Um algoritmo bioinspirado em polinização de plantas usando entropia populacional

Diego Darvy Moreira
Universidade Federal do Espírito Santo - UFES
Departamento de Informática
Av. Fernando Ferrari, 514
CEP 29060-270, Vitória, ES
diegodarvy@gmail.com

Renato A. Krohling
Universidade Federal do Espírito Santo - UFES
Departamento de Engenharia de Produção &
Programa de Pós-graduação em Informática, PPGI
Av. Fernando Ferrari, 514, Prédio CT X
CEP 29060-270, Vitória, ES
krohling.renato@gmail.com

RESUMO

O algoritmo bioinspirado na Polinização de Flores (em inglês: *Flower Pollination Algorithm*, abreviado por FPA) usa como metáfora a polinização de plantas através das flores. Resultados para problemas de otimização com este algoritmo têm se mostrado promissores, mas pode ocorrer que durante o processo de busca ele fique preso em mínimos locais. Neste trabalho, o algoritmo FPA será modificado usando a entropia populacional para controle entre os modos de busca local e global. Outra modificação será feita na forma de busca local do algoritmo com a inserção de um termo gerado pela distribuição de probabilidade Gaussiana. O algoritmo desenvolvido foi testado para problemas de otimização e comparado com a versão original e também com o algoritmo de Otimização por Enxame de Partículas (em inglês*: Particle Swarm Optimization*, abreviado por PSO). Resultados de simulação mostram a efetividade da nova versão do algoritmo FPA desenvolvida.

PALAVRAS CHAVE: Otimização baseado em polinização de plantas; algoritmos evolutivos; entropia*;* voos de Lévy.

Área Principal: Metaheurística

ABSTRACT

The bio-inspired algorithm Flower Pollination Algorithm (FPA), uses the metaphor of the pollination of plants through flowers. Results for optimization problems with this algorithm have shown promising results, but it may happen that during the search process it gets trapped into local minima. In this work, the FPA algorithm is modified by using population entropy for control between the local and global search modes. Another modification is proposed in the form of its local search by the insertion of a term generated by a Gaussian probability distribution. The developed algorithm was tested for optimization problems and compared with the original version of FPA and also with the Particle Swarm Optimization (PSO). Simulation results show the effectiveness of the new version of FPA algorithm developed.

KEYWORDS: Flower pollination algorithm; evolutionary algorithms; entropy*;* Lévy flights.

Main area: Metaheuristics

1. Introdução

Métodos clássicos para resolver problemas de otimização não lineares requerem uso de gradiente e da derivada segunda. Estes métodos apresentam limitações quando aplicados em problemas multimodais. Na computação bioinspirada os algoritmos são desenvolvidos com base no comportamento da natureza. Atualmente existe uma vasta literatura cobrindo algoritmos bioinspirados, como: Algoritmos Genéticos, Evolução Diferencial, Estratégia Evolutiva, entre outros aplicados na resolução de problemas de otimização não lineares (Binitha & Sathya, 2012).

O algoritmo baseado na Polinização de Flores foi proposto por Yang (2012). Resultados para problemas de otimização usando este algoritmo têm se mostrado promissores, mas pode ocorrer que durante o processo de busca ele falhe ao ficar preso em um mínimo local (Wang & Zhou, 2014). O FPA tem sido aplicado por exemplo para solucionar problemas de otimização com restrições (Raouf, Baset & El-Henawy, 2014a). Nesse caso foi feita a hibridização de FPA com o algoritmo PSO a fim de melhorar as características de sua busca local. Também foi aplicado para problemas de otimização do tempo de vida em uma rede de sensores sem fio (Sharawi et al., 2014). No despacho econômico de cargas, cuja principal tarefa consiste na otimização de sistemas geradores de energia. Foi investigado como minimizar o custo de combustível utilizado nos geradores de forma a obter a melhor geração de carga usando FPA (Prathiba, Moses & Sakthivel, 2014). Na resolução de quebra-cabeças do tipo Sudoku, o FPA é usado junto com o algoritmo Busca Harmônica Caótica (em inglês: *Chaotic Harmony Search*) para melhorar a eficácia de sua busca local (Raouf, Baset & El-Henawy, 2014b).

Neste trabalho, o algoritmo FPA será modificado através da introdução da entropia populacional para controle entre os modos de busca local e global. Outra modificação será feita na forma de busca local do algoritmo com a inserção de um termo gerado pela distribuição de probabilidade Gaussiana. O algoritmo modificado será aplicado para problemas de otimização não lineares multimodais. Um estudo de sensibilidade dos parâmetros dos algoritmos será feito para cada função de teste.

Este trabalho está organizado da seguinte forma: na seção 2 será descrito o algoritmo baseado em polinização de plantas - FPA. Na seção 3 serão desenvolvidas modificações no algoritmo original FPA. Na seção 4 serão apresentados os resultados de simulação para problemas de otimização. Também será feita uma comparação de desempenho entre a versão modificada e a versão original do FPA, além disso uma comparação com o algoritmo de otimização PSO. Conclusões e direções para trabalhos futuros finalizam o artigo na seção 5.

