**Universidade Tecnológica Federal do Paraná**

Campus Toledo

# Algoritmo *Particle Swarm Optimization* aplicado na função *Rosenbrock*

Luiz Fernando da Costa Pereira

16 de agosto de 2024

# 1 Introdução

Diversos avanços tecnológicos são inspirados inicialmente pela natureza. Na área da inteligência artificial, um dos exemplos mais famosos é o conceito de Redes Neurais Artificiais que foi modelado no intuito de espelhar o funcionamento de redes de neurônios biológicos (HOPFIELD, 1988). Outro exemplo seria o algoritmo *Particle Swarm Optimization* (PSO), inspirado no comportamento de enxame de animais como pássaros, peixes e abelhas.

A fim de apresentar o funcionamento do algoritmo PSO e sua eficiência, foi utilizada a função *Rosenbrock*, bem conhecida por ser uma função de *benchmark* para testes de modelos de otimização, como o modelo em questão. A seguir é apresentado os conceitos utilizados no algoritmo e a função objetivo em detalhes. Depois seguem os valores e metodologia aplicada, e os resultados e conclusões.

# 2 Referencial Teórico

Para concluir a contextualização e apresentação do problema, torna-se relevante mencionar a teoria por trás do algoritmo, bem como a abstração computacional do problema.

## 2.1 *Particle Swarm Optimization*

O algorítimo de enxame de partículas é um método não-linear eurístico de otimização baseado no comportamento populacional de rebanhos de aves, enxames de insetos, e semelhantes. Seu funcionamento se consiste num número de partículas instanciadas aleatoriamente, em que a cada iteração, suas posições e velocidades são atualizadas, até que uma condição seja satisfeita, ou atinja o máximo de iterações (KENNEDY; EBERHART, 1995). Geralmente em otimização, a condição se consiste em que as partículas convirjam no ótimo global da função (valor mínimo).

Os autores Kennedy e Eberhart (1995) exemplificam que quando há pássaros em grupo, se um deles encontra comida, é comum que os outros apareçam no mesmo lugar. Nota-se que embora o primeiro pássaro tenha encontrado o local sozinho, houve um compartilhamento de informações entre o rebanho a respeito da localização da comida. As partículas devem se comportar da mesma forma. Cada uma "explora" a área da função objetivo e adquirem uma memória própria do melhor local que já ela passou, e uma memória coletiva, do melhor local em que uma partícula do enxame já passou.

Partindo para as formulações matemáticas do modelo, as declarações do parágrafo anterior podem ser traduzidas para um comportamento de partículas com respeito as suas posições e velocidades, de modo que esse último seja representado pela equação:

$$V_i = \omega V_i + c_1 r_1 (pbest_i - p_i) + c_2 r_2 (gbest - p_i), \tag{1}$$

em que $V_i$ que aparece do lado esquerdo da igualdade, é a nova velocidade da partícula $i$ e, o $V_i$ da direita se refere à atual velocidade, e $p_i$ a posição. $r_1$ e $r_2$ são números randômicos, e $c_1$ e $c_2$ são coeficientes que ponderam os vetores de deslocamento baseado na própria exploração ($pbest$), e deslocamento baseado no melhor resultado do grupo $gbest$, respectivamente. A variável $\omega$ foi um aprimoramento de Shi e Eberhart (1998) chamado de inércia, que regula a contribuição da velocidade atual na soma dos vetores, e consequentemente contribui para uma exploração mais global (para maiores valores de $\omega$), ou mais local (valores mais baixos de $\omega$).

Figura 1: Comparação entre diferentes de $\omega$


(a) Alto valor de inércia

(b) Baixo valor de inércia

Dessa forma, percebe-se que o vetor velocidade é a soma de outros vetores, podendo ser reescrito com a expressão $V_i = V_0 + V_{c_1} + V_{c_2}$, de modo que $V_0$ = vetor inercia = $\omega V_i$, $V_{c_1}$ = exploração independente = $c_1 r_1(pbest_i - p_i)$ e $V_{c_2}$ = deslocamento em grupo = $c_2 r_2(gbest - p_i)$.

A segunda equação atualiza a posição da partícula com a nova velocidade $V_i$:

$$X_i = X_{i-1} + V_i, \tag{2}$$

onde $X_i$ é a nova posição da partícula $i$ e $X_{i-1}$ é a posição anterior.

