META-HEURÍSTICA PARTICLE SWARM OPTIMIZATION APLICADA AO PROBLEMA DO CAIXEIRO VIAJANTE ASSIMÉTRICO

CAUÊ D. M. TEIXEIRA[1], BRUNO R. GAMINO[2]

[1] Graduando em Bacharel em Engenharia de Controle e Automação, Bolsista PIBIFSP, IFSP, Campus Suzano, caue.daniel@aluno.ifsp.edu.br.

[2] Doutor em Engenharia Elétrica, Professor, IFSP, Campus Birigui, bruno.rafael@ifsp.edu.br.

Área de conhecimento (Tabela CNPq): 3.08.02.02-4 Programação Linear, Não-Linear, Mista e Dinâmica.

RESUMO: O Problema do Caixeiro Viajante Assimétrico (PCVA) é um desafio clássico de otimização combinatória em que um vendedor ambulante percorre um conjunto de cidades, visitando cada cidade exatamente uma vez, e retorna à cidade de origem, minimizando a distância total percorrida, porém a distância entre duas cidades pode variar dependendo do sentido do deslocamento. O *Particle Swarm Optimization*, uma técnica inspirada no comportamento social de organismos vivos, foi utilizado para resolver o problema. Sendo uma meta-heurística, exige um balanço entre diversidade e intensidade do método para alcançar boas soluções em tempo razoável, evitando assim, ótimos locais. Para isso, o peso de inércia é linearmente reduzido ao longo de cada iteração, explorando amplamente o espaço de busca no início do algoritmo e intensificando a busca na parte final do algoritmo. Visto que o algoritmo encontrou, pelo menos uma vez, a solução ótima global para todas as instâncias testadas, o objetivo proposto foi atingido.

PALAVRAS-CHAVE: *particle swarm optimization*; problema do caixeiro viajante assimétrico; otimização combinatória; metaheurística.

META-HEURISTIC PARTICLE SWARM OPTIMIZATION APPLIED TO THE ASYMMETRIC TRAVELING SALESMAN PROBLEM

ABSTRACT: The Asymmetric Traveling Salesman Problem (ATSP) is a classic combinatorial optimization challenge where a traveling salesman visits a set of cities exactly once and returns to the origin city, minimizing the total distance traveled. However, the distance between two cities can vary depending on the orientation of travel. Particle Swarm Optimization, a technique inspired by the social behavior of living organisms, was used to solve the problem. As a meta-heuristic, it requires a balance between diversity and intensity to achieve good solutions in a reasonable time, thus avoiding local optimal. For this purpose, the inertia weight, which is linearly reduced over each iteration, widely exploring the search space at the beginning of the algorithm and intensifying the search in the final part of the algorithm. Since the algorithm found the global optimal solution at least once for all tested instances, the proposed objective was achieved.

KEYWORDS: particle swarm optimization, asymmetric traveling salesman problem, combinatorial optimization; metaheuristic.

INTRODUÇÃO

O Problema do Caixeiro Viajante Assimétrico (ATSP, do inglês *Asymmetric Traveling Salesman Problem*) é um dos problemas de otimização mais desafiadores, sendo considerado um problema NP-Difícil (Garey; Johnson, 1979). Nele, um viajante deve percorrer um conjunto de cidades, visitando cada cidade exatamente uma vez, e retornar à cidade de origem, minimizando a distância total percorrida, porém a distância entre duas cidades pode variar dependendo do sentido do deslocamento, refletindo situações do mundo real, como restrições de tráfego (Reinelt, 1994).

O *Particle Swarm Optimization* (PSO) é uma metaheurística inspirada no comportamento social de organismos vivos, como bandos de pássaros e cardumes de peixes. Neste algoritmo, uma população de partículas se move em um espaço de busca multidimensional, cada uma representando uma possível solução para o problema de otimização (Kennedy; Eberhart, 1995). Para lidar com problemas de otimização discreta, como o ATSP, é necessário adaptar o PSO para sua forma discreta, fazendo-o operar em um espaço de busca discreto, o que significa que as soluções são representadas por um conjunto finito de valores discretos ao invés de valores contínuos (Yang; Wang; Nie, 2017).

As partículas são inicializadas aleatoriamente no espaço de busca, cada uma com uma posição e velocidade inicial. Em cada iteração do algoritmo, as partículas atualizam suas variáveis com base em três fatores principais: a sua experiência pessoal, a experiência do grupo e o peso de inércia. Com o passar das iterações, o peso de inércia é linearmente reduzido, o que reduz a velocidade da partícula, permitindo o aumento da intensidade da busca nas últimas iterações (Eberhart; Shi, 2001).

