https://eventos.utfpr.edu.br//sicite/sicite2020

Algoritmo de enxame de gatos para otimização binária

Cat swarm algorithm for binary optimization

Fernando Calixto Curi
fercuri@gmail.com
Universidade Tecnológica Federal do Paraná, Ponta Grossa, Paraná, Brasil

Hugo Valadares Siqueira
hugosiqueira@utfpr.edu.br
Universidade Tecnológica Federal do Paraná, Ponta Grossa, Paraná, Brasil

RESUMO

A constante busca pelo desenvolvimento e aperfeiçoamento das tecnologias existentes é uma necessidade cada vez mais fundamental para a sociedade. Nos últimos anos, com os avanços no campo computacional, abriram-se novas possibilidades para a solução de problemas antes vistos como muito difíceis e até impossíveis. A busca pela otimização de funções é um importante passo nessa caminhada, pois oferece uma solução satisfatória para problemas complexos em menos tempo, um recurso cada vez mais precioso nos dias atuais. Este trabalho teve como objetivo a realização de testes de desempenho de dois algoritmos computacionais de inteligência coletiva, o *binary cat swarm optimization* e o *boolean binary cat swarm optimization*, programas que tem seu funcionamento baseado na computação natural. Ambos foram empregados para a solução de um problema e seus resultados foram comparados para determinar qual conseguiu atingir uma solução ótima de forma mais eficiente.

PALAVRAS-CHAVE: Algoritmos computacionais. Computação natural. Inteligência coletiva.

ABSTRACT

The constant seek for the development and improvement of existing technologies is an increasingly fundamental need of the society. In recent years, with advances in the computational field, new possibilities have been opened for the solution of problems previously seen as very difficult and even impossible. The search for optimization of functions is an important step in this journey, as it offers a satisfactory solution to complex problems in less time, an increasingly precious resource nowadays. This paper has as purpose to perform performance tests of two computational collective intelligence algorithms, the binary cat swarm optimization and the boolean binary cat swarm optimization, programs that work based on natural computing. Both were used to solve a problem and their results were compared to determine which one was able to achieve an optimal solution more efficiently.

KEYWORDS: Computer algorithms. Natural computation. Swarm intelligence.

Recebido: 19 ago. 2020.
Aprovado: 01 out. 2020.

INTRODUÇÃO

Nos mais diversos campos do conhecimento humano, quando se utiliza a palavra otimizar busca-se sempre um ponto no qual uma determinada coisa acontece próxima ao máximo desejado. A gama de problemas passíveis de otimização é enorme, já que a complexidade de um problema é determinada pela quantidade de variáveis que o compõe.

Nas últimas décadas com o aumento da capacidade de processamento computacional vem sendo desenvolvidos métodos de otimização heurísticos, em que se busca não uma solução ideal do problema, mas uma gama de soluções que se aproximem desse ponto e sejam aceitáveis e passíveis de serem aplicados e que ao mesmo tempo não necessitem dispender tanto tempo na busca de uma resposta final. (SERAPIÃO, 2009)

Muitos desses métodos têm forte inspiração na natureza e no comportamento de certos animais em sua busca por comida e demais recursos necessários para sua sobrevivência. A esses métodos dá-se o nome de computação natural.

Segundo BALLARD (1999) a computação natural descreve todos os sistemas computacionais que são de alguma forma inspirados em algum mecanismo natural ou biológico de processamento de informação. Tal área pode ser dividida em três grandes grupos: computação inspirada na natureza, a simulação de fenômenos naturais e a computação utilizando meios naturais.

ALGORITMOS BIO-INSPIRADOS

Para a solução de problemas utilizando algoritmos o mais comum é aplicar a lógica encontrada no primeiro grupo da computação natural. São os denominados algoritmos bio-inspirados, assim descritos por se basearem na observação dos fenômenos da natureza.

