Tópicos Especiais em Otimização I
2023/2
Atividade 1: Otimização por Enxame de Partículas (PSO)

Filipe Gomes Arante de Souza

Setembro de 2023

Conteúdo

1 Introdução

A otimização por enxame de partículas (PSO) é um algoritmo evolutivo que surgiu dos estudos oriundos de simulações de modelos sociais simplificados de um bando de pássaros. Desde sua primeira publicação por James Kennedy e Russel Eberhart [1] em 1995, este algoritmo tem sido modificado continuamente na tentativa de melhorar as suas propriedades de convergência, possuindo diversas variantes que se aplicam em problemas de telecomunicações, controle, mineração de dados, design, otimização combinatória, sistemas de energia, processamento de sinais, dentre outras. Além disso, o PSO possui grande simplicidade e é de fácil implementação, o que o torna um dos algoritmos populacionais mais conhecidos e amplamente utilizados atualmente.

Este artigo tem o propósito de abordar a versão original do PSO, apresentando seu histórico, conceitos e um exemplo de aplicação para permitir uma compreensão geral sobre esta meta-heurística.

Palavras Chave: Otimização por Enxame de Partículas, Comportamento Social, Computação Natural.

2 Estado da Arte

O PSO tem sido muito efetivo na literatura em problemas de otimização contínua e combinatória. Cabe aqui destacar alguns deles:

- [2] resolve o problema do Caixeiro Viajante com uma versão discreta do algoritmo, nomeada *MDPSO*, apresentando uma nova definição para a velocidade baseada no Algoritmo Genético;
- [3] resolve o problema do Caminho Mais Curto Restrito utilizando o *PSO* junto ao *VNS* para um cálculo otimizado da posição das partículas;
- [4] faz uma análise do desempenho estático-dinâmico e ondas propagadas na nanoplaca magneto-eletro-elástica utilizando o *PSO* como um dos operadores do Algoritmo Genético;

3 Histórico

Por volta das décadas de 1980 e 1990, cientistas estavam interessados em compreender o movimento de grupos de espécies de animais, como por exemplo bandos de pássaros ou cardumes de peixes. Pesquisadores como Reynolds, Heppner e Grenander se intrigavam com a estética das aves em coreografia, pois elas voavam sincronizadas, trocando de direção repentinamente e se dispersando e reagrupando várias vezes. A hipótese inicial para formalização matemática do movimento era que seu comportamento se baseava numa função dos esforços dos indivíduos para manter uma distância ideal entre eles mesmos e seus vizinhos. [1]

Partindo deste contexto, várias simulações computacionais de um meio social simplificado foram desenvolvidas. Geralmente elas eram reproduzidas em ambientes de duas dimensões, onde cada membro do bando era considerado um ponto a prova de colisões.

O principal algoritmo que reproduzia a mobilidade do grupo foi o de *Correspondência de Velocidade e Loucura do Vizinho mais Próximo* [1], cuja ideia é a seguinte:

1. Inicializar os pontos aleatoriamente numa grade de pixels, cada um com velocidade (v_x, v_y).
2. A cada iteração os pontos adquirem a velocidade do vizinho mais próximo, com alguma variação estocástica (loucura).

Sabemos que essas espécies se movimentam em conjunto para se proteger de predadores e procurar alimentos, entretanto a simulação descrita acima apenas tenta locomover os pontos com algum determinado padrão. Isso trás alguns questionamentos interessantes (vamos pensar a partir de agora somente no bando de pássaros):

- Como os pássaros procuram comida?
- Se colocarmos comida em algum lugar, algumas horas depois vários pássaros encontram o alimento mesmo sem saber previamente sua localização. Como o primeiro pássaro que o achou influenciou no movimento do restante do bando?

Desse modo, o seguinte princípio foi definido:

> "Os membros individuais do bando podem aproveitar descobertas e experiências anteriores de todos os outros membros ao procurar comida. [1]"

Isso transformou a abordagem desse problema para um problema de otimização, sendo fundamental para o desenvolvimento do PSO como conhecemos hoje. Foi daí que surgiu a ideia do poleiro, que consiste nos pássaros voarem ao redor de um dado ponto que possui alimentos. Portanto, cada um deles possui memória de sua melhor posição já percorrida e da melhor posição do bando já encontrada. Assim, o movimento das partículas depende desses dois parâmetros, nomeados $PBest$ (melhor local) e $GBest$ (melhor global). Cada um dos pontos do espaço possui uma qualidade associada, que é calculada para determinar esses parâmetros. Essa ideia permitiu a eliminação da loucura do algoritmo e apresentou bons resultados.

