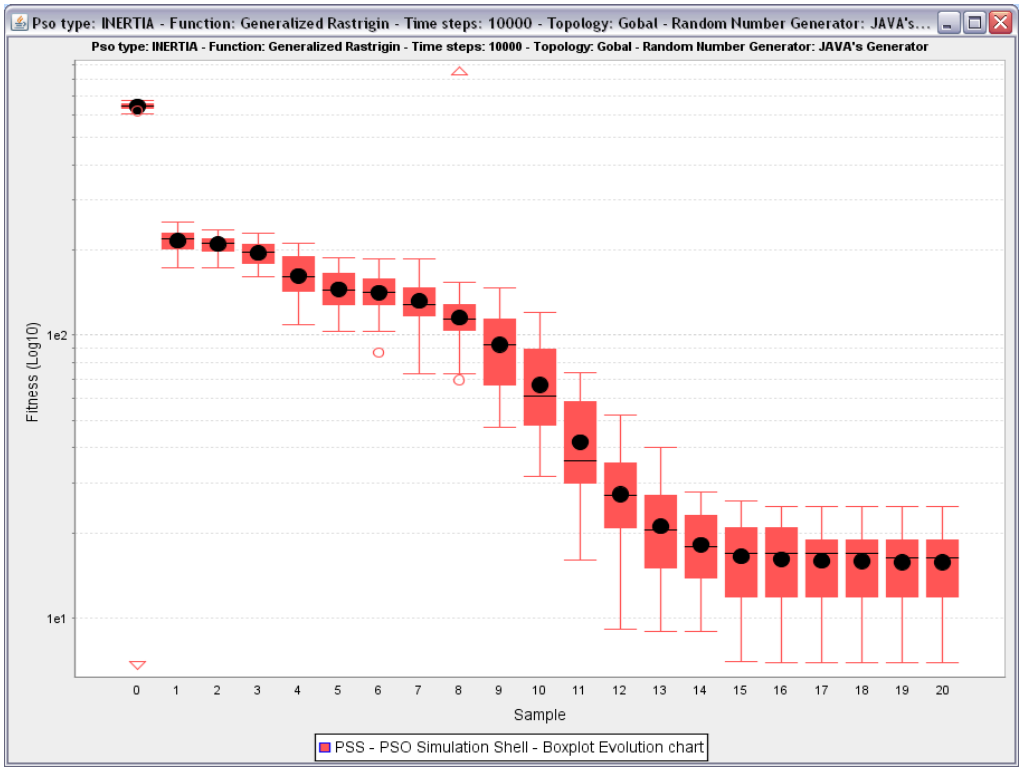


Figura 41. Evolução da topologia Global para a função Rastrigin.

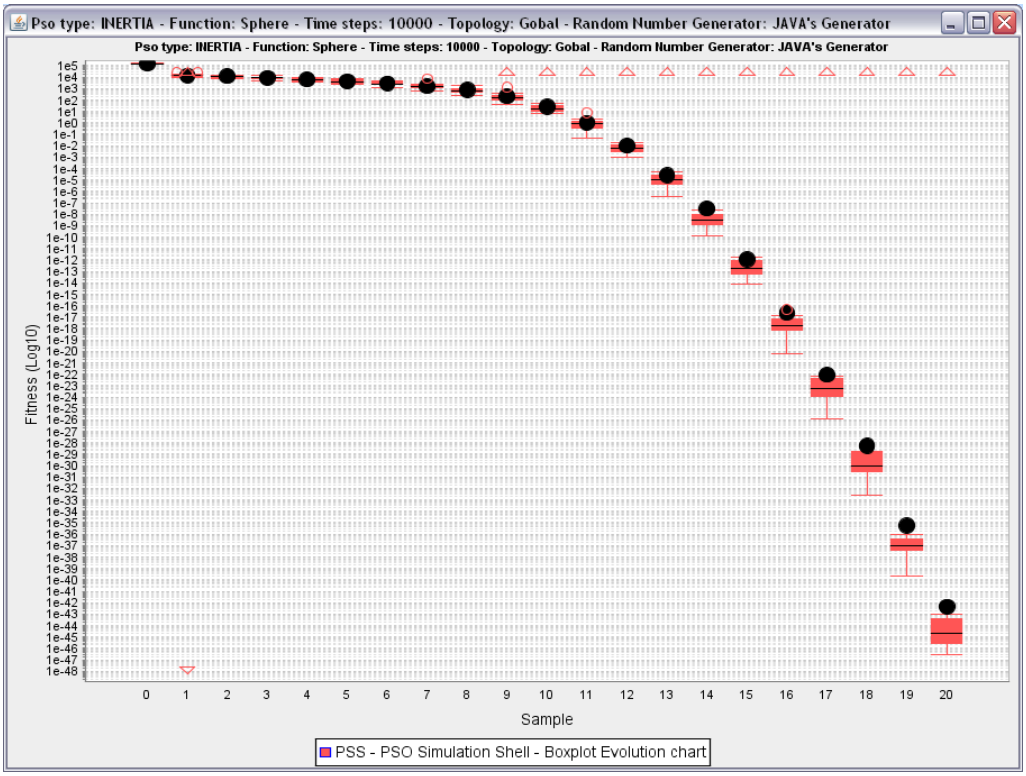


Figura 42. Evolução da topologia Global para a função Esfera.

Com os gráficos pode ser observado que o a topologia tende a ter um desempenho bem melhor em uma função unimodal. Além de conseguir chegar a um resultado bem melhor, o desempenho do enxame como um todo é bem mais uniforme durante a execução, como pode ser visto na curva da Figura 42. A caixa pequena confirma o desvio padrão pequeno. Como forma de comprovar isto pode-se realizar tambem o teste de Mann-Whitney. Por economia de espaço, foi realizado o teste entre o desempenho na função Esfera e o desempenho na função Ackley, que teve a segunda melhor média e desvio padrão. Seu resultado pode ser visto na Figura 43.

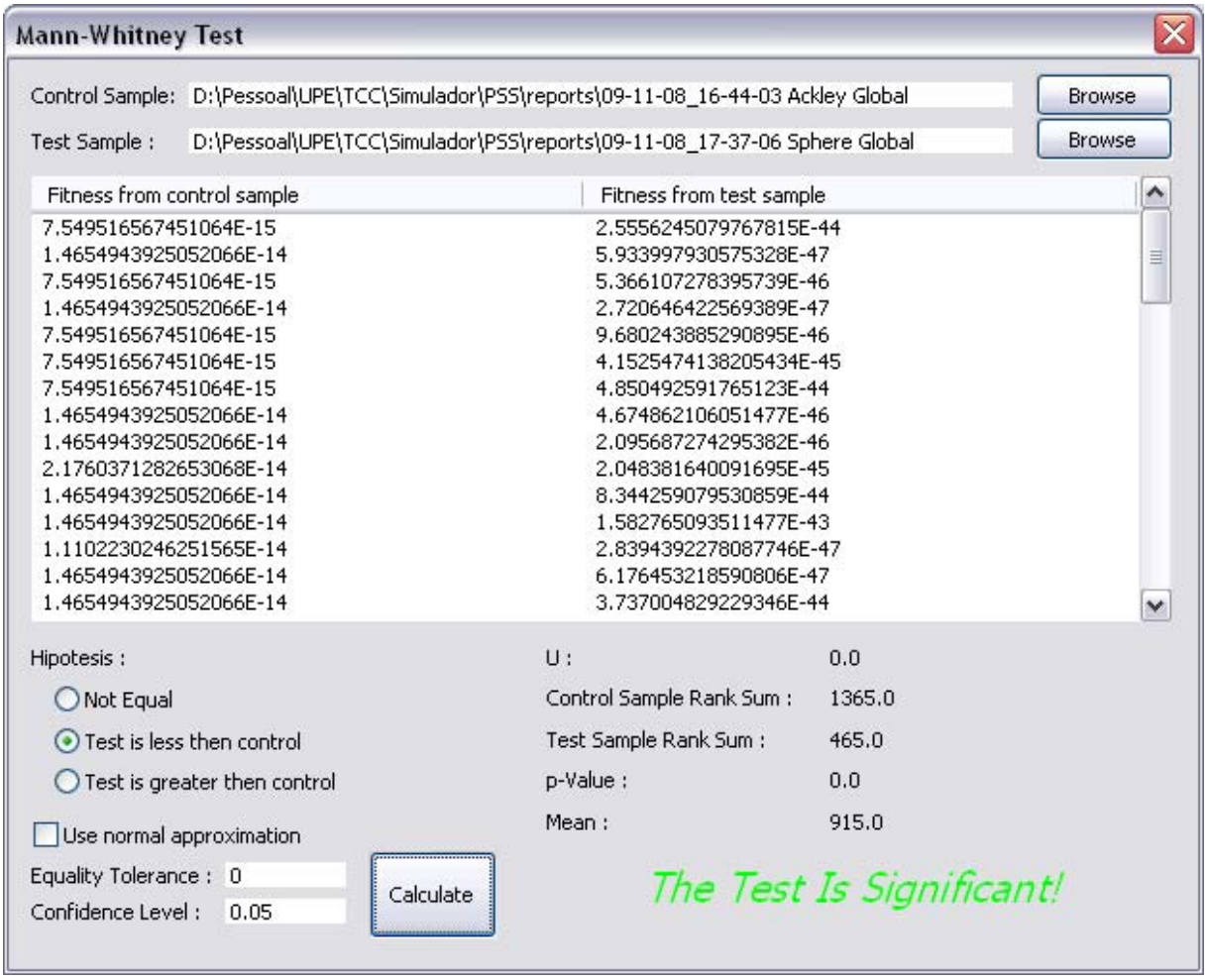


Figura 43. Resultado do teste de Mann-Whitney para as funções Ackley e Esfera.

O leitor pode visualizar todas as amostras que foram utilizadas na tabela exibida na tela do teste. O resultado do teste diz que para a hipótese selecionada (o conjunto de teste é menor que o conjunto de controle) existe 0% (*p*-value = 0.0) de chance de o resultado ter ocorrido por acaso, ou seja, a hipótese selecionada tem 100% de chance de estar correta.

5.2 Uma Análise da Topologia Multi-Ring Utilizando Fator de Constrição e Fator de Inércia

Nesta seção será feita uma análise acerca do desempenho do algoritmo com duas configurações específicas. Será analisada a topologia *Multi-Ring* (MR)

utilizando o fator de constrição e em um segundo momento utilizando o mecanismo de dispersão. A configuração das funções é a mesma utilizada na seção 5.1.

A Tabela 5 apresenta os valores de média e desvio padrão para a topologia *Multi-Ring* utilizando o fator de constrição. A

Tabela 6 apresenta os valores de média e desvio padrão para a topologia *Multi-Ring* utilizando o fator de inércia e mecanismo de dispersão ativo. Em ambos os casos foram realizadas 30 simulações, cada uma com 10000 iterações. A topologia Multi-Ring foi configurada de forma que cada anel contivesse 5 partículas, e usasse o valor 15 como limite de estagnação.

Tabela 5. Médias e devios padrão para a topologia *Multi-Ring* com fator de constrição.

Função	Média (Aproximada)	Desvio Padrão (Aproximado)
Rosenbrock	7,6662279813133328883	5,45323561163132541906861
Ackley	19,275161797479024918317	3,64050311670660908802688
Esfera	6,567903441990449829E-101	1,600459715356228251E-100
Rastrigin	35,321005346719431	7,39381478158431271197059
Schwefel 1.2	8,33346734554726786E-9	1,343220276826756753 88E-8

Tabela 6. Médias e desvios padrão para a topologia *Multi-Ring* utilizando fator de inércia e mecanismo de dispersão ativado.

Função	Média (Aproximada)	Desvio Padrão (Aproximado)
Rosenbrock	7,411549236591964775	7,02579878831134863048646
Ackley	7,0758214102776639E-15	1,2283362012573220601E-15
Esfera	3,739954425717899246E-138	2,040560839349539795E-137
Rastrigin	23,315190776467013	5,98507147313118181841673
Schwefel 1.2	4,28122366928480861E-9	6,831709413204920625 10E-9

Pode ser observado que a topologia MR tem um desempenho bastante interessante, pois ela possui características tanto da topologia Global quanto da Local, conseguindo realizar uma busca bastante balanceada. Isto pode ser verificado através dos valores do desvio padrão, que se mantêm sempre abaixo ou bastante próximo dos encontrados pelas topologias Global no experimento anterior.

Em alguns casos, o MR com mecanismo de dispersão tem um desempenho muito superior a outras topologias, até mesmo o próprio MR utilizando fator de constrição. Por exemplo, na função Esfera, o MR com dispersão consegue desempenho e desvio padrão da ordem 10^{-138} enquanto que o MR com fator de constrição obtém desempenho e desvio padrão da ordem de 10^{-100}. Em ambos os casos, o desempenho do MR foi bastante superior ao da topologia Global, que consegui desempenho de 10^{-40} no experimento anterior.

A partir da Figura 44 até a Figura 48 são exibidos os gráficos de *box plot* para a topologia MR com fator de constrição.

É possível observar que a topologia MR com fator de constrição conseguiu um desempenho muito bom para a função Esfera. Como a função Esfera é considerada “fácil” de ser solucionada, a análise aqui se restringe a velocidade de convergência. Vê-se que a curva de convergência na função esfera é bastante acentuada. Atente para o fato de que ao gráfico está configurado para exibir o eixo das ordenadas logarítmico. As caixas do *box plot* com uma amplitude pequena reforçam a análise do desvio padrão baixo.

A partir da Figura 49 até a Figura 53 são exibidos os gráficos de *box plot* para a topologia MR com fator de inércia e mecanismo de dispersão ativado. Novamente, na função Esfera a topologia tem o melhor desempenho, mas como a função Esfera analisa velocidade pode-se concluir analisando-se os gráficos da Figura 48 e Figura 53, que na primeira o enxame converge bem mais rapidamente que na segunda embora na segunda o enxame tenha obtido desempenho melhor em média

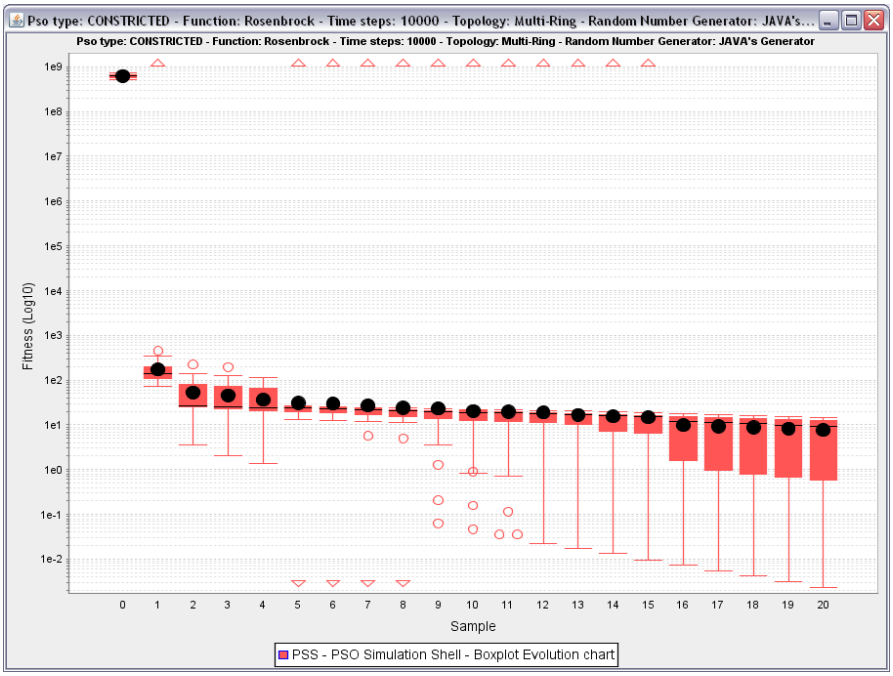


Figura 44. *Box plot* para topologia MR com fator de constrição para as função Rosenbrock.

