OTIMIZAÇÃO POR NUVEM DE PARTÍCULAS: DIFERENÇA ENTRE APLICAÇÕES A PROBLEMAS CONTÍNUOS E DISCRETOS

Marilyn Cristine Serafim de Oliveira[1], Thales Lima Silva[1], Dario José Aloise[1]
[1]Universidade Federal do Rio Grande do Norte
{dario,marilyn@dimap.ufrn.br}, {thales@engcomp.ufrn.br}

RESUMO

Otimização por Nuvem de Partículas (ONP) é uma metaheurística evolucionária que surgiu da intenção de se simular o comportamento de um conjunto de pássaros em vôo com seu movimento localmente aleatório, mas globalmente determinado. Esta técnica tem sido muito utilizada na resolução de problemas contínuos não-lineares e pouco explorada em problemas discretos. Este artigo tem como objetivo apresentar o funcionamento desta metaheurística e as adaptações necessárias para aplicação em problemas de otimização discreta. Além disso, são propostas algumas alterações feitas a fim de melhorar os resultados do algoritmo padrão. Os testes foram realizados em instâncias da TSPLIB a fim de demonstrar a eficiência do método.

Palavras-chave: Nuvem de partículas, Otimização Global, Otimização Combinatória.

ABSTRACT

Particle Swarm Optimization (PSO) is an evolutionary metaheuristic that emerged from the intention of simulating the behavior of a group of birds in flight with its locally random movement, while globally determined. This technique has been used widely in the resolution of non-linear continuous problems, but little explored for discrete problems. The objective of this paper is to present the functionality of this metaheuristic and the necessary adaptations for the application to discrete optimization problems. In addition, some modifications are proposed to improve the results of the standard algorithm. The tests were accomplished in instances of TSPLIB in order to demonstrate the efficiency of the method.

Keywords: Particle Swarm Optimization, Global Optimization, Combinatorial Optimization.

1. Introdução

Os métodos de otimização e busca estocástica baseados nos princípios e modelos da evolução biológica natural têm recebido crescente interesse nas últimas décadas, devido principalmente a sua versatilidade para a resolução de problemas complexos, nas áreas de otimização e aprendizado de máquina. O desenvolvimento de modelos computacionais, inspirados nos mecanismos evolutivos, caracteriza-se pela configuração de algoritmos de otimização robustos e sistemas adaptativos. Os Algoritmos Evolucionários (*AEs*) [1,2,3], metodologias da área computação evolucionária ou evolutiva (*CE*), não são algoritmos computacionais em seu significado usual, mas formam uma classe de métodos regidos por princípios similares. Estes princípios oriundos do "mundo biológico" são baseados na teoria da evolução Darwiniana [4]. Os *AEs* tentam abstrair e imitar alguns dos mecanismos evolutivos à resolução de problemas que requerem adaptação, busca e otimização.

Otimização por Nuvem de Partículas (ONP) [6] é uma técnica de computação evolucionária desenvolvida por James Kennedy, um psicólogo social, e por Russell Eberhart, um engenheiro elétrico, em 1995, inspirada na simulação de um sistema social simplificado. A intenção original era simular graficamente o comportamento de um bando de pássaros em vôo com seu movimento localmente aleatório, mas globalmente determinado.

Computacionalmente, os Algoritmos de *Nuvem* ou *Enxame de Partículas* são uma abstração desse processo natural, onde a procura pela posição mais apta é a busca de uma solução 'ótima' para um problema, sendo o conjunto de possíveis posições das partículas o espaço de busca do problema, e cada posição ocupada por uma partícula uma possível solução para o problema. O comportamento de cada partícula é baseado na sua experiência anterior e na experiência daqueles outros partículas com os quais ele se relaciona. Similarmente aos algoritmos genéticos, o conjunto das partículas tende a preservar aquelas posições que determinam uma maior aptidão e a descartar as posições de menor aptidão.

