Universidade Federal de São João del-Rei

Algoritmo Bioinspirados

Particle Swarm Optimization

Felipe Francisco Rios de Melo

1. INTRODUÇÃO

Problemas de otimização modelados através de uma função matemática podem ser resolvidos através de métodos clássicos de otimização, como a programação linear e inteira, onde o objetivo é localização do valor ótimo para a função que representa o problema. Embora estas técnicas sejam muito utilizadas, o processo de localização do ótimo pode ser computacionalmente custoso. Como alternativa podem ser utilizados os Algoritmos Bioinspirados, como o PSO, que não garantem uma solução ótima mas podem chegar a um resultado subótimo que seja satisfatório com custo computacional aceitável.

Otimização por enxame de partículas (PSO, do inglês Particle Swarm Optimization) é um algoritmo heurístico baseado no comportamento social de um bando de pássaros, e tem como objetivo buscar a solução ótima, em um espaço de busca, através da troca de informações entre indivíduos de uma população determinando qual trajetória cada um deles deverá tomar no espaço de busca.

No PSO, as partículas são os indivíduos da população. Fazendo uma analogia, são os pássaros de um bando. Esses pássaros exploram uma região, determinado pela função objetivo (ou fitness), a fim de encontrar a solução ótima para o problema. A posição da melhor partícula da população será a melhor posição individual.

A grande vantagem de utilizar o PSO é a sua fácil implementação, usando somente estruturas primitivas e operadores matemáticos sem grande custo computacional.

Este trabalho tem por objetivo a implementação e análise de heurística PSO para a otimização de um função matemática de 3 variáveis.

2. IMPLEMENTAÇÃO

A seguir, será mostrado um resumo das decisões tomadas para a implementação do PSO:

2.1. Estrutura de dados

Foi implementado uma classe nomeada PSO, que é instanciada uma única vez, contendo os atributos e métodos necessários para inicialização e manipulação do algoritmo.

- Atributos: tamanho do enxame, quantidade de iterações, parâmetro c1 (taxa de aprendizado cognitiva), parâmetro c2 (a taxa de aprendizado social), parâmetro w (ponderação de inércia) e vetor de partículas;
- Métodos: gerar a população de partículas, definir a melhor solução da vizinhança, calcular a função objetivo, atualizar a velocidade de uma partícula, atualizar a solução de uma partícula, e etc.

Além da classe PSO, existe também a classe Particle, usada para armazenar o vetor solução do indivíduo para a função matemática, a aptidão do indivíduo na função objetivo, o vetor velocidade, além da melhor solução encontrada pelo indivíduo até o momento (*pbest*) e da melhor solução encontrada na vizinhança do indivíduo (*gbest*).

2.2 Configuração Inicial

Para cada partícula do enxame a sua configuração inicial é dada pelo sorteio dos valores iniciais para as três variáveis que irão compor o vetor de solução da partícula. Os valores sorteados para as variáveis estarão contidos no espaço de busca determinado individualmente por cada uma delas.

2.3. Função Objetivo

A função objetivo mapeia a qualidade do indivíduo diante da solução. Nosso objetivo é minimizar uma função benchmark com 3 parâmetros reais, listada pelo seguinte site: https://arxiv.org/pdf/1308.4008.pdf.

A função a ser minimizada é dada abaixo:

$$f_{60}(\mathbf{x}) = \sum_{i=1}^{99} \left[\exp\left(-\frac{(u_i - x_2)^{x_3}}{x_i} \right) - 0.01i \right]^2$$

Onde $u_i = 25 + [-50 \ln(0.01i)]^{1/1.5}$

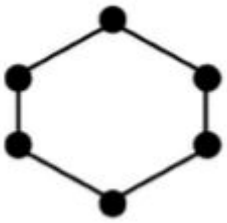
Sujeito ao seguinte espaço de busca: $0.1 \leq x_1 \leq 100$;
$0 \leq x_2 \leq 25.6$;
$0 \leq x_3 \leq 5$.

O mínimo global está localizado em: x* = f(50, 25, 1.5), f(x*) = 0.