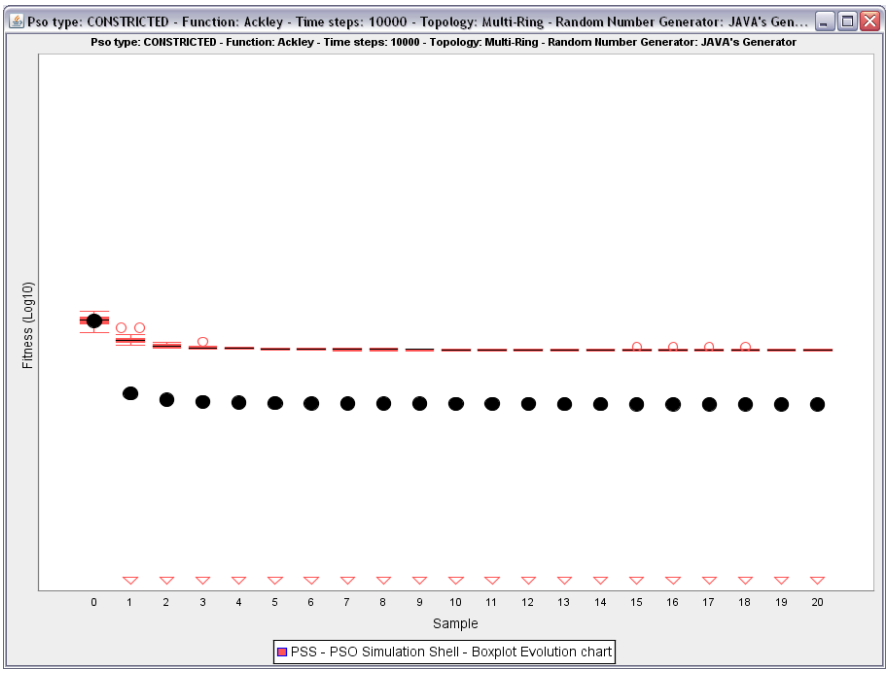


Figura 45. *Box plot* para topologia MR com fator de constrição para as função Ackley.

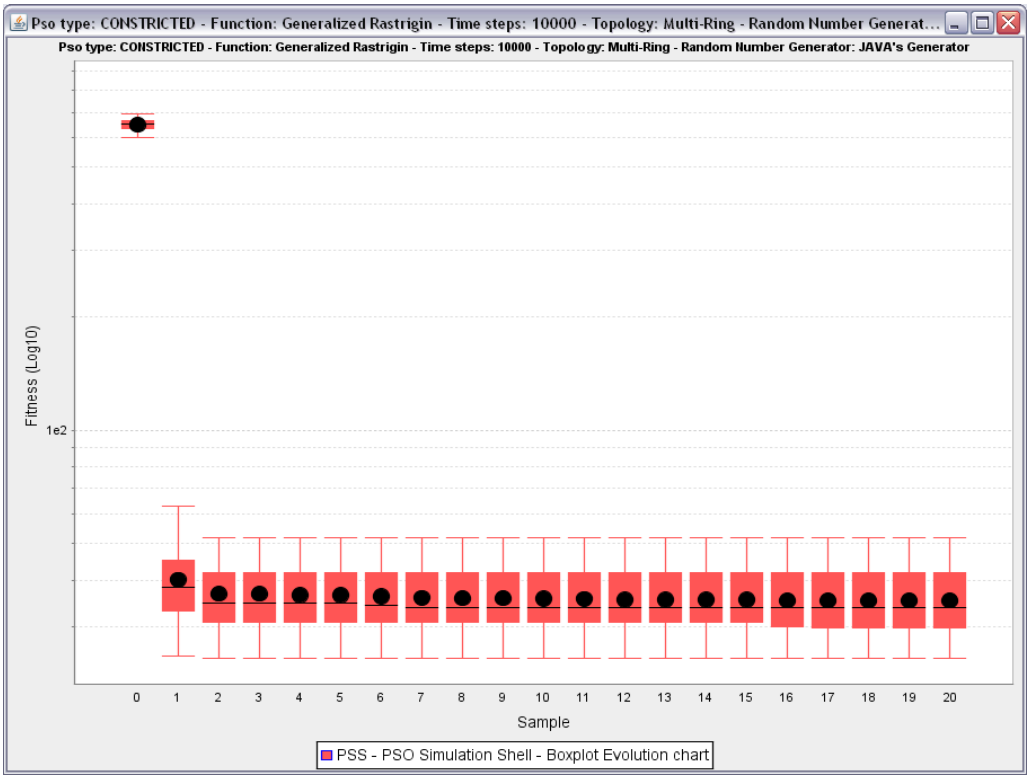


Figura 46. *Box plot* para topologia MR com fator de constrição para as função Rastrigin.

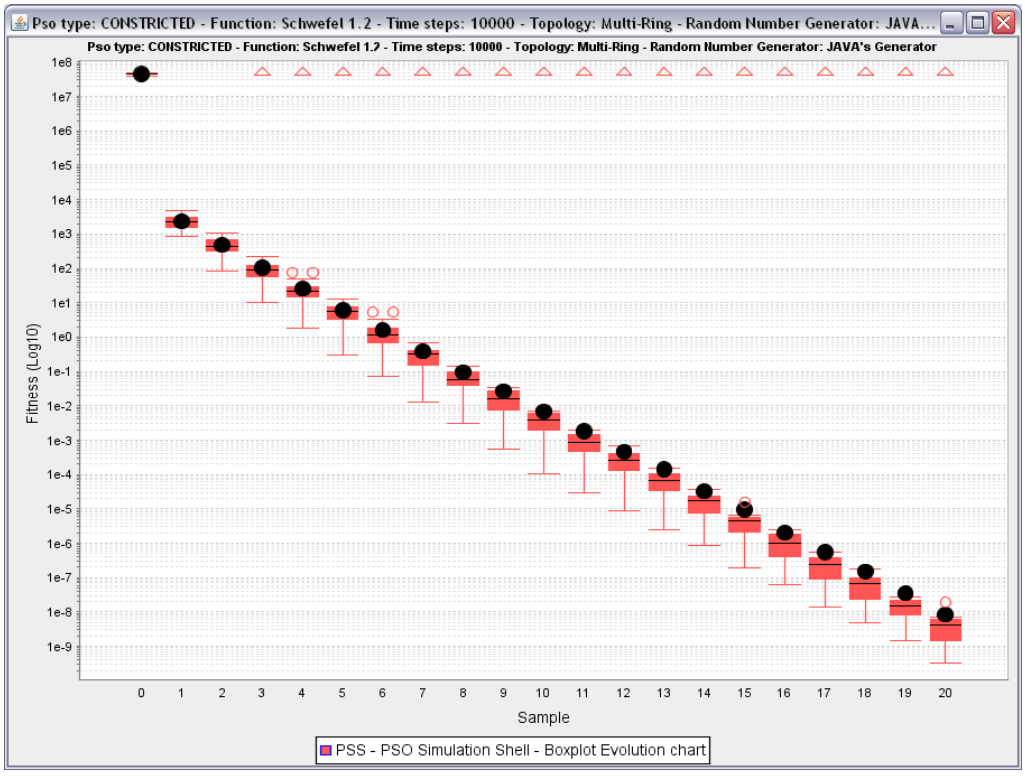


Figura 47. *Box plot* para topologia MR com fator de constrição para as função Schwefel 1.2.

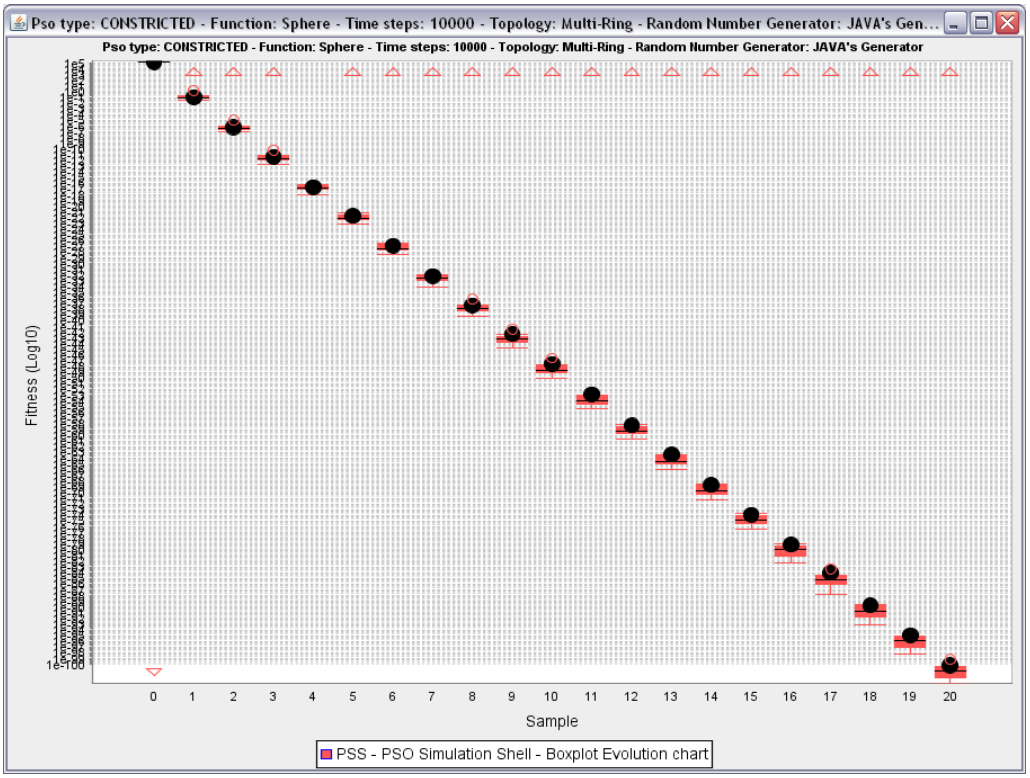


Figura 48. *Box plot* para topologia MR com fator de constrição para as função Esfera.

A análise de significância neste exemplo foi feita para a função Schwefel 1.2, pois nas duas configurações do experimento os enxames obtiveram média e desvio padrão bastante próximos. Este cenário é bastante interessante para se mostrar o teste de significância em ação, pois apenas analisando os gráficos ou o desvio padrão e média é bastante difícil decidir qual configuração tem melhor desempenho.

Como pode ser visto na Figura 54, a configuração do MR com dispersão teve um desempenho melhor. Esta afirmação tem um grau de confiabilidade de aproximadamente 99,995% (1 – *p*-valor).

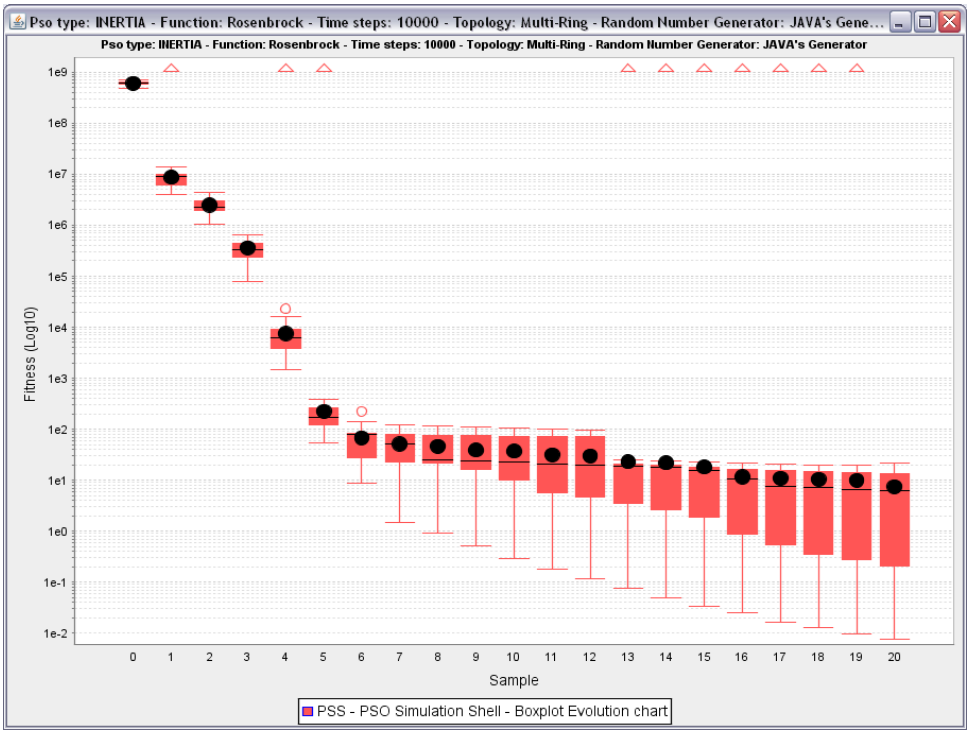


Figura 49. *Box plot* para a topologia MR fator de inércia e mecanismo de dispersão ativado para a função Rosenbrock.

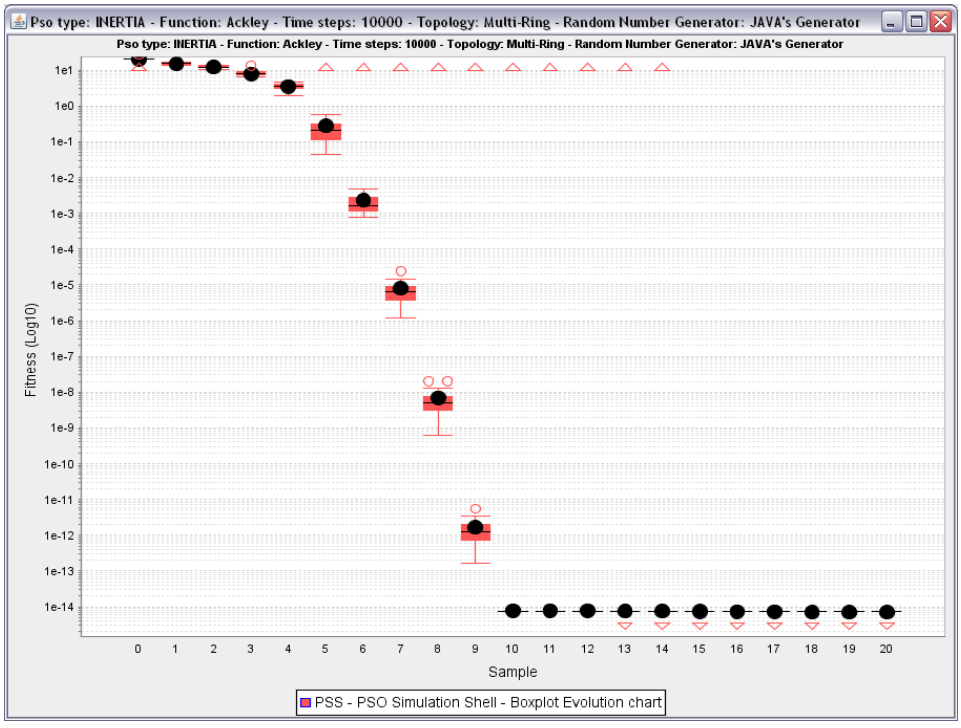


Figura 50. *Box plot* para a topologia MR fator de inércia e mecanismo de dispersão ativado para a função Ackley.

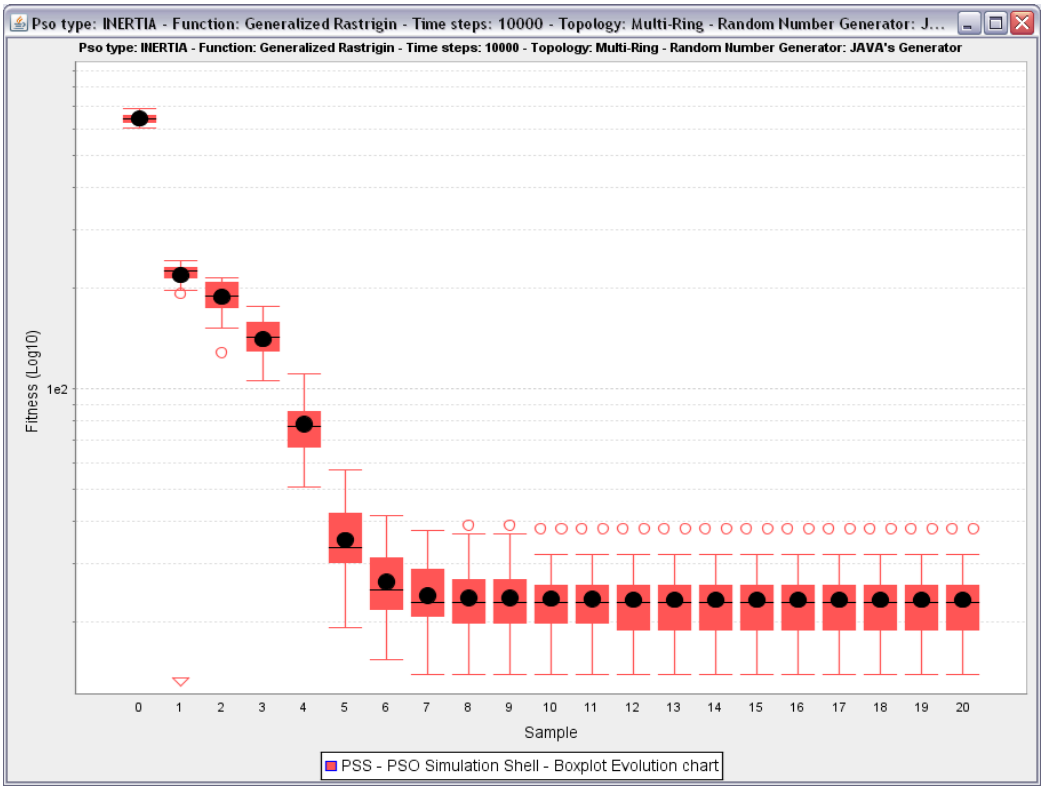


Figura 51. *Box plot* para a topologia MR fator de inércia e mecanismo de dispersão ativado para a função Rastrigin.