Este trabalho tem como objetivo a utilização do *Particle Swarm Optimization* discreto como forma de solucionar o Problema do Caixeiro Viajante Assimétrico.

MATERIAL E MÉTODOS

Foram escolhidas cinco instâncias para trabalho, extraídas de TSPLIB (2018), sendo elas: "br17" (39), “kro124p“ (36230), “rbg323” (1326), “rbg403” (2465) e “rbg443” (2720). Com isso, é possível determinar a eficácia do algoritmo tanto em instâncias pequenas e médias, de 10 a 350 cidades, quanto em instâncias grandes, acima de 350 cidades.

Mesmo sendo um problema com distâncias assimétricas de ida e volta de cada cidade, a abordagem utilizada foi a de distâncias euclidianas, portanto a matriz de entradas tem suas linhas e colunas com as cidades de seus índices e os itens representam a distância entre a cidade da linha e a cidade da coluna. A geração da população inicial é feita a partir da importação da matriz de distâncias de cada instância, que gera uma matriz de solução aleatória, onde cada linha representa uma partícula, e calcula sua função objetivo, que é obtida a partir da soma das distâncias entre as cidades na ordem definida pelo vetor solução gerado. O fluxograma do algoritmo *Particle Swarm Optimization* discreto é apresentado na Figura 1.

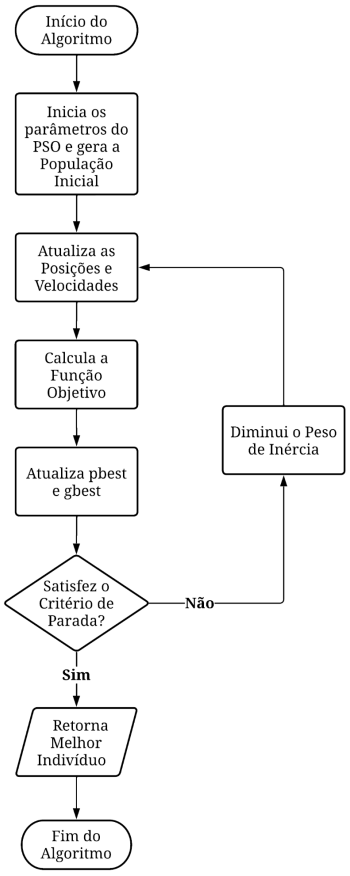

FIGURA 1. Fluxograma do *Particle Swarm Optimization.*

Após a determinação dos parâmetros, o código inicia com a geração da população inicial. Feito isso, as posições e velocidades de cada partícula são atualizadas e, na sequência, a melhor posição pessoal e a melhor posição global encontradas também são atualizadas. Neste momento, a melhor posição global é comparada com a solução ótima global e, caso seja igual, a solução atual se torna a solução final. Em caso negativo, o peso de inércia é linearmente diminuído e novas velocidades e posições são geradas a partir das anteriores. Este processo é repetido em *looping* até que a solução ótima global seja encontrada ou até que o número de iterações máximo seja atingido.

O fator de inércia é introduzido para controlar a influência da velocidade anterior de uma partícula na sua nova velocidade, porém, um fator de inércia fixo pode apresentar problemas significativos, independentemente de seu valor. Um valor alto de inércia promove a exploração, permitindo que as partículas se movam mais livremente pelo espaço de busca, mas pode dificultar a convergência para soluções ótimas, pois as partículas não se concentram nas melhores soluções encontradas. Por outro lado, um valor baixo de inércia favorece a intensificação, permitindo que as partículas se concentrem em áreas promissoras, mas aumenta o risco de ficarem presas em mínimos locais, limitando a capacidade de encontrar soluções globais (Chen et al., 2006).

Ajustar dinamicamente o fator de inércia ao longo do tempo pode melhorar o desempenho do algoritmo. Começar com um valor alto incentiva a exploração do espaço de busca, reduzindo a probabilidade de as partículas ficarem presas em mínimos locais e permitindo uma busca mais eficiente por soluções globais. À medida que a execução progride, diminuir o valor da inércia ajuda as partículas a se estabilizarem em torno das melhores soluções encontradas, promovendo uma convergência mais rápida e soluções de melhor qualidade, adaptando-se às necessidades do algoritmo em diferentes estágios da busca. A fórmula de atualização do peso de inércia adaptativo é apresentada na Equação (1) (Bansal et al., 2011):

$$w(t) = w_{\max} - [(w_{\max} - w_{\min})/T].\ t, \tag{1}$$

Onde $w(t)$ é o valor da inércia na iteração t, $w_{\max}$ e $w_{\min}$ são os valores máximo e mínimo da inércia, e T é o número total de iterações.