O valor resultante da função objetivo é armazenado no atributo fitness na instância da classe *Particle,* referente a partícula em questão.

2.4. Topologia

Foi utilizada uma topologia dada por uma vizinhança de partículas adjacentes. Isto é, cada partícula tem dois vizinhos, que são as partículas vizinhas imediatas. Como exemplificado no grafo abaixo:



A função que retorna os índices dos dois vizinhos de uma determinada partícula é dada por:

```
neighbor1 = (i-1) % swarm_size
neighbor2 = (i+1) % swarm_size
```

onde *i* é o índice da partícula no vetor de partículas e *swarm_size* é o tamanho deste vetor. Foi utilizado a função modular para que o primeiro indivíduo pudesse ser vizinho do último indivíduo e vice-versa.

2.5. Atualização do *pbest* e do *gbest*

Para cada partícula, em cada iteração, é verificado se sua solução atual é melhor que sua melhor solução encontrada (*pbest*) até então, se sim o *pbest* passa a ser a solução atual.

Já para o *gbest*, para cada partícula, em cada iteração, a partir dos índices dos dois vizinhos da partícula, é atualizado o vetor que representa a melhor solução encontrada na vizinhança (*gbest*) deste indivíduo para a melhor solução encontrada até o momento pela partícula e seus vizinhos.

2.6. Atualização de velocidade

Para cada partícula, é atualizado em cada iteração a sua velocidade, o cálculo da velocidade é dado pela equação abaixo:

$$v_{ij}^{k+1} = wv_{ij}^{k} + c_1 r_{1j}(p_{best_{ij}}^{k} - x_{ij}^{k}) + c_2 r_{2j}(g_{best_j}^{k} - x_{ij}^{k})$$

Onde i = [1, tamanho do enxame] e j = [1, quantidade de dimensões da função].

A parte em vermelho é responsável pela diversificação da solução, enquanto a parte em azul é responsável pela intensificação.

Os três fatores que influenciam a nova velocidade é:

- velocidade anterior (parte em vermelho);
- distância entre sua posição atual e melhor posição alcançada até então (cognitivo) (primeira soma da parte em azul);
- distância entre sua posição atual e a melhor posição do grupo conhecido por ela (social) (segunda soma da parte em azul);

2.7. Atualização da posição

Dada a nova velocidade, a posição de cada partícula no espaço de busca é atualizada.

A atualização da posição é dada pela soma vetorial da posição atual mais a nova velocidade calculada, como é mostrado na equação abaixo:

$$x_i = x_i + v_i$$

Onde *i* é a dimensão da função.

2.8. Processo Iterativo

O processo de aplicação na função objetivo (seção 2.3) até a atualização da posição (2.7) é executado para cada partícula, que por sua vez, é repetido para cada iteração.

3. ANÁLISE DOS RESULTADOS

Para calibrar os parâmetros do algoritmo aplicado a função matemática utilizada, foi realizado um experimento fatorial completo variando a quantidade de iterações, o parâmetro c1 (taxa de aprendizado cognitiva), o parâmetro c2 (taxa de aprendizado social) e parâmetro w (ponderação de inércia), nos seguintes valores:

Número de iterações	50	100	150

c1	0.5	1.5	2.5	
c2	0.5	1.5	2.5	
w	0.25	0.50	0.75	1

Variando estes parâmetros temos 108 combinações possíveis, e realizado 20 execuções do algoritmo, para obter um resultado confiável, resulta num total de 2160 execuções, estas foram efetuadas de maneira automatizada pelo arquivo python *analysis.py*.

O único parâmetro que não foi variado foi o tamanho da população. Na literatura, é recomendado por alguns autores, utilizar uma população 10x a dimensão do problema. Como estamos trabalhando com uma função de 3 dimensões, foi fixado o tamanho da população de partículas igual a 30.

A identificação da combinação de parâmetros referente aos dados mostrados nesta análise estarão no padrão (*Quantidade de iterações - c1 - c2 - w*). Por exemplo: 100-2.5-1.5-0.25.

Para iniciar a calibragem de parâmetros, a seguinte tabela, contém as combinações, junto à média do melhor fitness de cada uma das 20 execuções para cada combinação. É mostrado, na tabela, somente as 10 melhores e piores combinações, a tabela completa está no diretório do trabalho no arquivo *fitness.txt*.

