Solução aproximada do problema do caixeiro viajante utilizando enxame de partículas

Tiago Pinho da Silva*

2016

Introdução

O método de otimização enxame de partículas do inglês *particle swarm optmization* (PSO) foi originalmente proposto por Kennedy e Eberhart em 1995 (WANG L. HUANG; PANG, 2003; SHI Y.C. LIANG; WANG, 2007; SAHIN; DEVASIA, 2007) com intuito de otimizar soluções para problemas difíceis. Neste algoritmo um enxame de partículas é inicializado de forma aleatória (WANG L. HUANG; PANG, 2003). As partículas então se movem no espaço de busca com uma dada velocidade(SAHIN; DEVASIA, 2007). A localização de uma partícula em um determinado momento representa uma potencial solução, cuja aptidão é avaliada por uma função que será otimizada (SHI Y.C. LIANG; WANG, 2007; SAHIN; DEVASIA, 2007). Gerações subsequentes são geradas a partir de gerações de partículas anteriores. As partículas se movem no espaço de busca encontrando novas potenciais soluções e produzindo novas gerações, este processo continua até que um número de gerações seja alcançado ou a melhor partícula na população não possam mais ser melhorada(SAHIN; DEVASIA, 2007).

O comportamento do enxame é dado pelas seguintes equações:

$$v(t+1) = v(t) + \phi_1(x - x_p) + \phi_2(x - x_n) \tag{1}$$

$$x(t+1) = x(t) + v(t+1) \tag{2}$$

Nestas equações, v representa a velocidade da partícula, x é a posição, e ϕ_1 e ϕ_2 representam variáveis aleatórias que introduzem incerteza (SAHIN; DEVASIA, 2007). Os parâmetros ϕ_1 e ϕ_2 podem ser representados como constantes de aceleração $c1$ e $c2$ multiplicadas por uma função *random* (SAHIN; DEVASIA, 2007). O valor x_p representa a posição da partícula com o melhor

*tiagopinhodsr@dc.ufscar.br

valor de aptidão até então. e o valor x_n representa a posição da partícula com melhor valor de aptidão globalmente. A Equação 1 pode ser extendida incluindo uma constante que representa o coeficiente de inercia no intervalo de zero à um. Este coeficiente foi introduzido para balancear as buscas locais e globais (SAHIN; DEVASIA, 2007).

$$v(t+1) = wv(t) + \phi_1(x - x_p) + \phi_2(x - x_n) \tag{3}$$

O problema do caixeiro viajante do inglês *travelling salesman problem* (TSP) é conhecido e comumente utilizado como *benchmark* por desenvolvedores em otimização combinatória (WANG L. HUANG; PANG, 2003). A descrição do TSP envolve um caminho de um vendedor que passa por todas as cidades apenas uma vez percorrendo a menor distância e finalmente chegando à cidade inicial. As cidades são representadas como vértices de um grafo e a distância entre duas cidade é representada pelo custo da aresta que as conecta. Este trabalho, implementa uma solução aproximada utilizando PSO para o TSP descrita por Wang L. Huang e Pang (2003) e Goldbarg, Souza e Goldbarg (2008) em SCILAB sobre o conjunto de dados *eil51* disponível na biblioteca TSPLIB (2016)

PSO Discreto para o TSP

Goldbarg, Souza e Goldbarg (2008) descrevem um PSO discreto para encontrar uma solução aproximada ou ótima para o TSP. Segundo Goldbarg, Souza e Goldbarg (2008) as equações de velocidade Equação 1 e de movimento da partícula Equação 2 do PSO tradicional (WANG L. HUANG; PANG, 2003), não podem ser utilizadas como solução para o TSP, por serem exclusivamente para o domínio de dados contínuos. Sendo assim as equações 1 e 2 passam a ser as seguintes respectivamente:

$$v_p(t+1) = c_1 v_1(x_p) \oplus c_2 v_2(x_p, pbest_p) \oplus c_3 v_3(x_p, gbest_p) \tag{4}$$

