ESCOLA POLITÉCNICA
DE PERNAMBUCO

Influência da qualidade da geração dos números aleatórios em algoritmos de otimização por inteligência de enxames

Trabalho de Conclusão de Curso

Engenharia da Computação

Júlio Dantas de Andrade
Orientador: Prof. Carmelo José Albanez Bastos Filho

Júlio Dantas de Andrade

Influência da qualidade da geração dos números aleatórios em algoritmos de otimização por inteligência de enxames

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.

Agradecimentos

Eu agradeço a todos que me agüentaram nos anos de universidade. Professores e colegas de classe em geral. Um agradecimento especial aos professores Carmelo Bastos, Carlos Alexandre, Sérgio Campello e Alex Ramos por me aturar fora do seu tempo de ensino normal e ainda prover orientação em outros assuntos, acadêmicos ou não.

Uma menção honrosa também deve ser dita a todos que desenvolveram a ferramenta PSS antes/durante o período deste TCC, pois se eu tivesse que implementar tudo do início seria mais complicado. VALEU!

E para todos os conhecidos, desconhecidos e pessoas que não se enquadrem nessa categoria, apenas uma frase:

May the force be with us!

Resumo

Algoritmos de busca inteligente são altamente eficientes para resolver problemas onde não é possível aplicar métodos de busca exaustiva, nesta categoria se enquadra o algoritmo de Otimização por Enxame de Partículas (*Particle Swarm Optimization* - PSO). Apesar de ser capaz de fornecer boas soluções, o PSO ainda é um algoritmo computacionalmente custoso e dentre os fatores que influenciam esse custo está a utilização de um gerador de números pseudo-aleatórios. Entretanto, apesar de sua influência, não foi feito um estudo aprofundado de como esses geradores impactam no desempenho do PSO, seja este o desempenho da técnica ou seu tempo de execução. Este trabalho faz esse estudo utilizando diversas funções usadas como *benchmarks* para esta técnica conhecidas na literatura, medindo seu tempo de execução e os resultados obtidos.

Abstract

Intelligent search algorithms are highly efficient in solving problems which isn't possible to use exaustive solution search. The Particle Swarm Optimization algorithm (PSO) is included in this group. Even if the PSO is capable of providing good solutions, the PSO still is a computationally expensive algorithm. One of the factors that contribute for the high PSO's cost is the pseudo random generator factor, necessary for the correct operation of the PSO. However, despite its influence, a deep study about the impact of those generators hasn't been made yet. Be this study about its execution time or the quality of PSO's results. This work makes this study using various benchmark functions known in the literature, measuring and comparing obtained results and execution time.

Sumário

Índice de Figuras

Índice de Tabelas

Tabela de Símbolos e Siglas

AG – Algoritmo Genético

CI – Computação Inteligente

GCL – Gerador de Congruência Linear

GNA – Geradores de Números Aleatórios

OCR – *Optical Character Recognition* (Reconhecimento óptico de caracteres)

PSO – *Particle Swarm Optimization* (Otimização por enxame de partículas)

PSS – PSO *Simulation Shell*

RNA – Redes Neurais Artificiais

UPE – Univesidade de Pernambuco

Capítulo 1

Introdução

Devido à rápida evolução da eletrônica, atualmente é possível encontrar grande capacidade computacional, seja ela capacidade de processamento, memória e outros, em dispositivos cada vez menores. Esse aumento de poder computacional permite a utilização de técnicas antes inviáveis devido a um tempo de resposta longo, mesmo que a resposta da técnica fosse melhor para o problema. Neste cenário de aumento de capacidade computacional, surge a possibilidade de usar técnicas de Computação Inteligente (CI).

As técnicas de CI podem ser adaptadas de acordo com o escopo do problema, ou seja, são capazes de coletar informações do ambiente e tomar decisões para aumentar a chance de sucesso. Graças a essa habilidade, técnicas de CI são aptas a se generalizar para diversos problemas, sendo altamente eficientes para resolver situações dinâmicas e problemas não determinísticos, como por exemplo, reconhecimento de caracteres (OCR) manuscritos em textos digitalizados [1][2], roteamento em redes de comunicações [3], etc.

A grande maioria das técnicas de CI são derivadas da observação em outras áreas do conhecimento, com o objetivo de reproduzí-las em um computador. Um exemplo disso são as redes neurais artificiais (RNA), que se baseiam no modelo dos neurônios [8]. Neste trabalho, deseja-se especificamente abordar a técnica de *particle swarm optimization* (PSO, otimização por enxame de partículas), que se baseia no comportamento de enxames de animais, como pássaros, que utilizam um grupo de indivíduos para realizar buscas de forma eficiente.

Como dito anteriormente, o PSO utiliza comportamentos observados em enxames para realizar uma busca em um espaço limitado, tentando sempre achar a melhor posição. Por exemplo, no caso de uma função matemática, essa busca consistiria em achar soluções que maximizem ou minimizem o resultado dessa função, gerando assim o melhor resultado sem a necessidade de testar todas as possíveis soluções do problema.

Um dos fatores importantes para obter resultados satisfatórios usando o PSO consiste na utilização de números aleatórios. Entretanto, os computadores atuais não são capazes de gerar números totalmente aleatórios. Por isso, quando se necessita de números que possuem essas características, faz-se necessária a utilização de geradores de números pseudo-aleatórios, ou seja, números que aparentam possuir comportamento aleatório, mas podem ser previstos segundo fórmula matemática de seu gerador. Como estes geradores devem possuir boas propriedades estatísticas, pode-se considerar que estes são, para todos os efeitos, geradores de números aleatórios [5].

A utilização de geradores de números aleatórios pode influenciar diretamente na resposta do PSO, tornando este parâmetro vital para o sucesso ou fracasso do método. Este trabalho visa analisar o impacto da qualidade da geração dos números aleatórios no algoritmo PSO e, de modo geral, determinar o método computacionalmente mais adequado para gerar números aleatórios sem comprometer o desempenho do algoritmo de busca.

Este trabalho está dividido da seguinte forma: no Capítulo 2 são mostrados os conceitos básicos e particularidades do PSO; no Capítulo 3 é feita uma explanação geral sobre números aleatórios e seus geradores; o Capítulo 4 contém a definição do arranjo experimental utilizado para os experimentos; o Capítulo 5 possui os resultados obtidos com este estudo e, finalmente, no Capítulo 6 são apresentadas as conclusões e os trabalhos futuros.

Capítulo 2

Otimização por Enxame de Partículas

Este algoritmo teve como inspiração observação de bandos de pássaros [11] e as interações entre eles, quando o bando de pássaros busca algo, seja comida, o ninho ou outros. A esse algoritmo deu-se o nome de algoritmo de otimização por enxame de partículas (PSO – *Particle Swarm Optimization*).

O algoritmo de PSO é um algoritmo estocástico, com comportamento evolutivo baseado em convivência social, onde indivíduos simples se unem em torno de um objetivo e cooperam para solucionar um problema complexo. Este algoritmo é bastante utilizado para otimização de funções continuas que podem ser definidas num espaço *n*-dimensional.

A inteligência dessa técnica não se baseia em um individuo do bando ou colônia, mas nas interações sociais geradas por indivíduos bem sucedidos.

Diferentemente de outros algoritmos de busca da computação inteligente, o PSO troca informações sobre o histórico de buscas dos indivíduos de uma população, ao invés de combinar informações dos indivíduos, como no caso dos algoritmos genéticos (AG).

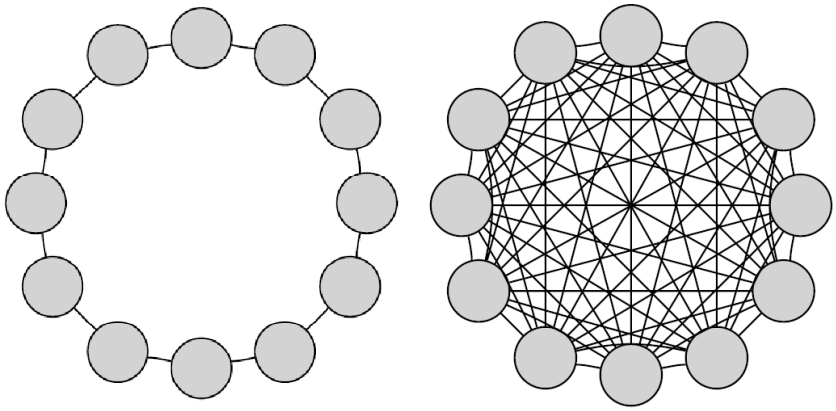
A utilização de múltiplos indivíduos no processo de busca torna menos provável que o PSO fique preso em mínimos locais, uma vez que, quando um indivíduo encontra um mínimo local, sua comunicação com os outros indivíduos faz com que este possa "perceber" que não possui a melhor solução e continue procurando, saindo assim desse mínimo.

2.1 Vizinhança

A técnica de PSO descreve diferentes relações entre as partículas, essas relações ditam como os participantes do bando se comunicam entre si. Se for gerado um grafo, assumindo os nós como as partículas e as conexões entre os nós representando se uma partícula se comunica ou não com outra partícula distinta, esses grafos caracterizam topologias, que definem a vizinhança.

Existem diversas definições de vizinhança, as quais se aplicam a vários problemas, porém as vizinhanças mais utilizadas são conhecidas como vizinhança local e vizinhança global. Na vizinhança local cada partícula se comunica com seus vizinhos mais próximos, ou seja, cada elemento se comunica com apenas duas outras partículas, formando um anel, como mostrado na

figura 1-a. Vale salientar que neste caso, a proximidade não é definida como distância euclidiana no espaço de busca, e sim como proximidade dos índices das partículas. Por exemplo, a partícula 2 é vizinha das partículas 1 e 3. Na vizinhança global todos os indivíduos se comunicam entre si, como pode ser visto na figura 1-b.



(a) (b)

Figura 1. Formatos de vizinhança do PSO. (a) Vizinhança local e (b) Vizinhança global.