TABELA 1. combinações e suas respectivas médias de melhor fitness encontrado, ordenado pela combinação com melhor fitness médio.

Combinação	**Fitness médio**
150-1.5-0.5-0.75	3.491692426246499e-05
150-0.5-0.5-0.75	5.833842029091679e-05
100-1.5-0.5-0.75	6.90625341324117e-05
150-1.5-1.5-0.25	0.00011633051838731425
100-0.5-0.5-0.75	0.00016919972394501348
150-0.5-1.5-0.5	0.00017439001516749 9
150-2.5-0.5-0.75	0.00018078722619083 3
100-0.5-1.5-0.25	0.00023744900657131
150-2.5-1.5-0.5	0.00025625574065259654

150-0.5-1.5-0.75	0.00025730457961277176
	.
	.
	.
50-1.5-0.5-0.5	0.0394956235203884
100-2.5-0.5-0.25	0.07442248786261899
150-2.5-0.5-0.25	0.08233646044296863
150-1.5-0.5-0.25	0.1638634163726583
50-2.5-0.5-0.25	0.16754281095580198
50-1.5-0.5-0.25	0.24853010355217778
100-1.5-0.5-0.25	0.25419591965751154
50-0.5-0.5-0.25	0.2693257180485127
150-0.5-0.5-0.25	0.43793947599028
100-0.5-0.5-0.25	0.45168989516451663

A notar pelas primeiras que são as combinações que alcançaram o melhores fitness médio, já podemos observar um certo padrão: as combinações com uma maior quantidade de iterações (100 e 150) se sobressaíram em relação às combinações com menor número de iterações (50). O que é de se esperar, pois quanto mais iterações mais “tempo” as soluções têm para se intensificarem. Porém em contrapartida, quanto mais iterações mais custoso computacionalmente se torna o algoritmo. As combinações com 50 iterações levaram um tempo médio de execução de 0.8s, enquanto que as combinações com 150 iterações levaram 0.24s para executarem.

Embora as combinações com um maior número de iterações conseguiram obter os melhores fitness médio, ainda assim, é possível observar algumas combinações com 100 e 150 iterações na parte inferior da tabela. Mas nestes casos, o parâmetro responsável pelo baixo desempenho em minimizar a função objetivo não é a quantidade de iterações, mas sim os outros parâmetros. Como por exemplo o parâmetro w, que na equação de velocidade é responsável por ponderar a influência da velocidade da iteração anterior na velocidade da iteração atual. Variando este parâmetro estamos ajustando a habilidade de busca global e local do PSO. Se w = 0 a velocidade da partícula irá depender apenas do *pbest* e do *gbest* corrente. Se w != 0, a partícula terá uma tendência a explorar novos espaços, onde que quanto maior o w, a partícula tenderá a explorar novos espaços de maneira mais incisiva.

Resumidamente, quanto maior o valor do parâmetro w, maior o fator de diversificação do PSO.

Notamos que, das 10 piores combinações 9 delas tem 0.25 como valor do parâmetro w, isto é, essas combinações têm um baixo grau de diversificação, o que pode acarretar em uma concentração de partículas em um ótimo local. No trecho mostrado da tabela 1 (a tabela completa está no diretório do trabalho), não é possível ver, mas logo após as 10 piores combinações temos uma sequência de 25 combinações com w = 1, em um curto espaço de 29 combinações. O w = 1 acarreta em alto grau de diversificação, logo pode ser que as partículas estão se diversificando tanto a ponto de não conseguirem se intensificar para formar uma solução mais robusta, não está havendo um bom equilíbrio entre diversificação e intensificação, o que é essencial para algoritmos de busca e otimização.

Nos gráficos abaixo pode ser visto o comportamento de três combinações iguais, alterando somente o parâmetro w.