$$x_p = x_p \oplus v_p(t+1) \tag{5}$$

Na Equação 4 as funções v_1, v_2, v_3 são meta-heurísticas que respectivamente move a partícula no seu próprio caminho, move a partícula em direção ao seu melhor local, move a partícula em direção ao melhor global. As constantes c_1, c_2, c_3 são limitadoras de cada meta-heurística, sendo a probabilidade delas acontecerem. Após os cáculos de cada função v_i com probabilidade c_i, composições entre elas são feitas. Como meta-herítiscas para v_1, v_2, v_3, foram escolhidos respectivamente os algoritmos de inversão e *path-relinking* pois foram os que obitiveram os melhores resultados até então, além disso aplica-se uma busca local após a função v_1 removendo-se sub-caminhos que se intersectam tornando assim a convergência mais rápida. A seguir o algoritmo do PSO descrito por Goldbarg, Souza e Goldbarg (2008) é apresentado:

Algoritmo 0.1: *PSO-Discreto* (GOLDBARG; SOUZA; GOLDBARG, 2008)

```
início
    /*Define as probabilidades para os movimentos das partículas*/
    pr1 ← a1 /*Probabilidade de seguir seu próprio caminho*/
    pr2 ← a2 /*Probabilidade de seguir seu melhor local*/
    pr3 ← a3 /*Probabilidade de seguir seu melhor global*/
    /*a1 + a2 + a3 = 1 */
    Inicialização aleatório da população de partículas;
    enquanto um critério de parada não for satisfeito faça
        para p = 1 : quantidadeParticulas faça
            value_p ← Fitness(x_p)
            se value(x_p) < value(pbest_p) então
                pbest_p = x_p
            fim
            se value(x_p) < value(gbest_p) então
                gbest_p = x_p
            fim
        fim
        para p = 1 : quantidadeParticulas faça
            velocidade_p = definirVelocidade(pr1, pr2, pr3)
            x_p = movimenta(x_p, velocidade_p)
        fim
    fim
fim
```

No Algoritmo 0.1, as funções $definirVelocidade$ e $movimenta$ estão sendo chamadas em cada passo, entretanto de acordo com Goldbarg, Souza e Goldbarg (2008) as duas funções podem ser substituída pela função apresentada no Algoritmo 0.2

Algoritmo 0.2: *Movimenta-Particula* (GOLDBARG; SOUZA; GOLDBARG, 2008)

Entrada: x_p,$pbest_p$,$gbest_p$
Saída: y_3

```
início
    y1 ← v1(xp)
    y1 ← removeIntersecções(y1)
    y2 ← v2(y1, gbestp)
    y3 ← v3(y2, pbestp)
fim
```

Execução

Para executar o código em SCILAB basta executar o script *Main.sce*, as configurações de cada parâmetro é feita no arquivo *teste.csv*, após o término da execução os resultados são mostrados no arquivo results.csv e são plotadas as curvas de resultados por iteração da melhor execução de cada teste, ou seja a execução que obteve o menor caminho.

Código

Scripts

O código deste trabalho foi dividido em quatro scripts:

- **Main.sce:** responsável pela execução dos testes e cálculos da média dos melhores resultados de cada teste e do desvio padrão;
- **PSO.sce:** responsável pela execução do PSO;
- **UtilsPSO.sce:** possui todas as funções utilitarias para o funcionamento do PSO;
- **UtilsFiles.sce:** possui todas a funções utilitárias para a leitura dos conjuntos de dados.

Funções

UtilsFiles.sce

- **LoadFile:** Carrega os dados do arquivo em variáveis, esta função recebe como parâmetro o nome do conjunto de dados e retorna um grafo representado por uma matriz de adjacência que possui as distancias entre cada cidade, uma matriz de coordenadas de cada cidade em um plano cartesiano, e a quantidade de cidades que o conjunto de dados possui;

- **CalculaDist:** Calcula o grafo baseado na matriz de coordenadas do conjunto de dados, recebe como parâmetro de entrada o nome do arquivo e retorna o grafo e a matriz de coordenadas.