ONP como em um Algoritmo Genético [5] (AG) é iniciado com uma população de soluções randômicas ou pré-estabelecidas. Entretanto, é diferente dos AGs no fato de que em cada solução potencial é também designada uma velocidade randômica e, as soluções potenciais, chamadas partículas, voam através do espaço de busca do problema. Cada partícula mantém o rastro de suas coordenadas no espaço de busca, que são associadas com a melhor solução (fitness) que ela tenha alcançado. O valor do fitness é também armazenado. Esse valor é chamado de *pbest*. Outro "melhor" valor que é rastreado pela *otimização por nuvem de partículas* é o melhor valor de todos os valores obtidos por qualquer partícula da população. Essa posição (solução) é chamada *gbest*. O princípio dessa metaheurística consiste de, em cada iteração e mudança de velocidade, as partículas voem em direção de suas posições de pbest e gbest [7].

2. Otimização por Nuvem de Partículas – Fundamentos Básicos

Diferentemente de outras técnicas de Computação Evolutiva com evoluções incentivadas (Algoritmos Genéticos), ONP não usa operadores genéticos. Ao invés disso, cada partícula (individualmente) ajusta seu vôo de acordo com sua própria experiência de vôo e na de seus companheiros. A partir disso, faz uso de um grupo (população) de partículas que são inseridas em um espaço de solução para procurar um ótimo local, fundamentadas em alguns procedimentos determinísticos. As partículas se comunicam entre si informando os valores da função objetivo em suas respectivas posições locais. Cada movimento de otimização da partícula é baseada em três parâmetros: Fator de sociabilidade, fator de individualidade e velocidade máxima. O algoritmo combina estes parâmetros com um número gerado aleatoriamente para determinar o próximo local da partícula. Ou seja:

- o *fator de sociabilidade* determina a atração das partículas para a melhor posição descoberta por qualquer elemento do enxame (nuvem);
- o *fator de individualidade* determina a atração da partícula com sua melhor posição já descoberta;
- a *velocidade máxima* delimita o movimento, uma vez que esse é direcional e determinado.

Além destes três fatores, têm-se ainda o número de partículas em um enxame, o número de enxames no espaço solução e os critérios de terminação. Cada partícula é tratada como um ponto em um espaço D-dimensional. A i-ésima partícula é representada como $X_i = (X_{i1}, X_{i2},\ldots, X_{iD})$. A melhor posição prévia (a posição que dá o melhor valor de aptidão) da i-ésima partícula é registrada e representada como $P_i = (P_{i1}, P_{i2},\ldots, P_{iD})$. O índice da melhor partícula entre todas as partículas na população é representado pelo símbolo "g". A taxa da mudança de posição (velocidade) para partícula i é representada como $Vi = (V_{i1}, V_{i2},\ldots, V_{iD})$. As partículas são manipuladas de acordo com as seguintes equações:

$$V_{id} = W* V_{id} + c_1 * \text{rand}() * (P_{id} - X_{id}) + c_2 * \text{Rand}() * (P_{gd} - X_{id}) \quad (1a)$$

$$X_{id} = X_{id} + V_{id} \quad (1b)$$

Onde: c_1 e c_2 - são duas constantes positivas que correspondem as componentes cognitivas e sociais;
rand () e Rand () - são duas funções aleatória no intervalo [0,1] e,
W - é o peso de inércia.

A Equação (1a) é usada para calcular a nova velocidade da partícula de acordo com sua velocidade anterior e as distâncias entre sua posição atual, sua melhor posição e a melhor posição do grupo. Então a partícula voa para uma nova posição de acordo com equação (1b). O desempenho de cada partícula está medido de acordo com uma função de aptidão pré-definida que é relacionada ao problema a ser resolvido. O peso de inércia W é empregado para controlar o impacto da velocidade anterior na velocidade atual, assim influenciando as habilidades de exploração global e local das partículas. Um peso de inércia maior facilita exploração global (procurando novas áreas), enquanto um peso de inércia menor tende a facilitar exploração local para refinar a área de procura atual. A seleção satisfatória do peso de inércia W pode prover um equilíbrio entre habilidades de exploração global e local, e assim pode requerer menos repetições, em média, para encontrar o valor ótimo.
Cada partícula mantém rastro de suas coordenadas, no espaço do problema que é associado com a melhor solução, sendo na verdade, o quanto a partícula deslocou-se. Este valor é chamado *pbest*. Outro valor que é determinado pelas partículas é o melhor valor obtido por qualquer partícula vizinha. Este local é chamado *lbest*. Quando uma partícula levar toda a população, ou seja , seus vizinhos, o melhor valor é chamado *gbest*.
o número de enxames em um espaço é claramente conhecido como um fator na probabilidade de achar o ótimo, pois quanto maior o número de partículas em um determinado espaço mais alta será a probabilidade de achar o ótimo. Porém, reciprocamente, um número maior de partículas resultará no aumento de pontos individuais que serão testados, aumentando assim o tempo de computação.
Um pseudo-código para a representação do algoritmo ONP pode ser representado da seguinte forma:

1 Inicialize parâmetros como:
2 Número de Partículas e Dimensões
3 Velocidade Máxima
4 Número Máximo de Iterações
5 O Peso de Inércia
6 Fitness Inicial
7 Fitness Alvo

```
8  Inicialize_partículas()
9        Para cada partícula
10             Para cada dimensão
11                  Seta o vetor velocidades
12             Fim para cada dimensão
13             Fitness_Partícula = Fitness Inicial
14       Fim Para cada partícula
15 Fim do Inicializa_Partículas
16 Executa_ONP()
17       Enquanto Num_Iterações < Num_Max_Iterações Faça
18             Para cada Partícula
19                  Testa_Fitness() //caso seja melhor local. Então
20                                  //atualiza as dimensões
21                  //caso seja igual a Fitness_ótimo Sai do Laço e Termina a Busca
22                  Para cada Dimensão
23                        Atualiza os Vetores Velocidades e Posição
24                  Fim para cada Dimensão
25             Fim para cada Partícula
26       Fim Enquanto
27 Fim Executa_ONP
```

No algoritmo, o enxame de partículas é lançado inicialmente dentro do espaço de busca, tendo cada partícula as seguintes características:

- Uma posição e uma velocidade;
- Conhecimento de sua posição e o valor da função objetivo para esta posição;
- Conhecimento sobre seus vizinhos (vizinhança): a melhor posição encontrada e o valor da sua função objetivo;
- Armazenamento de sua melhor posição encontrada

Em cada espaço de tempo, o comportamento de uma partícula é determinada dentre três possíveis escolhas:

- Seguir seu próprio caminho;
- Seguir para sua melhor posição encontrada;
- Seguir para a melhor posição encontrada por algum de seus vizinhos.

A condição de parada é ultrapassar o limite de número de iterações pré-definido ou quando não houver mais melhorias (estagnação).

3. Diferenças entre ONP Discreto e ONP Contínuo

Todas as características apresentadas até aqui se referem ao ONP Clássico [6], entretanto um outro modelo recente pode ser encontrado na literatura [8]: ONP Discreto.
No ONP discreto, tanto o algoritmo quanto as equações permanecem sem alteração. Os pontos a serem modificados são os operadores utilizados no espaço de busca e nas soluções encontradas, já que não estaremos mais em um espaço de busca contínuo.
Em relação à posição da partícula, no ONP discreto consideramos as arestas/nós como sendo cada dimensão da partícula, representando-os em um vetor posição e a mudança de posição consiste numa troca de posições no vetor posição (swap) da partícula.
Já no ONP aplicado a problemas contínuos, as partículas movimentam-se no espaço através de uma soma vetorial, ou seja, é feita uma soma algébrica do espaço atual com a velocidade para se obter uma nova posição.
Analisando a velocidade no ONP Discreto, pode-se concluir que ela consiste nas trocas de informações realizadas nos vetores posições das partículas, os "swaps", sendo sua soma, uma concatenação de listas de "swaps". A velocidade de uma partícula consiste em uma lista de "swaps" que será aplicada na posição da partícula para se obter uma nova posição. No ONP Contínuo a velocidade de uma partícula é um número (escalar) que será adicionado à sua posição para se obter uma nova posição no espaço de busca.

4. Utilização do ONP em Problemas Combinatórios

Para implementarmos o ONP em problemas combinatórios, faz-se necessária a introdução de alguns conceitos para que possamos utilizar os operadores matemáticos [8]. Abaixo descreveremos cada um destes:

4.1. Velocidade

Como dito anteriormente, a velocidade é uma lista de transposições. Esta pode ser equivalente, ou seja, se aplicarmos V_1 ou V_2 e obtivermos o mesmo resultado. Uma velocidade nula é uma lista de velocidade sem transposições, onde |V| significa a quantidade de transposições na lista velocidade.