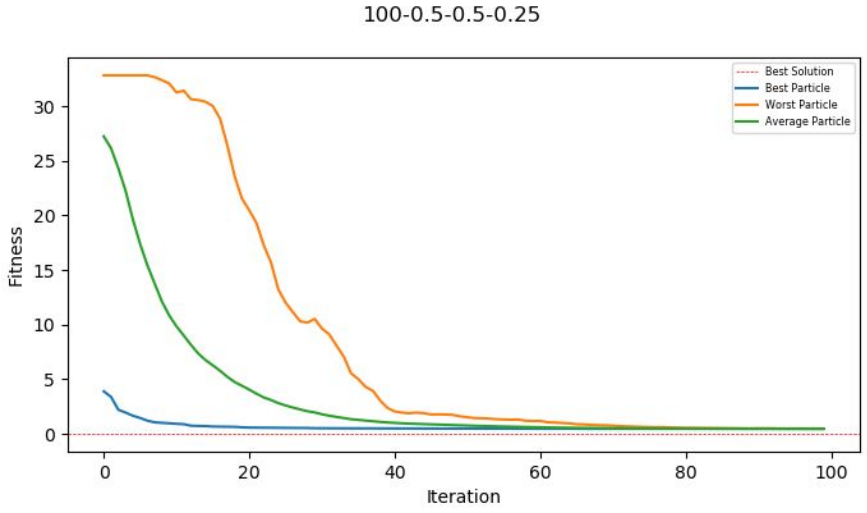


Gráfico 1. combinação com w = 0.25, fitness médio da melhor partícula = 0.4516

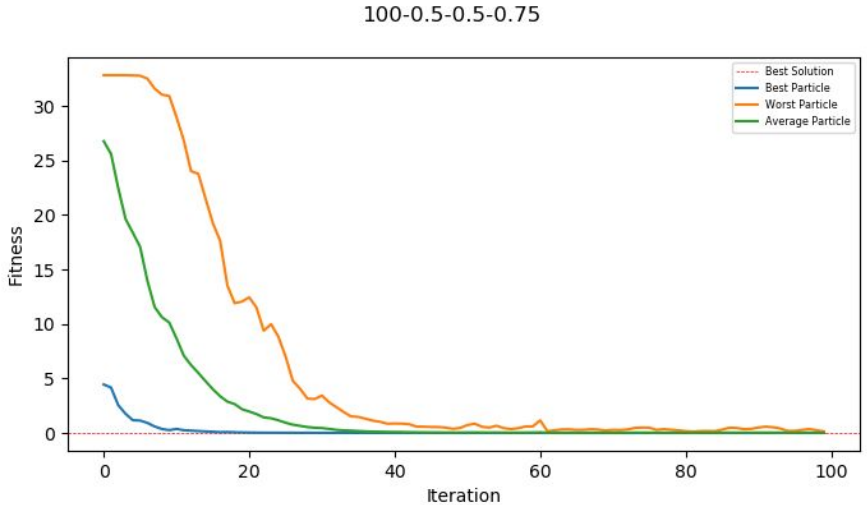


Gráfico 2. combinação com w = 0.75, fitness médio da melhor partícula = 0.0001

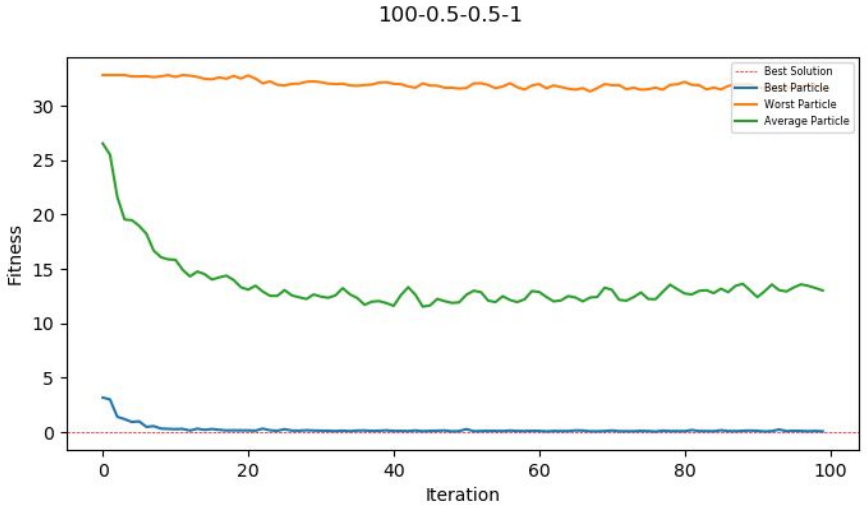


Gráfico 3. combinação com w = 1, fitness médio da melhor partícula = 0.0039

Vemos a partir do gráfico 3, que a população de um modo geral não conseguiu convergir na combinação com w = 1, embora o melhor indivíduo tenha chegado próximo do ótimo. Este comportamento de não conversão para w = 1, foi uma máxima para todas as combinações com esta característica, mostrando, não ser um valor apropriado se o objetivo for alcançar uma convergência geral. Quanto às combinações mostradas no gráfico 1 e 2, observamos que as populações em geral conseguiram acompanhar os seus respectivos melhores indivíduos, mas com uma diferença, a combinação com w = 0.75, de fato, chega muito próximo do ótimo, enquanto que a combinação com w = 0.25 se estabiliza em um certo ponto e não evolui mais. O que aconteceu foi que as