2.2 O Algoritmo Básico

No algoritmo de PSO um individuo participante do algoritmo de PSO é conhecido como partícula. Seja D a dimensão do espaço de busca. Cada partícula é composta de três vetores:

- $\vec{x}_i = (x_{i1}, x_{i2}, \dots, x_{iD})$: Vetor que mostra qual a posição atual da partícula nas D;
- $\vec{p}_i = (p_{i1}, p_{i2}, \dots, p_{iD})$: Armazena a melhor posição que foi individualmente encontrada durante o histórico de busca da partícula;
- $\vec{v}_i = (v_{i1}, v_{i2}, \dots, v_{iD})$: Velocidade atual da partícula em D.

A partícula se move através do espaço de busca seguindo as equações (1) e (2) para alteração de velocidade e posição, respectivamente.

$$\vec{v}_i(t+1) = \vec{v}_i(t) + c_1 r_1(\vec{p}_i(t) - \vec{x}_i(t)) + c_2 r_2\big(\vec{p}_g(t) - \vec{x}_i(t)\big), \quad (1)$$

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t), \quad (2)$$

onde c_1e c_2 são constantes, geralmente possuindo valor 2,05 [11][4], que representam os fatores cognitivos e sociais respectivamente, r_1 e r_2 são números aleatórios gerados por uma distribuição uniforme no intervalo [0,1] e o vetor $\vec{p}_g$ é a melhor posição encontrada dentro da vizinhança da partícula.

As equações (1) e (2) são repetidas para todas as partículas em cada instante de tempo discreto t até que algum critério de parada seja atingido. Abaixo está apresentado o pseudocódigo do algoritmo.

Inicialize posições $\vec{x}_i$ e velocidades $\vec{v}_i$ aleatoriamente
Encontre $\vec{p}_i$ e $\vec{p}_g$ iniciais e armazene
Enquanto (critério de parada não atingido)
    Faça para todas as partículas
        Atualizar valores de velocidade e posição das partículas
        Se (novo valor de *fitness* $[f(\vec{x}_i)]$) for melhor que $[f(\vec{p}_i)]$
            Atualizar $\vec{p}_i$
        Se (novo valor de *fitness* $[f(\vec{x}_i)]$) for melhor que $[f(\vec{p}_g)]$
            Atualizar $\vec{p}_g$

Figura 2. Pseudocódigo do algoritmo de PSO.

2.2.1 PSO com Inércia

O algoritmo original pode ser alterado de modo a considerar apenas uma parte da velocidade anterior da partícula. Essa alteração se dá multiplicando a velocidade no instante anterior por um parâmetro ω. Dessa forma, pode-se alterar a equação (1) para:

$$\vec{v}_i(t+1) = \omega(t)\vec{v}_i(t) + c_1 r_1(\vec{p}_i(t) - \vec{x}_i(t)) + c_2 r_2\big(\vec{p}_g(t) - \vec{x}_i(t)\big), \quad (3)$$

onde, de acordo com [7], o valor de ω deve ser alterado dinamicamente, tendo seu valor inicial próximo à 1,0 e depois diminuindo em função do tempo para valores bem menores que 1,0, fazendo assim que inicialmente seja incentivada a exploração em amplitude no espaço de busca e, após certo tempo, a busca se torne restrita à melhor área encontrada.

Contudo não existem valores precisos indicados, a estratégia mais utilizada é iniciar o fator ω com valor de 0,9 e diminuir linearmente para 0,4 na última iteração [6].

2.2.2 PSO com *Constriction Factor*

Outro método que altera o algoritmo original de modo a balancear melhor o algoritmo é o *Constriction Factor*, onde é introduzido um novo parâmetro χ, que é dado por:

$$\chi = \frac{2}{\left|2 - c - \sqrt{c^2 - 4c}\right|}, c = c_1 + c_2 \text{ e } c > 4, \tag{4}$$

gerando a seguinte equação para atualização da velocidade:

$$\vec{v}_i(t+1) = \chi\left(\vec{v}_i(t) + c_1 r_1(\vec{p}_i(t) - \vec{x}_i(t)) + c_2 r_2\left(\vec{p}_g(t) - \vec{x}_i(t)\right)\right). \tag{5}$$

Assim como no caso do fator de inércia, o objetivo dessa técnica é restringir a busca a uma pequena área do espaço total de busca. É dito também em [6] que o *constriction factor* é na verdade um caso especial do algoritmo com inércia, onde os parâmetros foram determinados analiticamente [4].

Capítulo 3

Números Aleatórios

Números aleatórios podem ser definidos a partir de uma sequência de números onde o próximo número da sequência não pode ser previsto, mesmo sabendo quais os números anteriores [5]. Enquanto esse tipo de número pode ser conseguido através de fenômenos físicos, atualmente não é possível obter números completamente aleatórios de programas de computadores.

Surge então um problema, pois aleatoriedade é essencial para alguns problemas de computação, dentre eles os que envolvem sistemas estocásticos (como o PSO), processos altamente dependentes de imprevisibilidade, como criptografia e outros.

Visando resolver esse problema, foram criados geradores de números pseudo-aleatórios. Geradores de números pseudo-aleatórios se baseiam em fórmulas matemáticas e em um valor inicial, conhecido como semente, para gerar números que são estatisticamente aleatórios, ou seja, apesar destes geradores não fornecerem números verdadeiramente aleatórios, eles podem ser usados como tal e são referenciados como geradores de números aleatórios (GNA). Observe que esses geradores atendem a critérios estatísticos sobre uma distribuição e que estes geradores, em geral, utilizam variáveis com distribuição uniforme.

Para uma variável discreta, uma distribuição uniforme é uma distribuição na qual todos os pontos podem surgir em um espaço específico (finito) com mesma probabilidade. Para variáveis continuas, a distribuição uniforme se caracteriza por todos os intervalos de mesmo tamanho serem equiprováveis. Faz-se necessária esta distinção, pois quando se trabalha com aritmética inteira, utiliza-se o caso discreto, onde os valores gerados pelos GNAs são discretos.

Apesar de os geradores de distribuição uniforme serem os mais desejados, pode-se necessitar que a distribuição dos números siga outro tipo de distribuição, como por exemplo, uma distribuição exponencial. Todavia, em sua grande maioria, números que seguem uma outra distribuição são gerados através de operações realizadas sobre uma ou mais distribuições uniformes [12].

Geralmente, distribuições uniformes geram valores entre 0 e 1, e a alteração desses valores para um intervalo qualquer pode ser feita através do método de transformação.

Vale salientar que estes geradores são limitados na quantidade de números que podem produzir. Essa quantidade é chamada de período do gerador. Quando essa quantidade máxima é atingida, o gerador volta a fornecer os mesmo números iniciais em ordem, por exemplo, um gerador de período 5 que provê os números 1, 2, 1, 5 e 8, após o fornecimento do número 8, este gerador volta a fornecer os números 1, 2, 1 e assim sucessivamente.

3.1 Geradores de Números Aleatórios

Existe hoje uma grande variedade de geradores de números aleatórios. Abaixo, serão destacados os geradores que serão utilizados nesse trabalho com suas características.

3.1.1 Gerador de Congruência Linear

O gerador de congruência linear (GCL) foi por muito tempo o gerador mais utilizado e pesquisado, e até hoje pode ser encontrado na grande maioria das linguagens de programação como sua função padrão para geração de números aleatórios, como por exemplo, a função *rand()* em C.

O GCL é definido pela equação (6), onde é fornecida uma semente X_0, a é um inteiro positivo chamado multiplicador, c é um inteiro não-negativo chamado de incremento, m é um inteiro conhecido como módulo [12] e X_n é o próximo valor da sequência gerada. O GCL é capaz de gerar números randômicos inteiros entre [0, $m-1$].

$$X_n = (a \cdot X_{n-1} + c) \bmod m. \tag{6}$$

O período deste gerador possui valor máximo m, onde este período máximo só será atingido se os valores de a e c forem corretamente escolhidos. Especialmente o valor de a, uma vez que o valor c muitas vezes assume o valor zero (também conhecido como GCL multiplicativo). Note que o valor a é, em geral, escolhido através de busca exaustiva, ou seja, escolhe-se um valor para a constante e realizam-se testes de estatísticos de aleatoriedade sobre as sequências X_n obtidas, se os resultados dos testes forem satisfatórios, o valor a é adotado, se não, escolhe-se outro valor para a e executam-se os testes para o novo valor de a e assim sucessivamente.

Entretanto o GLC é considerado um gerador fraco porque, se tuplas consecutivas de valores gerados pelo GLC forem escolhidos e utilizando-se dessas tuplas para plotar um gráfico, o GLC possui, no máximo, $m^{\frac{1}{n}}$ hiperplanos. Dependendo dos valores escolhidos (m, a e c), pode-

se gerar números praticamente não aleatórios. Além disso, caso m seja uma potência de 2 ($2^{30}, 2^{31}$ e etc) então os bits menos importantes não possuem aleatoriedade ou estes geram sequências de período muito baixo [11][12].

Observe que, enquanto bons geradores não necessariamente geram sequências com grandes períodos, estas devem possuir alto grau de aleatoriedade. No entanto, para o caso de m igual a uma potência de 2, o GLC falha neste quesito.

3.1.2 Gerador de Marsaglia

O gerador de números aleatórios de 64-bits de Marsaglia [10] é um gerador capaz de gerar números aleatórios entre [0,1), do tipo ponto flutuante de precisão dupla (*double* em C) e com período aproximado de 2^{202} (ou aproximadamente 10^{61}). O algoritmo é baseado em duas operações binárias, definidas nas equações abaixo:

$$x \bullet y = \{\text{if } x \geq y \text{ then } x - y, \text{else } x - y + 1\} \tag{7}$$

e

$$c \diamond d = \{\text{if } c \geq d \text{ then } c - d, \text{else } c - d + r\}, \tag{8}$$

onde na equação (8), o valor de d é um número racional, sendo $d > 0$ expressado por $k/2^{53}$ onde qualquer valor de k pode ser utilizado, desde que $k < p$, sendo p um valor primo igual a $2^{53} - 111$, e r é também um número racional, dado por $p/2^{53}$.