2. Algoritmo inspirado na polinização de plantas

2.1 Introdução

É estimado que existam mais de 250 mil tipos de plantas floríferas na natureza e que cerca de 80% de todas as espécies de plantas do planeta sejam também do tipo que produz flores. Plantas floríferas têm evoluído por mais de 125 milhões de anos e as flores tornaram-se muito influentes no processo de evolução de suas espécies. Não se consegue imaginar como seria o mundo das plantas sem as flores. O objetivo principal das flores é principalmente a reprodução via polinização.

Polinização é tipicamente associada com a transferência de pólen, e tal transferência, está geralmente ligada a polinizadores como: insetos, pássaros, morcegos e outros animais. De fato, algumas flores e insetos, coevoluíram em uma parceria muito bem especializada. Por exemplo, algumas flores podem somente atrair e somente depender de uma única espécie de inseto para ter sucesso na polinização (Glover, 2007).

A polinização pode ocorrer de duas formas principais: abiótica e biótica. Cerca de 90% das plantas floríferas pertencem ao tipo biótico, isto é, o pólen é transferido por um polinizador. O restante das plantas faz a polinização do tipo abiótica, vento e difusão na água ajudam na polinização destas plantas. A grama também serve como um bom exemplo (Glover, 2007).

Polinizadores podem ser bem diversos, é estimado que existam pelo menos 200 mil variedades de polinizadores. As abelhas são um bom exemplo de polinizador, e elas podem desenvolver a chamada lealdade a uma flor (em inglês: *flower constancy*) de acordo com Glover (2007). O que significa que estes polinizadores tendem a visitar exclusivamente certo tipo de espécie de flor. Tal lealdade a uma flor traz vantagens evolutivas pois maximizará a transferência de pólen entre flores da mesma espécie e consequentemente sua reprodução. Além do mais, a lealdade a uma flor pode ser utilizada como um passo incremental empregando à similaridade ou diferença entre duas flores como propôs Yang (2012).

Vale destacar que a polinização também pode ser alcançada via autopolinização ou polinização cruzada. Polinização cruzada ou alogamia significa que a polinização pode ocorrer do pólen de uma dada planta para outra diferente, enquanto a autopolinização é a fertilização entre plantas da mesma espécie, que pode ocorrer quando não há nenhum polinizador disponível (Glover, 2007).

A polinização biótica e cruzada geralmente ocorre a longas distâncias e os polinizadores como abelhas, pássaros e morcegos podem voar longas distâncias e desta forma podem ser considerados como polinizadores globais. Tal comportamento dos polinizadores pode ser modelado pelos voos de Lévy (em inglês: *Lévy flights*). Neste caso, o salto ou voo distante obedece a distribuição de probabilidade de Lévy (Tran, Nguyen & Nguyen, 2004).

2.2 Voos de Lévy

Basicamente, os voos de Lévy são um tipo de caminhada aleatória cujo tamanho do passo é originado a partir da distribuição de Lévy, que é expressa em termos de uma lei de potência, como descrito por Yang (2014):

$$L(s) \sim |s|^{-1-\beta} \tag{1}$$

onde β é um índice no intervalo [0, 2] e s uma variável que descreve o tamanho do passo. A distribuição de Lévy também pode ser expressa em termos da transformada de Fourier (Yang, 2014):

$$F(k) = \exp[-\alpha |k|^{\beta}], 0 < \beta \leq 2 \tag{2}$$

onde α é um parâmetro de escala e β é um índice. Para o caso de $\beta = 2$ temos um caso especial, a transformada inversa de Fourier da equação (2) corresponde a distribuição Gaussiana. Outro caso especial é quando $\beta = 1$ que corresponde a distribuição de Cauchy (Yang, 2014). Para casos gerais a integral inversa, descrita pela equação (3), pode ser estimada somente quando s é muito grande. A inversa desta integral não é fácil de ser calculada, pois ela não possui uma forma analítica, exceto para alguns casos especiais (Yang, 2014).

$$L(s) = \frac{1}{\pi} \int_0^{\infty} \cos(ks) \exp[-\alpha |k|^{\beta}] dk \tag{3}$$

Assim, quando $s \rightarrow \infty$:

$$L \sim \frac{\alpha \beta \Gamma(\beta) sen(\frac{\pi \beta}{2})}{\pi} \frac{1}{s^{1+\beta}}, (s >> s_0 > 0) \tag{4}$$

onde $\Gamma(\beta)$ é a função Gamma e s_0 é um passo mínimo.

A distribuição de Lévy possui como característica variância infinita, por isso é chamada de distribuição de calda longa (em inglês: *heavy tail*). No contexto dos algoritmos de polinização, algumas soluções devem ser geradas ao redor da melhor solução encontrada até então, o que pode acelerar a busca local. Entretanto, uma parcela maior de novas soluções deve ser gerada em locais distantes aleatórios, longe da melhor solução corrente assegurando maior exploração do espaço de busca e evitando também convergência prematura devido à queda em um mínimo local.