partículas ficaram presas em um ótimo local devido ao baixo valor de w. Enquanto que para w = 0.75 houve um bom equilíbrio entre explorar pouco e ficar estagnado em um ótimo local e explorar muitos novos espaços e não conseguir se intensificar, provando ser um bom valor para w.

Vamos agora, analisar os parâmetros de intensificação, c1 e c2 são parâmetros que significam respectivamente uma taxa de aprendizado cognitiva e uma taxa de aprendizado social. O c1 está diretamente relacionado à melhor solução encontrada pela partícula até o momento (*pbest*) e o c2 relacionado a melhor solução encontrada pela vizinhança até o momento (*gbest*). Se c1 = 0, a partícula irá basear seu próximo movimento apenas em seu conhecimento social, já se c2 = 0, a partícula irá basear seu próximo movimento apenas no seu conhecimento obtido pessoalmente.

Abaixo temos os gráficos das melhores combinações de c1 e c2, com o tamanho da população fixado em 150 e w = 0.75:

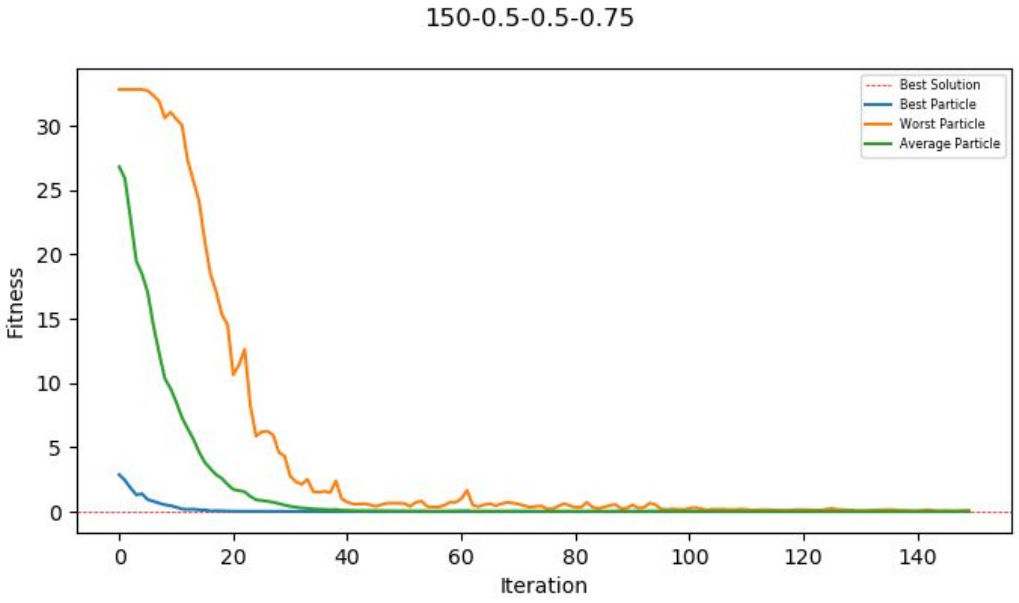


Gráfico 4. combinação com c1 = 0.5 e c2 = 0.5, fitness médio da melhor partícula = 0.00006