Diferentemente do GLC, este método tem uma quantidade de sementes variável, onde a quantidade de sementes escolhidas pelo usuário define quantas saídas possíveis o algortimo pode gerar. Para uma quantidade de saídas máxima, este método requer 97 sementes $x_1, x_2, \ldots, x_{97}$ iniciais e um valor c_1 do tipo ponto flutuante de dupla precisão (64 bits) que serão utilizados para gerar as sequências mostradas nas equações (9) e (10) respectivamente. Essas sequências serão, por sua vez, utilizadas para gerar os números aleatórios resultantes U_n, como mostrado na equação (11).

$$x_n = x_{n-97} \bullet x_{n-33}, \tag{9}$$

$$c_n = x_{n-1} \diamond r, \tag{10}$$

$$U_n = x_n \bullet c_n. \tag{11}$$

O pseudo-código do algoritmo está apresentado na Figura 3.

Inicializa vetor de sementes x[97]
Definir as constantes r e d
Inicializa valor de c_1
Inicialize índices de controle do vetor com valores 97 e 33
Função getRandom()
    Calcule $x_n = x_{n-97} \bullet x_{n-33}$
    Salve nova semente x_n no vetor
    Atualize os índices do vetor
    Calcule valor de c_n
    Calcule U_n
    Retorne U_n
Fim da função getRandom()

Figura 3. Pseudo-código do GNA de Marsaglia.

Capítulo 4

Arranjo Experimental

Neste capítulo serão mostradas quais foram as funções de testes utilizadas para os experimentos, os valores assumidos para as equações e a ferramenta utilizada para realizar os experimentos.

4.1 Funções de Teste

Para medir o impacto dos geradores de números aleatórios no desempenho do PSO será utilizada uma série de funções conhecidas na literatura para o PSO. Estas funções são problemas de minimização. Para todas as funções listadas abaixo, considere n o número de dimensões da solução e $\vec{x}_i$ a posição da partícula na dimensão i. A quantidade de dimensões utilizadas para todas as funções é de 30 dimensões.

Na Tabela 1 é possível observar as funções utilizadas, qual o domínio de busca possível para cada função e o valor no qual a função atinge seu ponto de mínimo. As fórmulas das funções são apresentas na Tabela 2.

Tabela 1. Funções de busca, domínio de busca possível e valor ótimo de cada função.

Função de Teste	Domínio de busca	Valor Ótimo	Região de Inicialização
Esfera (12)	$-100 \leq x_i \leq 100$	$0{,}0^D$	$50 \leq x_i \leq 100$
Rosenbrock (13)	$-30 \leq x_i \leq 30$	$1{,}0^D$	$15 \leq x_i \leq 30$
Griewank (14)	$-600 \leq x_i \leq 600$	$0{,}0^D$	$300 \leq x_i \leq 600$
Rastrigin (15)	$-5{,}12 \leq x_i \leq 5{,}12$	$0{,}0^D$	$2{,}56 \leq x_i \leq 5{,}12$
Schwefel 1.2 (16)	$-100 \leq x_i \leq 100$	$0{,}0^D$	$50 \leq x_i \leq 100$
Schwefel 2.6 (17)	$-500 \leq x_i \leq 500$	$0{,}0^D$	$-500 \leq x_i \leq -250$
Penalized Function P8 (18)	$-50 \leq x_i \leq 50$	$-1{,}0^D$	$25 \leq x_i \leq 50$
Penalized Function P16 (19)	$-50 \leq x_i \leq 50$	$1{,}0^D$	$25 \leq x_i \leq 50$

Tabela 2. Formulas das funções utilizadas para os *benchmarks* dos testes.

$$f(x) = \sum_{i=1}^{n} x_i^2 \tag{12}$$

$$f(x) = \sum_{i=1}^{n-1} [100 \cdot (x_{i+1} - x_i^2)^2 + (x_i - 1)^2] \tag{13}$$

$$f(x) = \frac{1}{4000} \cdot \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \tag{14}$$

$$f(x) = 10 \cdot n + \sum_{i=1}^{n} x_i^2 - 10 \cdot \cos(2\pi x_i) \tag{15}$$

$$f(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2 \tag{16}$$

$$f(x) = -\sum_{i=1}^{n} x_i \cdot \sin\sqrt{x_i} \tag{17}$$

$$f(x) = \frac{\pi}{n} \Bigg\{ 10 \cdot \sin^2(\pi \cdot y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 \cdot \{1 + 10 \cdot \sin^2(\pi \cdot y_i)\} + (y_n - 1)^2 \Bigg\} + \sum_{i=1}^{n} \mu(x_i, 10{,}100{,}4) \tag{18}$$

$$y_i = 1 + \frac{1}{4} \cdot (x_i + 1)$$

$$\mu(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \le x_i \le a \\ k(-x_i - a)^m, & x_i < a \end{cases}$$

$$f(x) = 0.1 \cdot \left\{ \sin^2(3\pi x_1) + \sum_{i=1}^{n-1} (x_i - 1)^2 \cdot \{1 + \sin(3\pi x_{i+1})\} \right.$$
$$\left. + (x_n - 1)^2 \cdot \{1 + \sin^2(2\pi x_n)\} \right\} \qquad (19)$$
$$+ \sum_{i=1}^{2} \mu(x_i, 5{,}100{,}4)$$

4.2 Ferramenta de Simulação

Será utilizada a ferramenta de simulação *PSO Simulation Shell - PSS* desenvolvida no Departamento de Sistemas e Computação da Universidade de Pernambuco (UPE) pelo aluno de trabalho de conclusão de curso Emanoel Barreiros, pelo aluno já formado Marcel Caraciolo, pelo aluno de iniciação cientifica Péricles Miranda e o mestre pela Universidade de Pernambuco, Danilo Carvalho, todos orientados do professor Carmelo J. A. Bastos Filho. Uma imagem de sua janela principal pode ser vista na Figura 4. Esta ferramenta foi iniciada por D. F. Carvalho, mestre formado no programa de pós-graduação em engenharia da computação da Escola Politécnica de Pernambuco/UPE.

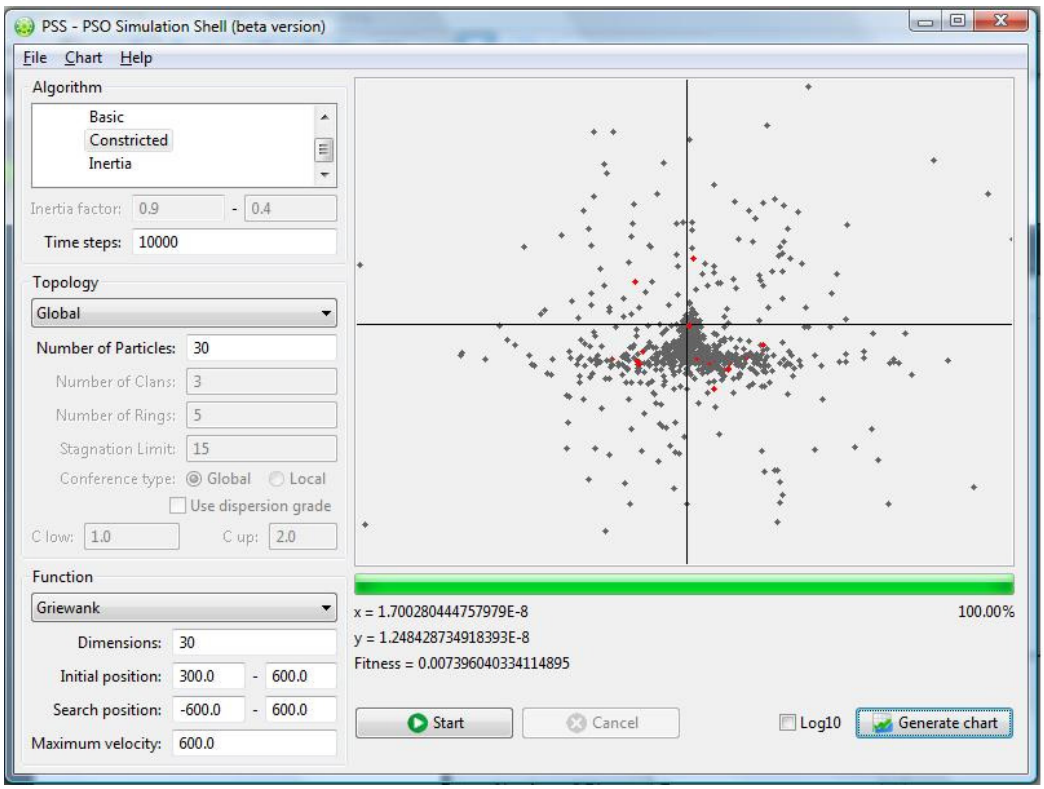


Figura 4. Janela principal do programa de simulação PSS.

Essa ferramenta está implementada em Java e utiliza também orientação a aspectos. O PSS é capaz de simular diversas condições de PSO, incluindo várias topologias, uma grande gama de funções de teste e pode gerar gráficos e relatórios para exibir o resultado das simulações.

Este trabalho adicionará a ferramenta meios de escolher o GNA a ser utilizado na simulação e modos de escolher os parâmetros destes geradores, como as sementes, por exemplo. Na Figura 5 pode-se observar a janela criada para configurar os GNAs.