Com o desenvolvimento computacional que ocorreu nas últimas décadas e com o avanço das pesquisas biológicas sobre o funcionamento e princípios dos elementos desta área, pesquisadores do campo da engenharia e computação perceberam que era possível usar estes princípios, teorias e modelos para a implementação de sistemas computacionais com grande potencial de resolver problemas. Assim surgiram diversas técnicas como as redes neurais artificiais, os algoritmos evolutivos, a inteligência coletiva/de enxame (*swarm intelligence*), os sistemas imunológicos artificiais, dentre outros. (DE CASTRO, 2004)

Segundo BALLARD (1999), baseado nesses conceitos, duas abordagens têm se destacado: os algoritmos evolutivos e os de enxame de partículas. O primeiro se baseia no princípio de diversidade e da sobrevivência de indivíduos mais fortes e bem adaptados ao ambiente, que se reproduzem através de operadores que emulam os conceitos genéticos, criando descendentes mais fortes que se aproximam da solução do problema.

Já os algoritmos de enxame são um conjunto de técnicas baseadas no comportamento coletivo de sistemas auto-organizados, distribuídos, autônomos, flexíveis e dinâmicos. Estes sistemas possuem a característica de compreender o ambiente e altera-lo através de seus agentes computacionais primários pelos quais

são formados, ocasionando uma influência entre si que modifica a forma como irão agir no futuro. Embora não exista uma central de comando que estabelece o padrão de comportamento dos agentes, nem um modelo explícito do ambiente, as interações locais geralmente levam ao surgimento de um comportamento global que se aproxima da solução do problema. (SERAPIÃO, 2009)

OTIMIZAÇÃO POR ENXAME DE GATOS

A otimização por enxame de gatos, conhecido por seu nome original em inglês *Cat Swarm Optimization* (CSO) é um algoritmo de enxame baseado nas habilidades dos gatos quando estão caçando ou atentos a possíveis perigos. Foi proposta inicialmente por Chu em 2006, motivada pelas técnicas de *Particle Swarm Optimization* (PSO) e *Ant Colony Optimization* (ACO).

Segundo os autores, o comportamento dos felinos possui características similares em todos os subgrupos da espécie, mesmo que se destaquem mais em algumas em particular. Nos gatos domésticos é comum que sejam assumidos dois modos principais de ações: a posição de descanso e a posição de caça. A esses modos foram dados os nomes de *seeking* e *tracing*. (CHU, 2006)

No primeiro modo o agente encontra-se "descansando", ou seja, estático, porém ao mesmo tempo analisando o próximo movimento a ser feito. Nesse modo são definidos quatro fatores:

a) O tamanho da memória de procura (*seeking memory pool* - SMP): refere-se ao número de pontos a serem explorados;

b) A dimensão selecionada do alcance de procura (*seeking range of the selected dimension* – SRD): representa a probabilidade de mutação para as dimensões determinadas;

c) Contagens de dimensões para mudar (*counts of dimension to change* - CDC): estabelece um número de dimensões D que serão passíveis de mutação;

d) A própria posição considerada (*self-position considering* - SPC): uma *flag* booleana que determina se a posição atual vai ou não ser considerada.

O funcionamento deste modo ocorre em 5 passos:

Passo 1: Faz j cópias da atual posição do gato k, onde j=SMP. Se o valor de SPC for verdadeiro então j=SMP-1 e armazena a atual posição como um dos candidatos;

Passo 2: Soma ou diminui aleatoriamente o valor de porcentagem de SRD em cada cópia e sobrescreve os valores antigos;

Passo 3: Calcula-se o valor de *fitness* (FS) de todos os pontos de candidatos;

Passo 4: Se os valores de FS não forem iguais, calcula-se a probabilidade selecionada de cada ponto de candidato utilizando a equação (1). Caso contrário define-se todas as probabilidades dos pontos de candidato para 1;

Passo 5: Escolhe aleatoriamente o ponto para mover a partir dos pontos candidatos e substitui a posição do gato k.

$$Pi = \frac{|FSi - FSb|}{FS\max - FS\min}, \text{ onde } 0 < i < j \quad (1)$$

No modo *tracing* foram modeladas as características do caso onde o gato encontra-se rastreando suas presas. Uma vez nesse modo o agente move-se de acordo com sua própria velocidade para cada dimensão. As ações deste modo podem ser descritas em três passos:

Passo 1: Atualiza-se a velocidade para cada dimensão de acordo com a equação (2);

$$vk,d = vk,d + r1.c1.(xbest,d - xk,d), \text{ onde d=1, 2, ..., M} \quad (2)$$

Sendo:

xbest,d é a posição do gato com o melhor valor de *fitness*

xk,d é a posição do gato k

c1 é a constante

r1 é um valor no *range* de [0,1]