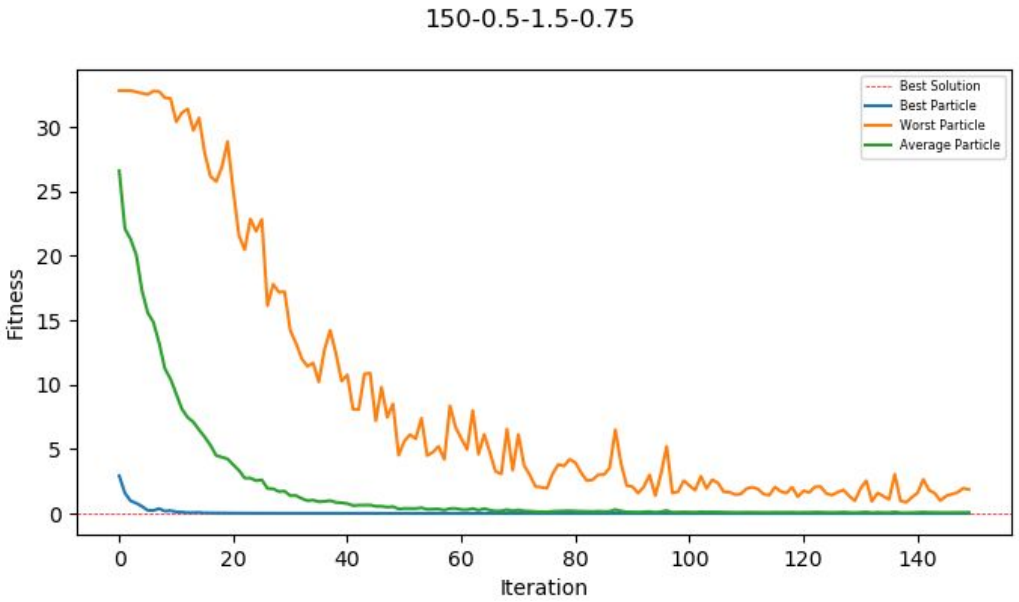


Gráfico 5. combinação com c1 = 0.5 e c2 = 1.5, fitness médio da melhor partícula =0.00025

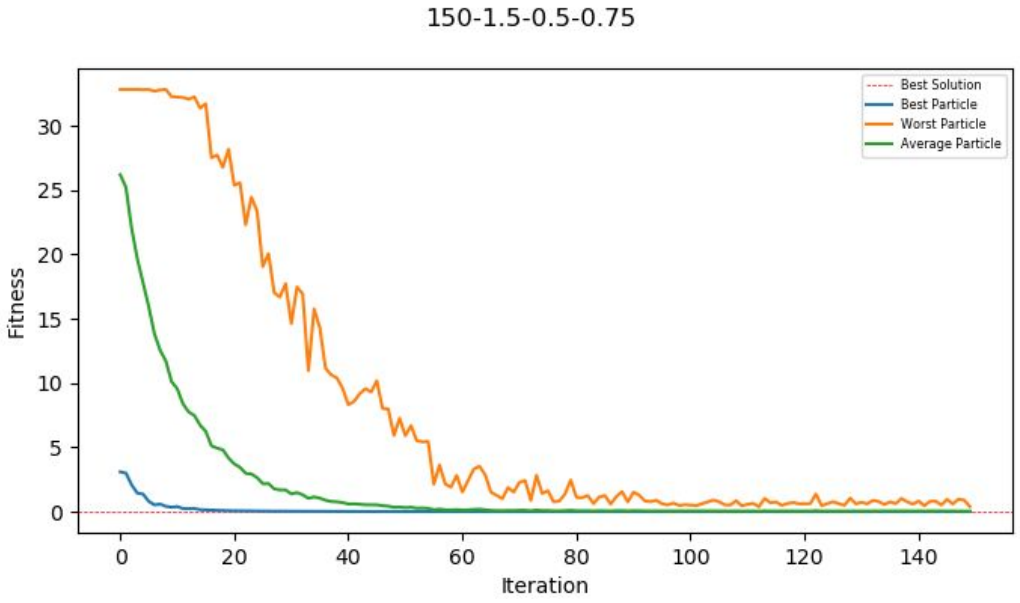


Gráfico 6. combinação com c1 = 1.5 e c2 = 0.5, fitness médio da melhor partícula = 0.00003