Do ponto de vista da implementação, a geração de números aleatórios com a distribuição de Lévy consiste em dois passos conforme descrito por Yang (2014): 1) escolha de uma direção aleatória e 2) geração de um passo que obedeça a distribuição de Lévy. A geração de uma direção aleatória pode ser feita usando uma distribuição uniforme. No entanto, a obtenção do passo não é trivial. Um algoritmo eficiente usado neste trabalho foi proposto por Mantegna para uma distribuição de Lévy simétrica. Neste caso, simétrica significa que o passo pode ser tanto positivo quanto negativo. No algoritmo de Mantegna, o tamanho do passo s é calculado da seguinte forma (Yang, 2014):

$$s = u / |v|^{1/\beta} \tag{5}$$

onde u e v são obtidos a partir de uma distribuição Gaussiana e β é um índice. Assim,

$$u \sim N(0,\sigma_u^2) \text{ e } v \sim N(0,\sigma_v^2) \tag{6}$$

$$\sigma_u = \left[\frac{\Gamma(1+\beta) sen(\frac{\pi\beta}{2})}{\Gamma(\frac{1+\beta}{2}) \beta 2^{(\beta-1)/2}} \right]^{1/\beta} \text{ e } \sigma_u = 1 \tag{7}$$

Esta distribuição obedece a distribuição esperada de Lévy para $|s| \geq s_0$, onde s_0 é um tamanho de passo mínimo. A princípio, s_0 pode ser tomado como um parâmetro de escala. Na implementação do algoritmo FPA, o parâmetro s_0 é chamado de η. Estudos comprovam que o voo de Lévy pode maximizar a eficiência de uma busca por recursos em ambientes desconhecidos (Tran, Nguyen, & Nguyen, 2004). De fato, voos de Lévy têm sido observados dentro do padrão de caça de albatrozes, leões e macacos aranha. Até humanos podem inconscientemente, somente por instinto, trilhar caminhos seguindo um padrão de voo de Lévy (Tran, Nguyen, & Nguyen, 2004).

2.3 O algoritmo FPA padrão

Yang (2012) idealizou um algoritmo baseado na polinização de plantas onde a lealdade a uma flor por parte de um polinizador e o seu comportamento obedecem às seguintes regras:

1. Polinização biótica e cruzada são consideradas como um processo de polinização global com os transportadores de pólen executando voos de Lévy.
2. Polinização abiótica e autopolinização são consideradas como polinização local.
3. Lealdade a uma flor pode ser considerada como tendo uma probabilidade de reprodução proporcional à similaridade das duas plantas envolvidas.
4. Polinização local e global são controladas por uma probabilidade $p \in [0,1]$. Devido à proximidade física e outros fatores como o vento, a polinização local pode ter uma significante parte nas atividades gerais de polinização.

Obviamente, cada planta pode ter múltiplas flores e cada flor libera milhares ou milhões de pólens. Entretanto, por simplicidade, no FPA é assumido que cada planta possui somente uma flor e cada flor produz somente um gameta. Assim no FPA não há necessidade de distinção entre

um gameta, uma flor ou uma planta como uma solução de um problema de otimização. Esta simplificação significa que uma solução candidata $X_i = [x_1, x_2, ..., x_D]$, onde *i* representa o índice de um indivíduo *X* e *D* a dimensão do vetor solução é equivalente a uma flor, planta e/ou um gameta.

O FPA possui dois elementos chave: a polinização global e a polinização local. Na polinização global, o pólen das plantas é carregado por polinizadores como insetos. Assim o pólen pode viajar longas distâncias porque insetos podem voar e mover-se em grandes espaços de busca. Isto assegura a polinização e reprodução da planta mais saudável (em inglês: *fittest)*. No FPA o indivíduo mais saudável é representado por g_*. A primeira regra juntamente com a lealdade a uma flor é descrita por:

$$X_i^{t+1} = X_i^t + L(X_i^t - g_*) \tag{8}$$

onde X_i^t é o pólen *i* no vetor solução X_i na iteração *t* e g_* é a melhor solução corrente encontrada entre todas as soluções na presente iteração. O parâmetro *L* é a força de polinização, que é essencialmente um valor gerado pela distribuição de Lévy. Desde que insetos podem mover-se a longas distâncias com percursos diferentes, pode-se usar o voo de Lévy para reproduzir esta característica (Tran, Nguyen, & Nguyen, 2004).
A polinização local de acordo com a regra 2, e a lealdade a uma flor é descrita por:

$$X_i^{t+1} = X_i^t + \varepsilon(X_j^t - X_k^t) \tag{9}$$

onde X_i^t é o pólen *i* no vetor solução X_i na iteração *t*, X_j^t e X_k^t são pólens de diferentes plantas da mesma espécie na mesma iteração. Isto essencialmente imita a lealdade de uma flor em uma vizinhança limitada. Se X_j^t e X_k^t são da mesma espécie ou são selecionadas da mesma população, isto se torna uma caminhada aleatória local (em inglês: *local random walk*) tomando-se ε de uma distribuição uniforme, ou seja, $\varepsilon \sim U(0,1)$ conforme Yang (2012).
A maioria das atividades de polinização pode ocorrer tanto em escala local ou global. Na prática, flores adjacentes ou não tão distantes da vizinhança são mais suscetíveis a polinização em comparação com aquelas mais distantes. Por isto, é usado uma probabilidade de troca de acordo com a regra 4, ou probabilidade de proximidade *p* para chavear entre fazer polinização global ou intensificar a polinização local. Na versão original do FPA o valor de *p* é de 0.8, favorecendo mais a busca local. No algoritmo 1 é listado o pseudo código do FPA proposto por Yang (2012).

3. Algoritmo FPA com entropia populacional e busca local Gaussiana (FPA-EG)

Um processo de otimização através de algoritmos bioinspirados deve considerar dois aspectos: exploração que descobre potenciais novos locais no espaço de busca (busca global) e intensificação que utiliza soluções promissoras já identificadas (busca local). O método de busca local é sensível às condições iniciais. Dependendo das condições dadas somente a busca local pode fácil e rapidamente encontrar o ótimo global (Mendel, Krohling & Campos, 2011). A exploração global é importante para possibilitar a procura por soluções em todo espaço de busca e prover um ponto inicial promissor para a intensificação através da busca local do algoritmo.

Sabe-se que manter a diversidade da população pode ajudar a reduzir a possibilidade de convergência prematura para um mínimo local (Liu, Mernik & Bryant, 2009). Para avaliar a diversidade da população algumas medidas têm sido propostas (Liu, Mernik & Bryant, 2009):

1. Desvio padrão da fitness populacional,
2. Desvio padrão da posição populacional,
3. Entropia da população.

Algoritmo 1: Pseudo código FPA

(1) **Parâmetros de entrada**: número de plantas *P*, escala do voo de Lévy η , limiar de probabilidade $p \in [0, 1]$, número máximo de iterações *max_iter*.

// Inicialização aleatória de uma população com posições X_i e dimensão *D* usando distribuição uniforme *U*(0,1)

// *Lb* e *Ub* são o limite inferior e superior do espaço de busca respectivamente.

(2) **para** *i=1:P*

(3) $X_i = Lb + (Ub - Lb) \cdot U(0,1)$

(4) $fit_i = f(X_i)$ // calcula fitness da solução

(5) **fim para**

(6) $g_* = \arg\min_{i=1,\ldots,P}\{fit_i\}$ // melhor solução na população inicial

(7) $f\min = f(g_*)$

(8) **enquanto** *t* < *max_iter*

(9) $rand \sim U(0,1)$

(10) **para** *i=1:P*

(11) **se** *rand* > *p*

(12) $L = \eta \cdot passo$ // $passo = u / |v|^{1/1.5}$ (Algoritmo de Mantegna)

(13) $X_i^{t+1} = X_i^t + L(X_i^t - g_*)$ // polinização global

(14) **senão**

(15) $\varepsilon \sim U(0,1)$

(16) aleatoriamente escolha *j* e *k* em $X_{i=1,\ldots,P}$

(17) $X_i^{t+1} = X_i^t + \varepsilon(X_j^t - X_k^t)$ // polinização local

(18) **fim se**

(19) $fit_{nova} = f(X_i^{t+1})$ // avalie a nova solução

(20) **se** $fit_{nova} < fit_i$ // se a nova solução é melhor, atualize ela

(21) $X_i^t = X_i^{t+1}$

(22) $fit_i = fit_{nova}$

(23) **fim se**

(24) // atualize o melhor global

(25) **se** $fit_{nova} < f_{\min}$

(26) $g_* = X_i^{t+1}$

(27) $f\min = fit_{nova}$

(28) **fim se**

(29) **fim para**

(30) **fim enquanto**

Na abordagem baseada na entropia da população, os valores da entropia são calculados em função da fitness e usados para chavear o algoritmo entre os regimes de busca global e local. Neste trabalho, será abordado a entropia da população como medida da diversidade e através de um estudo de sensibilidade será determinado um valor adequado para chavear entre busca global e local. Para mais informações sobre as duas primeiras formas de medir a diversidade populacional, i.e., desvio padrão da fitness populacional e desvio padrão da posição populacional, sugere-se a leitura de Liu, Mernik & Bryant (2009).