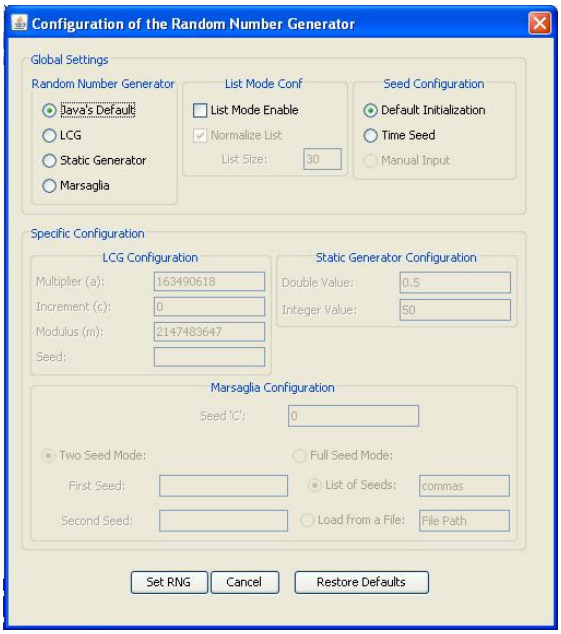


Figura 5. Janela de configuração de GNAs do programa PSS desenvolvida neste trabalho.

4.3 Parâmetros das Simulações

Os algoritmos de PSO usados nas simulações são as variações *constriction factor* e inércia. Nas simulações utilizando *constriction factor* foram utilizados os valores de constantes c_1 e c_2 com valor de 2,05, seguindo a condição apresentada pela equação (4). Para utilização do PSO com inércia, como explicado na sessão 2.2.1, o valor de $\omega(t)$ é inicializado com valor de 0,9 decaindo linearmente até o valor 0,4.

Apesar de existirem vários tipos de vizinhanças, serão utilizadas apenas as vizinhanças local e global, uma vez que são as mais utilizadas pela literatura atual, comprovando assim o efeito dos GNA na maior parte dos casos apresentados na literatura.

Para realizar o estudo do impacto dos GNA serão utilizadas os algoritmos mostrados na sessão 3.1, bem como a utilização de uma lista fixa de números gerados pelo algoritmo, fornecendo ao teste sequências que podem ser de grandes períodos ou de baixos períodos. Os

GNAs utilizados para gerar os valores aleatórios utilizados na equação de atualização de velocidade são:

- GNA1 – Gerador de Marsaglia utilizando duas sementes de 10 dígitos e o procedimento explicado em [10]. As sementes foram geradas pelo gerador padrão da linguagem, sendo novos valores fornecidos a cada simulação;
- GNA2 – Gerador padrão da linguagem JAVA, utilizando o método padrão de inicialização da semente;
- GNA3 – Gerador LCG utilizando os valores de $m = 2^{31} - 1$, $a = 163490618$ e $c = 0$ comprovadamente bons, obtidos de [9]. A semente foi gerada pelo gerador padrão do JAVA, sendo uma nova semente gerada por simulação. Números do tipo *double* são gerados através da divisão do número gerado pelo módulo;
- GNA4 – Gerador LCG com valor de $m = 2^{32}$, $a = 398347535$ e $c = 0$, sendo este considerado um gerador inferior que o GNA2, já que utiliza um módulo de potência de 2. A semente foi inicializada da mesma forma do GNA2. Números do tipo *double* são gerados através da divisão do número gerado pelo módulo;
- GNA5 – Gerador LCG utilizando $m = 2^{32}$, $a = 1$ e $c = 1$. Este gerador fornece números em sequência, ou seja, $1,2,3 \ldots 2^{32}$, logo é uma sequência que não possui nenhuma aleatoriedade. A semente foi inicializada da mesma forma do GNA2; Números do tipo *double* são gerados pela divisão do número gerado pelo módulo;
- GNA6 – Lista com 30 números gerados aleatoriamente por um gerador de marsaglia, de modo que após o fornecimento do $n - ésimo$ número o gerador volta a fornecer o primeiro número da lista. A lista foi normalizada para que a média fosse igual a 0,5;
- GNA7 – Lista com 300 números gerados aleatoriamente por um gerador de marsaglia, de modo que após o fornecimento do $n - ésimo$ número o gerador volta a fornecer o primeiro número lista. A lista foi normalizada para que a média fosse igual a 0,5;
- GNA8 – Lista com 3000 números gerados aleatoriamente por um gerador de marsaglia, de modo que após o fornecimento do $n - ésimo$ número o gerador volta a fornecer o primeiro número lista. A lista foi normalizada que a valor de média fosse igual a 0,5;
- GNA9 – Um gerador de apenas um número, estático, com valor do tipo *double* igual à 0,5 e valor inteiro igual a 50.

Todo o enxame possui posição e velocidade iniciadas aleatoriamente pelo algoritmo padrão do JAVA (GNA2) dentro do espaço de busca, utilizando 10.000 interações e para todas as funções de teste foram utilizadas 30 partículas.

Cada condição de simulação é executada 30 vezes para calculo da média e desvio padrão.

Capítulo 5

Resultados

Os resultados estão separados por seção, onde cada seção representa uma função de teste. O conjunto de funções de testes utilizado é bem conhecido e utilizado pela comunidade científica internacional [4]. Para cada seção existe duas subseções, onde estas representam o tipo de topologia usada e em cada subseção possui duas tabelas, para as abordagens *Constriction Factor* e inércia, respectivamente, onde em cada tabela está o GNA utilizado, o tempo médio em minutos de cada simulação, a média do resultado da função de *fitness* e seu desvio padrão.

5.1 Função Esfera

5.1.1 Topologia Global

A Tabela 3 e a Tabela 4 apresentam os resultados da função esfera para as abordagens *Constriction Factor* e Inércia, respectivamente, como topologia global.

Os resultados com o algoritmo de *Constriction Factor* (Tabela 3), para os GNA1, GNA2, GNA3 e GNA4 são equivalentes. Inclusive o desvio padrão é muito baixo. O mesmo ocorre para a abordagem da Inércia (Tabela 4), apesar da utilização do *Constriction Factor* oferecer melhores resultados (aproximadamente, ordem de 10^{-156}, frente a 10^{-43} de que quando se utiliza inércia). Os resultados para os geradores GNA5, GNA6, GNA7, GNA8 e GNA9 são sempre ruins e não se pode inferir muito sobre estes, pois possuem um desvio padrão muito maior do que a média. Note que o GNA5, que corresponde ao GCL de pior qualidade apresenta resultados muito ruins.

Tabela 3. Resultado da função Esfera com topologia global e algoritmo de *Constriction Factor*.

Gerador	Tempo (min)	Resultado
GNA1	0.05623	$2.82291 \cdot 10^{-154} \pm 2.33489 \cdot 10^{-306}$
GNA2	0.06735	$8.23945 \cdot 10^{-157} \pm 2.0075 \cdot 10^{-317}$
GNA3	0.0769067	$2.30039 \cdot 10^{-156} \pm 2.0075 \cdot 10^{-317}$
GNA4	0.0725067	$2.65581 \cdot 10^{-156} \pm 2.0075 \cdot 10^{-317}$
GNA5	0.0831533	$6028.87 \pm 6.74587 \cdot 10^{6}$
GNA6	0.0722	$11452.6 \pm 1.44103 \cdot 10^{7}$
GNA7	0.0761033	$4927.52 \pm 4.07047 \cdot 10^{6}$
GNA8	0.0949733	$4445.02 \pm 2.27433 \cdot 10^{6}$

GNA9	0.0591967	$7253.56 \pm 7.63741 \cdot 10^{6}$

Tabela 4. Resultado da função Esfera com topologia global algoritmo de Inércia.

Gerador	Tempo (min)	Resultado
GNA1	0.0543833	$1.41877 \cdot 10^{-43} \pm 7.97494 \cdot 10^{-86}$
GNA2	0.0657833	$3.34972 \cdot 10^{-43} \pm 2.62299 \cdot 10^{-84}$
GNA3	0.07485	$6.32722 \cdot 10^{-44} \pm 1.69077 \cdot 10^{-86}$
GNA4	0.07065	$2.78145 \cdot 10^{-43} \pm 4.66676 \cdot 10^{-85}$
GNA5	0.0870633	$2407.19 \pm 2.73511 \cdot 10^{6}$
GNA6	0.05806	$4739.62 \pm 4.24644 \cdot 10^{6}$
GNA7	0.0539533	1112.74 ± 706296
GNA8	0.0965367	$2585.54 \pm 1.97843 \cdot 10^{6}$
GNA9	0.0616	294.191 ± 25903.3

5.1.2 Topologia Local

A Tabela 5 e a Tabela 6 apresentam os resultados da função esfera para as abordagens *Constriction Factor* e Inércia, respectivamente, como topologia local.

Assim como na topologia global, os resultados com o algoritmo de *Constriction Factor* (Tabela 5), para os GNA1, GNA2, GNA3 e GNA4 são equivalentes. Inclusive o desvio padrão é muito baixo. O mesmo ocorre para a abordagem da Inércia (Tabela 6), apesar da utilização do *Constriction Factor* oferecer melhores resultados (aproximadamente, ordem de 10^{-70}, frente a 10^{-14} do que quando se utiliza inércia). Os resultados para os geradores GNA5, GNA6, GNA7, GNA8 e GNA9 são sempre ruins e não se pode inferir muito sobre estes, pois possuem um desvio padrão muito maior do que a média. Note que o GNA5, que corresponde ao GCL de pior qualidade apresenta resultados muito ruins.

Tabela 5. Resultado da função Esfera com topologia local e algoritmo de Constriction Factor.

Gerador	Tempo (min)	Resultado
GNA1	0.06754	$6.62008 \cdot 10^{-70} \pm 2.87547 \cdot 10^{-138}$
GNA2	0.07974	$1.72054 \cdot 10^{-69} \pm 5.65973 \cdot 10^{-137}$
GNA3	0.0876933	$3.60831 \cdot 10^{-70} \pm 4.90336 \cdot 10^{-139}$
GNA4	0.0839033	$1.00079 \cdot 10^{-69} \pm 4.6936 \cdot 10^{-138}$
GNA5	0.0820367	$3028.91 \pm 2.14552 \cdot 10^{6}$
GNA6	0.06458	$1121.15 \pm 1.2379 \cdot 10^{6}$
GNA7	0.06159	56.4391 ± 11107.7
GNA8	0.103317	$9738.97 \pm 6.97691 \cdot 10^{6}$
GNA9	0.0568	1743.26 ± 260799

Tabela 6. Resultado da função Esfera com topologia local e algoritmo de Inércia.