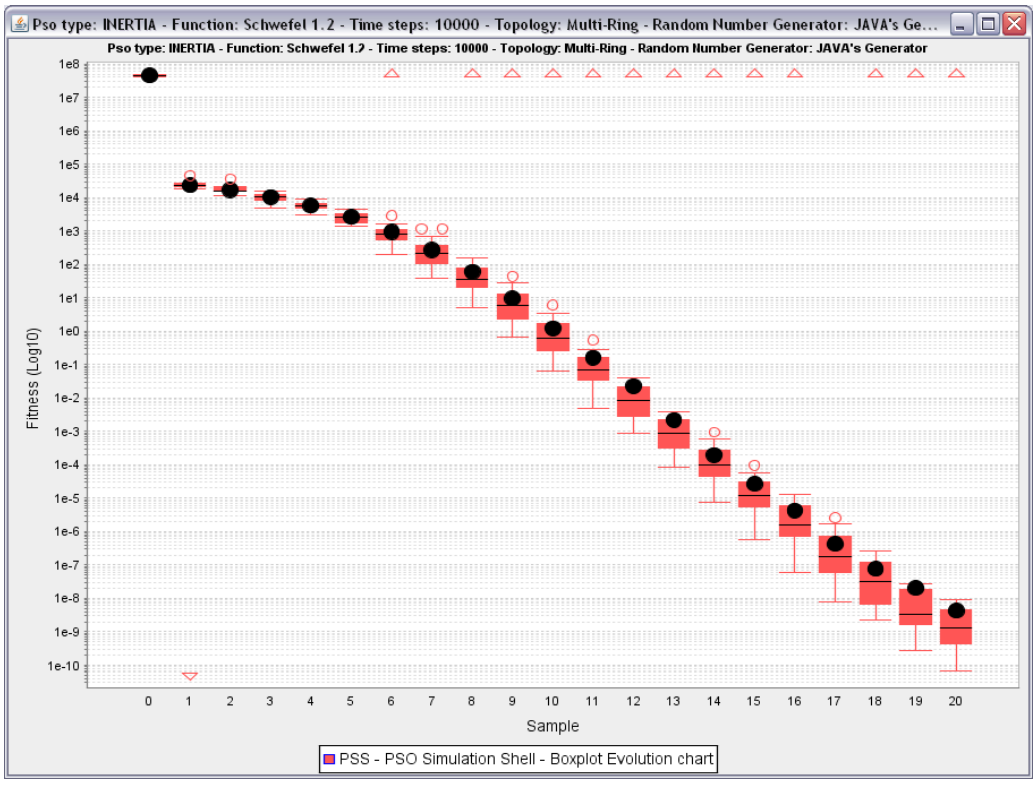


Figura 52. *Box plot* para a topologia MR fator de inércia e mecanismo de dispersão ativado para a função Scwefel 1.2.

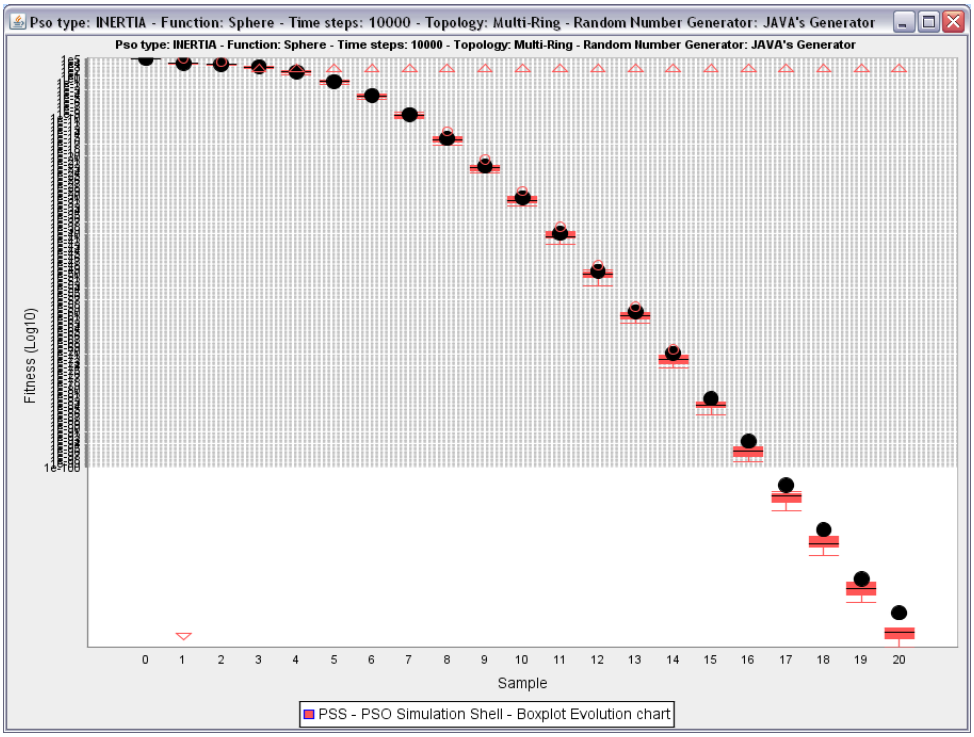


Figura 53. *Box plot* para a topologia MR fator de inércia e mecanismo de dispersão ativado para a função Esfera.

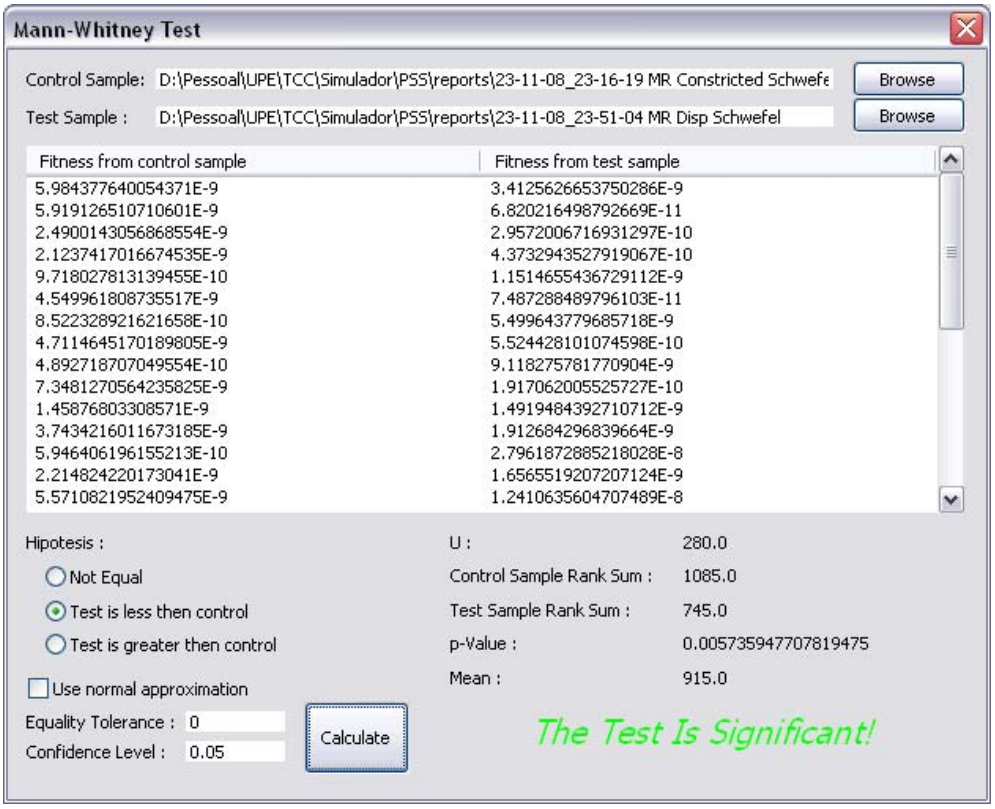


Figura 54. Teste de significância para MR com dispersão e MR com fator de constrição para a função Schwefel 1.2.

Capítulo 6

Conclusão e Trabalhos Futuros

Este trabalho se propõe a apresentar o PSS (*Particle Swarm Optimization Simulation Shell*), um simulador para algoritmos de otimização por enxame de partículas. A ferramenta foi desenvolvida com o objetivo de fornecer um ambiente bastante rico e amigável de simulação e análise de resultados.

Neste capítulo serão discutidas as contribuições, conclusões e trabalhos futuros para a possível extensão do presente trabalho.

6.1 Conclusão e Contribuições

A solução de problemas de otimização através da implementação de técnicas matemáticas clássicas geralmente é computacionalmente muito cara. Os algoritmos de otimização por enxame de partículas surgiram com o intuito de resolver de forma mais rápida alguns desses problemas. Tomando como base o trabalho seminal de Kennedy e Eberhart [4], pesquisadores foram capazes de desenvolver muitos trabalhos a avançar bastante na área.

Quase em sua totalidade, os trabalhos desenvolvidos neste campo dependem de simulações e observações sobre uma determinada proposta. Desde uma nova topologia até a análise do efeito da qualidade dos números aleatórios exigem que sejam realizados experimentos e comparações entre a nova proposta e o estado da arte.

Naturalmente, neste ambiente inerentemente experimental, a existência de uma ferramenta que agregue os conceitos de topologia, tipos de PSO, simulações e análises estatísticas é de grande valia. O PSS, fruto deste trabalho, é capaz de realizar tudo que foi citado acima, além de fornecer um mecanismo simples de adição de novas funcionalidades devido à tecnologia utilizada para sua construção.

Antes da existência desta ferramenta, por exemplo, o grupo de pesquisa em que o autor se insere tinha de implementar novamente quase todo o sistema para que fossem realizados novos estudos. Como muitas vezes, não se seguia o mesmo padrão ou parametrização, o resultado era um sistema pouco consistente. Com a introdução do PSS, o grupo de pesquisa (e a comunidade científica em geral) tem um ambiente muito mais estável e robusto para realização dos seus estudos.

Como principais contribuições, podem ser enumeradas:

a) o tempo necessário para o desenvolvimento de uma nova solução foi drasticamente reduzido, dado que a ferramenta já possui um grande arcabouço que pode ser reutilizado em trabalhos futuros, como a interface de interação com o usuário e os módulos de análise estatística;
b) a partir de agora é bem mais fácil guardar o histórico de estudos realizados anteriormente, pois a ferramenta se utiliza de um mecanismo padrão para geração dos resultados, o que facilita muito no momento de realizar as análises;
c) aumento da capacidade de análise de resultados, pois as ferramentas de análise estatística podem cobrir rapidamente grandes massas de dados;
d) além de fornecer um ferramental bastante completo para o desenvolvimento de trabalhos científicos, a ferramenta pode ser facilmente usada por docentes para o ensino de conceitos de técnicas de otimização por enxame de partículas.

Foi realizado um estudo de caso que demonstra as funcionalidades da ferramenta proposta. Foi analisada a eficácia da topologia global em problemas unimodais e multimodais. Foi realizada também uma análise sobre o desempenho da topologia Multi-Ring utilizando o mecanismo de dispersão.

Espera-se, que em breve, as facilidades geradas pela ferramenta sejam compartilhados com a comunidade científica.

6.2 Trabalhos Futuros

Como trabalho futuro, pode-se implementar mais variações do PSO. Neste trabalho foram implementadas apenas a *Charged PSO* e a *Fully Informed*. A existência de mais variações proporcionaria ao usuário um ambiente ainda mais flexível, haja vista que uma quantidade muito maior de comportamentos poderão ser simulados.

Poderia-se implementar uma arquitetura mais avançada do sistema, especificamente um que pudesse suportar o conceito de *plugins*. O desenvolvimento de novas topologias, por exemplo, seria suportado por uma série de bibliotecas fornecidas pela própria ferramenta. O desenvolvedor conduziria seu projeto de forma que ao final de sua implementação, um pequeno módulo seria gerado. Este módulo poderia ser imediatamente adicionado à ferramenta sem a necessidade de recompilação da mesma. Esta estratégia é bastante interessante pois isola o núcleo do sistema e permite que os mecanismos periféricos possam ser alterados livremente. Esta opção é possível pois a estrutura básica do algoritmo do PSO não muda, apenas o comportamento de certos blocos. De fato, uma arquitetura assim foi pensada inicialmente, mas por restrições de tempo, uma estratégia mais simples foi adotada, o que não comprometeu o objetivo do trabalho.

Outro ponto em que a ferramenta poderia ser melhorada é o módulo de simulação. Hoje, como está implementada, a ferramenta não toma total proveito de ambientes multi-processados. Poderiam ser feitas também algumas modificações a fim de serem utilizadas tecnologias de como RMI [28] ou CORBA® [29] para que o sistema pudesse executar em uma ambiente computacional distribuído, como por exemplo um *cluster* ou *grid*.

Por fim, a ferramenta fornece gráficos de evolução para uma determinada partícula. No entanto, os gráficos fornecem apenas a evolução de uma única partícula. Seria interessante que o gráfico pudesse agrupar a evolução de duas ou mais partículas para que fosse mais fácil verificar diferenças entre elas. O gráfico de *box plot* também poderia ser aplicado a mais de um conjunto de simulações.

Bibliografia

[1] ZITZLER, E.; DEB, K. e THIELE, L. **Comparison of multiobjective evolutionary algorithms: Empirical results**. Em: Evolutionary Computation, 8(2), p. 173-195, 2000.

[2] KNOWLES, J. e CORNE, D. **Approximating the non-dominated front using the pareto achieved evolution strategy.** Em: Evolutionary Computation, 8(2), p. 149-172, 2000.

[3] LAUMANNS, M.; ZITZLER, E. e THIELE, L. **A unified model for multi-objective evolutionary algorithms with elitism.** Em: Congress on Evolutionary Computation (CEC), p. 46-53, 2000.

[4] KENNEDY, J. e EBERHART, R. **Particle Swarm Optimization.** Em: Proceedings of IEEE International Conference on Neural Networks, p. 1942-1948, 1995.

[5] BLACKWELL, T.M. e BENTLEY, P. J. **Dynamic Search with Charged Swarms.** Em: Proceedings of the Genetic and Evolutionary Computation Conference, p. 19-26, 2002.

[6] VAN DER BERGH, F. e ENGELBRECHT, A. P. **A Cooperative Approach to Particle Swarm Optimization.** Em: IEEE Transactions on Evolutionary Computation, v. 8, n. 3, p. 225-239, 2004.

[7] MANN, H. B. e WHITNEY, D. R. **On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other.** Em: Annals of Mathematical Statistics, v. 18, n. 1, p. 50-60, mar. 1947.

[8] REYNOLDS, C. W. **Flocks, Herds and Schools: a Distributed Behavioral Model.** Em: Computer Graphics, v. 21, n. 4, p. 25-34, 1987.

[9] MILLONAS, M. M. **Swarms, Phase Transitions and Collective Intelligence.** Em: C. G. Langton, Ed. Artificial Life III, Addison Wesley, Reading, MA. 1994.

[10] WILSON, E. O. **Sociobiology: The new Synthesis.** 1 ed. Cambridge, Massachusetts: Belknap Press, 1975.

[11] EBERHART, R. e KENNEDY, J. **A New Optimizer Using Particle Swarm Theory.** Em: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, p. 39-43, 1995.