Entropia é um conceito da termodinâmica, teoria da informação e outros campos como mecânica estatística. Em teoria da informação, Shannon (1958) define entropia em termos de um evento aleatório discreto *u*, com possíveis estados *1, ..., H* e probabilidade p_i como:

$$H(u) = \sum_{i}^{H} p_i \log(\frac{1}{p_i}) = -\sum_{i}^{H} p_i \log p_i \tag{10}$$

Rosca (1995) introduziu a entropia na computação evolutiva adaptando a fórmula de Shannon. A entropia aplicada na computação evolutiva é computada classificando indivíduos dentro de uma classe de fitness *i* de acordo com seu valor normalizado. Quando a entropia é maior que um determinado limiar, o algoritmo é levado ao processo de intensificação em regiões visitadas; e caso contrário, quando a entropia é menor que um dado limiar, o processo tende a explorar novas regiões do espaço de busca. Em algoritmos evolutivos Rosca (1995) mostrou, através de simulações computacionais, que a população parece estar estagnada (presa) em um mínimo local quando a entropia não muda ou decresce monotonicamente em gerações sucessivas. Então, manter certa diversidade na população pode ajudar os indivíduos a manter a capacidade de exploração em locais não visitados no espaço de busca durante o processo evolutivo.

Assim, a entropia é uma forma de medir quando fazer a diversificação da população, mas é necessário definir também como fazer a diversificação. Isto depende do algoritmo utilizado. Por exemplo, no caso de um Algoritmo Genético (AG) isto pode ser feito reduzindo ou aumentando a probabilidade de mutação (Liu, 2009). Deste modo para um AG, se em um dado momento uma população está mais diversificada do que o esperado, a variável escolhida para como fazer, é então ajustada para a intensificação no espaço de busca, através de uma menor taxa de mutação dos indivíduos. Por outro lado, para evitar convergência prematura se a população está muito concentrada de acordo com o limiar de entropia, então a população é levada a diversificação via uma maior taxa de mutação dos indivíduos.

O cálculo da entropia populacional é descrito conforme os seguintes passos (Ni & Deng, 2014):

1. Compare o valor de fitness dos indivíduos da população $f_1(t), f_2(t),..., f_{P-1}(t), f_P(t)$ e encontre o mínimo $f_{\min}(t)$ e o máximo $f_{\max}(t)$ entre eles. Divida o intervalo $[f_{\min}(t), f_{\max}(t)]$ igualmente em *M* partes, e conte o número de indivíduos n_i $(i = 1,...M)$ em cada parte. *M* pode ser configurado usualmente igual a *P* (o tamanho da população).
2. Conte o número de elementos não nulos n_i, e calcule $p_i = \frac{n_i}{P}, i = 1,2,...,M$.
3. Calcule a entropia populacional *E*(*t*) na iteração *t* de acordo com a equação (10).

Normalmente, em computação evolutiva, a entropia é normalizada no intervalo [0, 1] a fim de tornar mais fácil e compreensível o seu comportamento ao longo da evolução do algoritmo. Como pode ser visto pela definição de entropia populacional, quando todos os indivíduos da população tem a mesma fitness, $M = 1$, i.e., a entropia populacional alcança o valor mínimo $E = 0$. Quando os valores de fitness estão distribuídos uniformemente a entropia alcança seu valor máximo $E = 1$.

No algoritmo 2 é listado o pseudo código para o cálculo da entropia populacional.

Algoritmo 2: Pseudo código para cálculo da Entropia Populacional

// A função entropia deve receber como parâmetros o vetor de fitness da iteração corrente *fitness*,
// o tamanho da população *P* e retornar o valor de entropia *E*.

(1) **função** $E = entropia(\ fitness)$

(2) $\max\_E = -\log(1/P)$ // entropia máxima

(3) $norm\_data = \dfrac{fitness - mínimo(\ fitness)}{máximo(\ fitness) - mínimo(\ fitness)}$ // normalização da entropia

// Conte o número de elementos não nulos (item 2), pode ser implementado através de uma função
// ***histograma*** que retorna um vetor com as frequências dos bins.

(4) $cnt_{i=1,\ldots,N} =$ ***histograma***$(norm\_data)$

(5) *n* = ***nãoZeros****(cnt)* // vetor contendo as frequências não nulas

(6) *M* = ***tamanho****(n)*

(7) **para** *i = 1:M*

(8) $p_i = n_i / P$

(9) **fim para**

(10) $E1 = -\sum_{i}^{m} p_i \cdot \log p_i$ // calcula entropia conforme equação (10)

(11) $E = E1/\max\_E$ // saída: entropia normalizada entre 0 e 1

A entropia populacional será introduzida no algoritmo de polinização FPA modificando uma das regras criadas por Yang (2012) já descrita, mas cabe ressaltar: *"Na prática, flores adjacentes ou não tão distantes da vizinhança são mais suscetíveis a polinização em comparação com aquelas mais distantes. Por isto, é usado uma probabilidade de troca (regra 4) ou probabilidade de proximidade p para chavear entre fazer polinização global ou intensificar a polinização local"*.

Assim, a modificação feita será retirando este parâmetro *p* e introduzindo o valor calculado de entropia da população para chavear entre os regimes de busca. O valor definido como limiar é obtido através de um estudo de sensibilidade realizado para todas as funções de teste.