## 2.2 Analisando os coeficientes

A cada iteração que as fórmulas 1 e 2 são computadas para cada dimensão, ocorre uma soma de vetores que definem uma área de deslocamento possível para cada partícula. A Figura 1 mostra como o valor da inércia $\omega$ afeta a área de exploração possível. Como já mencionado, um menor valor de $\omega$ reduz a magnitude do vetor de inércia e consequentemente a área de exploração em que a partícula poderá visitar, dependendo dos valores dos outros vetores. Caso contrário, a área de exploração é maior, resultando em uma busca global.

Por isso, Shi e Eberhart (1998) mostram que a solução mais eficiente é variar o valor de $\omega$ linearmente de forma decrescente a cada iteração. Isso significa que no início as partículas partem de uma busca global para uma local, até que atinjam ponto de convergência.

Já os coeficientes $c_1$ e $c_2$ foram analisados por Kennedy e Eberhart (1995) e testados diversos valores diferentes. Os coeficientes podem ser alterados conforme a necessidade do problema, entretanto se mantiverem valores $c_1 = c_2 = 2$, em média o peso de cada vetor (particular e social) é 1 (considerando $0 < r_1, r_2 < 1$), fazendo com que as partículas orbitem o ponto de convergência (KENNEDY; EBERHART, 1995).

## 2.3 *Exploitation vs Exploration*

As implicações dos valores de $c_1$ e $c_2$ portanto, dizem se as partículas tendem para uma abordagem mais *Exploration* (mais exploração independente,$c_1 > c_2$) ou *Exploitation* (uma exploração em conjunto, sempre tendendo ao último ótimo global, $c_1 < c_2$). Se ambos os coeficientes forem iguais, tem-se um equilíbrio entre as duas técnicas, o que é o ideal principalmente para esse algoritmo, que ao encontrar uma solução, pode acabar encontrando apenas uma solução local, e não o ótimo global (G.; MUTHUKUMARASWAMY; LOO, 2009).

### 2.4 A função *Rosenbrock*

A função objetivo adotada nesse trabalho foi a função *Rosenbrock* de duas dimensões, apresentada na equação 3.

$$f(x,y) = \sum_{i=1}^{N} [100(y - x^2)^2 + (x-1)^2]. \tag{3}$$

Além de ser uma ótima função para *benchmark* de algoritmos de otimização, seu valor ótimo global é conhecido: $f(1,1) = 0$. A função pode ser gerada com o código anexo a este relatório, e é apresentada na Figura 2

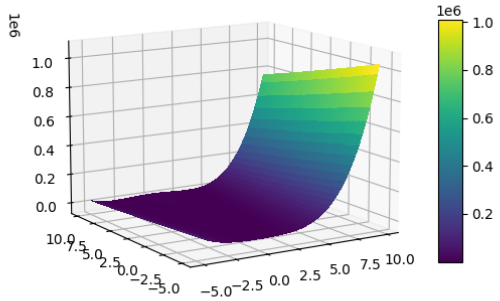


Figura 2: Função objetivo

## 3 Metodologia

Para a implementação do algoritmo PSO foram adotados os seguintes valores para os coeficientes: $c_1 = c_2 = 1.5$, adotado por meio de testes empíricos; 10 partículas e 100 iterações máximas, totalizando 1000 avaliações da função objetivo; e velocidade máxima de 2.5. Para analisar o impacto da inércia, o algoritmo foi implementado visando suportar os dois comportamentos de $\omega$: pode possuir valor constante de 0.9, ou ter um comportamento linear decrescente partindo de 0.9 até 0.4.

## 4 Resultados

Considerando a inércia constante, obteve-se como melhor posição $gbest_x = 0.98865707$, $gbest_y = 0.96034373$ com o melhor valor $f(gbest_x, gbest_y) = 0.02936646$. O resultado pode ser visto na Figura 3 onde mostra a função objetivo através do plano Z. Os pontos cinzas são as partículas e o ponto vermelho é o ótimo global.