4.2. Movimentação (Posição + Velocidade)

Dado P uma posição e V uma velocidade, logo, P' será uma nova posição onde P' = P + V, onde devem ser aplicadas todas as transposições pertencentes a V.

4.3. Obtenção da Velocidade (Posição – Posição)

Supondo duas soluções (posições) A e B, nós obteremos a velocidade V, subtraindo uma posição da outra, V = A – B, onde o sinal de " - " possui um novo significado. Através dessa diferença obteremos a lista de transposições, resultando na seguinte equação:

$$A = B + V.$$

Como podemos verificar no exemplo logo abaixo:

Exemplo: Dadas duas posições A e B:

A: (1 2 3 4 5) B: (2 3 1 5 4)

→ A (1) = B (3) = 1, logo a primeira transposição será (1,3), então teremos um B'= B + S(1,3).
→ B' = (1 3 2 5 4)
→ A (2) = B' (3) = 2, logo a segunda transposição será (2,3), então teremos um B''= B' + S(2,3).

→ B'' = (1 2 3 5 4)
→ Portanto o terceiro operador será (4,5), logo teremos a lista de transposição (V), que será igual
→ A – B = S(1,3), S(2,3), S(4,5).

4.4. Adição entre Velocidades (Velocidade + Velocidade)

A adição entre duas velocidades consiste na concatenação de listas de transposições. Resultando uma nova lista.

4.5. Multiplicação (Coeficiente * Velocidade)

Aqui, chamamos o coeficiente de C, onde podemos ter quatro casos para este coeficiente como podemos verificar a seguir:

(i) C = 0, teremos C*V = nulo;
(ii) C pertence [0,1], truncamos V por C*|V|;
(iii) C > 1, aplicamos V, C vezes mais a parte decimal de C*|V|;
(iv) C < 0, não é definido pra este tipo de implementação.

5. ONP Discreto Aplicado ao Problema do Caixeiro Viajante

Para realização dos testes, aplicamos o ONP em três problemas da TSPLIB: br17, ftv35 e swiss42. Como os resultados obtidos com o *ONP clássico* não foram muito satisfatórios foi necessário introduzirmos alguns artifícios, buscando uma melhoria nos resultados e na velocidade do ONP, explicitados abaixo:

i) Inserção de um contador para cada partícula. Caso esse contador venha a ultrapassar um certo número de iterações, decidido pelo usuário, sem melhorias no seu fitness, a partícula é colocada em sua melhor posição encontrada desde o início da execução do programa;
ii) Inclusão de um fator de tolerância. Em dado momento da execução, a partícula diminuirá sua velocidade até parar, de acordo com a equação (1a), pois ela tende a ficar com a melhor rota encontrada pelo enxame, então decidimos que, até que se atingisse um certo valor de tolerância das iterações sem que haja velocidade na partícula, a mesma fosse deslocada para uma posição melhor dentre todas posição encontrada por qualquer outra partícula de sua vizinhança;
iii) E caso não ocorra nenhum dos casos anteriores, a partícula é deslocada para uma posição qualquer no espaço.

A base para os artifícios descritos acima, foram inspirados nos modelos elaborados por Maurice Clerc [8], mas um dos nossos objetivos é implementar métodos diferentes dos já presentes na literatura e que sejam tão eficientes quanto.

Com a inserção destes artifícios, denominamos o algoritmo de Melhoria 1. Porém, como os resultados ainda ficaram muito distantes do valor ótimo informado nas instâncias testes da TSPLIB, aplicamos um procedimento de busca local e denominamos esse novo algoritmo de Melhoria 2. Os resultados para este último algoritmo foram bastante animadores, como pode ser verificado nas tabelas 2 e 3.