Passo 2: Checa se as velocidades estão no alcance da velocidade máxima. No caso de a velocidade ultrapassar o alcance, define-se como igual ao limite;

Passo 3: Atualiza a posição do gato k de acordo com a equação (3)

$$xk,d = xk,d + vk,d \quad (3)$$

Para combinação dos dois modos utiliza-se uma relação de mistura (MR), visando criar um algoritmo final denominado CSO. O processo deste algoritmo pode ser descrito em seis passos:

Passo 1: Criar *N* gatos no processo;

Passo 2: Distribuir aleatoriamente os gatos em um espaço de soluções *M* dimensional e selecionar valores aleatórios para o alcance do máximo velocidade de cada agente. Depois define-se um número de gatos para o modo *tracing* de acordo com a MR previamente escolhida e os demais para o modo *seeking*;

Passo 3: Avalia-se o valor de *fitness* de cada indivíduo aplicando as posições dos gatos em uma função fitness, que representa o critério do objetivo final. Mantem-se o melhor gato na memória (*xbest*);

Passo 4: Mover os gatos de acordo com seu modo selecionado (indivíduos do *tracing* realizam a função *tracing* e indivíduos da função *seeking* realizam a função *seeking*)

Passo 5: Escolher novamente o número de gatos a serem atribuídos a função *tracing* de acordo com o MR, e definir os demais para o modo *seeking*;

Passo 6: Checar a condição de parada. Caso satisfeita interrompe-se o processo, caso contrário repetem-se os passos 3 a 6;

Nesta pesquisa procurou-se aplicar o CSO para otimização em funções binárias. Assim algumas alterações foram feitas no algoritmo original com base nos trabalhos de Soto e Siqueira. O primeiro denominou seu método como *Binary Cat Swarm Optimization* (BCSO), e uma versão simplificada do cerne do código pode ser vista na Fig. 1.

Figura 1 - Pseudocódigo BCSO

```
(1) Criar C gatos
(2) Definir aleatoriamente a posição inicial de gatos com valores entre 1 e 0
(3) Inicializar a velocidade e o flag de cada gato
(4) while(i<numero_de_iteracoes) do
(5)         Avaliar os gatos de acordo com a função fitness
(6)         Armazenar a posição do melhor gato, que tem o melhor fitness
(7)         for(x=1 to C) do
(8)             if (numero_aleatorio<razao_de_mistura) then
(9)                 Aplicar o processo do modo seeking no gato x
(10)            else
(11)                Aplicar o processo do modo tracing no gato x
(12)            end if
(13)        Avaliar a nova solução, atualizar valores
(14)    end for
(15) end while
(16) Resultados pós processo e visualização
```

Fonte: adaptado de Soto, 2019

A segunda versão do algoritmo utilizada foi desenvolvida com base no trabalho de Siqueira e é intitulada *Boolean Binary Cat Swarm Optimization* (BBCSO). Seu pseudocódigo pode ser observado na Fig. 2.

Figura 2 - Pseudocódigo BBCSO

```
(1) Inicializar todas as posições dos gatos aleatoriamente, MR, SMP, CDC e
    PMO, e definir as velocidades como um vetor zerado com D dimensões
(2) while condição de parada não é atingida do
(3)     Avaliar cada gato e salvar o x_best
(4)     Escolher MR% dos gatos usando o método da roleta para executar o
        tracing mode e considerar os demais em seeking mode
(5)     for cada gato do
(6)         if x_i^t estiver em seeking mode then
(7)             Fazer SMP cópias do gato atual
(8)             for cada cópia do
(9)                 Selecionar CDC dimensões e alterar valores com
                    probabilidade PMO
(10)            end for
(11)            Avaliar o fitness das cópias SMP
(12)            Aplicar o método da roleta usando as cópias e a posição
                original, selecionando um candidato para permanecer no
                enxame
(13)        end if
(14)        if x_i^t estiver em tracing mode then
(15)            Atualizar a velocidade do gato
(16)            Atualizar a posição do gato
(17)        end if
(18)    end for
(19) end while
(20) Retornar o x_best como a solução final
```