A atualização da velocidade e posição de cada partícula é feita seguindo uma equação que considera a velocidade atual, a memória da melhor posição pessoal e a melhor posição global encontrada por qualquer partícula na população. A fórmula típica de atualização da velocidade em uma dimensão é demonstrada na Equação (2) e na Equação (3), respectivamente (Eberhart; Shi, 2001).

$$v_i(t+1) = w.\ v_i(t) + c_1.\ rand_1.\ \left(pbest_i - x_i(t)\right) + \ c_2.rand_2.\left(gbest_i - x_i(t)\right), \tag{2}$$

$$x_i(t+1) = x_i(t) \ + \ v_i(t+1). \tag{3}$$

A Equação (2) define como é realizado o controle da direção e da magnitude do movimento de cada partícula no espaço de busca. A velocidade da partícula i na iteração t é representada por $v_i(t)$. W é o peso de inércia, que determina a influência da velocidade anterior da partícula em seu movimento atual. Os coeficientes "c_1 e c_2" são constantes de aprendizado, também conhecidas como coeficientes de aceleração, e controlam a influência das melhores posições pessoais e global na atualização da velocidade. Para melhores resultados, a soma de ambas as constantes deve ser maior ou igual a 4, no algoritmo, c_1 foi definido como 2,1 e c_2 como 1,9, priorizando mais a intensificação do que a diversificação no algoritmo. Os números $rand_1$ e $rand_2$ são números aleatórios gerados entre 0 e 1, que introduzem aleatoriedade no movimento das partículas, promovendo uma exploração mais ampla do espaço de busca (Kennedy; Eberhart, 2019). O termo $pbest$ representa a melhor posição pessoal alcançada pela partícula i, enquanto $gbest$ representa a melhor posição global encontrada por qualquer partícula na população. Por fim, $x_i(t)$ denota a posição da partícula i na iteração t (Xiao; Yang; Cheng, 2006).

RESULTADOS E DISCUSSÃO

Para os testes preliminares, foram realizados 500 testes com 3.000 iterações cada, utilizando a fórmula típica de atualização de posição do PSO discreto. Com o método de conformidade sendo o de

arredondamento para o próximo número inteiro e o fator de inércia fixo, além de percorrer o vetor solução apenas em um sentido. A Tabela 1 contém os resultados obtidos para todas as instâncias escolhidas.

TABELA 1. Resultados dos testes preliminares para todas as instâncias escolhidas.

Instâncias	Média	Mínimo	Máximo
Br17 (39)	54	39	279
Kro124p (36230)	48745,16	36230	72648
Rbg323 (1326)	2158,66	1326	4607
Rbg403 (2465)	3481,01	2548	5728
Rbg443 (2720)	4233,66	3046	7466

Observa-se que o algoritmo foi capaz de alcançar a solução ótima global em três das cinco instâncias propostas, além de ter obtido resultados com valores de 4% a 12% maiores do que o ótimo global nas duas restantes. Sendo assim, fez-se necessário a implementação de técnicas adicionais para obtenção de melhores resultados, como o método de conformidade sendo o de arredondamento para o número inteiro mais próximo, o fator de inércia adaptativo e o deslocamento do vetor solução pelos dois sentidos. Foram realizados 1.000 testes com 3.000 iterações cada, com estas técnicas, e os resultados são apresentados na Tabela 2.

TABELA 2. Resultados dos testes finais para todas as instâncias escolhidas.

Instâncias	Média	Mínimo	Máximo	Taxa Conv.
Br17 (39)	43,71	39	137	86%
Kro124p (36230)	48643,79	36230	65374	52%
Rbg323 (1326)	2671,26	1326	3281	27%
Rbg403 (2465)	3963,54	2465	5248	12%
Rbg443 (2720)	4276,12	2720	8397	2,7%

Como pode ser visto na Tabela 2, o algoritmo atingiu o valor ótimo global para todas as instâncias. A média das duas menores instâncias foi menor, enquanto a das três maiores foi maior, quando se compara com os testes preliminares. Essa mudança deu-se pelo fato do peso de inércia, agora adaptativo, ser maior nas primeiras iterações, abrangendo um espaço de busca maior e, portanto, aumentando a diversidade do algoritmo. As taxas de convergência foram razoáveis e as médias ficaram altas, de 12% a 101% acima do ótimo global. No entanto, desta vez, o algoritmo conseguiu atingir o ótimo global para as maiores instâncias, pelo mesmo motivo da mudança anterior, um peso de inércia menor nas últimas iterações do algoritmo faz com que a intensidade também aumente, sendo capaz de alcançar melhores soluções. Outro fator impactante foi o deslocamento em ambos os sentidos do vetor, que fez com que fossem analisados o dobro de soluções a cada iteração, trazendo assim uma melhora nos resultados das maiores instâncias.