Todo o pseudo código não será repetido para evitar redundância, assim as modificações necessárias serão citadas conforme a linha do pseudo código. As demais linhas não citadas ficam inalteradas. Na linha (1) do pseudo código FPA, deve-se retirar a definição do limiar de probabilidade *p* e definir um *limiar de entropia* $\in [0, 1]$. Na linha (9) o número $rand \sim U(0,1)$ deve ser substituído por $E = entropia(\ fit_{i=1,\ldots,V})$ e assim o teste lógico da linha (11) fica:

se *E > limiar*
    // polinização local
senão
    // polinização global
fim se

Em Krohling (2004) foi proposto uma alteração no algoritmo PSO baseado na distribuição de probabilidade Gaussiana. Esta ideia será usada como segunda modificação no algoritmo de polinização. FPA usa um mecanismo semelhante a outro algoritmo evolutivo chamado Evolução Diferencial (em inglês: *Differential Evolution*, abreviado por DE) para fazer busca local. Resultados experimentais mostram que a habilidade de busca local do algoritmo DE é limitada

(Wang & Zhou, 2014). A fim de melhorar a habilidade da busca local será proposto para o algoritmo FPA uma alteração na forma de obtenção do parâmetro ε. Na versão original o parâmetro ε é obtido através de uma distribuição uniforme $U(0,1)$. Uma implementação numérica que produz números aleatórios positivos com média 0.8 e um desvio padrão 0.6 pode ser obtida usando o valor absoluto da distribuição de probabilidade Gaussiana, i.e., $\delta \sim abs[N(0,1)]$. O maior desvio padrão proporcionado pela distribuição Gaussiana comparado com uma distribuição uniforme, melhora a habilidade do algoritmo evolutivo de escapar de mínimos locais (Krohling, 2004). Em termos de pseudo código a única modificação deve ser feita na linha (15) da versão original. Assim, $\varepsilon \sim U(0,1)$ é substituído por $\delta \sim abs[N(0,1)]$.

4. Resultados de simulação

4.1 Funções de Teste

Neste trabalho foram consideradas funções com um mínimo local, alguns e muitos mínimos locais. As funções são apresentadas na tabela 1 com seus limites no espaço de busca, faixa de inicialização assimétrica, classe (unimodal ou multimodal) e número de dimensões. O número de dimensões foi fixado em 30, exceto para a função Schaffer que é definida apenas para duas variáveis.

Função	Espaço de Busca	Inicialização	Classe	Dimensões
Sphere	[-100, 100]	[50, 100]	Unimodal	30
Rosenbrock	[-50, 50]	[25, 50]	Unimodal	30
Schaffer	[-100, 100]	[50, 100]	Multimodal	2
Ackley	[-32, 32]	[16, 32]	Multimodal	30
Rastrigin	[-5.12, 5.12]	[2.56, 5.12]	Multimodal	30
Griewank	[-600, 600]	[300, 600]	Multimodal	30
Schwefel	[-500, 500]	[-500, 250]	Multimodal	30

Tabela 1 - Lista das funções de teste

A população de plantas foi fixada em 25 e o número de iterações em 2500 como sugerido por Yang (2012). Valores menores que 10^{-4} foram arredondados para 0 (zero). Na tabela 2 é mostrado um resumo dos melhores parâmetros, após testes de sensibilidade, utilizado para o algoritmo FPA-EG.

Na tabela 3 é mostrado uma primeira comparação entre o algoritmo FPA e a versão modificada proposta neste trabalho. Na tabela 3 são mostrados os menores valores alcançados pelo algoritmo nas funções de teste. FPA refere-se a versão original desenvolvida por Yang (2012) com o parâmetro η do voo de Levy fixo em 0.01, probabilidade de troca entre os regimes de busca de 0.8.

Na tabela 4 é mostrado uma comparação estatística dos resultados em termos de média dos menores valores obtidos e de seu desvio padrão, entre parênteses, ao término do número máximo de iterações. Na tabela 5 é mostrado o tempo médio de execução em segundos da versão FPA-EG e do algoritmo original.

Os resultados obtidos também foram comparados com versões padrão do algoritmo PSO (PSO-Gbest e PSO-Lbest) descritas por Mendel, Krohling e Campos (2011). Nas figuras de 1 a 7 são mostrados os box-plots indicando que o FPA-EG aprsesenta desempenho competitivo com as versões do PSO.