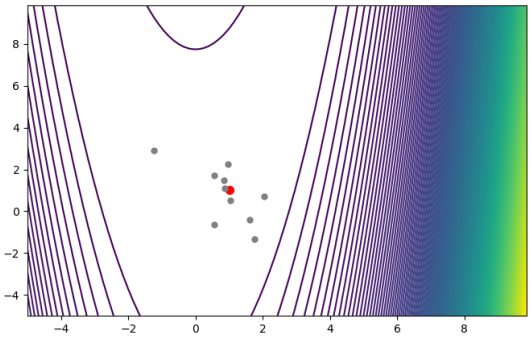

Figura 3: PSO com inércia constante

Agora com inércia linear, obteve-se os resultados: $gbest_x = 0.92038345$, $gbest_y = 0.84690989$ e o melhor valor $f(gbest_x, gbest_y) = 0.00634263$. A Figura 4 apresenta o resultado final.

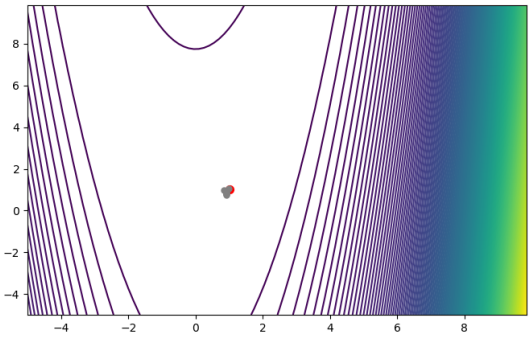

Figura 4: PSO com inércia linear decrescente

Comparando os dois algoritmos gerados, tem-se o gráfico da Figura 5 em que "PSO1" é o algoritmo com inércia constante, e "PSO2" ou "Improved PSO", de inércia linear.

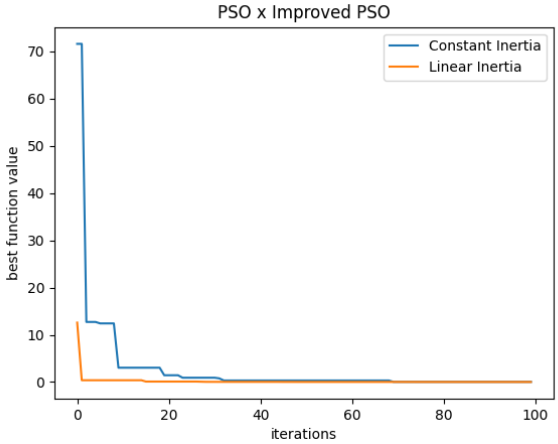


Figura 5: PSO com inércia linear decrescente

Note como "Improved PSO" mantém o menor erro desde o início é apenas aproximado por "PSO1" por volta da iteração de número 33.

# 5 Conclusão

Portanto, nota-se que o algoritmo é eficiente e simples, e através das comparações realizadas nesse trabalho, o impacto de uma inércia linear afeta positivamente o desempenho e eficiência do algoritmo. O modelo continua a evoluir, com modificações e melhoramentos feitos a cada estudo, como variar coeficientes $c_1$ e $c_2$ linearmente assim como a inércia, aplicar *cross-over* e erro quadrático médio (RMS), e assim por diante (NIU; DAI; PENG, 2012).

# Referências

G., Ramana Murthy; MUTHUKUMARASWAMY, Senthil Arumugam; LOO, Chu Kiong. Hybrid particle swarm optimization algorithm with fine tuning operators. **IJBIC**, v. 1, p. 14–31, jan. 2009. DOI: 10.1504/IJBIC.2009.022771.

HOPFIELD, J.J. Artificial neural networks. **IEEE Circuits and Devices Magazine**, v. 4, n. 5, p. 3–10, 1988. DOI: 10.1109/101.8118.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: PROCEEDINGS of ICNN'95 - International Conference on Neural Networks. [S.l.: s.n.], 1995. v. 4, 1942–1948 vol.4. DOI: 10.1109/ICNN.1995.488968.

NIU, Hui; DAI, Yongshou; PENG, Xing. Particle Swarm Optimization With Adaptive Parameters and Boundary Constraints. **International Journal of Engineering and Manufacturing**, v. 2, p. 19–28, ago. 2012. DOI: 10.5815/ijem.2012.04.03.

SHI, Y.; EBERHART, R. A modified particle swarm optimizer. In: 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360). [S.l.: s.n.], 1998. P. 69–73. DOI: 10.1109/ICEC.1998.699146.