Fonte: adaptado de Siqueira, 2018

RESULTADOS E DISCUSSÕES

Para testar os algoritmos desenvolvido nesta pesquisa optou-se por utilizar uma função de teste denominada *OneMax*. Este problema, também conhecido como *BitCounting*, tem por objetivo maximizar os números de 1 em uma *string*. Caso o algoritmo consiga encontrar uma solução onde todos os elementos são 1 pode-se afirmar a validade do funcionamento do programa. É importante salientar que, apesar do problema parecer ter uma solução trivial por já se ter conhecimento

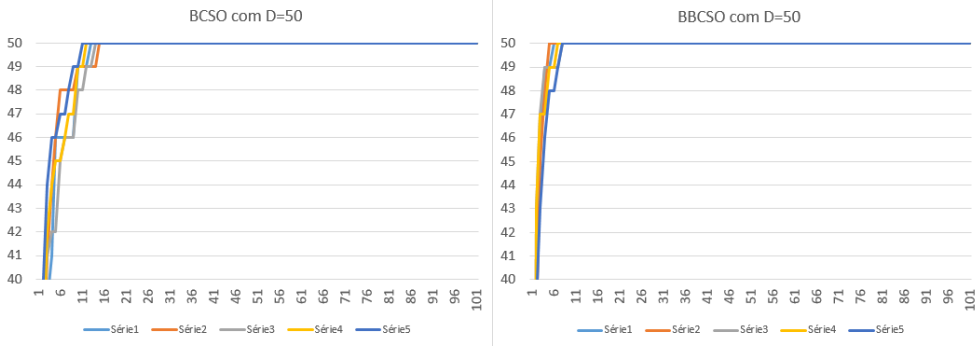
da resposta esperada, a proposta é que o algoritmo use apenas os passos projetados para encontrar tal solução, substituindo a cada iteração o melhor elemento dentre os testados. Assim, ao ser adaptado para outras situações espera-se que o programa tenha o mesmo comportamento e consiga encontrar uma solução ótima. Formalmente este problema pode ser descrito como achar uma *string* $\vec{x} = \{x_1, x_2, \dots, x_n\}$ *com* $x_i \in \{0,1\}$, que maximize a equação (4):

$$F(\vec{x}) = \sum_{i=1}^{N} x_i \tag{4}$$

O desenvolvimento dos algoritmos e a execução dos testes foi realizada na mesma máquina para que a quantidade de memória e os *clocks* de operação não tivessem efeito sobre o resultado final. A linguagem de programação utilizada foi a *python*, em sua versão 3.7.

Inicialmente executou-se os dois códigos com 10 partículas (gatos) e uma *string* com 50 dimensões. Definiu-se o número de iterações máximo como 100, sendo esse o critério de parada. Após 5 testes verificou-se que ambos os algoritmos conseguiram maximizar as 50 dimensões do problema, sendo o BBCSO um pouco mais rápido ao executar essa função. A Fig. 3 mostra os gráficos dos valores obtidos para cada série de testes.

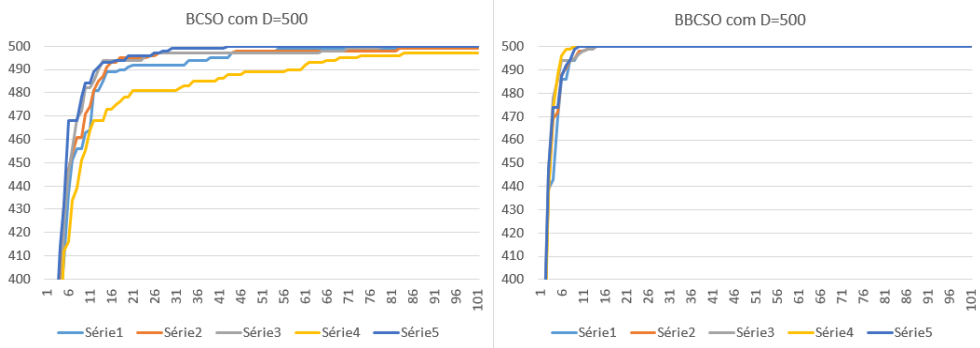
Figura 3 - Comparativo BCSO e BBCSO com 50 dimensões



Fonte: Autoria própria (2020).