[12] CLERC, M. e KENNEDY, J. **The Particle Swarm – Explosion, Stability, and Convergence in a Multidimensional Complex Space.** Em: IEEE Transactions on Evolutionary Computation, v. 6, n. 1, p. 58-73, 2002.

[13] ENGELBRECHT, A. P. **Computational Intelligence: An Introduction.** 2 ed. Estados Unidos: Wiley, 2007. 628 p.

[14] JANSON, S. e MIDDENDORF, M. **A Hierarchical Swarm Optimizer and Its Adaptive Variant.** Em: IEEE Transactions on Systems, Man, and Cybernetics, v. 35, n. 6, p. 1272-1282, 2005.

[15] KENNEDY, J. e MENDES, R. **Population Structure and Particle Swarm Performance.** Em: Proceedings of the IEEE Conference on Evolutionary Computation, p.1671-1676, 2002.

[16] BASTOS-FILHO, C. J. A.; CARACIOLO, M.; MIRANDA, P. e CARVALHO, D. **Multi Ring Particle Swarm Optimization.** Em: Simpósio Brasileiro de Redes Neurais, 2008.

[17] MENDES, R.; KENNEDY, J. e NEVES, J. **Watch the Neighbor or How The Swarm Can Learn From Its Environment.** Em: Proceedings of The IEEE Swarm Intelligence Symposium, p. 88-94, 2003.

[18] CLERC, M. **Back to Random Topology.** Relatório Técnico, mar. 2007, disponível em: < http://clerc.maurice.free.fr/pso/random_topology.pdf > Acesso em: 28 de dezembro de 2008.

[19] A. Hedar's Homepage. **Test Functions for Unconstrained Global Optimization.** Disponível em: <http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page364.htm> Acesso em: 19 de outubro de 2008.

[20] BLACKWELL, T. M. e BENTLEY, P. J. **Dynamic Search with Charged Swarms.** Em: Proceedigns of the Genetic and Evolutionary Computation Conference, p. 19-26, 2002.

[21] MENDES, R.; KENNEDY, J. e NEVES, J. **The Fully Informed Particle Swarm: Simpler, Maybe Better.** Em: IEEE Transactions on Evolutionary Computation, v.8, n.3, p.204-210, 2005.

[22] NEYMAN, J. e PEARSOM, E. S. N. **On the Use and Interpretation of Certain Test Criteria for Purposes of Statistical Inference, Part I.** Em: Joint Statistical Papers, Cambridge University Press, p. 1-66, 1967.

[23] MORETTIN, P. A. e BUSSAB, W. de O. **Estatística Básica.** 5 ed. São Paulo. Editora Saraiva, 2002.

[24] CARVALHO, D. F. e BASTOS-FILHO, C. J. A. **Clan Particle Swarm Optimization.** Em: IEEE World Congress on Computational Intelligence, p. 3044-3051, 2008.

[25] CAI, X.; CUI, Z.; ZENG, J e TAN, Y **Dispersed Particle Swarm Optimization.** Em: Elsevier Information Processing Letters, v. 105, p. 231-235, 2008.

[26] BRATTON, D. e KENNEDY, J. **Defining a Standard for Particle Swarm Optimization.** Em: Proceedings of the IEEE Swarm Intelligence Symposium, p. 120-127, 2007.

[27] JSC Home Page. **Java Statistical Classes.** Disponível em: <http://www.jsc.nildram.co.uk/>. Acesso em 20 de novembro de 2008.

[28] Remote Method Invocation Home. **Remote Method Invocation (RMI).** Disponível em: < http://java.sun.com/javase/technologies/core/basic/rmi/index.jsp>. Acesso em: 20 de novembro de 2008.

[29] CORBA® FAQ. **CORBA® Basics.** Disponível em: <http://www.omg.org/gettingstarted/corbafaq.htm>. Acesso em : 20 de novembro de 2008.

Apêndice A

Funções de Otimização

Nesta seção o leitor encontrará o gráfico para as funções de otimização tratadas durante o texto.

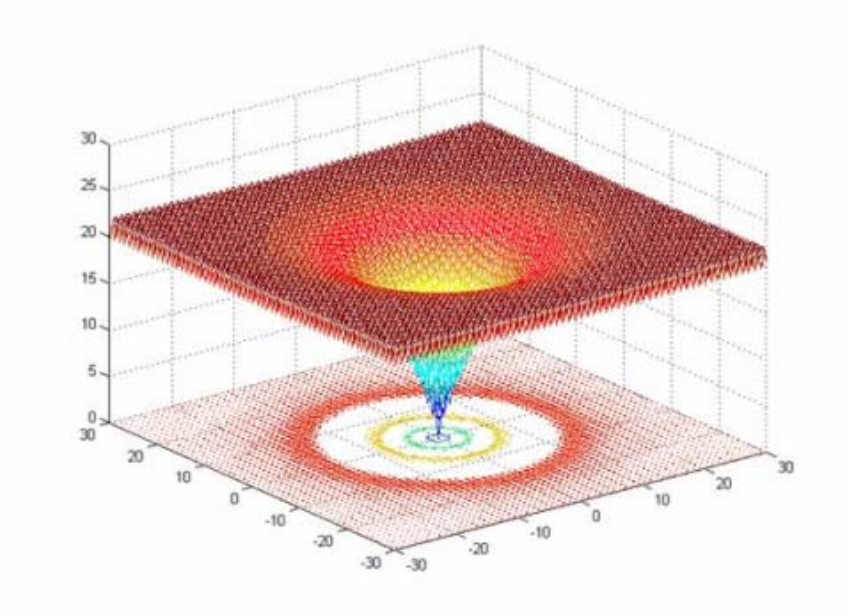


Figura 55. Função Ackley.

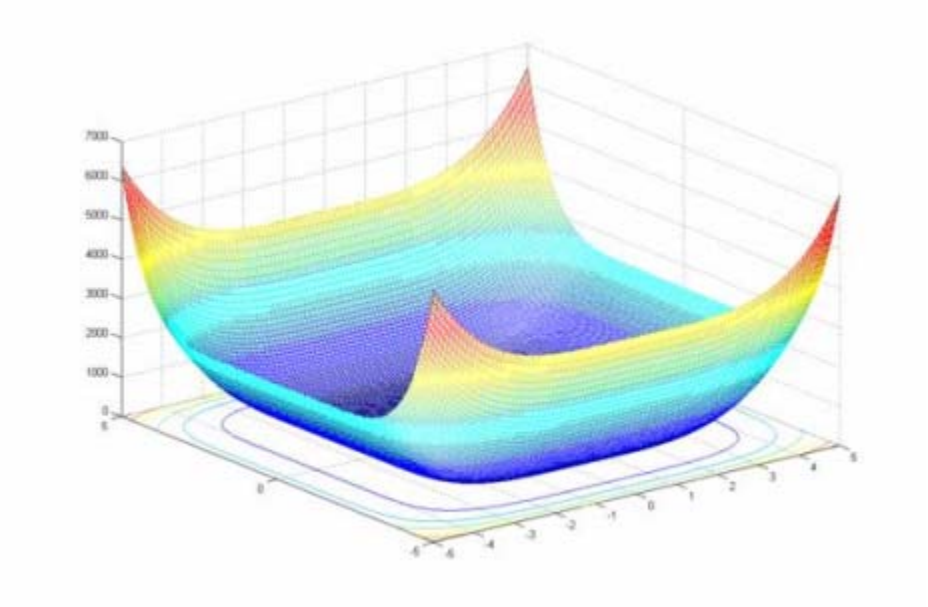


Figura 56. Função Six Hump Camel Back.

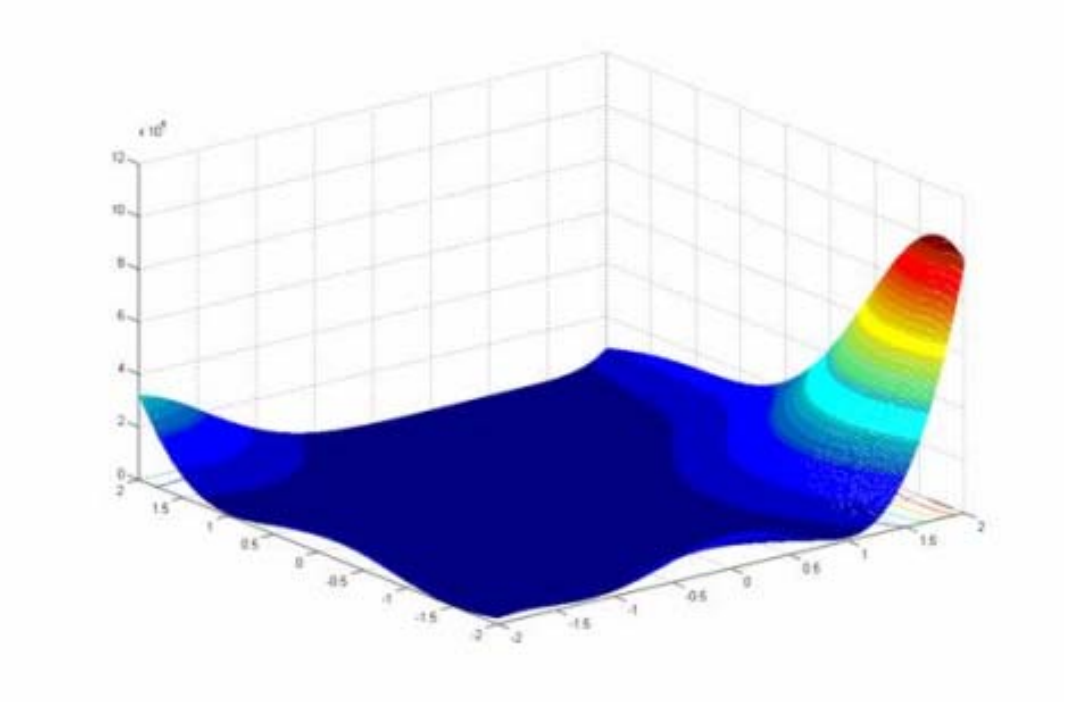


Figura 57. Função Goldstein & Price.

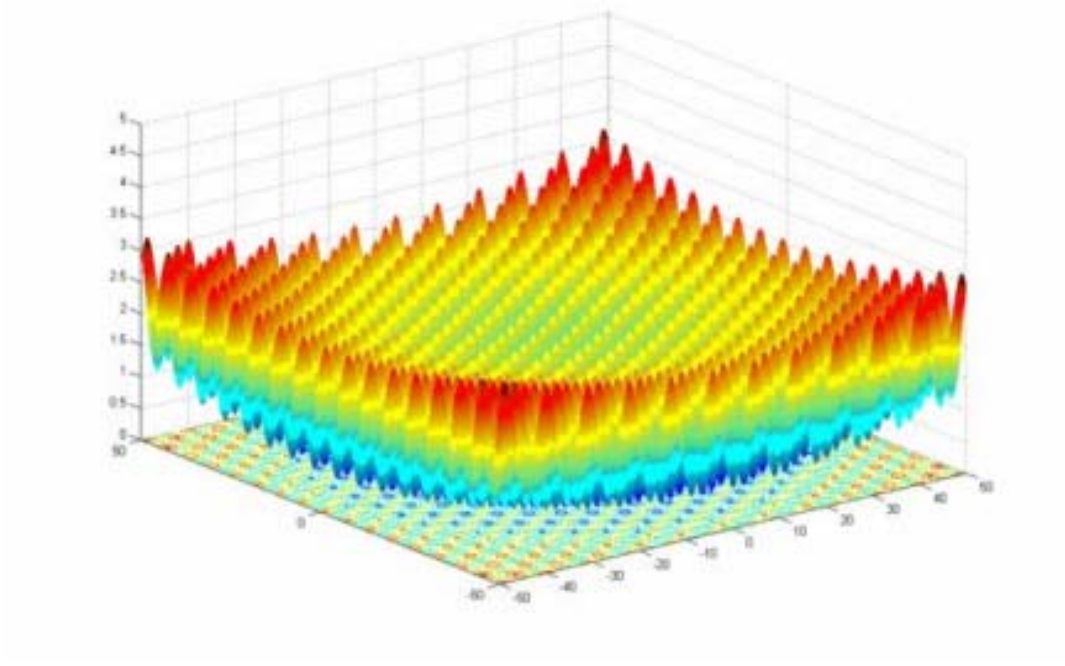


Figura 58. Função Griewank.

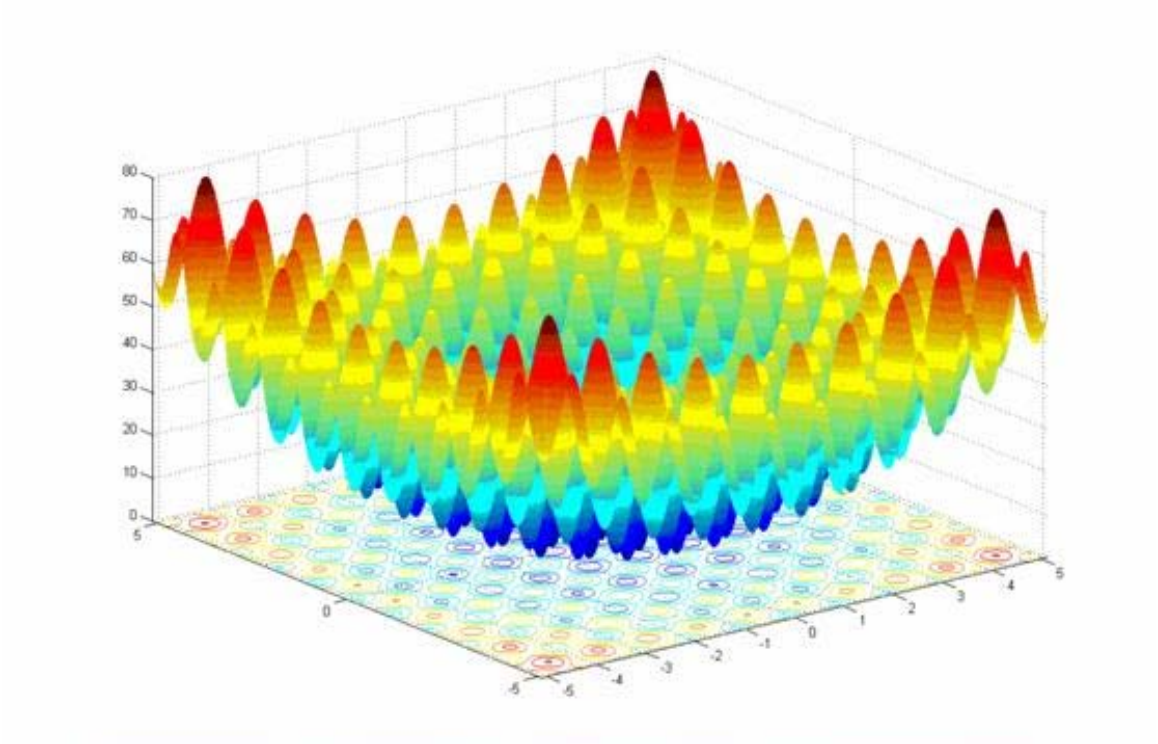


Figura 59. Função Rastrigin.

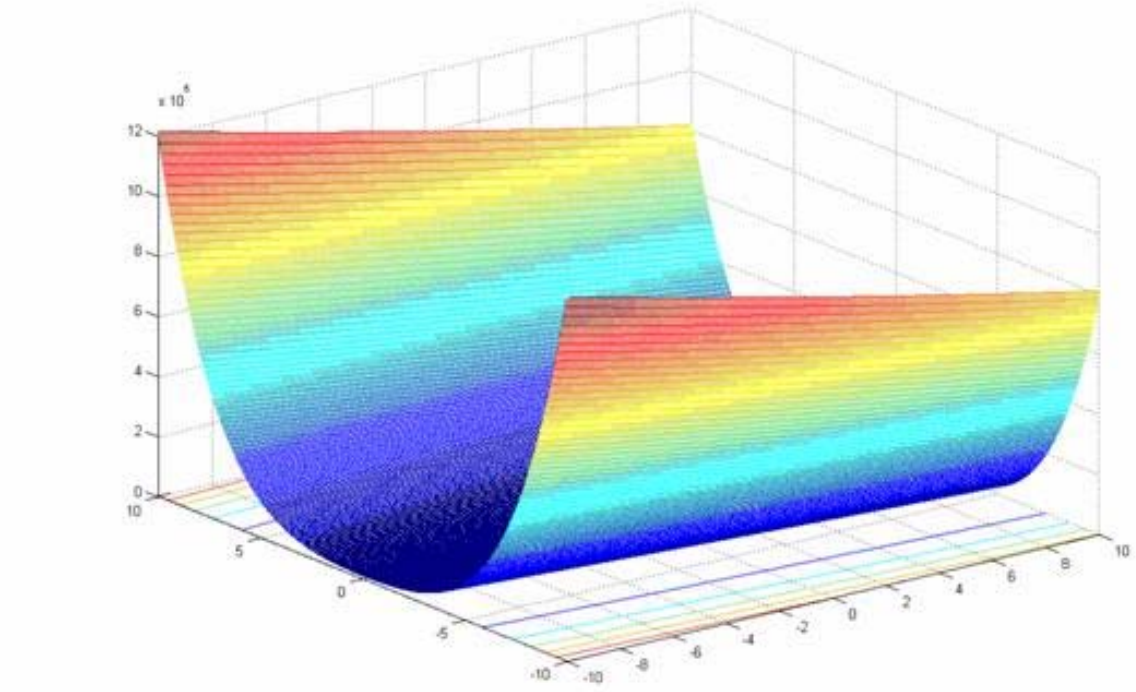


Figura 60. Função Rosenbrock.