PSO.sce

- **PSO:** responsável pela execução do PSO, recebe como entrada o nome do cojunto de dados a quantidade de partículas no enxame, a quantidade de iterações e as constantes de probabilidade c_1, c_2, c_3.

UtilsPSO.sce

- **Initialize:** inicializa a população inicial, os melhores locais e o melhor global;
- **MoveParticle:** esta função recebe como parâmetros de entrada as posições de partícula $x_p, pbest_p, gbest_p$, e as probabilidades pr_1, pr_2, pr_3, aplica o Algoritmo 0.2 e retorna a particula movimentada;
- **Invertion:** Faz sucessivas inversões em uma dada partícula x_p com uma probabilidade pr_1 dessas inversões serem feitas, além de sempre checar a aptidão das novas partículas geradas a fim de sempre manter a que possui melhor aptidão;
- **PathRelinking:** Faz sucessivas trocas em uma partícula x_p, afim de movimentá-la em direção à uma particula x_best. Entretanto as trocas só são realizadas com base em uma probabilidade pr_2 e se a aptidão do novo x_p melhorar;
- **Swap:** Faz sucessivas trocas em uma partícula x_p baseadas em uma sequência de trocas dadas como parâmetro;
- **Reverse:** Realiza um inversão em um sub-caminho de índices $[i, j]$ de uma partícula x_p;
- **OnSegment:** Verifica se um dado ponto r se encontra em uma reta limitada pelos pontos p e q;
- **PointsOrientation:** Verifica a orientação de três pontos p, q, r se são colineares, ou estão em uma orientação horária ou anti-horária;
- **linesIntersect:** Verifica se duas linhas cada uma limitada respectivamente pelos pares de pontos $p1, q1$, e $p2, q2$ se intersectam.
- **removeCrossovers:** Dado uma partícula, esta função verifica se existem sub-caminhos que se intersectam e então os remove fazendo uma inversão entre o segundo desse sub-caminho e o penúltimo.

Resultados

Para os conjunto de testes, foram feitas alterações nas constantes de probabilidade e quantidade de partículas no enxame, afim de verificar o comportamento do algoritmo entre diferentes valores para cada constante e quantidade de partículas, foram feitos 10 testes cada um com 200 iterações no conjunto de dados eil51, que possui 51 cidades.

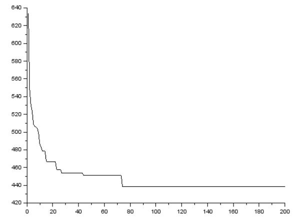
Tabela 1: Testes em constantes de probabilidade

Teste	Partículas	Iterações	pr_1	pr_2	pr_3	Stdev	Mean	Best	Erro%
1	20	200	0,2	0,4	0,4	5,09776	445,9152	438,419	4,674925
2	20	200	0,1	0,1	0,8	6,269657	445,3595	433,6888	4,544491
3	20	200	0,1	0,8	0,1	3,370082	445,5727	439,4765	4,594528
4	20	200	0,8	0,1	0,1	4,244361	445,7349	436,4129	4,632615

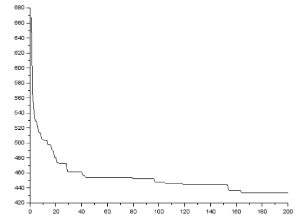
Tabela 2: Testes em quantiade de partículas

Teste	Partículas	Iterações	Pr1	Pr2	Pr3	Stdev	Mean	Best	Erro%
1	20	200	0,2	0,4	0,4	5,189795	443,3277	436,6862	4,06754
2	30	200	0,2	0,4	0,4	4,197978	444,5604	439,2558	4,356906
3	40	200	0,2	0,4	0,4	2,901483	442,8121	438,6253	3,946493
4	50	200	0,2	0,4	0,4	3,675375	443,4447	436,8808	4,095