RESULTADOS OBTIDOS COM A METAHEURÍSTICA ONP DISCRETO

Tabela 1 – Resultados obtidos com o algoritmo PSO Melhoria 1

br17.atsp	ótimo: 39	20 partículas			30 partículas			50 partículas	
TOLER.=	4	Iterações	Fitness		Iterações	Fitness		Iterações	Fitness
		398	39		2000	41		890	39
		1812	39		220	39		636	39
		2000	42		1377	39		1292	39
		1391	39		197	39		252	39
		2000	41		2000	41		182	39
		122	39		2000	44		530	39
		2000	41		338	39		85	39
		2000	40		2000	41		43	39
		772	39		2000	41		569	39
		2000	42		2000	41		334	39
		1449,5	40,1		1413,2	40,5		481,3	39

Tabela 2 – Resultados obtidos com o algoritmo PSO Melhoria 1 e Melhoria 2, para 5000 iterações

ftv35	ótimo:1473	50 partículas			80 partículas			120 partículas	
TOLER.=	4	Melhoria 1	Melhoria 2		Melhoria 1	Melhoria 2		Melhoria 1	Melhoria 2
		2095	1780		2103	1653		2170	1709
		2394	1829		2114	1768		1993	1758
		1935	1981		1811	1938		2081	1773
		2314	1829		2272	1754		2116	1705
		2346	1670		2164	**1599**		2312	1612
		2216,8	1817,8		2092,8	1742,4		2134,4	1711,4

Tabela 3 – Resultados obtidos com o algoritmo PSO Melhoria 1 e Melhoria 2, para 5000 iterações

swiss42	ótimo:1273	50 partículas			80 partículas			120 partículas	
TOLER.=	4	Melhoria 1	Melhoria 2		Melhoria 1	Melhoria 2		Melhoria 1	Melhoria 2
		1863	1696		1554	1751		1778	1532
		2062	1609		1824	1655		1701	1690
		1978	1703		1761	1456		1866	1649
		1927	1681		1715	1477		1895	**1419**
		1992	1537		1781	1599		1947	1420
		1964,4	*1645,2*		*1727*	*1587,6*		*1837,4*	*1542*

6. CONCLUSÕES

O ONP refere-se a uma família relativamente nova de algoritmos que podem ser usados para encontrar soluções ótimas (ou próximas do ótimo). Nos últimos anos, o ONP vem obtendo bons resultados, sendo aplicado em muitas pesquisas em diversas áreas do conhecimento. Principalmente, em problemas contínuos não-lineares nas áreas de engenharia. Nossa meta, nesse artigo, foi apresentar a aplicação dessa técnica em problemas de otimização combinatória NP-árduos, além de fazer alguns experimentos a fim de acelerar e melhorar a performance dessa metaheurística evolucionária.

Também aplicamos o nosso algoritmo a instâncias maiores na TSPLIB, mas verificamos que ainda há uma certa distância entre os valores encontrados para o valor ótimo, o que mostra que ainda há um campo de pesquisa a ser explorado, que está sendo a meta de nosso estudo atual.

Nossa linha de atuação em trabalhos futuros consistirá em trabalharmos no sentido de melhorarmos os resultados obtidos pelo ONP discreto utilizando outros mecanismos: hibridização, paralelização, etc., em instâncias maiores.

7. REFERÊNCIAS BIBLIOGRÁFICAS

[1] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence Through Simulated Evolution*. New York, NY: John Wiley & Sons, 1966.

[2] H.-P. Schwefel, *Evolution and Optimum Seeking*. New York: John Wiley & Sons, 1995.

[3] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence Through Simulated Evolution*. New York, NY: John Wiley & Sons, 1966.

[4] Darwin, C. Origin of species by means of natural selection, or the preservation of favored races in the struggle for life. 6th Edition, v. I and II. John Murray: London, Albemarle Street, 1859 (1a. ed), disponível em: *http://honors.ccsu.ctstateu.edu/Honors/EText/Darwin/DarwinOriginContents.html*

[5] Goldberg, D. E., *Genetic Algorithms in Search, Optimization, and Machine Learning, Reading MA*: Addison-Welsey, 1989.

[6] J. Kennedy and R.C. Eberhart, "*Particle Swarm Optimization", Proceedings of the International Conference on Neural Networks"*, Perth, Australia, 1995.

[7] Eberhart, R.C., Shi, Y.. Evolving artificial neural networks. In: *Proceedings of the International Conference on Neural Networks and Brain,* pp. PL5–PL13, 1998.

[8] *Discrete Particle Swarm Optimization Illustrated by the Traveling Salesman Problem* Maurice Clerc, 29 February 2000, disponível em *http://www.mauriceclerc.net*