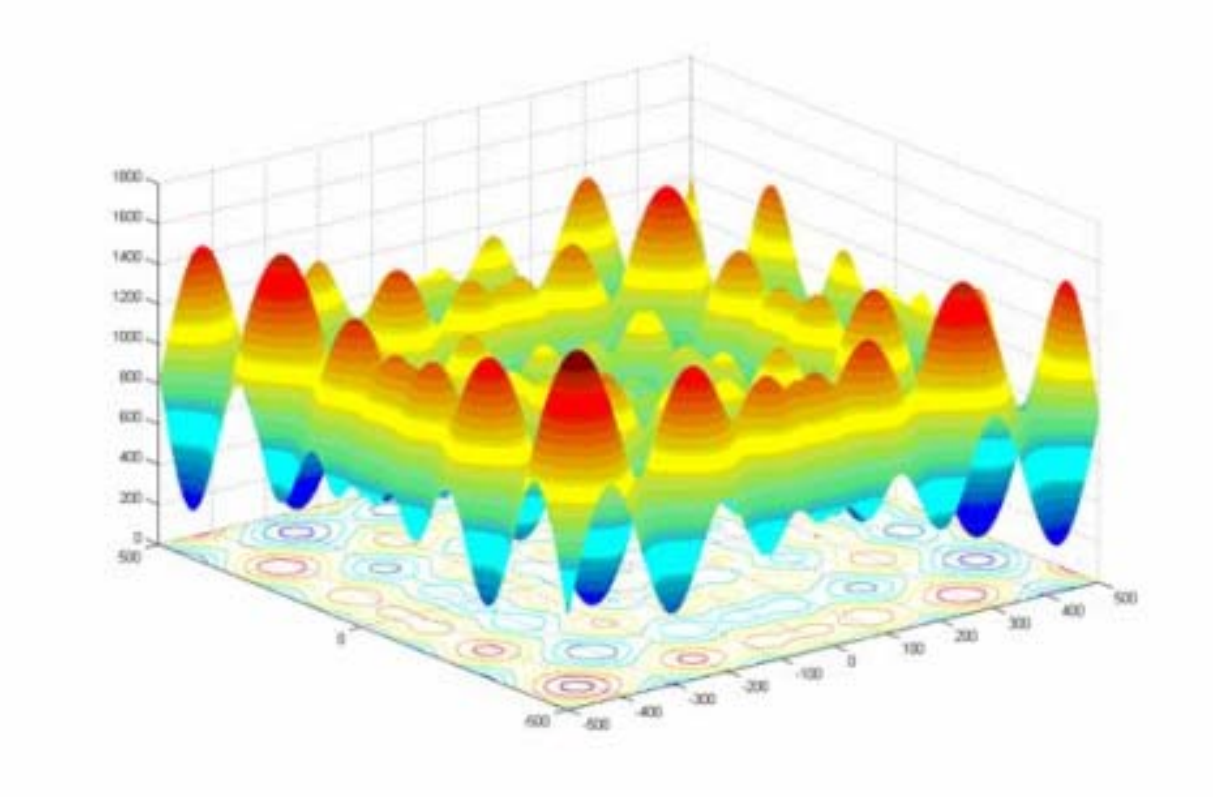


Figura 61. Função Schwefel.

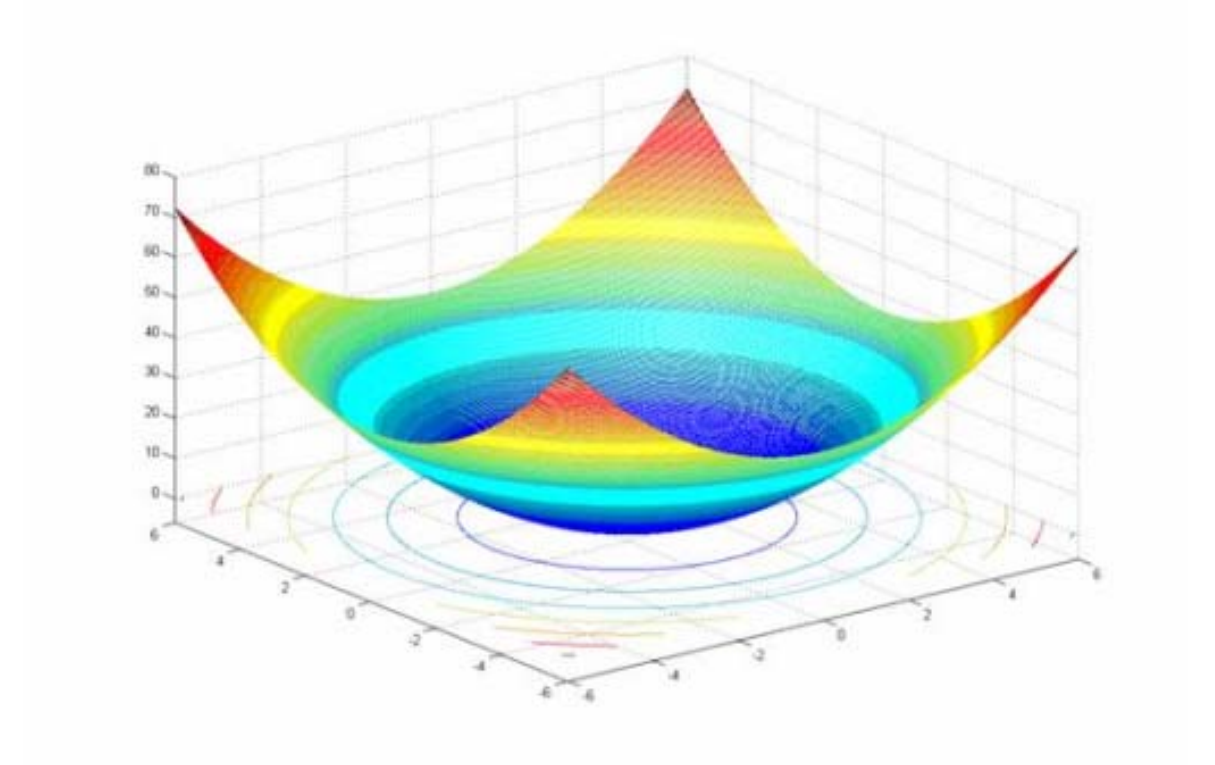


Figura 62. Função Esfera.

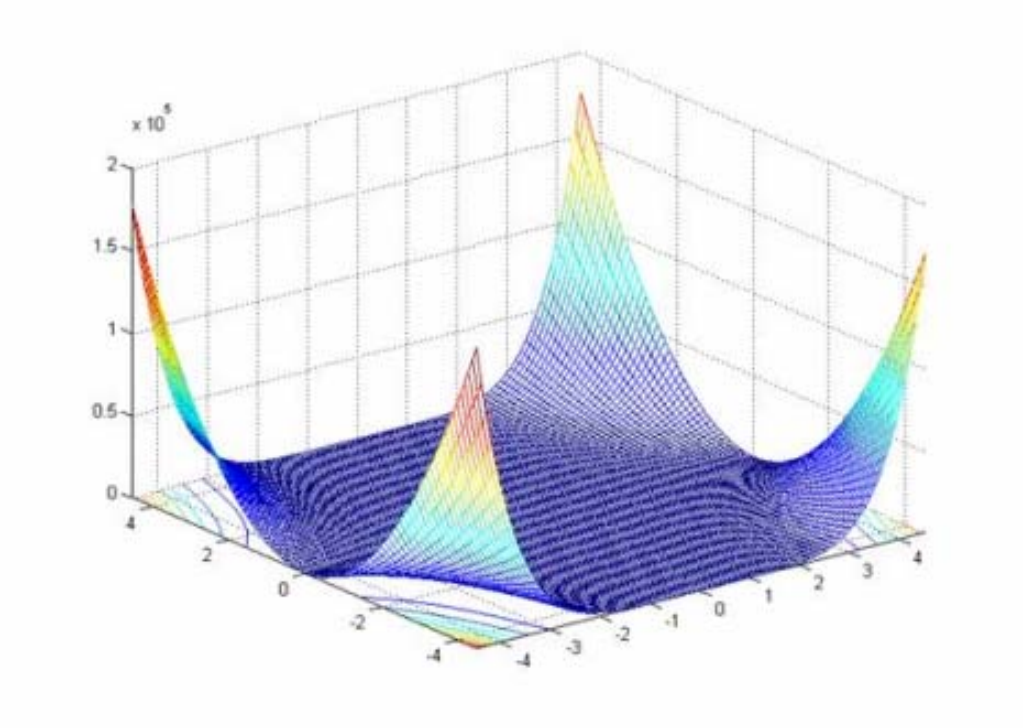


Figura 63. Função Beale.

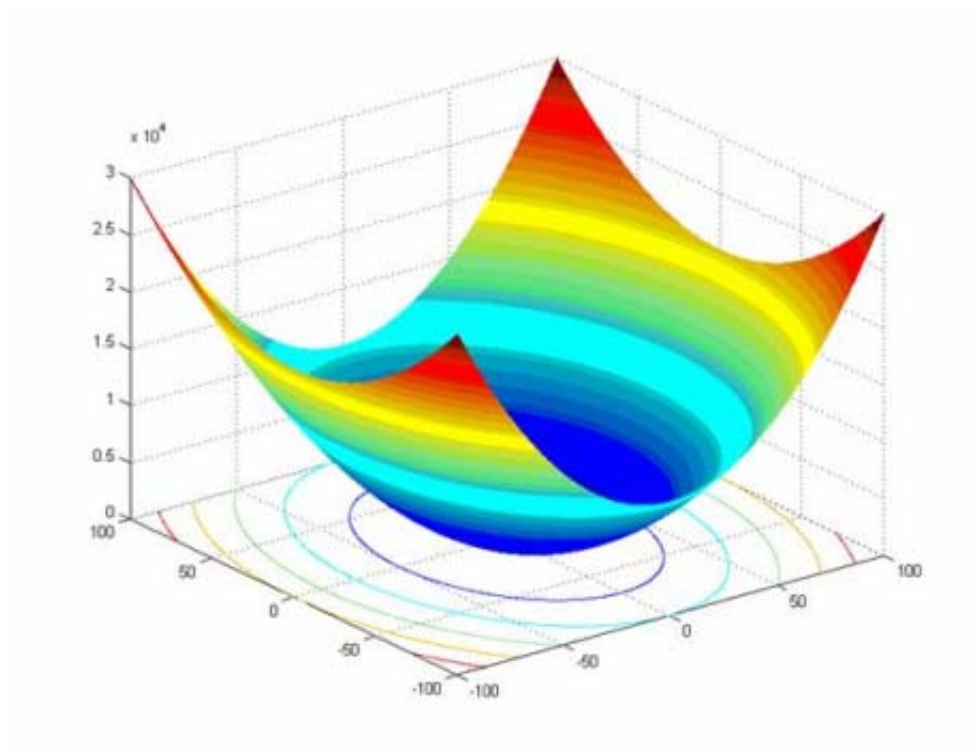


Figura 64. Função Bohachevsky.

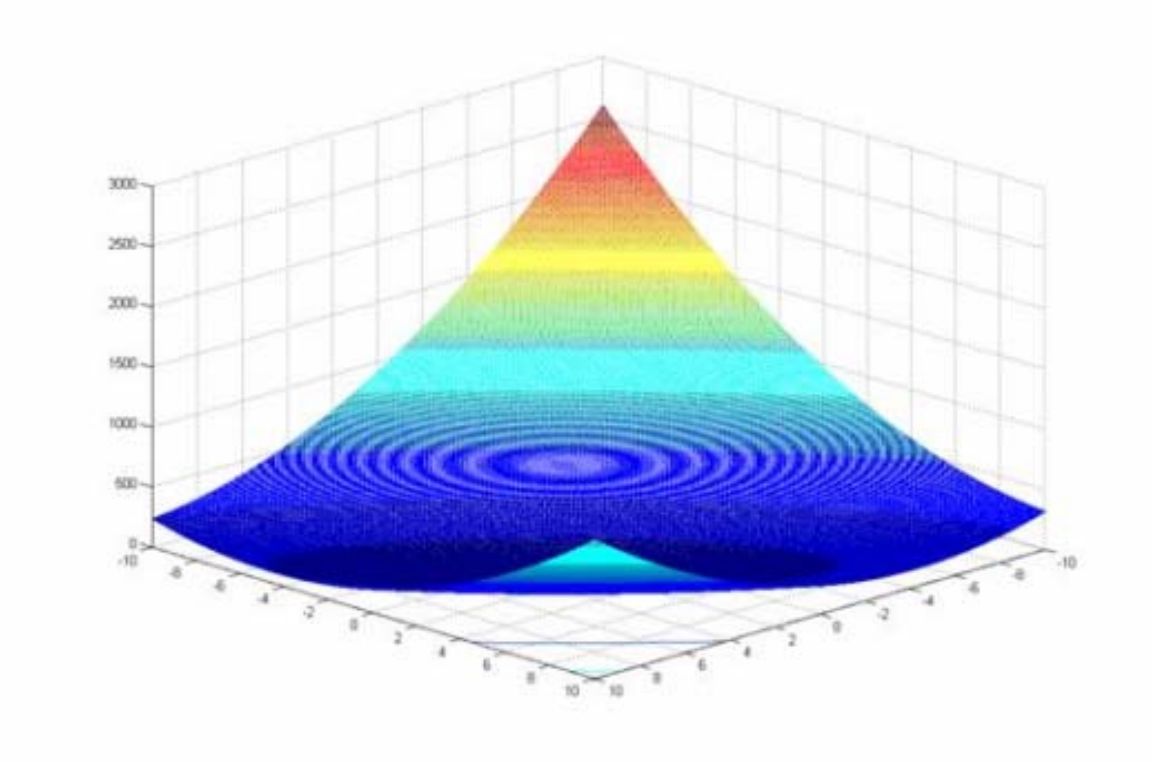


Figura 65. Função Booth.

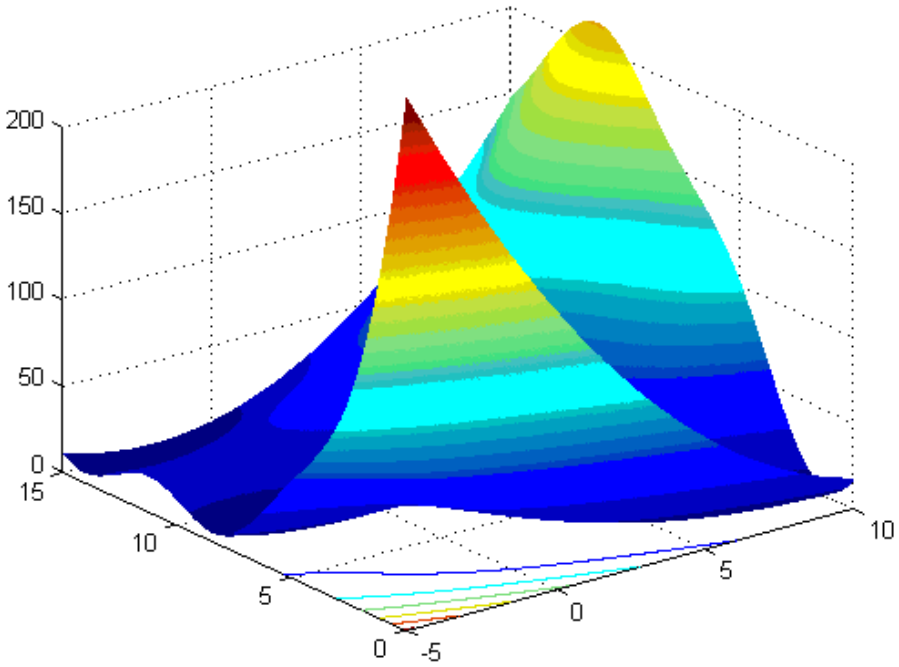


Figura 66. Função Branin.