Gerador	Tempo (min)	Resultado
GNA1	0.0648567	$9.80604 \cdot 10^{-14} \pm 1.41362 \cdot 10^{-26}$
GNA2	0.0772567	$9.46187 \cdot 10^{-14} \pm 1.80298 \cdot 10^{-26}$
GNA3	0.08633	$1.57199 \cdot 10^{-13} \pm 1.49771 \cdot 10^{-25}$
GNA4	0.0824267	$8.88248 \cdot 10^{-14} \pm 1.07506 \cdot 10^{-26}$
GNA5	0.0809367	998.807 ± 359174
GNA6	0.05865	$1671.21 \pm 4.4077 \cdot 10^{6}$
GNA7	0.06234	$2253.69 \pm 3.43332 \cdot 10^{6}$
GNA8	0.102563	$5980.97 \pm 2.76275 \cdot 10^{6}$
GNA9	0.0562733	129.428 ± 2163.81

5.2 Função Rosenbrock

5.2.1 Topologia Global

A Tabela 7 e a Tabela 8 apresentam os resultados da função rosenbrock para as abordagens *Constriction Factor* e Inércia, respectivamente, como topologia global.

Os resultados com o algoritmo de *Constriction Factor* (Tabela 7), para os GNA1, GNA3 e GNA4 são equivalentes. Contudo o GNA2, que representa o gerador padrão da linguagem, se mostra um pouco inferior aos GNA1, GNA2 e GNA4. Para os outros GNAs os resultados são sempre ruins e não se pode inferir muito sobre estes, pois possuem um desvio padrão muito maior do que a média.

Os mesmos comentários sobre os GNA5, GNA6, GNA7, GNA8 e GNA9 podem ser aplicados a abordagem da Inércia (Tabela 8), entretanto, o GNA1 se destaca dos GNA2, GNA3 e GNA4, sendo esse pouco melhor.

Tabela 7. Resultado da função Rosenbrock com topologia global algoritmo de *Constriction Factor*.

Gerador	Tempo (min)	Resultado
GNA1	0.0728633	2.73693 ± 9.90472
GNA2	0.08405	7.16534 ± 311.65
GNA3	0.0922767	2.49694 ± 12.3369
GNA4	0.0880033	2.31481 ± 8.33844
GNA5	0.12194	$2.11221 \cdot 10^{6} \pm 2.88307 \cdot 10^{12}$
GNA6	0.0969067	$6.86632 \cdot 10^{6} \pm 1.83346 \cdot 10^{13}$
GNA7	0.0955567	$2.1995 \cdot 10^{6} \pm 5.20532 \cdot 10^{12}$
GNA8	0.11463	$2.59092 \cdot 10^{6} \pm 3.95846 \cdot 10^{12}$
GNA9	0.0979533	$4.51594 \cdot 10^{6} \pm 8.9012 \cdot 10^{12}$

Tabela 8. Resultado da função Rosenbrock com topologia global e algoritmo de Inércia.

Gerador	Tempo (min)	Resultado
GNA1	0.0733833	26.3679 ± 1450.39
GNA2	0.0842667	40.1777 ± 1094.02
GNA3	0.0928733	33.4791 ± 1309.25
GNA4	0.0888533	32.3836 ± 1076.99
GNA5	0.113563	$650842 \pm 3.50306 \cdot 10^{11}$
GNA6	0.0775867	$2.47726 \cdot 10^{6} \pm 5.22827 \cdot 10^{12}$
GNA7	0.07261	$371549 \pm 3.57058 \cdot 10^{11}$
GNA8	0.114327	$1.11254 \cdot 10^{6} \pm 4.15781 \cdot 10^{11}$
GNA9	0.0888867	$7372.67 \pm 4.18146 \cdot 10^{7}$

5.2.2 Topologia Local

A Tabela 9 e a Tabela 10 apresentam os resultados da função rosenbrock para as abordagens *Constriction Factor* e Inércia, respectivamente, como topologia local.

Os resultados com o algoritmo de *Constriction Factor* (Tabela 9), para os GNA1, GNA2 e GNA4 são equivalentes. Contudo o GNA3, que representa o gerador padrão da linguagem, se mostra um pouco inferior aos GNA1, GNA2 e GNA4. Para os outros GNAs os resultados são sempre ruins e não se pode inferir muito sobre estes, pois possuem um desvio padrão muito maior do que a média.

Os mesmos comentários sobre os GNA5, GNA6, GNA7, GNA8 e GNA9 podem ser aplicados a abordagem da Inércia (Tabela 10), entretanto, o GNA1 se destaca dos GNA2, GNA3 e GNA4, sendo esse um pouco melhor.

Tabela 9. Resultado da função Rosenbrock com topologia local e algoritmo de *Constriction Factor*.

Gerador	Tempo (min)	Resultado
GNA1	0.0836433	9.9752 ± 38.675
GNA2	0.09569	10.9494 ± 200.182
GNA3	0.10475	17.4609 ± 467.276
GNA4	0.100737	9.93441 ± 48.8543
GNA5	0.10201	$745523 \pm 4.01904 \cdot 10^{11}$
GNA6	0.0841	$375765 \pm 2.51576 \cdot 10^{11}$
GNA7	0.0846333	$49429.9 \pm 1.12166 \cdot 10^{10}$
GNA8	0.12235	$1.09004 \cdot 10^{7} \pm 2.21777 \cdot 10^{13}$
GNA9	0.0767633	$262534 \pm 1.7809 \cdot 10^{10}$

Tabela 10. Resultado da função Rosenbrock com topologia local e algoritmo de Inércia.

Gerador	Tempo (min)	Resultado
GNA1	0.08384	18.6618 ± 453.061
GNA2	0.0965167	21.684 ± 609.347

GNA3	0.105747	24.9816 ± 554.026
GNA4	0.100467	22.5036 ± 607.772
GNA5	0.102303	$138614 \pm 2.55744 \cdot 10^{10}$
GNA6	0.0783533	$2.04845 \cdot 10^{6} \pm 1.61922 \cdot 10^{13}$
GNA7	0.0812233	$712444 \pm 1.12472 \cdot 10^{12}$
GNA8	0.122463	$4.33997 \cdot 10^{6} \pm 3.16439 \cdot 10^{12}$
GNA9	0.0764633	$2233.04 \pm 5.82207 \cdot 10^{6}$

5.3 Função Rastrigin

5.3.1 Topologia Global

A Tabela 11 e a Tabela 12 apresentam os resultados da função rastrigin para as abordagens *Constriction Factor* e Inércia, respectivamente, como topologia global.

Para os GNA1, GNA2, GNA3 e GNA4, os resultados obtidos com *Constriction Factor* (Tabela 11) são inferiores aos obtido com a abordagem da Inércia (Tabela 12), pois esses são mais próximos de 0. Os resultados para os geradores GNA5, GNA6, GNA7, GNA8 e GNA9 são sempre piores e não se pode inferir muito sobre estes, pois possuem um desvio padrão muito maior do que a média. Novamente nota-se que o GNA5 possui um resultado muito pior que os outros GLCs (GNA3 e GNA4).

Tabela 11. Resultado da função Rastrigin com topologia global e algoritmo de *Constriction Factor*.

Gerador	Tempo (min)	Resultado
GNA1	0.0747533	50.179 ± 196.464
GNA2	0.0846	50.6102 ± 191.415
GNA3	0.0950067	53.3297 ± 199.257
GNA4	0.0911033	55.386 ± 289.904
GNA5	0.11417	181.677 ± 637.103
GNA6	0.07705	175.112 ± 625.786
GNA7	0.08162	128.896 ± 1120.94
GNA8	0.11955	147.7 ± 870.598
GNA9	0.0975567	170.741 ± 743.202

Tabela 12. Resultado da função Rastrigin com topologia global e algoritmo de Inércia.

Gerador	Tempo (min)	Resultado
GNA1	0.0741367	14.3606 ± 29.6766
GNA2	0.0839367	18.0751 ± 38.7841
GNA3	0.0947	15.256 ± 34.0221
GNA4	0.0907933	14.7514 ± 33.3121
GNA5	0.112657	120.635 ± 671.09
GNA6	0.07133	158.728 ± 2118.92

GNA7	0.06965	99.2754 ± 806.623
GNA8	0.11828	115.555 ± 798.975
GNA9	0.0829067	118.981 ± 851.858

5.3.2 Topologia Local

A Tabela 13 e a Tabela 14 apresentam os resultados da função rastrigin para as abordagens *Constriction Factor* e Inércia, respectivamente, como topologia local.

Para os GNA1, GNA2, GNA3 e GNA4, os resultados obtidos com *Constriction Factor* (Tabela 13) são inferiores aos obtido com a abordagem da Inércia (Tabela 14), pois esses são mais próximos de 0. Os resultados para os geradores GNA5, GNA6, GNA7, GNA8 e GNA9 são sempre piores e não se pode inferir muito sobre estes, pois possuem um desvio padrão muito maior do que a média. Novamente nota-se que o GNA5 possui um resultado muito pior que os outros GLCs (GNA3 e GNA4).

Tabela 13. Resultado da função Rastrigin com topologia local e algoritmo de *Constriction Factor*.

Gerador	Tempo (min)	Resultado
GNA1	0.0864233	41.7555 ± 99.6395
GNA2	0.0960967	41.1581 ± 55.9474
GNA3	0.108327	41.6887 ± 68.6366
GNA4	0.104157	39.4998 ± 64.3903
GNA5	0.10785	204.886 ± 417.308
GNA6	0.0852867	163.501 ± 1186.93
GNA7	0.0901967	166.384 ± 681.923
GNA8	0.12506	191.535 ± 466.483
GNA9	0.0816233	195.657 ± 291.585

Tabela 14. Resultado da função Rastrigin com topologia local e algoritmo de Inércia.