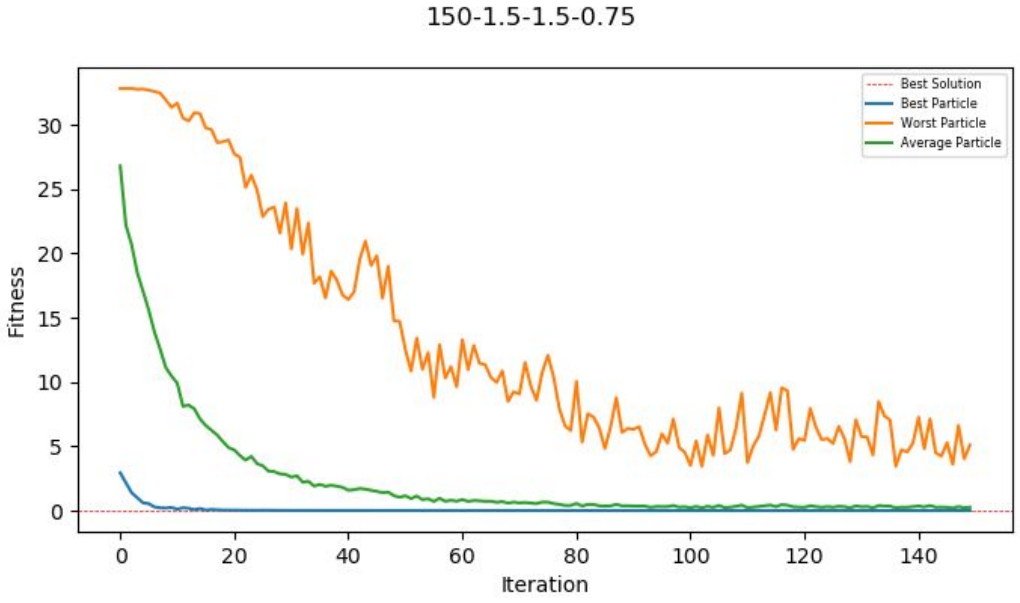


Gráfico 7. combinação com c1 = 1.5 e c2 = 1.5, fitness médio da melhor partícula = 0.00037