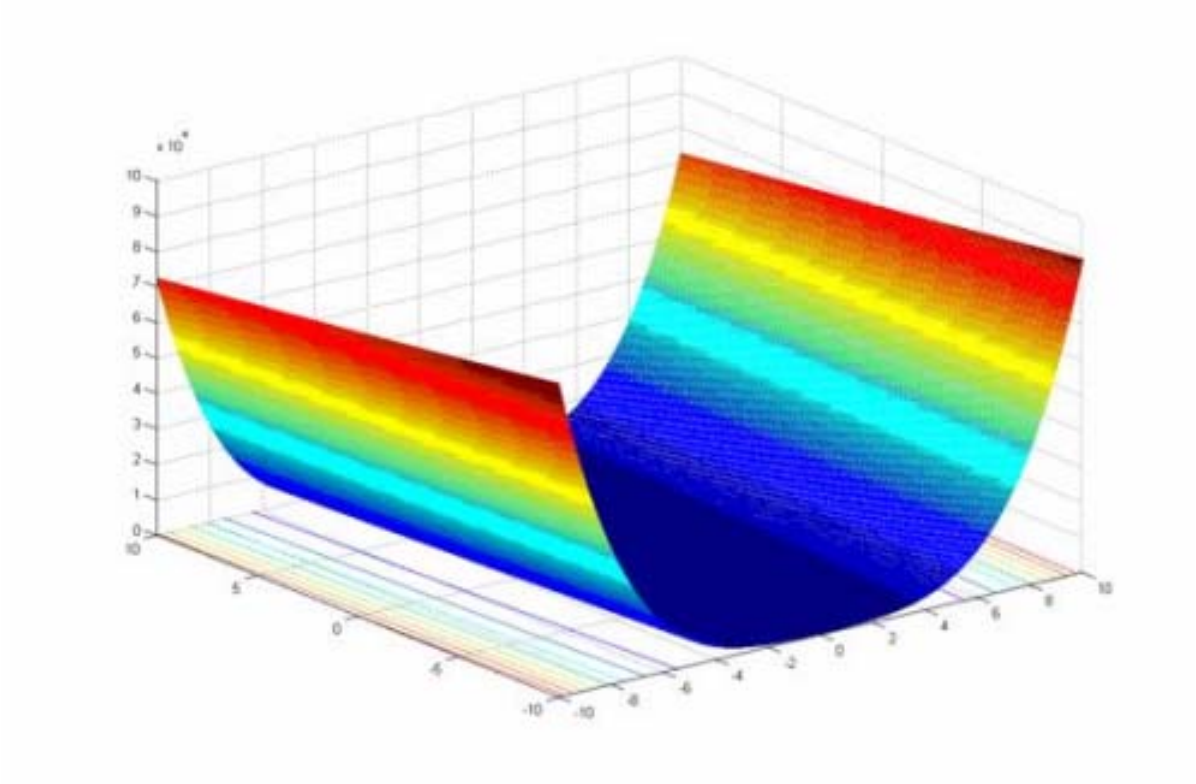


Figura 67. Dixon & Price.

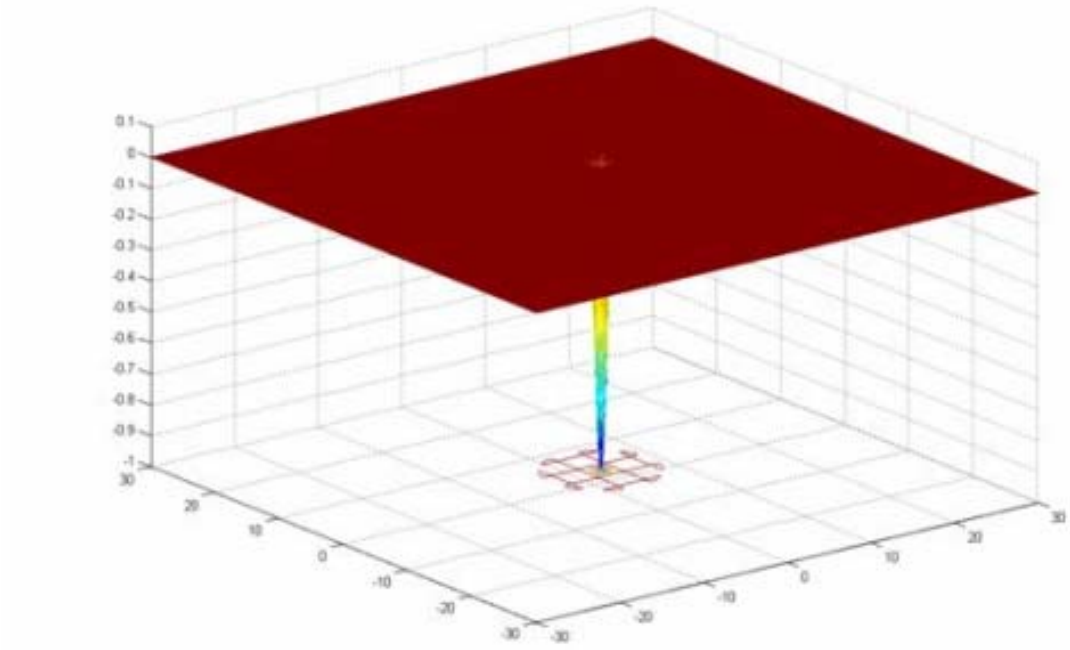


Figura 68. Função Easom.

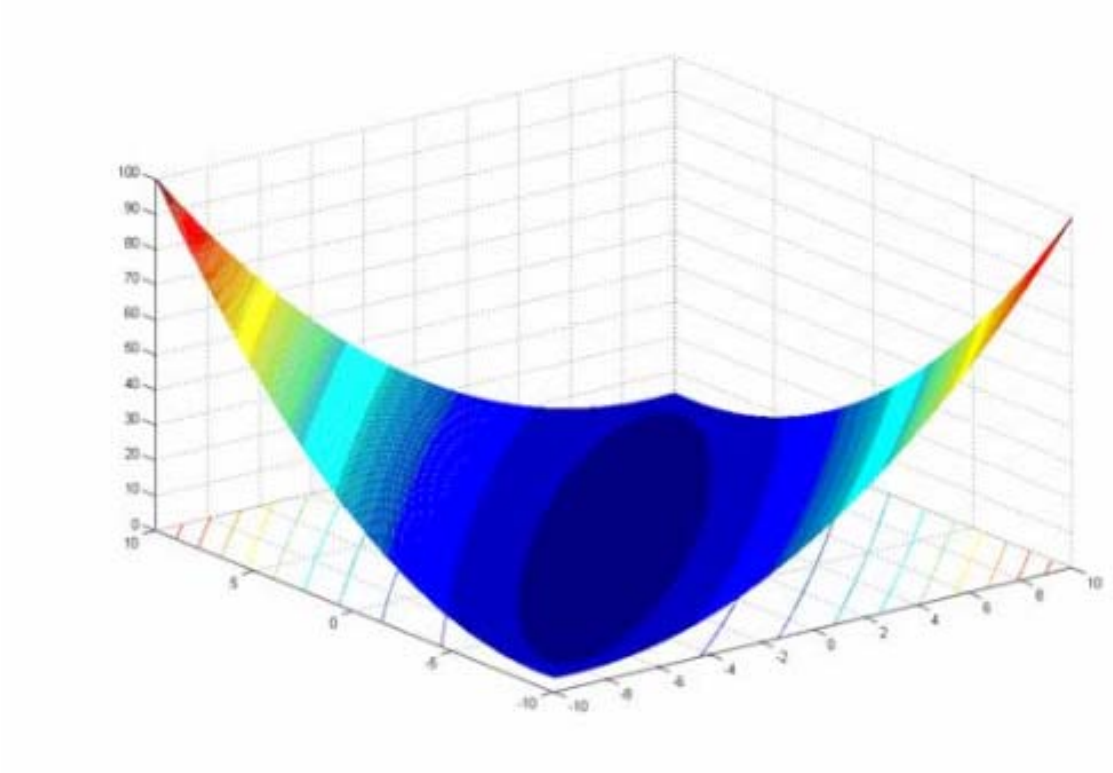


Figura 69. Função Matyas.

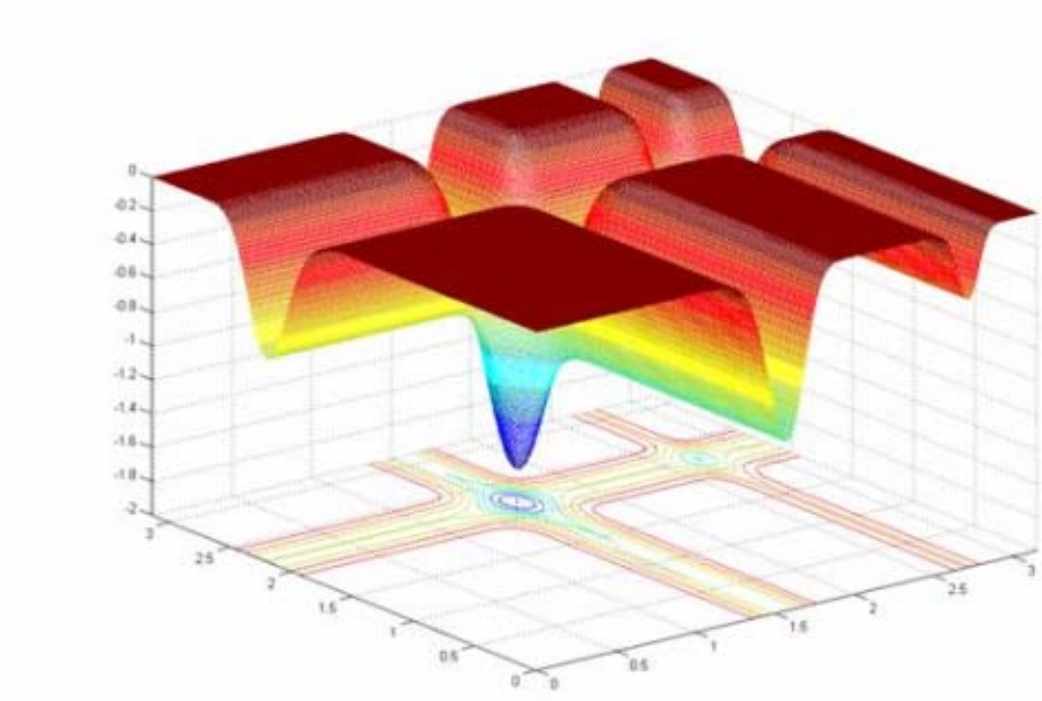


Figura 70. Função Michalewicz.

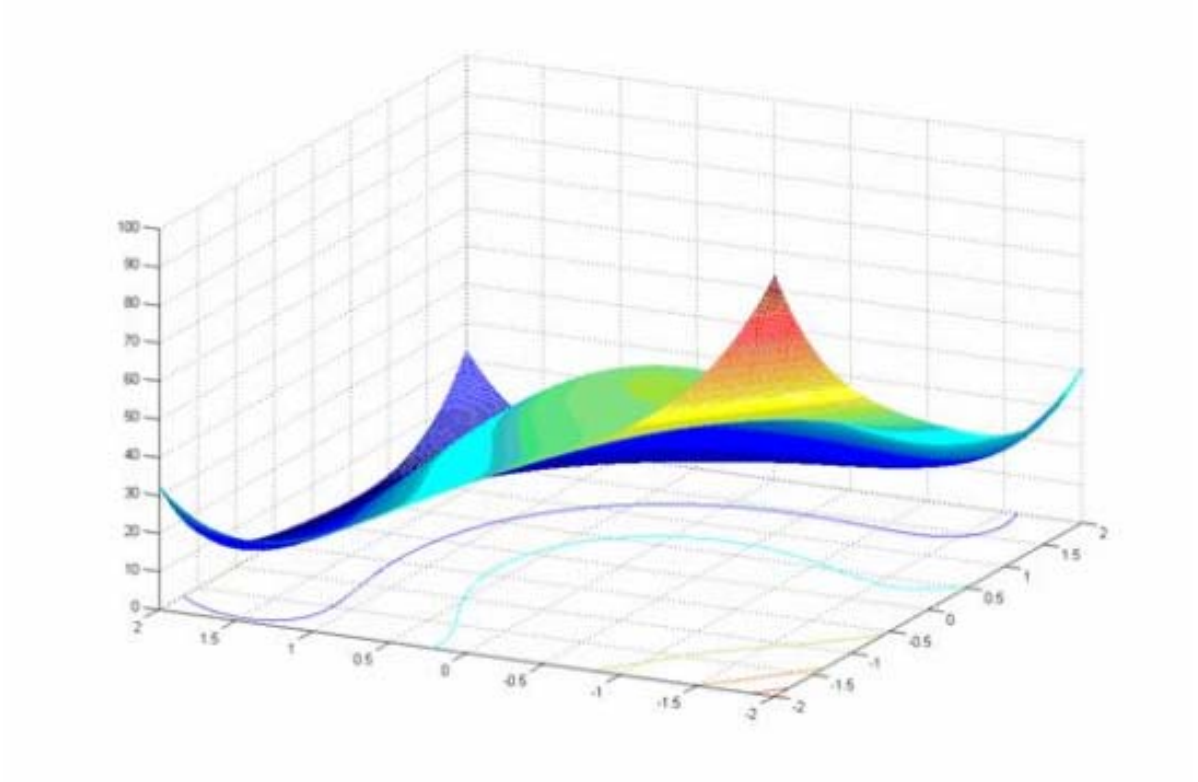


Figura 71. Função Perm.

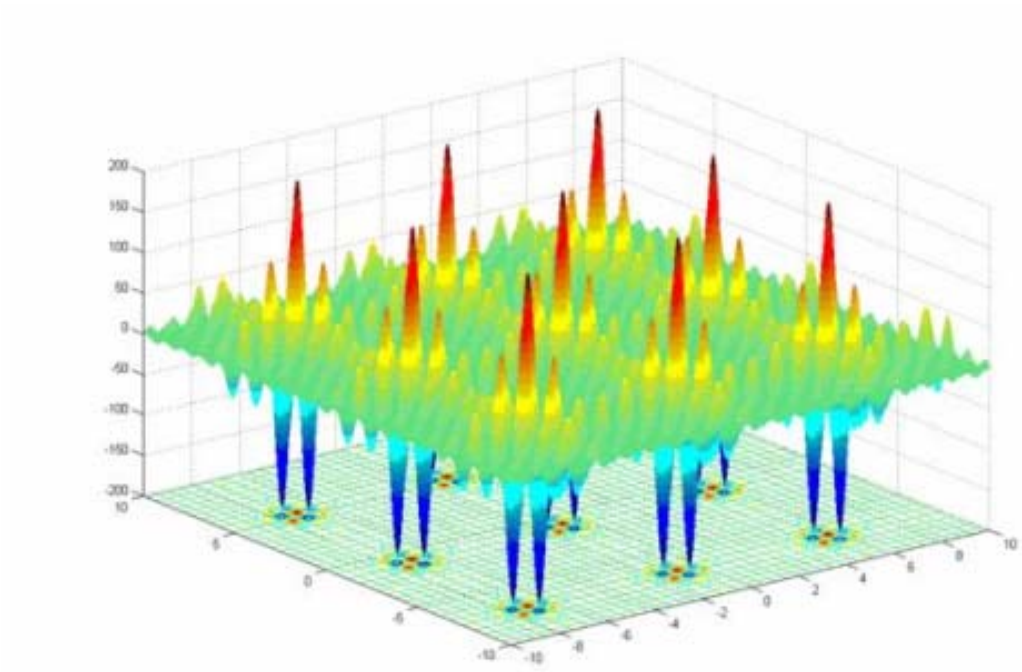


Figura 72. Função Shubert.