Gerador	Tempo (min)	Resultado
GNA1	0.0864167	22.3016 ± 36.4058
GNA2	0.0965067	20.1328 ± 26.0749
GNA3	0.107563	21.2642 ± 19.9101
GNA4	0.103533	21.0917 ± 20.3406
GNA5	0.107647	176.934 ± 1444.76
GNA6	0.08097	140.42 ± 1647.95
GNA7	0.08039	136.351 ± 791.629
GNA8	0.12449	146.326 ± 409.807
GNA9	0.07659	122.602 ± 648.392

5.4 Função Griewank

5.4.1 Topologia Global

A Tabela 15 e a Tabela 16 apresentam os resultados da função griewank para as abordagens *Constriction Factor* e Inércia, respectivamente, como topologia global.

Para os GNA1, GNA2, GNA3 e GNA4, os resultados com *Constriction Factor* (Tabela 15) e com o algoritmo Inércia (Tabela 16) são equivalentes. Os resultados para os geradores GNA5, GNA6, GNA7 e GNA8 são sempre piores e não se pode inferir muito sobre estes, pois possuem um desvio padrão muito maior do que a média. O GNA9 apesar de ser ruim utilizando *Constriction Factor*, quando se utiliza Inércia conseguiu um resultado melhor que os GNA5, GNA6, GNA7 e GNA8, apesar deste não ser bom se comparado com os GNA1, GNA2, GNA3 e GNA4.

Tabela 15. Resultado da função Griewank com topologia global e algoritmo de *Constriction Factor*.

Gerador	Tempo (min)	Resultado
GNA1	0.06807	0.0283581 ± 0.00126846
GNA2	0.0800933	$0.0335562 \pm 0.000968054$
GNA3	0.0889667	0.0331571 ± 0.0012762
GNA4	0.0848	0.0434009 ± 0.00119284
GNA5	0.110527	53.0398 ± 540.951
GNA6	0.0977367	94.983 ± 932.213
GNA7	0.100753	37.9808 ± 303.923
GNA8	0.13225	40.5723 ± 354.248
GNA9	0.0859033	73.5234 ± 703.807

Tabela 16. Resultado da função Griewank com topologia global e algoritmo de Inércia.

Gerador	Tempo (min)	Resultado
GNA1	0.0752133	0.013029 ± 0.000260615
GNA2	0.0868033	$0.0156394 \pm 0.000404592$
GNA3	0.0961667	0.0238986 ± 0.00057745
GNA4	0.0914367	0.0211427 ± 0.00024541
GNA5	0.114103	22.8122 ± 183.355
GNA6	0.0816933	54.5268 ± 720.827
GNA7	0.07624	12.6316 ± 43.3868
GNA8	0.130557	24.9317 ± 132.715
GNA9	0.07846	3.4068 ± 2.38403

5.4.2 Topologia Local

A Tabela 17 e a Tabela 18 apresentam os resultados da função griewank para as abordagens *Constriction Factor* e Inércia, respectivamente, como topologia local.

Assim como a topologia global, para os GNA1, GNA2, GNA3 e GNA4, os resultados com *Constriction Factor* (Tabela 17) e com o algoritmo Inércia (Tabela 18) são equivalentes. Os resultados para os geradores GNA5, GNA6, GNA7 e GNA8 são sempre piores e não se pode inferir muito sobre estes, pois possuem um desvio padrão muito maior do que a média. O GNA9 apesar de ser ruim utilizando *Constriction Factor*, quando se utiliza Inércia conseguiu um resultado melhor que os GNA5, GNA6, GNA7 e GNA8, apesar deste não ser bom se comparado com os GNA1, GNA2, GNA3 e GNA4.

Tabela 17. Resultado da função Griewank com topologia local e algoritmo de *Constriction Factor*.

Gerador	Tempo (min)	Resultado
GNA1	0.0767333	$0.00139676 \pm 1.03417 \cdot 10^{-5}$
GNA2	0.0892033	$0.00147814 \pm 1.53164 \cdot 10^{-5}$
GNA3	0.09814	$0.00164266 \pm 1.52072 \cdot 10^{-5}$
GNA4	0.0948433	$0.00188875 \pm 2.03691 \cdot 10^{-5}$
GNA5	0.11019	30.7623 ± 88.8672
GNA6	0.0852833	9.0945 ± 17.0386
GNA7	0.0803333	1.79847 ± 1.83
GNA8	0.14289	90.2366 ± 356.173
GNA9	0.08605	17.1498 ± 25.7423

Tabela 18. Resultado da função Griewank com topologia local e algoritmo de Inércia.

Gerador	Tempo (min)	Resultado
GNA1	0.08713	$0.00221505 \pm 3.60226 \cdot 10^{-5}$
GNA2	0.09997	$0.00320372 \pm 3.22974 \cdot 10^{-5}$
GNA3	0.108527	$0.000590021 \pm 4.88346 \cdot 10^{-6}$
GNA4	0.104897	$0.00295383 \pm 4.23086 \cdot 10^{-5}$
GNA5	0.107363	9.65085 ± 29.2328
GNA6	0.0804033	17.9148 ± 781.69
GNA7	0.0869467	18.4751 ± 126.836
GNA8	0.141763	55.5901 ± 174.811
GNA9	0.07951	2.08789 ± 0.154926

5.5 Função Schwefel 1.2

5.5.1 Topologia Global

A Tabela 19 e a Tabela 20 apresentam os resultados da função schwefel 1.2 para as abordagens *Constriction Factor* e Inércia, respectivamente, como topologia global.

Para os GNA1, GNA2, GNA3 e GNA4, os resultados com o algoritmo de *Constriction Factor* (Tabela 19) são muito melhores do que os resultados utilizando Inércia (Tabela 20). Vale salientar que os GNA1, GNA2, GNA3 e GNA4 são equivalentes quando comparados dentro da mesma abordagem. Os resultados para os geradores GNA5, GNA6, GNA7, GNA8 e GNA9 são sempre ruins e não se pode inferir muito sobre estes, pois possuem um desvio padrão muito maior do que a média.

Tabela 19. Resultado da função Schwefel 1.2 com topologia global e algoritmo de *Constriction Factor*.

Gerador	Tempo (min)	Resultado
GNA1	0.06052	$4.1181 \cdot 10^{-21} \pm 3.0089 \cdot 10^{-40}$
GNA2	0.07356	$5.8068 \cdot 10^{-21} \pm 3.13723 \cdot 10^{-40}$
GNA3	0.0815067	$4.75053 \cdot 10^{-21} \pm 2.09848 \cdot 10^{-40}$
GNA4	0.0760033	$5.58959 \cdot 10^{-21} \pm 3.55398 \cdot 10^{-40}$
GNA5	0.09787	$27965.3 \pm 4.05411 \cdot 10^{8}$
GNA6	0.0665833	$15582.2 \pm 3.74116 \cdot 10^{7}$
GNA7	0.0663733	$9013.8 \pm 1.47462 \cdot 10^{7}$
GNA8	0.11024	$17228.8 \pm 2.10032 \cdot 10^{7}$
GNA9	0.0751467	$15865.5 \pm 4.93132 \cdot 10^{7}$

Tabela 20. Resultado da função Schwefel 1.2 com topologia global e algoritmo de Inércia.

Gerador	Tempo (min)	Resultado
GNA1	0.0598233	4.41685 ± 11.3759
GNA2	0.0720333	4.06773 ± 13.6046
GNA3	0.0812667	5.65194 ± 54.6952
GNA4	0.0752267	6.1196 ± 53.5311
GNA5	0.09548	$7644.33 \pm 2.36221 \cdot 10^{7}$
GNA6	0.06068	$9200.26 \pm 2.4422 \cdot 10^{7}$
GNA7	0.0577133	$4873.07 \pm 8.00973 \cdot 10^{6}$
GNA8	0.110133	$12790.4 \pm 1.1495 \cdot 10^{7}$
GNA9	0.0761033	$4035.81 \pm 7.05473 \cdot 10^{6}$

5.5.2 Topologia Local

A Tabela 21 e a Tabela 22 apresentam os resultados da função schwefel 1.2 para as abordagens *Constriction Factor* e Inércia, respectivamente, como topologia local.

Para os GNA1, GNA2, GNA3 e GNA4, os resultados com o algoritmo de *Constriction Factor* (Tabela 21) são muito melhores do que os resultados utilizando Inércia (Tabela 22). Vale salientar que os GNA1, GNA2, GNA3 e GNA4 são equivalentes quando comparados dentro da mesma abordagem. Os resultados para os geradores GNA5, GNA6, GNA7, GNA8 e GNA9 são sempre ruins e não se pode inferir muito sobre estes, pois possuem um desvio padrão muito maior do que a média.

Tabela 21. Resultado da função Schwefel 1.2 com topologia local e algoritmo de *Constriction Factor*.

Gerador	Tempo (min)	Resultado
GNA1	0.0738567	$0.000253211 \pm 8.58115 \cdot 10^{-8}$
GNA2	0.08669	$0.000464972 \pm 2.26397 \cdot 10^{-7}$
GNA3	0.0945533	$0.000318765 \pm 8.77186 \cdot 10^{-8}$
GNA4	0.0892733	$0.00038942 \pm 1.429 \cdot 10^{-7}$
GNA5	0.0908	$5466.6 \pm 4.93658 \cdot 10^{7}$
GNA6	0.0754167	$7837.5 \pm 3.23988 \cdot 10^{7}$
GNA7	0.0774567	$11567.2 \pm 2.1149 \cdot 10^{7}$
GNA8	0.121363	$23776.8 \pm 2.18084 \cdot 10^{7}$
GNA9	0.0665967	2137.21 ± 878627

Tabela 22. Resultado da função Schwefel 1.2 com topologia local e algoritmo de Inércia.

