Particle Swarm Optimization

1st Marlon Henry Schweigert
Departamento de Computação
Centro de Ciências Tecnológicas - UDESC
Joinville, Brasil
marlon.henry@magrathealabs.com

2nd Rafael Stubs Parpinelli
Departamento de Computação
Centro de Ciências Tecnológicas - UDESC
Joinville, Brasil
rafael.parpinelli@udesc.br

***Resumo*—Este meta-artigo descreve o funcionamento computacional do algoritmo Particle Swarm Optimization, utilizando o seu algoritmo padrão, alteração para busca com fator cognitivo e busca com inércia.**

***Index Terms*—PSO, inteligência artificial, modelagem matemática**

I. Introdução

Particle Swarm Optimization é um algoritmo de otimização para funções contínuas. Nesse sentido, em seu campo de busca pode ser qualquer função que receba como entrada valores de ponto flutuante, simbolizados como posições em um espaço de busca, a qual um conjunto de partículas procuram o seu ponto mínimo utilizando troca de informações entre entre os elementos destes grupos.

Entretanto, pode-se adicionar diversos componentes para a movimentação das partículas, sendo eles fatores cognitivos, sociais, ruído e inércia. Dessa forma, torna-se atrativo comparar o ganho a qual tem-se ao utilizar estas abordagens.

II. Fator Social

Este é o componente básico do PSO, na qual a partícula tenderá em direção da melhor partícula do enxame. Nesse sentido, este componente tende a realizar uma busca global no sistema.

Nesse sentido, a busca tende a ser somente global, convergindo de forma mais lenta e nem sempre alcançando o ponto mínimo da função.

III. Fator Cognitivo

Este componente armazena o melhor ponto encontrado durante a vida da partícula, colocando na soma de vetores, um vetor que aponte para a melhor posição da vida útil desta partícula.

Nesse sentido, a busca tende a ser local, já que cada partícula irá utilizar uma informação somente sua, evitando que diversas partículas sigam o mesmo caminho de busca.

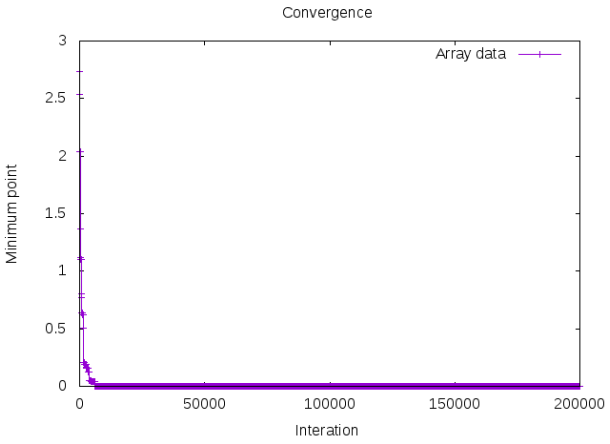
IV. Inércia

Este componente armazena a direção na qual a partícula está direcionando, para manter uma velocidade para aquela direção.

Este fator contribui para adicionar um comportamento a qual mantenha a partícula em movimento para o ponto mínimo, mesmo que este ainda não seja o ponto mínimo.

V. Análise utilizando a função Rastrigin

A convergência do algoritmo utilizando os 3 fatores percorre da seguinte maneira na Função Rastrigin:



Média de 10 execuções para a função rastrigin.

A busca foi realizada no espaço de -512.0 até 512.0, utilizando Percebe-se que a sua convergência é rápida, encontrando um ponto muito próximo do mínimo nas primeiras interações, com precisão de 10^-7.