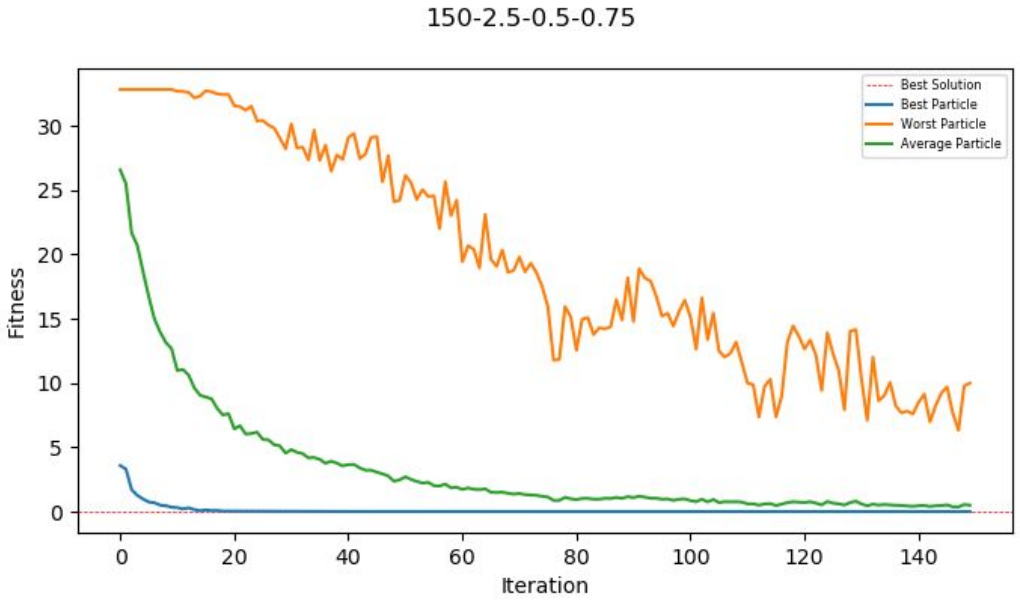


Gráfico 8. combinação com c1 = 2.5 e c2 = 0.5, fitness médio da melhor partícula = 0.0039

Apesar de a melhor solução ser dada apenas por um único indivíduo, àquele que chegou mais próximo do ótimo global, é interessante que a população de modo geral também chegue próximo desse ótimo, pois quanto mais partículas “voando” no espaço de busca próximo do ótimo (sem permitir agrupamento em ótimos locais), maiores as chances de aprimoramento da melhor solução encontrada.

Logo, analisando a convergência da população de uma maneira geral, vemos que a combinação do gráfico 8, com c1 = 2,5 e c2 = 0.5, a média das partículas (linha verde) demorou a se aproximar da média melhor partícula (linha azul) levando mais de 100 iterações para convergir. Em contrapartida, a combinação apresentada no gráfico 4, com c1 = 0.5 e c2 = 0.5, convergiu rapidamente, precisando somente de 40 iterações.

Na literatura, estudos iniciais do PSO, sugeriam que c1 = c2 = 2.0. Embora bons resultados tenham sido obtidos, foi observado que as velocidades explodiram rapidamente para grandes valores. Consequentemente, as partículas têm grandes atualizações de posição, com partículas extrapolando os limites de seu espaço de busca.

Empiricamente, as combinações com os coeficientes de aceleração c1 e c2 menores ou iguais a 1.5, resultaram em um bom comportamento de convergência.

4. CONCLUSÃO

Com este exercício de implementação foi possível aplicar o conceito de um algoritmo baseado em enxame de partículas na otimização de uma função matemática de 3 variáveis.

Em relação ao experimento fatorial completo, os melhores valores para a constante de inércia (w) foi 0.75, bons resultados foram obtidos com 0.5 também. Para os valores de c1 e c2, as melhores soluções foram obtidas com c1 e c2 menores ou iguais a 1.5, a melhor combinação foi c1 = c2 = 0.5. Quanto a quantidade de iteração, 150 iterações se mostrou um bom valor, não custa muito tempo e promove uma boa solução.

Para análises futuras, seria interessante agora realizar um experimento variando o w entre 0.5 e 0.75, e variando o c1 e c2 para uma maior quantidade de valores entre 0.5 e 1.5 para obter resultados mais concisos, além também de experimentar novas topologias, que pode ser bem influente no resultado obtido pelo algoritmo.

5. REFERÊNCIAS BIBLIOGRÁFICAS

Jamil, Momin, and Xin-She Yang. "A literature survey of benchmark functions for global optimisation problems." *International Journal of Mathematical Modelling and Numerical Optimisation* 4.2 (2013): 150-194. Acesso em: 22 Novembro 2020.

Pacheco, André. “Otimização por enxame de partículas - PSO.” *Computação Inteligente*, 2016, http://computacaointeligente.com.br/algoritmos/otimizacao-por-enxame-de-particulas/. Acesso em: 22 Novembro 2020.

Silva, Allan F., Afonso CC Lemonge, and Beatriz SLP Lima. "Algoritmo de Otimização com Enxame de Partículas auxiliado por Metamodelos." *XI Simpósio de Mecânica Computacional, II Encontro Mineiro de Modelagem Computacional, SIMMEC/EMMCOMP* (2014). Acesso em: 23 Novembro 2020.

He, Yan, Wei Jin Ma, and Ji Ping Zhang. "The parameters selection of pso algorithm influencing on performance of fault diagnosis." *MATEC Web of conferences*. Vol. 63. EDP Sciences, 2016. Acesso em: 23 Novembro 2020.