Gerador	Tempo (min)	Resultado
GNA1	0.07283	148.136 ± 5483.16
GNA2	0.08605	148.296 ± 7736.67
GNA3	0.0944467	133.998 ± 4295.81
GNA4	0.08909	138.396 ± 4079.06
GNA5	0.0894167	1550.7 ± 705320
GNA6	0.0683733	$5461.23 \pm 2.18158 \cdot 10^{7}$
GNA7	0.0720667	$6752.2 \pm 1.27903 \cdot 10^{7}$
GNA8	0.122013	$15984.6 \pm 1.17946 \cdot 10^{7}$
GNA9	0.0654033	401.868 ± 48670.9

5.6 Função Schwefel 2.6

5.6.1 Topologia Global

A Tabela 23 e a Tabela 24 apresentam os resultados da função schwefel 2.6 para as abordagens *Constriction Factor* e Inércia, respectivamente, como topologia global.

Todos os geradores fornecem resultados ruins, tanto para *Constriction Factor* (Tabela 23), como para Inércia (Tabela 24), pois o desvio padrão de todos é sempre muito maior que sua média, entretanto os geradores GNA1, GNA2, GNA3 e GNA4 se saem melhor que os outros.

Tabela 23. Resultado da função Schwefel 2.6 com topologia global e algoritmo de *Constriction Factor*.

Gerador	Tempo (min)	Resultado
GNA1	0.0689733	4557.85 ± 261751
GNA2	0.0812867	4604.32 ± 192287
GNA3	0.0896333	4599.2 ± 307773
GNA4	0.0840467	4620.24 ± 315258
GNA5	0.10867	$6657.23 \pm 2.49049 \cdot 10^6$
GNA6	0.06897	7760.1 ± 614189
GNA7	0.0728733	$7121.86 \pm 1.48362 \cdot 10^6$
GNA8	0.11883	5515.32 ± 774132
GNA9	0.0813033	$6666.64 \pm 2.21008 \cdot 10^6$

Tabela 24. Resultado da função Schwefel 2.6 com topologia global e algoritmo de Inércia

Gerador	Tempo (min)	Resultado
GNA1	0.0683833	2334.56 ± 119502
GNA2	0.0807733	2223.36 ± 111698
GNA3	0.08952	2277.97 ± 92378.9
GNA4	0.0837533	2210.2 ± 193096
GNA5	0.0922	$7198.72 \pm 1.6312 \cdot 10^6$
GNA6	0.0626367	8177.38 ± 700908
GNA7	0.0627667	7945.22 ± 604963
GNA8	0.118417	6024.1 ± 495274
GNA9	0.06231	$6480.94 \pm 1.80416 \cdot 10^6$

5.6.2 Topologia Local

A Tabela 25 e a Tabela 26 apresentam os resultados da função schwefel 2.6 para as abordagens *Constriction Factor* e Inércia, respectivamente, como topologia local.

Similar aos resultados da topologia global, todos os geradores fornecem resultados ruins, tanto para *Constriction Factor* (Tabela 25), como para Inércia (Tabela 26), pois o desvio padrão de todos é sempre muito maior que sua média, entretanto os geradores GNA1, GNA2, GNA3 e GNA4 se saem melhor que os outros.

Tabela 25. Resultado da função Schwefel 2.6 com topologia local e algoritmo de *Constriction Factor*.

Gerador	Tempo (min)	Resultado
GNA1	0.08004	4621.08 ± 113270
GNA2	0.0937267	4371.94 ± 192169
GNA3	0.101007	4352.55 ± 311005
GNA4	0.0963	4535.76 ± 225459
GNA5	0.0973333	$7692.03 \pm 1.79284 \cdot 10^6$
GNA6	0.0783967	7639.26 ± 777052

GNA7	0.0802133	7648.73 ± 832241
GNA8	0.129953	6856.08 ± 254446
GNA9	0.0727	6567.36 ± 956441

Tabela 26. Resultado da função Schwefel 2.6 com topologia local e algoritmo de Inércia

Gerador	Tempo (min)	Resultado
GNA1	0.0795667	3047.25 ± 170951
GNA2	0.0924833	2906.29 ± 174526
GNA3	0.100673	2888.08 ± 187176
GNA4	0.09486	2996.92 ± 207537
GNA5	0.09803	$7247.4 \pm 1.00543 \cdot 10^{6}$
GNA6	0.0731567	8268.09 ± 379183
GNA7	0.0735533	8621.31 ± 180115
GNA8	0.12948	7201.23 ± 392716
GNA9	0.0726267	5133.23 ± 791043

5.7 Função *Penalized P8*

5.7.1 Topologia Global

A Tabela 27 e a Tabela 28 apresentam os resultados da função *Penalized* P8 para as abordagens *Constriction Factor* e Inércia, respectivamente, como topologia global.

Para os GNA1, GNA2, GNA3, os resultados com *Constriction Factor* (Tabela 27) e com o algoritmo Inércia (Tabela 28) são equivalentes. Os resultados para os geradores GNA5, GNA6, GNA7 e GNA8 são sempre piores e não se pode inferir muito sobre estes, pois possuem um desvio padrão muito maior do que a média. O GNA9 apesar de ser ruim utilizando *Constriction Factor*, quando se utiliza Inércia conseguiu um resultado melhor que os GNA5, GNA6, GNA7 e GNA8, apesar deste não ser bom se comparado com os GNA1, GNA2, GNA3.

Perceba também que nestes resultados, para a abordagem *Constriction Factor*, o GNA4, que é um GLC de qualidade mediana, não conseguiu resultados tão bons quanto os GNA1, GNA2, GNA3. Para Inércia o GNA4 possui um bom resultado.

Tabela 27. Resultado da função *Penalized* P8 com topologia global e algoritmo de *Constriction Factor*.

Gerador	Tempo (min)	Resultado
GNA1	0.0540067	$4.60693 \cdot 10^{-32} \pm 3.24799 \cdot 10^{-63}$
GNA2	0.0664333	$5.96807 \cdot 10^{-32} \pm 3.00076 \cdot 10^{-63}$
GNA3	0.0759267	$4.50223 \cdot 10^{-32} \pm 2.30897 \cdot 10^{-63}$
GNA4	0.0709333	0.0148616 ± 0.00662598
GNA5	0.13629	$126268 \pm 8.25918 \cdot 10^{10}$

GNA6	0.114257	$2.58712 \cdot 10^{6} \pm 1.39016 \cdot 10^{13}$
GNA7	0.0985667	$108921 \pm 3.27086 \cdot 10^{10}$
GNA8	0.147037	$1.48775 \cdot 10^{6} \pm 4.67538 \cdot 10^{12}$
GNA9	0.124537	$1.284 \cdot 10^{6} \pm 4.21459 \cdot 10^{12}$

Tabela 28. Resultado da função *Penalized* P8 com topologia global e algoritmo de Inércia.

Gerador	Tempo (min)	Resultado
GNA1	0.0772767	$5.13045 \cdot 10^{-32} \pm 2.60157 \cdot 10^{-63}$
GNA2	0.08814	$7.38156 \cdot 10^{-32} \pm 3.52385 \cdot 10^{-63}$
GNA3	0.09814	$4.13577 \cdot 10^{-32} \pm 2.16863 \cdot 10^{-63}$
GNA4	0.0931767	$4.55458 \cdot 10^{-32} \pm 2.47256 \cdot 10^{-63}$
GNA5	0.108457	$6852.16 \pm 2.08367 \cdot 10^{8}$
GNA6	0.0785567	$636299 \pm 2.09315 \cdot 10^{12}$
GNA7	0.0598667	$8179.48 \pm 7.29917 \cdot 10^{8}$
GNA8	0.143853	$183920 \pm 5.65106 \cdot 10^{10}$
GNA9	0.04788	$0.00131018 \pm 3.80024 \cdot 10^{-5}$

5.7.2 Topologia Local

A Tabela 29 e a Tabela 30 apresentam os resultados da função *Penalized* P8 para as abordagens *Constriction Factor* e Inércia, respectivamente, como topologia local.

Para os GNA1, GNA2, GNA3 e GNA4, os resultados com *Constriction Factor* (Tabela 29) e com o algoritmo Inércia (Tabela 30) são equivalentes. Os resultados para os geradores GNA5, GNA6, GNA7 e GNA8 são sempre piores e não se pode inferir muito sobre estes, pois possuem um desvio padrão muito maior do que a média. Contudo o GNA9 se mostra estável para utilização com com *Constriction Factor*, mesmo não sendo um bom resultado, porém para Inércia é um resultado tão bom quantos os quatro primeiros geradores.

Tabela 29. Resultado da função *Penalized* P8 com topologia local e algoritmo de *Constriction Factor*.

Gerador	Tempo (min)	Resultado
GNA1	0.06288	$3.14109 \cdot 10^{-33} \pm 4.08267 \cdot 10^{-65}$
GNA2	0.0758467	$8.37624 \cdot 10^{-33} \pm 1.65575 \cdot 10^{-64}$
GNA3	0.08613	$1.8323 \cdot 10^{-32} \pm 1.37932 \cdot 10^{-63}$
GNA4	0.0814633	$9.42327 \cdot 10^{-33} \pm 1.63307 \cdot 10^{-64}$
GNA5	0.0897333	$4098.43 \pm 2.27617 \cdot 10^{8}$
GNA6	0.07226	$39417.9 \pm 1.09843 \cdot 10^{10}$
GNA7	0.0652867	$4807.55 \pm 3.11249 \cdot 10^{8}$
GNA8	0.159487	$8.83513 \cdot 10^{6} \pm 3.44743 \cdot 10^{13}$
GNA9	0.0585267	0.155943 ± 0.104368

Tabela 30. Resultado da função *Penalized* P8 com topologia local e algoritmo de Inércia.

Gerador	Tempo (min)	Resultado
GNA1	0.0885467	$3.14109 \cdot 10^{-33} \pm 4.08267 \cdot 10^{-65}$
GNA2	0.10142	$4.71163 \cdot 10^{-33} \pm 5.3585 \cdot 10^{-65}$
GNA3	0.111197	$5.40725 \cdot 10^{-33} \pm 5.57282 \cdot 10^{-65}$
GNA4	0.106957	$6.80569 \cdot 10^{-33} \pm 1.64724 \cdot 10^{-64}$
GNA5	0.0827933	99.0416 ± 294068
GNA6	0.0729867	$109140 \pm 8.22408 \cdot 10^{10}$
GNA7	0.0776867	$44923.5 \pm 8.98807 \cdot 10^{9}$
GNA8	0.159703	$996046 \pm 6.75361 \cdot 10^{11}$
GNA9	0.0522633	$5.75866 \cdot 10^{-33} \pm 5.92554 \cdot 10^{-65}$

5.8 Função *Penalized P16*

5.8.1 Topologia Global

A Tabela 31 e a Tabela 32 apresentam os resultados da função *Penalized* P16 para as abordagens *Constriction Factor* e Inércia, respectivamente, como topologia global.