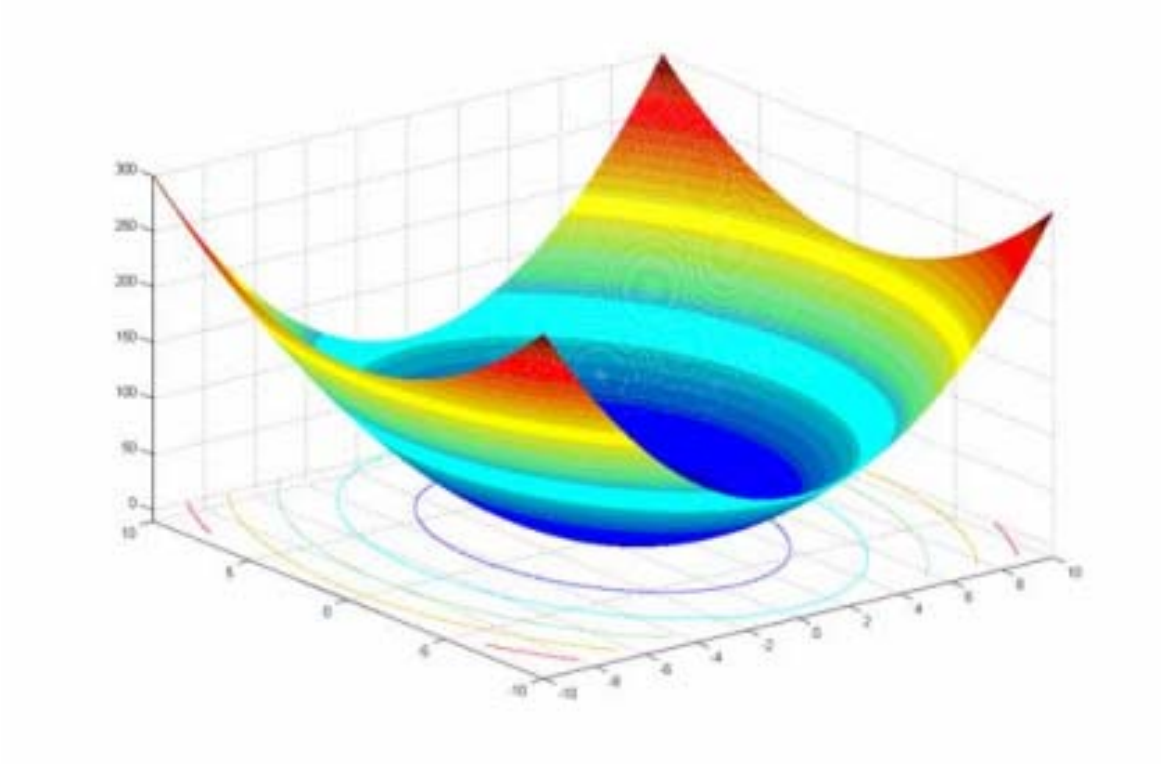


Figura 73. Função Soma de Quadrados.

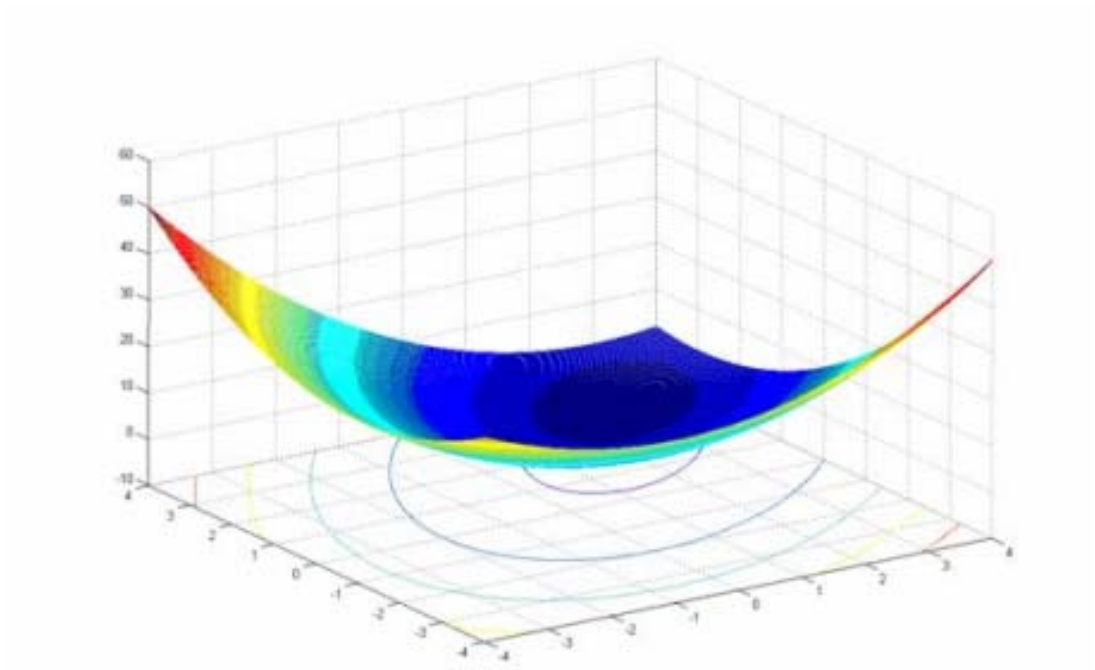


Figura 74. Função Trid.

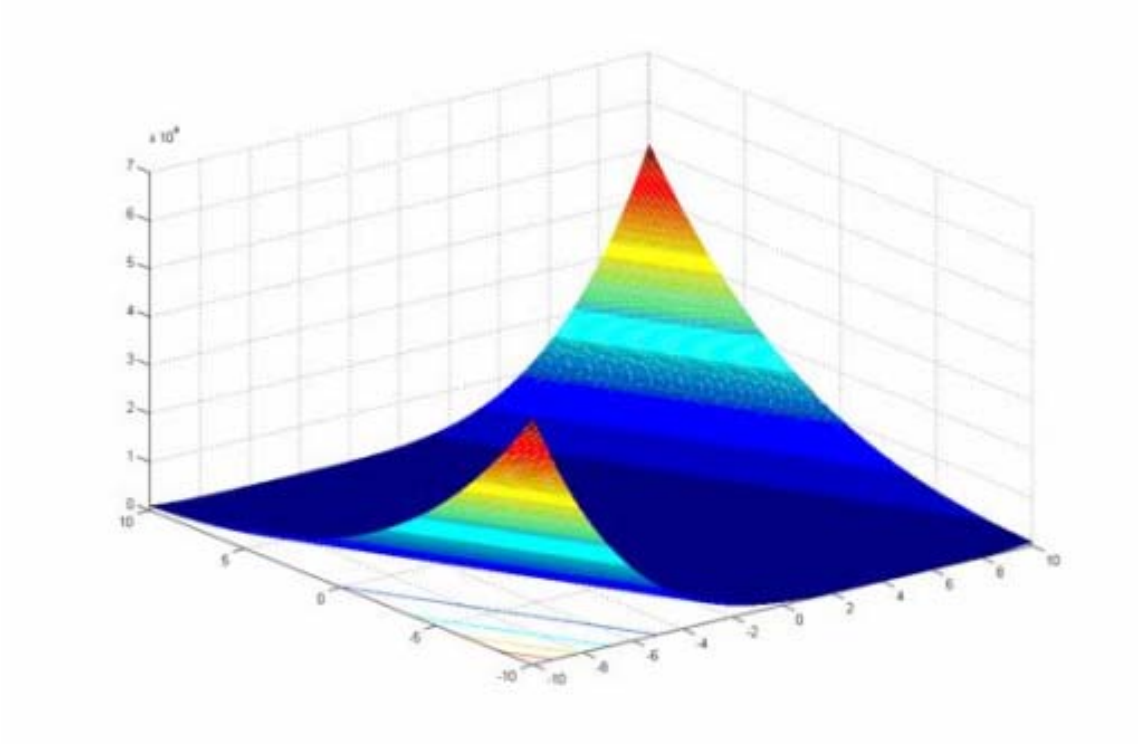


Figura 75. Função Zakharov.

Apêndice B

Casos de Uso do PSS

ESCOLA POLITÉCICA DE PERNAMBUCO

d.s.c. UPE Departamento de Sistemas e Computação

Documento de Casos de Uso

PSS – PSO Simulation Shell

Data	29/12/2008
Responsável	Emanoel Barreiros

Recife
Pernambuco

Histórico de Revisão

Data	Revisão	Autor	Papel	Descrição
29/12/2008	1.0	Emanoel Barreiros	Desenvolvedor	Criação do documento

Sumário

Introdução

Este documento tem como objetivo especificar o caso de uso Manter funcionalidades da solução Controle de Acesso, fornecendo informações para a equipe de desenvolvimento do projeto.

Casos de Uso

[UC001] – Realizar Simulação

Descrição: Este caso de uso tem a funcionalidade de realizar a simulação para uma determinada configuração do algoritmo do PSO.

Prioridade: *Essencial*

Atores: Usuário

Dependência: Não se aplica.

Fluxo Principal

Pré-condições: Não se aplica.

Passos:

[P1] O sistema apresenta a tela inicial.

[P2] O usuário seleciona o tipo de PSO a ser utilizado. (A01) (RN01)

[P3] O usuário preenche/confirma os valores nos campos *Time Steps* e *Simulation Repetitions*.

[P4] O usuário seleciona as variações que deseja que executem.

[P5] O usuário seleciona a topologia. (A03) (RN02)

[P6] O sistema habilita o campo *Number of Paticles*.

[P7] O usuário seleciona a função de teste. (RN03)

[P8] O usuário informa/sugere os valores para a configuração da função de teste.

[P9] O usuário pressiona o botão *Start*. (E01)

Pós-condições: n/a

Fluxo Alternativo

(A01) – Seleção do tipo de PSO

Pré-condições: No fluxo principal [P2], caso o usuário tenha selecionado o tipo *Inertia*, os campos *Inertia Factor*, *C up* e *C low* devem ser habilitados.

Passos:

[A1.1] O sistema habilita os campos *Inertia Factor*.

[A1.2] O sistema sugere valores padrão para os campos *Inertia Factor*.

[A1.3] O usuário informa se deseja que seja utilizado o mecanismo de dispersão (seleciona campo *Use Dispersion Grade*). (A02)

Pós-condições: n/a

(A02) – Seleção do campo *Use Dispersion Grade*

Pré-condições: No fluxo principal [P2], caso o usuário tenha selecionado o tipo *Inertia*, os campos *Inertia Factor*, *C up* e *C low* devem ser habilitados.

Passos:

[A2.1] O sistema habilita os campos *C up* e *C low*.

[A2.2] O usuário informa/confirma os valores sugeridos para os campos.

(A03) – Preenchimento de campos adicionais

Pré-condições: Nos fluxo principal (P5) o usuário selecionou a topologia.

Passos:

[A2.1] O usuário informa/confirma os valores para os campos que forem habilitados.

Pós-condições: n/a

[UC002] – Gerar gráfico de box plot

Descrição: Este caso de uso tem a funcionalidade de gerar o gráfico de *box plot* a partir da massa de dados gerada por uma simulação.

Prioridade: *Essencial*

Atores: Usuário

Dependência: Não se aplica.

Fluxo Principal

Pré-condições: Um diretório com vários arquivos de simulação, o arquivo informativo de configuração do algoritmo.

Passos:

[P1] O usuário seleciona a opção *Box Plot* no menu *Chart*.

[P2] O sistema apresenta a tela para geração do *box plot*.

[P3] O usuário informa/confirma o valor de *offset* sugerido pelo sistema.

[P4] O usuário informa se deseja que o gráfico gerado possua eixo Y logarítmico.

[P5] O usuário pressiona o botão *Generate*.

Pós-condições: O gráfico deve ser gerado.

[UC003] – Gerar gráfico de evolução

Descrição: Este caso de uso tem a funcionalidade de gerar o gráfico de *evolução* a partir do arquivo de evolução para um enxame.

Prioridade: *Essencial*

Atores: Usuário

Dependência: Não se aplica.

Fluxo Principal

Pré-condições: Um arquivo com os valores de *fitness* do enxame, o arquivo informativo de configuração do algoritmo.

Passos:

[P1] O usuário seleciona a opção *Evolution* no menu *Chart*.

[P2] O sistema apresenta a tela para geração do gráfico de evolução.

[P3] O usuário informa se deseja que o gráfico gerado possua eixo Y logarítmico.

[P4] O usuário pressiona o botão *Generate*.

Pós-condições: O gráfico deve ser gerado.

[UC004] – Realizar teste de Mann-Whitney

Descrição: Este caso de uso tem a funcionalidade de realizar o teste de Mann-Whitney a partir das massas de dados geradas por duas simulações.

Prioridade: *Essencial*

Atores: Usuário

Dependência: Não se aplica.

Fluxo Principal

Pré-condições: Dois diretórios com vários arquivos de simulação, o arquivo informativo de configuração do algoritmo para as duas massas de dados. A duas massas de dados devem ser geradas com a mesma quantidade de iterações.

Passos:

[P1] O usuário seleciona a opção *Mann-Whitney* no meu *Statistics*.

[P2] O sistema apresenta a tela para geração do gráfico de evolução.

[P3] O usuário informa o diretório com as amostras de controle.

[P4] O usuário informa o diretório com as amostras de teste.

[P5] O usuário seleciona uma hipótese a ser testada.

[P6] O usuário informa se deseja utilizar a aproximação normal.

[P6] O usuário informa/confirma o valor para o campo *Equality Tolerance*.

[P7] O usuário informa/confirma o valor para o campo *Confidence Level*.

[P8] O usuário pressiona o botão *Calculate*. (E02)

Pós-condições: O sistema preenche a tabela de *fitness* e exibe os indicadores do teste de Mann-Whitney.

[UC005] – Salvar arquivo de simulação

Descrição: Este caso de uso tem a funcionalidade de salvar um arquivo de simulação.

Prioridade: *Essencial*

Atores: Sistema

Dependência: Não se aplica.

Fluxo Principal

Pré-condições: A simulação deve ser inicializada.

Passos:

[P1] O usuário realiza o (UC001).

[P2] O sistema cria o diretório para comportar os relatórios de simulação.

[P2] O sistema grava no diretório criado o relatório para cada simulação realizada.

Pós-condições: Relatório de simulação gravado no diretório correto.

[UC006] – Salvar arquivo informativo de configuração

Descrição: Este caso de uso tem a funcionalidade de salvar um arquivo informativo da configuração da simulação.

Prioridade: *Essencial*

Atores: Sistema

Dependência: Não se aplica.

Fluxo Principal

Pré-condições: Todos as simulações devem ser finalizadas.

Passos:

[P1] O usuário realiza o (UC001).

[P2] O sistema cria o diretório para comportar os relatórios de simulação.

[P3] Após a realização das simulações, o sistema salva, no diretório das mesmas, o arquivo informativo de configuração utilizada pelo algoritmo.

Pós-condições: Arquivo informativo de configuração da simulação gravado no diretório correto.

[UC007] – Calcular média e desvio padrão

Descrição: Este caso de uso tem a funcionalidade de calcular media e desvio padrão para um conjunto de relatórios de simulação.

Prioridade: *Essencial*

Atores: Sistema

Dependência: Não se aplica.

Fluxo Principal

Pré-condições: Todos os relatórios de simulação salvos no diretório da simulação.

Passos:

[P1] O usuário realiza o (UC001).

[P2] O sistema cria o diretório para comportar os relatórios de simulação.

[P3] Após a realização das simulações, o sistema salva, no diretório das mesmas, o arquivo informativo de configuração utilizada pelo algoritmo.

[P4] O sistema calcula a média e desvio padrão para o desempenho do algoritmo nas simulações.

Pós-condições: Médias e desvios padrão calculados.

[UC008] – Salvar relatório de média e desvio padrão

Descrição: Este caso de uso tem a funcionalidade de salvar o relatório de média e desvio padrão.

Prioridade: *Essencial*

Atores: Sistema

Dependência: Não se aplica.

Fluxo Principal

Pré-condições: Médias e desvios padrão calculados.

Passos:

[P1] O sistema realiza o (UC007).

[P2] O sistema grava o relatório de média e desvio padrão no diretório da simulação.

Pós-condições: Relatório de médias e desvios padrão salvo.

Fluxo de Exceções

(E01) – Dados obrigatórios não informados

Pré-condições: No fluxo principal [P5] do (UC001) o usuário não informou os dados obrigatórios.