Até então, todo esse progresso nos estudos do PSO havia ocorrido em ambientes bidimensionais. Porém, foi apenas questão de tempo para ele ser adaptado para uma quantidade arbitrária de dimensões, introduzindo os conceitos de *Busca Multidimensional* e *Aceleração por Distância* ao algoritmo. [1]

Assim, chegamos até a versão atual do da Otimização por Enxame de Partículas, que é abordada com mais detalhes na próxima seção.

4 Definição e descrição do PSO

O PSO é um algoritmo heurístico inspirado no comportamento social de um bando de pássaros. Seu objetivo é buscar a solução ótima num determinado espaço de busca através da troca de informações entre indivíduos de uma população determinando qual trajetória cada membro dessa população deve tomar no espaço de busca.

4.1 Movimento das Partículas

Resumidamente, cada partícula k do enxame possui uma posição x_k no espaço d-dimensional e uma velocidade v_k. Elas variam, a cada geração, da seguinte maneira: [5]

$$v_{k+1} = w \cdot v_k + c_1 \cdot r_1 \cdot (p_{best_k} - x_k) + c_2 \cdot r_2 \cdot (g_{best} - x_k)$$

$$x_{k+1} = x_k + v_k$$

Onde w é o coeficiente inercial, c_1 é o coeficiente individual, c_2 é o coeficiente social, r_1 e r_2 são valores aleatórios entre 0 e 1, p_{best_k} é a melhor posição já encontrada da partícula k e g_{best} é a melhor posição já encontrada em todo o enxame. [5]

Veja que os vetores $(p_{best_k} - x_k)$ e $(g_{best} - x_k)$ funcionam como atratores da partícula para as regiões dos pontos p_{best_k} e g_{best}, respectivamente. Desse modo, o parâmetro c_1 regula a etapa de diversificação, enquanto o parâmetro c_2 regula a etapa de intensificação do algoritmo. Portanto, deve-se escolher bem os valores dessas variáveis para evitar a estagnação em mínimos locais e/ou convergência prematura.

4.2 Pseudocódigo

Segue abaixo pseudocódigo da meta-heurística: [6]

Algorithm 1 Particle Swarm Optimization

```
1:  function PSO()
2:      S ← INITIALIZE_SWARM()
3:
4:      while (stop condition is not met) do
5:          for ∀ particle p ∈ S do
6:              if f(p) < f(best_p) then
7:                  best_p ← p
8:              end if
9:
10:             if f(best_p) < f(best_swarm) then
11:                 best_swarm ← best_p
12:             end if
13:         end for
14:
15:         UPDATE_SWARM_VELOCITIES(S)
16:         UPDATE_SWARM_POSITIONS(S)
17:     end while
18:
19:     return best_swarm
20: end function
```

4.3 Observações

Essa versão do PSO é boa para lidar com problemas de otimização contínua, pois as partículas são codificadas como um conjunto de variáveis reais que representam a sua localização num espaço multidimensional.

Quando aplicado a problemas combinatórios, as equações do movimento frequentemente geram soluções inválidas para o problema abordado. Portanto, o algoritmo deve passar por uma série de adaptações para ser capaz de resolver problemas discretos.

5 Exemplo de aplicação do PSO

O PSO pode ser adaptado em vários problemas. Um deles é o treinamento de IAs para elas praticarem algum jogo. O exemplo que vem a seguir é a aplicação dessa meta-heurística para automação do jogo do Chrome Dino, do Google.

5.1 Chrome Dino

O **Chrome Dino** é um jogo integrado ao navegador Google Chrome. Seu objetivo é, controlando um dinossauro, se esquivar dos obstáculos do terreno e sobreviver o maior tempo possível.

Para automatizá-lo, podemos colocar um classificador de árvore de decisão para tomar as ações do personagem, que de acordo com o ambiente ao seu redor decide se pula ou abaixa. A ideia implementada no classificador consiste em determinar o melhor momento para pular um obstáculo e o melhor momento para abaixar quando se está no ar. Portanto, foram definidos dois parâmetros para a árvore de decisão:

- **Alpha (α)**: É uma constante de proporcionalidade entre a distância mínima que se deve pular e a velocidade do jogo. Desse modo, ela determina quando o dinossauro pula algum obstáculo. Além disso,

quanto mais rápido o jogo, mais longe do obstáculo o dinossauro irá pular, portanto a tendência após o salto é cair no chão a uma distância razoável do próximo obstáculo.