Além disso, para questão de visualização dos resultados, foram gerados gráficos para as instâncias “br17”, “rbg323”, “rbg403” e “rbg443”, demonstrando a evolução da melhor solução dentre as partículas na iteração atual (denominado “pbest” apenas para fins práticos) e da melhor solução global (gbest) ao longo da execução do código, que podem ser vistos na Figura 2.

Na Figura 2, o eixo *x* representa as iterações, com os valores variando de 0 a 3000 com pontos a cada 100 iterações realizadas. O eixo *y* indica o valor da função objetivo. Nota-se que a linha azul do gráfico que representa gbest nunca aumenta seu valor, dado que ela representa, sempre, a melhor solução encontrada, diferentemente da linha laranja, que representa a melhor solução dentre as partículas na iteração atual, apresentando flutuações ao longo das iterações, refletindo a exploração do espaço de soluções pelo algoritmo.

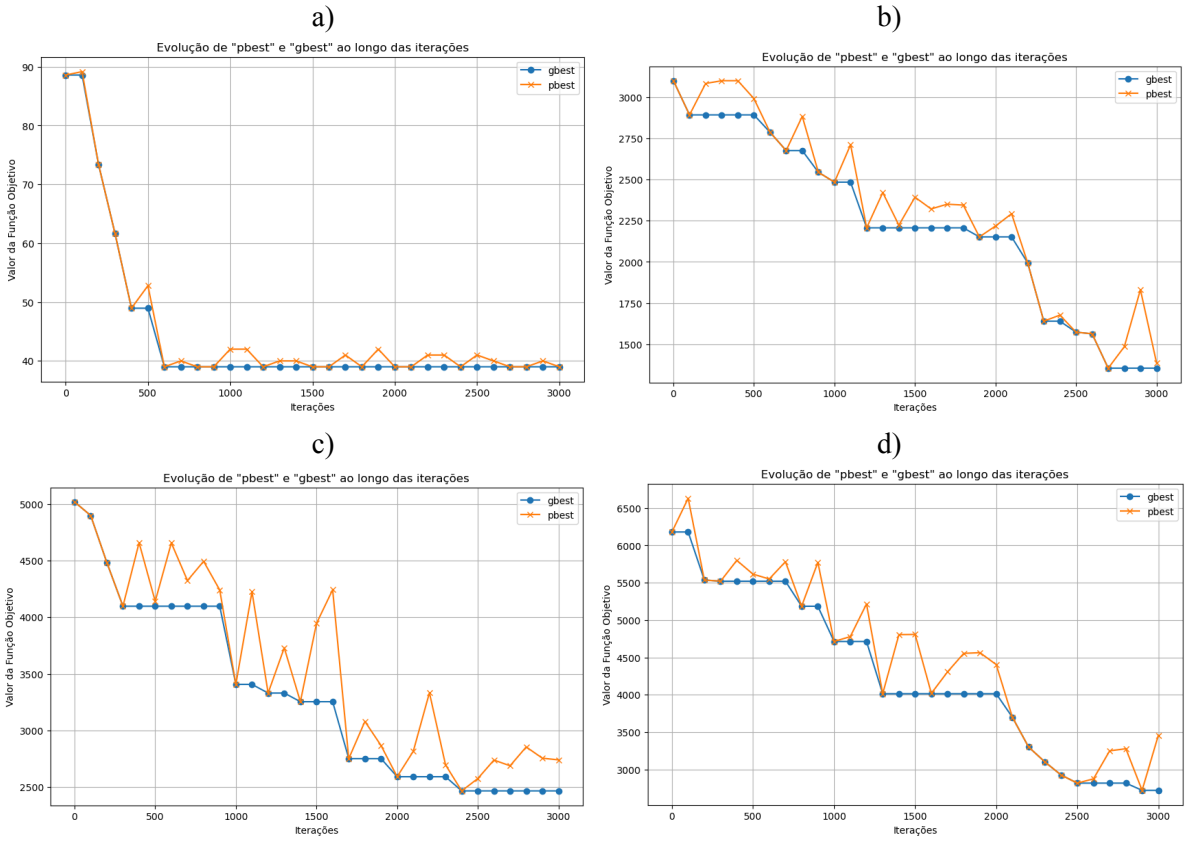

FIGURA 2. Evolução da melhor solução atual e da melhor solução global ao longo da execução a cada 100 iterações, para as instâncias: a) “br17”, b) “rbg323”, c) “rbg403” e d) “rbg443”.