Passos:

[E1.1] Caso o usuário não tenha informado todos os dados o sistema exibe uma mensagem informando o campo obrigatório que deve ser preenchido.

Pós-condições: n/a

(E02) – Hipótese, amostras de controle e teste não informadas

Pré-condições: No fluxo principal [P8] do (UC004) o usuário não informou a hipótese a ser testada, as amostras de teste ou controle.

Passos:

[E2.1] Caso a população de controle não tenha sido informada o sistema informa que ela é obrigatória.

[E2.2] Caso a população de teste não tenha sido informada o sistema informa que ela é obrigatória.

[E2.3] Caso a hipótese não tenha sido informada o sistema informa que ela é obrigatória.

Pós-condições: n/a

Requisitos Especiais

Não se aplica.

Regras de Negócio

[RN01] – Habilitação dos campos Inertia Factor.

Caso o usuário tenha selecionado o tipo de PSO *Inertia*, o sistema deverá habilitar os campos *Inertia Factor*, que definem os valores inicial e final para o peso de inércia.

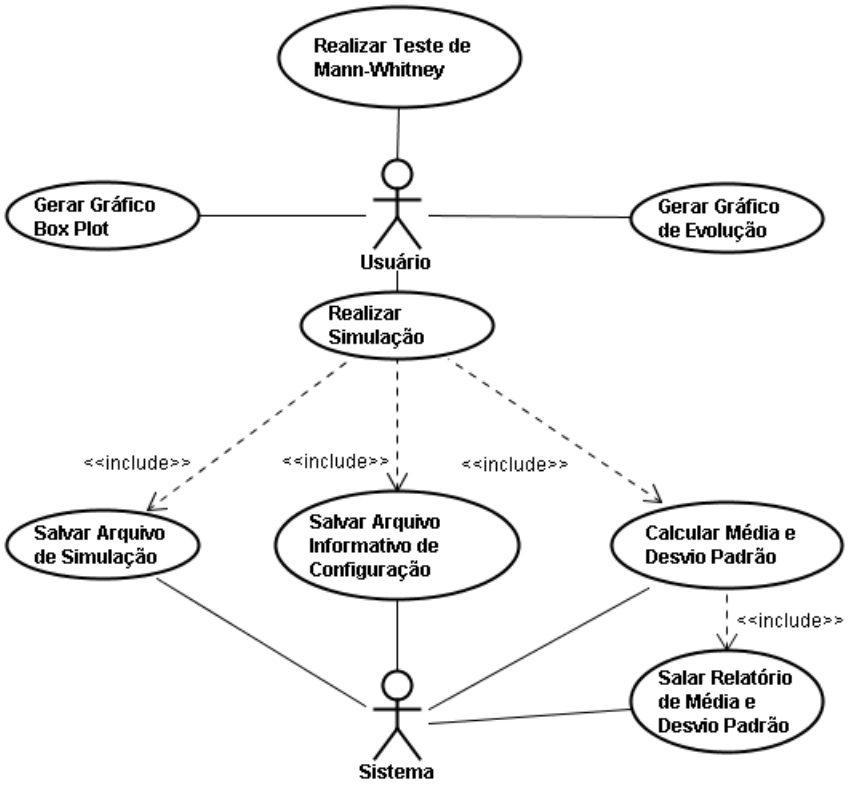
[RN02] – Habilitação dos campos Number of Clans, Number of Rings, Stagnation Limit, Conference Type, C low e C up.

Caso o usuário selecione a topologia *Clan* ou *Four Clusters*, os campos *Number of Clans* e *Conference Type* devem ser habilitados.

Caso o usuário selecione as topologias *Multi-Ring* ou *Multi-Ring* Dynamic o campo *Number of Rings* e *Stagnation Limit* devem ser habilitados.

Caso o usuário selecione o tipo de PSO

Diagrama Caso de Uso



Interfaces com Software

Não se aplica.

Interfaces com Hardware

Não se aplica.

Observações

Não se aplica.

Referências

Não se aplica.

Anexo I

Aprovação do Documento

Declaram estar de acordo e cientes de suas responsabilidades descritas neste documento, os seguintes aprovadores:

Nome	Assinatura	Data
Emanoel Barreiros		29/12/2008

Apêndice C

Documento de Escopo do PSS

Documento de Escopo

PSS – PSO Simulation Shell

Data	29/12/2008
Responsável	Emanoel Barreiros

Histórico de Revisão

Data	Versão	Autor	Papel	Descrição
29/12/2008	1.0	Emanoel Barreiros	Desenvolvedor	Criação do documento

Sumário

1 Introdução

Este documento define o escopo do projeto PSS – Particle Swarm Optimization Simulation Shell. O objetivo deste sistema é:

- Realizar simulações de algoritmos de otimização por enxame de partículas.
- Manter histórico das simulações realizadas.
- Manter histórico de configuração da execução das simulações.
- Gerar gráficos de evolução (*fitness*) individuais e do enxame a partir de histórico de simulações previamente realizadas.
- Realizar análise estatística sobre os resultados das simulações.
- Calcular média e desvio padrão das simulações.

Este escopo servirá como base para construção de uma ferramenta capaz de agilizar a pesquisa na área de algoritmos de otimização por enxame de partículas. Como a ferramenta está inserida em um ambiente inerentemente experimental, ela será capaz de auxiliar nos seguintes pontos:

- Reduzir o tempo para desenvolvimento de uma nova proposta de melhoria no algoritmo.
- Facilitar o armazenamento de histórico de simulações.
- Aumentar a capacidade de análise de resultados através de ferramentas estatísticas.
- Fornecer um ambiente que pode ser facilmente utilizado com objetivos educacionais.

1.1 Definições, Acrônimos e Abreviações

Abaixo são elencadas as definições de termos, acrônimos e abreviações necessárias para a interpretação deste documento.

Termo	Descrição
PSO	Particle Swarm Optimization (otimização por enxame de partículas)
Fitness	Desempenho de uma partícula durante a execução do algoritmo.
Partícula	Indivíduo que compõe o enxame.
Enxame	Agrupamento de partículas que atuam de forma ordenada em busca de uma solução para o problema proposto.
Simulação	Método que tenta reproduzir um problema real avaliando o desempenho do enxame.

1.2 Visão Geral

Este documento está estruturado da seguinte maneira:

Seção 2 – apresenta o cenário atual e motivação para o desenvolvimento deste trabalho.

Seção 3 – apresenta os problemas encontrados.

Seção 4 – apresenta as necessidades e restrições do sistema.

Seção 5 – apresenta o contexto do sistema com outros softwares e/ou hardwares.

2 Cenário Atual

A solução de problemas de otimização através de técnicas matemáticas clássicas geralmente é computacionalmente muito custosa. Os algoritmos de otimização por enxame de partículas, surgiram com o intuito de resolver de forma mais rápida alguns destes problemas. Quase em sua totalidade, os trabalhos desenvolvidos neste campo dependem de simulações e observações sobre uma determinada proposta. Desde uma nova topologia até a análise do efeito da qualidade dos números aleatórios exigem que sejam realizados experimentos e comparações entre a nova proposta e o estado da arte.

3 Descrição do Problema

Logo abaixo é descrita a situação atual do problema e o impacto que a nova solução irá prover.

O problema é:	Não existência de um padrão para a manutenção do histórico das simulações realizadas. Tempo gasto para desenvolvimento e teste de uma nova proposta de melhoria do algoritmo. Necessidade de ferramentas de análise estatística dos resultados das simulações.
Que afeta...	Pesquisadores e profissionais da área.
O impacto disto é...	Os pesquisadores não possuem um ambiente padrão para a realização de simulações. O tempo para que se desenvolva uma nova proposta é bastante alto. Não se tem um padrão para a manutenção do histórico das simulações. Não existe uma maneira fácil de executar testes significância estatística.

A solução seria...	Desenvolver uma ferramenta que permita realizar simulações com as mais variadas configurações do PSO: topologias, tipos de PSO, funções de teste e variações do PSO. O simulador também deve fornecer ferramentas de análise estatística para que possam ser realizados testes nas massas de dados geradas pelas simulações.

4 Visão Geral do Produto

4.1 Necessidades

Necessidade	Prioridade
Configurar simulação (topologia, tipo do PSO, variações e função de teste).	Alta
Permitir a ativação de uma ou mais variações do PSO para a simulação.	Alta
Construir arquitetura que comporte extensão através de *plugins*.	Baixa
Permitir que o sistema execute em ambiente distribuído (*cluster*).	Baixa
Gerar gráficos de evolução para uma simulação.	Alta
Gerar gráficos de evolução comparativos para uma ou mais simulações.	Baixa
Gerar gráficos de *box plot* para um conjunto de simulações.	Alta
Gerar gráficos de box plot comparativos para um ou mais conjuntos de simulação	Baixa
Ambos os tipos de gráficos devem possuir a opção para serem gerados com eixo Y logarítmico.	Alta
Permitir que seja definido um intervalo para leitura dos valores para montagem do *box plot*.	Alta
Permitir que seja utilizada a aproximação normal para cálculo do teste de Mann-Whitney.	Média

4.2 Restrições

Restrição	Prioridade
O sistema deve ter uma boa usabilidade e ser intuitivo.	Alta
O sistema deve ser *multiplataforma*.	Alta
O sistema deve prover uma arquitetura de fácil extensão.	Alta

4.3 Escopo Negativo

- A ferramenta não contemplará os seguintes pontos:
- Extensão por meio de *plugins*.
- Tratamento de problemas dinâmicos.
- Geração de gráficos comparativos entre duas partículas ou dois enxames.
- Simulação em ambientes distribuídos.

5 Papéis e Responsabilidades

Esta seção apresenta os stakeholders do sistema, ou seja, as pessoas ou organizações que serão afetadas pelo sistema e que direta ou indiretamente tem influência sobre o mesmo.

Nome	Descrição	Papel	Contato	Responsabilidades
Carmelo Bastos	Líder do projeto	Orientador	carmelofilho@dsc.upe.br	- Validar Requisitos - Validar o sistema - Fornecer Requisitos
Danilo Carvalho	Coordenador do projeto	Co-orientador	dfc@dsc.upe.br	- Validar sistema - Consultoria
Emanoel Barreiros	Responsável pelo desenvolvimento do projeto	Colaborador	efsb@dsc.upe.br	- Implementação - Coordenação técnica
Péricles Miranda	Colaborador	Colaborador	periclesmiranda@gmail.com	- Implementação
Marcel Caraciolo	Colaborador	Colaborador	mpc@dsc.upe.br	- Implementação

Referências

Não se aplica.

Anexo I

Aprovação do Documento

Declaram estar de acordo e cientes de suas responsabilidades descritas neste documento, os seguintes aprovadores:

Nome	Assinatura	Data
Emanoel Barreiros		29/12/2008

Apêndice D

Documento Requisitos do PSS

Documento de Requisitos

PSS – PSO Simulation Shell

Data	29/12/2008
Responsável	Emanoel Barreiros

Recife
Pernambuco

Histórico de Revisões

Data	Revisão	Autor	Papel	Descrição
29/12/2008	1.0	Emanoel Barreiros	Desenvolvedor	Criação do documento.

Sumário

1 Introdução

Este documento apresenta os requisitos do projeto COMPASS. O documento de requisitos é a especificação oficial dos requisitos do sistema para clientes, usuários finais e desenvolvedores de software.

2 Estrutura dos Requisitos

Os requisitos listados neste documento devem seguir a seguinte estrutura:

Requisito: descreve a identificação e nome do requisito. Requisitos funcionais e não-funcionais devem ser identificados, respectivamente, segundo o padrão [RF00X] ou [RNF00X], sendo que X representa um número que caracteriza sua identificação. A identificação do caso de uso não deverá mudar ao longo das revisões, pois é necessário garantir o controle de rastreabilidade e configuração.

Descrição: descreve o requisito.

Prioridade: é o grau de relevância do requisito em relação à implementação da aplicação. Requisitos mais relevantes (de maior prioridade) devem ser implementados primeiro. Esse atributo pode assumir os seguintes valores:

- *Essencial:* requisito básico, sem o qual o sistema não entra em funcionamento. Requisitos são requisitos imprescindíveis, têm que ser implementados impreterivelmente.
- *Importante:* requisito que não impede o funcionamento do sistema, mas a sua ausência pode causar insatisfação. Estes requisitos devem ser implementados, mas, se não forem, o sistema poderá ser implantado e usado mesmo assim.
- *Desejável:* requisito que não compromete o funcionamento do sistema, sua ausência não interfere na satisfação. Estes requisitos podem ser deixados para versões posteriores da solução, caso não haja tempo hábil para implementá-los na versão que está sendo especificada.

3 Requisitos Funcionais

3.1 Unidades

3.1.1 [RF001] Configurar simulação

Descrição: O sistema deve permitir a que o usuário configure todos os parâmetros para a execução da simulação de forma fácil e intuitiva.

Prioridade: Essencial

3.1.2 [RF002] Gerar gráfico de *box plot*

Descrição: O sistema deve gerar o gráfico de *box plot* para um conjunto de simulações. O sistema deve permitir que o usuário opte que o eixo dos Y's seja linear ou logarítmico.

Prioridade: Essencial

3.1.3 [RF003] Gerar gráfico de evolução

Descrição: O sistema deve gerar o gráfico de evolução para apenas uma simulação. O sistema deve permitir que o usuário opte que o eixo dos Y's seja linear ou logarítmico.

Prioridade: Essencial

3.1.4 [RF004] Realizar teste de significância estatística

Descrição: O sistema deve realizar pelo menos um teste de significância estatística para comparação de duas populações geradas por configurações diferentes do algoritmo. Sugere-se o teste de Mann-Whitney que é bastante indicado para populações que não seguem a normal.

Prioridade: Essencial

3.1.5 [RF005] Calcular média e desvio padrão

Descrição: O sistema deve ser capaz de calcular a média e desvio padrão para um conjunto de simulações.

Prioridade: Essencial

3.1.6 [RF006] Salvar relatório de simulação

Descrição: Ao fim de cada simulação, o sistema deve salvar um relatório da execução do enxame naquela simulação.

Prioridade: Essencial

3.1.7 [RF007] Criar diretório para simulações realizadas em batch

Descrição: O sistema deve criar um diretório para armazenar os arquivos de simulação realizados no mesmo comando de simulação em batch.

Prioridade: Essencial

3.1.8 [RF008] Salvar arquivo informativo de configuração do algoritmo

Descrição: O sistema deve salvar no diretório das simulações um arquivo que informe qual a configuração utilizada para gerar aqueles relatórios de simulação.

Prioridade: Essencial

3.1.9 [RF009] Salvar relatório de média e desvio padrão

Descrição: O sistema deve salvar em um arquivo independente com os valores de média e desvio padrão calculados para as simulações de um mesmo diretório.

Prioridade: Essencial

3.1.10 [RF010] Implementar variações do PSO

Descrição: O sistema deve implementar variações do PSO para que possa fornecer um ambiente mais flexível de execução.

Prioridade: Importante

3.1.11 [RF011] Sugerir valores padrão para funções de teste

Descrição: O sistema deve fornecer valores padrão para cada função de teste que conste no sistema.

Prioridade: Essencial

3.1.12 [RF012] Exibir movimento das partículas

Descrição: O sistema deve permitir que o usuário observe o movimento das partículas no enxame. Talves seja necessário adaptar o vetor de posições para que seja possível a visualização do usuário.

Prioridade: Essencial

3.1.13 [RF013] Exibir fitness da melhor partícula em tempo real

Descrição: O sistema deve exibir em tempo real o desempenho da melhor partícula do enxame.

Prioridade: Essencial

3.1.14 [RF014] Escolher offset para captura dos fitness no box plot

Descrição: O sistema deve proporcionar ao usuário a definição do valor de offset para leitura do fitness nos relatórios de simulação.

Prioridade: Essencial

3.1.15 [RF015] Permitir tolerância de igualdade no teste de significância estatística

Descrição: O sistema deve permitir que o usuário defina o grau de tolerância para identificação de igualdade entre duas amostras quando do cálculo do teste de significância estatística.

Prioridade: Essencial

Referências

Documento de escopo – PSS Escopo.

Glossário

Essa subseção descreve as definições de todos os termos, acrônimos e abreviações necessárias para a interpretação deste documento.

Termo	Descrição
PSO	Particle Swarm Optimization (otimização por enxame de partículas)
PSS	Particle Swarm Optimization Simulation Shell
Fitness	Desempenho da partícula durante a execução do algoritmo.