Novamente os testes foram repetidos, desta vez alterando a quantidade de dimensões para 500. Observou-se que em duas das séries executadas o BCSO não conseguiu maximizar a string do *OneMax.* Já o BBCSO obteve êxito em todas as tentativas. A Fig. 4 mostra melhor esta discrepância e a diferença de desempenho entre as versões binária e booleana do algoritmo.

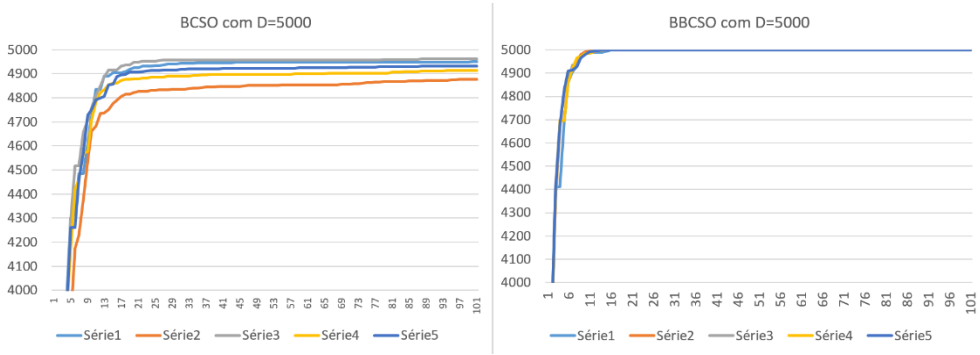
Figura 4 - Comparativo BCSO e BBCSO com 500 dimensões



Fonte: Autoria própria (2020).

Por fim executou-se um novo teste com 5000 dimensões. Apesar de ter permanecido com 10 agentes e um número máximo de 100 interações, o BBCSO conseguiu atingir o limite do problema em todos os testes realizados. Já o BCSO não conseguiu maximizar o problema proposto em nenhuma das tentativas, porém os valores encontrados foram bem próximos dos desejados, conforme a Fig. 5 exemplifica.

Figura 5 - Testes com 5000 dimensões



Fonte: Autoria própria (2020).

CONCLUSÕES

O presente trabalho apresentou a aplicação de dois métodos de algoritmos de otimização bio-inspirados, o *Binary Cat Swarm Optimization* e o *Boolean Binary Cat Swarm Optimization*. Estes foram aplicados para a resolução do problema *OneMax*, sendo feito três testes diferentes: para 50 dimensões, para 500 dimensões e para 5000 dimensões.

De acordo com os testes realizados o BBCSO provou ser mais eficiente que o BCSO, dada a natureza do seu algoritmo ser mais simplificada, além de uma maior competência na utilização dos recursos computacionais.

Futuros trabalhos poderão ser desenvolvidos para comprovar a aplicabilidade destes algoritmos, como na resolução do problema da mochila, no problema do caixeiro viajante e em diversos outros problemas de otimização combinatória

REFERÊNCIAS

BALLARD, Dana H. **An introduction to natural computation**. Mit Press, 1999.

CHU, Shu-Chuan; TSAI, Pei-Wei; PAN, Jeng-Shyang. **Cat swarm optimization**. In: Pacific Rim International Conference on Artificial Intelligence. Springer, Berlin, Heidelberg, 2006. p. 854-858.

DE CASTRO, Leandro N. et al. **Computação natural**: Uma breve visão geral. In: Workshop em nanotecnologia e Computação Inspirada na Biologia. 2004.

SERAPIÃO, Adriane Beatriz de Souza. **Fundamentos de otimização por inteligência de enxames**: uma visão geral. Sba: Controle e Automação Sociedade Brasileira de Automatica, v. 20, n. 3, p. 271-304, 2009.

SIQUEIRA, Hugo et al. **Boolean binary cat swarm optimization algorithm**.2018 IEEELatin American Conference on Computational Intelligence (LA-CCI), p. 1–6,2018.

SOTO, Ricardo et al. **Solving the manufacturing cell design problem through binary cat swarm optimization with dynamic mixture ratios**. Computational Intelligence and Neuroscience, v. 2019, p. 1–16, 02 2019.