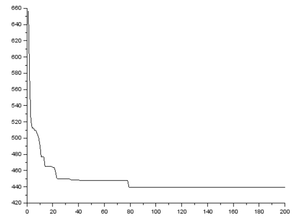
Através da Tabela 1 e Tabela 2 verificamos que apesar de muito pequenas as diferenças no erro e nas médias, houve uma certa diferença no desvio padrão para os testes que possuíam um valor de probabilidade maior para o pr_2 e pr_1 e quantidade de partículas maior. Entretanto não podemos dizer com certeza as causas desses comportamentos, visto que a quantidade de testes feitas não foi o suficiente para extrair algum resultado.



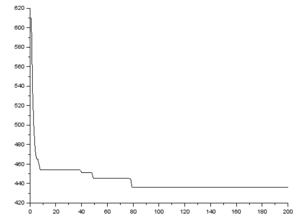
(a) Teste 1



(b) Teste 2

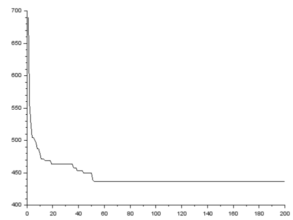


(c) Teste 3

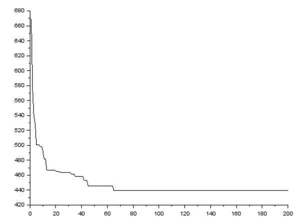


(d) Teste 4

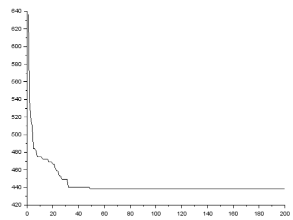
Figura 1: Melhores resultados para cada teste do conjunto de teste que modifica as varáveis de probabilidade



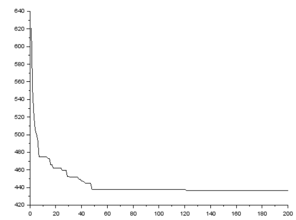
(a) Teste 1



(b) Teste 2



(c) Teste 3



(d) Teste 4

Figura 2: Melhores resultados para cada teste do conjunto de teste que modifica a quantidade de partículas

Consderações Finais

O algoritmo implementado neste trabalho apresentou bons resultados e uma convergência próxima de 100 iterações para os testes em constantes Figura 1 e próxima de 50 iterções Figura 2, e conseguiu se aproximar aos resultados propostos neste trabalho. Como trabalho futuro poderá ser feita uma análise mais aprofundada do comportamento desde algoritmo através da mudança de parâmetros e das meta-heurísticas envolvidas no cálculo da velocidade.

Referências

GOLDBARG, E. F.; SOUZA, G. R. de; GOLDBARG, M. C. *Particle swarm optimization algorithm for the traveling salesman problem*. [S.l.]: INTECH Open Access Publisher, 2008. Citado 3 vezes nas páginas 2, 3 e 4.

SAHIN, F.; DEVASIA, A. *Distributed Particle Swarm Optimization for Structural Bayesian Network Learning*. [S.l.]: 1-Tech Education and Publishing - Vienna, 2007. Citado 2 vezes nas páginas 1 e 2.

SHI Y.C. LIANG, H. L.-C. L. X.; WANG, Q. Particle swarm optimization-based algorithms for tsp and generalized tsp. *Information Processing Letters*, v. 103, p. 169–176, 2007. Citado na página 1.

TSPLIB. 2016. Wiki do abnTeX2. Disponível em: <http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/>. Acesso em: 05.10.2016. Citado na página 2.

WANG L. HUANG, C.-G. Z. K.-P.; PANG, W. Particle swarm optimization for traveling salesman problem. *International Conference on Machine Learning and Cybernetics*, p. 1583–1585, 2003. Citado 2 vezes nas páginas 1 e 2.