Funções	Parâmetros	
	η	Limiar de Entropia
Sphere	0.12	0.85
Schaffer	0.85	0.80
Ackley	0.09	0.72
Rosenbrock	0.08	0.80
Rastrigin	0.01	0.80
Griewank	0.12	0.80
Schwefel	0.15	0.75

Tabela 2 - Parâmetros FPA-EG após testes de sensibilidade

Funções	FPA	FPA-EG
Sphere	42.3	0
Schaffer	0	0
Ackley	1.5	0
Rosenbrock	7480	1.12
Rastrigin	47.5	41.7
Griewank	1.2	0
Schwefel	3310	124

Tabela 3 - Comparação do valor mínimo alcançado entre FPA e FPA-EG

Funções	FPA média (std)	FPA-EG Média (std)
Sphere	$1.71x10^{-2}$ ($1.12x10^{-2}$)	0 (0)
Schaffer	0 (0)	0 (0)
Ackley	0.11 (0.12)	4.82 (5.56)
Rosenbrock	708.33 (709.58)	38.54 (46.54)
Rastrigin	86.41 (15.16)	71.64 (18.52)
Griewank	0.10 (0.05)	0 ($1.64x10^{-3}$)
Schwefel	$5.96x10^{3}$ (888.18)	$1.46x10^{3}$ (662.26)

Tabela 4 - Comparação estatística entre FPA e FPA-EG em termos de média e desvio padrão

Funções	FPA	FPA-EG
Sphere	3.66	4.58
Schaffer	1.20	1.52
Ackley	4.70	6.42
Rosenbrock	4.11	6.24
Rastrigin	4.14	5.29
Griewank	4.86	3.82
Schwefel	3.84	4.84