Os resultados com o algoritmo de *Constriction Factor* (Tabela 31), para os GNA1, GNA2, GNA3 e GNA4 são equivalentes. Apesar de um o desvio padrão é mais alto que a média. O mesmo ocorre para a abordagem da Inércia (Tabela 32). Os resultados para os geradores GNA5, GNA6, GNA7, GNA8 e GNA9 são sempre ruins, não se pode inferir muito sobre estes, pois possuem um desvio padrão muito maior do que a média.

Tabela 31. Resultado da função *Penalized* P16 com topologia global e algoritmo de *Constriction Factor*.

Gerador	Tempo (min)	Resultado
GNA1	0.09329	1.61547 ± 7.0551
GNA2	0.10505	2.83883 ± 35.2962
GNA3	0.113313	0.843086 ± 2.38121
GNA4	0.109063	1.28163 ± 7.08446
GNA5	0.113797	129.455 ± 7355.93
GNA6	0.0858333	175.854 ± 15645.9
GNA7	0.0861867	138.69 ± 7471.68
GNA8	0.141317	210.865 ± 11155.2
GNA9	0.0967367	122.852 ± 2937.67

Tabela 32. Resultado da função *Penalized* P16 com topologia global e algoritmo de Inércia.

Gerador	Tempo (min)	Resultado
GNA1	0.0912567	7.61763 ± 78.3364
GNA2	0.103253	6.11634 ± 115.204
GNA3	0.11278	7.37853 ± 141.031
GNA4	0.108143	9.10813 ± 211.078
GNA5	0.11838	49.4064 ± 1915.2
GNA6	0.0847967	70.4312 ± 3180.58
GNA7	0.08513	57.9035 ± 2057.01
GNA8	0.139513	75.8137 ± 1559.26
GNA9	0.0983933	2.73055 ± 5.21571

5.8.2 Topologia Local

A Tabela 33 e a Tabela 34 apresentam os resultados da função *Penalized* P16 para as abordagens *Constriction Factor* e Inércia, respectivamente, como topologia global.

Os resultados com o algoritmo de *Constriction Factor* (Tabela 33), para os GNA1, GNA2, GNA3 e GNA4 são bons. Assim como quando se utiliza a abordagem da Inércia (Tabela 34). Os outros geradores das duas abordagens fornecem resultados ruins e possuem um desvio padrão muito superior a média. Entretanto o GNA9 consegue um resultado diferenciado dos GNA5, GNA6, GNA7 e GNA8 utilizando a abordagem da Inércia, porém com resultados não tão bons se comparado com os GNA1, GNA2, GNA3 e GNA4.

Tabela 33. Resultado da função *Penalized* P16 com topologia local e algoritmo de *Constriction Factor*.

Gerador	Tempo (min)	Resultado
GNA1	0.103473	0.207151 ± 0.0966217
GNA2	0.116033	0.218944 ± 0.212254
GNA3	0.125853	0.122288 ± 0.0422931
GNA4	0.120687	0.20827 ± 0.104209
GNA5	0.122373	138.165 ± 4377.96
GNA6	0.09415	119.443 ± 4314.44
GNA7	0.0953633	142.238 ± 4413.54
GNA8	0.15123	240.937 ± 3935.69
GNA9	0.0966633	48.0452 ± 303.311

Tabela 34. Resultado da função *Penalized* P16 com topologia local e algoritmo de Inércia.

Gerador	Tempo (min)	Resultado
GNA1	0.101817	1.38405 ± 3.16874
GNA2	0.114577	0.877457 ± 1.31094
GNA3	0.12319	1.01459 ± 1.83368
GNA4	0.117953	1.89187 ± 5.3476
GNA5	0.120417	23.4228 ± 297.604

GNA6	0.0942433	60.2633 ± 1564.18
GNA7	0.0948967	70.0344 ± 1047.23
GNA8	0.150807	99.7387 ± 1270
GNA9	0.09483	4.05183 ± 3.94668

Capítulo 6

Conclusões e Trabalhos Futuros

Utilizando a bateria de testes proposta no trabalho, pode-se concluir que o PSO necessita de um gerador com qualidade mínima e período mínimo para ser capaz de convergir.

A qualidade mínima é uma conclusão dos resultados obtidos através dos testes com GNA5, ou seja, um GCL de baixa aleatoriedade, sendo este capaz apenas de gerar sequências numéricas em série, apesar de possuir um alto período.

A necessidade de um período mínimo pode ser observada pela análise do resultado para os GNA6, GNA7, GNA8 e GNA9, que são uma lista normalizada com 30, 300 e 3.000 elementos e um gerador estático, respectivamente. Apesar das listas possuírem boas qualidades aleatórias, uma vez que os números foram gerados por um gerador de Marsaglia, essas demonstram que por causa de seu baixo período não são capazes de gerar um bom resultado.

Isso se torna especialmente importante caso se deseje embarcar o algoritmo de PSO em algum sistema (roteadores, celulares e etc.), pois o custo embutido de um gerador de números aleatórios tem que ser considerado para que o algoritmo possa prover uma solução aceitável.

O tempo de execução pode ser considerado equivalente para todos os GNAs, não influenciando no tempo necessário para se obter as soluções, com uma pequena vantagem para o GNA1 e o GNA9, ou seja, um gerador de marsaglia e o gerador estático, respectivamente. Entretanto, para listas de maior quantidade de elementos o tempo cresce, enquanto o comportamento esperado era que esse se mantivesse estável, isso ocorre devido a dificuldades da linguagem JAVA em tratar grandes quantidades de memória alocada.

Dentre os possíveis trabalhos futuros, pode-se citar a necessidade de analisar o comportamento do PSO quando se fixa o GNA para os valores aleatórios na equação de velocidade e altera o GNA que inicializa as partículas no espaço de busca, verificando qual o impacto na capacidade de convergência algoritmo. Outro trabalho futuro é o teste com listas de maior valor (30.000, 300.000 e 3.000.000) para verificar qual o limiar mínimo necessário para que o algoritmo trabalhe de modo correto.

Bibliografia

[1] BASTOS FILHO, C. J. A. ; MELLO, Carlos Alexandre de Barros ; ANDRADE, Júlio Dantas de ; FALCÃO, Davi Marinho de Araújo ; LIMA, Marília Portela de ; SANTOS, W. P. ; OLIVEIRA, Adriano Lorena Inácio de . Thresholding Images of Historical Documents with Back-to-Front Interference based on Color Quantization by Genetic Algorithms. In: IEEE International Conference on Tools with Artificial Intelligence (ICTAI), 2007, Patras - Greece. Proceedings of IEEE International Conference on Tools with Artificial Intelligence (ICTAI), 2007. v. 1. p. 488-491.

[2] BASTOS FILHO, C. J. A. ; MELLO, Carlos Alexandre de Barros ; ANDRADE, Júlio Dantas de ; LIMA, Marília Portela de ; SANTOS, W. P. ; OLIVEIRA, Adriano Lorena Inácio de ; FALCÃO, Davi Marinho de Araújo . Thresholding Images of Historical Documents based on Color Quantization by Genetic Algorithms. In: Aleksandar Lazinica. (Org.). Tools in Artificial Intelligence. Vienna, Austria, EU: I-Tech Education and Publishing KG, 2008, v. , p. -.

[3] BASTOS FILHO, C. J. A. ; SCHULER, Wesnaida Holanda ; OLIVEIRA, Adriano Lorena Inácio de . A Fast and Reliable Routing Algorithm Based on Hopfield Neural Networks Optimized by Particle Swarm Optimization. In: 2008 International Joint Conference on Neural Networks (IJCNN2008) as part of 2008 IEEE World Congress on Computational Intelligence (WCCI2008), 2008, Hong Kong. Proceedings of 2008 IEEE World Congress on Computational Intelligence (WCCI2008), 2008.

[4] BRATTON, D., KENNEDY, J.: Defining a Standard for Particle Swarm Optimization. Proceedings of the IEEE Swarm Intelligence Symposium (2007).

[5] CASSIA-MOURA, R., SOUSA, C. S., RAMOS, A.D., COELHO, L. C. B. B., VALENÇA, M. M.: Yet another Application of the Monte Carlo Method for Modeling in the Field of Biomedicine. Computer Methods and Programs in Biomedicine 78, pp. 223-235 (2005).

[6] EBERHART, R. C.; SHI Y.; Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization. Proceedings of Congress on Evolutionary Computation, 2000, San Diego, CA. pp 84-88.

[7] Engelbrecht, A. P.; Computational Intelligence: An Introduction. Willey, First Edition, 2003.

[8] HAYKIN, S.; Neural Networks: A Comprehensive Foundation. Prentice Hall, Second Edition, 1998.

[9] L'ECUYER, P.; Tables of Linear Congruential Generators of Different Sizes and Good Lattice Structure. American Mathematical Society. Mathematics of Computation, Vol. 68, No. 225, 1999, pp 249-260.

[10] MARSAGLIA, G., Tsang, W.W.: The 64-bit Universal RNG. Statistics & Probability Letters 66, pp. 183-187 (2004).

[11] PITA, M. R. de S., BASTOS-FILHO, C. J. A.: Impacts of Random Numbers Generators Quality and Distribution in Particle Swarm Optimization Algorithms.

[12] PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T., FLANNERY, B. P.: Numerical Recipes, The Art of Scientific Computing. Third Edition, International Student Edition. Encontrado em http://sdu.ictp.it/nr/index.html, visitado em 12/08/2008.