CONCLUSÕES

Após a implementação do *Particle Swarm Optimization*, em sua forma discreta, para solucionar o Problema do Caixeiro Viajante Assimétrico, considerando cinco diferentes instâncias, foram obtidas soluções de boa qualidade em tempos computacionais razoáveis, atendendo ao objetivo proposto. O método implementado foi capaz de encontrar o valor ótimo global em cada uma das instâncias testadas, mesmo não tendo médias próximas do ótimo global. O peso de inércia adaptativo trouxe consigo um maior balanço de diversidade e intensidade, fazendo assim, o algoritmo percorrer um maior espaço de busca no início e obter melhores resultados ao fim, evitando ficar preso em ótimos locais, aumentando a eficácia do algoritmo. Pode-se focar, em futuros projetos, na utilização de outros métodos para a resolução do TSP ou outros problemas combinacionais.

CONTRIBUIÇÕES DOS AUTORES

C.D.M.T contribuiu com a coleta, geração, curadoria e análise dos dados e com a elaboração, codificação e correção do algoritmo. B.R.G contribuiu com os objetivos e a metodologia do trabalho, com a elaboração dos métodos do algoritmo e com a análise dos resultados.

Todos os autores contribuíram com a redação e revisão do trabalho e aprovaram a versão submetida.

AGRADECIMENTOS

Agradecemos ao Instituto Federal de Educação, Ciência e Tecnologia de São Paulo - Campus Suzano e ao programa PIBIFSP pelo apoio financeiro, tornando possível a realização deste trabalho.

REFERÊNCIAS

BANSAL, J. C..; SINGH, P. K.; SARASWAT, M; VERMAM, A.; JADON, S. S.; ABRAHAM, A. Inertia weight strategies in particle swarm. In: **WORLD CONGRESS ON NATURE AND BIOLOGICALLY INSPIRED COMPUTING**, 3., 2011, Salamanca. *Proceedings*... Salamanca: IEEE, 2011. p. 633-640.

CHEN, G.; MIN, Z; JIA, J; XINBO, H. Natural exponential inertia weight strategy in particle swarm optimization. In: **WORLD CONGRESS ON INTELLIGENT CONTROL AND AUTOMATION**, 6., 2006, Dalian. *Proceedings*... Dalian: IEEE, 2006. p. 3672-3675.

GAREY, M. R.; JOHNSON, D. S. **Computers and Intractability:** A Guide to the Theory of NP-Completeness. New York: W. H. Freeman & Co Ltd, 1979. 340 p.

EBERHART, R. C.; SHI, Y. Particle swarm optimization: developments, applications and resources. In: **CONGRESS ON EVOLUTIONARY COMPUTATION**, 1., 2001, Piscataway. *Proceedings*... Seoul: IEEE, 2001. p. 81-86.

KENNEDY, J.; EBERHART, R. C. Particle swarm optimization. In: **INTERNATIONAL CONFERENCE ON NEURAL NETWORKS**, 3., 1995, Perth. *Proceedings*... Perth: IEEE, 1995. p. 1942-1948.

REINELT, G. **The Traveling Salesman:** Computational Solutions for TSP Applications. Heidelberg: Springer-Verlag, 1994. 223 p.

TSPLIB. **University of Heidelberg**, 2018. Disponível em: <http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp>. Acesso em: 10 mar. 2024.

XIAO, J.; YANG, J.; CHENG, H. Particle swarm optimization algorithm for TSP. In: **INTERNATIONAL CONFERENCE ON COMPUTATIONAL SCIENCE AND ITS APPLICATIONS**, 2., 2006, Berlin. *Proceedings*... Berlin: ICCSA, 2006. p. 597-606.

YANG, S.; WANG, Y.; NIE, X. An effective discrete particle swarm optimization algorithm for the asymmetric traveling salesman problem. **Journal of Computational and Theoretical Nanoscience**. v.14, p. 3177-3184, 2017.