Tabela 5 - Tempo de execução dos algoritmos em segundos

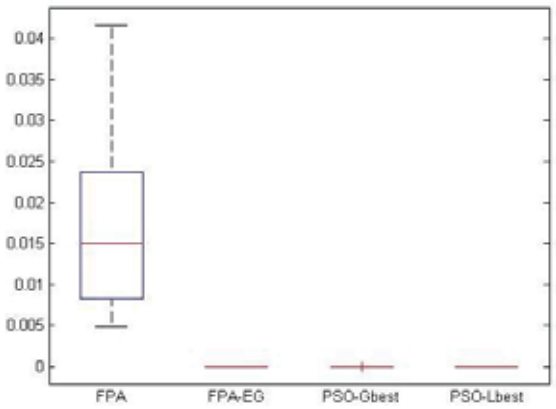

Figura 1 - Função Sphere

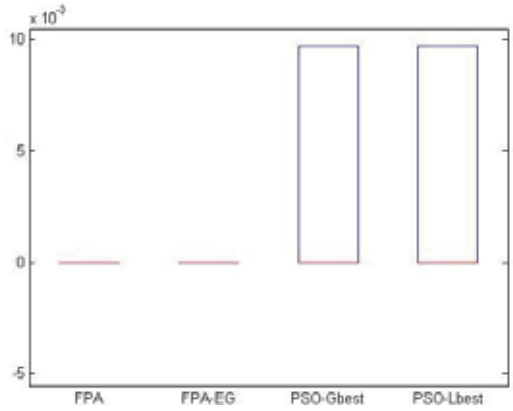

Figura 2 - Função Schaffer

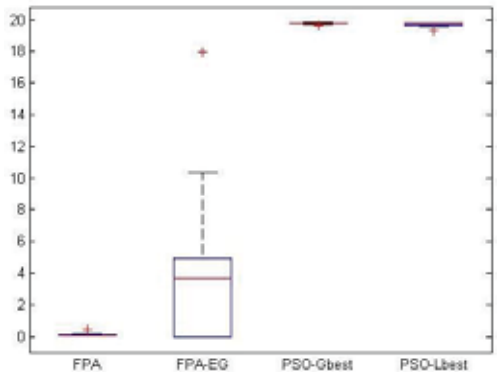

Figura 3 - Função Ackley

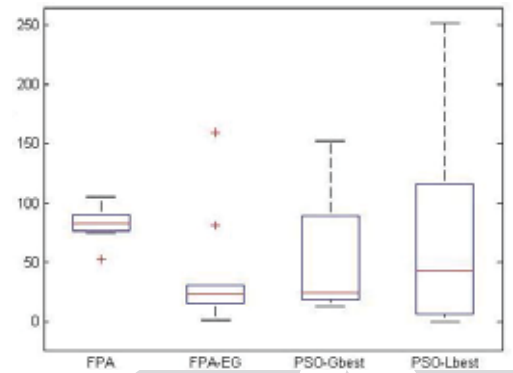

Figura 4 - Função Rosenbrock

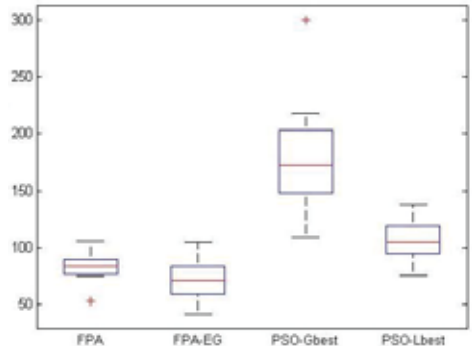

Figura 5 - Função Rastrigin

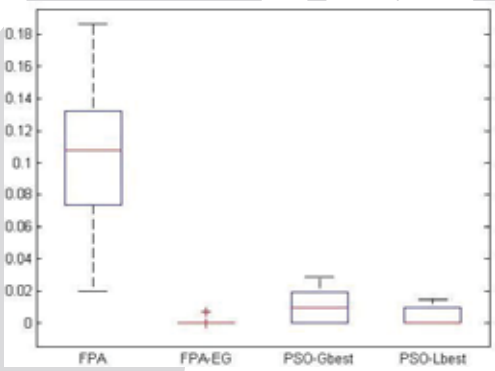

Figura 6 - Função Griewank

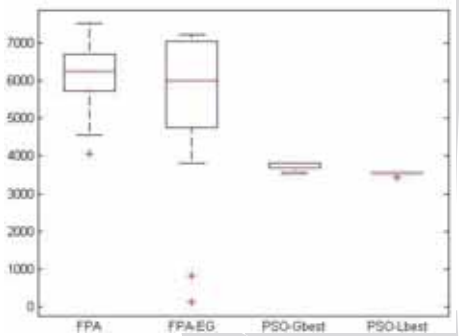

Figura 7 - Função Schwefel

4. Conclusões

Neste trabalho foi desenvolvido uma versão modificada do algoritmo inspirado em polinização de plantas FPA através da introdução de entropia populacional e busca local Gaussiana. As modificações foram testadas para problemas de otimização não lineares e os resultados comparados. De acordo com os resultados obtidos, nota-se que a versão desenvolvida FPA-EG apresenta melhor desempenho do que a versão original FPA. O desempenho do algoritmo FPA-EG deve-se principalmente devido a técnica de busca global via voo de Lévy, que com o parâmetro de escala de seu passo corretamente ajustado proporciona a adequada exploração do espaço de busca. A busca local usando a distribuição de probabilidade Gaussiana pôde acelerar e aumentar a efetividade do algoritmo FPA-EG. A abordagem de diversificação da população de soluções com o uso da entropia populacional para chavear entre os modos de busca local e global é de fundamental importância para os bons resultados obtidos.

Referências Bibliográficas

Binitha, S. e Sathya, S. (2012), A survey of Bio Inspired Optimization Algorithms, International Journal of Soft Computing and Engineering, 2, 137-151.

Glover, B. J. (2007), Understanding Flowers and Flowering: An Integrated Approach, Oxford: Oxford University Press.

Krohling, R. A. (2004), Gaussian Swarm: A Novel Particle Swarm Optimization Algorithm, Proceedings of the 2004 IEEE. Conference on Cybernetics and Intelligence Systems, 372-376.

Liu, S., Mernik, M. e Bryant, B. R. (2009), To explore or to exploit: An entropy-driven approach for evolutionary algorithms, International Journal of Knowledge-based and Intelligent Engineering Systems 13, 185-206.

Mendel, E., Krohling, R. A. e Campos, M. (2011), Swarm Algorithms with Chaotic Jumps Applied to Noisy Optimization Problems, Information Sciences 181, 4494-4514.

Ni, Q. e Deng, J. (2014), Analysis of Population Diversity of Dynamic Probabilistic Particle Swarm Optimization Algorithms, Mathematical Problems in Engineering, 10-18.

Prathiba, R., Moses, M. e Sakthivel, S. (2014), Flower Pollination Algorithm Applied for Different Economic Load Dispatch Problems, International Journal of Engineering and Technology, 6, n. 2, 1009-1016.

Raouf, O. A., Baset, M. A. e El-Henawy, I. (2014a), A New Hybrid Flower Pollination Algorithm for Solving Constrained Global Optimization Problems, International Journal of Applied Operational Research, 4, n. 2, 1-13.

Raouf, O. A., Baset, M. A. e El-Henawy, I. (2014b), A Novel Hybrid Flower Pollination Algorithm with Chaotic Harmony Search for solving Sudoku Puzzles, International Journal of Modern Education and Computer Science, 3, 38-44.

Rosca, J. (1995), Entropy-Driven Adaptive Representation, Workshop on Genetic Programming: From Theory to Real-World Applications, 23-32.

Shannon, C. (1958), A Mathematical Theory of Communication, Bell Systems Technical Journal, 27, 379-423 e 623-656.

Sharawi, M., Emary, E., Saroit, I. e El-Mahdy, H. (2014), Flower Pollination Optimization Algorithm for Wireless Sensor Network Lifetime Global Optimization, International Journal of Soft Computing and Engineering (IJSCE), 4, 54-59.

Tran, T., Nguyen, T. T. e Nguyen, H. L. (2004), Global Optimization using Levy Flight, Proceedings of ICT, 1-12.

Wang, R. e Zhou, Y. (2014), Flower Pollination Algorithm with Dimension-by-Dimension Improvement, Mathematical Problems in Engineering, 1-9.

Yang, X. (2012), Flower Pollination Algorithm for Global Optimization, Unconventional Computation and Natural Computation, Lecture Notes in Computer Science, 7445, 240-249.

Yang, X. (2014), Nature Inspired Optimization Algorithms, Primeira Edição, Londres, Elsevier, 82-85.