- **Beta (β)**: É uma constante de proporcionalidade entre a distância limite que se deve abaixar e a velocidade do jogo. Desse modo, ela determina quando o dinossauro abaixa após um salto. A ideia deste parâmetro é chegar ao solo o mais rápido possível para ter uma reação melhor ao próximo obstáculo.

A partir destas constantes, são determinadas os valores limite **limDistUp** e **limDistDown**, que são calculados da seguinte maneira:

$$limDistUp = \alpha \cdot gameSpeed$$

$$limDistDown = \beta \cdot gameSpeed$$

Com base nessa intuição, a árvore de decisão ficou da seguinte forma:

Figura 1: Árvore de decisão que determina as ações do dinossauro.

Desse modo, é responsabilidade do PSO descobrir valores bons para α e β a fim de maximizar a pontuação no jogo, que é, neste caso, a função objetivo.

Os valores de w, c_1 e c_2 que melhor se adaptaram ao experimento foram:

$$w = 0.08, \quad c_1 = 0.2, \quad c_2 = 0.05$$

Todas as partículas do enxame foram inicializadas com uma posição de valor aleatório entre 0 e 1 nas componentes α e β. Já suas velocidades foram inicializadas em 0.

A quantidade de iterações determinadas para treinar o algoritmo foi 10.000, com 50 dinossauros em sua população. Em cada geração serão feitas 5 corridas, a fim de se obter um resultado consistente para cada partícula numa determinada iteração, devido a função objetivo ser não determinística. Esta etapa levou cerca de 8 horas para ser concluída.

Os melhores valores encontrados para os parâmetros foram $\alpha = 23.891$ e $\beta = 6.298$. Veja abaixo os resultados obtidos da melhor pontuação do Enxame e da média do enxame em função da iteração.

As melhores pontuações ficaram no range $[2500, 3500]$. Já a pontuação média do enxame no range $[500, 1000]$.

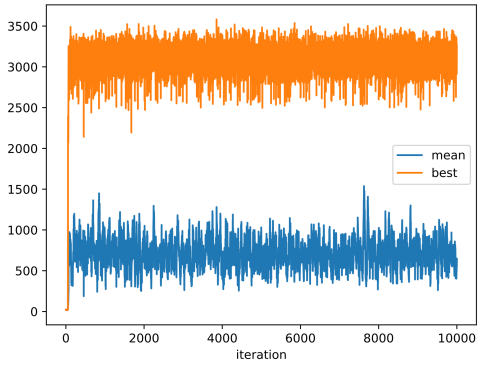

Figura 2: Melhor resultado e média da pontuação em função da iteração do PSO.

Note que houve rápida convergência. Esse fato pode ser indício de uma estagnação num máximo local, entretanto, durante o desenvolvimento do projeto, foram explorados diversos outros espaços de busca e todos tiveram resultados muito inferiores. Portanto, há uma boa chance deste máximo local também ser global.

Clique aqui para conferir o resultado do treinamento do personagem do jogo.

Referências

[1] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948 vol.4, 1995.

[2] Zhong Liu and Lei Huang. A mixed discrete particle swarm optimization for tsp. In *2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)*, volume 2, pages V2–208–V2–211. IEEE, 2010.

[3] Yannis Marinakis, Athanasios Migdalas, and Angelo Sifaleras. A hybrid particle swarm optimization – variable neighborhood search algorithm for constrained shortest path problems. *European journal of operational research*, 261(3):819–834, 2017.

[4] Jian Jiao, Seyed-mohsen Ghoreishi, Zohre Moradi, and Khaled Oslub. Coupled particle swarm optimization method with genetic algorithm for the static–dynamic performance of the magneto-electro-elastic nanosystem. *Engineering with computers*, 38(Suppl 3):2499–2513, 2022.

[5] Diogo Freitas, Luiz Guerreiro Lopes, and Fernando Morgado-Dias. Particle swarm optimisation: A historical review up to the current developments. *Entropy*, 22(3), 2020.

[6] M. Dorigo, M. A. Montes de Oca, and A. Engelbrecht. Particle swarm optimization. *Scholarpedia*, 3(11):1486, 2008. revision #91633.