VI. Análise utilizando a função Schewefel

A convergência do algoritmo utilizando os 3 fatores percorre da seguinte maneira na Função Schewefel:

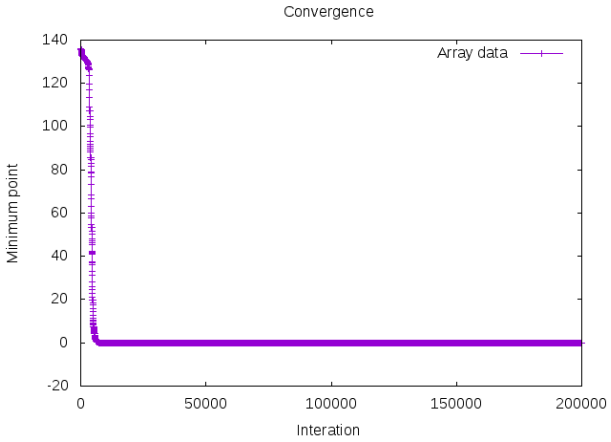
Percebe-se que a pesar da complexidade da função, o algoritmo encontra rapidamente o seu ponto de convergência muito próximo do ótimo, com precisão de 10^-6.

VII. Influência dos fatores

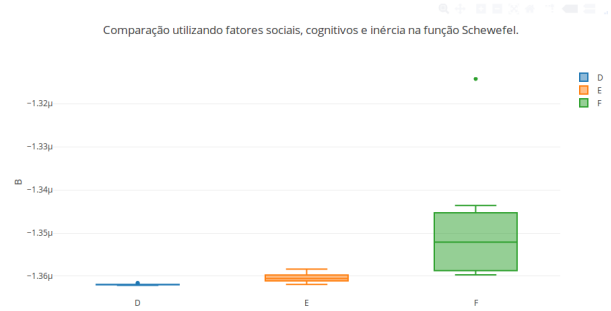
Para comparar a influência dos fatores, utilizamos um bloxplot, onde quanto menor a sua variação e o seu ponto médio, melhor em relação aos demais.

Neste gráfico, são emparelhados os experimentos A (busca utilizando fator cognitivo, social e inércia), B (busca utilizando fator cognitivo e social) e C (busca utilizando somente fator social).

Neste gráfico, são emparelhados os experimentos D (busca utilizando fator cognitivo, social e inércia), E (busca utilizando fator cognitivo e social) e F (busca utilizando somente fator social).



Média de 10 execuções para a função schewefel.



Comparação utilizando os fatores do PSO na função schewefel.

Percebe-se que utilizando somente busca global, a variação e precisão do resultado foi diminuído, em ambas as funções. Em contra partida, utilizando os demais fatores, a estabilidade da resposta foi maior, também contribuindo para uma melhor precisão.

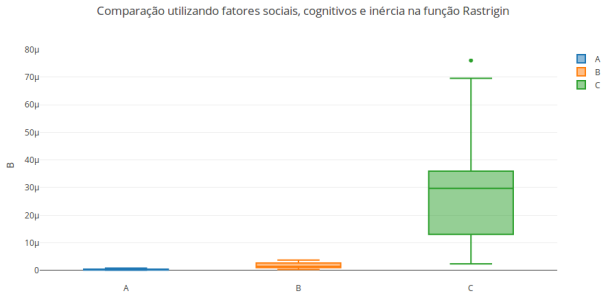
VIII. CONCLUSÃO

Percebe-se, ao analisar os gráficos de convergência, a facilidade a qual este algoritmo tem para otimizar funções multimodais. Comparado ao Simulated Annealing, este algoritmo utiliza busca local tardiamente, possivelmente pelo fato da distribuição uniforme dos elementos do enxame obter um posicionamento próximo ao ótimo no inicio da busca, facilitando a busca global, focando posteriormente na busca local.

Analisando a influência dos fatores, percebe-se que estes fatores contribuem para uma melhor otimização e por fim, uma estabilidade maior do resultado. Isto prova que estes fatores de fato contribuem para a otimização do problema.

Outros aspecto é que este algoritmo permite facilmente utilizar operações em GPU, acelerando o seu processo, visto que as suas operações são em grande parte multiplicações escalares e soma de vetores. Nesse sentido, pode-se obter um excelente aumento de desempenho, comparado a execução sobre CPU. Utilizando a biblioteca Matrix, nativa da linguagem Ruby, obteve-se um ganho significativo para a otimização destes casos de teste.

Outro ponto de desempenho que pode-se implementar sobre esse algoritmo é a implementação utilizando threads. Nos casos de testes deste trabalho foi utilizado 8 threads concorrentes.



Comparação utilizando os fatores do PSOna função rastrigin.